DBA在系统设计、开发中的重要性 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/143/2021_2022_DBA_E5_9C_A8_E7_B3_BB_E7_c102_143083.htm 许多应用系统的性能或稳定性并不理想，这在系统上线后不久就逐渐变为棘手的问题，造成这些问题的原因，往往体现了一点：开发设计这些系统的人，对数据库本身不是很了解！而DBA又不了解业务！这就导致了很多本来可以避免的问题产生；另一方面，随着数据库自我调整、管理的能力不断加强，而应用又往往是系统性能最大的杀手，所以，DBA的工作范畴，从只负责数据库服务器维护，逐步走向管理应用系统的设计、开发，是必然的趋势！ 一、 现阶段DBA对系统性能及稳定性所做的调整工作 目前DBA对系统性能的调整工作大致是这么几个方面：1、 在硬件层面进行调优，这通常就是直接花钱，买设备、扩容。 2、 在DB层面进行调优，比如调整初始化参数，调整数据库物理结构。 3、 对应用的SQL进行优化，比如在数据库分析statspack，调整Top SQL。 4、 只有非常少数的，通常是对系统稳定要求较高的一些公司的应用，才会在新的应用上线前，让DBA对sql进行充分的审核与评估。 问题：在应用系统的分析、设计、开发阶段，就目前情况看，很少有DBA参与，而应用系统上线或者开发工作基本结束后，DBA所能做的调优工作其实是很有限的。 二、 许多应用系统的性能或稳定性仍不理想 许多应用系统的性能并不理想，或者系统数据会出现一些难以重现的奇怪的错误，这些问题（尤其是性能问题）有时并不是在系统初期就会体现出来，但是随着系统的运行、数据的增多而逐步变得难以解决，给系统后期的功

能扩展和用户使用上带来了不少麻烦，造成这些问题的原因，往往体现了一点：开发、设计这些系统的人不了解数据库！以基于Oracle的应用为例，简要举例说明：底层数据结构不合理 由于缺少专业DBA的协助，很多系统设计出来的底层数据库表结构问题重重。而做过系统的人都知道，底层数据库结构不合理，带来的改造代价之大几乎等于一次重构！我见过一个OLTP系统，其核心表竟有100个字段，平均一条记录超过8K，如果按Oracle默认的8K一个Block，一半以上的行必须产生行链接！ 而最糟糕的是，设计这样表结构的人还认为自己充分利用了冗余来降低表之间的连接，事实上，其人根本不晓得什么是范式、什么是更新异常，按照范式，这个表应该拆分为两个表的，但如果要改几乎所有的程序都要改！虽然范式不是越高越好，但绝对是设计的人必须吃透的一个东西。在冗余上，相信大多数DBA都认为，级联更新的代价是非常高的，因此冗余应当避免发生级联更新的情况，对于关系型数据库设计中冗余的使用，绝不是门很容易掌握的技巧。 不合理的底层数据库结构设计，给系统的性能埋下了重磅的定时炸弹，这个系统在客户那里跑不到一年，数据量稍微上去些，性能、稳定性就直线下降，而重构的成本又极高，买新服务器肯定是只能治标。而假如底层数据表结构是资深DBA设计的又会如何？当然，如果完全让DBA去做数据库表结构的设计，DBA就必须非常清楚地了解整个系统的业务细节信息，这在DBA来说，人力资源上是有一定困难的，毕竟维护好线上服务器就已经占用了DBA很多的资源，并且领导们通常更看重这点。 很少有领导能认识到DBA在系统开发设计中所起到的作用，和维护线上系统、处理DB故障相比

，对整个系统的稳定性和性能，是同样重要的！ SQL性能问题 系统的开发，通常和DBA是没有什么关系的，但是，如果DBA对系统有足够的了解，这时候也是可以做不少贡献的。比如，检查系统业务的数据流是否正确，这个需要通过一些手段，比如sqltrace、10046等，详细对系统的逻辑实现进行检查，一方面查出系统中过于消耗资源的或编写不规范的SQL及时进行调整优化，另一方面，查出系统中不合理的数据库访问，不要到了线上才发现问题，那时可能已经宕机了。简单举个例子，当一个页面需要多处显示商品的类目列表时，程序往往容易犯一个错误，就是多次以同样的SQL读取出同样的数据，并应用于每一个列表显示上，如果你只读取一次，或者干脆在Web层进行cache（要有适当的刷新策略），就可以大大减少单次访问该页面在DB上的I/O消耗。有时甚至会检查出根本不需要被执行的SQL，也在这些和自己毫不相干的功能中频繁地执行着……同时，对数据流的检查还能够查出一些隐藏得较深的系统Bug，这个更需要基于DBA对业务细节的了解。 谁说DBA只会花钱？如果一个服务器I/O负载达到极限，大多数人只能选择扩容，最多重构部分功能来作些优化，而从statspack往往可以看出，系统的I/O资源多数是被一些并不该如此频繁执行的SQL给占用了，它们单次执行并不慢，但占用系统资源比例却异常高，这些问题，细化在每一个业务中，对这些问题的检查和数据流优化，就是对系统资源的最大节省，就是省钱！这个工作，或许只有DBA才能称职。 并发问题 谁都知道系统有并发存在，可是我们在设计系统的时候，又往往是按照单一业务的思维模式来设计、编码，很少考虑同一业务、不同业务之间并发运作

可能产生的问题。通常，系统无规律地出现一些“奇怪的”、“不可能的”错误，极有可能就是并发惹的祸，而背后的问题，往往体现了设计人员不了解数据库的锁机制，无法和业务很好地结合。设计的人不了解数据库，而DBA又不了解业务，这就导致了很多本来可以避免的问题产生。 最经典的就是Tom Kytes举的酒店预定的例子，当两个服务员同时按下查找预定房间的按钮，结果是两个人都预订到了同一间客房，这个问题很经典，在目前看来也很容易解决，不就是加上锁么？但是，这只是一个例子，在你实际应用的系统中，你这样贸然地加上for 0update，又可能导致别的问题！比如：死锁。在一个复杂的业务系统中，死锁不难见到，这个是设计者的设计漏洞，需要设计者全面衡量业务关系，然后对表的锁定制定规则来尽量避免的。学会使用锁来保证数据的完整性还是不够的，还要灵活应用锁，适当采用乐观锁定。 如果对于重要的业务，一律免谈，直接悲观锁定也是不可取的，会给系统的维护带来一些问题，某些业务这样做甚至会带来数据的大面积锁定时，在OLTP上的代价很高，严重影响系统并发能力。我曾经碰到一个错误数据的问题，分析后，确定是两个不同的业务间并发，同时缺少必要的锁定，而造成的错误数据。但基于该业务的特殊性，加锁的代价是昂贵的，在DBA的仔细追究下，确认了可以通过乐观锁定也即提交时检验的方式来达到两全其美的目的。从这里可以看出，数据的健康和完整性，与系统的并发能力有时是矛盾的，但有经验的DBA能够教你如何获取最佳方案，当然，前提是DBA参与设计并熟悉业务。 系统架构的问题 DBA不是系统架构师，但数据库是一个应用系统核心的部分，同时，由于数据库服

务器不像应用服务器那样便于扩展，因此往往也是整个系统性能的瓶颈所在，所以架构师在设计系统架构时，应该充分考虑DBA的意见，要考虑到DBA对数据库中的SQL进行性能调整的便利性甚至是可行性！否则就可能导致DBA及开发团队对系统的性能问题反应过慢甚至束手无策！我曾经见过一个架构，它无法实现oracle最普通的分页SQL！绑定变量就根本不在考虑中！再就是有些第三方组件或架构，能够帮助我们的系统生成SQL，这当然很省事，能够加快开发速度，可是在这样的系统中，DBA如果想要优化一条SQL可能很难，因为开发人员要修改的东西相对较多，修改的工作量大、耗时长，并且工作量多肯定就更容易带来新的错误!Oracle大师Tom Kytes也曾在经典著作Export one on one中反对使用这种自动产生SQL的组件或架构,这种东西很可能给你的系统带来性能和维护上的问题！这些问题，如果咨询过资深DBA，相信会尽早发现并在架构上得到修复或调整。而到了系统开发的后期，架构的问题已经很难做本质的调整了。假如你的系统要求非常高效，并且并发访问较大，那么建议架构师倾向于尊重DBA的意见，这对整个系统的性能以及持续地调优将非常重要。而DBA，对系统架构也要有一定的认识，并明确自己在现有架构中遇到的困难是什么。 三、 提高应用系统的性能、稳定性 除了DBA原本的DB调优、SQL调优、服务器维护等日常工作以外，扩展DBA的工作范畴，强化DBA在系统开发过程中的控制能力和决定权。 1、 让DBA参与到需求分析中去，并充分理解用户需求，从DB的角度来理解和考虑这些需求实现的成本。 2、 Schema的设计必须由DBA设计确定或者审核确定，这点也要求DBA必须了解业务系统，才能整

理出正确的、有良好扩展性的E-R关系。 3、 让DBA更深入的参与系统的设计，尽可能地让DBA了解应用的业务设计细节，这对于DBA审核数据流是起到决定性作用的，如果有条件，业务的数据流应当作为系统的文档之一，以便将来的反复核查。 4、 在系统上线之前，由DBA审核sqltrace中的sql以及数据流逻辑，最好是能给出一些重要业务功能在DB成本（比如I/O）上的评估结果。 5、 系统上线后的性能监控，及时作出调整甚至一定范围内重构优化数据访问逻辑。 如上所述，则DBA的人力资源必然不足，因此，细化DBA的工作，进行分工是正确并且高效的，在一些公司，已经将DBA分为专管线上服务器的产品DBA和专管开发、参与系统设计的开发DBA，从不同方面全面保障系统的稳定和高效，值得借鉴！