1. **找出总成绩最高的那个人和所有课程的成绩**
2. **找出每门成绩都高于80的人**
3. **left join和inner join的区别**

left join 是做左外关联，主表内容都会显示；符合关联条件的附表内容才会显示出来。inner join 是内关联，没有主表附表的概念；两个表中，同时符合关联条件的数据才会显示出来。

sql尽量使用数据量小的表做主表，这样效率高，但是有时候因为逻辑要求，要使用数据量大的表做主表，此时使用left join 就会比较慢，即使关联条件有索引。在这种情况下就要考虑是不是能使用inner join了。因为inner join在执行的时候回自动选择最小的表做基础表，效率高。

概念：根据两个表或多个表的列之间的关系，从这些表中查询数据。分为三种：内连接、外连接、交叉连接

目的：实现多个表查询操作。

内连接(INNER JOIN)：仅将两个表中满足连接条件的行组合起来作为结果集。在内连接中，只有在两个表中匹配的行才能在结果集中出现。

等值连接 自然连接 不等连接

外连接(OUTER JOIN)：在内连接的基础上，还包含表中所有不符合条件的数据行，并将相对应的表列填写NULL

交叉连接(CROSS JOIN)：没有WHERE 子句，它返回连接表中所有数据行的笛卡尔积

1. **数据库里面innodb和另一个的区别**

基本的差别为：MyISAM类型不支持事务处理等高级处理，而InnoDB类型支持。MyISAM类型的表强调的是读写的性能，其执行数度比InnoDB类型更快，但是不提供事务支持，而InnoDB提供事务支持以及外部键等高级数据库功能。

事务：InnoDB 是事务型的，可以使用 Commit 和 Rollback 语句。

并发：MyISAM 只支持表级锁，而 InnoDB 还支持行级锁。

外键：InnoDB 支持外键。

备份：InnoDB 支持在线热备份。

崩溃恢复：MyISAM 崩溃后发生损坏的概率比 InnoDB 高很多，而且恢复的速度也更慢。

其它特性：MyISAM 支持压缩表和空间数据索引。

应用场景：MyISAM管理非事务表。它提供高速存储和检索，以及全文搜索能力。如果应用中需要执行大量的SELECT查询，那么MyISAM是更好的选择。

InnoDB用于事务处理应用程序，具有众多特性，包括ACID事务支持。如果应用中需要执行大量的INSERT或UPDATE操作，则应该使用InnoDB，这样可以提高多用户并发操作的性能。

**5.数据库的三大范式**

范式理论是为了解决以上提到四种异常。高级别范式的依赖于低级别的范式，1NF是最低级别的范式。

1. 第一范式 (1NF) 属性不可分；

2. 第二范式 (2NF) 每个非主属性完全函数依赖于键码，可以通过分解来满足。

3. 第三范式 (3NF) 非主属性不传递函数依赖于键码。

第一范式（1NF）无重复的列

第一范式（1NF）中数据库表的每一列都是不可分割的基本数据项

同一列中不能有多个值

即实体中的某个属性不能有多个值或者不能有重复的属性。

简而言之，第一范式就是无重复的列。

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

第二范式（2NF）属性完全依赖于主键[消除部分子函数依赖]

满足第二范式（2NF）必须先满足第一范式（1NF）。

第二范式（2NF）要求数据库表中的每个实例或行必须可以被惟一地区分。

为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是属性完全依赖于主键。

第三范式（3NF）属性不依赖于其它非主属性[消除传递依赖]

满足第三范式（3NF）必须先满足第二范式（2NF）。

简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性。

**6.数据库的事务是什么**

数据库事务（简称：事务）是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。

为数据库操作序列提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。

当事务被提交给了DBMS（数据库管理系统），则DBMS（数据库管理系统）需要确保该事务中的所有操作都成功完成且其结果被永久保存在数据库中，如果事务中有的操作没有成功完成，则事务中的所有操作都需要被回滚，回到事务执行前的状态;同时，该事务对数据库或者其他事务的执行无影响，所有的事务都好像在独立的运行。  
**7.数据库索引的类型**

Mysql各种索引区别：

普通索引：最基本的索引，没有任何限制

唯一索引：与"普通索引"类似，不同的就是：索引列的值必须唯一，但允许有空值。

主键索引：它 是一种特殊的唯一索引，不允许有空值。

全文索引：仅可用于 MyISAM 表，针对较大的数据，生成全文索引很耗时好空间。

组合索引：为了更多的提高mysql效率可建立组合索引，遵循”最左前缀“原则。

最左前缀原则：顾名思义，最左优先，比如，我们建立了一个以（a,b,c）为组合的索引，那么将会得到：a, ab,abc三种索引。若我们按列“b”进行查找，或者按列（bc）查找，都不会使用到索引，只有以上三种索引可以使用。

哪些情况索引会失效：

如果条件中有or，即使其中有条件带索引也不会使用(这也是为什么尽量少用or的原因，要想使用or，又想让索引生效，只能将or条件中的每个列都加上索引）

对于多列索引，不是使用的第一部分，则不会使用索引

like查询是以%开头

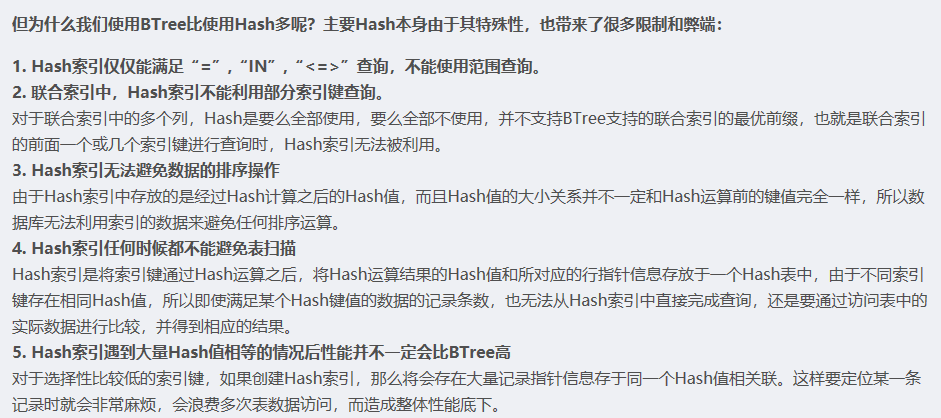
如果列类型是字符串，那一定要在条件中将数据使用引号引用起来,否则不使用索引

如果mysql估计使用全表扫描要比使用索引快,则不使用索引

索引有B树索引和Hash索引

BTree索引是最常用的mysql数据库索引算法，因为它不仅可以被用在=,>,>=,<,<=和between这些比较操作符上，而且还可以用于like操作符，只要它的查询条件是一个不以通配符开头的常量；如果一通配符开头，或者没有使用常量，则不会使用索引。

Hash索引只能用于对等比较，例如=,<=>（相当于=）操作符。由于是一次定位数据，不像BTree索引需要从根节点到枝节点，最后才能访问到页节点这样多次IO访问，所以检索效率远高于BTree索引。





**聚集索引和非聚集索引区别**

聚集索引是InnoDB中的表是索引组织表，即表中的数据按主键顺序存放。聚集索引即按照每张表的主键构造一棵B+树，同时叶子节点中存放的即为整张表的行记录数据。也将聚集索引的叶子节点称为数据页，每个数据页都通过一个双向链表来进行链接。对于聚集索引来说，索引即数据，数据即索引。因为一张表中只能有一个主键（但主键可由多列组成），因此每张表只能拥有一个聚集索引。

辅助索引，也称非聚集索引，叶子节点中并不包括记录的全部数据，只包含一个指示与索引相对应的行数据位置的书签。因为InnoDB使用索引组织表，所以这个书签就是相应的行数据的聚集索引键。

两者的根本区别是表记录的排列顺序和与索引的排列顺序是否一致。

聚集索引一个表只能有一个，而非聚集索引一个表可以存在多个。

聚集索引存储记录是物理上连续存在，而非聚集索引是逻辑上的连续，物理存储并不连续。

聚集索引查询数据速度快，插入数据速度慢；非聚集索引反之。

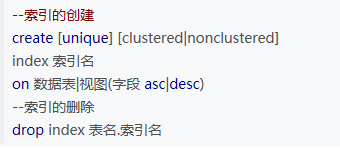
聚集索引：逻辑有序，物理也有序，一个表只能有1个；

  非聚集索引：逻辑有序，物理无序，一个表只能有249个；

**8.数据库有必要建索引吗？**

一个没有索引的数据库表就相当于一本没有索引的新华字典，当你想找出其中一个汉字的时候，无异于寻找MH370碎片啊。我们为指定的字段创建索引之后，当我们根据条件查找数据的时候，数据库引擎就可以利用查找算法（二分查找法）很高效的查出来。

索引用于快速找出在某个列中有一特定值的行，不使用索引，MySQL必须从第一条记录开始读完整个表，直到找出相关的行，表越大，查询数据所花费的时间就越多，如果表中查询的列有一个索引，MySQL能够快速到达一个位置去搜索数据文件，而不必查看所有数据，那么将会节省很大一部分时间。



**9.你觉得mysql有什么缺点？**

关系型数据库的优缺点

优点：保持数据的一致性（数据库的事务）！

由于以标准化为前提，数据更新的开销很小（相同的字段基本上只有一处）

可以进行JOIN等复杂的操作

存在很多实际成果和专业技术信息（成熟的技术）

缺点：大量数据的写入处理

为有数据更新的表做索引或变更表结构

字段不固定时使用

对于简单查询需要快速返回结果的业务处理

适合处理一般量级的数据（银行转账和钱）

非关系数据库的优缺点（redis和MangDB）

为了处理海量数据，非关系数据库设计之初就是为了替代关系型数据库的关系

优点：1.海量数据的增删改查是可以的

2.海量数据的维护和处理非常轻松

缺点：1.数据和数据没有关系，他们之间就是单独存在的

2.非关系数据库没有关系，没有强大的事务关系，没有保证数据的完整性和安全性

适合处理海量数据，保证效率，不一定安全（统计数据，例如微博数据）

**10.除了mysql有用过其他数据库吗？知道数据仓库吗？**

Redis是一款基于内存的且支持持久化、高性能的Key-Value NoSQL 数据库，其支持丰富数据类型(string，list，set，sorted set，hash)，常被用作缓存的解决方案。Redis具有以下显著特点：速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)；

支持丰富数据类型，支持string，list，set，sorted set，hash；

支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行；

丰富的特性：可用于缓存消息，按key设置过期时间，过期后将会自动删除。



**11.隔离和锁，多线程**

事务的特性：

原子性：原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

一致性：一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

隔离性：隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

永久性：持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

隔离的四个级别：一致性从低到高

**读未提交：**一致性最差，**脏读**是指在一个事务处理过程里读取了另一个未提交的事务中的数据。

**读提交**：只能避免脏读，**不可重复读**是指在对于数据库中的某个数据，一个事务范围内多次查询却返回了不同的数据值，这是由于在查询间隔，被另一个事务修改并提交了。快照读完成高并发查询。

不可重复读和脏读的区别是，脏读是某一事务读取了另一个事务未提交的脏数据，而不可重复读则是读取了前一事务提交的数据。

**可重读：**事务启动后，不允许修改操作，可insert和delete,使用快照读，锁住被读取的记录。这是MySQL的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行。不过理论上，这会导致另一个棘手的问题：幻读 （Phantom Read）。简单的说，幻读指当用户读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会发现有新的“幻影” 行。InnoDB和Falcon存储引擎通过多版本并发控制（MVCC，Multiversion Concurrency Control）机制解决了该问题

**可串行化：**这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之，它是在每个读的数据行上加上共享锁。在这个级别，可能导致大量的超时现象和锁竞争。执行效率差，性能开销大。

**锁**：

死锁的解除策略：

进行等待，直到超时，通过参数innodb\_lock\_wait\_timeout控制超时时间

发起死锁检测，主动回滚死锁链条中的某一个事务，让其他事务得以继续进行。通过参数innodb\_deadlock\_detect进行控制

锁机制

1.记录锁

InnoDB实现了如下两种标准记录锁（行级锁）：

共享锁（S Lock）：允许事务读一行数据。当事务拿到某一行记录的共享锁，才可以读取这一行，多个事务可以拿到一把共享锁（读与读可以并行）

排它锁（X Lock）：允许事务删除或更新一行数据。当事务拿到某一行记录的排它锁，才可以修改和删除行，只有一个事务可以拿到排它锁，（写写/读写必须互斥）

行级锁的S锁和X锁的兼容性如下：（需要注意的是，这里的S锁和X锁都是行级锁，兼容性是指对同一记录锁的兼容情况）

S锁和X锁也可加到表上，详见第三小节“多粒度锁”。

2.范围锁

InnoDB有如下两种范围锁算法：

Gap Lock：间隙锁，锁定一个范围，但不包含记录本身。封锁索引记录中的间隔，或者第一条索引记录之前的范围，又或者最后一条索引记录之后的范围。

Next-Key Lock：Gap Lock + Record Lock，锁定一个范围，并且锁定记录本身。

范围锁也是行级锁的一种，不过锁定的是多行记录。

在 repeatable read 和 seraliable 隔离级别下，会使用Next-Key Lock来做范围锁定。从而避免幻读。

需要注意的是，如果定值查询的索引是唯一索引，InnoDB会对Next-Key Lock进行优化，将其降级为Record Lock，即仅锁住索引本身，而不是范围：由range降级为point。但如果是范围查询且是一致性锁定查询，则即使是唯一索引，也会使用Next-Key Lock进行锁定。例如：select \* from t where id > 2 for update;。

3.多粒度锁

InnoDB支持多粒度锁定，这种锁定允许事务在行级上的锁和更高级（表、段、区、页）上的锁同时存在。

为了支持在不同粒度上进行加锁，InnoDB支持一种额外的加锁方式，称之为意向锁。

InnoDB支持两种意向锁：

意向共享锁（IS Lock）：事务想要获得一张表中某几行的共享锁。

意向排它锁（IX Lock）：事务想要获得一张表中某几行的排它锁。

若将上锁的对象看成一棵树，那么对最下层（最细粒度）的对象上锁，需要先对粗粒度的对象上锁。如下图所示，如果需要对页上的记录上X锁，那么分别需要对数据库A、表、页上意向锁IX，最后对记录上X锁，若其中任何一个部分导致等待，那么该操作需要等待粗粒度锁的完成：

粗粒度锁也会有S锁和X锁的情况。举例来说，全表扫、alter表等。

意向锁存在的意义是：减少逐行检查锁标志的开销。

比如事务B要在一个表上加S锁，如果表中的一行已被事务A加了X锁，那么事务B锁的申请也应被阻塞。如果表中的数据很多，逐行检查锁标志的开销将很大，系统的性能将会受到影响。为了解决这个问题，可以在表级上引入新的锁类型来表示其所属行的加锁情况，这就引出了“意向锁”的概念。举个例子，如果表中有1亿条记录，事务A把其中有几条记录上了行锁了，这时事务B需要给这个表加表级锁，如果没有意向锁的话，那就要去表中查找这一亿条记录是否上锁了。如果存在意向锁，那么假如事务Ａ在更新一条记录之前，先加意向锁，再加Ｘ锁，事务B先检查该表上是否存在意向锁，存在的意向锁是否与自己准备加的锁冲突，如果有冲突，则等待直到事务Ａ释放，而无须逐条记录去检测。事务Ｂ更新表时，其实无须知道到底哪一行被锁了，它只要知道反正有一行被锁了就行了。

**12.什么是存储过程？有哪些优缺点？**

　　存储过程是事先经过编译并存储在数据库中的一段SQL语句的集合。进一步地说，存储过程是由一些T-SQL语句组成的代码块，这些T-SQL语句代码像一个方法一样实现一些功能（对单表或多表的增删改查），然后再给这个代码块取一个名字，在用到这个功能的时候调用他就行了。存储过程具有以下特点：

存储过程只在创建时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次，所以使用存储过程可提高数据库执行效率；

当SQL语句有变动时，可以只修改数据库中的存储过程而不必修改代码；

减少网络传输，在客户端调用一个存储过程当然比执行一串SQL传输的数据量要小；

通过存储过程能够使没有权限的用户在控制之下间接地存取数据库，从而确保数据的安全

Distinct关键字作用的范围是整个查询列表,而不是单个的列，因此distinct要放在select后面。

涉及空值的查询，要用exp is [not] null，而不能用”=“ 或者 ”！=“ 或者 ” <>“代替。

如果在select中用了计算列，并且要求按这个计算列进行排序，则在order by子句中有三种方式表示:

  1.计算列的顺序编号 2.计算列的表达式 3.计算列的别名

select 学号,成绩,成绩+10 as 新成绩

from 学生表

order by 3 --第二种方式：计算列的顺序编号

order by 成绩 +10 --第二种方式：计算列的表达式

order by 新成绩 --第三种方式：计算列的别名12345

分组查询

  Group by 子句可以将查询结果集按一列或多列取值相等的原则进行分组。

**Spark  
1.Hdfs的文件结构**

HDFS metadata以树状结构存储整个HDFS上的文件和目录，以及相应的权限、配额和副本因子（replication factor）等。

一、NameNode

HDFS metadata主要存储两种类型的文件

1、fsimage

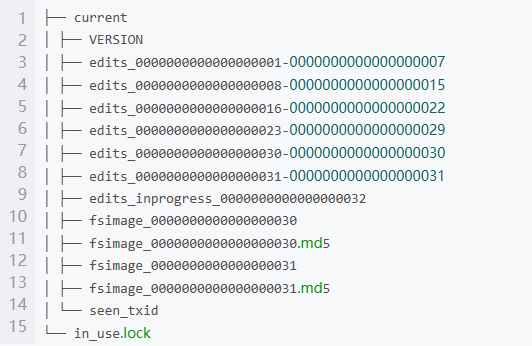
记录某一永久性检查点（Checkpoint）时整个HDFS的元信息

2、edits

所有对HDFS的写操作都会记录在此文件中

Checkpoint介绍

HDFS会定期（dfs.namenode.checkpoint.period，默认3600秒）的对最近的fsimage和一批新edits文件进行Checkpoint（也可以手工命令方式），Checkpoint发生后会将前一次Checkpoint后的所有edits文件合并到新的fsimage中，HDFS会保存最近两次checkpoint的fsimage。Namenode启动时会把最新的fsimage加载到内存中。

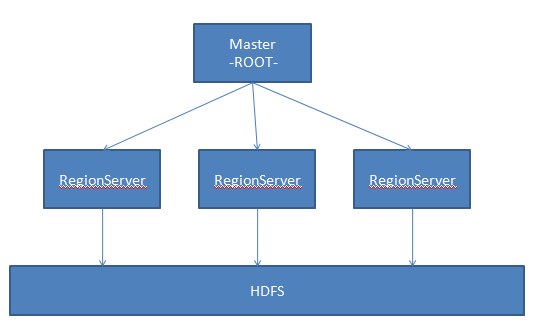


二、Datanode

Datanode主要存储数据，下面是一个标准的dfs.datanode.data.dir目录结构

  
**2.Hbase的数据结构，时间戳是干嘛的(版本)，最主要的设计是什么(行键)，行键是索引吗？**

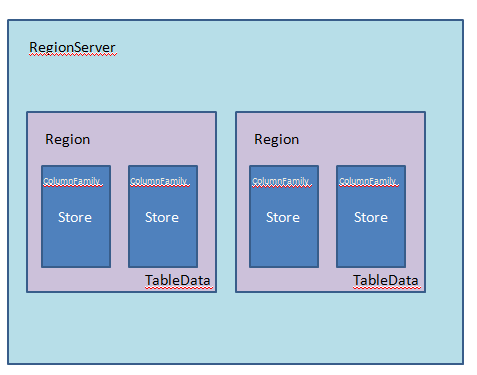
Hbase是一个分布式的数据库，使用Zookeeper来管理集群。在架构层面上分为Master（Zookeeper中的leader）和多个RegionServer，基本架构如图：



在Hbase的概念中，RegionServer对应于集群中的一个节点，而一个RegionServer负责管理多个Region。一个Region代 表一张表的一部分数据，所以在Hbase中的一张表可能会需要很多个Region来存储其数据，但是每个Region中的数据并不是杂乱无章 的，Hbase在管理Region的时候会给每个Region定义一个Rowkey的范围，落在特定范围内的数据将交给特定的Region，从而将负载分 摊到多个节点上，充分利用分布式的优点。另外，Hbase会自动的调节Region处在的位置，如果一个RegionServer变得Hot（大量的请求 落在这个Server管理的Region上），Hbase就会把Region移动到相对空闲的节点，依次保证集群环境被充分利用。

二、存储模型

有了架构层面的保证，接下来的事情就只是关注于数据的具体存储了。这里就是每个Region所承担的工作了。我们知道一个Region代表的是一张 Hbase表中特定Rowkey范围内的数据，而Hbase是面向列存储的数据库，所以在一个Region中，有多个文件来存储这些列。Hbase中数据 列是由列簇来组织的，所以每一个列簇都会有对应的一个数据结构，Hbase将列簇的存储数据结构抽象为Store，一个Store代表一个列簇。

  
**3.为什么要有列族，只有列不行吗**

所以在这里也可以看出为什么在我们查询的时候要尽量减少不需要的列，而经常一起查询的列要组织到一个列簇里：因为要需要查询的列簇越多，意味着要扫描越多的Store文件，这就需要越多的时间。

1. **Hadoop了解过吗？**

**网络和操作系统  
1.线程和进程的区别**

进程是资源分配的最小单位，线程是程序执行的最小单位。

进程有自己的独立地址空间，每启动一个进程，系统就会为它分配地址空间，建立数据表来维护代码段、堆栈段和数据段，这种操作非常昂贵。而线程是共享进程中的数据的，使用相同的地址空间，因此CPU切换一个线程的花费远比进程要小很多，同时创建一个线程的开销也比进程要小很多。

线程之间的通信更方便，同一进程下的线程共享全局变量、静态变量等数据，而进程之间的通信需要以通信的方式（IPC)进行。不过如何处理好同步与互斥是编写多线程程序的难点。

但是多进程程序更健壮，多线程程序只要有一个线程死掉，整个进程也死掉了，而一个进程死掉并不会对另外一个进程造成影响，因为进程有自己独立的地址空间。

进程与线程的同步

进程：无名管道、有名管道、信号、共享内存、消息队列、信号量

进程：互斥量、读写锁、自旋锁、线程信号、条件变量  
**2.网络协议**

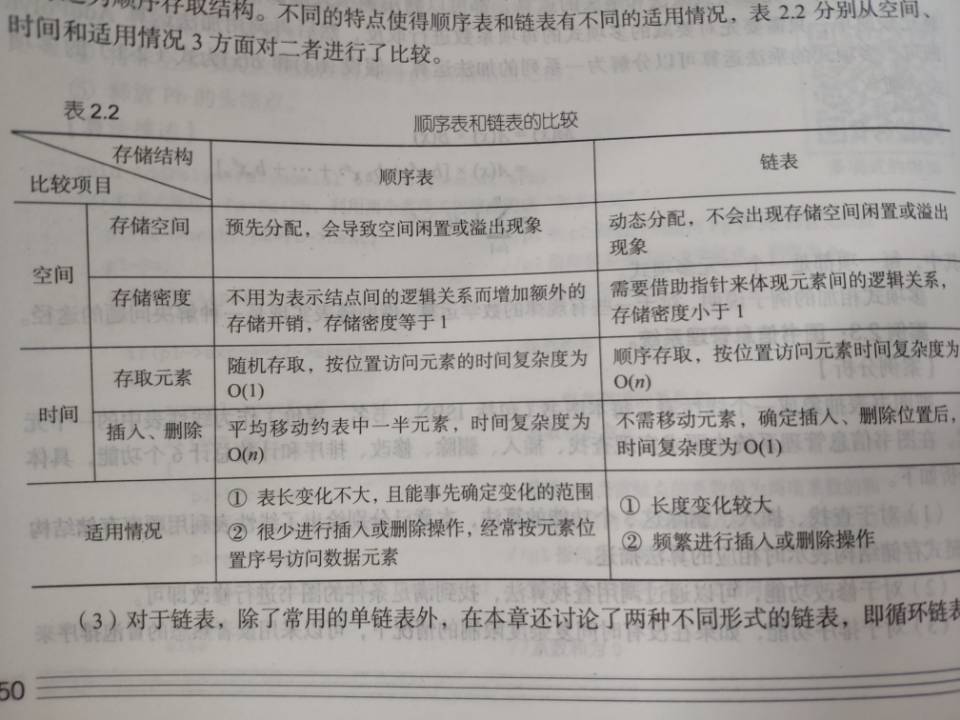
**数据结构：**

1. **动态规划和递归哪个好，有什么区别**

其实递归与动态规划有紧密的关系，且一般递归都可以转化为动态规划。这个问题从一般的递归构成就能够解释清楚，首先，问题可以分解，拆成很多重叠子问题才可以求解，而动态规划也是这一思路，说白了动态规划其实就是记忆化了的递归程序。动态规划把很多递归问题的解存储下来，这样就省去了求许多子问题的解，从而达到了快速求解的目的。

递归其实就是自上往下求解，从顶部一直向下迭代，这点与动态规划相反，动态规划的思路常常是从底向上，以空间换时间。这两种形式是相反的，但是解决问题的形式是一样的，都是不断迭代到底层，递归只不过较多的堆栈存储临时数据而已。

1. **数组和链表的区别？应用场景是什么？读取和插入的复杂度？**

****

**算法  
1.机器学习算法你都知道哪些  
2.xgboost用过吗  
3.随机森林用过吗  
4.RNN和lstm的区别~好在哪**

**个人**

**1.你觉得书上都是真的吗  
2.你一般通过哪些途径去学习新技术  
3.你为什么要做一个任务，你得预期和目的是什么，从产品的角度  
4.遇到什么困难，怎么解决呢？  
5.有没有团队合作的经历？  
6.如果团队中有意见分歧怎么办？坚持自己的意见吗？  
7.你们有过复盘吗？**

作者：优秀的张先生  
链接：<https://www.nowcoder.com/discuss/109518?type=2&order=0&pos=98&page=4>  
来源：牛客网

一面（简单的聊了聊，10min）：

1、介绍项目，以及滴滴的实习经历

2、JVM内存的划分

3、垃圾收集算法

4、数据建模，星型模型和雪花模型

5、数仓层级的划分，怎么对接到mysql拿数据

二面：

1、sql题：写一条sql删除订单表中重复的记录

2、sql题：一张网页浏览信息表，有两列，一列是网页ip，一列是浏览网页的用户（比如a或者b、c、d直到z），求这些网页被a和b或者a和c或者b和c两两组合访问的次数

3、hive数据倾斜产生的原因，怎么解决

4、设计学生成绩管理系统，符合第三范式要求，并绘出UML图

5、算法题：斐波那契数列

6、spark程序的运行流程

7、spark streaming从kafka中读数据的两种方式

8、讲讲数据库索引，B树和B+树

9、Elasticsearch的索引，单field索引和多field的联合索引

10、linux查看某文件的大小，vim中怎么替换内容

11、海量数据的Count问题（单机），如果把大文件hash成不同的小文件，此时小文件装不下某个key对应的数据，该怎么办

12、智力题：8升水，有一个5L的杯子和3L的杯子，怎么得到4升水

三面：

1、osi七层模型，三次握手和四次挥手，为什么两次握手不行

2、kafka怎么保证高吞吐量，项目中有测过吞吐量吗，相比于其它MQ，为什么会选择kafka，kafka怎么保证exactly once语义

3、了解hbase吗，hbase为什么查询速度快

4、hive sql怎么转换成底层的MapReduce程序，以及shuffle的过程

5、算法题：被围绕的区域，leetcode第130题原题

6、智力题：一头母牛每年生一头小母牛，每头小母牛从第四年开始，每年也会生一头小母牛，写个公式求第n年会有多少头牛

蘑菇街：

1、OLTP和OLAP在结构设计上有什么区别？ 2、视图和表的区别？为什么不把数据存在表里而要用视图？视图查询的很慢，为什么不直接查表？... 3、星型模型和雪花模型的区别？说说他们的结构，实际中为什么用星型模型比较多？ 4、数据仓库分层知道吗？ 5、SQL优化说一下？ 6、MySQL的索引有哪些？（B+树和哈希索引），两者主要应用在哪些方面？

作者：优秀的张先生  
链接：<https://www.nowcoder.com/discuss/109518?type=2&order=0&pos=98&page=4>  
来源：牛客网

**腾讯IEG**（offer）

一面（全程问基础）：

1、介绍项目

2、String、StringBuffer、StringBuilder的区别，怎么理解String不变性

3、==和equals的区别，如果重写了equals()不重写hashCode()会发生什么

4、volatile怎么保证可见性，synchronized和lock的区别，synchronized的底层实现

5、sleep和wait的区别，sleep会不会释放锁，notify和notifyAll的区别

6、了不了解线程的局部变量，讲讲线程池参数

7、什么情况会发生死锁，死锁的处理方法

8、Cookie和Session的区别，怎么防止Cookie欺骗

9、从用户在浏览器输入域名，到浏览器显示出页面的过程

二面（全程怼项目，压力面）：

1、看你写过UDF，谈谈对UDF的理解，写UDF的目的，代码怎么写的

2、改造hive表后怎么进行数据一致性校验的，有没有自动化流程

3、看你读过kafka源码，讲讲kafka broker的源码里面你最熟悉的类，以及这个类的主要方法，用的什么设计模式

4、项目里面从数据采集到最终的数据可视化，每个环节都有可能丢数据，怎么判断数据有没有丢，如果丢了如何定位到在哪一个环节丢的

5、项目里面为什么要用kafka stream做实时计算，而不是用spark或者flink，kafka sql和spark sql了解过吗

6、项目里面用到了时序数据库opentsdb，为什么要用这个，有没有跟其它的时序数据库对比过

7、平时逛不逛社区，有没有参与过开源项目

三面（接着怼项目）：

1、看你写了实时计算的程序，你怎么保证计算的结果肯定是对的

2、数据接入的时候，怎么往kafka topic里面发的，用的什么方式，起了几个线程，producer是线程安全的吗

3、kafka集群有几台机器，怎么确定你们项目需要用几台机器，有评估过吗，吞吐量测过吗

4、spark streaming是怎么跟kafka交互的，具体代码怎么写的，程序执行流程是怎样的，这个过程中怎么确保数据不丢

5、kafka监控是怎么做的，kafka中能彻底删除数据吗，怎么做的

面委会（全程聊天）：

平时是怎么学习的，爱看哪些博客，怎么看待加班，有没有成为leader的潜力

**网易考拉**（offer）

一面：

1、sql题：学生成绩表，把每科最高分前三名统计出来

2、算法题：二维数组中的查找

3、kafka如何保证高吞吐的，了不了解kafka零拷贝，具体怎么做的

4、sql有几种join，map join了解过没

5、hbase中row key该怎么设计

6、hdfs文件上传流程，hdfs的容错机制

7、怎么解决hive数据倾斜问题

二面（全程写写写）：

1、算法题：二维矩阵相乘

2、算法题：链表中环的入口

3、写一下mysql binlog的数据格式，怎么进行数据清洗的

4、写一个正则表达式进行手机号匹配

5、讲一下数据仓库层级的划分，每层的作用

**美团新到店**（offer）

去了北京美团公司里面试，一上午面完，第二天通知高分通过

一面（简单的聊了聊，10min）：

1、介绍项目，以及滴滴的实习经历

2、JVM内存的划分

3、垃圾收集算法

4、数据建模，星型模型和雪花模型

5、数仓层级的划分，怎么对接到mysql拿数据

二面：

1、sql题：写一条sql删除订单表中重复的记录

2、sql题：一张网页浏览信息表，有两列，一列是网页ip，一列是浏览网页的用户（比如a或者b、c、d直到z），求这些网页被a和b或者a和c或者b和c两两组合访问的次数

3、hive数据倾斜产生的原因，怎么解决

4、设计学生成绩管理系统，符合第三范式要求，并绘出UML图

5、算法题：斐波那契数列

6、spark程序的运行流程

7、spark streaming从kafka中读数据的两种方式

8、讲讲数据库索引，B树和B+树

9、Elasticsearch的索引，单field索引和多field的联合索引

10、linux查看某文件的大小，vim中怎么替换内容

11、海量数据的Count问题（单机），如果把大文件hash成不同的小文件，此时小文件装不下某个key对应的数据，该怎么办

12、智力题：8升水，有一个5L的杯子和3L的杯子，怎么得到4升水

三面：

1、osi七层模型，三次握手和四次挥手，为什么两次握手不行

2、kafka怎么保证高吞吐量，项目中有测过吞吐量吗，相比于其它MQ，为什么会选择kafka，kafka怎么保证exactly once语义

3、了解hbase吗，hbase为什么查询速度快

4、hive sql怎么转换成底层的MapReduce程序，以及shuffle的过程

5、算法题：被围绕的区域，leetcode第130题原题

6、智力题：一头母牛每年生一头小母牛，每头小母牛从第四年开始，每年也会生一头小母牛，写个公式求第n年会有多少头牛

**小米**（offer）

一面：

1、java和python的区别，对面向对象的理解，和面向过程相比有什么区别

2、java为什么不能多继承

3、讲一下java抽象类和接口

4、java中为什么要写非static方法

5、volatile和synchronized的区别

6、算法题：跳台阶问题

7、算法题：树的非递归后序遍历

8、设计题：一个停车场有一些大车位和小车位，大车只能停大车位，小车既能停大车位又能停小车位，实现这种场景下的调度系统

二面：

1、算法题：输入一个字符串，输出该字符串中字符的所有排列

**贝壳**（offer）

一面：

1、synchronized的底层实现

2、线程等待时位于哪个区域，具体讲一下

3、谈谈对kafka的理解，能讲多少讲多少

4、算法题：二分查找

5、快排的时间复杂度和空间复杂度，最优情况和最差情况分别是多少，是稳定排序吗，快排为什么快

二面：

1、介绍项目，项目中涉及到了一些算法，介绍一下

2、两道算法题：路径问题，leetcode上63题和64题原题

3、写正则表达式匹配电话号码

4、智力题：一张圆桌子，我和面试官轮流往桌子上放硬币（随便放），直到桌子放不下为止，最后一个放硬币的人赢，如果我先放，怎么保证我肯定赢

**华为**（offer）

一轮玄学面：

面试官是做安卓的，瞧不起大数据，觉得大数据很虚，我跟他bb了一堆。然后问我有没有女朋友，我说以前有，现在分了；问我什么时候谈的，什么时候分的，我说本科谈的，毕业分了；问我为什么要分，此处省略一万字......问我现在想没想过再谈，我说毕竟转专业过来的，想趁在校期间利用好短暂的时光提升自己的技术水平（其实因为找不到）；然后面试官说以后工作了就不好找咯，我说您说的有道理............

**快手**（offer）

一面：

1、jvm类加载机制，类加载器，双亲委派模型

2、java实现多线程的方式

3、spark怎么划分stage，宽窄依赖，各包括哪些\*\*\*作

4、zookeeper怎么保证原子性，怎么实现分布式锁

5、写个快排，为什么要用三数取中法，好处是什么

二面：

1、sql题：找出单科成绩高于该科平均成绩的同学名单（无论该学生有多少科，只要有一科满足即可）

2、sql题：找出单科成绩高于该科平均成绩的同学名单（该学生所有科都必须满足）

3、算法题：求数组中连续子数组的最大和

4、算法题：使用最小花费爬楼梯，leetcode746题原题

三面：

1、讲一下java IO

2、算法题：输入n个整数，找出其中最大的k个数

3、算法题：给一个整数数组和一个目标值，找出数组中和为目标值的两个数

**完美世界**（offer）

**京东广告部**（四面完没了消息）

**阿里菜鸟**（三面完已回绝）

阿里的面试还是比较重视基础的，应该是bat里面问基础问的最多的

一面：

1、HashMap和HashTable的区别，HashMap怎么解决hash冲突，jdk1.8后对HashMap的改进

2、讲讲ConcurrentHashMap，ConcurrentHashMap怎么保证线程安全，HashTable怎么保证线程安全

3、HashSet的底层实现，是不是线程安全的

4、ArrayList和LinkedList的区别，是不是线程安全的

5、讲讲设计模式，最常用哪种设计模式，单例模式的实现方式

6、进程和线程，Java实现多线程的方式，什么是线程安全，怎么保证多线程线程安全

7、可重入锁的可重入性是什么意思，哪些是可重入锁

8、为什么要用线程池，线程池的好处

9、JVM垃圾处理方法，对象什么时候进入老年代，什么时候进行FullGC

10、Java堆溢出问题怎么处理，内存泄漏和内存溢出的区别

11、智力题：50个红球和50个黑球往两个桶里放，然后自己去抽，怎么样才能使抽到红球的概率最高

二面：

1、讲讲数据库存储引擎

2、介绍一下索引，索引设置的规则，聚簇索引和非聚簇索引的区别，索引的最左前缀原则

3、用过redis吗，redis支持哪些数据类型，redis与mysql的区别

4、了解垃圾收集器吗，分别介绍介绍

5、jvm调优做过没，-Xms和-Xmx分别指什么

6、算法题：输入两个字符串，输出它们合并排序后的结果

三面：

1、讲讲数据库的范式

2、Linux进程通信和线程通信

3、线程池的参数

4、什么是内部类，什么是匿名内部类

5、设计题：一个市有9个消防站，现在要新增3个消防站，这3个消防站应该放在哪里

作者：bybywudi  
链接：<https://www.nowcoder.com/discuss/117077?type=2&order=0&pos=101&page=1>  
来源：牛客网

**Java基础：**

hashmap:构成原理，扩容过程，put过程，为什么长度总是2的N次方，是否线程安全;

list,set,queue;

一些关键词的作用:final,static等等；

多线程：sychronized,Lock,volatile,CAS。每一个的实现原理和常用场景，可重入锁和公平锁的原理，jdk的锁优化；

线程池：常用的参数，工作过程；

Spring:ioc和aop的实现原理，jdk动态代理和cglib的区别；

单例模式，工厂模式，代理模式等等手写；

Object类中的常用方法：equla,hashcode,wait,notify等等，经常被问，尤其是hashcode的重写，hashcode在容器中的运用；

装箱和拆箱；

异常和error；

jvm：类加载机制，垃圾回收策略，垃圾回收算法的好处，常见的调优方式。

**数据库**：

基本的写sql能力：灵活使用join,groupby,order by以及常见的聚合函数；

索引的原理：b+树的好处，b+树和b树的区别；

联合索引的设计：比如（A,B,C）索引，在where a=1,b>1,c=1中哪些列可以用到索引？where b=1,c=1可不可以用到索引，为什么（联合索引的结构）？where a=1,c=1时可不可以用到索引？where a=1,b=1 order by c可不可以用到？

索引设计：根据一些常用的查询条件设计索引；

范式；

MVCC；

事务，相应的隔离级别，默认的隔离级别；

**redis:**

redis的模型，redis为什么快？redis为什么是单线程的？多核心机器如何部署redis（单进程占用单核）？

epoll,poll和select；

redis，mysql\*\*\*作的时间数量级；

redis各种集群的方式，哨兵模式；

redis持久化的方式，aof文件太大了怎么办，主从同步的方式；

redis的五种常用数据结构，以及对应的底层数据结构的实现（zpilist,skiplist,quicklist,sds等等要有些了解）；

**消息队列：**

常用的消息队列设计和适用的各种场景；

kafka的原理，kafka作为消息队列和redis的区别；

一个topic中的partition是不是一定散布在同一个broker中？

如果要保证消息全局有序，怎么做？

leader选举是怎么选的？

kafka中consumer怎么保持状态的？

kafka是哪个公司出的？（还真老有人问）

**网络：**

tcp的各种东西，udp和tcp的区别和常用场景；

http的各种状态码，以及相应的排查错误的方式；

http请求头等；

老生常谈的输入url按下回车的全过程；

\*\*\*作系统：

经常问的就是linux指令，死锁的各种问题，虚拟内存什么的，这块掌握的不是很好。