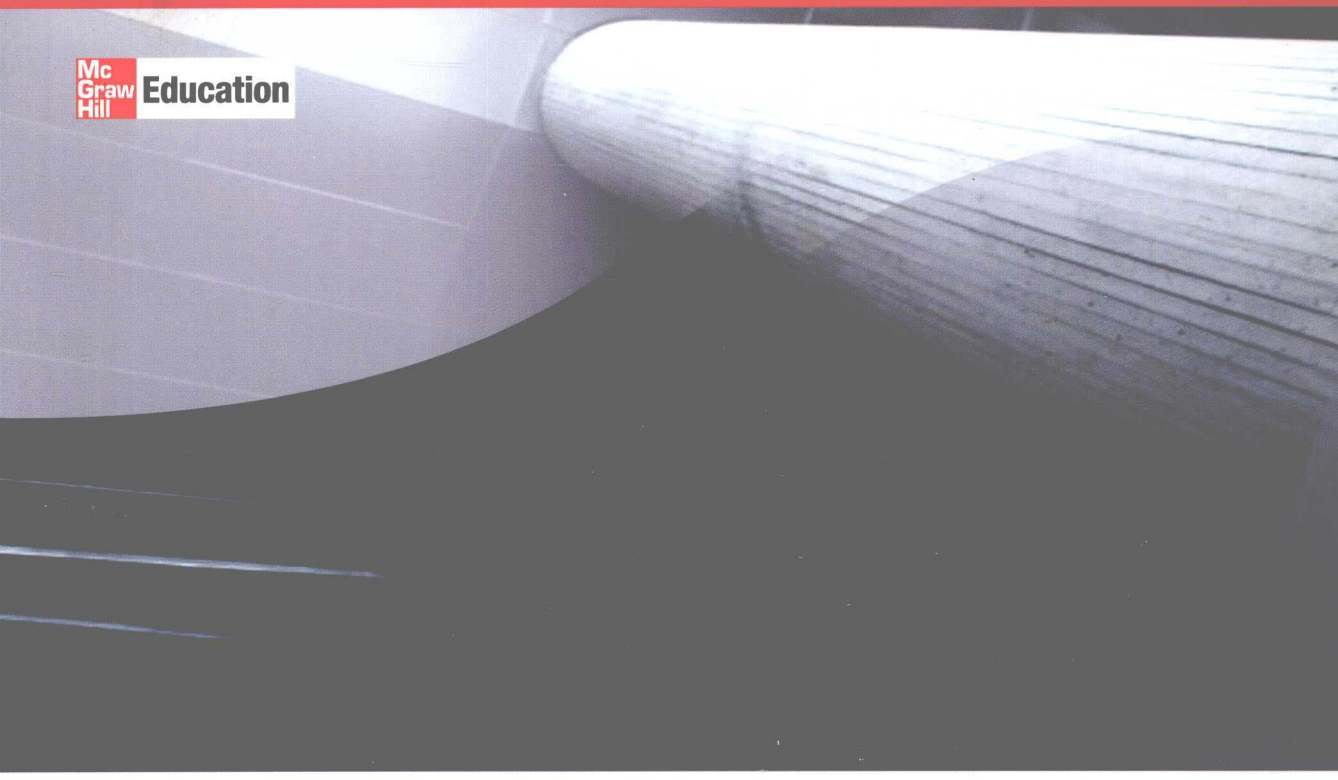
ORACLE



Mc Graw Hill Education

Oracle Database 11g & MySQL 5.6 Developer Handbook

MySQL

Oracle Database 11g & MySQL 5.6开发手册

[美] Michael McLaughlin 著
潘凝 译

清华大学出版社

Oracle Database 11g & MySQL 5.6 开发手册

[美] Michael McLaughlin 著
潘凝 译

清华大学出版社
北 京

Michael McLaughlin
Oracle Database 11g & MySQL 5.6 Developer Handbook
EISBN：978-0-07-176885-6

图书在版编目(CIP)数据

Oracle Database 11g & MySQL 5.6 开发手册/(美)麦克劳克林(McLaughlin, M.) 著；潘凝 译. —北京：清华大学出版社，2013.2

书名原文：Oracle Database 11g & MySQL 5.6 Developer Handbook

ISBN 978-7-302-31031-0

Ⅰ.①O…　Ⅱ.①麦…　②潘…　Ⅲ.①关系数据库系统—技术手册　Ⅳ.①TP311.138-62

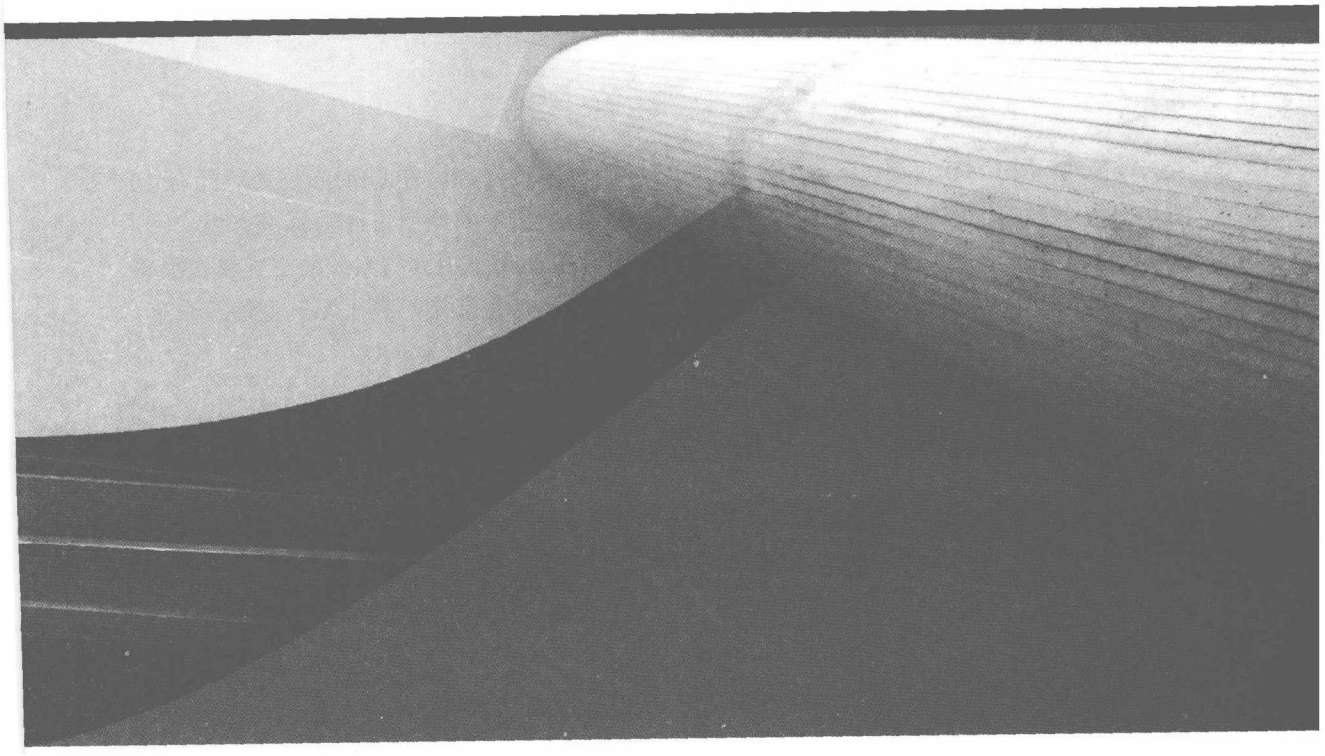
中国版本图书馆 CIP 数据核字(2012)第 304886 号

责任编辑：王　军　于　平
装帧设计：牛静敏
责任校对：蔡　娟
责任印制：沈　露

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 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：36　　**字　　数**：832 千字
版　　次：2013 年 2 月第 1 版　　**印　　次**：2013 年 2 月第1 次印刷
印　　数：1～3000
定　　价：79.80 元

产品编号：043728-01



作 者 简 介

Michael McLaughlin 是美国爱达荷州杨百翰大学的一位计算机信息技术系(Computer Information Technology，CIT)教授，也是 Oracle 公司的王牌员工。Michael 参与 Oracle 公司系列产品的研发已经有 20 年了，担任过开发人员、DBA 以及电子商务套件应用程序 DBA。

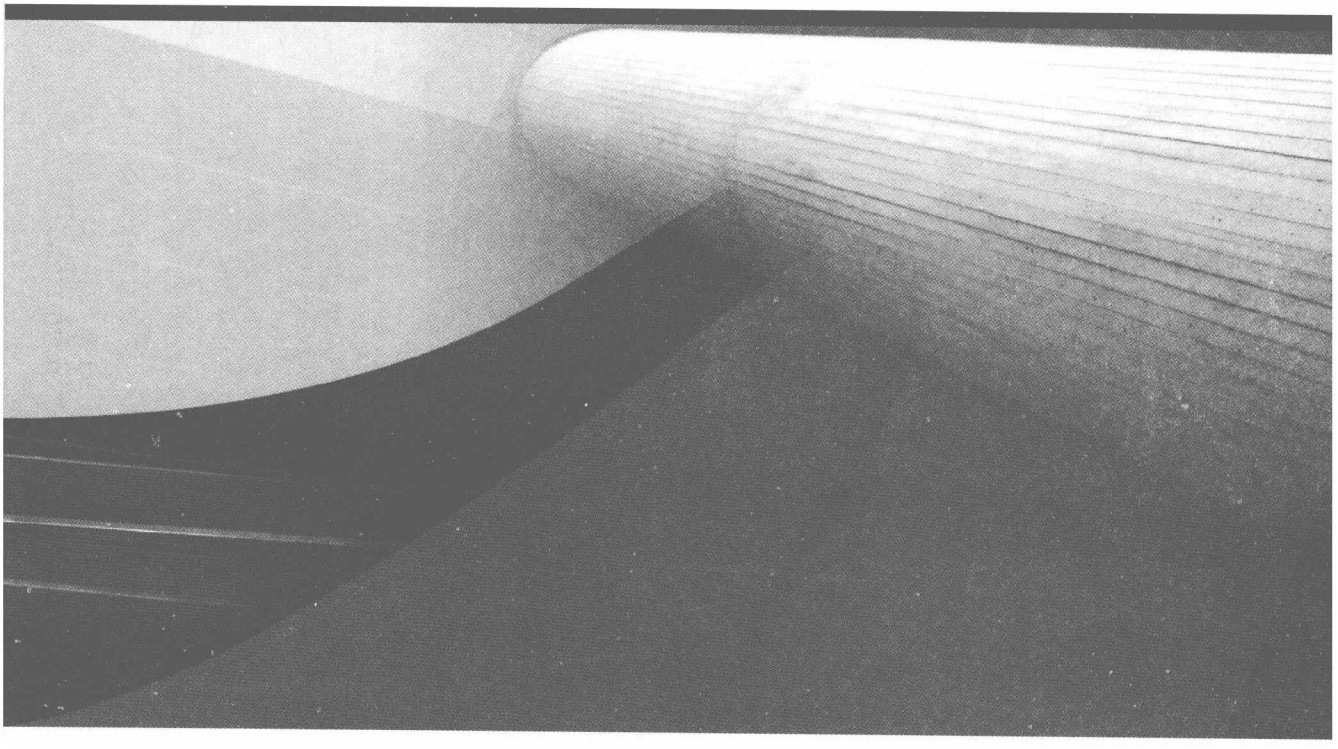
从 Oracle 公司离职前，Michael 在 Oracle Applications Division 担任 Senior Upgrade Manager。他在 Oracle 公司工作已经超过了 8 年，在咨询、支持和开发方面都有建树。他还是 ATOMS 事务架构(美国专利号#7 206 805 和#7 290 056)的发明者。这两项专利均归属 Oracle 公司所有。

他撰写了 6 本有关 Oracle 技术的著作，其中包括清华大学出版社引进并出版的《Oracle Database 11g PL/SQL 程序设计》和《Oracle Database 10g PL/SQL 程序设计》。

技术编辑简介

A. Scott Mikolaitis 是 Oracle 公司的一名应用程序设计师，在 Oracle 公司已经有 10 年以上的工作经历了。他正负责 Oracle Fusion 中 SOA 技术的原型设计与标准开发。

Scott 也很乐于研究基于 Java 的网页服务，还对在 Jabber 中人与系统的交互模式很感兴趣。在业余时间，他喜欢在家里鼓捣些 DIY 家居设备，也非常热衷于燃气型遥控汽车。

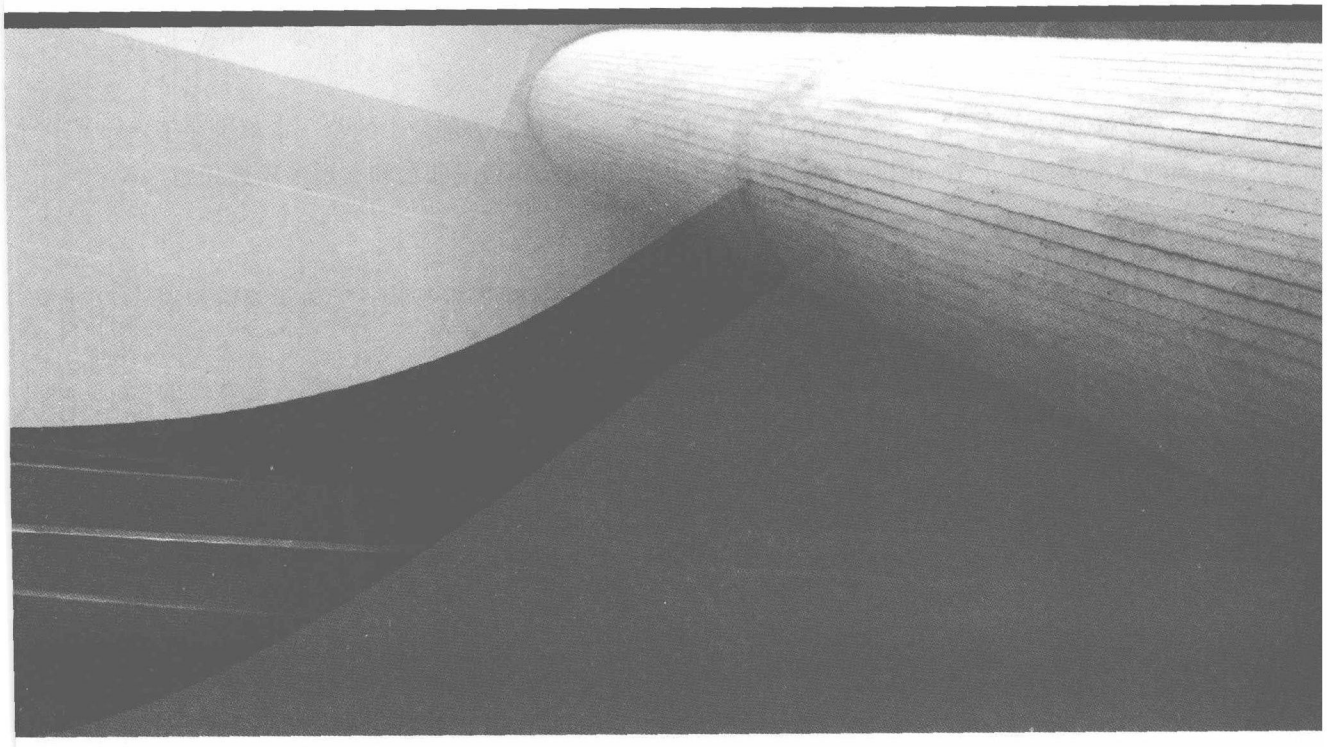


致　　谢

非常感谢 McGraw-Hill 出版公司的 Wendy Rinaldi 和 Stephanie Evans，他们不知疲倦地为本书出谋划策，并自始至终给我提供有力的支持。

特别感谢 Ian McLaughlin 帮我校对，感谢 Kent Jackson 帮我检查提纲，感谢 Michael Stokes、Benjamin Benson 以及 William Graham 对代码部分进行编辑和测试，感谢 Rex Barzee 与我共同探讨内容组织方面的想法，还要感谢 CIT 系的讲座教授 Art Ericson 对本书的支持。

最后要感谢的是本书的制作部门，他们勤勤恳恳地检查每一处细节，尽全力将各个部分联系为一个整体。我要感谢 Lisa Theobold 帮我对本书加以润色、Bev Weiler 找出了疏漏和错误并调整了内容的组织方式，还有 Sandhya Gola 在编辑、校对以及最终成书准备的过程中将一切都打理得井井有条。



前　言

前言向来都是最后动笔完成的，不过有时候笔者倒觉得应该首先把前言写好。这次笔者就在动笔之前就打好了前言的草稿，这让笔者胸有成竹地知道一切正按照计划进行。制作人员也在帮助笔者进行校对，只要看看“致谢”那一页你就知道他们付出了多少心血和才能了，要让一本高品质的图书最终付诸印刷，这些努力都是至关重要的。

本前言中包含了如下几个方面：

- “本书提纲”部分将每一章的内容总结为一两句话，可供你快速浏览本章内容。
- “字典”部分将会告诉你本书中各种命名惯例的基本规则，并给出了一些可以减少代码调试时间的建议。
- “数据模型”部分描述了本书所有例子的基础，并告诉读者如何寻找、创建并开始操作学生数据库的代码。

本书提纲

本书由两大部分组成："开发组件"、"SQL 开发"。这两大部分中的每一章都讲述了一系列基本法则，最后还附有习题。每一组习题中都有 10 道是非题和 5 道多项选择题。

第 I 部分：开发组件

- 第 1 章"架构"解释了 Oracle 11g 和 MySQL 5.6 的开发架构，并着重对客户端与服务器环境的各个方面作了比较。
- 第 2 章"客户端接口"解释并演示了 SQL*Plus 和 MySQL Monitor 客户端软件的基本使用方法。
- 第 3 章"安全模型"解释了数据库服务器的安全屏障与数据控制语言(Data Control Language，DCL)命令，让你在数据库服务器中管理用户和账户权限。
- 第 4 章"事务模型"解释了 ACID 兼容事务的本质特点，并用 INSERT、UPDATE 和 DELETE 语句演示了双步提交(Two-phase Commit，2PC)的过程。
- 第 5 章"约束"，解释了 5 种基本的数据库级别约束，并阐述了检查、非空、唯一、主键和外键约束。

第 II 部分：SQL 开发

- 第 6 章"创建用户和结构"解释了如何创建用户、数据库、表、序列和索引。
- 第 7 章"修改用户和结构"解释了如何修改用户、数据库、表、序列和索引。
- 第 8 章"插入数据"解释了如何向表中插入数据。
- 第 9 章"更新数据"解释了如何更新表内的数据。
- 第 10 章"删除数据"解释了如何从表中删除数据。
- 第 11 章"查询"解释了如何从单表、从两张或多张联表以及通过集合运算符从两次或多次联询中查询数据。
- 第 12 章"合并数据"解释了如何从外部表或源文件中导入非规格化数据以及在规格化的表中插入或更新表内记录。
- 第 13 章"PL/SQL 程序设计语言"解释了使用 PL/SQL 编写事务性代码块的基本方法。
- 第 14 章"SQL/PSM 基础"解释了使用 SQL/PSM 编写事务性代码块的基本方法。
- 第 15 章"触发器"解释了如何在 Oracle 和 MySQL 数据库中编写数据库触发器。
- 附录中包含了每章末尾习题的解答。

字典

编写程序的方法有很多种，对于不同的编程语言其方法通常也不尽相同。SQL、PL/SQL 和 SQL/PSM 代码也都具有这样的共性：它们是不同的语言，需要不同的方法来实现。

SQL 字典

对于 SQL 语句，笔者的建议是要将关键字排列在左边。也就是说，要将 SELECT 列表逗号以及 WHERE 子句的逻辑 AND [NOT]或 OR [NOT]语法放在左边，因为这样可以迅速地阅读代码寻找错误。这些建议都是很简单的，不过对于如何编写连接语法(join-syntax)，笔者的建议会复杂得多，因为可以使用 ANSI SQL-89 或 ANSI SQL-92 标准来编写连接。ANSI SQL-89 允许将表组织为逗号分隔的列表，而 ANSI SQL-92 能使用关键字来指定连接的类型。

对于连接语法有如下建议：

- 总是使用表别名，因为它们可以确保，当 SELECT 列表可能返回带有相同名称的两列或多列时也不会遭遇含糊不清的列错误，当连接了享有相同列名称的表时就可能发生这种情况。在编写单表的查询时，使用别名也是一种很好的习惯，因为之后你可能会通过连接添加另一张表。第 11 章讲解了与本建议相关的 SELECT 语句和语法。
- 在使用 ANSI SQL-89 及逗号分隔表时，应该将每张表放在独立的行中，相分隔的列放在左侧，就像 SELECT 列表列那样。这样有助于更直观地阅读程序。但这并不适用于在第 9 章和第 10 章中的多表 UPDATE 和 DELETE 语句，你应该分别参考这两章的例子来编写。
- 在使用 ANSI SQL-92 时，应该使用 ON 或 USING 子子句将连接条件放在 FROM 子句内部。在处理带有 ON 或 USING 子子句的 FROM 子句时，大部分开发者都经常使用两种方法：在较小的两表连接或者最多三表连接中，将 ON 或 USING 子子句放在同一行的连接语句后面；而在较大的三表或更多表的连接中，则将 ON 或 USING 子子句放在连接语句所在行的下面。当连接涉及多列时，要将逻辑 AND [NOT]或 OR [NOT]语法靠左对齐以便更直观地阅读代码。
- 在使用 ANSI SQL-89 语法将连接语句包含在查询的 WHERE 子句中时，最好将连接写成子句中的第一条语句。ANSI SQL-89 的 WHERE 子句也包含了过滤器结果集合的语句，而 ANSI SQL-92 的 WHERE 子句只包含了过滤语句。WHERE 子句中的语句是用逻辑 AND [NOT]或 OR [NOT]语法来连接的。AND 逻辑操作符优先级要高于 OR 逻辑操作符，除非使用圆括号修改了操作顺序。应该将逻辑 AND [NOT]或 OR [NOT]语法靠左对齐以便直观地阅读代码。

- ANSI SQL-92 允许使用完整描述的关键字或者只使用必需的关键字。虽然笔者们中大部分人都喜欢尽可能输入最少的字符，但最终笔者们的代码会到技术支持人员的手上，因此其清晰程度可以有助于避免许多轻率的 bug 报告。所以，应该考虑使用 INNER JOIN 来替换 JOIN，用 LEFT OUTER JOIN 或 RIGHT OUTER JOIN 来替换 LEFT JOIN 和 RIGHT JOIN，用 FULL OUTER JOIN 来替换 FULL JOIN。本书将这些语法都缩写了，原因只是为了满足印刷的需要，本书的代码页面篇幅只能容纳每行 67 个字符(否则就要缩小字体，这样的话读起来就吃力了)。

有关工具的一点说明

虽然本书主要讲解的是在命令行下编写 SQL(因为在 C++、C#、Java 或 PHP 程序中都需要使用命令行)，但是 CASE(Computer-aided Software Engineering，计算机辅助软件工程)在与数据库进行交互的方面是很有效的。它们可以帮助读者发现可用的语法和解决方法，但不应该完全依赖于它们。

简言之，要用工具学习，但不能成为工具的奴隶。要时刻学习工作原理以及可能的改进方法。如果你做到了这一点，那么 CASE 工具就会成为一个帮助你完成任务的好帮手，而不是阻碍你发展的诅咒。

既然笔者已经写到了，那么就让笔者来分享一些经验来谈谈如果不按照语法建议会有怎样的后果。笔者在 IBM 公司的 Santa Teresa 实验室(现在是 IBM 的硅谷实验室)的导师在 1985 年教了笔者如何编写 SQL(实际上是 SQL/DS[Structured Query Language/Data System，结构化查询语言/数据系统])。他让笔者将逗号放在左侧，从而节省了寻找漏掉逗号所浪费的时间。笔者忽略了他的建议，将逗号放在右侧也就是行的末尾，过了几个月才终于意识到导师是对的。在那一周他教了笔者一句至理名言："优秀的程序设计遵循简单的原则。"

如今在学校里，笔者每个学期都会跟学生们强调这一建议。有些学生欣然接受，但有的固执己见。那些固执己见的学生从头到尾都在纠结于语法，因为他们总是在尝试从 SQL 语句中找出遗漏的逗号或者其他组件。SQL 并不是一种简单易学的语言，因为它要求你创建一个数据的空间分布图，这并不是所有开发者都能立刻学会的技巧。有时候想要理清关系型数据库中数据之间的关系需要耗费相当多的时间。但如果你能尽量维护好语句的清晰度和方法的一致性，并持续选择使用可移植的 SQL 语法，那么时间长了就会越来越轻松。

PL/SQL 和 SQL/PSM 存储程序

PL/SQL 和 SQL/PSM 是完全成熟的程序设计语言。它们用于编写存储在数据库中的程序，该数据库将 SQL 语句的集合管理成完整的事务。

第 13 章、第 14 章和第 15 章讲解了存储程序设计语言，其中包括了各种类型的变量。在一些组织中，变量命名惯例可能会存在争议，因为许多开发人员都认为变量应该在语义上具有意义，也就是说其名称应该描述其用途或目的。但反对命名惯例的人们认为，这些惯例(例如前缀)会降低代码的可读性。不过，这种争论和生活中所遇到的其他争论一样——都是概念上的冲突。争论的双方都有其合理的优势，在某些场合也都具有优于对方的特点。据笔者所知，关键就是找出平衡点，既能为公司或企业增强稳定性，也能提供有意义的变量名称。

本书将尝试使用一贯的前缀。在某些地方笔者会选择在变量名称中实现语义上的清晰(例如第 2 章中 Oracle 数据库的会话或捆绑变量:whom)。笔者认为使用前缀可以增强代码的可读性，而且建议使用表 1 中的前缀，该表在 Oracle 和 MySQL 数据库中存在映射关系。

表 1　PL/SQL 的变量前缀

前缀	示　例	描　述
cv	cv_input_var	代表游标参数变量。这些是 PL/SQL 存储程序中游标的按值传递输入参数。但 SQL/PSM 存储程序并不支持游标参数
lv	lv_target_var	代表在 PL/SQL 或 SQL/PSM 存储程序中定义的本地变量
pv	pv_exchange_var	代表 PL/SQL 和 SQL/PSM 存储函数和过程的参数。它们并不只是专门的输入参数，因为 PL/SQL 支持存储函数和过程的输入和输出参数，而 SQL/PSM 只支持存储过程的输入和输出参数(正如第 14 章中所说，它并不支持函数的输入输出参数)
bv sv	bv_global_var 或 sv_global_var	代表会话变量或 Oracle 数据库的捆绑变量；它们是在客户端连接数据库期间内的全局变量。Oracle 数据库允许使用在块中的变量名称前面加上冒号(:sv_global_var)，这种方法在匿名块之间共享这些变量的值，它们被称为捆绑变量(因此有了另一种:bv_global_var)。MySQL 数据库允许在命名块中引用它们，并且需要添加一个@符号将变量名称变为:@sv_global_var。可以在第 2 章和第 14 章中找到更多有关 MySQL 数据库会话变量的内容

一些高级的变量数据类型也被称为复合变量，既需要前缀，也需要后缀。前缀指出了复合变量的类型。这些要求是 Oracle 数据库中所独有的。表 2 给出了对于 Oracle 数据库的复合数据类型的后缀的建议(以下划线开头)，表 2 还给出了后缀的长、短名称。

通常情况下，在复合数据类型上使用前缀是广为接受的做法，因为它们是用户定义类型(user-defined type，UDTs)。不过这并不是 PL/SQL 程序设计语言中的规定或要求。在第 13 章中总结了 PL/SQL 程序设计语言的细节，也可以在《Oracle Database 11g PL/SQL 程序设计》中了解更多完整的介绍。

表 2 PL/SQL 的变量后缀

前缀		描述
长	短	
_ATABLE _AARRAY	_ATAB, _AA	_ATABLE、_AARRAY、_ATAB 和_AA 是用于描述 PL/SQL 的相关数组的。笔者喜欢使用_ATABLE 或_ATAB 后缀，因为其他前缀看起来并不直观，并且需要在代码的文档中进行说明
_CURSOR	_CUR _C	_CURSOR、_CUR 和_C 是用于描述基于在 PL/SQL 的本地声明块或包规格中定义的游标结构的变量的。笔者喜欢使用_CURSOR 或_C 后缀。也可以声明一个游标结构，这会在第 13 章中讲解
_EXCEPTION	_EXCEPT _EX _E	_EXCEPTION、_EXCEPT、_EX 和_E 是用于描述 PL/SQL 中的用户定义异常的。笔者喜欢使用_EXCEPTION 或_E 后缀
_OBJECT	_OBJ _O	_OBJECT、_OBJ 和_O 是用于描述 SQL 和 PL/SQL 中的 UDT 的。对象类型就像 PL/SQL 中的 RECORD 数据类型，后者是用于记录数据结构的。它们两者是有区别的，因为它们是模式级 SQL UDT 而不仅仅是 PL/SQL 的 UDT。对象类型也可以是可实例化的对象，例如 C++、C#和 Java 类，可以参考《Oracle Database 11g PL/SQL 程序设计》一书的说明。笔者喜欢使用_OBJECT 或_O 后缀
_NTABLE _TABLE	_NTAB _TAB	_NTABLE、_TABLE、_NTAB 和_TAB 是用于描述嵌套表的，在 SQL 和 PL/SQL 中这些嵌套表是集合类型。它们就像是列表，因为在集合中所能存放的元素数量上并没有上限。笔者喜欢使用_TABLE 或_TAB 后缀，因为嵌套表是与其他程序设计语言中的列表最相似的集合
_RECORD	_REC _R	_RECORD、_REC 和_R 用于描述 PL/SQL 中专门的 UDT 的。它们是 PL/SQL 对记录数据结构的实现方法。它们可以是 PL/SQL 集合的元素，但不能是 SQL 集合的元素。笔者喜欢使用_RECORD 或_R 后缀，因为一个是完整描述，另一个是首字母缩写，但许多开发者都喜欢用_REC
_TYPE	_T	_TYPE、_T 是用于描述 UDT 的，例如在第 13 章中所讲述的普通标量数据类型的子类型。这些前缀笔者都使用过，但在代码仓储中似乎_TYPE 更常见
_VARRAY	_VARR _VA	_VARRAY、_VARR、_VA 是用于描述 VARRAY(对于这种 Oracle 数据类型，笔者将其理解为虚拟数组 virtual array)的。VARRAY 是最类似于程序设计语言的标准数组的集合，因为它具有最大大小，且必须有一个有序的索引值。它可以用于定义 SQL 和 PL/SQL 的集合。笔者喜欢使用_VARRAY 或_VA 后缀，因为_VARR 与通用变量缩写太像了

PL/SQL 是一种强类型语言，PL/SQL 和 SQL/PSM 程序都是块程序。块程序使用关键字来启动和终止程序单元，而不是使用在 C++、C#、Java 或 PHP 中的花括号。在 GeSHi (Generic Syntax Highlighter，通用语法高亮)库中可以看到，PL/SQL 和 SQL/PSM 的块关键字都是大写字母形式的，本书也一直遵循了这一惯例。

数据模型

数据模型是一个小型音像店。可以在本书出版社的网站上找到 Oracle 和 MySQL 数据库中创建和设置数据模型的源代码，在本书撰写时网址为 http://www.mhprofessional.com/product.php?isbn=0071768858。

图 1 给出了示例程序中的基础表或核心表。该图是用 MySQL Workbench 5.2 及物理模式的反向工程绘制的。如果你并不熟悉 MySQL Workbench，那么建议学会它，因为这是一个非常棒的工具，可以有效地支持你进行 MySQL 数据库的建模和设计工作。

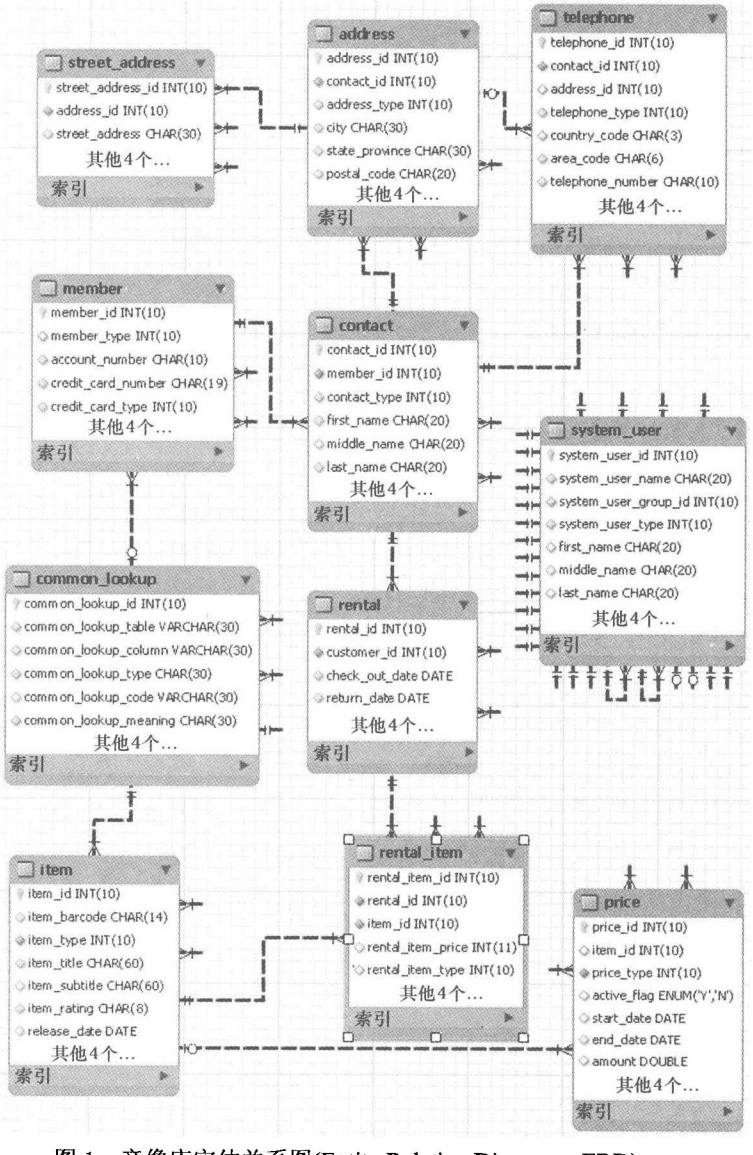


图 1　音像店实体关系图(Entity Relation Diagram，ERD)

对于模型中的一张表可能需要做一些解释，那就是 common_lookup 表。common_lookup 表是一张多表的表，如图 2 所示。

common_lookup_id	common_lookup_table	common_lookup_column	common_lookup_type	common_lookup_meaning
1	SYSTEM_USER	system_user_id	SYSTEM_ADMIN	System Administrator
2	SYSTEM_USER	system_user_id	DBA	Database Administrator
3	CONTACT	CONTACT_TYPE	EMPLOYEE	Employee
4	CONTACT	CONTACT_TYPE	CUSTOMER	Customer
5	MEMBER	MEMBER_TYPE	INDIVIDUAL	Individual Membership
6	MEMBER	MEMBER_TYPE	GROUP	Group Membership
7	MEMBER	CREDIT_CARD_TYPE	DISCOVER_CARD	Discover Card
8	MEMBER	CREDIT_CARD_TYPE	MASTER_CARD	Master Card
9	MEMBER	CREDIT_CARD_TYPE	VISA_CARD	VISA Card
10	ADDRESS	ADDRESS_TYPE	HOME	Home
11	ADDRESS	ADDRESS_TYPE	WORK	Work
12	ITEM	ITEM_TYPE	DVD_FULL_SCREEN	DVD: Full Screen
13	ITEM	ITEM_TYPE	DVD_WIDE_SCREEN	DVD: Wide Screen
14	ITEM	ITEM_TYPE	NINTENDO_GAMECUBE	Nintendo GameCube
15	ITEM	ITEM_TYPE	PLAYSTATION2	PlayStation2
16	ITEM	ITEM_TYPE	XBOX	XBOX
17	ITEM	ITEM_TYPE	VHS_SINGLE_TAPE	VHS: Single Tape
18	ITEM	ITEM_TYPE	VHS_DOUBLE_TAPE	VHS: Double Tape
19	TELEPHONE	TELEPHONE_TYPE	HOME	Home
20	TELEPHONE	TELEPHONE_TYPE	WORK	Work
21	PRICE	ACTIVE_FLAG	YES	Yes
22	PRICE	ACTIVE_FLAG	NO	NO

图 2　公共查找表(多表的表)

唯一识别行的一系列属性(列)就是自然键，它由表和列名称以及类型组成。类型是大写字符串，并加上了下划线进行连接，这使得在查询这些查找集合时就会更方便(在 Oracle 数据库中，所有的元数据字符串都是以大写文本存储的，而 MySQL 数据库则以小写文本字符串来存储元数据)。有意义的列可以为读者提供信息供最终用户在下拉列表中进行选择。

common_lookup 表的主键是一个代理键列，common_lookup_id(遵循使用表名称和 _id 后缀来组成主键列名称的习惯)。该值的副本被存储在表和列中，例如 item 和 item_type。在这种设计方案中，可以在同一个地方将 XBOX 显示值修改为 Xbox，而所有的代码模块和表值都不会被改动。这是一种强大的建模工具，因为它可以避免在网页表单(嵌入选项)中放置例如性别、人种或是/非答案等组件，而且可以减少应用程序在部署之后的管理成本。

目　录

第 I 部分　开发组件