了解Hibernate的FlushMode.NEVER模式 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/182/2021_2022__E4_BA_86_E8_A7_A3Hibe_c104_182510.htm 摘要: Hibernate并没有为巨型数据集合提供良好的帮助，这也许是开发者认为这样没有太大必要，反而增加Hibernate框架复杂性的缘故吧。最近在Hibernate的官方坛子上看到Gavin写给初级用户的“ understand FlushMode.NEVER ”，并参考了一下Stripes项目（本人时常关注的时髦项目）作者Tim的blog。在阅读两位大家言论后，和大家share一下。 一、案件背景：图片来自于电影《天生杀人狂》Hibernate并没有为巨型数据集合提供良好的帮助，这也许是开发者认为这样没有太大必要，反而增加Hibernate框架复杂性的缘故吧。于是“ 极大数据量==批量处理 ”、“ Hibernate/java不是批处理的最佳场所 ”的观念在Hibernate开发中大行其道，有些开发者甚至直接利用Hibernate建立session，获取其connection进而进行jdbc操作。Jdbc并不是古董，但在Hibernate中再次call它，难免有些令人无奈。最近在Hibernate的官方坛子上看到Gavin写给初级用户的“ understand FlushMode.NEVER ”，并参考了一下Stripes项目（本人时常关注的时髦项目）作者Tim的blog。在阅读两位大家言论后，和大家share一下。二、性能杀手何在？图片来自于电影《这个杀手不太冷》 Tim在其Blog写道：“ 我目前的DNA重组系统，具有复杂而海量的OLTP数据，对付这些在内存的复杂对象（数千个）的方式是依赖用户接口（非批量处理）来实现用例驱动。 ”这句半开玩笑的话，是我想起了那男耕女织的生产力低下的生活，真的让每个开发者都使用

算盘运算吗？session.setFlushMode(FlushMode.NEVER). 这条语句及其简单，但解决了大问题。它告知Hibernate session无论何时也不要flush任何的状态变化到数据库，除非开发者直接调用session.flush()。听上去很合乎逻辑，但它为何在一些场景中对性能影响甚深，而在其他的场景中却好似轻如鹅毛般？在Tim的项目中存在着一个十分典型的case（我也不大了解生物，这不能怪我）：在实验中利用PCR Primers对遗传基因（genes）和DNA中的核苷酸序列（exons），这里的PCR Primers是在PCR处理过程中用于检测DNA片段的物质，对不起大家，本人对生物学词汇实在无能为力。检测匹配过程大致分为以下7步：1．发现本次实验中所有exons（个数在5000个以上）；2．查询本次实验所有已经排序的PCR Primers；3．查询本次实验所有待排序的PCR Primers；4．得到与exons对应的Primers找出那些无需转换的部分；5．在系统中为无需转换的区域查询所有可能的PCR Primers；6．测试每个primer找出最佳exons匹配者（Primer）；7．保存找出的Primer。 不用担心，步骤细节不大明白也不会影响后面的理解。 由于domain model极其海量，在第4步我们可能在一个session中排序20000-30000个对象。而在5、6步的查询将带来0-20个附加对象。有趣之处在于当执行第7步将对象save到数据库时，没有一个前面装载的对象被修改过。整个实验的目的就是仅仅获得这0-20个对象。在回顾了Tim的生物学场景之后，让我们重新回到FlushMode.NEVER的讨论上来吧。你可能认为既然直到最后一步都没有修改或是持久化任何东西，那么改变flush模式将收效甚微。当然这是不正确的未参透实质的理解。实际上，在上面流程的起始设置Never这个flush

模式、在流程终点手动flush将节省一半的run time，请注意这里仅提到了run time而没有将内存、IO计算在内。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com