 使用作为 Oracle 高级安全选件（版本 10.2 及更高版本）的一部分引入的 Oracle 数据库透明数据加密（TDE），可以有选择地对保留在数据库底层数据文件中的敏感数据库数据以及所有下游文件组件（如联机重做日志、归档重做日志和数据库备份）进行加密。 TDE 的基本目标是保护在这些原始操作系统文件中发现的敏感数据，防止不怀好意的人访问磁盘或备份磁带时对这些数据进行窥探，然后尝试还原数据库或扫描原始操作系统文件中的数据，如个人可识别信息或信用卡信息。 作为我的咨询惯例的一部分，我已经实施 TDE 多次。但是，在其中一个最近的合约之前，我一直使用 TDE 对现有表中的新列或属于全新表的列进行加密。在这两种情况下使用 TDE 非常简单，因为目标列为空，因此由于缺乏数据和现有应用程序相关性而不会涉及较大的风险。 我最近实施 TDE 的体验有所不同。我帮助一家大型公司对一个已超过一百万行的表中的现有列进行加密。还有一个依赖于列的关键任务应用程序，因此，您可以设想一下，在开始工作之前有很多重要的事情要考虑。在 Internet 上搜索可提供经验的类似情形之后，我发现只有几个优秀的资源可以帮助我。 本文概述了我在通过使用 TDE 对现有数据进行加密的过程中总结出的经验教训。如果您尝试对现有列数据使用 TDE，我希望此处提供的信息可帮助您迅速有效地开展类似工作。 确定可能的限制 研究客户的系统时，我做的第一件事情就是查找与目标列有关的将禁止我们

对列加密的数据模型特征，或者查找可能对现有操作产生负面影响的有关列的事项。该研究包括查找列索引和完整性约束。 正如 Oracle 文档明确声明，当您想对具有索引的某个列进行加密时，需要了解很多限制条件。Oracle 不允许对具有位图索引的列进行加密，这与我们的情况没有密切关系。但是，目标列具有多个普通的（B 树）索引。尽管 Oracle 允许对具有普通索引的列进行加密，但是 Oracle 禁止对索引列进行“ salt 处理 ”加密。Salt 处理通过在加密之前向数据添加随机字符串来提高重复数据的安全性，因此窃贼使用模式匹配识别技术更加难于破解加密的数据。总而言之，经过这个最初的分析之后，我们会遇到一种情况，那就是我们可以对列进行加密，但不能进行 salt 处理。 对列索引进行分析后，我本可以到此为止，但是我想回答的下一个问题是“ 使用这些索引合适吗？ ”我的思考过程是这样：如果索引没有用，那么我会将其删除，从而减少维护索引条目所必需的系统开销，尤其是考虑到加密的额外负担。要判断索引是否有用，我使用 Oracle 数据库的索引监视特性。我发现，实际上索引正处于使用当中，因此我们必须对其继续进行维护。 接下来，我查看了引用完整性约束条件中是否涉及目标列。由于每个表都具有其自己的加密密钥，因此 Oracle 不允许您使用 TDE 对外键关系中涉及的列进行加密。在我们的情况下，引用完整性约束条件中未涉及目标列。 评估性能开销 我的客户询问的第一组问题之一就是“ TDE 对我的应用程序的一般性能影响如何？ ”Oracle 文档中有一小部分论述了一般情况下 TDE 对相关应用程序性能的影响。但是我的客户希望获得一些具体的统计信息，以帮助他们了解 TDE 如何影响日常进行的有

严格时间要求的数据加载过程。为了满足客户需求，我计算了每天在有严格时间要求的过程中插入到目标表中的平均行数。然后，我在客户端的相同沙箱环境中创建了一个类似的测试表和索引，测量在加密目标列前后插入相同数量的行所花费的时间。时间消耗上的差别让我们更好地了解了在该过程中对列数据进行加密所造成的“性能损失”。列表 1 是我如何使用 SQL*Plus 执行该操作的示例。 SQLgt. -- Configure Oracle-Managed (Data) Files SQLgt. -- Create two new tablespaces for the demo, SQLgt. CREATE TABLESPACE data_001 2 DATAFILE SIZE 1G. Tablespace created. SQLgt. -- Create a user for the demo SQLgt. GRANT CREATE SESSION, CREATE TABLE TO app_001. Grant succeeded. SQLgt. CONNECT app_001/app. Connected. SQLgt. CREATE TABLE app_001.transactions (2 trans_id INTEGER 3 CONSTRAINT transactions_pk PRIMARY KEY 4 USING INDEX TABLESPACE indx_001, 5 credit_card INTEGER NOT NULL 6). Table created. SQLgt. CREATE INDEX app_001.transactions_ndx1 2 ON app_001.transactions(credit_card) 3 TABLESPACE indx_001. Index created. SQLgt. SET TIMING ON. SQLgt.0, high=gt.0, high=gt.0, high=gt. SET TIMING OFF. SQLgt. TRUNCATE TABLE app_001.transactions. Table truncated. SQLgt. ALTER TABLE app_001.transactions 2 MODIFY (credit_card ENCRYPT NO SALT). Table altered. SQLgt. SET TIMING ON. SQLgt.0, high=gt.0, high=gt.0, high=gt. SET TIMING OFF. 列表 1 使用与您的生产环境相同的沙箱环境，简单比较启用列加密前后加载代表性数据集所花费的时间，以使您更好地了解列加密对

生产系统性能的影响。 和所有的性能测试一样，我怀疑对列进行加密所造成的性能损失会因系统而异，具体取决于普通变量（CPU、平均负载等）。在列表 1 中，您注意到计算的性能损失为 36% (((56.14-76.31)/56.14)*100)，但是，使用我们在客户系统中收集的实验证据，预计数据加载过程所耗费的时间应该大约增加 11%，这与在生产中使用 TDE 获得结果完全一样。 在本例中，我侧重于对具有索引的数据加载过程估计数据加密的性能损失。如果您的系统具有不同类型的关键过程，如要求苛刻的报表生成周期，那么我建议您使用沙箱环境来比较数据加密前后该过程所花费的时间。本文后面的“ 确定潜在查询计划更改 ”部分将讨论查询和数据加密的特别注意事项。 处理停机和维护时间 我的客户比较关心的另一个问题是，在加密约一百万行的表中的现有列数据时，需要对哪些生产应用程序（如果有）进行必要的停用。我最初想法是，理论上不需要停止任何应用程序 毕竟，Oracle 文档明确表示了对现有列的数据进行加密本质上就是对整个表进行多行更新。如果没有更多地考虑这件事，我不会明白为什么新行无法并发插入到表中以及为什么现有行更新无法继续。当我咕哝着熟悉的 Oracle 口号“ 读取方不会阻止写入方，写入方也不会阻止读取方 ”时，我的确没有想到列加密会影响查询。 但是，在长时间从事 DBA 工作后，我才总结出，若要对生产系统进行最终的实际更改，需要对理论进行测试，以避免出现意外问题，这一点非常重要。您瞧，当我在加密列期间，针对沙箱数据库对应用程序本身进行了测试，从而发现了很多问题。最重要的是，我发现进行中的加密延长了某些查询的响应时间，以至于应用程序会遇到响应超时。这些

超时又会造成连接断开，然后导致后续的事务失败，进而会更加麻烦 我将为您提供详细信息。 必须一提的是，测试之后，我了解到停止应用程序运行绝对不是没有理由的。但下一个问题是，生产应用程序需要脱机多久？在计划每个周末进行的正常两小时的维护时间之内能够对列进行加密吗？或者，需要更长的停机时间？为了弄清这个问题，我只需测量在沙箱环境中对列进行加密所花费的时间，因为沙箱环境与生产环境具有相同的服务器硬件和数据集。我发现，列加密要花费一个小时多一点的时间才能完成。坦白地说，由于我使用类似数据在笔记本电脑上模拟测试加密运行才花费了不到 5 分钟的时间，因此对于它花费这么长时间，我感到非常震惊。但是当我们在生产数据库系统中对列进行加密时，最要紧的是要使用陈旧服务器硬件所发生的情况。 了解到在正常维护时间内执行其他任务需要更多时间，我决定必须找到减少加密列花费时间的方法。我的第一个直觉就是删除包含目标列的两个索引。这样，Oracle 只需加密表本身中的列数据，之后我可以有效地重建索引，而没有日志记录开销。经过一些新的测试之后，我将加密列以及相关索引所需的时间从 70 分钟（在加密期间存在索引）减少到仅 20 分钟（加密列后重建索引）。列表 2 是我用来得出结论的测试示例（从我们在列表 1 中停止的位置继续）。此外，请注意，列表中的时间来自用来编写本文的测试系统，而不是来自我的客户端使用的实际系统。 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