面试流程

2018年8月26日 星期日 21:46

面试3个环节

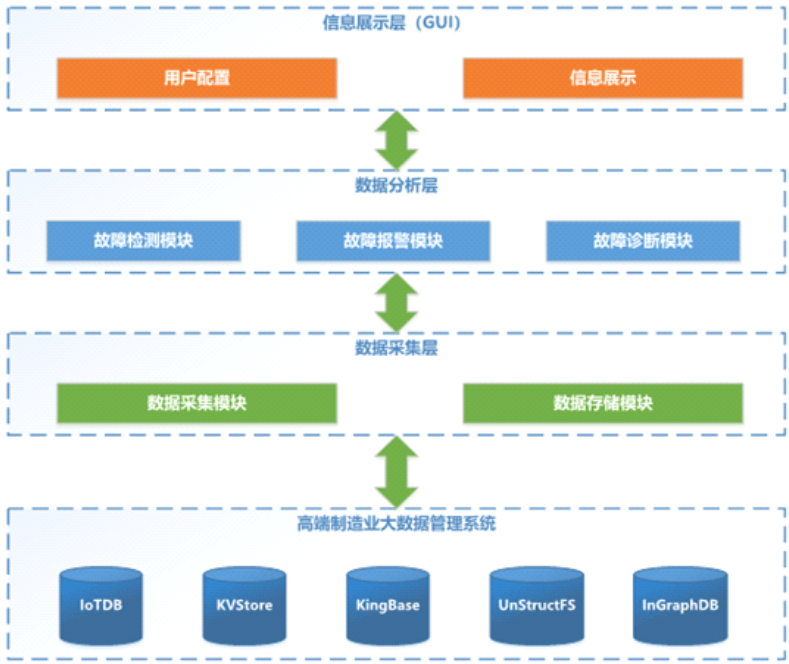
1. 行为面试环节：30秒到1分钟的自我介绍，主要介绍自己的主要学习和工作经历。然后问一些项目相关的问题。
    a. 描述每个项目的一般流程：（要突出自己的工作，而非项目背景）
        i. 简短的项目背景：规模、功能、目标用户等。
        ii. 自己完成的任务：注意参与和负责！
        iii. 为完成任务自己做了哪些工作，怎么做的：系统架构，基于什么工具在哪个平台应用了哪些技术；
        iv. 自己的贡献：具体些，最好用数字加以说明，参与开发，按时完成多少功能；
        v. 在该项目中碰到的最大问题是什么，是怎么解决的？
        vi. 从这个项目中学到了什么？
        vii. 什么时候会和其他团队成员有什么样的冲突，怎么解决冲突的。
    b. 掌握的技能：了解（只是上过课，除非岗位需要才列上去），熟悉（简历中的技能大部分应该是熟悉），精通（真的精通！能回答出大部分的问题）！
2. 技术面试环节

    5种素质：
    a. 扎实的基础知识：编程语言、数据结构、算法。
    b. 能写高质量的代码：边界条件、特殊输入。
    c. 分析问题时思路清晰：不要在没有形成清晰思路之前草率的开始写代码！三种方法解决难题：举例、画图、分解。
        i. 举几个简单的具体例子让我们理解问题；
        ii. 试着用图形表示抽象的数据结构，如二叉树等
        iii. 试着把复杂的问题分解为若干个简单的子问题，再一一解决。
    d. 能优化时间效率和空间效率：提示更好的解法的时候，不能放弃思考，努力寻找时间空间复杂度更优的解。
    e. 学习沟通等各方面能力：最近看什么书、学到了什么技术；知识迁移能力；
3. 应聘者提问环节：为每一轮面试准备2~3个问题，不要问公司发展战略之类的，不要问薪水，不要打听面试结果。
    a. 推荐问一些与应聘职位或者项目相关的问题。提前搜索相关职位的信息；留意面试官说过的话；要提前学习哪些必备的知识，掌握程度要怎样？

百度面试-java开发工程师

2018年9月17日 星期一　　0:52

- 首先介绍了自己在做的比较熟的项目，围绕项目，画架构图说明项目结构。
  - 各部分如何实现的
  - 数据分析层如何做的
  - 数据量有多少，如果要获取半年前的数据能否高效获取。
  - ElasticSearch集群的规模，有几个master，几个node
  - 时序数据库是单点的还是分布式集群的



- 网络知识
  - TCP/IP层次体系结构
  - TCP三次握手和四次挥手：四次挥手的最后一次确认不发送可不可以，如果双方同时结束发送数据，是否仍然需要4次挥手。
    - 三次握手

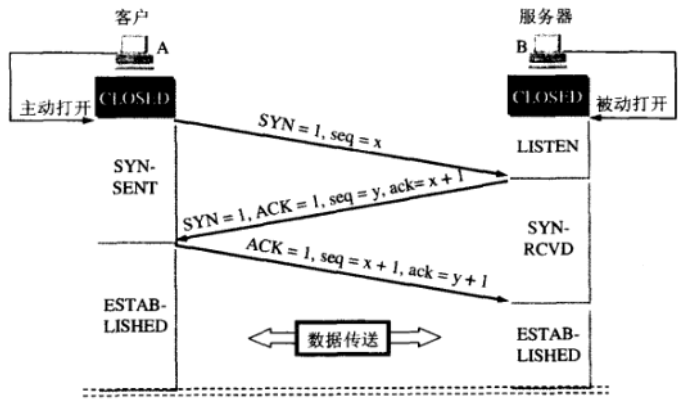


图 5-31　用三次握手建立 TCP 连接

    - 四次挥手

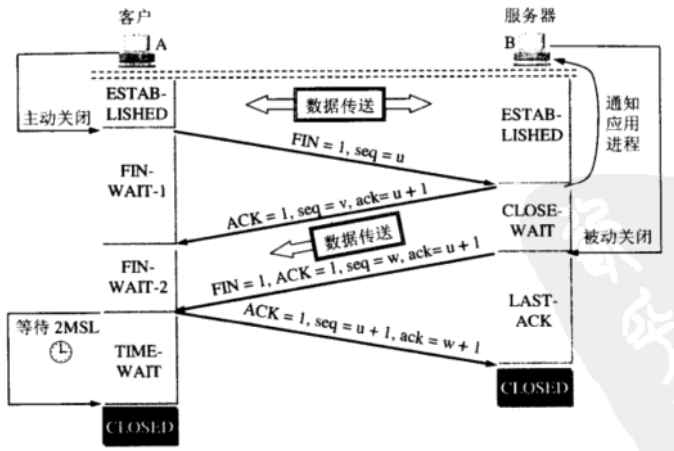


图 5-32　TCP 连接释放的过程

- Java知识
  - 线程同步有哪些方式
    - Synchronized和Lock的区别
    - volatile 有什么作用
      - volatile是轻量级的synchronized，在多处理器开发中保证共享变量的可见性。可见性是指当一个线程修改一个共享变量，另一给线程能够读取到这个修改的值。
      - 与synchronized相比，使用成本比较低，不会引起上下文的切换和调度。
  - ThreadLocal作用
  - 画一下java集合类的继承关系图

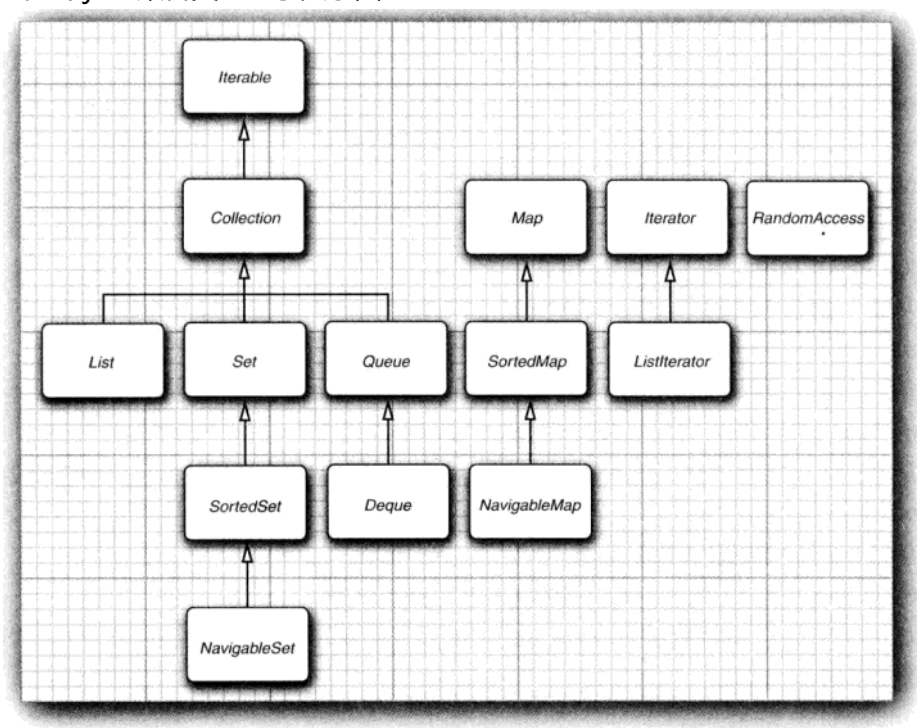


图 9-4　集合框架的接口

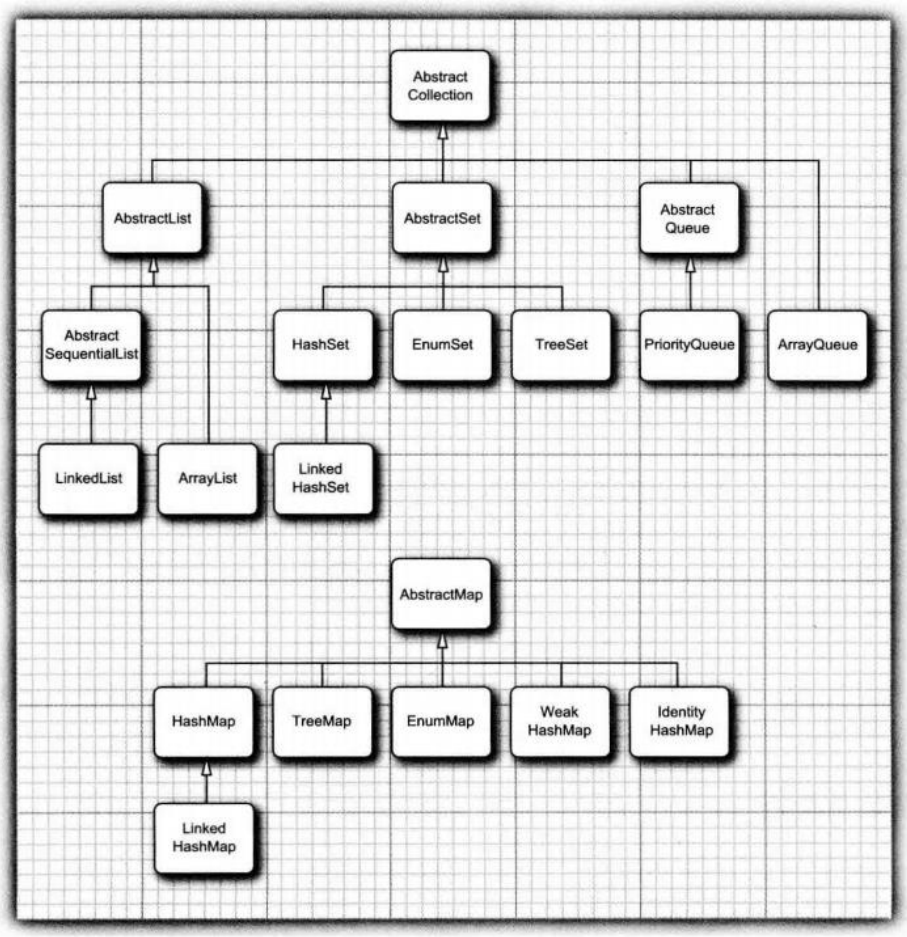
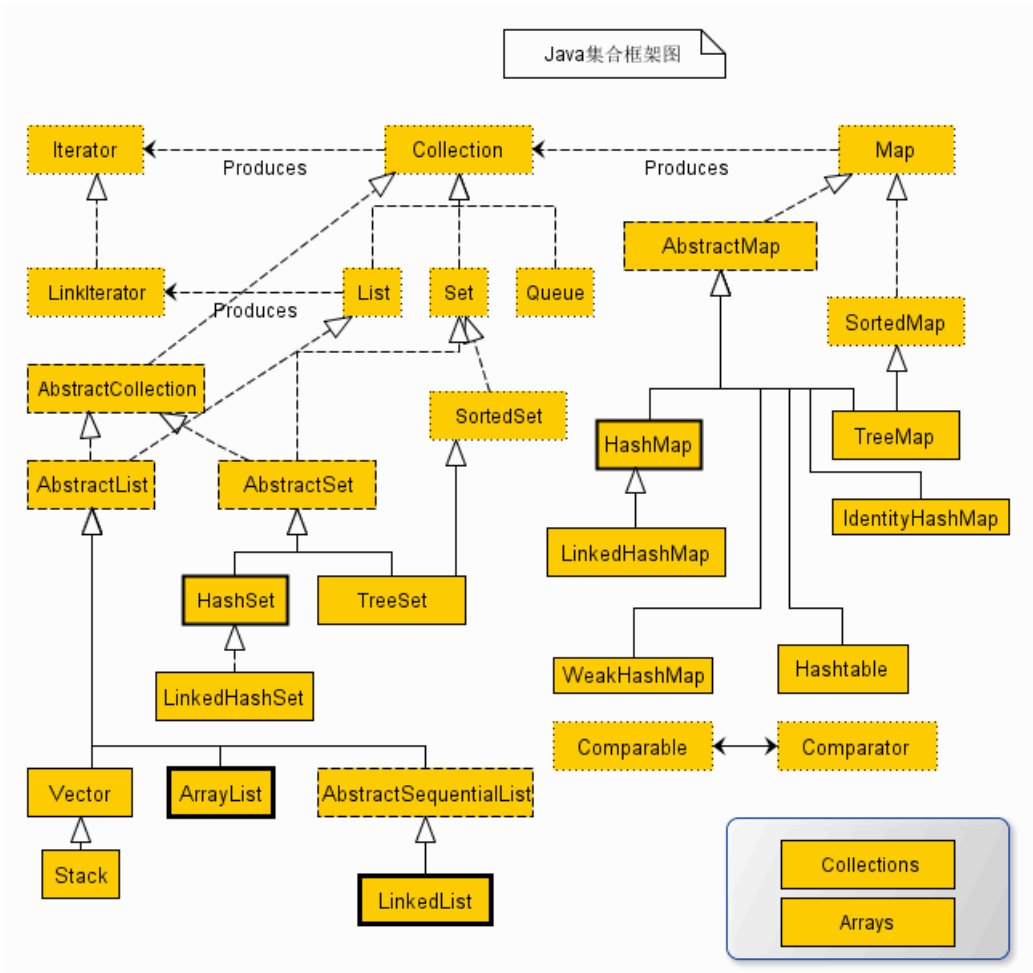


图 9-5　集合框架中的类



- equals和hashcode方法的关系

        - equals 方法用于检测一个对象是否等于另外一个对象。默认是判断两个对象引用是否相同。可以使用Objects类的equals方法。
        - hashcode方法
            - hashcode由对象导出的整数值，无规律。 每个对象的默认hashcode为对象的存储地址。可以合理地组合实例域的散列码，使得不同对象产生的散列码均匀。可以使用Objects对象的hash方法快速求hashcode。
            - 相同字符串String的hashcode是相同的，String重写了hashcode是根据字符串内容导出。
        - 两者的关系
            - equals和hashCode方法的定义必须一致：x.equals(y)为true，则x.hashCode()就必须与y.hashCode()具有相同的值。两个对象相同hashcode必须相同。
            - hashcode主要是用在hash容器中，如hashMap，hashSet等；这些容器首先比较hashcode，hashcode相同时才进行equals比较。
            - 两者的作用类似，都是用来比较两个对象是否相等一致，为何需要hashcode，这是由于equals是比较完整的比较，效率较低，而hashcode进行对比，效率很高；但是equals的比较是完全可靠的，hashcode的比较不是绝对可靠的。equals相等hashcode一定相等，hashcode相等，equals不一定相等。
            - 每当需要对比的时候，首先用hashCode()去对比，如果hashCode()不一样，则表示这两个对象肯定不相等（也就是不必再用equal()去再对比了）,如果hashCode()相同，此时再对比他们的equal()，如果equal()也相同，则表示这两个对象是真的相同了，这样既能大大提高了效率也保证了对比的绝对正确性！
            - 重写了equals，一定要重写hashcode方法。
- Linux知识
    - 显示进程的命令有哪些
        - ps aux | less

          -A：显示所有进程

          a：显示终端中包括其它用户的所有进程

          x：显示无控制终端的进程
        - Top
        - Pstree pstree以树状显示正在运行的进程。
        - pgrep firefox pgrep能查找当前正在运行的进程并列出符合条件的进程ID。
    - 显示端口信息的命令是什么
        - **netstat命令各个参数说明如下**:

          -t : 指明显示TCP端口

          -u : 指明显示UDP端口

          -l : 仅显示监听套接字(所谓套接字就是使应用程序能够读写与收发通讯协议(protocol)与资料的程序)

          -p : 显示进程标识符和程序名称，每一个套接字/端口都属于一个程序。

          -n : 不进行DNS轮询，显示IP(可以加速操作)

          netstat -ntulp | grep 8088 查看所有tcp udp的8088端口
        - Lsof lsof 可以用来查看指定端口所运行的程序 lsof -i :8088
        - ps 是LINUX下最常用的也是非常强大的进程查看命令

          下面对命令选项进行说明：

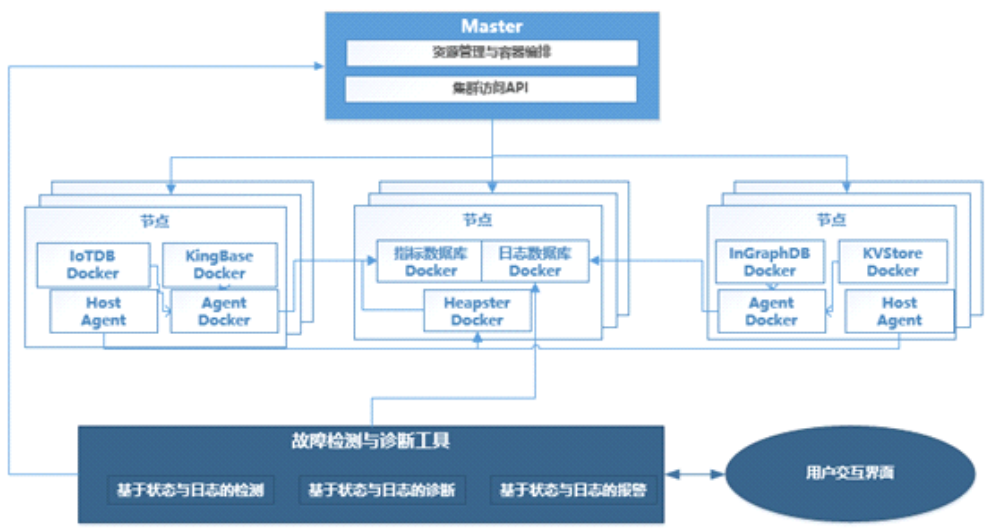
          -e 显示所有进程。

          -f 全格式。

- 算法小题
  - 2个人轮流取100根火柴，一次可以拿1个或2个，问如何使自己拿到第100根。

网易互娱-基础平台架构工程师

2018年9月17日 星期一 1:37

- 自我介绍
- 介绍一下现在做的这个项目
- 数据采集部分是怎么做的
- 

- 收集了哪些指标信息

表 2-1 **CPU相关指标**

cpu/node_reservation	**Node保留的CPU Share**
cpu/node_utilization	Node的CPU使用时间
cpu/request	cpu request，单位为ms
cpu/usage	全部Core的CPU累计使用时间
cpu/usage_rate	全部Core的CPU累计使用率，单位为ms

表 2-2 **Memeory相关指标**

memory/limit	**内存hard limit， 单位字节**
memory/major_page_faults	major page faults数量
memory/major_page_faults_rate	每秒的major page faults数量
memory/node_reservation	Node保留的内存Share
memory/node_utilization	Node的内存使用值
memory/page_faults	page faults数量
memory/page_faults_rate	每秒的page faults数量
memory/request	Memory request，单位为字节
memory/usage	总内存使用量
memory/working_set	总的Working set usage， Working set是指不会被kernel移除的内存

表 2-3 **FileSystem相关指标**

filesystem/usage	**文件系统已用的空间，单位为字节**
filesystem/limit	文件系统总空间限制，单位为字节
filesystem/available	文件系统可用的空间，单位为字节

表 2-4 **NetWork相关指标**

network/rx	**累计接受的网络流量字节数**
network/rx_errors	累计接受的网络流量错误数
network/rx_errors_rate	每秒接受的网络流量错误数
network/rx_rate	每秒接受的网络流量字节数
network/tx	累计发送的网络流量字节数
network/tx_errors	累计发送的网络流量错误数
network/tx_errors_rate	每秒发送的网络流量错误数
network/tx_rate	每秒发送的网络流量字节数

数据库版本	iotdb的运行版本号
数据库配置参数	在监控页面中向用户展示需要的配置参数。即，监控工具获得所有配置项的key，由用户自定义选择显示哪些配置项，把这些配置项的key和value均展现给用户。 方式：开放接口，用于获取所有配置项的key，同时能够根据key获取配置项的value。由于暂时没有动态参数，获取的时候可以不考虑此问题，但是尽量能够支持以后向动态参数变化的扩展性。
系统启动时间	如名称
storage group数量	如名称
timeseries数量	如名称
内存占用信息	已完成
每秒写入点数	如名称
当前连接数	客户端连接数
平均查询延迟	查询响应时间的平均值
GC信息	JDK工具获得
存储空间占用（索引、缓存、TsFile等）	总使用空间大小（total以及不同磁盘上的占用空间大小）
	tsfile文件所占空间（tsfile文件在不同磁盘的占用量，以及总占用量）
	overflow文件所占空间
	index文件所占空间
	日志文件所占空间

表 2-6 **KVStore相关指标**

ActiveTasks	此线程池正在处理的任务数
PendingTasks	在此线程池上排队的任务数
CompletedTasks	完成的任务数
TotalBlockedTasks	由于队列饱和而被阻塞的任务数
CurrentlyBlockedTask	由于重试操作解除阻塞的任务数
MaxPoolSize	此线程池的最大线程数
Timeouts	遇到的读、写超时次数
Failures	遇到读、写事务的失败次数
HitRate	所有时间内的缓存命中率
Exceptions	捕获的内部异常数。 在正常的情况下，这应该是零
Load	此节点管理的磁盘数据的大小（以字节为单位）
TotalHintsInProgress	当前尝试发送的提示数
TotalHints	自启动/重启以来写入此节点的提示消息数。包括每个提示对每个主机条目提示
BufferPool.Size	受管缓冲池的大小（以字节为单位）
BufferPool.Misses	池中的未命中率。 更多的在分配时发生。
Client.connectedNativeClients	连接到此节点本机协议服务器的客户端数
Client.connectedThriftClients	连接到节点thrift协议服务器的客户端数
jvm.memory.Committed	提交给JVM使用的内存量（以字节为单位）
jvm.memory.Init	JVM最初从操作系统请求的内存量（以字节为单位）
jvm.memory.Max	可用于内存管理的最大内存量（以字节为单位）
jvm.memory.Usage	已使用的比率与最大内存
jvm.memory.Used	使用的内存量（以字节为单位）

表 2-7 **InGraphDB相关指标**

check_point.total_time	到目前为止检查点所花费的总时间
ids_in_use.relationship	存储在数据库中的关系的总数
ids_in_use.node	存储在数据库中的节点总数
page_chache.eviction_exceptions	在页面缓存被替换过程期间的异常总数
transaction.peak_concurrent	此机器上发生的并发事务的最高峰值
transaction.active	当前活动事务的数量
transaction.active_read	当前活动的读事务的数量
transaction. active_write	当前活动的写事务的数量
transaction.committed	已提交事务的总数
transaction. committed_read	已提交读事务的总数
transaction.rollbacks	回滚事务的总数

transaction. rollbacks_read	回滚读事务的总数
transaction. rollbacks_write	回滚写事务的总数
transaction.terminated	已终止事务的总数
transaction. terminated_read	已终止读事务的总数
transaction. terminated_write	已终止写事务的总数
transaction.last_committed_tx_id	最后提交的事务的ID
transaction. last_closed_tx_id	上次关闭的事务的ID
cyper.replan_events	Cyper决定重新计划一次查询的总次数
pull_updates	异步事务更新的请求数
pull_update_highest_tx_id_requested	从核心服务器接收的最高事务ID
pull_update_highest_tx_id_received	从核心服务器收到的最后一个事务ID
log_rotation.events	到目前为止执行的事务日志的总数
log_rotation.total_time	到目前为止日志花费的总时间
log_rotation.log_rotation_duration	日志事件的持续时间
network.slave_network_tx_writes	从设备到主设备传输数据的字节数
network.master_network_store_write	从一台机器到另外一台机器拷贝传输的字节数
network.master_network_tx_writes	主机到从机传输的字节数
cluster.slave_pull_updates	此实例执行的更新拉取总数
cluster.slave_pull_update_store_writes	在此实例的最后一次拉更新中拉出的最高事务ID
cluster.is_master	此实例是否是集群中的主节点
cluster.is_available	此实例在集群中是否可用
causal_clustering.core.is_leader	这个服务器是否为领导者
causal_clustering.core.dropped_messages	被丢弃的RAFT消息
causal_clustering.core.queue_sizes	RAFT消息排队数
causal_clustering.core.commit_index	显示已安全提交到其Raft日志中的事务数
causal_clustering.core.leader_not_found	显示找不到Raft协议的领导者的次数
causal_clustering.core.tx_pull_requests_received	显示已接收的事务日志传送请求的数量
causal_clustering.core.is_leader	显示是否扮演Raft领导者角色

- 数据库连接数从哪里获取，mysql的performance表中
- performance_schema的performance_schema.global_status表中
- 项目中遇到过什么困难，如何避免，如何解决
- Java知识
- abstract类与接口的区别
  - 抽象类
    - 抽象方法是一种特殊的方法：它只有声明，而没有具体的实现。
    - 抽象方法必须用abstract关键字进行修饰。如果一个类含有抽象方法，则称这个类为抽象类，抽象类必须在类前用abstract关键字修饰，但是抽象类中可以不含有抽象方法。
    - 父类实现没有意义，不同的子类有不同的实现，这时可以使用抽象类。
    - 抽象类不能创建对象。
    - 抽象类与普通方法一样，可以定义成员变量和普通的成员方法，抽象方法必须是public或者protected的，默认为public；抽象方法不能是static的方法。
    - 子类继承自抽象类，必须实现父类的抽象方法，如果不实现，这个类也必须定义为abstract的类。
  - 接口
    - 设计理念是对行为的抽象，接口一般只用来定义某类行为。
    - 接口中可以有变量和方法，变量隐式转为public static final即常量类型。方法被隐式转换为并且只能是public abstract方法，（接口方法为public是因为要开放给外部使用），即接口中的所有方法都是抽象方法，不能有具体的实现。
    - 接口中的变量和方法一般不加修饰符，这样默认便是public static final 和public abstract，且

只能是这两种，换成其他修饰符编译报错。
    - 极度抽象的概念，比抽象类更抽象的概念。
  - 两者的区别
    - 语法层面的区别
      - 抽象类可以提供成员方法（非抽象）的实现细节，而接口中只能存在public abstract的方法。（抽象类既可以有抽象方法，也可以有非抽象方法，接口只能是抽象方法）。
      - 抽象类中的成员变量和普通类一样，可以是各种类型的(private protected等)，接口只能是public static final类型的。
      - 接口中不能含有静态代码块以及静态方法，抽象类可以有静态代码块和静态方法。
      - 一个类可以实现多个接口，但是只能继承一个抽象类。
    - 设计层面的区别
      - 抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。
      - 设计层面不同，抽象类作为很多子类的父类，它是一种模板式设计。而接口是一种行为规范，它是一种辐射式设计。如果需要添加一个方法，抽象类中定义了，子类自动继承，不需要再修改子类；而接口中添加一个方法，所有的子类都需要进行修改。
  - 什么是反射？有什么作用
    - 能够分析类能力的程序称为反射；
    - 基本定义：反射机制是程序在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性（包括私有和共有，但是需要设置类的accessible）；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。
    - 基本功能（反射机制最重要的功能-检查类的结构）
      - **获取Class对象**：3种方法获取Class对象
      - **实例化一个类的对象**：2中方式实例化类对象
      - **获取父类和接口（单继承多实现）**
      - **获取一个类的全部属性**
      - **获取一个类的全部方法**
      - **调用某个类的方法**
    - 作用
      - 一般由工具开发人员创建，可以在运行时查看对象，例如可以创建一个toString方法供所有的类使用，使用反射可以简化toString方法，不同的类不需要重写toString方法。
      - 结合工厂模式：可以使添加功能类的时候不需要修改工厂类。Spring的IoC的实现原理就是工厂模式加反射机制。
      - 在运行期间判断一个对象所属的类。
      - 动态代理中使用了反射使得动态执行代码过程中，可以组装更强大的功能，如AOP。
    - 缺点：对性能有影响，解释型操作。

- Linux知识
  - 如何删除七天前的日志信息

    find /opt/soft/log/ -mtime +7 -name "*.log" -exec rm -rf {} \;

将/opt/soft/log/目录下所有30天前带".log"的文件删除。

可以将其写在一个可执行的shell脚本中，再设置cron调度执行，系统便会自动清理；

Crontab -e

10 0 * * * /opt/soft/log/auto-del-7-days-ago-log.sh >/dev/null 2>&1

- 通过什么可以获取文件的创建日期
  - 在Linux中，没有文件创建时间的概念。只有文件的访问时间、修改时间、状态改变时间。
  - 使用stat命令 stat demo.txt

    可以显示文件的创建日期、修改日期、访问日期

```
[haohaosong@localhost dir]$
[haohaosong@localhost dir]$ ll
total 0
-rw-rw-r--. 1 haohaosong haohaosong 0 Mar 17 13:16 1.txt
[haohaosong@localhost dir]$
[haohaosong@localhost dir]$
[haohaosong@localhost dir]$ stat 1.txt
  File: `1.txt'
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 802h/2050d      Inode: 399004      Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  500/haohaosong)   Gid: (  500/haohaosong)
Access: 2017-03-17 13:16:18.066984602 +0800
Modify: 2017-03-17 13:16:18.066984602 +0800
Change: 2017-03-17 13:16:18.066984602 +0800
[haohaosong@localhost dir]$
```

- Netstat nltp 参数的作用

  -t : 指明显示TCP端口

  -u : 指明显示UDP端口

  -l : 仅显示监听套接字(所谓套接字就是使应用程序能够读写与收发通讯协议(protocol)与资料的程序)

  -p : 显示进程标识符和程序名称，每一个套接字/端口都属于一个程序。

  -n : 不进行DNS轮询，显示IP(可以加速操作)

- http知识
  - 后台数据发送方式有哪几种，get 和 post，这两者有什么区别。

    get是幂等的，post是非幂等的

	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。 在发送密码或其他敏感信息时绝不要使用 GET！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。

  get把请求的数据放在url上，即HTTP协议头上，其格式为：

以?分割URL和传输数据，参数之间以&相连。
数据如果是英文字母/数字，原样发送，
如果是空格，转换为+，
如果是中文/其他字符，则直接把字符串用BASE64加密，及“%”加上“字符串的16进制ASCII码”。
post把数据放在HTTP的包体内（requrest body）。
get提交的数据最大是2k（原则上url长度无限制，那么get提交的数据也没有限制咯？限制实际上取决于浏览器，(大多数)浏览器通常都会限制url长度在2K个字节，即使(大多数)服务器最多处理64K大小的url。也没有卵用。）。
post理论上没有限制。实际上IIS4中最大量为80KB，IIS5中为100KB。
GET产生一个TCP数据包，浏览器会把http header和data一并发送出去，服务器响应200(返回数据);
POST产生两个TCP数据包，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok(返回数据)。
GET在浏览器回退时是无害的，**POST**会再次提交请求。
GET产生的URL地址可以被Bookmark，而**POST**不可以。
GET请求会被浏览器主动cache，而**POST**不会，除非手动设置。
GET请求只能进行url编码，而**POST**支持多种编码方式。
GET请求参数会被完整保留在浏览器历史记录里，而**POST**中的参数不会被保留。
GET只接受ASCII字符的参数的数据类型，而**POST**没有限制
那么，post那么好为什么还用get？**get**效率高！。

- Restful url的幂等性
  GET，PUT，DELETE都是幂等操作，而POST不是，
  首先GET请求很好理解，对资源做查询多次，此实现的结果都是一样的。
  PUT请求的幂等性可以这样理解，将A修改为B，它第一次请求值变为了B，再进行多次此操作，最终的结果还是B，与一次执行的结果是一样的，所以PUT是幂等操作。
  同理可以理解DELETE操作，第一次将资源删除后，后面多次进行此删除请求，最终结果是一样的，将资源删除掉了。
  POST不是幂等操作，因为一次请求添加一份新资源，二次请求则添加了两份新资源，多次请求会产生不同的结果，因此POST不是幂等操作。

CVTE

2018年9月19日 星期三 23:17

- 常用的集合类
- HashMap原理，扩容为什么是2倍
- ConcurrentHashMap 数据结构是怎样的（存储），怎么实现线程安全的
- 1.8之前hashMap使用链表解决冲突，效率低，冲突多，红黑树
- Lock与Synchronized区别
- Lock底层如何实现，如何加锁的，队列同步器
- 线程有哪些状态，如何转换
- Java的异常体系，RunTimeException
- 手撕代码：递增数组找两个数和为某个值
- MVC设计模式，SpringMVC请求过程
- Yield join wait notify sleep
- 创建线程
- 手撕单例模式
- 二叉树的递归的中序遍历
- 数据库Union与union all的区别
- AOP的切点
- 如何通过URL找servlet
- Synchronized
- 单链表反转
- Socket编程有哪些函数
- JVM的内存结构

CVTE-Web后台开发工程师

2018年9月20日 星期四 11:18

- 抽象类与接口的区别
- HashMap中的key是否有序，需要做什么（重写equals和hashcode方法）
- 写一个数据库SQL，有没有做过SQL优化，如何知道SQL语句的性能（explain 需要如何处理），在哪些键上建立索引
- 什么情况下索引会失效
- 数据库中事务的隔离级别
- Spring中Controller中写一段异常处理的代码，并向上抛出异常
- 有一个方法，逻辑比较复杂，请问你会如何进行优化
- 描述一下线程池的工作原理，当任务数量多余线程数量会发生什么
- 有哪几种设计原则，介绍一下开闭原则
- 对于多线程系统中，如何设计避免线程中的一些问题
- JVM中的GC有哪些算法，新生代使用哪种算法，如何进行的。

美团

2018年9月20日 星期四 21:45

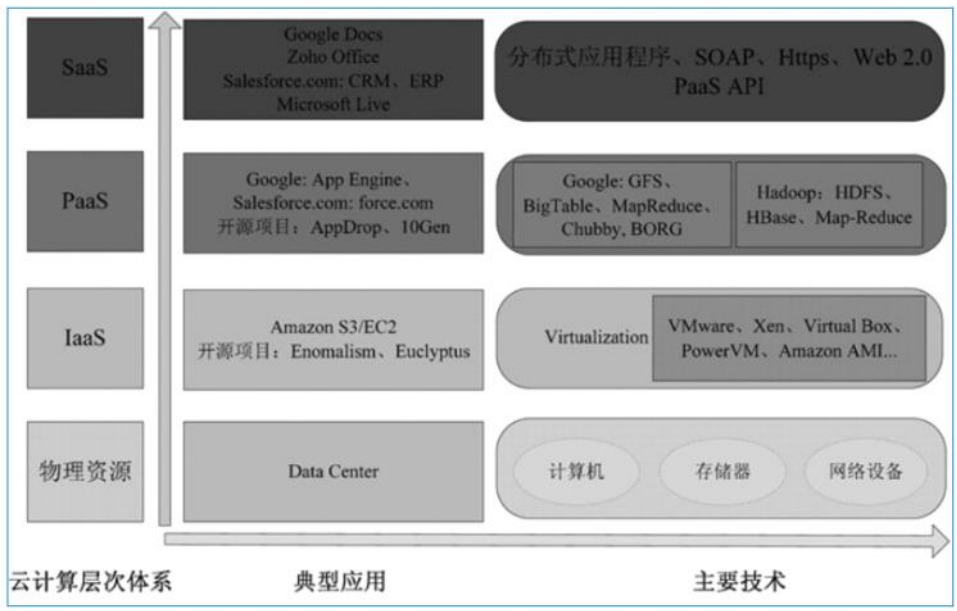
- 手撕代码：最大组合数
- 浏览器输入网址发生的全过程
- 操作系统逻辑地址转换为物理地址
- 进程与线程区别
- 存储引擎
- Jdk1.7与1.6有什么更新

腾讯运营开发面试

2018年9月26日 星期三 23:21

云计算分层概念

其实云计算是分三层的，分别是Infrastructure（基础设施）-as-a-Service，Platform（平台）-as-a-Service，Software（软件）-as-a-Service。基础设施在最下端，平台在中间，软件在顶端。别的一些“软”的层可以在这些层上面添加。



- IaaS：基础设施服务，Infrastructure-as-a-service，IaaS 是云服务的最底层，主要提供一些基础资源 Amazon EC2。
- PaaS：平台服务，Platform-as-a-service，PaaS 提供软件部署平台（runtime），抽象掉了硬件和操作系统细节，可以无缝地扩展（scaling）。
- SaaS：软件服务，Software-as-a-service，将应用作为服务提供给客户。
- PaaS的实质是将互联网的资源服务化为可编程接口，为第三方开发者提供有商业价值的资源和服务平台。简而言之，IaaS就是卖硬件及计算资源，PaaS就是卖开发、运行环境，SaaS就是卖软件。
- 目前很多的容器云平台通过Docker及Kubernetes等技术提供应用运行平台，从而实现运维自动化，快速部署应用、弹性伸缩和动态调整应用环境资源，提高研发运营效率。基于Docker及Kubernetes技术构建容器云（PaaS）平台。

云平台建设

- 运营架构、数据技术应用、推广拓展等不同平台的开发工作；
- 建设日理万机的管控系统，打造cool且强大的运营架构平台，大数据云计算；

有没有参加过实习

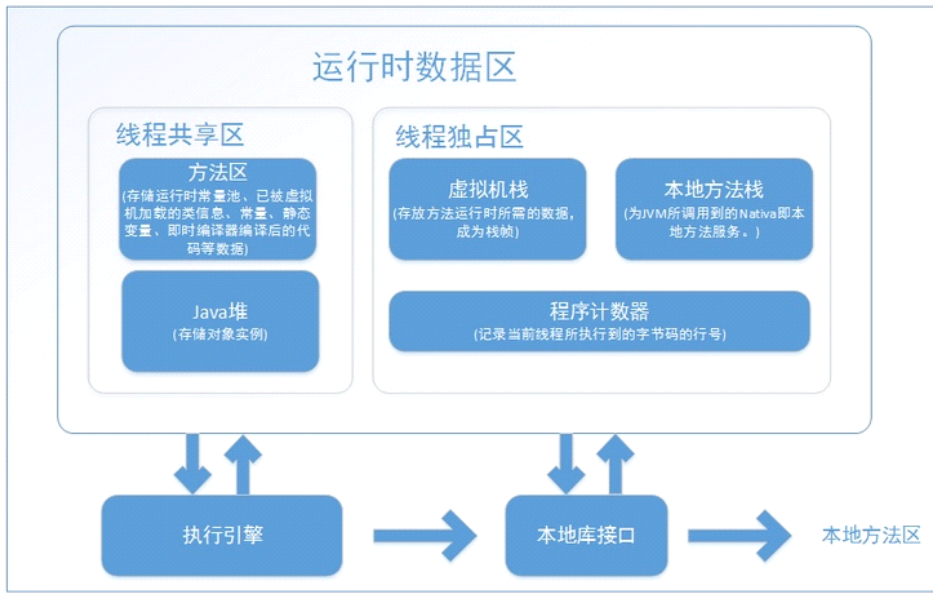
- 没有，老师这边不允许实习
- 课题组这边，我是小组长，每年年中都会进行项目中期检查，我负责工具的设计与开发，无法参与实习；

对linux熟悉吗

- 熟悉，对于常用的各种操作比较熟悉（grep、awk、sed、ps、find）；掌握各种deb、rpm软件包的安装方式；熟悉linux文件系统和目录结构，每个目录的作用；掌握vim，gcc，gdb等编辑、编译、调试器。

java相关

- jvm结构分区以及调优

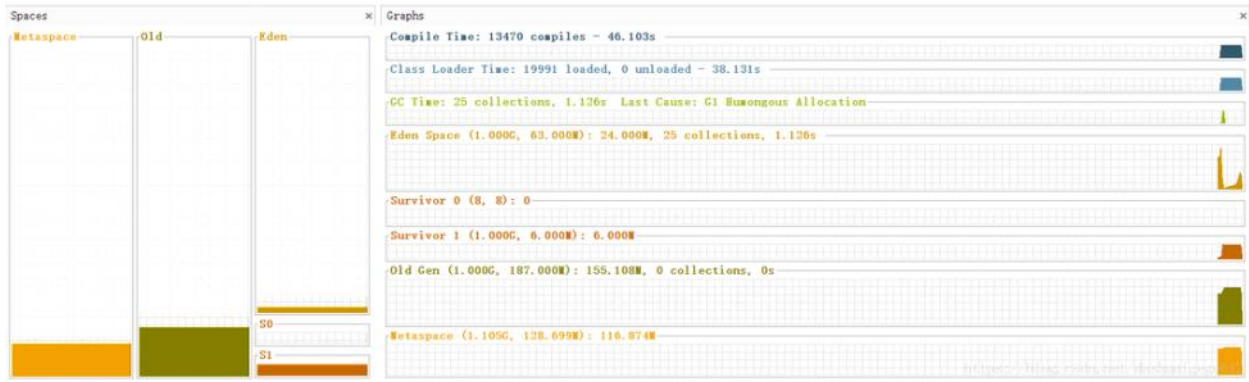


  - 对eclipse进行过性能调优，根据运行插件的时间进行统计
    - 编译时间和类加载时间应该是优化的重头戏了
    - 由于类加载需要进行字节码验证耗时，考虑到eclipse使用者众多，它的编译代码我们认为是可靠的，不需要加载的时候再进行字节码验证，因此通过参数-Xverify:none禁止掉字节码验证过程也可以认为是优化手段。
    - 可以看出每次的Full GC 都是有Metadata GC Threshold造成的，也就是元空间引发的full GC。 从JDK8开始，永久代(PermGen)的概念被废弃掉了，取而代之的是一个称为Metaspace的存储空间。Metaspace使用的是本地内存，而不是堆内存，也就是说在默认情况下Metaspace的大小只与本地内存大小有关。每次Full GC，Metaspace都在调整阈值。因此，再加入-XX:MetaspaceSize=256M
    - 调大最小堆 最大堆，年轻代**-Xms2048m
      -Xmx2048m
      -Xmn512m**

下面是初始的Eclipse配置文件eclipse.ini

```
-startup
plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.551.v20171108-1834
-product
org.eclipse.epp.package.jee.product
-showsplash
org.eclipse.epp.package.common
--launcher.defaultAction
openFile
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vmargs
-Dosgi.requiredJavaVersion=1.8
-Dosgi.instance.area.default=@user.home/eclipse-workspace
-XX:+UseG1GC
-XX:+UseStringDeduplication
--add-modules=ALL-SYSTEM
-Dosgi.requiredJavaVersion=1.8
-Xms256m
-Xmx1024m
--add-modules=ALL-SYSTEM
-Dcom.sun.management.jmxremote
```

初始设置指定了1.8版本的JDK，采用G1收集器，设置最大堆为1024M以及开启了JMX管理。

- 

- 线程池的各个参数以及值，用了什么队列（LinkedBlockQueue、synchronousQueue），拒绝策略（丢弃）
  1、AbortPolicy：直接抛出异常
  2、CallerRunsPolicy：只用调用所在的线程运行任务
  3、DiscardOldestPolicy：丢弃队列里最近的一个任务，并执行当前任务。
  4、DiscardPolicy：不处理，丢弃掉。

线程池的优点

1、线程是稀缺资源，使用线程池可以减少创建和销毁线程的次数，每个工作线程都可以重复使用。

2、可以根据系统的承受能力，调整线程池中工作线程的数量，防止因为消耗过多内存导致服务器崩溃。

线程池的创建

- ```
  public ThreadPoolExecutor(int corePoolSize,
                            int maximumPoolSize,
                            long keepAliveTime,
                            TimeUnit unit,
                            BlockingQueue<Runnable> workQueue,
                            RejectedExecutionHandler handler)
  ```

corePoolSize：线程池核心线程数量

maximumPoolSize:线程池最大线程数量

keepAliverTime：当活跃线程数大于核心线程数时，空闲的多余线程最大存活时间

unit：存活时间的单位

workQueue：存放任务的队列

handler：超出线程范围和队列容量的任务的处理程序
  ```

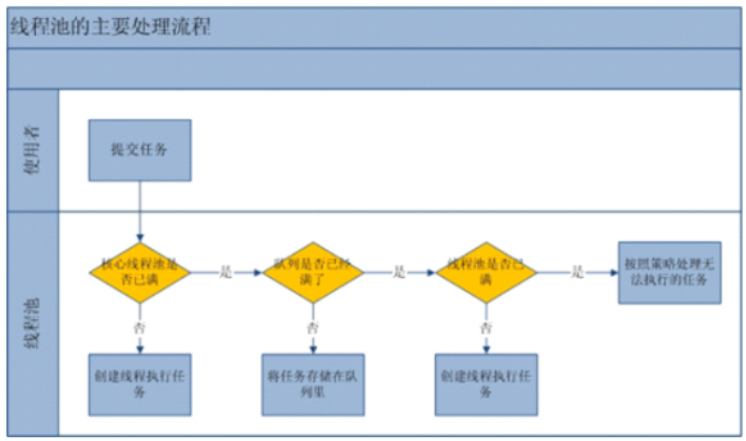
线程池的实现原理

提交一个任务到线程池中，线程池的处理流程如下：

1、判断**线程池里的核心线程**是否都在执行任务，如果不是（核心线程空闲或者还有核心线程没有被创建）则创建一个新的工作线程来执行任务。如果核心线程都在执行任务，则进入下个流程。

2、线程池判断工作队列是否已满，如果工作队列没有满，则将新提交的任务存储在这个工作队列里。如果工作队列满了，则进入下个流程。

3、判断**线程池里的线程**是否都处于工作状态，如果没有，则创建一个新的工作线程来执行任务。如果已经满了，则交给饱和策略来处理这个任务。

- 

- synchronized与Lock的区别

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，不然容易造成线程死锁
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，具体下面会说道，大致就是可以尝试获得锁，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

ElasticSearch

- ```
  http.cors.enabled: true
  http.cors.allow-origin: "*"

  cluster.name: demo
  node.name: master
  node.master: true

  network.host: 127.0.0.1
  transport.tcp.port: 9300
  ```
  ```

```
http.cors.enabled: true
http.cors.allow-origin: "*"

cluster.name: demo
node.name: slave1

network.host: 127.0.0.1
#transport.tcp.port: 9300
http.port: 8200
discovery.zen.ping.unicast.hosts: ["127.0.0.1"]
```

数据库问题汇总

2018年9月27日 星期四 2:31

什么是数据完整性规则?

答：数据完整性是指数据库中存储的数据是有意义的正确的。关系数据模型中数据完整性规则是指对二维表的定义和操作过程遵循某些约束条件。

数据完整性主要包括：1）实体完整性，指每张数据表都必须有主键，而且表中不允许存在无主键的记录和主键值相同的记录。2）参照完整性，指一张数据表中某列的取值受另一张数据表中某列取值范围的约束，表述了多张表之间的关联关系。3）用户定义完整性，指针对某一具体应用而定义的数据库约束条件，比如限制属性的取值类型及范围。

索引的作用及其优缺点?

答：索引就像书的目录一样，通过目录，我们可以很快找到想要读取的内容。索引本质上是数据结构，是表中一个列或多个列的值进行排序的结构。索引的数据结构，常用B-Tree（Balance Tree 平衡树）。

优点：加快查询速度

缺点：一是增加了数据库的存储空间，二是在插入、删除和修改数据时要花费较多的时间(因为索引也要随之变动)。

什么样的字段适合建索引

唯一、不为空、经常被查询的字段

触发器的作用?

触发器是一中特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

什么是存储过程？用什么来调用?

存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL语句执行要快。 调用： 1）可以用一个命令对象来调用存储过程。 2）可以供外部程序调用，比如：java程序。

存储过程的优缺点?

优点： 1）存储过程是预编译过的，执行效率高。 2）存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯。 3）安全性高，执行存储过程需要有一定权限的用户。 4）存储过程可以重复使用，可减少数据库开发人员的工作量。 缺点：移植性差

存储过程与函数的区别

存储过程	函数
用于在数据库中完成特定的操作或者任务（如插入、删除等）	用于特定的数据（如选择）
程序头部声明用procedure	程序头部声明用function
程序头部声明时不需描述返回类型	程序头部声明时要描述返回类型，而且PL/SQL块中至少要包括一个有效的return语句
可以使用in/out/in out 三种模式的参数	可以使用in/out/in out 三种模式的参数
可作为一个独立的PL/SQL语句来执行	不能独立执行，必须作为表达式的一部分调用
可以通过out/in out 返回零个或多个值	通过return语句返回一个值，且改值要与声明部分一致，也可以是通过out类型的参数带出的变量
SQL语句(DML 或SELECT)中不可调用存储过程	SQL语句(DML 或SELECT)中可以调用函数

什么是事务？什么是锁?

事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过ACID测试，即原子性，一致性，隔离性和持久性。 锁：在所以的DBMS中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

什么叫视图？游标是什么?

视图：是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多个表的行或列的子集。对视图的修改会影响基本表。它使得我们获取数据更容易，相比多表查询。 游标：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

在数据库中查询语句速度很慢，如何优化?

1.建索引 2.减少表之间的关联 3.优化sql，尽量让sql很快定位数据，不要让sql做全表查询，应该走索引,把数据 量大的表排在前面 4.简化查询字段，没用的字段不要，已经对返回结果的控制，尽量返回少量数据 5.尽量用PreparedStatement来查询，不要用Statement

数据库三范式是什么?

第一范式：列不可再分 第二范式：行可以唯一区分，主键约束 第三范式：表的非主属性不能依赖与其他表的非主属性 外键约束 且三大范式是一级一级依赖的，第二范式建立在第一范式上，

第三范式建立第一第二范式上

union和union all有什么不同?

UNION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。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表UNION。 UNION ALL只是简单的将两个结果合并后就返回。这样，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。 从效率上说，UNION ALL 要比UNION快很多，所以，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用UNION ALL。

Varchar2和varchar有什么区别?

Char的长度是固定的，而varchar2的长度是可以变化的，比如，存储字符串“abc”对于char(20)，表示你存储的字符将占20个字节，包含17个空，而同样的varchar2（20）只占了3个字节，20只是最大值，当你存储的字符小于20时，按实际长度存储。 char的效率要被varchar2的效率高。 目前varchar是varchar2的同义词，工业标准的varchar类型可以存储空字符串，但是oracle不能这样做，尽管它保留以后这样做的权利。Oracle自己开发了一个数据类型varchar2,这个类型不是一个标准的varchar，他将在数据库中varchar列可以存储空字符串的特性改为存储null值，如果你想有向后兼容的能力，oracle建议使用varchar2而不是varchar

事务的特性有哪些?

事务隔离级别包括： 原子性，即不可分割性，事务要么全部被执行，要么就全部不被执行； 一致性或可串性，事务的执行使得数据库从一种正确状态转换成另一种正确状态； 隔离性，在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事务； 持久性,事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

三、MySQL事务隔离级别

事务隔离级别	脏读	不可重复读	幻读
读未提交（read-uncommitted）	是	是	是
不可重复读（read-committed）	否	是	是
可重复读（repeatable-read）	否	否	是
串行化（serializable）	否	否	否

Read uncommitted

读未提交，顾名思义，就是一个事务可以读取另一个未提交事务的数据。

Read committed

读提交，顾名思义，就是一个事务要等另一个事务提交后才能读取数据。

Repeatable read

重复读，就是在开始读取数据（事务开启）时，不再允许修改操作

Serializable 序列化

Serializable 是最高的事务隔离级别，在该级别下，事务串行化顺序执行，可以避免脏读、不可重复读与幻读。但是这种事务隔离级别效率低下，比较耗数据库性能，一般不使用。

浦发体检邮寄流程

2018年10月18日 星期四 14:13

- 三甲医院 体检截止到2018年10月24日
- 邮寄三份材料
  - 体检发票，个人姓名抬头，机打
  - 体检报告
  - 银行卡号，开户行，姓名，复印件
- 邮寄地址：上海市黄浦区中山东1路12号315办公室 胡老师 021-61618027