

## 第8章 Oracle数据库系统管理概述

本章要点：

什么是数据库的生命周期

DBA是什么

建立一个数据库的系统管理步骤

本章将讨论数据库系统管理员（DBA）是什么和做什么，数据库生命周期由哪些部分组成，创建 Oracle 数据库时 DBA 应遵照执行的步骤和 Oracle DBA 的维护职责。正如能从 RDBMS 映射到 Oracle 的 RDBMS，这一章的许多内容也可以应用到 DBA 支持的几乎任何其他的 RDBMS 中。这是由于包括管理在内的数据库技术和理论能够适用于所有 RDBMS。正如我们在第 4 章“Oracle 的解决方案”中提到的，包括 Oracle 在内的 RDBMS 都有不同方面的缺陷并在很大程度上都忠实于他们对关系模型的特定实现方法。

### 8.1 什么是数据库的生命周期

软件的生命周期是一种描述如何创建和进化软件的方法。一个经常用来描述一个软件的生命周期的谚语是“从摇篮到坟墓”，其意思是从开始到结束。包括数据库在内，所有软件的生命周期都要经过一系列阶段。这些阶段是分开的，它们使软件开发过程朝着更深层次方向发展，换言之，它们描述了生命周期的发展。这些过程是系列化的和相互依赖的。例如通常说，如果阶段 A 先于 B，那么阶段 B 依赖于 A，更进一步，B 可以对 A 提供反馈，循环进行。要想简单地表示先于和依赖于，你可以使用 A->B。要增加反馈表示，你仅需要画一条从 B 返回到 A 的弧线。

有许多模型提供各种不同的方法来表示（图形化或其他的）软件的生命周期。最典型的软件生命周期模型是瀑布模型。还有其他模型存在，例如螺旋模型。首先，让我们看一个最基本的软件生命周期的描述。它是一个简单的瀑布模型。

分析 设计 实现 维护

这个模型是所有软件开发的基线。现在简要地概括描述每个阶段。

#### 8.1.1 分析阶段

分析阶段代表用户需求的全面收集和文档编制，另外分析这些需求的操作意义是什么，因此通常称分析阶段为需求分析阶段。根据需求的不同，客户提供不同层次的合作。有些提供十分详细的技术工作的描述，而另一些可能只简单地用言辞表述。尽管通常焦点放在略述要创建的系统的功能性要求，它可能包括也可能不包括处理诸如用户接口或性能考虑这样的项目的详细说明。详细说明书简单地说是需求的另一种说法，当一个程序员想开始设计他的程序时，他必须查看“说明书”。

### 8.1.2 设计阶段

设计阶段主要包括算法设计，也包括输入/输出考虑；如果是交互式的程序，还有格式化和显示考虑；或者如果程序是用于科学方面的，还有精度问题。无设计的编码等同于无计划的长途旅行。任何情况下，在设计阶段所做的许多重要选择不仅影响功能和显示，而且还会影响纯粹的执行。毫无疑问，数据结构和算法的正确选择会导致或打断该程序的后续执行。

### 8.1.3 实现阶段

实现阶段主要是程序或系统的编码。在你有了正确选择的数据结构、算法和其他类似的构建块后，你能够用所选择的语言开始编程。在编程本身，也有许多选择要做。例如，哪种控制流能最好地实现一个特定算法？另外，在这个阶段，你也能够进行隐含或明确选择，如对存储的速度（反之亦然）进行选择。这时，软件开发实际上经常受到理论上的约束。例如，你或许必须使用某种类型的程序接口，不是因为它们在某种程度上更好或者更快，而是由于它们被用一种或另一种方法标准化了。当然，实现还应该说包括其他阶段，通常所指的是测试和展开。

#### 1. 测试阶段

测试阶段正如你能想到的一样。例如，在测试阶段，你可能首先测试系统的所有独立组件（单元测试），然后把它们放在一起进行端到端的测试（开始到完成），这也是众所周知的整体或系统测试。你做一下改变，然后返回（回归），重新测试每一项（或者至少测试可能受影响的每一项），比较输出结果，确保功能没有丢失。其他测试方法是装载测试和执行测试。装载测试一个系统时，开始用所希望的最少用户数运行测试，然后增加用户数接近你希望的最大的用户数。比较每一个运行结果，看发生了什么影响。如果有，在你的系统执行中增强并发性。另一种装载测试方法是增加通过你的系统的数据流量，这正好与增加访问用户数相反。在多种情况下，两种情况你都能见到。

在执行测试时，你要确立你的执行目标是什么。在很多种情况下，如果客户没有明白地表达过执行目标，只需简单地询问客户。通常，你考虑三个执行措施或尺度中的一个：经过时间、反应时间和吞吐量。经过时间，也称为分界时间或时钟时间，它是进程持续了多长时间，用结束时间减去开始的时间，这对批处理系统是有用的；反应时间是一个进程对一个请求做出响应所花的时间，因此，这个措施对交互式系统（例如 DSS系统）是有益的；吞吐量是一个工作单位时间内通过系统的块数，通常是业务定义的“事务”的度量，例如，每秒钟的事务数是否吞吐量的一种度量方式，这种度量方式对 OLPT系统是有用的。执行测试可以被定义为功能需求的一部分。

#### 2. 展开阶段

展开阶段发生在系统通过它的所有测试阶段之后，并准备在它的所有终端系统、客户和服务器展开或安装时。安装所有支持的硬件和软件过程，当然也包括应用软件自身的运行形式都有关于展开的问题。根据合约声明，一旦每件事情被展开，它可能要进行简单的再测试，然后经过一些事务阶段，有效地“移交”给负责的用户（如果在合同中该项目是外协的，有可能要提供一些现场的应用软件支持）。展开完成后，产品进入维护阶段。

### 8.1.4 维护阶段

维护阶段是许多软件的生命周期中最长的部分，估计许多软件在维护阶段的时间占 70% 或更多。因此，仅就此而言，这可能是最重要的阶段，因为在软件开发中的错误决定能够削减软件某方面的正确功能。这里的决定包括许多与“开放”系统有关的问题，例如，可移植性和互操作能力，它会可能影响软件长期的生活中的一部分。维护也涉及许多系统集成问题，例如替换过时的现有硬件和软件组件，由于各种因素，随时间删除某些其他组件，并引入一些新的组件。由于各种原因，正如你想象的一样，涉及的部分越多，集成可能变得越复杂，因为涉及到许多部分之间的相互作用。在执行阶段，当预测什么是想要的行为时，标准化的接口变得更加重要。关于系统集成，有条原则“移动的部分越多，就越有可能发生问题（和失败）”。集成问题背后的原因之一是异质性，也就是，如果许多或所有不同组件来自于不同厂商，无缝集成（也就是，没有任何不兼容性）的概率是很小的。

现在你了解了通常软件开发的整个生命周期，考虑一个数据库的特定生命周期。相对于一般的软件模型，数据库特定的活动是什么？以下是一些解答：

分析——概念模型（或业务模型）开始成形。

设计——概念模型被限定。逻辑设计包括将概念模型初步映射为关系模型和实体关系图（ERD）。执行规范化。然后规划物理设计。同样地，只要逻辑设计一完成，数据库的前端和后端程序设计便能够开始。

实现——对一个数据库系统，这个阶段是通常所说的建立，就是用数据库语言（DDL）建立数据库结构；如果初始装载是适当的（与超时装载相对），装载就是用数据库操纵语言（DML）或厂商的应用工具最初将数据装载到数据库中。同时，对一个数据库系统，程序的编码能够从设计阶段开始或连续。

测试——在数据库创建和装载之后，程序被写入，全部进行测试，进行单元和整体测试。

展开——在许多现代系统中，展开意思是在 PC（客户）和服务器（Web 服务器和数据库服务器）上安装软件。

维护——这与其他软件系统没有任何不同，除了它是更多地属于以数据为中心的软件而不是以进程为中心的软件。

## 8.2 DBA是什么

我的学生经常请求我描述当前市场上所有可获得的数据库职业有什么不同。当然，DBA 是数据库人员中处于中心位置的，所有其人员都围绕着他。在考虑 DBA 是什么之前，首先考虑一般能够在市场上找到的所有与数据库相关的职位：

设计者。

建筑师。

工程师。

开发者。

程序员。

操作员。

数据库系统管理员（DBA）。

首先考虑设计者。设计者收集初始用户的需求（如果还没有收集）。然后他开始分析、精炼和推断，使其适合标准化数据库的模型输入，例如使它们符合关系模型。最初他与用户讨论，随后分析帮助形成典型的关系模型，这些在第一部分“数据库管理原理”和第2章“逻辑数据库的设计和标准化”中有一些描述。概念模型也可能看作业务模型。它不仅包括已有的业务数据模型，而且还有业务规则，这将在后面作为在任意给定的RDBMS实现中的约束、过程和触发器来明确它们。正如你猜想的，设计者的责任范围跨越需求分析、逻辑设计和物理设计阶段。

建筑师，有时也称数据库系统设计师或信息系统设计师。他对在设计和工程中的所有硬件、软件和包含在企业中的网络部件承担很大的责任。因此建筑师部分是设计者，部分是工程师，在他的职业中可能有一些或全部与其他所有数据库职位相关的职务。建筑师通常不仅是在数据库技术领域，而且也在信息技术领域有多年的工作经验（例如10~15年或更多）。

与建筑师相比，工程师明显地有更窄范围的责任，只注重于数据库技术或仅是一个组织中的一个部门或一个组织分枝。他的范围可能在他的技术与组织范围内更窄。但是，工程师一般是数据库系统管理员或有多年工作经验的开发者（例如5~10年或更长）并有足够的经验，并理解与数据库相关的技术和可能更多的外围知识。

开发者，有时与程序员意思相同，但能够做一个区分。数据库开发者一般用前端程序工作，前端程序提供交互式输入（例如Oracle Forms）和输出（例如Oracle Reports）。因此，数据库开发的交互特性使许多开发者用Microsoft Visual Basic、Microsoft Access、Sybase(PowerSoft) PowerBuilder和Borland Delphi这样的程序工作。所有这些程序都是关于建立图形用户界面（GUI）的，例如，那些典型地遵循Microsoft或Web浏览器的界面设计。这些基于Microsoft Windows的前端，有时不严格地称为在客户/服务器模式中的客户应用，或者甚至有时不严格地简称为客户/服务器。那些基于Web浏览器和Web技术的前端被简单称为基于Web。不幸的是，这种不严格的分类是误导的，因为基于Web的系统是简单的客户/服务器系统的特例。然而，开发者负责建立前端、GUI-数据库、输入/输出程序。但是他们生产的实际程序代码量是很少的，因为几乎所有的代码都由前端工具自己生成。当开发者必须写实际代码时，他们通常也只仅需写相对高级的语言，如第4代语言（4GL），包括宏和脚本语言。许多这些工具既基于ODBC技术，又可以基于本地厂商的RDBMS协议，例如Oracle SEL\*Net或Sybase Open Client(DB\_Lib和Net\_Lib)。

与开发者相比，数据库的程序员实际上编写很多代码，经常使用的是SQL和基于厂商的、控制流的、基于SQL的第4代语言，如Oracle PL/SQL和SQL server Transact-SQL（T-SQL）。但是你将经常见到数据库的程序员编写嵌入式SQL或本地应用程序接口（API），如Oracle Call Interface（OCI）或Sybase Open Client。正因为这样，当然你将发现这些数据库程序员经常使用C、C++和Java。因此他们与使用COBOL、FORTRAN访问文件系统而不是数据库的前辈没有太多的不同。实际上，在许多方面，这些程序员是同一个人或者是他们更新了他们的技术。

操作员可能被认为是支持操作（也就是所谓的维护和操作，或M&O）的技术人员。他的明确职责是负责定期维护确定的硬件和软件。你将发现操作员一天24小时一周7天轮流值班。操作员典型的职责是负责磁带（和光盘）归档，包括数据库归档。随着现代的磁带（和光盘）自动设备（接卡箱、自动光盘机等）的使用，操作员在这个舞台上一点儿也没有变得不重要，

但可能会减少一些工作量，因为这些设备很少需要人关心和进行输入。操作员除了履行第三位的存储职责（如“悬挂磁带”），还可能履行“悬挂电缆”（物理安装网络电缆）的角色，在计算机房布局电源分布电缆，连接第二存储设备（硬盘），建立支持多计算机机架，甚至于建立一些其他设备。

最后是数据库管理员（DBA）。他是数据库的中心职位。通常看作是级别在程序员之上，工程师之下。DBA经常发展成为工程师，或者甚至是建筑师，而且他经常已是开发者、程序员或操作员DBA和操作系统的系统管理员（SA）负责维护和执行任务。同时，DBA必须与开发者和程序员紧密工作，以建立一个完全的、集成良好的数据库系统；其中，DBA将精力集中在工作在后端，开发者/程序员将精力集中在前端。另外，DBA既可已经是或可能将成为设计者。在任何一种情况下，DBA必须与设计者特别紧密地合作，在设计到建立（实现）之间建立联系的桥梁。

在现实世界里，你将经常发现有智慧的人扮演重要的角色。例如，一个人可能是设计者、开发者/程序员和DBA。在某些情况下，你将发现一个人同时作为DBA和SA。在有些组织中，以上所有的职业是分开的和有明确区分的工作位置。一个人担任多种角色可能由于种种因素，包括一个人技术才能的广度和深度，或者公司不愿意分得这么细或者缺乏对不同角色的需要的远见卓识。另外，有时一个公司既不能负担与公司规模相对应的全部位置，又缺乏对数据库重要性的认识。同时日益增长更多的情况是公司不能雇用和补充全部的多个位置，因为相对于需求而言，仍然缺少与数据库相关的人员。这种倾向不仅使单个位置的潜在薪水提高，而且还有鼓励有能力履行职责的工作人员担任多个角色。

现在看一看与通常的DBA相比，Oracle DBA有什么特殊性。

### 8.3 建立一个数据库的系统管理步骤

对于Oracle DBA，在创建Oracle数据库时有几个“管理步骤”，它们中的许多只需少量的修改就能应用于任何其他厂商的数据库中。有很长一段时间，我有很多工作在不同背景的数据库（包括通常任何类型的数据库和特定的用Oracle数据库）的同事和学生，询问在创建数据库时DBA应遵循的主要步骤是什么。这根本不是一个不合理的问题，因为事务的正确顺序经常被掩盖了。更常见的是，有许多参考和资料用低层次的说明掩盖了高级顺序。例如你找到了如何编辑和改变在init.ora文件中参数，但是怎么适合“大图形”？要获得全部的顺序和事件，下列清单足够详细，但还不够深入：

- 1) 安装Oracle软件（RDBMS、网络、客户等等）。
- 2) 编辑初始化参数文件（init.ora）。
- 3) 创建Oracle实例。
- 4) 创建Oracle数据库，包括系统表空间和重做日志。
- 5) 创建数据字典（也就是运行cat脚本）。
- 6) 创建其余的表空间和回滚段。
- 7) 创建表。
- 8) 添加主键和唯一键（未使能）。
- 9) 使能主键和唯一键（创建索引）。
- 10) 增加外键（已使能）。

- 11) 创建外键索引和为其他适当的列创建索引。
- 12) 增加检查约束和写适当的触发器。
- 13) 编写适当的过程、函数和其他代码 (也就是 Pro\*C)。

尽管这些步骤中有些能够更进一步分割 (或者合并), 以我的工作经验, 在创建数据库中, 这种DBA (和部分的开发者) 职责的分割方法是有很高的灵活性和可伸缩性。

1) 当安装Oracle软件时, 不仅必须考虑你当前的需要和数据库的初始状态, 而且还要考虑将来的需要和预期的增长。使用优化弹性体系结构 (OFA) 将是十分有益的。优化弹性体系结构的更进一步信息 (包括目录和文件的创建), 请参见第7章“探讨Oracle环境”和附录A“在Solaris系统上的Oracle”, 它们都包含有实现优化弹性体系结构数据库的概述。

2) 当然, 除非你安装和希望使用缺省的 Oracle实例和数据库 (ORCL), 你需要创建自己的数据库。要做这一工作, 你首先必须编辑初始化参数文件 (init.ora或init<SID>.ora)。你至少应指定db\_name和control\_files参数, 但可能还有其他一些参数, 如 db\_block\_size尺寸。最后, 当你创建了你的回滚段并想在启动时作为不变的基础用 rollback\_segs打开它, 你还必须重新编辑init.ora文件。另外, 还有很多其他你可能调整或使能的事情, 如归档日志模式。

3) 然后, 创建一个实例, 首先创建包含控制文件声明的已编辑的 init.ora文件。在UNIX系统中, 当你在svrmgrl中启动为非安装状态时, 你已“创建”了你的实例。在UNIX系统中, 能够用命名为dba的UNIX组连接internal。在NT系统中, 你将使用 Oracle NT实例管理器 (orsdim80.exe) 的命令行方式或图形用户接口方式, 创建你的实例。这也将创建口令文件和相关服务 (用启动文件)。当你创建了NT实例时, 你为internal指定了一个口令, 以使你能在svrmgr30.exe中连接internal。参看第7章“探讨Oracle环境”和附录B“Windows NT上的Oracle”, 其中有关于NT中的Oracle的更详细信息; 参看第7章和附录A, 其中有关于UNIX中的Oracle的更详细信息。

4) 接下来的任务是在非安装状态下创建你的数据库, 当你编写创建数据库脚本时, 至少应指定创建系统表空间和初始的重做日志, 但是你还应该利用这个机会设置诸如MAXDATAFILES和MAXLOGFILES这样的参数为你希望的值。一个初始的系统回滚段被创建, 该回滚段以后将被删除。在数据库创建时, 读取你的 init.ora参数, 在指定位置被创建你的控制文件。

5) 你将运行 Oracle提供的通常为任意 Oracle数据库运行的基本包: catalog.sql、catproc.sql、pupbld.sql; 可能还要运行其他的一些包, 如 utlexcpt.sql、utlxpln.sql或catrep.sql, 这取决于你的应用。这实质上是在 Oracle的数据字典 (X\$表) 中建立你的系统管理访问和视图 (对SYS、SYSTEM和其他用户)。创建V\$视图、DBA\_表和USER\_表。

6) 然后你能创建余下的用户表空间, 它将服务于你的交互用户和应用的需要。在这时, 你将指定你的表 (和索引) 正确的尺寸并应该能够根据你的估计提供合理的表空间大小。参看第4章“Oracle的解决方案”。如果你想使系统监控 (SMON) 能够自动合并表空间中的碎片, 记住设置PCTINCREASE为1。另外, 在这里你将需要创建你希望的用户和应用回滚段。等做完这些, 要记住返回去重新编辑 init.ora中的rollback\_segs参数。

7) 现在再使用你的尺寸输入, 准备好创建你的空表。你的考虑应该包括覆盖表空间的存储参数和考虑什么数据类型最适合存放你的数据。例如, 使用 VLDB, 逻辑压缩可能十分重要。

记住，一定将具有大量的连结和其他有同时大量访问的表分开，不仅分开到不同的表空间，而且最好是分开到在不同的磁盘上。参看第3章“物理数据库的设计、硬件和相关问题”和第4章“Oracle的解决方案”、第16章“性能调整基础”和第19章“调整输入/输出”，以得到更详细的关于I/O分配和优化的指导。

8) 在Oracle中记住增加使能（缺省值）的主键将在拥有者的缺省表空间中自动地创建索引，这通常总不是一个好办法。因此，应定义你的所有全部主键初始无效。主键所保持的为真，对唯一键也同样保持为真。

9) 使用特定的物理参数分别使能所有你的主键，这保持了逻辑定义与物理布局分开。这总是一个好办法，特别是对VLDB和关心可移植性的软件。同样，唯一键也能照同样方法处理。

10) 在Oracle中，使外键使能（缺省值）不会在任何地方自动地为它们创建索引。了解这一点是很重要的，因为当大表被连接时，外键可能会成为性能损害。

11) 在大表中为希望连接的外键创建索引，因为没有它们，父表被行级锁定。使用它们，就可像普通表一样，既可有行级锁定，也可没有锁定要求。另外，索引大型表中那些可能用SELECT的列列表部分或用WHERE语句进行大量访问的列。

12) 附加的约束和触发器使关系数据库实现数据一致性。没有它们，除了主键和外键约束外，你实际只是拥有了另一个数据库。用这些附加的结构，你能增加业务规则，如强迫域完整性（如一系列的有效值范围）和数据流的依赖性（例如，向数据库中插入一个雇员时，首先将他的识别信息插入到一个表中，最后将他的其他信息插入到定义雇员的其他表中）。记住，当约束使能时，对每一行的每一次访问都进行检查，这是要付出很大的代价的。为识别任何对约束的违反（例外），你一般应在任何大块的装入或大的修改之后启用它们，然后禁用它们。定义有触发器的表进行修改时触发器被触发，然后它能够任何其他表上操作（或简单地做一些计算或文件处理），但是它们不能在定义它们自身的表上操作，与检查约束正好相反，检查约束仅能在定义它们的表上操作（“检查”列）。约束是简单的数据库定义语言（DDL）语句，触发器通过数据库定义语言（DDL）定义，但它从本质上讲是一个过程体（DML）。

13) 过程、函数、游标和嵌入式SQL做约束和触发器不能做的工作。例如过程和函数不依赖于单个表的修改事件，它们能够独立于任何给定表的修改运行。当基于集的访问在某种程度不足够（有时是这样）时，游标使过程、函数或嵌入式SQL内部的实时记录访问成为可能。嵌入式SQL提供紧密集成，特别是使用游标与使用第三代语言，如C。借助于所有这些程序工具，一个Oracle DBA或开发者/程序员能够完成几乎任何相关数据库的编程任务。正如你预料的一样，这里正是DBA离开和开发者/程序员进入的地方，尽管在理论上，早在逻辑设计完成时，他们可能已经开始设计和编写程序代码了（他们只是还没有编译它，因为数据库必须被创建）。

对大系统，这些步骤的伸缩性是，除了前三步，每一个剩下的步骤可能符合一个或多个SQL\*Plus脚本。例如，你可能具有表8-1中的脚本。

为什么要完全分开帮助？它在多层次上进行帮助。首先，它系统化组织事情。例如，当你创建你的数据库时每次创建一步，如果某事失败，你能精确知道失败发生的地方。经常，你仅需要修复你所做的并简单地重新运行那一步，恰好与完全删除并返回到很大的“创建每一件事”脚本的起始点相反。因此，这些步骤帮助你模块化你的数据库创建。这样做有很多

的原因，至少不仅仅是查找在创建中出现的故障，更进一步，这些步骤倾向于从物理层中分离出逻辑定义。这意味着，当对一个或另一个（通常很可能是物理的）进行改变时，你只需修改一个脚本，不是对每个逻辑-物理对中的两个进行修改，如表-表空间、addpk-enablepk等等。另外，正如你想要的，如果总的应用是足够大的，当然这些脚本能够更进一步分为不同的区域组。最后一个动机是考虑在另一个平台上使用不同的物理层导入，并重建这个数据库。这些步骤和脚本肯定是会有帮助的（如移植），甚至比导入和导出对象更有帮助。

表8-1 伸缩性建立脚本

步骤	脚 本	描 述
(4)	crdb.sql	创建数据库
(5)	runcat.sql	运行cat脚本（创建catalogs）
(6)	tspaces.sql	创建表空间
(6)	rollback.sql	创建回滚段
(7)	tables.sql	创建表
(8)	aadpk.sql	增加主键(无效的)
(8)	adduk.sql	增加唯一键（无效的）
(9)	enablepk.sql	使能主键(和创建它们的索引)
(9)	enableuk.sql	使能唯一键(和创建它们的索引)
(10)	addfk.sql	增加外键（使能的）
(11)	crfk.sql	创建外键索引
(12)	addck.sql	增加检查约束
(12)	trig.sql	创建触发器
(13)	procs.sql	创建过程
(13)	funcs.sql	创建函数

最后需要注意的一点：你将在哪里放置一个大块的载入操作（在哪些步骤之间）？用来装入数据的一个正常的地方（如以SQL\*Loader为例），将正好是在创建你的表之后，但在你创建键值、索引和约束之前，也就是在第七步之后，但在第八步之前。为什么不在第八步之后装入？因为有使能键，索引和约束可能使装入数据操作速度相当慢。数据一致性怎么样？在你使能你的键和约束的同时，将发现那时有一些或所有的完整性和约束违背。然后你能修补数据，根据需要增加或全部重新安装。用这种方式，既不牺牲速度又不牺牲完整性。

## 8.4 小结

你已经了解了一般的软件开发周期和一个数据库特定的软件生命周期怎样映射到常用模型。你看到没有太多的不同；但是，从DBA的观点看，又有些不同。例如，在设计时，你的焦点更多地数据模型上而不是在进程上，尽管后者也是同等重要的，特别是对开发者和程序员而言。在实现时，你的焦点更多地集中在建立数据库结构上，正好与编写代码相反，后者是开发者的工作，也是程序员工作的难题。

你还要考虑有哪些主要的数据库职业和它们之间的不同，不仅在技能水平和经验上，还有相对于他们的同事，将他们安置在数据库生命周期的哪里更合适。正如你所见，许多这些职业经过一段时间，可能导致从一个向另一个过渡。因此，在数据库领域有相当大的职业增长空间。

接下来，足够详细地理解创建数据库的生命周期阶段，主要集中在灵活性和可伸缩能力来考虑Oracle DBA必须做什么。书本和其他资源经常给合适的事件顺序披上了外衣，使你只

见树木，不见森林。本章尽力明确说明这些情况，枚举 Oracle DBA 要遵循的主要创建步骤并强调他们重用能力，如何使它们松散、分离以及将它们写成脚本。对更多深层次 DBA 职责的概括，请参见第五部分“管理 Oracle 数据库”和第六部分“Oracle 接口和应用工具”。在第五部分包含的章节帮你管理存储、资源使用、安全性、备份 / 恢复和一致性。另外，在第六部分的章节中，包括能更有效地管理你的 Oracle 数据库的基本 Oracle 工具，包括 SQL\*Plus、PL/SQL 和 Oracle Enterprise Manager ( OEM )，最后一个应用工具是一个 GUI 工具，它将为 DBA 提供更多的管理能力，等效于许多 SQL\*Plus 脚本或 PL/SQL 程序的工作。