### java相关

多个线程访问，每个线程都需要加多个锁，比如线程1要对 a b c三个资源都加锁了，才能操作，那么其他的线程如果要加锁的话顺序必须要和 a b c一致，如果你是c b a顺序加的锁，那么完了，肯定一个拿a等c，一个拿c等a，死锁了！！！

并发修改的时候避免修改丢失，需要加锁，如果冲突几率不是很大的话乐观锁，乐观冲突很大就悲观锁。

注意所有包装类比如integer直接的比较是否相同，一定要用equals，不要用==，因为有integer缓存的存在，所以会有大问题，就是要用equals来比较是否相同。

局部变量的基本类型使用int啥的就可以，而pojo类里面的基本类型一定要使用他的包装类型，还有就是rpc远程调用的参数和返回值一定要使用包装类型。

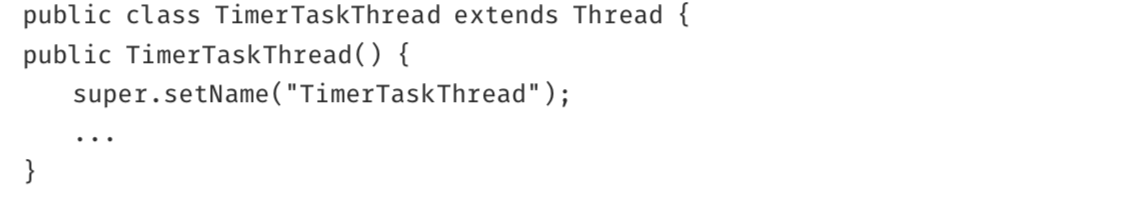
定义集合的时候最好是要指定好初始大小，尤其是那种比较大的hashmap，不然他经常的resize是十分消耗性能的。

map的遍历要使用entryset的方式遍历而不要使用keyset的方式。

只有hashmap的key和value都是可以存储null值的，别的treemap啊，hashmap啊，concurrenthashmap啊都是不能存null的。

创建线程或者线程池的时候请指明有意义的线程名称，方便出问题的时候回溯。

就是这个意思，你创建一个线程后就要给这个线程调用supersetName(xxx)方法来指明线程的名称。



能不使用锁就不要使用锁，锁的范围越小越好。

不允许自己手动的创建线程，必须要使用线程池来创建线程，而且尽量不要用提供的方法来创建线程，避免理解错误。

pojo中的布尔类型变量不要写成isxxx的形式，但是数据库中表的列名要是isxxx的，不然在某些框架下会有问题。

如果在方法中或者类中使用了设计模式，那么在方法名或者类名一定要体现出你使用了这个设计模式。

接口中的方法和变量都不要加什么static啊final啊啥的，就直接方法名打头或者变量列席打头。

枚举类最后要以Enum结尾，枚举类型必须大写。

service和dao层和数据库相关的方法注意要和操作是一样的，getxxx updatexxx countxxx listxxx获取多个对象。

数据对象 xxxDO xxx就是表名。

数据传输对象 xxxDTO xxx就是业务领域相关名称。

展示对象 xxxVO xxx一般为页面名称。

pojo是上面三个的统称，禁止命名为pojo。

不允许有任何未定义的魔法值，也就是不能有任何未定义的常量，就“dsd” + “党风建设”这种就是不可以的。

不要有那种保存了很多的常量类，这样的话不太好，常量跟着他相关的业务一起。

如果变量值仅在一个范围内变化，就要弄成枚举类。

使用//注释的时候注释内容要和//有一个空格的间距。

访问静态内容的话注意要使用Class名来访问，不要使用对象来去访问静态的内容。

object的equals方法很容易出现空指针异常，注意要用常量调用equals方法。

对于pojo类的属性不要搞任何默认值，不然数据还没来为什么有默认值呢。

构造方法里面不要写任何逻辑，如果要有逻辑的话就搞一个init方法。

pojo类必须要重写tostring方法，如果还有父类的话要先调用父类tostring方法。

这样的话当发生异常的时候我们调用一个tostring方法就可以打印这个对象的信息类了方便定位问题。

String s = “a,b,c,,”;对这个东西进行spilt(“,”)分割得到的数组是a b c而不是a b c ，

类中的方法public的在最上面其次是protected的，然后是private的，最下面才是set和get方法。

clone方法默认是浅拷贝。

foreach循环的集合的时候，比如我循环一个list，在循环的过程中不要是用list.remove()方法，这样会报异常，而如果你用的是iterator来遍历，是可以调用iterator的remove()方法的，这样是没有问题的。

random类不要被多线程使用，虽然安全，但是效率低，可以使TheadlocalRandom类，这个是好的。

Volatile对于一写多读问题，是可以解决变量同步问题的，但是如果对于多写就也没啥用类，可以使用并发包atomicinteger，这种，对于1.8来说，推荐使用longadder对象，性能更高。

对于多个if else和switch结构可以使用策略者模式，所谓的卫语句就是ifxxx return；

Try catch不要写在循环里面，在循环外面写try catch。

注意if的判断里面不要写很长的表达式，而是把里面的判断提取出来搞一个boolean变量。

Math.random方法的返回值是double的，是0<=x<1的，也就是可以取到0的，注意别除0了，如果想获取随机的整数就用random.nextint（）就好了。

异常不要做流程控制，要使用条件语句，异常的效率要比条件语句低。

try的时候尽量小精准，不要对一大块代码try。

捕获异常和抛异常要是一样的，不要是父类关系。

防止npe问题，一定要检验使用的东西是不是null的。

### mysql相关

不要搞一个那种大而全的更新接口，传进去的是一个对象，结果就是把这个对象所有的字段都给更新了，这样不好，到低更新哪几个单独去更新。

更新记录的时候主要更新表中的modifytime字段。

单表行数超过500万行或者表容量超过个2T,才要使用分库分表，三年之内不到这个数量级不要分库分表。

数据库里的列的长度不要太大，比如人的寿命200岁就能拦住，就不用弄成一个int，浪费，恐龙化石的寿命4亿，这样我们可以用个long咯。节约资源的目的主要是提升索引查找的效率，如果你的索引很大的话查找的速度是很低的。

在数据表中表达是否的概念，必须用is-xxx的方式，数据类型必须是unsigned tinyinit.

数据库的表名和字段的名字必须要都用小写，因为不同的操作系统对于mysql的大小写支持不同，所以我们全用小写就好了。

主键索引的字段用pk-字段名 唯一索引用uk-字段名 普通索引用idx-字段名

mysql中的小数用decimal来表示，对应的java代码是bigdecimal对象。

Varchar是可变长字符串，不预先分配存储空间，长度不要超过5000，如果存储长度超过此值，定义字段类型为text，独立出一张表来，使用主键来一一对应，避免影响其他字段索引。text的东西都要独立出一张表的。

表必备三个字段，id creat-time数据创建时间 modify-time数据修改时间

表的命名最好是业务名称-表的作用，库名最好和应用名一致。

不用使用count（列名）或者count（常量）来代替count（\*），要用count（\*）或者count主键，因为count（列名）如果这个列名是null的话就不会统计这个的。

使用sum（）的时候要注意NPE问题，因为如果被累加的列如果都是null的话，那么sum出来的结果也是null。

注意和null相关的操作要用is null，和null相关的操作的结果都是null，null=null的结果并不是true，而是null,null<>null的结果也不是boolean类型，而是null。

不得使用外键与级联，外键就不能用，级联就是你sudentid是主键，成绩表的外加是studentid，那么当sudentid的值变了，外键表也要跟着变吗，这样是不对的。

外键就不能用，级联更新更是强阻塞，存在数据库更新风暴的风险，

外键影响数据库的插入速度。

外键和级联只适用于单机低并发，分布式是玩不了的。

禁止使用存储过程，难以调试和扩展，更没有移植性。

修改数据删除数据的时候一定要先查后改。

在表查询的过程中，一律不要使用\*作为查询的字段列表，需要使用哪些字段必须写出。

pojo类的布尔属性不能是isxxx，而数据库字段必须加isxxx，要求在resultmap中进行字段与属性之间的映射。

### ？？？

在varchar字段上建立索引时，必须指定索引的长度，没有必要对全字段建立索引，根据实际文本区分度决定索引长度即可

如果有 order by 的场景，请注意利用索引的有序性。order by 最后的字段是组合 索引的一部分，并且放在索引组合顺序的最后，避免出现 file\_sort 的情况，影响查询性能。 正例:where a=? and b=? order by c; 索引:a\_b\_c 反例:索引中有范围查找，那么索引有序性无法利用，如:WHERE a>10 ORDER BY b; 索引 a\_b 无法排序。

利用覆盖索引进行查询操作，避免回表。

利用延迟关联或者自查询优化超多分页场景。

sql性能优化的目标，只要要达到range级别，要求是ref级别，如果可以是const最好。

建组合索引的时候，区分度最高的在左边。

防止因字段乐境不同造成的饮食转换，胆汁索引失效

错误理解，宁滥勿缺，认为一个查询就需要建一个索引。

宁缺勿蓝，认为索引会消耗空间，严重拖慢更新和新增的速度。