ORACLE

ORACLE PRESS™ — EXCLUSIVELY FROM McGRAW-HILL/OSBORNE

Oracle Database 10g PL/SQL Programming

Oracle Database 10g PL/SQL 程序设计

ORIGINAL • AUTHENTIC
Oracle Press
ONLY FROM OSBORNE

10 YEARS
Oracle Press
2.5 MILLION SOLD

Scott Urman
(美) Ron Hardman 著
Michael McLaughlin
彭 珲 译

McGraw Hill Education
清华大学出版社

Oracle Database 10g
PL/SQL程序设计

Scott Urman
(美) Ron Hardman 著
Michael McLaughlin
彭 珲 译

清华大学出版社
北 京

Scott Urman，Ron Hardman，Michael McLaughlin
Oracle Database 10g PL/SQL Programming
EISBN: 0-07-223066-5

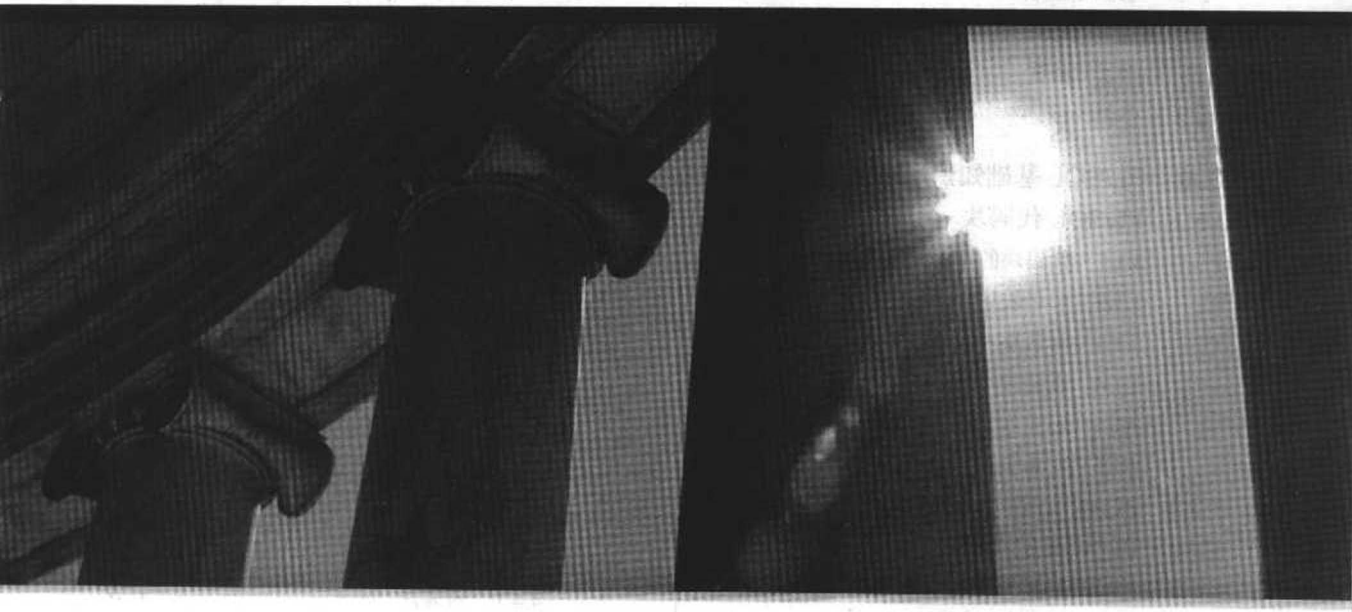
图书在版编目(CIP)数据

Oracle Database 10g PL/SQI. 程序设计/(美)俄曼(Urman,S). (美)哈德曼(Hardman,R),(美)麦克罗克林(Mclaughlin,M.)著；彭珲译.
—北京：清华大学出版社，2005.11
书名原文：Oracle Database 10g PL/SQL Programming
ISBN 7-302-11892-2

Ⅰ.O… Ⅱ.①俄… ②哈… ③麦… ④彭… Ⅲ. 关系数据库—数据库管理系统，Oracle 10g—程序设计
Ⅳ.TP311.138

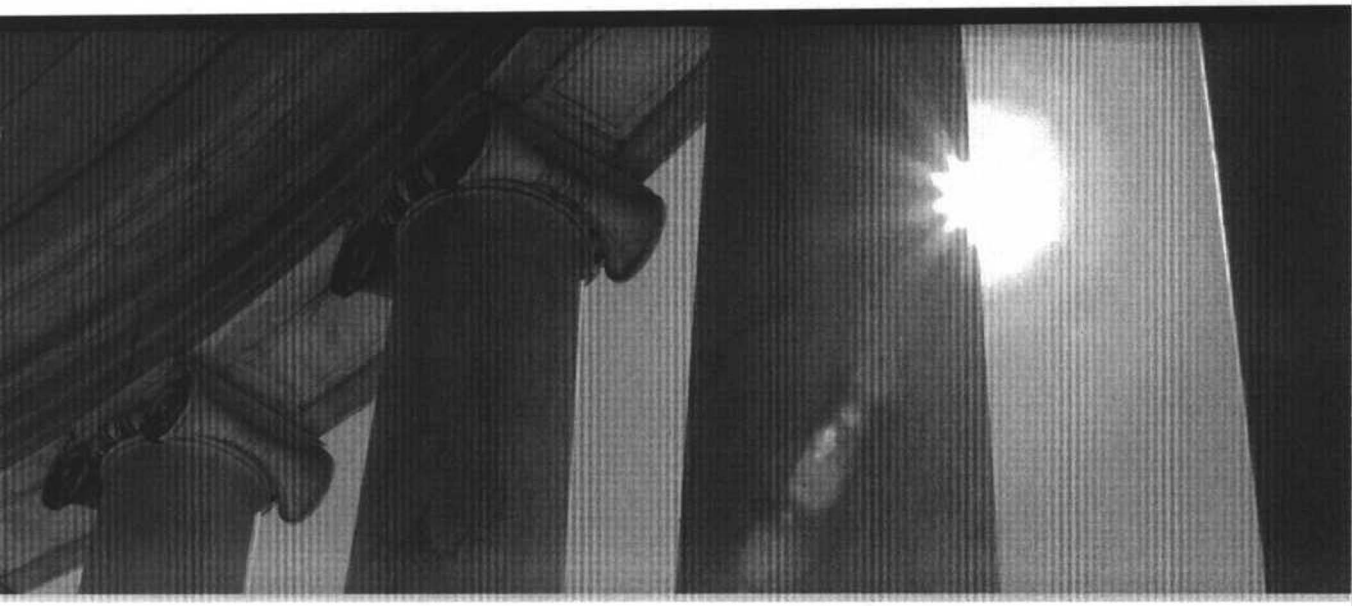
中国版本图书馆 CIP 数据核字(2005)第 111724 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **客户服务**：010-62776969
组稿编辑：曹 康
文稿编辑：李 阳
封面设计：孔祥丰
版式设计：孔祥丰
印 刷 者：北京市清华园胶印厂
装 订 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 **印张**：42.25 **字数**：1082 千字
版 次：2005 年 11 月第 1 版 2005 年 11 月第1 次印刷
书 号：ISBN 7-302-11892-2/TP·7722
印 数：1~4000
定 价：85.00 元

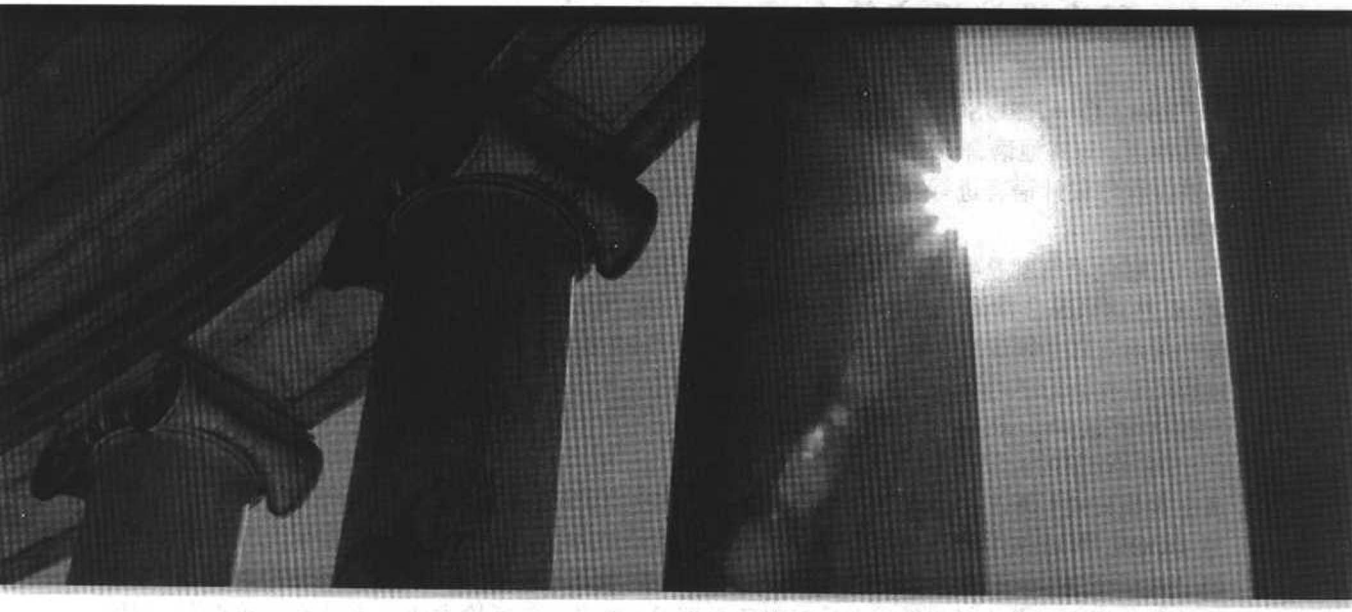


目　录

第Ⅰ部分　绪　论



第Ⅰ部分　绪　　论



第1章

PL/SQL 入门

我们都见过代码写得很漂亮但运行起来很糟糕的应用程序。看看那些编写“非常精美”的病毒，或那些现在已经倒闭的软件公司开发的那些非常华丽但又毫无用处的应用程序吧！程序设计绝对不仅是语法、句法的堆积，它是一种很神圣的职业，在这种职业中，一个人的知识可以与他的聪明才智、交际能力、敬业态度和纪律完美地结合在一起，也只有这种结合，才能成就一个人的事业，才能让他编写出一流的应用程序。

在本书中，我们的重点不仅仅在程序设计的语法与规则上。在接触到某些新功能的时候，我们一般都会有这样的疑问：“我为什么要使用它？”本书将会回答这个问题。我们讨论的范围将超越Oracle能做什么，而是还将进一步地介绍它是“怎样”做的，以及它“为什么”要这样做。

在本章中，将首先明确本书的范围，将介绍以下几点：

- SQL 以及它与关系数据库的交互作用
- PL/SQL 是怎样使用 SQL 来提高性能的
- 程序设计的概念，并且将过程化语言与面向对象的程序设计进行比较
- PL/SQL 的历史和功能特性
- 该语言的优缺点
- 怎样利用本书的其他章节并从精心组织的内容中最大获益

1.1 程序设计语言简介

Java、C++、PL/SQL 和 Visual Basic 等都是当今非常流行的程序设计语言，每一种语言都各具特点，与其他语言截然不同，但是其中的某些语言却具有一些共同的特征。根据这些共性，可以将程序设计语言进行简单的分类。可以将程序设计语言分为两大类：过程化语言和面向对象语言。

过程化语言都是线性的语言，像 PL/SQL 和 Visual Basic 等。它们以 begin 开始，以 end 结束。虽然这只是非常简化的定义，但是它指出了过程化语言与面向对象语言之间最根本的区别。每一条语句都必须等到它之前的语句执行完毕以后才能执行。对很多初级程序员来说，最好先从过程化语言开始学习程序设计。程序必须执行的一连串步骤，就是代码执行的过程—— 一步一步地执行(step-by-step)。

面向对象的程序设计(Object-oriented programming，OOP)语言，例如 Java 和 C++等，从本质上来说相对抽象一些。OOP 语言使用的是称作对象(object)的结构。例如，可以创建一个名为 BOOK 的对象，而不是通过编写代码，直接从数据结构中将与书有关的信息全部堆积在一起。每一个 BOOK 对象都有一些属性：页数、定价以及题目等。属性描述对象的特征，而方法就是作用在对象上的动作。方法对对象数据进行操作，包括检索或修改对象数据的值。例如，如果您想更改书的定价，那您就可以调用一个方法来完成这个修改任务。这一点与过程化语言有较大的区别—— 在过程化语言中，您需要执行一系列步骤才能实现相同的目的。

另一种较少见的双重分类将 PL/SQL 认为是过程化程序设计语言，也可以将其认为是面向对象的程序设计语言。Oracle 8 引入了对象的概念，但是在最初的发行版中，并不支持像继承、类型演化和动态方法分派等这些高级的面向对象特性。在 Oracle 9iR1 中，Oracle 开始把完全支持 PL/SQL 的面向对象程序设计功能作为一个主要卖点。到 Oracle 10g 发行时，Oracle 已经完全支持绝大部分面向对象特性了。

注意：

本书将在第 14 章、第 15 章详细介绍这里提到的面向对象特性。

初级程序员注意事项

像大多数程序开发人员一样，笔者刚学习编程使用的也是 Basic。Basic 语法很容易学习，但是 Basic 使用的那些程序设计“原理”同样适用于其他大多数程序设计语言。相信您也会发现这些原理同样适用于 PL/SQL。

在 PL/SQL 的特性中，笔者所喜欢的不是它与数据库的紧密结合(虽然它与数据库结合得非常完美)，不是它拥有的高级语言的概念和功能(学完本书后，您会被 PL/SQL 所能实现的事情所折服)，也不是它提供的某一种功能。笔者所钟爱的是它结构化的程序设计方法。每一个 BEGIN，都会有一个 END；而每一个 IF，都会有一个 END IF。

作为一名教授过很多初学编程(不仅仅是刚刚开始学习 PL/SQL 编程)的学生 PL/SQL 这门课程的教师，笔者向您保证您一定能学会这种语言。PL/SQL 是一种结构化的、线性的但又不是十分宽松的程序设计语言。它是一门优秀的程序设计语言。您能在其中学到结构和规则。如果不遵守这些规则，在试图运行代码的时候，马上就会看到下面这样的反馈信息：

```
Warning: Procedure created with compilation errors.
```

提示：

显然，结构并不能保证代码的优秀性，它只能使这种语言更易于学习。请谨慎地使用良好的组织形式、采用适当的命名约定、将操作文档化，并进行实践、实践再实践。不能为了走一点捷径而降低代码的效率和可维护性。使用任何一种语言，都可能编写出编译以后变得十分可怕的代码。

也许您已经注意到了一点：最优秀的程序员并不一定是最聪明的程序员。那些顶尖的程序员一般都是优秀的外交家，他们有将自己完全融入到用户和消费者之间的能力。在设计阶段，这种能力至关重要。您需要与项目经理、其他开发人员、DBA、终端用户、QA(质量评价)工程师及管理部门进行交流。在整个系统的开发生命周期中，每组人员的目标不同，也都会对您提出不同的要求。您与人们进行交流的能力和态度，将会直接导致项目的成功或失败；这种交际能力和态度，也最终决定您在这个行业中能够走多远。

1.2 什么是 PL/SQL

那么，什么是 PL/SQL 呢？它是 SQL 向过程化(有时是面向对象的)程序设计语言方向上进行的扩展。它是由 Oracle 开发，且为专用于 Oracle 的一种程序设计语言。如果您对另一种程序设计语言 Ada 比较熟悉的话，就会在 PL/SQL 中发现很多类似的地方。二者相似的原因很简单：PL/SQL 就是从 Ada 发展过来的，它实际上借用了很多 Ada 的概念。

PL/SQL 中的 PL 的意思就是过程化语言。PL/SQL 是一种专用语言，不能在 Oracle 数据库以外的地方使用。它是第 3 代语言(third-generation language，3GL)，支持与其他 3GL 语言相似的程序构造块，包括变量的声明、循环的使用以及出错处理等。从其根源上说，PL/SQL 只是一种过程化语言。但是，正如我们在前面已经提到的一样，现在也可以将 PL/SQL 认为是一种面向对象的程序设计语言。那么，我们是否应该将 PL/SQL 这个名称更改为 PL/OO/SQL 呢？

1.2.1 结构化查询语言 SQL

PL/SQL 中 SQL 的意思就是结构化查询语言。我们使用 SQL 语句完成数据的查询(SELECT)、插入(INSERT)、更新(UPDATE)和删除(DELETE)。我们使用 SQL 来创建并管理对象与用户，同时控制对数据库实例的访问权限。

SQL(可以将其读作 sequel，也可以直接按其字母缩写进行发音)是一个数据库的入口(或叫窗口)。它是一种朝着易学和易用方向发展的第 4 代语言(fourth-generation language，4GL)。SQL 的基本语法也并不是 Oracle 创建的，它起源于在 20 世纪 70 年代的初期，汇集了 E.F. Codd 博士和 IBM 的工作研究成果。美国国家标准化协会(American National Standards Institute，ANSI)认可了 SQL 并公布了该语言的标准。

Oracle 支持 ANSI 标准的 SQL，但是在它的 SQL*Plus 实用程序中添加了它自己的一些东西。它通过 SQL*Plus，支持不属于 ANSI 标准的一些附加命令和功能。在多种平台上都可以使用 SQL*Plus 实用程序：

- **命令行**：从 Unix 提示符或 DOS 提示符下键入命令；
- **图形用户界面(GUI)**：SQL*Plus 客户、SQL Worksheet 或企业管理器；
- **Web 页面**：iSQL*Plus、10g 的企业管理器。

只需要安装一台客户机，我们就可以对网络进行配置连接到远程数据库。Oracle 10g 通过基于浏览器的企业管理器和 iSQL*Plus(这二者都是在安装时配置的)使配置工作更加简化。

1.2.2 关系数据库简介

SQL 是数据库的一个窗口，那么到底什么是数据库呢？广义上，存储数据的任何东西都可以叫作数据库。最简单的电子数据库可以是一个电子表格，或一份字处理文档。

也许您已经想到，如果在电子表格或字处理文档中存储大量的数据，那么数据可能很快就会泛滥成灾。这种单维数据库没有有效的方法来过滤冗余数据、保证数据录入的一致性，也不能处理信息检索的问题。

Oracle 是一个关系数据库管理系统，或简称 RDBMS。关系数据库都是在表中存储数据的。表由多个列组成，在列上定义存储在其中数据的类型(字符型数据或数字型数据等)。表至少包含一个列。在表里存放数据的时候，数据是逐行存储的。这适用于所有的关系数据库如图 1-1 所示。

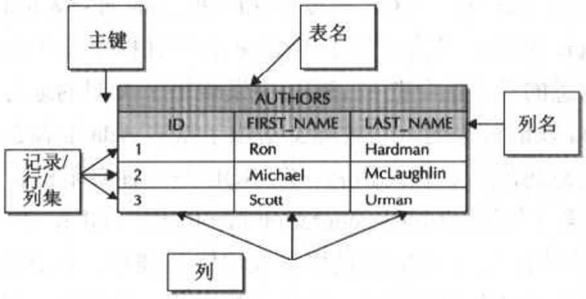


图 1-1 表结构

在 Oracle 中，表归用户或模式拥有。模式就是数据库用户所拥有对象(例如表)的一个集合。在同一个 Oracle 数据库中，是允许存在名称相同的两个表的，只要这两个表分别属于不同的用户。

其他的供应商一般不需要遵守这种方法。例如 SQL Server，它就使用一套不同的概念。其实，SQL Server 数据库更像 Oracle 数据库的一个模式，而 SQL Server 服务器则更像 Oracle 数据库。但是结果是一样的。对象(例如表)，总是会有它的一个拥有者。

也可以像电子表格一样，将我们所有的数据都存储在一个表中，但是这样就不能充分利用 Oracle 关系数据库的特性。例如，如果没有作者信息，那么存储 Oracle 出版社图书信息的表就会变得不完整。可能存在一个作者编写多部图书的情况。在平面文件，或单一表、模型中，会将作者重复列出多次。这种冗余其实是可以避免的，只要我们将数据分成两个不同的表，并使用一个将相关数据链接在一起的列就可以了。图 1-2 演示了将这一个表分割成两个单表的过程。

图 1-2 中有两个表，AUTHORS 和 BOOKS。作者信息中存储作者的姓(first name)和名(last name)。为每一行数据都定义了一个 ID，这个 ID 用来保证数据的惟一性和非空性(null 的意思是空，not null 的意思就是非空)。

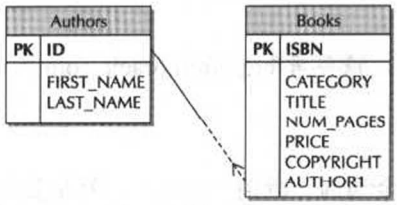


图 1-2 表 AUTHORS 和 BOOKS 的实体关系图

有了 AUTHORS 表以后，就不需要再为某个作者所编写的每一本书，一次又一次地重复这个作者的信息了。在 BOOKS 表中添加一个 AUTHOR1 列，对 BOOKS 表中的每一本书，都在这个 AUTHOR1 列中插入 AUTHORS 表中适当的 ID 值。在 BOOKS.AUTHOR1 列上使用一个 FOREIGN KEY(外键)，就可以使用 SQL 将这两个表关联在一起。下面来分析一下这个示例：

注意：

在本书 Web 站点上本章的目录中，有一个 CreateUser.sql 脚本，您可能会需要这个脚本。这个脚本创建了一个名为 plsql 的用户，并授予了该用户一些必要的权限。

```
-- Available online as part of PlsqlBlock.sql
CREATE TABLE authors (
  id          NUMBER PRIMARY KEY,
  first_name  VARCHAR2(50),
  last_name   VARCHAR2(50)
);
CREATE TABLE   books (
isbn          CHAR(10) PRIMARY KEY,
category      VARCHAR2(20),
title         VARCHAR2(100),
num_pages     NUMBER,
price         NUMBER,
copyright     NUMBER(4),
author1       NUMBER CONSTRAINT books_author1
              REFERENCES authors(id)
);
```

先在这两个表中分别插入几条数据记录，然后，我们就可以执行 SQL 的 SELECT 操作，根据它们之间的关系，将它们连接在一起。

```
SELECT b.title, a.first_name, a.last_name
FROM authors a, books b
WHERE b.author1 = a.id;
```

这条 SELECT 语句就将这两个表连接在一起，并且检索数据的结果就像您将数据存储在一个平面文件中一样—— 您应该已经非常清楚在平面文件中存储数据的形式了。二者的区别在于，使用多表存储可以降低数据冗余，并减少出错概率，提高存储的灵活性。如果要增加出版社的信息，只需要增加一个名为 PUBLISHER 的表，这个表中要包含一个 ID 列；然后再在 BOOKS 表中添加一个列，最后在这个列上添加一个指向 PUBLISHER.ID 列的一个 FOREIGN KEY。

注意：

关于 SQL 更多的参考信息，请参考 http://otn.oracle.com 上的在线文档。

1.2.3 PL/SQL 与 SQL

SQL 能够实现对数据的完全访问。所谓“完全”，就是我们最终总是可以获取我们所要的东西——虽然很多时候都不是通过最理想的方式获取的。SQL 不能保证访问数据的效率，大多数语言所具有的那种程序设计功能在 SQL 中也很少见。SQL 不具有的功能有：

- 在多条记录上进行循环，一次性地完成对它们的操作
- 通过加密手段，保护代码的安全，并在服务器上而不是在客户机上长久存储代码
- 处理异常
- 与变量、参数、集合、记录、数组、对象、游标、异常和 BFILE 等一起使用

虽然 SQL 的功能非常强大，并且 SQL*Plus(Oracle 专用的 SQL 界面)也包含了 ANSI 标准中没有提供的某些命令和内置函数，但 SQL 更多还是用作访问数据库的方法，而不是一种程序设计语言。PL/SQL 通过增加这里(及本书其他地方)提到的那些特性，弥补了 SQL 的某些不足。

注意：

如果您还不太了解这里提到的程序设计特性到底是什么，也不要担心。这正是本书要解决的问题。后续章节中将会详细地解释这些特性。

理论上，在 PL/SQL 中可以使用 SQL 的所有功能。事实上，从 Oracle 9iR1 开始，PL/SQL 的解析器就与 SQL 的解析器完全相同，保证了命令无论是在哪里执行，都能得到相同的处理。在 Oracle 9iR1 以前，还会发生 SQL 语句得到完全不同处理的情况，现在再也不会发生这种事情了。

我们重新使用前面那条对 BOOKS 和 AUTHORS 表进行查询的语句，这次在一个 PL/SQL 示例中使用这个查询。

```
-- Available online as part of PlsqlBlock.sql
SET SERVEROUTPUT ON
DECLARE
   v_title books.title%TYPE;
   v_first_name authors.first_name%TYPE;
   v_last_name authors.last_name%TYPE;

   CURSOR book_cur IS
      SELECT b.title, a.first_name, a.last_name
      FROM authors a, books b
      WHERE a.id = b.author1;
BEGIN
      DBMS_OUTPUT.ENABLE(1000000);
      OPEN book_cur;
      LOOP
         FETCH book_cur INTO v_title, v_first_name, v_last_name;
         EXIT WHEN book_cur%NOTFOUND;

         IF v_last_name = 'Hardman'
         THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Ron Hardman co-authored '||v_title);
         ELSE
            DBMS_OUTPUT.PUT_LINE('Ron Hardman did not write '||v_title);
         END IF;
      END LOOP;

CLOSE book_cur;

EXCEPTION
   WHEN OTHERS
      THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

在这个示例中，我们使用了前面的查询语句(select)，只是这次它是在整个查询结果集上进行循环，并判断作者的姓氏是否为“Hardman”，如果是，就进行格式化输出。这样就将 SQL 第 4 代语言(4GL)的强大功能与过程化第 3 代语言(3GL)的程序设计特性紧密地结合在一起了。

注意：

注意最后一个代码块的结构。每段代码有始也有终，如 Loop...End Loop, If...End If。

1.2.4 PL/SQL 与 Java

Oracle 8i 在数据库中引入了对 Java 和 Java 存储过程的支持。那么我们为什么不只使用 Java 呢？

PL/SQL 一直都与 Oracle 数据库紧密地集成在一起。Oracle 也连续不断地通过增加与 PL/SQL 的集成特性(例如 PL/SQL 代码的本地编译)，来提高 PL/SQL 的执行性能。这意味着，对代码进行编译时，代码就会转化为 C 语言(Oracle 就是使用 C 语言编写的)。代码运行时，就不再需要进行 PL/SQL 语法和 C 语言之间的解析了。这样 PL/SQL 代码的执行性能就得到了很大的提高——与解析模式(默认模式)相比，执行效率提高 30%。

PL/SQL 的另一个优势是简洁性。可以将一条 SQL 语句直接转化为一个 PL/SQL 块(块的概念将在第 3 章中进行介绍)，方法就是在其前面添加一个 BEGIN，在其后面添加一个 END。而对 Java 来说，就不能这样做了。下面这个示例代码就是使用 PL/SQL 创建的最基本的代码块：

```
BEGIN
   NULL;
END;
/
```

试一试——果然能够运行。它绝对不会做任何事情，但是确实运行了。

下面是 PL/SQL 与众不同的一些其他特性：

- PL/SQL 与 SQL 共用同一个解析器，因此也就保证了它们接口之间的一致性；
- PL/SQL 可以脱离 SQL 执行；
- 很多面向对象的特性正不断地被添加到 PL/SQL，这也就让我们没有理由丢弃 PL/SQL，而去使用 Java。

这也并不是说您就应该总是使用 PL/SQL，而永远不要使用 Java。Java 中包括了 PL/SQL