TURING 图灵程序设计丛书 数据库系列

Apress®

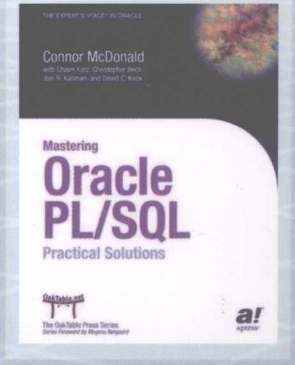
Mastering Oracle PL/SQL Practical Solutions

精通Oracle PL/SQL

[澳] Connor McDonald
[加] Chaim Katz
[美] Christopher Beck 著
[美] Joel R. Kallman
[美] David C. Knox

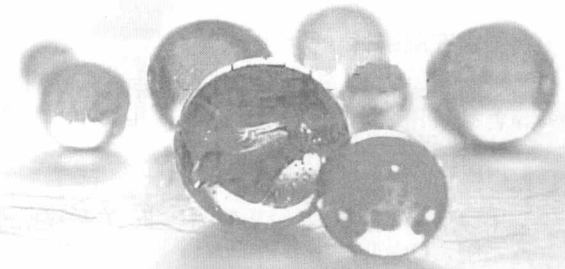
蔡伟毅 译

- Amazon五星图书，五位世界级技术专家联袂巨献
- 提供大量的实战解决方案
- 教你编写健壮、高效且易于维护的PL/SQL代码



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 数据库系列



Mastering Oracle PL/SQL Practical Solutions

精通Oracle PL/SQL

[澳] Connor McDonald
[加] Chaim Katz
[美] Christopher Beck 著
[美] Joel R. Kallman
[美] David C. Knox

蔡伟毅 译

人民邮电出版社
北 京

人民邮电出版社
样书
专用章

图书在版编目（CIP）数据

精通 Oracle PL/SQL /（澳）麦克唐纳（McDonald, C.）等著；蔡伟毅译 .—北京：人民邮电出版社，2009.9
（图灵程序设计丛书）
ISBN 978-7-115-20838-5

Ⅰ. 精… Ⅱ. ①麦…②蔡… Ⅲ. 关系数据库－数据库管理系统，Oracle Ⅳ. TP311.138

中国版本图书馆CIP数据核字（2009）第068948号

内 容 提 要

本书旨在教授读者写出健壮、高效且易于维护的 PL/SQL 代码。全书涵盖了 PL/SQL 提供的大量功能，包括高效数据处理、安全、触发器、DBA 包以及高效的调试技术等。此外，书中含有丰富的示例，并提供了大量提示和技巧。

本书结构清晰，示例丰富，实践性强，适用于 DBA 和数据库开发人员。

图灵程序设计丛书

精通Oracle PL/SQL

◆ 著 [澳] Connor McDonald [加] Chaim Katz
[美] Christopher Beck [美] Joel R. Kallman
[美] David C. Knox

译 蔡伟毅

责任编辑 刘艳娟

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

北京顺义振华印刷厂印刷

◆ 开本：800×1000 1/16

印张：29

字数：685千字 2009年 9 月第 1 版

印数：1－3 000册 2009年 9 月北京第 1 次印刷

著作权合同登记号 图字：01-2008-5179号

ISBN 978-7-115-20838-5/TP

定价：69.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223

反盗版热线：(010)67171154

版权声明

Original English language edition, entitled *Mastering Oracle PL/SQL: Practical Solutions* by Connor McDonald, Chaim Katz, Christoper Beck, Joel R. Kallman, and David C. Knox, published by Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705.

Copyright © 2004 by Connor McDonald, with Chaim Katz, Christoper Beck, Joel R. Kallman, and David C. Knox. Simplified Chinese-language edition copyright © 2009 Posts & Telecom Press. All rights reserved.

本书中文简体字版由Apress L.P.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制本书内容。

版权所有，侵权必究。

致　谢

首先，我要感谢这些年来OakTable给予我的巨大帮助。我从未遇到过这么慷慨地分享自己的时间和知识（而且是大量的知识）的一群人。这些成绩是与他们的帮助分不开的。特别要感谢Tom Kyte、Jonathan Lewis和Dave Ensor，他们鼓励我更深地探索Oracle，还有Mogens Nørgaard，他是OakTable的奠基人之一，感谢他的好客和他的威士忌！也感谢我的编辑Tony Davis，我既感激他，又想要把他打得不省人事，他太执着于做好他的本职工作了。

我还要感谢我所在的公司一直使用着Oracle。要发现PL/SQL等Oracle技术的威力，最好的方法是在日常工作中不断挑战难题，更加精到地使用Oracle基础设施。

最重要的是，我想要感谢我的妻子Gillian支持并宽容我花时间探索Oracle——这些时间我本该用来陪她的！探索技术经常意味着把自己锁在黑暗的房间里若干小时甚至好几天对着电脑显示屏，但只要想到在几步之遥的隔壁房间里，你拥有世界上最漂亮和最完美的女士为妻，就能很容易找到灵感。

——Connor McDonald

我想要感谢Apress出版社的Tony Davis与我并肩作战，即使我总是交稿很晚，他也始终不离不弃。再次说声对不起，Tony。我还要感谢我的妻子Marta，感谢她对我不变的鼓励和支持。我要深情地对她说：“我爱你。”

——Christopher Beck

首先，我想要感谢自己的妻子Sandy支持我编写这本书。如果没有她允许我在“假期”参与这个项目，那么我肯定无法完成这项工作。我也要感谢我在Oracle的同事们，尤其是Tom Kyte和Patrick Sack，感谢他们以高超的技术水准提出的极有价值的见解。我还要感谢我的编辑Tony Davis、本书合作者和技术审稿人，是他们促成了这本书的成功。

——David C. Knox

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

最近，我在一家网上书店搜索关于PL/SQL的图书，结果返回38条记录，还不包括这本书。38本书！据我所知，它们中没有一本书可以作为风靡全球的畅销书摆放于哈利·波特那些书的旁边，那么究竟是什么鼓舞着我们这群作者走到一起写出关于这个主题的第39本书呢？

原因是，无论可用的图书如何过剩，我们仍然在Oracle应用中遇到了许多低劣或陈旧的PL/SQL代码。我个人曾经使用过世界各地的许多Oracle系统，虽然应用程序、架构和方法迥异，但我发现几乎所有这些系统中都有两个共同点。它们要么没有用到Oracle特有的功能，要么是以无计划和不太理想的方式在使用。这种情况在PL/SQL上最为明显，在我遇到的很多系统中，PL/SQL被真正利用的很少，大部分都是误用。

至少部分问题在于大多数的PL/SQL书只关注语法。它们会展示如何编写PL/SQL代码以通过编译并在系统上运行（有些书进一步提供了良好的命名标准和编程结构的指导思想）。但是，就像其他编程语言一样，会用一门语言和用好这门语言有着天壤之别。要构建成功的应用程序，关键在于要巧妙地运用语法知识来编写出健壮、高效且易于维护的程序。这就是写作本书并起这么个书名的动机。我们不是要把你变成PL/SQL程序员，而是要把你变成一位精明的PL/SQL程序员。

本书内容

本书提供了大量的提示、技巧和完整策略，供你在公司中最充分地利用PL/SQL的优点。学完这本书之后，你将像我们一样确信，PL/SQL不只是一个有用的工具，它更是你所要开发的任何Oracle应用程序的有机组成部分。

我们将示范适用于Oracle所有版本（从8*i*到10*g*）的技术细节。本书中大多数例子是用Oracle9*i* R2测试的，你需要做的就是在SQL*Plus中运行它们。

接下来我们逐章介绍要涉及的主题。

- **安装**：这部分展示如何搭建高效的SQL*Plus环境以及如何启动并运行书中用到的性能工具，即AUTOTRACE、SQL_TRACE、TKPROF和RUNSTATS。
- **第1章：高效能的PL/SQL**。这一章给出了我们认为的“高效PL/SQL”的定义，并引入了贯穿全书的主题可论证性（demonstrability），即最终需要证明你的代码在所有合理的条件下都满足性能指标。这一章说明了为什么PL/SQL几乎总是数据库编程的正确工具，但也探讨了PL/SQL并不适合于哪些场合，这时需要创新性地使用SQL来完全避免过程式代码。

- **第2章：全部打包**。包不只是过程的逻辑组合，它们具有很多优势，既有重载和封装，又可有效防止依赖和重编译问题。这一章清晰地展示了这些优势，也讨论了Oracle提供的一些包的有趣应用。
- **第3章：令人困惑的游标**。关于隐式游标和显式游标孰优孰劣一直存在着争议。这一章讨论了为什么显式游标并不像你想象中用得那么多，并介绍了在分布式应用程序中高效使用游标变量和游标表达式的一些情况。
- **第4章：高效数据处理**。这一章展示了如何把数据库中的数据结构和PL/SQL程序中的数据结构最大限度地整合在一起，使代码健壮且易于修改。同时还讨论了如何更好地利用集合把数据从程序批量复制给数据库，抑或反之。
- **第5章：PL/SQL优化技巧**。这一章提供了一些在PL/SQL开发中经常遇到的问题的现成解决方案。展示了如何避免一些隐藏的开销，并强调了一些容易使人犯错的“陷阱”(gotcha)。
- **第6章：触发器**。这一章讲解了基本的触发器原理和高效使用各类触发器的一些方法，还研究了Oracle Streams（流）这一较新主题，并展示了如何使用它们来实现集中式数据审计跟踪。
- **第7章：DBA包**。这一章介绍了“DBA工具包”——一组可以用来自动重现管理活动的包，例如用于性能诊断和解决故障、备份和恢复以及监控数据库故障。
- **第8章：安全包**。这一章介绍PL/SQL包和触发器的使用，以在数据库中实现高效的安全机制。它讲解了一些基本问题，如调用者和定义者权限模型的使用、包的构建和模式设计，继而讨论了审计数据库活动和保护源代码等问题的具体解决方案。
- **第9章：Web包**。这一章研究了一系列内建的数据库包，总称为PL/SQL Web工具包，该工具包可以让开发者以动态网页的方式直接展现数据库。这一章也讲解了cookie的使用、表和文件的管理以及如何从PL/SQL存储过程中直接调用Web服务等内容。
- **第10章：PL/SQL调试**。很少有人第一次就能正确使用它，所以这一章对高效调试PL/SQL代码的技术做了大量陈述，从DBMS_OUTPUT的简单使用到DBMS_APPLICATION_INFO和UTL_FILE复杂包的使用。本章最后开发了一个巧妙的自定义调试工具DEBUG。
- **附录A：构建DEBUG**。这个附录列出了第10章中用到的DEBUG工具的全部代码。

读者对象

本书主要面向DBA或致力于在Oracle数据库中实现高效数据处理、安全和数据库管理机制的开发者。对于在Oracle数据库上开发应用程序的人员和想学习如何高效使用PL/SQL的读者来说，本书也非常适用。

如果你刚接触PL/SQL，则在学习本书前需要花些时间来熟悉这门语言。它并不是为新手而写的。而一旦你开始学习它了，你会发现本书是一本非常优秀的指导手册，可以确保你所构建的PL/SQL解决方案是健壮、高效且易于维护的。

——Connor McDonald

安　装

这部分描述了如何搭建起可以运行书中示例的环境。我们将涉及以下这些内容：

- 如何安装 SCOTT/TIGER示例模式；
- 如何配置SQL*Plus环境；
- 如何配置AUTOTRACE，AUTOTRACE是一个SQL*Plus工具，它可以展示Oracle已经或即将如何执行一个查询，并提供该查询处理过程的统计数据；
- 如何设定和使用两个参数SQL_TRACE、TIMED_STATISTICS和一个命令行工具TKPROF，它们将揭示应用程序执行了哪些SQL语句以及SQL是怎样执行的；
- 如何设置和使用RUNSTATS实用工具。

这里我们只提供各种性能工具的基本设置指令，可以让你很快地配置环境来运行书中的例子。至于完整指令以及怎样解读这些工具所提供的数据，建议你参考Oracle文档集或相关的书，比如Thomas Kyte的*Expert One-on-one Oracle*[①]（Apress，ISBN:1-59059-243-3）。

安装 SCOTT/TIGER 模式

书中很多例子都是在SCOTT模式下描述EMP/DEPT（员工/部门）表。建议你用其他账户而不是SCOTT来安装这些表，以避免其他用户使用和修改相同数据造成的负面影响。要在你自己的模式中创建SCOTT示例表，只需执行如下步骤：

(1) 从命令行运行cd [ORACLE_HOME]/sqlplus/demo；

(2) 以要求的用户身份登录SQL*Plus；

(3) 运行@DEMOBLD.SQL。

DEMOBLD.SQL脚本将会为你创建和填充5张表。完成时，它会自动退出SQL*Plus，所以发现运行完脚本后SQL*Plus就突然消失，不必大惊小怪。如果想要删除这个模式，可以随时运行[ORACLE_HOME]/sqlplus/demo/demodrop.sql。

SQL*Plus 的环境

书中的例子是设计为在SQL*Plus环境中运行的。SQL*Plus提供了很多有用的选项和命令，

① 本书新版的中文版《Oracle 9*i* & 10*g*编程艺术》已经由人民邮电出版社出版。——编者注

在整本书中都会频繁用到。例如，本书中很多例子都以某种方式用到了DBMS_OUTPUT。为了启用DBMS_OUTPUT，必须执行下面的SQL*Plus命令：

```
SQL> set serveroutput on
```

另外一种方法是，在SQL*Plus里设置一个LOGIN.SQL文件，这个文件是每次启动SQL*Plus会话都会运行的脚本。在文件中设置参数SERVEROUTPUT为自动。比如这么设置LOGIN.SQL脚本（可以根据自己的环境进行改动）：

```
define _editor=vi

set serveroutput on size 1000000

set trimspool on
set long 5000
set linesize 100
set pagesize 9999
column plan_plus_exp format a80
```

而且我们可以使用这个脚本来格式化SQL*Plus的提示符，这样我们总是可以知道以什么用户名登录到哪个数据库。例如，当你在学习本书时，你会遇到下面格式的提示符：

```
scott@oracle9i_test>
```

这说明你以SCOTT用户名登录了ORACLE9I_TEST数据库。下面就是LOGIN.SQL脚本实现该功能的代码：

```
column global_name new_value gname
set termout off
select lower(user) || '@' ||
global_name from global_name;
set sqlprompt '&gname> '
set termout on
```

这个登录脚本只会在启动时运行一次。所以，如果开始时以SCOTT用户名登录，然后切换到另一个账户，则新账户就不会体现在你的提示符上：

```
SQL*Plus: Release 8.1.7.0.0 - Production on Sun Mar 16 15:02:21 2003

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Enter user-name: scott/tiger

Connected to:
Personal Oracle8i Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

scott@ORATEST> connect tony/davis
```

```
Connected.
scott@ORATEST>
```

下面的CONNECT.SQL脚本能够解决这个问题：

```
set termout off
connect &1
@login
set termout on
```

每次当你想更改账户时，运行这段脚本即可（这段脚本首先连接数据库，然后运行登录脚本）：

```
scott@ORATEST> @connect tony/davis
tony@ORATEST>
```

为了使SQL*Plus能在启动时自动运行登录脚本，需要将它保存在一个目录下（将CONNECT.SQL也放进这个相同的目录），然后把SQLPATH环境变量也指向该目录。如果你在用Windows操作系统，则点击Start按钮，选择Run，并输入regedit。转到HKEY_LOCAL_MACHINE/SOFTWARE/ORACLE并找到SQLPATH文件（我的是在HOME0）。双击它并设置路径为存储该脚本的目录（例如C:\oracle\ora81\sqlplus\admin）。

在 SQL*Plus 中设置 AUTOTRACE

学习本书时，始终监控所执行的查询的性能将会很有帮助。SQL*Plus提供了AUTOTRACE工具，可用来查看执行查询的运行计划和占用的资源。报告将会在SQL DML（数据操纵语言）成功运行后自动生成。本书大量使用了AUTOTRACE工具。配置AUTOTRACE工具有多种方法，而以下是我推荐的办法：

(1) 切换到目录$ORACLE_HOME/rdbms/admin。

(2) 以任何有CREATE TABLE（创建表）和CREATE PUBLIC SYNONYM（创建公有同义词）权限的用户登录到SQL*Plus。

(3) 运行@UTLXPLAN来创建一张由AUTOTRACE使用的PLAN_TABLE表。

(4) 运行CREATE PUBLIC SYNONYM PLAN_TABLE FOR PLAN_TABLE，这样每个用户不用指定模式就能访问这张表。

(5) 运行GRANT ALL ON PLAN_TABLE TO PUBLIC，这样每个用户都能访问这张表。

(6) 退出SQL*Plus并切换目录如下：

```
cd $ORACLE_HOME/sqlplus/admin.
```

(7) 以SYSDBA登录到SQL*Plus。

(8) 运行@PLUSTRCE。

(9) 运行GRANT PLUSTRACE TO PUBLIC。

可以通过启用AUTOTRACE和执行一段简单的查询来测试你的设置：

```
SQL> set AUTOTRACE traceonly
SQL> select * from emp, dept
  2   where emp.deptno=dept.deptno;
```

```
14 rows selected.

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   MERGE JOIN
   2    1     SORT (JOIN)
   3    2       TABLE ACCESS (FULL) OF 'DEPT'
   4    1     SORT (JOIN)
   5    4       TABLE ACCESS (FULL) OF 'EMP'

Statistics
----------------------------------------------------------
          0  recursive calls
          8  db block gets
          2  consistent gets
          0  physical reads
          0  redo size
       2144  bytes sent via SQL*Net to client
        425  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
         14  rows processed

SQL> set AUTOTRACE off
```

关于AUTOTRACE的详细用法和对其提供数据的解释，请查看Oracle文档集中*Oracle 9i Database Performance Tuning Guide and Reference*的第11章或*SQL * Plus User's Guide and Reference*的第9章。

性能工具

除了AUTOTRACE，书中还使用了其他各种各样的性能工具。我们将在这部分简要介绍其安装。

TIMED_STATISTICS

TIMED_STATISTICS参数可以指定Oracle是否测量各种内部操作的执行时间。如果没有设置这个参数，跟踪文件输出的价值就会锐减。与其他参数一样，你可以在实例级（在INIT.ORA中）或会话级设置TIMED_STATISTICS。前者不会影响性能，所以推荐使用。把下面这行添加到INIT.ORA文件中，那么下次重新启动数据库后，跟踪就会启用：

```
timed_statistics=true
```

若在会话级上设置，则应这样写：

此为试读，需要完整PDF请访问：www.ertongbook.com

```
SQL> alter session set timed_statistics=true;
```

SQL_TRACE 和 TKPROF

SQL_TRACE工具和TKPROF命令行工具可用来对数据库内部发生的活动进行跟踪。简言之，SQL_TRACE用于将每条SQL语句的性能信息写到数据库服务器文件系统的跟踪文件中。通常状况下，这些跟踪文件很难直接理解。可以使用TKPROF工具，从给定的跟踪文件的输入产生基于文本的报表文件。

SQL_TRACE

SQL_TRACE工具用来跟踪特定数据库会话或实例的活动，并将记录写进数据库服务器的操作系统跟踪文件中。跟踪文件中的每行都会记录Oracle服务器进程在处理SQL语句时的特定操作。SQL_TRACE最早是做调试用的，而且它现在也很适合这个用途，但是它也可以用来分析数据库的SQL活动，以达到优化语句的目的。

- 设置SQL_TRACE

SQL_TRACE可以在单个会话或整个数据库实例中被启用。然而，它很少用于数据库级，因为这样会导致严重的性能问题。记住SQL_TRACE会伴随着I/O操作，将处理的每个SQL语句记录到日志文件中。

要启用当前会话的跟踪功能，应该使用ALTER SESSION，如下所示：

```
SQL> alter session set sql_trace=true;
```

应在指定时间间隔内跟踪一个会话，避免跟踪的有效时间太长。要禁用当前的跟踪功能，执行如下语句：

```
SQL> alter session set sql_trace=false;
```

- 控制跟踪文件

由SQL_TRACE产生的跟踪文件最后会变得很大。INIT.ORA文件中设置数据库实例或会话的一些全局初始化参数影响着跟踪文件。如果启用这些参数，SQL_TRACE将会把内容写入USER_DUMP_DEST初始化参数指定的操作系统目录下的某个文件中。注意，USER进程（专用服务器）的跟踪文件保存在USER_DUMP_DEST目录下，Oracle后台进程（如使用MTS的共享服务器、使用任务队列的任务队列进程）产生的跟踪文件保存于BACKGROUND_DUMP_DEST目录下。在共享服务器配置下使用SQL_TRACE是不推荐的，因为会话会从一个共享服务器跳到另一个共享服务器，产生的跟踪信息散落在多个文件里，实际上无法使用。

跟踪文件一般命名为：

```
ora<spid>.trc,
```

<spid>是跟踪功能打开的会话的服务器进程ID号。在Windows中，下面的查询用来重新获取会话的跟踪文件名：

```
SQL> select c.value || '\ORA' || to_char(a.spid,'fm00000') || '.trc'
  2    from v$process a, v$session b, v$parameter c
```

```
  3   where a.addr = b.paddr
  4     and b.audsid = userenv('sessionid')
  5     and c.name = 'user_dump_dest';
```

在Unix中，下面的查询用来重新获取会话的跟踪文件名：

```
SQL> select c.value || '/' || d.instance_name || '_ora_' ||
  2               to_char(a.spid,'fm99999') || '.trc'
  3    from v$process a, v$session b, v$parameter c, v$instance d
  4   where a.addr = b.paddr
  5     and b.audsid = userenv('sessionid')
  6     and c.name = 'user_dump_dest';
```

跟踪文件的大小受数据库实例的INIT.ORA文件中初始化参数MAX_DUMP_FILE_SIZE的值的约束。也可以在会话级上使用ALTER SESSION命令来改变这个参数，例如：

```
SQL> alter session set max_dump_file_size = unlimited;
Session altered.
```

TKPROF

TKPROF实用工具以SQL_TRACE文件作为输入，并产生一个基于文本的报表文件作为输出。它是一个很简单的工具，在一个给定的跟踪文件中把大量的细节信息分类归纳，使其易于理解，便于进行性能优化。

- 使用TKPROF

TKPROF是一个简单的命令行工具，用来把一组原始跟踪文件翻译成一个更容易理解的报表。TKPROF最简单的形式可以这样使用：

```
tkprof <trace-file-name> <report-file-name>
```

为了阐明如何联合使用TKPROF和SQL_TRACE，我们举一个简单的例子。具体的做法是，我们会跟踪以前的AUTOTRACE例子中的查询，并由得到的跟踪文件产生一个报表。首先，以自己想要的用户名登录SQL*Plus，并执行下面的代码：

```
SQL> select c.value || '\ORA' || to_char(a.spid,'fm00000') || '.trc'
  2      from v$process a, v$session b, v$parameter c
  3     where a.addr = b.paddr
  4       and b.audsid = userenv('sessionid')
  5       and c.name = 'user_dump_dest';

C.VALUE||'\ORA'||TO_CHAR(A.SPID,'FM00000')||'.TRC'
------------------------------------------------------------
C:\oracle\admin\oratest\udump\ORA01528.trc

SQL> alter session set timed_statistics=true;

Session altered.
SQL> alter session set sql_trace=true;

Session altered.
```

```
SQL> select * from emp, dept
  2    where emp.deptno=dept.deptno;

SQL> alter session set sql_trace=false;

SQL> exit
```

现在，我们只需用TKPROF命令行来规范跟踪文件的格式，如下所示：

```
C:\oracle\admin\oratest\udump>tkprof ORA01528.TRC tkprof_rep1.txt
```

现在可以打开TKPROF_REP1.TXT文件看报表了。在这里不打算讨论输出的细节，简要地说，在报表的上端能看到SQL语句。接着，会看到这段语句的执行报告。这个报表通过3个不同的Oracle SQL处理阶段——解析、运行和提取来阐明。对于每个处理阶段，我们用如下指标来衡量：

- 该阶段发生的次数；
- 该阶段经历的CPU时间；
- 经历的真正时间；
- 发生于磁盘上的I/O物理操作次数；
- 在“读一致性”（consistent-read）模式中处理块的数量；
- 在“当前”（current）模式中读的块的数量（在语句处理期间，数据被一个外部进程改变的时候发生的读取）；
- 受该语句影响的块的数量。

执行报表如下：

```
call     count       cpu    elapsed       disk      query    current        rows
-------  ------  --------  ---------  ----------  ----------  ----------  ------
Parse        1      0.01       0.02          0           0           0           0
Execute      1      0.00       0.00          0           0           0           0
Fetch        2      0.00       0.00          0           2           8          14
-------  ------  --------  ---------  ----------  ----------  ----------  ------
total        4      0.01       0.02          0           2           8          14
```

从执行报表中可以看到所用的优化器方法以及开启跟踪的会话的用户ID号（可以在ALL_USERS表中匹配这个ID号来获得当前的用户名）：

```
Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 52
```

另外，我们发现了语句没在库缓存中找到的次数。语句第一次执行时，计数为1，但如果在后续调用中使用绑定变量，则计数变为0。此外，注意绑定变量的缺失，大量的库缓存丢失就说明缺失了绑定变量。

最后，报表显示了这个语句的执行计划。该信息类似于AUTOTRACE提供的信息，有一个重要的区别就是，我们可以看到计划中每一步的实际输出行数：

```
Rows     Row Source Operation
-------  ---------------------------------------------------
```

```
14  MERGE JOIN
 5  SORT JOIN
 4  TABLE ACCESS FULL DEPT
14  SORT JOIN
14  TABLE ACCESS FULL EMP
```

关于使用SQL_TRACE和TKPROF的全部细节，以及对跟踪数据的解释，参考*Oracle 9i Database Performance Tuning Guide and Peference*的第10章。

RUNSTATS

RUNSTATS是一个简单的测试工具，它可以比较两组代码的执行，显示它们的消耗时间、会话级统计数据（例如解析调用）和闩定差异。闩定是这个工具提供的最关键的信息。

注意 RUNSTATS工具最先是由Tom Kyte 编写的，他也是http://asktom.oracle.com网站的创办人。RUNSTATS的完整信息和使用例子可以在http://asktom.oracle.com/~tkyte/runstats.html找到。在第4章我们使用集合对这个工具进行了定制。

要运行RUNSTATS测试工具，必须有访问V$STATNAME、V$MYSTAT和V$LATCH的权限。还必须有对SYS.V_$STATNAME、SYS.V_$MYSTAT和SYS.V_$LATCH的直接SELECT权限（不是通过角色授权）。可以创建以下的视图：

```
SQL> create or replace view stats
  2  as select 'STAT...' || a.name name, b.value
  3        from v$statname a, v$mystat b
  4       where a.statistic# = b.statistic#
  5      union all
  6      select 'LATCH.' || name,  gets
  7        from v$latch;

View created.
```

你所需要的只是一张存储这些统计数据的小表：

```
create global temporary table run_stats
( runid varchar2(15),
  name varchar2(80),
  value int )
on commit preserve rows;
```

测试工具包的代码如下：

```
create or replace package runstats_pkg
as
    procedure rs_start;
    procedure rs_middle;
    procedure rs_stop( p_difference_threshold in number default 0 );
end;
/
```

```
create or replace package body runstats_pkg
as

g_start number;
g_run1  number;
g_run2  number;

procedure rs_start
is
begin
    delete from run_stats;

    insert into run_stats
    select 'before', stats.* from stats;

    g_start := dbms_utility.get_time;
end;

procedure rs_middle
is
begin
    g_run1 := (dbms_utility.get_time-g_start);

    insert into run_stats
    select 'after 1', stats.* from stats;
    g_start := dbms_utility.get_time;

end;

procedure rs_stop(p_difference_threshold in number default 0)
is
begin
    g_run2 := (dbms_utility.get_time-g_start);

    dbms_output.put_line
    ( 'Run1 ran in ' || g_run1 || ' hsecs' );
    dbms_output.put_line
    ( 'Run2 ran in ' || g_run2 || ' hsecs' );
    dbms_output.put_line
    ( 'run 1 ran in ' || round(g_run1/g_run2*100,2) ||
      '% of the time' );
    dbms_output.put_line( chr(9) );
insert into run_stats
select 'after 2', stats.* from stats;

dbms_output.put_line
```

```
( rpad( 'Name', 30 ) || lpad( 'Run1', 10 ) ||
  lpad( 'Run2', 10 ) || lpad( 'Diff', 10 ) );

for x in
( select rpad( a.name, 30 ) ||
         to_char( b.value-a.value, '9,999,999' ) ||
         to_char( c.value-b.value, '9,999,999' ) ||
         to_char( ( (c.value-b.value)-(b.value-a.value)), '9,999,999' ) data
    from run_stats a, run_stats b, run_stats c
   where a.name = b.name
     and b.name = c.name
     and a.runid = 'before'
     and b.runid = 'after 1'
     and c.runid = 'after 2'
     and (c.value-a.value) > 0
     and abs( (c.value-b.value) - (b.value-a.value) )
           > p_difference_threshold
   order by abs( (c.value-b.value)-(b.value-a.value))
) loop
    dbms_output.put_line( x.data );
end loop;

dbms_output.put_line( chr(9) );
dbms_output.put_line
( 'Run1 latches total versus runs -- difference and pct' );
dbms_output.put_line
( lpad( 'Run1', 10 ) || lpad( 'Run2', 10 ) ||
  lpad( 'Diff', 10 ) || lpad( 'Pct', 8 ) );

for x in
( select to_char( run1, '9,999,999' ) ||
         to_char( run2, '9,999,999' ) ||
         to_char( diff, '9,999,999' ) ||
         to_char( round( run1/run2*100,2 ), '999.99' ) || '%' data
    from ( select sum(b.value-a.value) run1, sum(c.value-b.value) run2,
                  sum( (c.value-b.value)-(b.value-a.value)) diff
             from run_stats a, run_stats b, run_stats c
            where a.name = b.name
              and b.name = c.name
              and a.runid = 'before'
                  and b.runid = 'after 1'
                  and c.runid = 'after 2'
                  and a.name like 'LATCH%'
                )
    ) loop
        dbms_output.put_line( x.data );
```

此为试读,需要完整PDF请访问: www.ertongbook.com