Oracle Database 12cR2 In-Memory 性能优化实战

[美] 乔吉特·班纳吉 (Joyjeet Banerjee) 著

张骏温 许向东 王 健 译

清华大学出版社

北 京

Joyjeet Banerjee
Oracle Database 12*c* Release 2 In-Memory: Tips and Techniques for Maximum Performance
EISBN 978-1-259-58616-3

北京市版权局著作权合同登记号　图字：01-2017-8966

图书在版编目(CIP)数据

Oracle Database 12*c*R2 In-Memory 性能优化实战 / (美) 乔吉特·班纳吉(Joyjeet Banerjee) 著；张骏温，许向东，王健译. —北京：清华大学出版社，2019

书名原文：Oracle Database 12*c* Release 2 In-Memory: Tips and Techniques for Maximum Performance

ISBN 978-7-302-52760-2

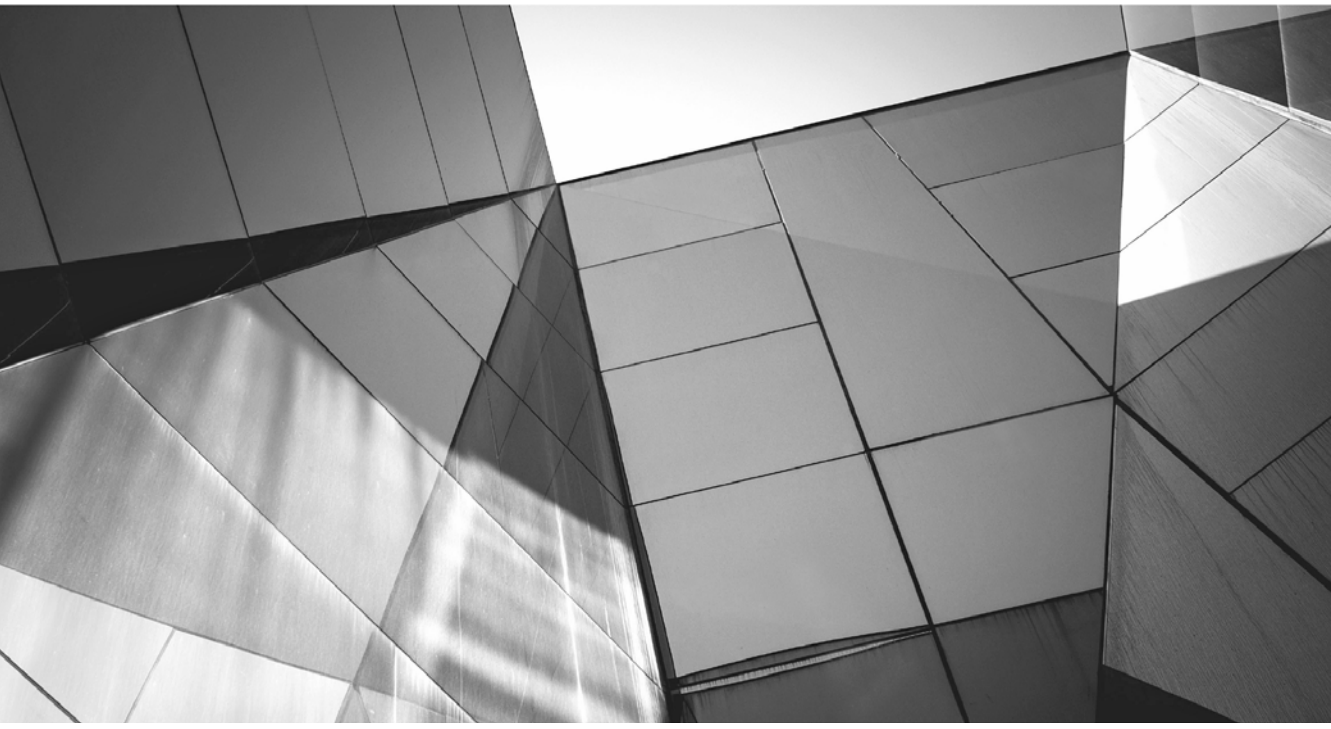
①O…　Ⅱ. ①乔… ②张… ③许… ④王… Ⅲ. ①关系数据库系统　Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字(2019)第 070970 号

责任编辑：王　军
封面设计：孔祥峰
版式设计：思创景点
责任校对：牛艳敏
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　**印　　张**：13.5　　**字　　数**：265 千字
版　　次：2019 年 5 月第 1 版　　**印　　次**：2019 年 5 月第 1 次印刷
定　　价：59.80 元

产品编号：077461-01



译 者 序

现代商业的运作模式在不断变化中，为了能够快速应对需求的不断增长，企业客户需要能够快速地获取支持其决策的信息。这就要求企业客户除了在传统数据仓库上执行分析外，还需要直接在其交易型系统上运行分析。在这种需求驱动下，Oracle Database 12*c* 推出了一项开创性的新特性——内存数据库(Oracle Database In-Memory)。

Oracle 内存数据库的创新之处是在内存中支持行和列两种格式的存储，即独特的双模结构。行格式适合 OLTP 应用访问，列格式适合 OLAP 分析型应用。新的列格式完全基于内存，可以将整个表空间、表、表的部分列、分区和子分区置于内存中。借助 Oracle Database In-Memory 的这些特性，单个数据库就可以高效地支持混合工作负载，在为事务型操作提供最佳性能的同时，也支持实时分析和报表。

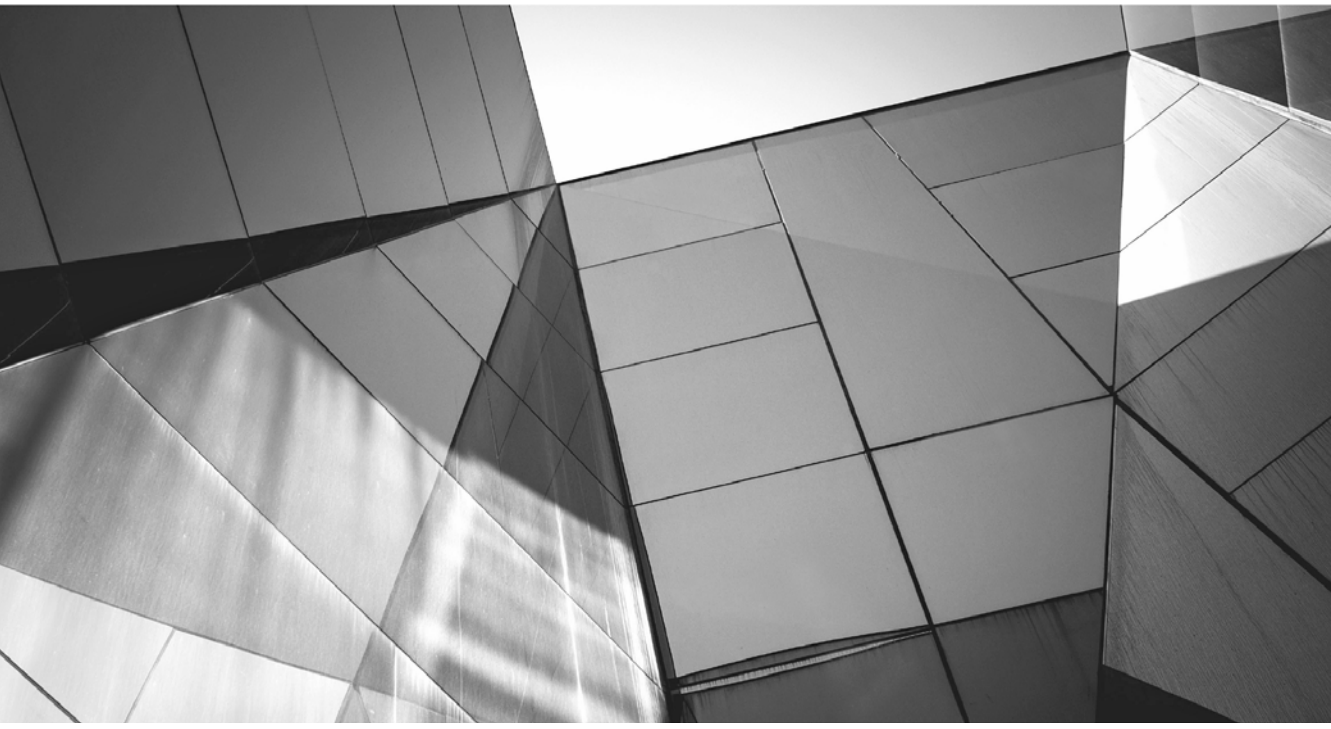
Oracle 内存数据库与已有的 Oracle Database 企业版中的各种功能，如 RAC、

多租户、ADG 等充分地兼容并结合使用，同时确保对应用层不做任何更改。在我们的实际测试案例中，Oracle 内存数据库不但能够透明地、成倍地加速分析型查询的响应速度，实现实时的商业决策，同时在混合负载的环境里也能加速操作型事务处理。

本书循序渐进地介绍如何使用 Oracle Database 12*c* Release 2 In-Memory 来优化数据库性能、缩短事务处理时间。作者从 Oracle 企业架构师的视角给出了实现数据库最佳性能的建议，以及实际操作说明。通过阅读本书，你可以学会如何快速部署 Oracle 内存数据库软件、使用 In-Memory Advisor、构建查询以及在 Oracle RAC 集群和多租户环境中使用 In-Memory，并通过具体案例研究说明如何在实际应用中实施 In-Memory。

本书系统地阐述了 Oracle 内存数据库的特性、原理、部署和操作实践，是一部关于 Oracle 内存数据库特性的专题书籍，我们希望这本汇集了原作者和译者实践经验的中文版译著，能够帮助国内的用户、数据库管理人员和技术人员更深入地了解 Oracle 内存数据库技术，本书第 1～3 章由张骏温翻译，第 4 章由许向东翻译，第 5～8 章及附录由王健翻译。欢迎读者提出宝贵的意见和建议。

译者　许向东



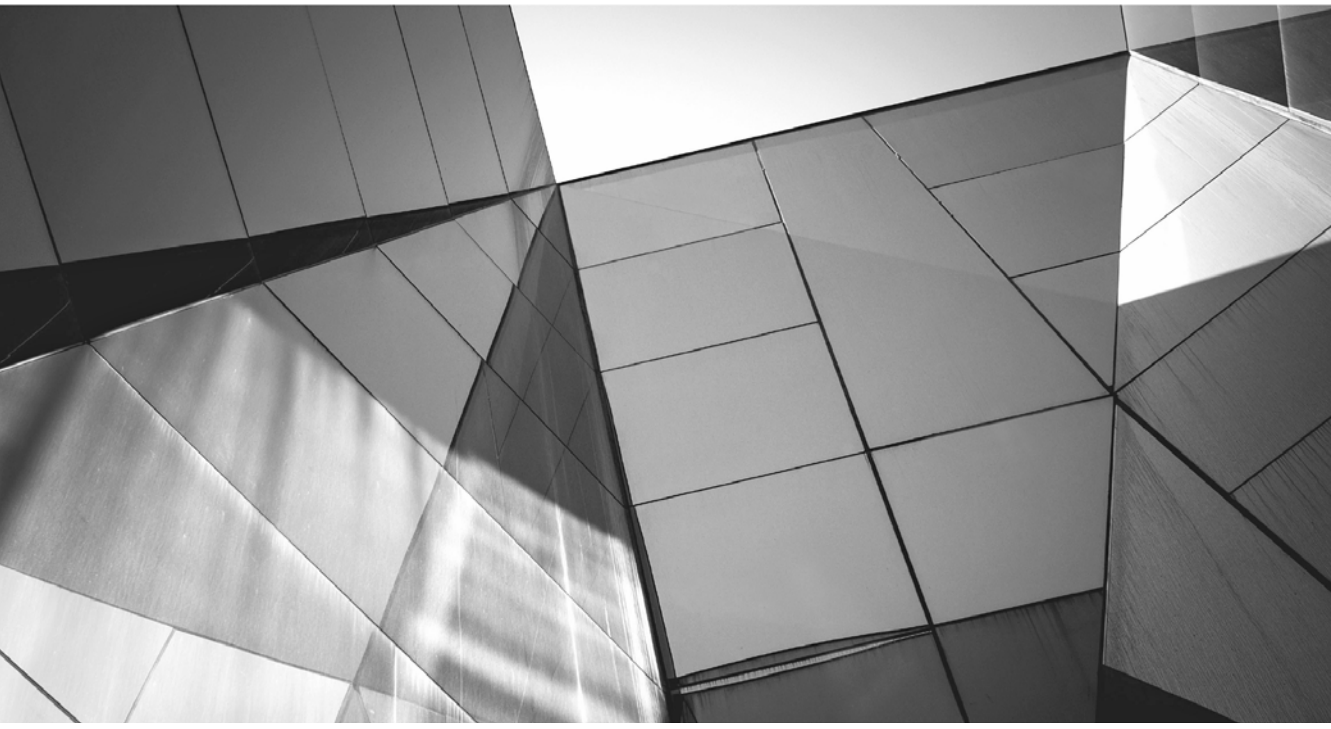
译 者 简 介

张骏温，北京交通大学计算机与信息技术学院副研究员。多年来一直从事 IT 技术相关专业的教学和科研工作，主持或参加了多项国家级和省部级重大科研项目，培养了多名硕士研究生。

2012 年主编了书籍《计算机网络系统工程监理规范理解与实施》；2013—2018 年，与清华大学出版社合作，翻译出版了《Effective MySql 之备份与恢复》《MySQL Workbench 数据建模与开发》《Oracle PL/SQL 性能调优诀窍与方法》和《精通 Oracle R Enterprise 大数据》四本书籍。目前研究方向包括：数据库、网络工程、信息安全、信息化工程监理、软件测试和现代通信技术。

许向东，甲骨文(中国)软件系统有限公司、云平台事业部售前总监，2002 年加入 Oracle 公司。对金融、政府、交通、电力等行业的 Oracle 数据库及相关选件产品的设计、开发、运维管理和培训工作做技术支持。2014 年、2015 年、2016 年和 2017 年，分别与同事一起翻译了《深入解析 Oracle Enterprise Manager Cloud Control 12*c*》《Oracle 大数据解决方案》《Oracle Database 12*c* 完全参考手册》《Oracle 数据库升级、迁移和变更最佳实践》《构建 Oracle 数据库云最佳实践——使用 Oracle Enterprise Manager Cloud Control 13*c*》五本书籍(均已由清华大学出版社出版)。

王健，甲骨文(中国)软件系统有限公司、云平台事业部售前顾问，2006 年加入 Oracle 公司，支持过 Oracle 数据库及相关选件产品的设计、开发、运维管理和培训工作。目前主要负责招商银行、平安集团、广发银行、平安银行等金融行业的技术支持，提供数据库相关解决方案、应用数据平台和混合云平台架构设计。具备丰富的行业经验和技术积累，对公有云、数据库技术有深入理解。



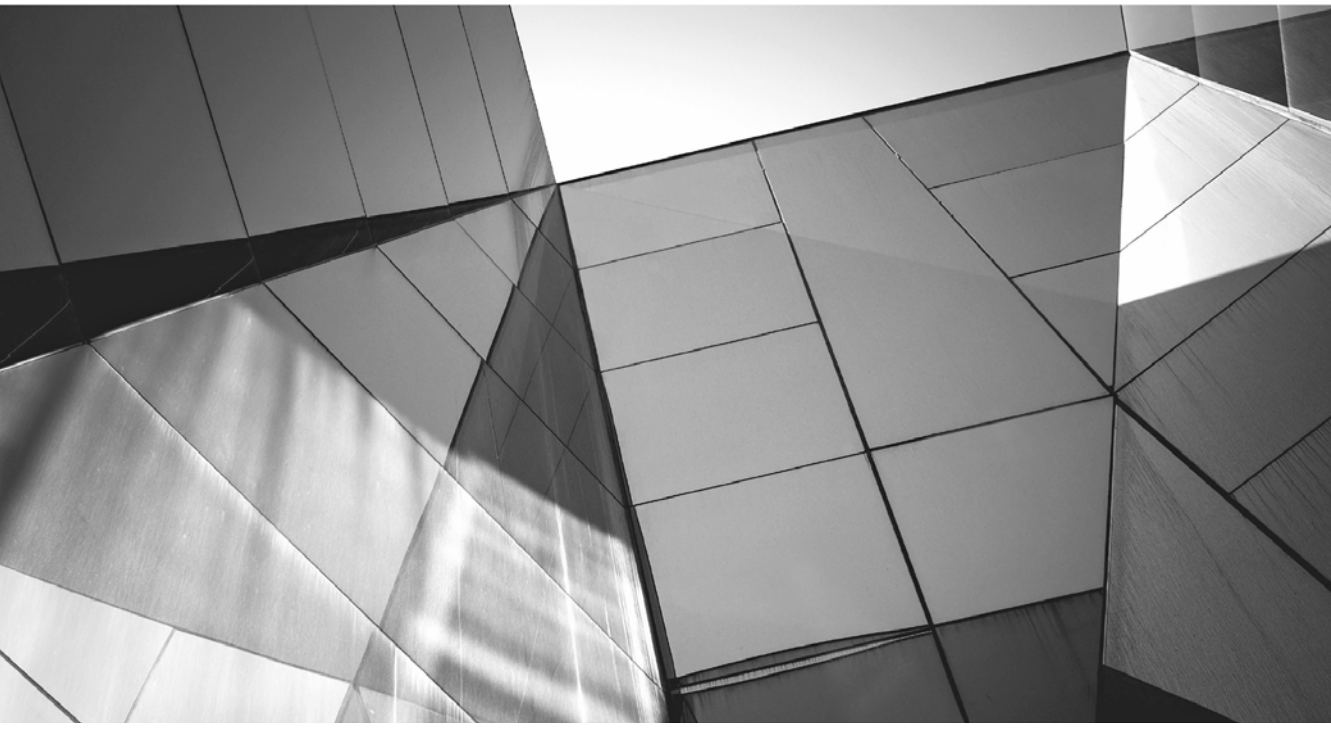
作 者 简 介

Joyjeet Banerjee 是亚马逊网络服务(Amazon Web Services)的企业解决方案架构师，负责开发能够满足客户业务需求的高可扩展的、灵活的、弹性云架构。Joyjeet 帮助机构了解基于云的高级解决方案的最佳实践以及如何将现有工作负载迁移到云上。在加入 Amazon Web Services 之前，Joyjeet 曾在位于加利福尼亚州的 Oracle 总部、Oracle 工程化系统团队担任企业架构师。他的主要职责是帮助 Oracle 企业客户从传统系统迁移到工程化系统。在加入工程化系统团队之前，Joyjeet 曾在 Oracle 咨询部门工作，协助 Oracle 数据库、RAC 和 EBS 客户实施 Oracle 产品。Joyjeet 于 2003 年加入 Oracle，担任电子商务套件(E-Business Suite)开发组织的数据库管理员，专门负责 Oracle 数据库的应用性能。

Joyjeet 是 Oracle Applications Users Group(OAUG)Collaborate 和 Oracle OpenWorld 的资深演讲嘉宾。他曾在 Oracle University 的专家小组(panel)中工作，负责评估课程内容并为 Oracle 数据库、RAC 和 Oracle Applications 的 OCP 提

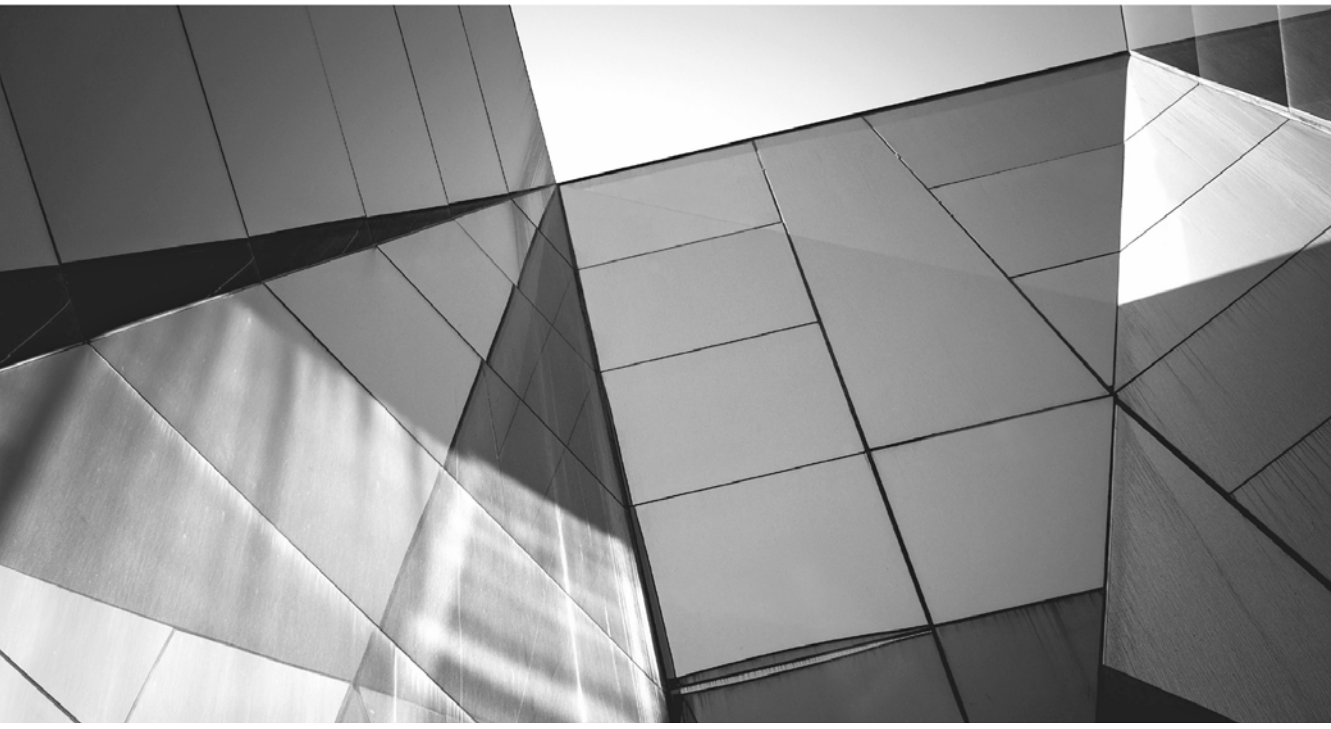
供问题建议。

Joyjeet 拥有系统和运营管理 MBA 学位，最近也被授予 Oracle Innovator 称号。(www.oracle.com/us/products/applications/ebusiness/conversations-with-innovators-070240.html)



技术编辑简介

Deba Chatterjee 是 Robin Systems 的产品总监。在担任现职之前，他是 Oracle 公司的高级首席产品经理，负责 Oracle Multitenant 选件以及 Oracle Diagnostics & Tuning 软件包。在从事产品管理工作之前，Deba 曾在 Oracle Product Development IT 的性能服务团队工作，负责管理大型数据仓库的性能。他之前曾在 Oracle 咨询部门、甲骨文印度公司、法国 Clermont-Ferrand 的 Michelin Tires 和 Tata 咨询服务公司工作过。Deba 获得技术管理硕士学位，拥有一个宾州工程与沃顿商学院的联合项目。



致　　谢

有很多人为本书的成功出版做出了巨大的努力。我想借此机会对他们致以感谢。

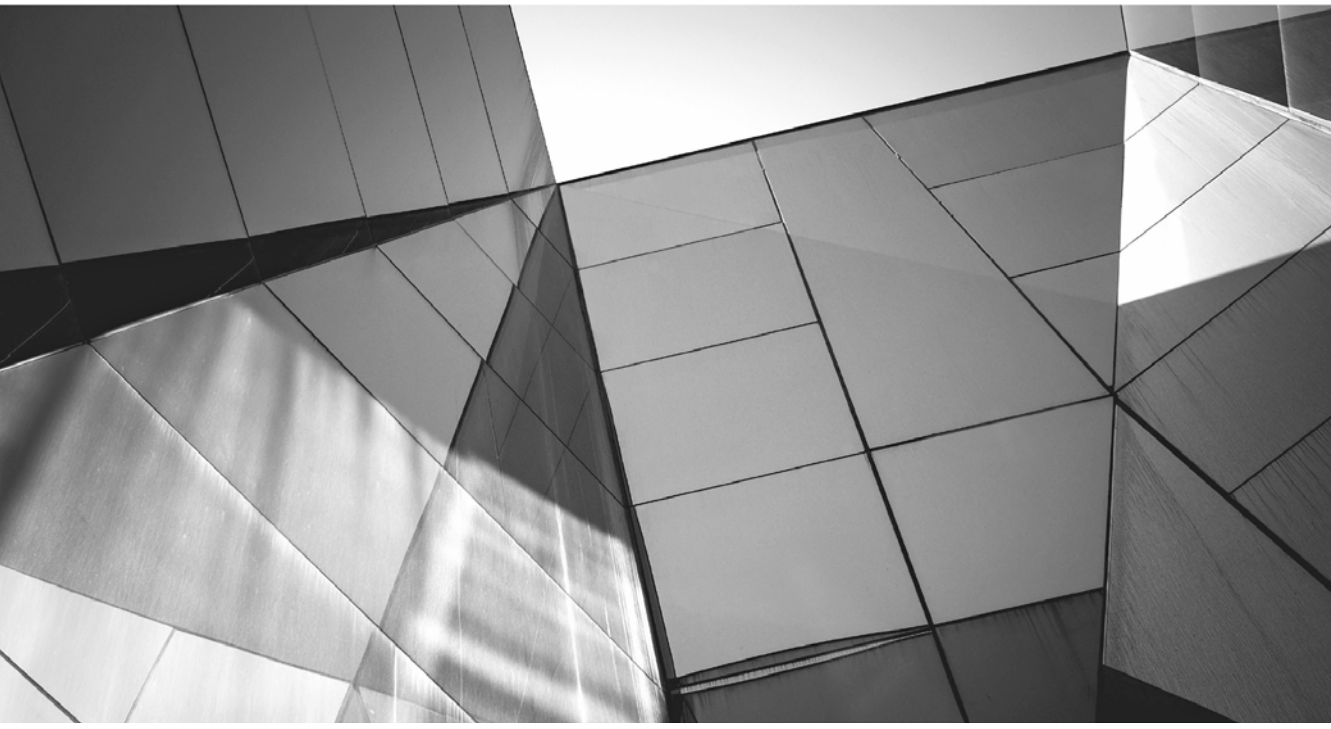
没有他们的帮助，我很难完成这本书。

我要感谢 McGraw-Hill 出版社的编辑部主任 Wendy Rinaldi，感谢她的热情支持和鼓励，她以各种可能的方式帮助我，使这本书顺利出版。McGraw-Hill 出版社的组稿协调员 Claire Yee 负责管理本项目。Cenveo Publisher Services 的 Anubhav Singh 在本书制作的各个阶段都给我提供了支持。Lisa Theobald 编辑了本书，Lisa McCoy 对本书进行了校对。

我要感谢 Robin Systems 的产品总监 Deba Chatterjee 对本书所做的技术评审，感谢他提供的宝贵建议和反馈。我也要感谢 Oracle Database In-Memory 产品选件的产品经理 Maria Colgan，她审核了本书的核心章节并提供了宝贵的意

见。我要感谢数据库服务器技术部门(Database Server Technologies)的执行副总裁 Andy Mendelsohn 快速批准这个项目；感谢 Sheila Cepero 紧随着我的步伐非常迅速地获得了所有必要的批文；感谢管理顾问 Todd Alder 以及 Oracle 法律团队的 Mark Schreier 提供了所有必要的许可。

最后我要感谢所有的朋友和同事，他们无一例外地激励并鼓励我，使本书问世成为现实。



前　言

现代商业运作的模式已经发生了转变。企业希望能够找到帮助他们快速地、轻松地做出有效决策的工具。通常，为了快速应对需求的增长，除了数据仓库外，公司还直接在其交易型系统上运行分析。这导致了在事务型工作负载(需要频繁插入和更新)和需要扫描大量数据的报表类查询之间的不确定性操作。

借助 Oracle Database In-Memory(Oracle 内存数据库)，单个数据库现在就可以有效地支持混合工作负载，为事务型操作提供最佳性能的同时，还支持实时分析和报表。用 Oracle Database In-Memory 独特的双模结构实现这一点是本书的精髓所在。双模结构即以行格式维护 OLTP 操作的数据，以 In-Memory(内存)列格式维护分析处理的数据。

Oracle Database 的 In-Memory 选件与现有应用程序兼容，并可使用 Oracle Database 中所有现有的功能，而且在不改变应用程序的情况下使用。

通过本书，你将学习并练习以下操作：

- 配置 Oracle Database 12*c* 并构建启用 In-Memory 的数据库。
- 在图形化界面上编辑和控制 In-Memory 选件。
- 利用 Oracle Real Application Cluster 部署 In-Memory。
- 使用 In-Memory Advisor 确定哪些对象要保存在内存中。
- 使用组、表达式和聚合来优化 In-Memory 查询。
- 使用 Oracle Exadata Database Machine 和 In-Memory 选件使性能达到最优。
- 使用 Swingbench 创建数据并模拟真实系统的工作负载。

本书包括以下 8 章内容。

第 1 章“数据库体系架构”　在该章中，你将了解 Oracle Database 体系结构，将学习 Oracle Database 中所包含的组件，将了解 Oracle Database 引擎的工作原理并了解所有的后台进程。如果你不熟悉 Oracle Database，该章提供了一个快速入门。如果你已熟悉 Oracle Database，该章将有助于你复习其功能和体系结构。

第 2 章“In-Memory 体系架构”　该章介绍 In-Memory 选件及其体系结构，讨论运行 Database In-Memory 的益处，并说明它的工作原理以及数据在内存中的存储和处理方式。你还将了解 In-Memory 所涉及的所有后台进程。

第 3 章“部署 In-Memory 选件”　在该章中，你将学习如何在数据库中部署 In-Memory 功能。你将分配 In-Memory 工作区并学习与 In-Memory 相关的所有初始化参数，如何在 In-Memory 中加载数据以及如何为各种数据库对象(诸如表、列、物化视图等)启用 In-Memory。

第 4 章“Database In-Memory 与 RAC 和 Multitenant 协同工作的方式”　在该章中，你将了解 Oracle Real Application Clusters，学习并行查询的工作原理，了解在 RAC 系统上运行 In-Memory 的要求，学习在 RAC 中是如何加载数据的，以及 In-Memory 是如何与 RAC 和 Multitenant 协同工作的。

第 5 章“Database In-Memory Advisor”　在该章中，你将学习如何使用 In-Memory Advisor 来决定哪些对象最好保存在内存中。你将学习如何安装和运行 In-Memory Advisor，并学习各种 In-Memory Advisor 的方法。本章还提供了一个运行 In-Memory Advisor、实施 Advisor 建议和比较结果的示例。

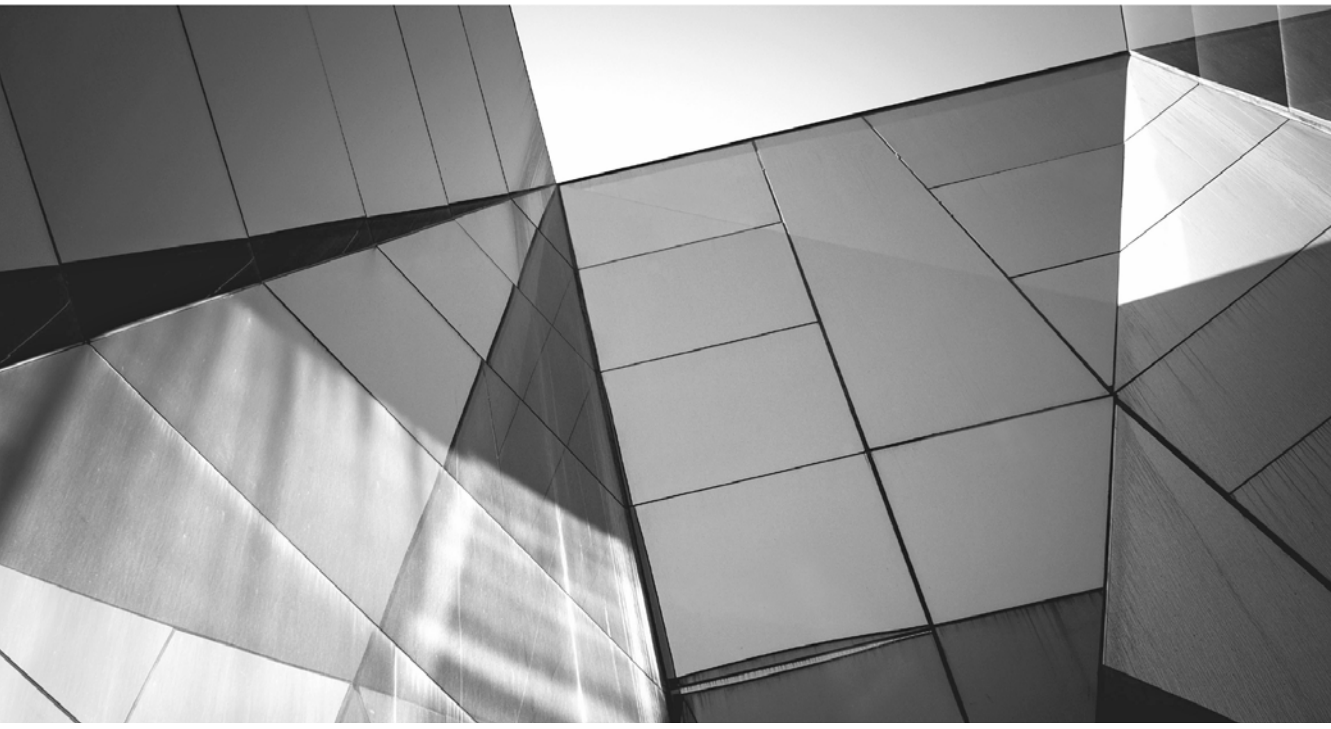
第 6 章“Database In-Memory 查询优化”　在该章中你将学习如何优化 In-Memory 查询以获得最大的性能提升。你将学习如何使用 join group、In-Memory 表达式和 In-Memory 聚合操作。

第 7 章“In-Memory 和工程化系统”　在该章中，你将学习 Exadata 的架

构和功能、Exadata 的优势，在 Exadata 中运行 In-Memory 的益处以及如何在 Exadata 上运行 In-Memory。

第 8 章“In-Memory 动手实验” 在该章中，你将使用 In-Memory 功能。该章包含用于部署 In-Memory 功能的示例和场景。你将使用 Swingbench 工具在自己的数据库中生成数据，并将针对该数据集运行与 In-Memory 相关的各种查询。你将获得对 In-Memory 足够的了解，以便在自己的工作区中部署它。

附录 A“安装 Oracle Database 和启用 In-Memory” 附录 A 介绍了 Oracle Database 的完整安装过程，涵盖了 Oracle 软件的使用和 In-Memory 选件的启动。



目　　录

第 1 章　数据库体系结构……1
1.1　Oracle 数据库与实例……2
1.2　数据库的存储结构……3
1.2.1　物理结构……3
1.2.2　逻辑结构……11
1.3　内存结构……13
1.3.1　系统全局区(SGA)……13
1.3.2　程序全局区(PGA)……17
1.4　进程结构……17
1.4.1　Process Monitor (PMON)……18
1.4.2　System Monitor (SMON)……18
1.4.3　数据库写进程 (DBW*n*)……18
1.4.4　日志写进程(LGWR)…18
1.4.5　Checkpoint(CKPT)……19
1.4.6　Recoverer(RECO)……19
1.4.7　其他进程……19
1.5　本章小结……25
第 2 章　In-Memory 体系架构……27
2.1　行处理与列处理……30

2.2 Database In-Memory 体系结构……31
2.3 GET_COMPRESSION_RATIO 过程……38
2.3.1 语法……38
2.3.2 SIMD 矢量处理……42
2.4 将数据加载到 In-Memory 列存储中……42
2.4.1 IM FastStart……44
2.4.2 具有 ADO(自动数据优化)功能的 In-Memory……45
2.4.3 条件限制和用法……46
2.5 本章小结……48

第 3 章 部署 In-Memory 选件……51
3.1 分配 In-Memory Area……52
3.2 In-Memory 的初始化参数……54
3.3 向 IMCS 中加载数据……60
3.4 在数据库级别启用 IMCS……62
3.5 针对表启用/禁用 IMCS……63
3.6 针对虚拟列启用/禁用 IMCS……65
3.7 针对表空间启用/禁用 IMCS……65
3.8 针对物化视图启用/禁用 IMCS……66
3.9 启用 IM FastStart……66
3.10 启用 ADO 功能和 In-Memory 选件……67
3.11 IMCS 和 Data Pump……68
3.12 不能被加载到 IMCS 中的对象……69
3.13 In-Memory 视图……69
3.14 本章小结……70

第 4 章 Database In-Memory 与 RAC 和 Multitenant 协同工作的方式……73
4.1 RAC 基础介绍……74
4.1.1 RAC 架构及概念……74
4.1.2 RAC 后台进程……78
4.2 利用 RAC 执行并行操作……79
4.3 RAC 对内存的要求……84
4.4 RAC 和 DBIM……85
4.4.1 在 RAC 环境下将数据加载到 IMCS 中……86
4.4.2 IMCS 数据和服务……87
4.5 Multitenant 和 IMDB……89
4.6 本章小结……90

第 5 章 Database In-Memory Advisor……91
5.1 安装 In-Memory Advisor……92
5.2 运行 In-Memory Advisor……95
5.3 实施 In-Memory Advisor 建议并比较结果……104
5.4 本章小结……109

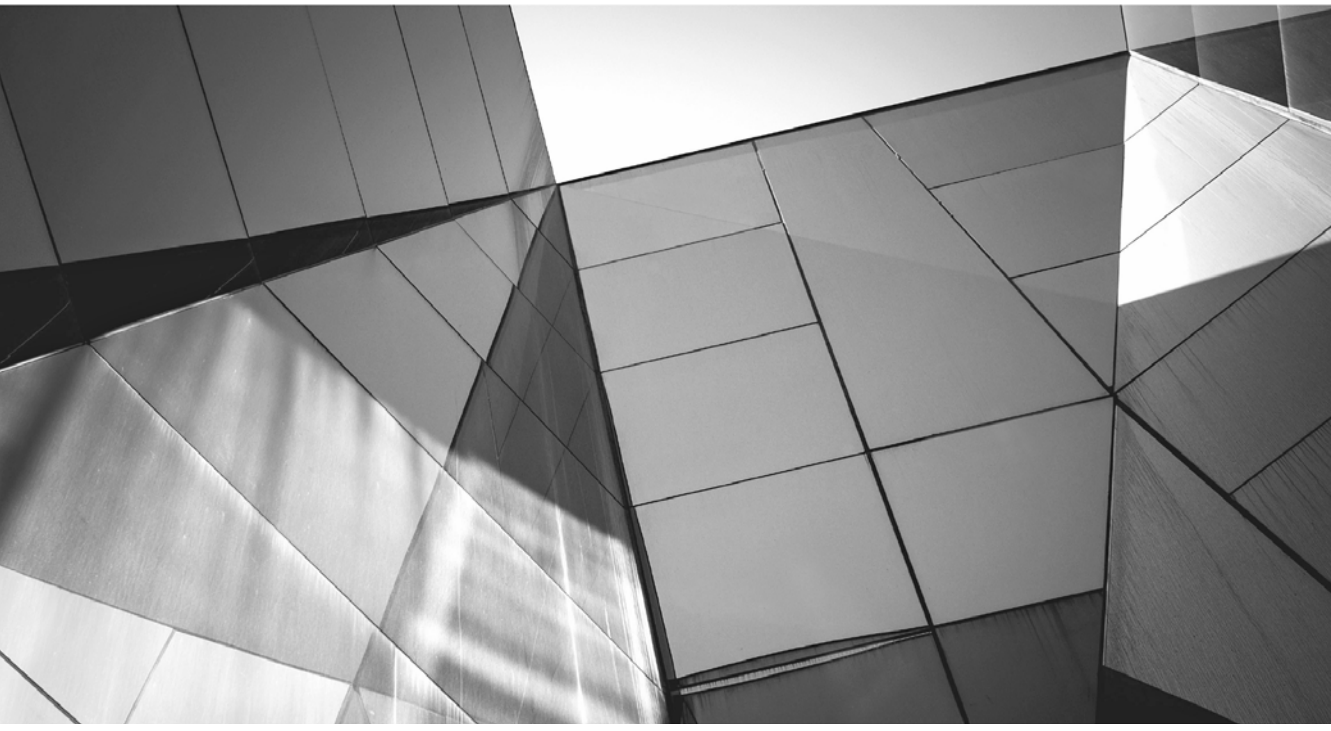
第 6 章 Database In-Memory 查询优化……111
6.1 使用连接组……111
6.2 使用 IM 表达式……115

6.3 使用 In-Memory 聚合 … 119
6.4 本章小结 …… 122

第 7 章 In-Memory 和工程化系统 ……125
7.1 Exadata 的优点 …… 126
7.2 Exadata 体系结构 …… 127
7.2.1 数据库服务器或计算节点 …… 128
7.2.2 Exadata 存储单元或 Exadata 存储服务器 …… 129
7.2.3 InfiniBand …… 130
7.3 Exadata 硬件配置 …… 130
7.4 Exadata 特性 …… 131
7.4.1 混合列压缩 …… 131
7.4.2 HCC 和 Database In-Memory 的优势 …. 134
7.5 Smart Scan 与传统架构 …… 135
7.6 Exadata Smart Flash Cache …… 139
7.7 本章小结 …… 141

第 8 章 In-Memory 动手实验 …. 143
8.1 查看服务器内存 …… 144
8.2 使用 SwingBench 加载数据 …… 148
8.2.1 为销售历史生成数据 …… 153
8.2.2 检查内存相关参数 …… 154
8.2.3 启动数据填充 …… 155
8.2.4 创建物化视图 …… 158
8.3 编辑对象的 In-Memory 属性和设置 …… 159
8.4 禁用 In-Memory 选件 … 164
8.5 安装 In-Memory Advisor …… 166
8.6 本章小结 …… 170

附录 A 安装 Oracle Database 和启用 In-Memory …… 171
A.1 预安装步骤 …… 172
A.1.1 临时文件系统 …… 173
A.1.2 根据 RAM 设置 SWAP 空间 …… 173
A.1.3 Oracle Inventory Directory …… 173
A.1.4 用户和用户组 …… 174
A.1.5 环境变量 …… 175
A.1.6 root 用户访问 …… 175
A.1.7 Oracle 软件目录 …. 176
A.1.8 Grid Infrastructure (Conditional) …… 176
A.1.9 分享软件 …… 176
A.2 运行 Oracle Universal Installer …… 177
A.3 安装后步骤 …… 192
A.4 启用 In-Memory 选件 …. 193
A.5 本章小结 …… 194



第1章

数据库体系结构

对于任何组织而言，数据都是最关键的资产。随着公司不断扩大，对于信息存储和访问的需求呈指数级增长，而且在未来对于数据存储和检索的需求也不会停止。到目前为止，绝大多数组织已经将文件和文档存储在数据库中了。数据库的易用性和高效率使其成功地成为许多公司重要数据的存储库。数据库不仅提供了一个信息存储的地方，而且更重要的是，如果数据被合理地组织，数据库则支持高效的、有价值的信息检索。

数据库管理系统(Database Management System，DBMS)是管理数据存储、结构和检索的软件。按照 Codd 给出的定义规则，关系数据库管理系统(Relational

Database Management System，RDBMS)是适用于具有良好规范的对象存储或结构的关系模型。这些存储和结构就是大家所熟知的运算符和完整性规则，它们被清晰地定义为用来管理和控制数据库中的数据和结构的各种操作。

注意：按照数据库关系模型的鼻祖 Edgar F. Codd 的定义，任何 DBMS 要成为 RDBMS，必须遵循 12 条黄金法则，通常称为 Codd 12 条法则。如果想了解有关 RDBMS 的更多 Codd 法则，可以访问 http://dl.acm.org/citation.cfm?id=362685。

1979 年，Oracle 推出了第一个商用版的 RDBMS 数据库——Oracle V2(版本 2)。Oracle 数据库通过不断的创新和演进，已经远远地超越了 RDBMS 的功能。在每一个新版本中，Oracle 都会推出新的特性。1999 年，Oracle 发布了 Oracle8*i*(*i* 代表 Internet)，这是一个被设计用来在多层环境中部署并支持互联网计算的数据库。2001 年，Oracle9*i* 推出了一项创新技术——Oracle Real Application Clusters(RAC，真正应用集群)。RAC 能够支持从多个实例同时访问同一个数据库，使数据库具备高可用性。2003 年，Oracle 10*g*(*g* 代表 grid)推出了网格计算(grid computing)架构，强调用一组廉价的商用服务器(网格架构)构建一个集群，其上运行 Oracle 数据库。Oracle11*g* 推出了 Active Data Guard 选件，该选件支持备用数据库以只读方式打开并为报表提供服务。

Oracle Database 12*c* 将数据库带入了云时代。Oracle12*c* 推出了两项开创性的新特性：

- **数据库多租户架构** Oracle 数据库具有多租户容器数据库(Container Database，CDB)的功能，CDB 可以只包含一个可插拔数据库 (Pluggable Database，PDB)，也可以包含多个可插拔数据库。
- **Database In-Memory(内存数据库)** 这是本书的重点，Oracle Database In-Memory 能够透明地、成倍地加速分析型查询的响应速度，实现实时的商业决策，同时在混合负载的环境里也能加速操作型事务处理。将内存数据库的功能嵌入现有的 Oracle 数据库软件中，能够确保充分地兼容已有的各种特性，同时确保对应用层不做任何更改。

在我们直接进入 Oracle Database In-Memory 的深入讨论之前，先了解 Oracle 数据库的体系结构。

1.1 Oracle 数据库与实例

数据库是由存储了整个数据的所有物理文件和实例构成。一个实例包含了一组内存结构和管理物理文件的后台进程。而用来存储数据的物理文件有时也被称作数据库。

Oracle 数据库服务器 = 物理文件+实例

或者

Oracle 数据库服务器 = 数据库(物理文件)+实例

一个数据库由以下三种主要结构组成：

- 存储结构
- 内存结构
- 进程结构

存储结构由物理存储(或物理文件)和逻辑存储(数据的逻辑分布)组成。逻辑存储以表、索引和表空间等形式存在，主要用于高效地管理物理结构。

尽管存储结构是数据库的组成部分，但是内存结构和进程结构却是实例的组成部分。

一个数据库可以有一个或多个实例。只有一个实例的数据库被称为单实例数据库。如果数据库有多个实例并且所有实例同时访问相同的一个数据库，则该数据库被称为 RAC 数据库。默认情况下，一个 RAC 数据库总是处于 active-active 状态。如果两个数据库实例被配置成只有一个节点是在任何时间处于 active 状态，则该数据库被称为 active-passive 配置。Oracle 数据库支持 active-active 和 active-passive 配置。Oracle Database 12*c* 也引入了 CDB 和 PDB 架构，在后续章节你将会学习到相关的内容。

一个实例在任何时间只能访问一个数据库。尽管一个数据库可以有多个实例(RAC)，但一个实例在同一时间不能访问多个数据库。因此，在特定的时间内，一个实例只能对应一个数据库。一个实例能够独立于数据库而存在，反之亦然。通常，当使用 create database 命令创建数据库时，实例已经存在，而数据库并不存在。同样，当备份数据库时，只是对构成数据库的物理文件执行操作而不是对实例操作。为了有一个正常运行的数据库，需要同时拥有数据库和实例。

1.2　数据库的存储结构

如前所述，数据库的存储结构由物理存储和逻辑存储构成。Oracle 对物理存储结构和逻辑存储结构分别进行管理。这些存储结构独立工作，对其中的一个进行管理并不会影响到另一个。例如，如果你重新命名一个表，则包含该表的数据库文件并不会被重新命名。

1.2.1　物理结构

数据库的物理结构由下列文件组成：

- database files (数据库文件)
- control files (控制文件)
- online redo log files (在线重做日志文件)
- archived redo log files (归档重做日志文件)
- parameter file (参数文件)
- password file (口令文件)
- alert log file, trace files, dumps and core files (告警日志文件、跟踪文件、dumps 和 core 文件)
- backup files (备份文件)

1. 数据库文件

数据库文件通常被称为 datafile，它包含了所有数据。每个数据库应该至少有一个 datafile，用于数据的存储；如果没有 datafile，则数据库就没有任何意义。datafile 是 Oracle 数据库创建的一个物理文件。除了存储数据，datafile 也包含了数据结构，诸如表、索引和物化视图。datafile 中的数据以二进制格式保存，如果没有 Oracle 数据库的存在，datafile 是不能被读取的——因此，如果使用 vi 或一些其他类型的编辑器打开 datafile，则不能读取其中的数据。

每个表空间对应一个或多个 datafile。因此，如果数据库有 10 个表空间，那么它至少会有 10 个 datafile。一个 datafile 只能对应一个表空间。

通常，datafile 文件使用 .dbf(数据库文件)作为扩展名。datafile 能够被存储在 SAN(Storage Area Network，存储区域网络)、NAS(Network Attached Storage，网络连接式存储)、文件系统、NFS(Network File System，网络文件系统)或 Oracle ASM(Automatic Storage Management，自动存储管理)中。Oracle 推荐使用 ASM 存储 datafile，因为 ASM 管理 datafile 不需要人工干预。而且，ASM 支持对 datafile 的条带化和镜像操作。条带化通过将数据分散到多个磁盘实现负载平衡，从而提供了更好的性能。镜像保护了数据、避免故障发生。在 ASM 中，Oracle Managed Files (OMF)的功能简化了数据库创建和 datafile 命名的操作，因此不需要 DBA 手工执行这些操作。

datafile 文件中的第一个块存储了块头(block header)信息，它包含了有关 datafile 文件的详细信息、datafile 文件的个数、datafile 文件是 offline 还是 online 状态、在 datafile 文件中有多少可用空间和空闲空间是可以使用的，以及 datafile 文件检查点和恢复点的信息。

临时文件(.tmp)是数据库文件的一种类型，它被用于临时操作，例如执行查询时的排序操作。这些 tempfile 中的数据被临时地存储，并在查询完成后被丢弃。在数据库中，tempfile 中的空间能够被其他操作所重用。

能够从数据字典视图 V$DATAFILE 中查询 datafile 文件，能够从数据字典

视图 V$TEMPFILE 中查询 tempfile 文件：

```
SQL> select name from v$datafile ;

NAME
------------------------------------------------------------
/u01/app/oracle/oradata/orcl/system01.dbf
/u01/app/oracle/oradata/orcl/ts_data.dbf
/u01/app/oracle/oradata/orcl/sysaux01.dbf
/u01/app/oracle/oradata/orcl/undotbs01.dbf
/u01/app/oracle/oradata/orcl/example01.dbf
/u01/app/oracle/oradata/orcl/users01.dbf
/u01/app/oracle/oradata/orcl/ts_data2.dbf
/u01/app/oracle/oradata/orcl/sh1.dbf
/u01/app/oracle/oradata/orcl/enc128_ts0.dbf

9 rows selected
SQL> Select name from v$tempfile;

NAME
--------------------------------------------------------------------------
/u01/app/oracle/oradata/orcl/temp01.dbf
```

2. 控制文件

与 datafile 文件不同，控制文件(control file)并不包含任何表、索引或者物化视图的数据；相反，它们包含了数据库的最关键信息。由于控制文件是非常重要的，它们通常会被镜像和多重复制。如果一个控制文件出现了问题，Oracle 会从镜像拷贝中重新读取。控制文件的内容不会被存储在其他地方，甚至不会在 datafile 文件中存储，因此，保护好控制文件变得非常重要。控制文件包含下列信息：

- 数据库被创建时的名称、日期和时间，以及数据库唯一识别号
- datafile 和重做日志文件(redo log file)的详细信息，包括文件的大小、位置和文件的个数
- Recovery Manager (RMAN)备份的信息
- 表空间的详细信息
- 重做日志文件(redo log file)、归档日志文件(archived log file)和日志历史的信息
- 在事件失败时用来恢复数据库的所有日志的详细信息
- 检查点的详细信息
- 恢复点的详细信息

不管何时数据库中有结构化的变更，诸如增加一个 datafile 文件或调整重做日志文件的大小，这些变更都会在控制文件中更新。通常控制文件的扩展名为.ctl 或.con。控制文件的位置由初始化参数文件中 control_files 参数指定。因

为控制文件总是被多重复制，不管何时文件被更新，Oracle 的同步进程都会更新所有的控制文件副本，以确保所有的控制文件是在同步状态。更新控制文件的后台进程被称为 checkpoint(CKPT)。Oracle 推荐创建多个控制文件的副本并将其部署在不同的磁盘上，以避免任何磁盘发生故障。

控制文件由多个部分组成，各个部分存储了数据库的相关信息。每个部分包含了数据库特定区域的一组记录，诸如 datafile 文件的记录跟踪等。控制文件包含下列两种类型的记录：

- **循环可重用记录** 这部分记录包含了非核心信息，能够根据需要被重写。例如，无论何时使用 RMAN 执行一个新的备份，以前的备份记录就会被重写，或者当归档重做日志文件成为旧文档后，信息就会被重写。
- **非循环可重用记录** 这部分记录包含了关键信息，不能被重写。它所存储的大部分信息很少会发生变更——例如，datafile 文件、表空间、重做日志文件等的信息。在某些特定情况下，它可能也会被重用，例如，当 datafile 文件被删除后。

控制文件能够从数据字典视图 V$CONTROLFILE 查询到：

```
SQL> select name from v$controlfile ;

NAME
--------------------------------------------------------------------------------
/u01/app/oracle/oradata/orcl/control01.ctl
/u02/app/oracle/oradata/orcl/control02.ctl
/u03/app/oracle/oradata/orcl/control02.ctl
```

提示与技巧：

视图 V$CONTROLFILE_RECORD_SECTION 包含了控制文件记录部分的信息。

3. 联机重做日志文件

联机重做日志文件(online redo log file)通常简称为日志文件，记录了数据库中发生的任何变更。重做日志文件的重要作用是保护数据库不会丢失数据。重做日志文件也会捕获 undo 变更信息，因此，也会对 undo 数据进行保护。数据库中的两个或多个重做日志文件能够捕获所有的数据变更。

重做日志文件(redo log file)包含了重做记录(redo records)或重做条目(redo entries)，而重做记录或重做条目由一组变更向量构成。每个向量记录了对数据库中一个块所做的变更。除了变更向量之外，重做日志文件包含了详细的变更信息，诸如时间戳、系统变更号(SCN)、事务 ID 号、提交或未提交事务、变更操作以及数据段(data segment)类型。

Log Writer (LGWR)进程以循环方式对重做日志文件执行写操作。首先对第一个日志文件执行写操作，当第一个文件被写满后，开始对下一个文件执行写

操作，并以这种循环方式执行写操作，直到最后一个文件。当最后一个重做日志文件被写满后，进程返回到第一个重做日志文件，并再次执行写操作。该进程重用已有的日志文件，同时覆盖掉原有的内容。当然，在数据被覆盖掉之前，数据库写进程会确保数据已被写入数据文件中。一个被写满的日志文件在所有变更已经被写入数据文件后是可以被重用的。如果归档已经被激活，一个被写满的日志文件是可以被重用的，前提条件是在所有变更已经被写入数据文件并且该文件已经被归档——也就是说，被写到归档日志文件中了。下一节将讨论归档日志文件。

无论何时，当一个事务被提交时，LGWR 都会分配一个系统变更号(SCN)来标识该事务，然后将 SCN 写入数据文件。

每一时刻只有一个重做日志处于活动状态。LGWR 进程正在写入的那个活跃的重做日志文件是当前(current)的重做日志文件。那些在实例恢复时所需的重做日志文件就是 active 的重做日志文件。实例恢复时不需要的重做日志文件就是 inactive 的重做日志文件。

针对特定数据库实例的重做日志文件是重做线程(redo thread)。在单实例数据库中，只有一个重做日志线程存在。对于 RAC 数据库，每个实例都会有一个单独的重做日志线程，所以一个四节点的 RAC 数据库将有四个不同的重做日志线程。

如前所述，数据库应该至少有两个重做日志文件，以便总有一个文件始终可用于写入。当数据库停止写入一个重做日志文件并开始写入下一个时，该过程称为日志切换。日志切换通常在当前重做日志文件已满而又必须连续写入时发生。此外，可将日志切换配置为以固定的时间间隔进行，而与所写入的重做日志文件无关。

在发生日志切换时，数据库为该重做日志文件分配一个新的日志序列号，之后 LGWR 便开始写入该日志文件。每个重做日志文件都有一个唯一的日志序列号。当日志文件被覆盖时，将为新写入的日志文件分配一个新的唯一日志序列号。当重做日志文件被写入归档重做日志文件时，相同的日志序列号会保留在归档重做日志文件中。日志序列号可以在实例恢复或故障期间恢复数据库。

与控制文件类似，重做日志文件也会被重用或镜像，以防止故障。LGWR 会同时写入重做日志文件的所有副本，消除了单点故障的可能性。通过创建重做日志文件组来对日志文件进行镜像；一个重做日志文件组包含重做日志文件及其所有镜像副本。作为日志文件组的一部分，每个重做日志文件都被称为一个成员。例如，如果有两个日志组，每个组中有两个成员，则数据库中的日志文件总数为四个。

我们来看一个示例。一个数据库有两个日志组，A 和 B。每个日志组有两个成员——重做日志组 A 包含成员 A1 和 A2，重做日志组 B 包含成员 B1 和

B2。当 LGWR 在日志组 A 中写入时，它同时写入 A1 和 A2，每当有日志切换时，它就开始写入日志组 B，同时写入 B1 和 B2。

提示与技巧：

同一个日志组的成员应该存储在不同的磁盘上，以避免磁盘故障引发的问题。

4. 归档重做日志文件

归档重做日志文件只是重做日志文件的副本。归档日志文件不应与重做日志文件的镜像或副本混淆，后者也称为重做日志文件。归档日志文件由 Archiver (ARC*n*)进程执行写操作，该进程将重做日志文件的内容复制到归档日志文件中，以便将数据归档。归档日志文件在启用了归档进程且设置数据库以 archivelog mode 运行时创建。当数据库运行于 noarchivelog mode 时，归档进程被禁用，并且不会创建归档重做日志文件。重做日志文件和归档日志文件的日志序列号相同，因为相同的数据存在于两个文件中。

归档重做日志文件为数据提供了额外的保护。虽然归档重做日志文件在 CPU、I/O 和存储方面有一些开销，但它们提供的好处使得这些开销可以忽略不计。在发生磁盘故障时，利用重做日志文件和归档日志文件连同已有的数据库全备份一起，能够恢复数据库中所有已提交的事务。以 archivelog mode 启动数据库还能提供联机备份(online backup)——也就是说，可以在数据库启动并运行时对其进行备份。归档重做日志文件还能用来保持备用数据库与生产数据库的一致，因为归档重做日志文件能被传输到备用数据库中。

与重做日志文件和控制文件一样，可以保存多个归档重做日志文件的副本。这种情况下需要考虑的一个问题是空间管理。

提示与技巧：

在生产环境中，将数据库配置为 archivelog mode 模式是有意义的。对于诸如测试环境之类的非生产环境，通常是不需要对补丁归档。然而，有时非生产系统，如生产支持或用户测试环境，可能仍需要打开归档功能。

5. 参数文件

在启动数据库之前，Oracle 数据库使用参数文件读取所有参数。该文件包含与数据库功能相关的各种参数。初始化参数设置了对整个数据库、用户或进程或某一特定的数据库资源的限制。DBA 使用此文件来控制数据库，给数据库指定更多内存或限制数据库中的 CPU 使用率。DBA 还可以使用此文件来控制最大用户数量，设置阈值限制，并启用审计或跟踪。

Oracle 数据库有数以百计的参数可被使用，它们分为两种类型：基本的和可选的。基本的参数是强制性的，如果没有它们，数据库将无法启动。可选的

参数提供对数据库的更多控制，但它们不是强制性的。

Oracle 数据库支持两种类型的参数文件：

- 初始化参数文件
- 服务器参数文件

初始化参数文件是包含所有参数的纯文本文件。这个文件通常被称为 PFILE 或 init.ora 文件。该文件包含参数名称及与之对应的参数值。该文件中的条目不区分大小写。该文件的默认位置是$ ORACLE_HOME / dbs 目录。用于该文件的命名规则是 init <*DB_ NAME*> .ora——例如，如果数据库名称是 DBIM，则该文件名将是 initDBIM.ora。

服务器参数文件是初始化参数文件的二进制版本。它的名称为 SPFILE。此文件不能直接编辑，但可以利用 SPFILE 创建一个 PFILE，对其进行编辑，然后将其转换为 SPFILE。SPFILE 也位于$ORACLE_HOME / dbs 目录中。SPFILE 的命名规则与 PFILE 相似：spfile <DB_NAME> .ora。因此，如果数据库名称是 DBIM，那么 SPFILE 的名称将是 spfileDBIM.ora。

下面是 init.ora 参数文件的一个示例：

```
orcl.__data_transfer_cache_size=0
orcl.__db_cache_size=3238002688
orcl.__java_pool_size=16777216
orcl.__large_pool_size=33554432
orcl.__oracle_base='/u01/app/oracle'#ORACLE_BASE
orcl.__pga_aggregate_target=1073741824
orcl.__sga_target=3758096384
orcl.__shared_io_pool_size=67108864
orcl.__shared_pool_size=352321536
orcl.__streams_pool_size=33554432
*.audit_file_dest='/u01/app/oracle/admin/orcl/adump'
*.audit_trail='db'
*.compatible='12.1.0.2.0'
*.control_files='/u01/app/oracle/oradata/orcl/control01.ctl',
'/u02/app/oracle/oradata/orcl/control02.ctl',
'/u03/app/oracle/oradata/orcl/control01.ctl'
*.db_block_size=8192
*.db_domain=''
*.db_name='orcl'
*.diagnostic_dest='/u01/app/oracle'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=orclXDB)'
*.inmemory_size=0
*.job_queue_processes=8
*.local_listener='LISTENER_ORCL'
*.memory_max_target=0
*.memory_target=0
*.open_cursors=300
*.pga_aggregate_limit=2147483648
*.pga_aggregate_target=1073741824
*.processes=300
```

```
*.remote_login_passwordfile='EXCLUSIVE'
*.resource_manager_plan='DEFAULT_PLAN'
*.sga_max_size=3758096384
*.sga_target=3758096384
*.undo_tablespace='UNDOTBS1'
```

6. 口令文件

口令文件使用户能够以管理员的身份远程登录到数据库。用户可以使用口令文件作为 SYS、SYSDBA、SYSOPER、SYSBACKUP、SYSDG 和 SYSKM 用户执行连接。口令文件是在运行数据库配置助手(Database Configuration Assistant，DBCA)时创建的。可以随时使用 ORAPWD 实用程序创建或修改它。口令文件的位置是在$ORACLE_HOME/dbs 目录中。口令文件中的密码是区分大小写的。

提示与技巧：

操作系统的身份验证高于口令文件的身份验证。口令文件的身份验证主要是在远程连接数据库时使用，而操作系统的身份验证是从数据库服务器直接登录时使用。

7. 告警日志文件、跟踪文件、导出和内核文件

数据库的告警日志文件按时间顺序记录了数据库中发生的所有错误和消息。在数据库发生了任何问题的情况下，这里是所有信息写入的地方。DBA 可以使用此工具分析数据库中发生的任何问题。告警日志包含有关启动/停止的信息、有关创建或删除某个对象的信息、数据库正在经历的任何 ORA 错误以及所有其他的数据库事件。

告警日志文件被写为一个文本文件和一个 XML 文件。这两个文件都位于 ADR Home 目录中。自动诊断信息库(Automatic Diagnostic Repository，ADR)是各种日志的文件系统信息库。每个实例都有自己的 ADR 目录。Oracle 建议通过云控制或自动诊断信息库命令解释器(Automatic Diagnostic Repository Command Interpreter，ADRCI)实用程序，并利用 XML 版本的告警日志文件进行监控，而告警日志文件的文本版本则用于向后兼容。

ADR 的根目录，包括 ADR Home 目录，是 ADR 的基础。通过设置参数 DIAGNOSTIC_DEST 可指定 ADR 在初始化参数文件中的位置。如果此参数未在 PFILE/SPFILE 中设置，则 DIAGNOSTIC_DEST 将被设置为由$ORACLE_BASE 指定的目录。如果连$ORACLE_BASE 也没有设置，则 DIAGNOSTIC_DEST 将被设置为$ORACLE_HOME/log。

每个 Oracle 数据库后台进程都会在自己的跟踪文件(.trc)中记录与其相关的信息。与错误、状态、告警和进程相关的信息都被写入跟踪文件。数据库后台进程的跟踪文件使用 ORACLE_SID 和 OS 进程 ID 进行命名。例如，对于

ORACLE_SID 为 DBIM、进程 ID 为 2239 的数据库，Oracle 后台进程跟踪文件名将为 dbim_pmon_2239.trc。服务器进程的跟踪文件名包括 ORACLE_SID、字符串 *ora* 和进程 ID。因此，例如，对于 ORACLE_SID 为 DBIM、进程 ID 为 2245 的数据库，跟踪文件名将为 dbim_ora_2245.trc。

跟踪文件有时会附带扩展名为.trm 的跟踪元数据文件，这些文件包含了关于跟踪文件的结构信息。

dump(导出)文件是一种基于事故/事件的跟踪文件。它通常是关于某特定事件发生的诊断数据的一次性输出。导出文件被写入事件目录中。每个事件都包含一个事件编号(incident number)。

内核文件是内存 dump 文件。这个文件是用二进制格式写的。它主要由 Oracle Support 使用。文件名包含字符串"core"和操作系统进程 ID。

提示与技巧

要查找所有跟踪、导出和日志目录的详细路径名，请运行以下查询：

```
select * from v$diag_info
```

8. 备份文件

备份文件是一些物理文件，并不用于数据库的功能实现。通常在以下情况下使用它们：

- 如果数据库、表或对象需要被恢复
- 如果想要将数据库恢复到某个特定的时间点
- 如果想对现有的数据库进行克隆操作
- 如果想创建一个备用数据库

备份文件可以是包含数据文件、联机重做日志文件、归档重做日志文件、SPFILE/PFILE 和控制文件的 RMAN 备份文件。如果不使用 RMAN，备份文件将包含所有这些文件而不考虑备份方法。如果备份是在存储级别或使用快照方法进行的，则备份文件将是使用该方法时的输出文件。备份文件还包含一个 ORACLE_HOME 的备份。如果备份是通过 export 导出的，则备份文件将是 export dump 文件。

1.2.2　逻辑结构

数据库的逻辑结构使数据得到更好的控制，以便对之进行处理、存储和检索。逻辑结构的四个组件分别是块、区、段和表空间。

1. 块(Block)

数据以数据块为单位存储在 Oracle 数据库中。块是数据的最小单位，并且与操作系统(OS)的块的大小有直接关系——每个块的大小是 OS 块大小的倍数。

Oracle 的块不应小于操作系统的块大小。

数据库中的所有数据都存储在由 OS 块组成的数据文件中。OS 块是操作系统可以读或写的最小的数据单位，而 Oracle 块则是一种逻辑的存储结构。Oracle 数据库请求的数据量是 Oracle 块的大小的倍数，而不是 OS 块的大小的倍数。每当 Oracle 数据库请求数据块时，一个 OS 操作会将此请求转换为物理永久存储——从而将物理数据块与逻辑数据块分离开来。因此，数据库的数据可以驻留在多个磁盘上，因为 Oracle 知道逻辑数据块的地址。

块大小是在数据库创建时指定的，一旦数据库被创建完成，块的大小就确定了，不能再改变。尽管存在某些例外情况，例如重新创建数据库、导出整个数据库以及重新导入数据库，但这些过程并不简单。

由于数据块是数据存储的最小单位，Oracle 将读取的任何查询操作都是从最小的一个数据块开始的。

2. 区(Extent)

区是在一次分配中获得的一组连续的数据块。各个区之间不一定是连续的，因此，一个区可以分布在多个磁盘上。区只存储单个类型的信息并只能被存储在一个数据文件中。

3. 段(Segment)

段是为特定对象分配的一组区。段存储物理数据，因此表、索引、分区、聚簇表和物化视图都是段。而不存储物理数据的对象不是段，例如序列、视图、同义词和过程。段包含了表空间内逻辑存储结构的数据。段可以分为三类：

- **用户段** 存储了用户的对象数据，例如用户创建的表、索引等。
- **临时段** 在对一个查询进行排序时，Oracle Database 通常需要一个临时区域。Oracle Database 为此会在临时表空间中创建临时段。临时文件被映射到临时表空间上。同样，对于临时表和索引，会使用临时段。
- **Undo 段** 数据库的所有 **Undo** 数据都存储在 **Undo** 段中，这些段被映射到 **Undo** 表空间中。

4. 表空间(Tablespaces)

数据以表空间的形式逻辑地存储在数据库中。与最小的数据单位“块”相比，表空间是数据单位的最高级别。所有的数据文件都存储在表空间中，而表空间可以包含多个数据文件。表空间可以是两种类型之一：永久表空间或临时表空间。

永久表空间存储永久数据库对象。这些永久对象总是存储在这个数据库中。Oracle Database 使用两个永久表空间来管理数据库：SYSTEM 和 SYSAUX。SYSTEM 表空间由 SYS 用户使用，包括数据字典、表、视图和其他包含用于

管理数据库的重要信息的对象。SYSAUX 表空间为 SYSTEM 表空间提供了补充。它减少了 SYSTEM 表空间的负载，是许多数据库特性的默认表空间。这些表空间都不能被删除或重命名，它们都是必须有的。Oracle 建议将用户数据存储在自己的表空间中，而不要存储在 SYSTEM 或 SYSAUX 表空间中。

Undo 表空间是用于管理 undo 数据的永久表空间。Undo 表空间也包含 undo 数据文件，就像任何其他永久表空间一样。

临时表空间包含了与某个特定用户会话相关的信息的临时数据。临时表空间中的数据是暂时的。映射到临时表空间的数据文件是临时文件(tempfiles)。临时表空间不能被永久化。

表空间可以处于读/写模式或只读模式。SYSTEM、SYSAUX 和临时表空间永远处于读/写模式，不能更改为只读模式。

1.3　内存结构

内存结构驻留在数据库实例中。无论何时，只要实例启动，就会将操作系统的一部分 RAM 分配给数据库，而且 Oracle 数据库会进一步分配一个内存区域。该内存区域包含常用的代码、常用的数据、连接会话的详细信息以及数据库正在处理的查询的详细信息。Oracle 数据库的内存结构由系统全局区(System Global Area，SGA)和程序全局区(Program Global Area，PGA)组成。

1.3.1　系统全局区(SGA)

无论何时，当启动数据库实例时，都会给它分配 SGA 内存，而在实例关闭时收回 SGA 内存。SGA 由几种不同类型的内存结构组成。SGA 中的不同部分存储着代码、数据以及有关数据库当前状态的重要信息。SGA 的某些内存结构是不可缺少的，有些内存结构是可选的，这取决于你的系统所使用的数据库选项。以下是 SGA 的内存结构组成：

- Database Buffer Cache (数据库缓冲区缓存)
- In-Memory Column Store (内存列存储)
- Redo Log Buffer (重做日志缓冲区)
- Shared Pool (共享池)
- Large Pool
- Java Pool
- Streams Pool
- Fixed Area (固定区域)

1. Database Buffer Cache (数据库缓冲区缓存)

数据库缓冲区缓存的主要目的是将数据库数据块缓存到内存中。将数据缓存到数据库缓冲区缓存中，可以减少从物理磁盘读取数据的需要，因而可以更快地执行 SQL 查询。因此，数据库缓冲区缓存提供了更快的数据访问。

由于数据库缓冲区缓存用于存储数据，因此它是 SGA 中最大的区域。数据库缓冲区缓存可以缓存数据库中所有对象的数据，并最终存储诸如表、分区、聚簇表、索引和物化视图之类的数据。Oracle 使用 Least Recently Used(最近最少使用，LRU)算法来管理数据库缓冲区缓存中的数据。

数据库缓冲区缓存中的缓冲区(数据库块)可以有以下三种状态之一：

- **Unused(未被使用的)** 该缓冲区要么未被使用，要么从未被使用过。该数据总是在数据库缓冲区缓存中供使用。
- **Clean(干净的)** 缓冲区已准备好被重新使用。该缓冲区之前已被使用过，但其中包含的所有数据都已被写入磁盘，因此覆盖它不会有什么问题。
- **Dirty(脏的)** 缓冲区中包含尚未被写入磁盘的数据。在缓冲区被重新使用之前，必须将其中的数据写到磁盘中。

另外，每个缓冲区都有一个访问模式来说明该数据块是否能被重用。如果数据缓冲区是 pinned 访问模式，则显式地表明它已被保存在数据库缓冲区缓存中，不能被重新使用。如果数据缓冲区的访问模式为 free 或 unpinned，则能被重新使用。unused 状态的缓冲区不能有 pinned 的访问模式，而 clean 状态的缓冲区则可以有 pinned 或 free 的访问模式。

Oracle 使用 Least Recently Used 算法淘汰数据库缓冲区缓存中的数据。被重复访问的数据块，称为热缓冲区(hot buffer)，驻留在 LRU 的热(hot)端，而长时间未被访问的数据块则是冷缓冲区(cold buffer)，位于 LRU 的冷(cold)端。冷端的数据首先被淘汰。

当某个查询从数据库缓冲区缓存读取数据时，根据所执行的 SQL 语句的类型，会检索当前模式或一致模式下的数据。数据库缓冲区缓存中包含已被提交的和未被提交的数据。例如，如果未被提交的事务更新了某个块中的一行，除非提交实际发生，否则数据不会被写到物理磁盘中。

在这种情况下，数据库缓冲区缓存和磁盘中的数据是不同的，直到数据被写入磁盘为止。数据库缓冲区缓存中有数据的最新副本，而数据文件则有数据的读一致版本。如果执行一个 SQL 查询进行数据修改，则将使用数据库缓冲区缓存中的数据——这被称为当前模式(current mode)。如果执行 SQL 查询是为了检索数据，则会从物理磁盘或数据文件中检索读一致块，而不是从数据库缓冲区缓存中检索，这被称为一致模式(consistent mode)。

在发生检查点(checkpoint)操作时，Database Writer 进程(DBW*n*)会将数据库缓冲区缓存中的数据写到物理磁盘中。当空闲的缓冲区减少并下降到某个阈值以下时，脏块会被写到磁盘中。

为了能更好地管理数据库缓冲区缓存，可以将数据库缓冲区缓存分为一个或多个池：

- **Default (默认)**　这是默认的数据库缓冲区缓存。所有数据块都被缓存在这里。无论是否配置其他池，该默认池都会存在。其他两个池是可选的，且对默认池没有影响。
- **Keep(保留)**　这是为保留数据而创建的可选池。如果不希望数据因超期而被移出默认池，则可以使用此空间将这些数据块始终保留在内存中。保留池也使用 LRU 算法。
- **Recycle(循环)**　该可选池用于很少被访问的数据块。

2. 内存列存储(In-Memory Column Store)

内存列存储是 Oracle Database 12*c* 中新引入的特性。我们将在第 2 章中详细介绍这个主题。

3. 重做日志缓冲区(Redo Log Buffers)

重做日志缓冲区是 SGA 中的一个区域，用来存储与数据库所做的变更相关的重做条目(redo entry)。所有被捕获的变更记录，也被称为重做条目或变更向量，它们都使用数据定义语言(Data Definition Language，DDL)或数据操纵语言(Data Manipulation Language，DML)来编写。如果系统出现故障，这些重做条目会很有帮助，数据库使用这些重做条目来重建丢失的变更。

重做日志缓冲区是 SGA 中的循环缓冲区。将数据从重做日志缓冲区写入物理磁盘的后台进程是 Log Writer(LGWR)。

可以使用参数 LOG_BUFFER 在初始化参数文件中指定日志缓冲区的大小。

4. Shared Pool (共享池)

共享池用于存储代码，与数据库缓冲区缓存存储数据的方式类似。任何由 Oracle Database 处理的代码都被缓存在共享池中，包含 SQL 语句、PL/SQL (Procedural Language/Structured Query Language，过程语言/结构化查询语言)代码、SQL 语句的执行计划、数据字典信息、SQL 和 PL/SQL 结果缓存等。共享池进一步分为以下几个部分。

- Library Cache (库缓存)
- Dictionary Cache (字典缓存)
- Server Result Cache (服务器结果缓存)
- Reserved Pool (预留池)

Library Cache(库缓存) 库缓存是存储 SQL 和 PL/SQL 语句的共享池的一部分。每次查询执行时，数据库首先查询库缓存以查看 SQL 语句是否存在。如果 SQL 语句存在于库缓存中，则数据库会在软解析中重用相同的 SQL 代码或游标。如果该语句不存在，则数据库会在硬解析中重新构建一个该 SQL 代码的可执行版本。

库缓存进一步细分为两个区域：共享 SQL 区和私有 SQL 区。共享 SQL 区存储了被解析的 SQL 语句、SQL 执行计划以及被解析和编译的 PL/SQL 单元。共享 SQL 区处理第一次出现的 SQL 语句。共享 SQL 区被数据库中的所有用户所共享。当用户或会话发出一条 SQL 语句时，它首先使用 PGA 中的私有 SQL 区。而提交了相同语句的每个用户都有一个私有 SQL 区，这些私有 SQL 区都指向同一个共享 SQL 区。因此，不同 PGA 中的许多私有 SQL 区能够被映射到相同的共享 SQL 区。

Dictionary Cache(字典缓存) 字典缓存缓存数据库的所有字典对象。数据字典对象包含有关数据库的信息，并由表和视图构成。在解析 SQL 语句时，Oracle 会使用数据字典缓存。数据字典缓存也称为行缓存，因为它将数据保存为行而不是缓冲区。Oracle 使用 LRU 算法来维护字典缓存中的对象。

Server Result Cache(服务器结果缓存) 服务器结果缓存存储 SQL 语句和 PL/SQL 函数的实际结果集。因此，它有两个部分：一个用于 SQL，另一个用于 PL/SQL。服务器结果缓存能跳过 SQL 语句的执行，因为它能够让用户会话直接从这里获取最终结果。通过使用服务器结果缓存，Oracle 数据库避免了重新读取数据并重新计算结果。这带来了系统性能的提升。对于 PL/SQL 块，它们的函数结果集也存储在这里。因此，对于任何 PL/SQL 函数调用，如果结果已存在于服务器结果缓存中，便不会再次执行 PL/SQL 语句。

Reserved Pool(预留池) 预留池用于分配连续的大内存块。它允许大于 5KB 的对象(Java、SQL 游标或 PL/SQL)直接加载到内存中，而不需要一个独立的大内存区域。其结果是减少了内存碎片并提高了性能。

5. Large Pool

Large Pool 是 SGA 中的一个可选区域。它主要用于需要分配大内存的操作。由于操作所需的内存非常大，因此无法将其存储在共享池中。该区域主要用于 RMAN 的备份和恢复、并行执行语句和 Oracle 共享服务器。Large Pool 不使用 LRU 算法。一旦使用 Large Pool 的会话释放了内存，其他会话便可以重新使用它。

6. Java Pool

Java Pool 是 SGA 中的另一个可选区域。顾名思义，它用于存储所有与 Java 相关的对象。它将所有与 Java 相关的代码和数据存储在 Java 虚拟机(JVM)中。

7. Streams Pool

Streams Pool 也是 SGA 中的一个可选区域。它用于存储 Oracle Streams 的捕获和应用进程所产生的缓冲队列消息。由于 Oracle Streams 已被弃用，Oracle 建议使用 Oracle GoldenGate 来代替 Oracle Streams 的所有复制的功能。

注意：Oracle Streams 在 Oracle Database 12*c* 中已被弃用，并且可能在以后的 Oracle Database 版本中不被支持并且不可用。

8. Fixed Area(固定区域)

SGA 中的固定区域是一个用于内部管理的地方。它包含有关数据库和实例状态的信息：Oracle 数据库需要访问的后台进程或数据库中跨各个进程之间的通信。固定区域的大小由数据库自动设置，不能手动设置或更改。

1.3.2　程序全局区(PGA)

PGA 是 SGA 之外的一个内存区域。它包含了某个进程或线程的信息和数据，包括专用或共享服务器进程所需的与会话相关的变量。专用或共享服务器进程分配了 PGA 中所需的内存结构。PGA 被分为两个主要的区域：私有 SQL 区和 SQL 工作区。

1. 私有 SQL 区(Private SQL Area)

私有 SQL 区包含了被解析的 SQL 语句和其他要处理的专有会话的信息。每当服务器进程执行 SQL 或 PL/SQL 代码时，该进程都使用私有 SQL 区存储绑定的变量值、查询执行状态信息和查询执行工作区。私有 SQL 区包含了一个存储查询执行状态的运行区和一个存储绑定变量值的持久区。

2. SQL 工作区(SQL Work Area)

SQL 工作区是 PGA 内存的一个私有分配区域，用于内存密集型操作，如排序操作、位图操作和散列连接。

1.4　进程结构

Oracle 数据库的一个实例由多个在启动该实例时被激活的后台进程组成。Oracle 数据库使用这些后台进程来完成多个任务，并且每个后台进程都只有一项单独的任务。一些后台进程依赖于数据库的功能。例如，如果运行的是 RAC 数据库，则会发现几个与 RAC 相关的后台进程，它们没有运行在单实例数据库中。

某些后台进程是必不可少的，而且是任何数据库配置都必须有的。以下部分将讨论那些最重要的、必不可少的进程。

1.4.1 Process Monitor(PMON)

PMON 监控数据库中运行的所有其他后台进程。如果某个进程突然终止或出现故障，PMON 会执行该进程的恢复。PMON 还负责清理数据库缓冲区缓存并释放资源。

1.4.2 System Monitor(SMON)

SMON 负责监控整个系统，同时在实例运行期间当实例出现故障时执行恢复。SMON 使用重做日志文件执行实例恢复。如果有多个 RAC 节点，并且其中一个数据库实例失败了，则 SMON 可以负责恢复该失败的实例。SMON 还可以恢复任何被遗漏的事务，该事务可能是由于某个对象脱机而被遗漏，一旦对象恢复联机，SMON 就可以对其执行恢复。例如，如果在实例恢复期间、一个表空间发生脱机而且要恢复的这个事务被遗漏，那么 SMON 将在表空间重新联机时恢复这些事务。另外，SMON 还清除未使用的临时段，并负责合并表空间中的连续可用空间。

1.4.3 数据库写进程(DBW*n*)

DBW*n* 进程实际上对数据文件进行写操作。它将数据库缓冲区缓存中的所有脏缓冲区内容写入数据文件。在数据库中可能会运行多个 DBW*n* 进程来并行地写入。前 36 个 DBW*n* 进程被命名为 DBW0~DBW9 和 DBWa~DBWz。第 37 至第 100 个 DBW*n* 进程的名称是 BW36~BW99。DBW*n* 在出现以下任一事件时，将脏缓冲区内容写入数据库文件中：

- 服务器进程在扫描了最小阈值数后无法找到干净的或空闲的缓冲区。
- 发生检查点操作。

1.4.4 日志写进程(LGWR)

LGWR 将 SGA 区中的日志缓冲区的内容写入联机重做日志文件中，之后日志缓冲区便可被重用。在出现下列事件时，LGWR 开始将日志缓冲区内容写入重做日志中：

- 有提交操作时
- 发生日志切换时
- LGWR 上次写操作完成后三秒钟
- 当重做日志缓冲区达到 1/3 满或包含了 1MB 的缓冲数据时
- 当 DBW*n* 将脏块写入数据文件时

如前所述，日志文件总是被镜像/复用的，因此 LGWR 总是要同时写入多个日志文件的副本中。

1.4.5　Checkpoint(CKPT)

检查点事件会触发从数据库缓冲区缓存到数据文件的数据写操作。当检查点发生时，它会使用检查点信息更新控制文件和数据文件的头部，其中包括SCN、在联机重做日志文件中的位置以及其他信息。

1.4.6　Recoverer(RECO)

RECO 进程负责解决分布式事务中的故障。一个节点的 RECO 进程会自动连接到有故障的分布式事务所涉及的其他数据库上。一旦 RECO 建立了连接，它就会自动解决所有失败的/不确定的事务。

1.4.7　其他进程

Oracle 数据库还运行一些其他进程。但要详细讨论每个进程超出了本书的范围，表 1-1 中汇集了一些进程和它们的简要说明。第 2 章会介绍与内存列存储相关的进程。

表 1-1　Oracle 后台进程

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
ABMR	自动块介质恢复后台进程	协调任务的执行，诸如过滤重复的块介质恢复请求和执行流量控制等	否	否
AP*nn*	逻辑备份/Streams应用进程的协调进程	从读进程服务器获取事务，并将它们传递给应用进程服务器	否	否
ARC*n*	归档进程(可能有30个)	当重做日志文件已满或发生联机重做日志切换时，将它们复制到归档存储中	否	否
BMR*n*	自动块介质恢复从属进程池	从实时可读的备用数据库中获取块	否	否
BW*nn*	数据库写进程(可能有20个)	将修改后的块从数据库缓冲区缓存写入数据文件。第37~100个数据库写进程被命名为BW36~BW99	是	是

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
CJQ0	作业队列协调进程	生成作业队列从属进程(J*nnn*)，以便执行队列中的作业	否	是
CKPT	检查点进程	在执行检查点进程时启动DBW*n*并更新数据库的所有数据文件和控制文件，以显示最新的检查点	是	是
CP*nn*	数据库捕获进程	通过使用工具LogMiner来捕获重做日志中的数据库更改	否	否
CS*nn*	I/O校验进程	将I/O发送到存储作为存储校验的一部分。在数据库的每个节点上的每个CPU都有一个从进程	是	是
CX*nn*	流传播发送进程	将LCR发送给传播接收进程	否	否
CTWR	变更跟踪写进程	跟踪已更改的数据块，并作为RMAN块变更跟踪特性的一部分	否	否
DBRM	数据库RMAN进程	设置资源计划并执行其他RMAN任务	否	是
DBW*n*	数据库写进程	将修改后的块从数据库缓冲区缓存写入数据文件。可以有1~100个DBW*n*进程。前36个DBW*n*进程命名为DBW0~DBW9和DBWa~DBWz。第37~100个DBW*n*进程被命名为BW36~BW99	是	是

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
DIA0	诊断进程0(尽管可能有10个进程，但0是当前被使用的)	负责挂起检测和死锁解决方案。触发DIAG进程执行诊断任务	是	是
DIAG	诊断捕获进程	执行诊断导出并执行全局oradebug命令	是	是
DM*nn*	Data Pump Master进程	协调由Data Pump工作者进程执行的Data Pump作业任务并处理客户端交互	否	否
D*nnn*	调度进程	在共享服务器配置中，调度进程将连接请求放入连接请求队列中	否	是
DW*nn*	Data Pump工作者进程	执行由Data Pump Master进程分配的Data Pump任务	否	否
EMNC	事件监控(EMON)协调进程	协调数据库中的事件管理和通知活动，包括Stream事件通知、连续查询通知和快速应用通知。生成EMON从进程	否	否
E*nnn*	EMON从进程	执行数据库事件管理和通知	否	否
FBDA	闪回数据归档进程	把被跟踪表的历史行数据归档到闪回数据存储中，并管理归档空间、结构和数据保留期限	否	否
FMON	文件映射监控进程	生成FMPUTL进程，这是一个与存储厂商提供的映射库进行通信的外部、非Oracle Database进程。它负责管理映射信息	否	否

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
GEN0	通用任务执行进程	执行所请求的任务，包括SQL和DML	是	是
I*nnn*	磁盘和磁带I/O从进程	由DBWR、LGWR或RMAN备份会话而产生，作为I/O从进程	否	否
IMCO	内存协调进程	将内存压缩单元(In-Memory Compression Unit，IMCU)恢复出来	否	否
J*nnn*	作业队列从进程	处理队列中的作业；由CJQ0产生	否	是
LG*n*	日志写从进程	在多处理器系统上，LGWR生成从进程以提高写入重做日志的性能。当有一个SYNC备用目标时，不会使用LGWR从进程	否	否
LGWR	日志写进程	将日志缓冲区的内容写入重做日志	是	是
L*nnn*	池化的服务器进程	处理数据库驻留连接池(Database Resident Connection Pooling，DRCP)中的客户端请求	否	否
LREG	监听器注册进程	向监听器通告实例、服务、处理程序和端点信息	是	是
MMAN	内存管理进程	作为SGA内存代理并协调内存组件的大小	否	是
MMNL	管理特性监控小进程	执行经常性、轻量级的与管理特性相关的任务，例如会话历史捕获和度量计算	否	是

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
MMON	管理特性监控进程	为自动工作负载资料库(Automatic Workload Repository，AWR)收集统计信息	否	是
M*nnn*	MMON从进程	代表MMON执行管理任务	否	否
MS*nn*	LogMiner工作者进程	读取重做日志文件并将之翻译、打包为事务	否	否
N*nnn*	连接代理进程	监控空闲的连接并关闭数据库常驻连接池中的活跃连接	否	否
OFSD	Oracle文件服务器后台进程	监听新的文件系统请求，包括管理(如mount、unmount、export)和I/O请求，并使用Oracle线程执行它们	是	是
PMON	进程监控	恢复失败的进程资源。如果使用的是共享服务器体系结构，则PMON将监控并重启任何失败的调度或服务进程	是	是
P*nnn*	并行查询从进程	根据需要启动和停止参与并行查询的操作	否	否
PR*nn*	并行恢复进程	执行由协调进程执行并行恢复所分配的任务	否	否
PSP0	进程生成(Process Spawner)进程	启动和停止Oracle进程。通过启动/停止ASM重平衡从进程来减少ASM重平衡主进程(Rebalance Master Process，RBAL)的工作负载	否	是
RCBG	结果缓存后台进程	支持SQL查询和PL/SQL函数结果缓存	否	否

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
RECO	恢复进程	解决涉及分布式事务的故障	否	是
RM	真实应用测试(Real Application Testing，RAT)遮蔽(Masking)从进程	与数据遮蔽(data masking)和RAT一起使用	否	是
RP*nn*	捕获处理工作者进程	处理一组工作负载捕获文件	否	否
RPOP	实时恢复重构daemon进程	重新创建和/或重构快照文件和备份文件中的数据文件。它与实时恢复特性配合使用，确保对数据文件的实时访问。本地实例可立即访问远程快照文件的数据，而被恢复的主数据文件的重构也同时生成。对数据的任何更改都将在实例的DBW进程和RPOP之间进行管理，以确保将最新的数据副本返回给用户	否	否
RVWR	恢复写进程	将闪回数据写到闪回恢复区中的闪回数据库日志中	否	否
SA*nn*	SGA分配进程	在实例启动期间会分配一小部分的SGA区。SA*nn*进程以小块(chunk)的形式分配SGA的其余部分。该进程在SGA区分配完后退出	否	是
SMCO	空间管理协调进程	协调各种与空间管理相关的任务的执行，例如主动地分配空间和回收空间	否	是

(续表)

简称	进程名称	描述	是不是基本DB操作所要求的	是否默认启动
SMON	系统监控进程	执行关键任务，例如实例恢复、不活动事务恢复，以及维护任务，例如临时空间回收、数据字典清理和undo表空间管理	是	是
S*nnn*	共享服务器进程	在共享服务器配置中，共享服务器检查连接请求队列(由调度进程产生)并为连接请求提供服务	否	是
VKRM	用于RMAN进程的虚拟调度程序	RMAN活动的集中调度程序	否	是
VKTM	时间的虚拟保留进程	负责提供钟表时间(每秒更新一次)及时间参考计数器(每20毫秒更新一次，并只在高优先级的运行状态时才可用)	是	是
W*nnn*	空间管理从进程	由 SMCO 产生的从进程，执行空间管理任务	否	是

1.5　本章小结

在本章中，我们介绍了数据库是由存储整个数据集的物理文件和一个实例组成。一个实例具有一组用于管理物理文件的内存结构和后台进程。因此，Oracle Database 服务器=物理文件+实例。

一个数据库可以有一个或多个实例。如果数据库只有一个实例，则称之为单实例数据库；如果它有多个实例，则称之为 RAC 数据库。

数据库可分为三种主要结构：存储、内存和进程。存储结构可以再分为物理存储和逻辑存储。物理存储是指物理文件，逻辑存储是指以表、索引、表空间等形式表现的数据的逻辑分布。

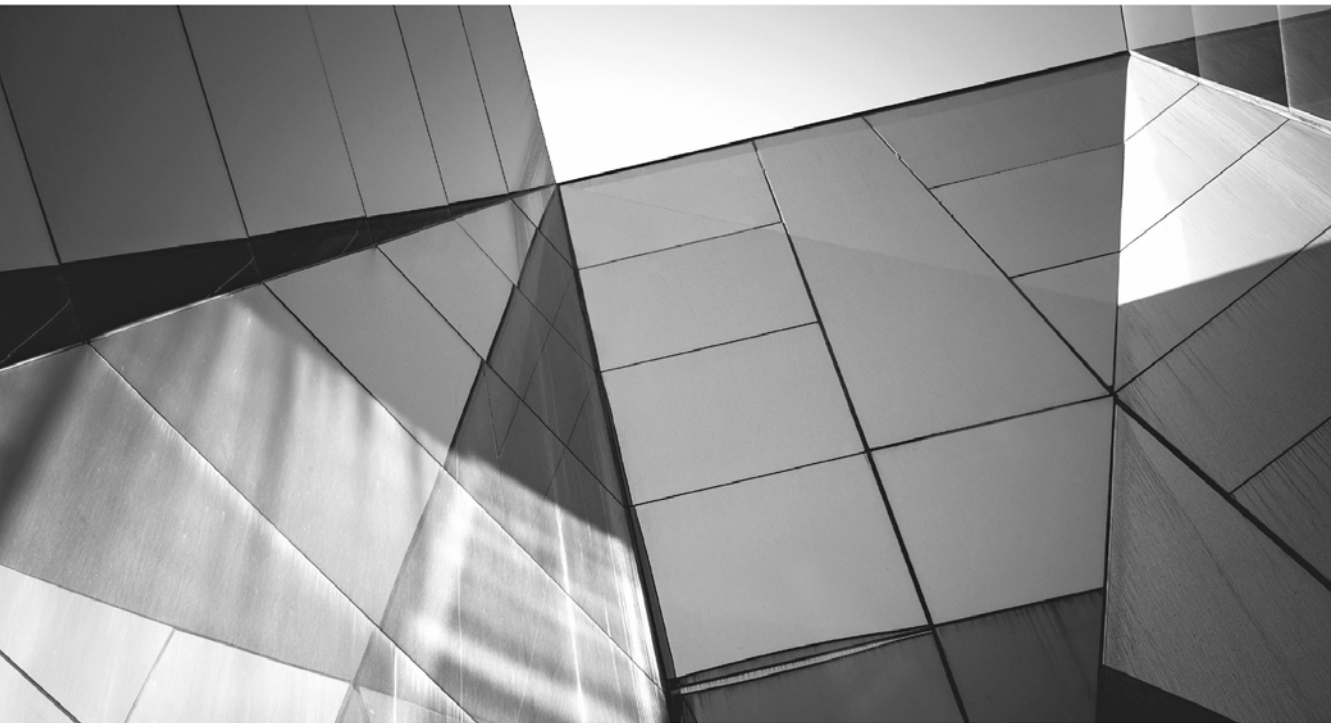
数据库的物理结构包括数据库文件、控制文件、联机重做日志文件、归档重做日志文件、参数文件、口令文件、告警日志文件、跟踪文件、导出和内核

文件以及备份文件。

数据库的逻辑结构可以更好地对数据处理、存储和检索进行控制。数据库的逻辑结构有四个部分：块、区、段和表空间。

内存结构驻留在数据库的实例中。内存区域包含常用代码、常用数据、连接会话的详细信息以及正在被数据库处理的查询的详细信息。Oracle Database 的内存结构由 SGA 和 PGA 组成。SGA 由数据库缓冲区缓存、内存列存储、重做日志缓冲区、共享池、Large Pool、Java Pool、Streams Pool 和固定区域组成。

Oracle 数据库的实例由多个后台进程组成，这些后台进程在实例启动时被激活。Oracle Database 使用这些后台进程完成多个任务，每个后台进程都有自己唯一的一项任务。最重要的且必不可少的进程是 Process Monitor(PMON)、System Monitor(SMON)、Database Writer(DBRW)、Log Writer(LGWR)、Checkpoint (CKPT)和 Recoverer(RECO)进程。



第 2 章

In-Memory 体系结构

早在 2005 年，Oracle 数据库的平均数据量是几个 TB 大小。那时，对于数据库管理员(DBA)来说，管理几个 TB 的数据库是一项艰巨的任务。然而，当今的数据库是以 PB 为单位的，管理 PB 大小的数据库对于 DBA 来说已经不是什么大问题。技术进步使数据库具有比以往更多的功能。甚至昨天最快的速度都无法和今天的速度相比。连快速的定义也已经完全改变了！