Expert Oracle Database Architecture

Third Edition

Thomas Kyte

Darl Kuhn

Apress®

Expert Oracle Database Architecture

Copyright © 2014 by Thomas Kyte and Darl Kuhn

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4302-6298-5

ISBN-13 (electronic): 978-1-4302-6299-2

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr

Lead Editor: Jonathan Gennick

Developmental Editor: Douglas Pundick

Technical Reviewers: Melanie Caffrey and Christopher Beck

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Louise Corrigan, Jim DeWolf, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Steve Weiss

Coordinating Editor: Jill Balzano

Copy Editor: Kim Burton-Weisman

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Contents at a Glance

About the Authors.....	xvii
About the Technical Reviewers	xix
Acknowledgments	xxi
Introduction	xxiii
Setting Up Your Environment.....	xxvii
■ Chapter 1: Developing Successful Oracle Applications	1
■ Chapter 2: Architecture Overview	53
■ Chapter 3: Files	73
■ Chapter 4: Memory Structures	127
■ Chapter 5: Oracle Processes	173
■ Chapter 6: Locking and Latching.....	205
■ Chapter 7: Concurrency and Multiversioning	253
■ Chapter 8: Transactions.....	275
■ Chapter 9: Redo and Undo	307
■ Chapter 10: Database Tables	355
■ Chapter 11: Indexes.....	439
■ Chapter 12: Datatypes	511
■ Chapter 13: Partitioning	581
■ Chapter 14: Parallel Execution	659
■ Chapter 15: Data Loading and Unloading	697
Index.....	767

Contents

About the Authors.....	xvii
About the Technical Reviewers	xix
Acknowledgments	xxi
Introduction	xxiii
Setting Up Your Environment.....	xxvii
 ■ Chapter 1: Developing Successful Oracle Applications	 1
My Approach	2
The Black Box Approach.....	3
How (and How Not) to Develop Database Applications	11
Understanding Oracle Architecture.....	12
Understanding Concurrency Control.....	23
Multiversioning	27
Database Independence?	34
How Do I Make It Run Faster?	48
The DBA-Developer Relationship.....	51
Summary	51
 ■ Chapter 2: Architecture Overview	 53
Defining Database and Instance	54
The SGA and Background Processes.....	61
Connecting to Oracle	63
Dedicated Server	63
Shared Server	65
Mechanics of Connecting over TCP/IP	67

Pluggable Databases.....	69
Reduced Resource Utilization.....	70
Reduced Maintenance	70
How Is a Pluggable Database Different?	71
Summary	71
■ Chapter 3: Files	73
Parameter Files	74
What Are Parameters?	75
Legacy init.ora Parameter Files	78
Server Parameter Files (SPFILEs).....	80
Converting to SPFILEs.....	80
Trace Files	88
Requested Trace Files.....	89
Trace Files Generated in Response to Internal Errors.....	94
Trace File Wrap-up.....	98
Alert File	99
Data Files	102
A Brief Review of File System Mechanisms	102
The Storage Hierarchy in an Oracle Database.....	104
Dictionary-Managed and Locally-Managed Tablespaces	107
Temp Files	109
Control Files	111
Redo Log Files.....	111
Online Redo Log.....	112
Archived Redo Log.....	114
Password Files	115
Change Tracking File	118
Flashback Logs	119
Flashback Database	119
Fast Recovery Area.....	120

DMP Files (EXP/IMP Files)	121
Data Pump Files	122
Flat Files.....	124
Summary	125
■ Chapter 4: Memory Structures	127
The Process Global Area and User Global Area	128
Manual PGA Memory Management	129
Automatic PGA Memory Management.....	135
Choosing Between Manual and Auto Memory Management.....	146
PGA and UGA Wrap-up	147
The System Global Area	148
Fixed SGA	153
Redo Buffer.....	153
Block Buffer Cache	155
Shared Pool	162
Large Pool.....	164
Java Pool	165
Streams Pool	165
SGA Memory Management.....	166
Summary	171
■ Chapter 5: Oracle Processes	173
Server Processes	173
Dedicated Server Connections	174
Shared Server Connections	176
Database Resident Connection Pooling (DRCP).....	177
Connections vs. Sessions	177
Dedicated Server vs. Shared Server vs. DRCP	183
Dedicated/Shared Server Wrap-up	186
Background Processes.....	187
Focused Background Processes	188
Utility Background Processes.....	199

Slave Processes	201
I/O Slaves.....	202
Pnnn: Parallel Query Execution Servers	202
Summary.....	203
■ Chapter 6: Locking and Latching.....	205
What Are Locks?.....	205
Locking Issues.....	208
Lost Updates.....	208
Pessimistic Locking.....	209
Optimistic Locking.....	210
Optimistic or Pessimistic Locking?.....	217
Blocking.....	217
Deadlocks.....	220
Lock Escalation.....	224
Lock Types.....	225
DML Locks	225
DDL Locks.....	236
Latches	240
Mutexes.....	249
Manual Locking and User-Defined Locks	250
Summary.....	251
■ Chapter 7: Concurrency and Multiversioning.....	253
What Are Concurrency Controls?	253
Transaction Isolation Levels	254
READ UNCOMMITTED.....	256
READ COMMITTED	257
REPEATABLE READ.....	259
SERIALIZABLE	261
READ ONLY.....	263

Implications of Multiversion Read Consistency.....	264
A Common Data Warehousing Technique That Fails.....	264
An Explanation for Higher Than Expected I/O on Hot Tables.....	265
Write Consistency.....	267
Consistent Reads and Current Reads	268
Seeing a Restart	270
Why Is a Restart Important to Us?	273
Summary	274
■ Chapter 8: Transactions.....	275
Transaction Control Statements	275
Atomicity	277
Statement-Level Atomicity	277
Procedure-Level Atomicity.....	279
Transaction-Level Atomicity	283
DDL and Atomicity	283
Durability	283
WRITE Extensions to COMMIT.....	284
COMMITs in a Nondistributed PL/SQL Block	285
Integrity Constraints and Transactions.....	287
IMMEDIATE Constraints	287
DEFERRABLE Constraints and Cascading Updates.....	287
Bad Transaction Habits.....	291
Committing in a Loop.....	292
Using Autocommit	298
Distributed Transactions.....	299
Autonomous Transactions	300
How Autonomous Transactions Work.....	301
When to Use Autonomous Transactions.....	303
Summary.....	305

■ Chapter 9: Redo and Undo	307
What Is Redo?	307
What Is Undo?	308
How Redo and Undo Work Together	312
Example INSERT-UPDATE-DELETE-COMMIT Scenario	312
Commit and Rollback Processing.....	316
What Does a COMMIT Do?	316
What Does a ROLLBACK Do?	323
Investigating Redo.....	324
Measuring Redo.....	324
Can I Turn Off Redo Log Generation?	326
Why Can't I Allocate a New Log?	330
Block Cleanout.....	331
Log Contention.....	334
Temporary Tables and Redo/Undo	336
Investigating Undo.....	341
What Generates the Most and Least Undo?.....	341
ORA-01555: Snapshot Too Old Error	343
Summary	354
■ Chapter 10: Database Tables	355
Types of Tables	355
Terminology.....	357
Segment	357
Segment Space Management	359
High-water Mark.....	360
FREELISTS	362
PCTFREE and PCTUSED	366
LOGGING and NOLOGGING.....	369
INITRANS and MAXTRANS	370

Heap Organized Tables	370
Index Organized Tables	373
Index Organized Tables Wrap-up	388
Index Clustered Tables	388
Index Clustered Tables Wrap-up	396
Hash Clustered Tables	396
Hash Clustered Tables Wrap-up	404
Sorted Hash Clustered Tables	405
Nested Tables	408
Nested Tables Syntax	408
Nested Table Storage	415
Nested Tables Wrap-up	418
Temporary Tables	419
Statistics Prior to 12c	422
Statistics Starting with 12c	426
Temporary Tables Wrap-up	430
Object Tables	431
Object Tables Wrap-up	437
Summary	438
■ Chapter 11: Indexes	439
An Overview of Oracle Indexes	439
B*Tree Indexes	441
Index Key Compression	444
Reverse Key Indexes	447
Descending Indexes	452
When Should You Use a B*Tree Index?	455
B*Trees Wrap-up	465

Bitmap Indexes.....	465
When Should You Use a Bitmap Index?	466
Bitmap Join Indexes	470
Bitmap Indexes Wrap-up	473
Function-Based Indexes.....	473
A Simple Function-Based Index Example	474
Indexing Only Some of the Rows	483
Implementing Selective Uniqueness	484
Caveat Regarding ORA-01743	485
Function-Based Indexes Wrap-up	486
Application Domain Indexes	486
Invisible Indexes.....	487
Multiple Indexes on the Same Column Combinations	489
Indexing Extended Columns	489
Virtual Column Solution	490
Function-Based Index Solution.....	492
Frequently Asked Questions and Myths About Indexes.....	494
Do Indexes Work on Views?	494
Do Nulls and Indexes Work Together?	494
Should Foreign Keys Be Indexed?	497
Why Isn't My Index Getting Used?	498
Myth: Space Is Never Reused in an Index	504
Myth: Most Discriminating Elements Should Be First	507
Summary	510
■ Chapter 12: Datatypes	511
An Overview of Oracle Datatypes	511
Character and Binary String Types	514
NLS Overview	514
Character Strings.....	518

Binary Strings: RAW Types	525
Extended Datatypes	527
Number Types.....	529
NUMBER Type Syntax and Usage.....	531
BINARY_FLOAT/BINARY_DOUBLE Type Syntax and Usage.....	535
Non-Native Number Types.....	536
Performance Considerations	536
Long Types	538
Restrictions on LONG and LONG RAW Types.....	538
Coping with Legacy LONG Types.....	539
Dates, Timestamps, and Interval Types	545
Formats	545
DATE Type	546
TIMESTAMP Type	552
INTERVAL Type	559
LOB Types.....	562
Internal LOBs	563
BFILEs.....	577
ROWID/UROWID Types	579
Summary	579
■ Chapter 13: Partitioning	581
Partitioning Overview	582
Increased Availability.....	582
Reduced Administrative Burden	584
Enhanced Statement Performance.....	588
Reduced Contention in an OLTP System.....	590
Table Partitioning Schemes.....	590
Range Partitioning	591
Hash Partitioning	594
List Partitioning	599

Interval Partitioning	601
Reference Partitioning	606
Interval Reference Partitioning	611
Virtual Column Partitioning	613
Composite Partitioning	614
Row Movement	617
Table Partitioning Schemes Wrap-up	619
Partitioning Indexes	620
Local Indexes vs. Global Indexes	621
Local Indexes	621
Global Indexes	627
Partial Indexes	642
Partitioning and Performance, Revisited	644
Ease of Maintenance Features	650
Multiple Partition Maintenance Operations	650
Cascade Truncate	653
Cascade Exchange	654
Auditing and Segment Space Compression	656
Summary	657
■ Chapter 14: Parallel Execution	659
When to Use Parallel Execution	660
A Parallel Processing Analogy	661
Oracle Exadata	662
Parallel Query	663
Parallel DML	668
Parallel DDL	672
Parallel DDL and Data Loading Using External Tables	673
Parallel DDL and Extent Trimming	675

Procedural Parallelism	684
Parallel Pipelined Functions	684
Do-It-Yourself Parallelism	687
Old School Do-It-Yourself Parallelism	691
Summary	695
■ Chapter 15: Data Loading and Unloading	697
External Tables	697
Setting Up External Tables	698
Dealing with Errors	707
Using an External Table to Load Different Files	711
Multiuser Issues	711
Preprocessing	712
External Tables Summary	722
Data Pump Unload	723
SQLLDR	725
Loading Data with SQLLDR FAQs	729
SQLLDR Caveats	755
SQLLDR Summary	756
Flat File Unload	756
Summary	765
Index	767

About the Authors

I am **Tom Kyte**. I have been working for Oracle since version 7.0.9 (that's 1993 for people who don't mark time by Oracle versions). However, I've been working with Oracle since about version 5.1.5c (the \$99 single-user version for DOS on 360KB floppy disks). Before coming to work at Oracle, I worked for more than six years as a systems integrator, building large-scale, heterogeneous databases and applications, mostly for military and government customers. These days, I spend a great deal of my time working with the Oracle database and, more specifically, helping people who are using the Oracle database. I work directly with customers, either in specifying and building their systems or, more frequently, in helping them rebuild or tune them ("tuning" frequently being a synonym for rebuilding). In addition, I am the Tom behind the "Ask Tom" column in *Oracle Magazine*,

where I answer people's questions about the Oracle database and tools. On a typical day, I receive and answer dozens of questions at <http://asktom.oracle.com>. Every two months, I publish a "best of" in the magazine (all of the questions asked are available on the Web, stored in an Oracle database, of course). Additionally, I give technical seminars covering much of the material you'll find in this book. Basically, I spend a lot of my time helping people be successful with the Oracle database. Oh yes, in my spare time, I build applications and develop software within Oracle Corporation itself.

This book is a reflection of what I do every day. The material within covers topics and questions that I see people struggling with every day. These issues are covered from a perspective of "When I use this, I do it this way." It is the culmination of many years of experience using the product in myriad situations.

Darl Kuhn I'm a DBA/developer working for Oracle. I also teach Oracle classes at Regis University in Denver, Colorado and am also an active member of the Rocky Mountain Oracle Users Group. I enjoy sharing knowledge and that has led to several book projects over the years.

About the Technical Reviewers

Melanie Caffrey is a senior development manager for Oracle Corporation, providing front-end and back-end Oracle solutions for the business needs of various clients. She is co-author of several technical publications including *Expert Oracle Practices: Oracle Database Administration from the Oak Table* published by Apress, as well as *Oracle Web Application Programming for PL/SQL Developers*, *The Oracle DBA Interactive Workbook*, and *Oracle Database Administration: The Complete Video Course*, all published by Prentice Hall. She instructed students in Columbia University's Computer Technology and Applications program in New York City, teaching advanced Oracle database administration and PL/SQL development. She is also a frequent Oracle conference speaker.

Christopher Beck has a degree in computer science from Rutgers University and has been working with multiple DBMSs for more than 19 years. He has spent the last 15 years as an Oracle employee, where he is currently a principal technologist focusing on core database technologies. He is a co-inventor of two US patents on software methodologies, which were the basis for what is now known as Oracle Application Express. Chris has reviewed other Oracle books, including Tom's first book, *Expert One-On-One* (Apress, 2003), and is himself the co-author of two books, *Beginning Oracle Programming* (Apress, 2003) and *Mastering Oracle PL/SQL* (Apress, 2004). He resides in Northern Virginia with his wife, Marta, and four children, and when not spending time with them, can usually be found wasting time playing video games or watching Series A football.

Acknowledgments

I would like to thank many people for helping me complete this book.

First, I would like to thank you, the reader of this book. There is a high probability that if you are reading this book, you have participated in my site <http://asktom.oracle.com> in some fashion, perhaps by asking a question or two. It is that act—the act of asking questions, and of questioning the answers—that provides me with the material for the book and the knowledge behind the material. Without the questions, I would not be as knowledgeable about the Oracle database as I am. So, it is you who ultimately makes this book possible.

I would like to thank Tony Davis for his previous work making my work read well. If you enjoy the flow of the sections, the number of section breaks, and the clarity, then that is in some part due to him. I have worked with Tony writing technical material since the year 2000 and have watched his knowledge of Oracle grow over that time. He now has the ability to not only edit the material, but in many cases tech edit it as well. Many of the examples in this book are there because of him (pointing out that the casual reader was not going to “get it” without them). Tony did not edit the second edition of this book, but the contents rely heavily on the first edition. This book would not be what it is without him.

Without a technical review team of the caliber I had during the writing of this book and the previous edition, I would be nervous about the content. The first edition had Jonathan Lewis, Roderick Manalac, Michael Möller, and Gabe Romanescu as technical reviewers. They spent many hours poring over the material and verifying it was technically accurate as well as useful in the real world. This second edition had a team of similar caliber: Melanie Caffrey, Christopher Beck, and Jason Straub. I firmly believe a technical book should be judged not only by who wrote it, but also by who reviewed it. Given these seven people, I feel confident in the material.

At Oracle, I work with the best and brightest people I have ever known, and they all have contributed in one way or another. I would like to thank Ken Jacobs in particular for his support and enthusiasm over the years. Ken is unfortunately (for us) no longer with Oracle Corporation, but his impact will long be felt.

Lastly, but most important, I would like to acknowledge the unceasing support I’ve received from my family. You know you must be important to someone when you try to do something that takes a lot of “outside of work hours” and that someone lets you know about it. Without the continual support of my wife, Melanie (who also was a technical reviewer on the book), son Alan, and daughter Megan, I don’t see how I could have finished this book.

—Tom Kyte

I’d like to thank Tom for inviting me to work with him on this book; this is a great technical honor. I’d also like to acknowledge Jonathan Gennick; his guidance (over many years and books) laid the foundation for me being able to work on a book of this caliber. And I’d like to thank Heidi, Lisa, and Brandi; without their support, I could not have successfully participated.

—Darl Kuhn

Introduction

The inspiration for the material contained in this book comes from my experiences developing Oracle software, and from working with fellow Oracle developers to help them build reliable and robust applications based on the Oracle database. The book is basically a reflection of what I do every day and of the issues I see people encountering each and every day.

I covered what I felt was most relevant, namely the Oracle database and its architecture. I could have written a similarly titled book explaining how to develop an application using a specific language and architecture—for example, one using JavaServer Pages that speaks to Enterprise JavaBeans, which in turn uses JDBC to communicate with Oracle. However, at the end of the day, you really do need to understand the topics covered in this book in order to build such an application successfully. This book deals with what I believe needs to be universally known to develop successfully with Oracle, whether you are a Visual Basic programmer using ODBC, a Java programmer using EJBs and JDBC, or a Perl programmer using DBI Perl. This book does not promote any specific application architecture; it does not compare three tier to client/server. Rather, it covers what the database can do and what you must understand about the way it works. Since the database is at the heart of any application architecture, the book should have a broad audience.

As the title suggests, *Expert Oracle Database Architecture* concentrates on the database architecture and how the database itself works. I cover the Oracle database architecture in depth: the files, memory structures, and processes that comprise an Oracle database and instance. I then move on to discuss important database topics such as locking, concurrency controls, how transactions work, and redo and undo, and why it is important for you to know about these things. Lastly, I examine the physical structures in the database such as tables, indexes, and datatypes, covering techniques for making optimal use of them.

What This Book Is About

One of the problems with having plenty of development options is that it's sometimes hard to figure out which one might be the best choice for your particular needs. Everyone wants as much flexibility as possible (as many choices as they can possibly have), but they also want things to be very cut and dried—in other words, easy. Oracle presents developers with almost unlimited choice. No one ever says, “You can’t do that in Oracle.” Rather, they say, “How many different ways would you like to do that in Oracle?” I hope that this book will help you make the correct choice.

This book is aimed at those people who appreciate the choice but would also like some guidelines and practical implementation details on Oracle features and functions. For example, Oracle has a really neat feature called parallel execution. The Oracle documentation tells you how to use this feature and what it does. Oracle documentation does not, however, tell you when you should use this feature and, perhaps even more important, when you should not use this feature. It doesn't always tell you the implementation details of this feature, and if you're not aware of them, this can come back to haunt you (I'm not referring to bugs, but the way the feature is supposed to work and what it was really designed to do).

In this book I strove to not only describe how things work, but also explain when and why you would consider using a particular feature or implementation. I feel it is important to understand not only the “how” behind things, but also the “when” and “why” as well as the “when not” and “why not!”

Who Should Read This Book

The target audience for this book is anyone who develops applications with Oracle as the database back end. It is a book for professional Oracle developers who need to know how to get things done in the database. The practical nature of the book means that many sections should also be very interesting to the DBA. Most of the examples in the book use SQL*Plus to demonstrate the key features, so you won't find out how to develop a really cool GUI—but you will find out how the Oracle database works, what its key features can do, and when they should (and should not) be used.

This book is for anyone who wants to get more out of Oracle with less work. It is for anyone who wants to see new ways to use existing features. It is for anyone who wants to see how these features can be applied in the real world (not just examples of how to use the feature, but why the feature is relevant in the first place). Another category of people who would find this book of interest is technical managers in charge of the developers who work on Oracle projects. In some respects, it is just as important that they understand why knowing the database is crucial to success. This book can provide ammunition for managers who would like to get their personnel trained in the correct technologies or ensure that personnel already know what they need to know.

To get the most out of this book, the reader should have

- *Knowledge of SQL.* You don't have to be the best SQL coder ever, but a good working knowledge will help.
- *An understanding of PL/SQL.* This isn't a prerequisite, but it will help you to absorb the examples. This book will not, for example, teach you how to program a FOR loop or declare a record type; the Oracle documentation and numerous books cover this well. However, that's not to say that you won't learn a lot about PL/SQL by reading this book. You will. You'll become very intimate with many features of PL/SQL, you'll see new ways to do things, and you'll become aware of packages/features that perhaps you didn't know existed.
- *Exposure to some third-generation language (3GL), such as C or Java.* I believe that anyone who can read and write code in a 3GL language will be able to successfully read and understand the examples in this book.
- *Familiarity with the Oracle Database Concepts manual.*

A few words on that last point: due to the Oracle documentation set's vast size, many people find it to be somewhat intimidating. If you're just starting out or haven't read any of it as yet, I can tell you that the *Oracle Database Concepts* manual is exactly the right place to start. It's about 450 pages long (I know that because I wrote some of the pages and edited every one) and touches on many of the major Oracle concepts that you need to know about. It may not give you each and every technical detail (that's what the other 10,000 to 20,000 pages of documentation are for), but it will educate you on all the important concepts. This manual touches the following topics (to name a few):

- The structures in the database, and how data is organized and stored
- Distributed processing
- Oracle's memory architecture
- Oracle's process architecture
- Schema objects you will be using (tables, indexes, clusters, and so on)
- Built-in datatypes and user-defined datatypes
- SQL stored procedures
- How transactions work
- The optimizer
- Data integrity
- Concurrency control

I will come back to these topics myself time and time again. These are the fundamentals. Without knowledge of them, you will create Oracle applications that are prone to failure. I encourage you to read through the manual and get an understanding of some of these topics.

How This Book Is Structured

To help you use this book, most chapters are organized into four general sections (described in the list that follows). These aren't rigid divisions, but they will help you navigate quickly to the area you need more information on. This book has 15 chapters, and each is like a "minibook"—a virtually stand-alone component. Occasionally, I refer to examples or features in other chapters, but you could pretty much pick a chapter out of the book and read it on its own. For example, you don't have to read Chapter 10 on database tables to understand or make use of Chapter 14 on parallelism.

The format and style of many of the chapters is virtually identical:

- An introduction to the feature or capability.
- Why you might want to use the feature or capability (or not). I outline when you would consider using this feature and when you would not want to use it.
- How to use this feature. The information here isn't just a copy of the material in the SQL reference; rather, it's presented in step-by-step manner: here is what you need, here is what you have to do, and these are the switches you need to go through to get started. Topics covered in this section will include:
 - How to implement the feature
 - Examples, examples, examples
 - How to debug this feature
 - Caveats of using this feature
 - How to handle errors (proactively)
 - A summary to bring it all together

There will be lots of examples and lots of code, all of which is available for download from the Source Code area of www.apress.com.

Source Code and Updates

The best way to digest the material in this book is to thoroughly work through and understand the hands-on examples. As you work through the examples in this book, you may decide that you prefer to type in all the code by hand. Many readers choose to do this because it is a good way to get familiar with the coding techniques that are being used.

If you like to type in the code, you can use the source code files to check the results you should be getting—they should be your first stop if you think you might have typed an error. If you don't like typing, then downloading the source code from the Apress web site is a must! Either way, the code files will help you with updates and debugging.

Errata

Apress makes every effort to make sure that there are no errors in the text or the code. However, to err is human, and as such we recognize the need to keep you informed of any mistakes as they're discovered and corrected. Errata sheets are available for all our books at www.apress.com. If you find an error that hasn't already been reported, please let us know. The Apress web site acts as a focus for other information and support, including the code from all Apress books, sample chapters, previews of forthcoming titles, and articles on related topics.

Setting Up Your Environment

In this section, I will cover how to set up an environment capable of executing the examples in this book. Specifically:

- How to setup the EODA account used for many of the examples in this book
- How to set up the SCOTT/TIGER demonstration schema properly
- The environment you need to have up and running
- Configuring AUTOTRACE, a SQL*Plus facility
- Installing StatsPack
- Installing and running runstats, and other custom utilities used throughout the book
- The coding conventions I use in this book

All of the non-Oracle supplied scripts are available for download from the www.apress.com website. If you download the scripts, there will be a chNN folder that contains the scripts for each chapter (where NN is the number of the chapter). The ch00 folder contains the scripts listed here in the Setting Up Your Environment section.

Setting up the EODA Schema

The EODA user is used for most of the examples in this book. This is simply a schema that has been granted the DBA role and granted execute and select on certain objects owned by SYS:

```
connect / as sysdba
define username=eoda
define usernamepwd=foo
create user &username identified by &usernamepwd;
grant dba to &username;
grant execute on dbms_stats to &username;
grant select on V_$STATNAME to &username;
grant select on V_$MYSTAT to &username;
grant select on V_$LATCH to &username;
grant select on V_$TIMER to &username;
conn &username/&usernamepwd
```

You can setup whatever user you want to run the examples in this book. I picked the username of EODA simply because it's an acronym for the title of the book.

Setting up the SCOTT/TIGER Schema

The SCOTT/TIGER schema will often already exist in your database. It is generally included during a typical installation, but it is not a mandatory component of the database. You may install the SCOTT example schema into any database account; there is nothing magic about using the SCOTT account. You could install the EMP/DEPT tables directly into your own database account if you wish.

Many of my examples in this book draw on the tables in the SCOTT schema. If you would like to be able to work along with them, you will need these tables. If you are working on a shared database, it would be advisable to install your own copy of these tables in some account other than SCOTT to avoid side effects caused by other users mucking about with the same data.

Executing the Script

In order to create the SCOTT demonstration tables, you will simply:

- `cd $ORACLE_HOME/sqlplus/demo`
- run `demobld.sql` when connected as any user

The `demobld.sql` script will create and populate five tables. When it is complete, it exits SQL*Plus automatically, so don't be surprised when SQL*Plus disappears after running the script—it's supposed to do that.

The standard demo tables do not have any referential integrity defined on them. Some of my examples rely on them having referential integrity. After you run `demobld.sql`, it is recommended you also execute the following:

```
alter table emp add constraint emp_pk primary key(empno);
alter table dept add constraint dept_pk primary key(deptno);
alter table emp add constraint emp_fk_dept foreign key(deptno) references dept;
alter table emp add constraint emp_fk_emp foreign key(mgr) references emp;
```

If you would like to drop this schema at any time to clean up, you can simply execute `$ORACLE_HOME/sqlplus/demo/demodrop.sql`. This will drop the five tables and exit SQL*Plus.

■ **Tip** You can also find the SQL to create and drop the SCOTT user in the `$ORACLE_HOME/rdbms/admin/utlsampl.sql` script.

Setting Your Environment

Most of the examples in this book are designed to run 100 percent in the SQL*Plus environment. Other than SQL*Plus though, there is nothing else to set up and configure. I can make a suggestion, however, on using SQL*Plus. Almost all of the examples in this book use `DBMS_OUTPUT` in some fashion. In order for `DBMS_OUTPUT` to work, the SQL*Plus command

```
SQL> set serveroutput on
```

must be issued. If you are like me, typing this in each and every time would quickly get tiresome. Fortunately, SQL*Plus allows us to setup a `login.sql` file, a script that is executed each and every time we start SQL*Plus. Further, it allows us to set an environment variable, `SQLPATH`, so that it can find this `login.sql` script, no matter what directory it is in. The `login.sql` script I use for all examples in this book is:

```
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
set sqlprompt '&_user. @&_connect_identifier.> '
```

The last bit in the `login.sql` sets up my SQL*Plus prompt for me:

```
set sqlprompt '&_user. @&_connect_identifier.> '
```

That makes my prompt look like this

```
EODA@ORA12CR1>
```

so I know who I am as well as where I am.

Setting up Autotrace in SQL*Plus

AUTOTRACE is a facility within SQL*Plus to show us the explain plan of the queries we've executed, and the resources they used. This book makes extensive use of this facility. There is more than one way to get AUTOTRACE configured.

Initial Setup

AUTOTRACE relies on a table named `PLAN_TABLE` being available. Starting with Oracle 10g, the `SYS` schema contains a global temporary table named `PLAN_TABLE$`. All required privileges to this table have been granted to `PUBLIC` and there is a public synonym (named `PLAN_TABLE` that points to `SYS.PLAN_TABLE$`). This means any user can access this table. You must also create and grant the `PLUSTRACE` role:

- `cd $ORACLE_HOME/sqlplus/admin`
- log into SQL*Plus as `SYS` or as a user granted the `SYSDBA` privilege
- `run @plustrce`
- `run GRANT PLUSTRACE TO PUBLIC;`

You can replace `PUBLIC` in the `GRANT` command with some user if you want.

Controlling the Report

You can automatically get a report on the execution path used by the SQL optimizer and the statement execution statistics. The report is generated after successful SQL DML (that is, `SELECT`, `DELETE`, `UPDATE`, `MERGE`, and `INSERT`) statements. It is useful for monitoring and tuning the performance of these statements.

Setting up StatsPack

StatsPack is designed to be installed when connected as SYS or as a user granted the SYSDBA privilege. In many installations, installing StatsPack will be a task that you must ask the DBA or administrators to perform. Installing StatsPack is trivial. You simply run `@spcreate.sql`. This script will be found in `$ORACLE_HOME/rdbms/admin` and should be executed when connected as SYS via SQL*Plus.

You'll need to know three pieces of information before running the `spcreate.sql` script. They are:

- The password you would like to use for the PERFSTAT schema that will be created
- The default tablespace you would like to use for PERFSTAT
- The temporary tablespace you would like to use for PERFSTAT

Running the script will look something like this:

```
$ sqlplus / as sysdba
```

The script will prompt you for the needed information as it executes. In the event you make a typo or inadvertently cancel the installation, you should use `spdrop.sql` found in `$ORACLE_HOME/rdbms/admin` to remove the user and installed views prior to attempting another install of StatsPack. The StatsPack installation will create a file called `spcpkg.lis`. You should review this file for any possible errors that might have occurred. The user, views, and PL/SQL code should install cleanly, however, as long as you supplied valid tablespace names (and didn't already have a user PERFSTAT).

■ **Tip** StatsPack is documented in the following text file: `$ORACLE_HOME/rdbms/admin/spdoc.txt`

Custom Scripts

In this section, I will describe the requirements (if any) needed by various scripts used throughout this book. As well, we will investigate the code behind the scripts.

Runstats

Runstats is a tool I developed to compare two different methods of doing the same thing and show which one is superior. You supply the two different methods and Runstats does the rest. Runstats simply measures three key things:

- Wall clock or elapsed time— This is useful to know, but not the most important piece of information.
- System statistics—This shows, side by side, how many times each approach did something (such as a parse call, for example) and the difference between the two.
- Latching—This is the key output of this report.

As we'll see in this book, latches are a type of lightweight lock. Locks are serialization devices. Serialization devices inhibit concurrency. Applications that inhibit concurrency are less scalable, can support fewer users, and require more resources. Our goal is always to build applications that have the potential to scale—ones that can service one user as well as 1,000 or 10,000. The less latching we incur in our approaches, the better off we will be. I might

choose an approach that takes longer to run on the wall clock but that uses 10 percent of the latches. I know that the approach that uses fewer latches will scale substantially better than the approach that uses more latches.

Runstats is best used in isolation; that is, on a single-user database. We will be measuring statistics and latching (locking) activity that result from our approaches. We do not want other sessions to contribute to the system's load or latching while this is going on. A small test database is perfect for these sorts of tests. I frequently use my desktop PC or laptop, for example.

In order to use Runstats, you need to set up access to several V\$ views, create a table to hold the statistics, and create the Runstats package. You will need access to four V\$ tables (those magic, dynamic performance tables): V\$STATNAME, V\$MYSTAT, V\$TIMER and V\$LATCH. Here is a view I use:

```
create or replace view stats
as select 'STAT...' || a.name name, b.value
    from v$statname a, v$mystat b
    where a.statistic# = b.statistic#
union all
select 'LATCH.' || name, gets
    from v$latch
union all
select 'STAT...Elapsed Time', hsecs from v$timer;
```

■ **Note** The actual object names you need to be granted access to will be V_\$STATNAME, V_\$MYSTAT, and so on—that is, the object name to use in the grant will start with V_\$ not V\$. The V\$ name is a synonym that points to the underlying view with a name that starts with V_\$. So, V\$STATNAME is a synonym that points to V_\$STATNAME – a view. You need to be granted access to the view.

You can either have SELECT on V\$STATNAME, V\$MYSTAT, V\$TIMER, and V\$LATCH granted directly to you (so you can create the view yourself) or you can have someone that does have SELECT on those objects create the view for you and grant SELECT privileges on the view to you.

Once you have that set up, all you need is a small table to collect the statistics:

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

Last, you need to create the package that is Runstats. It contains three simple API calls:

- RS_START (Runstats Start) to be called at the beginning of a Runstats test
- RS_MIDDLE to be called in the middle, as you might have guessed
- RS_STOP to finish off and print the report

The code for creating this package can be downloaded from the Apress website (www.apress.com). In the code section for this book look for the `runstats.sql` script in the `ch00` folder. It contains all code required to create the body of the `runstats_pkg` package. Here is the package specification:

```
EODA@ORA12CR1> create or replace package runstats_pkg
2  as
3      procedure rs_start;
4      procedure rs_middle;
5      procedure rs_stop( p_difference_threshold in number default 0 );
6  end;
7  /
```

The parameter, `p_difference_threshold`, is used to control the amount of data printed at the end. Runstats collects statistics and latching information for each run, and then prints a report of how much of a resource each test (each approach) used and the difference between them. You can use this input parameter to see only the statistics and latches that had a difference greater than this number. By default, this is zero, and you see all of the outputs.

The `RS_START` routine will simply clear out our statistics holding table and then populate it with the “before” statistics and latches. It will then capture the current timer value, a clock of sorts that we can use to compute elapsed times in hundredths of seconds.

The `RS_MIDDLE` routine simply records the elapsed time for the first run of our test in `G_RUN1`. Then it inserts the current set of statistics and latches. If we were to subtract these values from the ones we saved previously in `RS_START`, we could discover how many latches the first method used, how many cursors (a statistic) it used, and so on.

The next and final routine in this package is the `RS_STOP` routine. Its job is to print out the aggregate CPU times for each run and then print out the difference between the statistic/latching values for each of the two runs (only printing out those that exceed the threshold).

By way of example, we'll demonstrate how to use Runstats to see which is more efficient, a single bulk INSERT versus row-by-row processing. We'll start by setting up two tables into which we'll insert 1,000,000 rows (the `BIG_TABLE` table creation script is provided later in this section):

```
EODA@ORA12CR1> create table t1
2  as
3  select * from big_table
4  where 1=0;
```

```
EODA@ORA12CR1> create table t2
2  as
3  select * from big_table
4  where 1=0;
```

And now we are ready to perform the first method of inserting the records, using a single SQL statement. We start by calling `RUNSTATS_PKG.RS_START`:

```
EODA@ORA12CR1> exec runstats_pkg.rs_start;
```

```
EODA@ORA12CR1> insert into t1
2  select *
3  from big_table
4  where rownum <= 1000000;
1000000 rows created.
```

```
EODA@ORA12CR1> commit;
```


Now we are ready to perform the second method, row by row insertion of data:

```
EODA@ORA12CR1> exec runstats_pkg.rs_middle;

EODA@ORA12CR1> begin
2         for x in ( select *
3                     from big_table
4                     where rownum <= 1000000 )
5         loop
6             insert into t2 values X;
7         end loop;
8         commit;
9     end;
10 /
```

And finally, we'll generate the report:

```
EODA@ORA12CR1> exec runstats_pkg.rs_stop(1000000)
```

Here's a partial listing of the output:

```
Run1 ran in 119 cpu hsecs
Run2 ran in 3376 cpu hsecs
run 1 ran in 3.52% of the time
```

Name	Run1	Run2	Diff
STAT...execute count	29	1,000,032	1,000,003
...			
STAT...logical read bytes from	1,114,251,264	9,300,803,584	8,186,552,320

Run1 latches total versus runs -- difference and pct			
Run1	Run2	Diff	Pct
555,593	6,795,317	6,239,724	8.18%

This confirms you have the RUNSTATS_PKG package installed and shows you why you should use a single SQL statement instead of a bunch of procedural code when developing applications whenever possible!

Mystat

The `mystat.sql` and its companion, `mystat2.sql`, are used to show the increase in some Oracle “statistic” before and after some operation. The `mystat.sql` script captures the begin value of some statistic

```
set echo off
set verify off
column value new_val V
define S="&1"

set autotrace off
select a.name, b.value
from v$statname a, v$mystat b
```

```
where a.statistic# = b.statistic#
and lower(a.name) = lower('&S')
/
set echo on
```

and `mystat2.sql` reports the difference (&V is populated by running the first script, `mystat.sql`—it uses the SQL*Plus `NEW_VAL` feature for that. It contains the last `VALUE` selected from the query above):

```
set echo off
set verify off
select a.name, b.value V, to_char(b.value-&V,'999,999,999,999') diff
from v$statname a, v$mystat b
where a.statistic# = b.statistic#
and lower(a.name) = lower('&S')
/
set echo on
```

For example, to see how much redo is generated by some `UPDATE` statement we can:

```
EODA@ORA12CR1> @mystat "redo size"
EODA@ORA12CR1> set echo off
```

NAME	VALUE
redo size	491167892

```
EODA@ORA12CR1> update big_table set owner = lower(owner)
2 where rownum <= 1000;
```

1000 rows updated.

```
EODA@ORA12CR1> @mystat2
EODA@ORA12CR1> set echo off
```

NAME	V	DIFF
redo size	491265640	97,748

This shows our `UPDATE` of 1,000 rows generated 97,748 bytes of redo.

Show_Space

The SHOW_SPACE routine prints detailed space utilization information for database segments. Here is the interface to it:

```
EODA@ORA12CR1> desc show_space
```

```
PROCEDURE show_space
```

Argument Name	Type	In/Out	Default?
P_SEGNAME	VARCHAR2	IN	
P_OWNER	VARCHAR2	IN	DEFAULT
P_TYPE	VARCHAR2	IN	DEFAULT
P_PARTITION	VARCHAR2	IN	DEFAULT

The arguments are as follows:

- P_SEGNAME - Name of the segment—the table or index name, for example.
- P_OWNER - Defaults to the current user, but you can use this routine to look at some other schema.
- P_TYPE - Defaults to TABLE and represents the type of object you are looking at. For example, `select distinct segment_type from dba_segments` lists valid segment types.
- P_PARTITION - Name of the partition when you show the space for a partitioned object. SHOW_SPACE shows space for only a partition at a time.

The output of this routine looks as follows, when the segment resides in an Automatic Segment Space Management (ASSM) tablespace:

```
EODA@ORA12CR1> exec show_space('BIG_TABLE');
```

```
Unformatted Blocks ..... 0
FS1 Blocks (0-25) ..... 0
FS2 Blocks (25-50) ..... 0
FS3 Blocks (50-75) ..... 0
FS4 Blocks (75-100)..... 0
Full Blocks ..... 14,469
Total Blocks..... 15,360
Total Bytes..... 125,829,120
Total MBytes..... 120
Unused Blocks..... 728
Unused Bytes..... 5,963,776
Last Used Ext FileId..... 4
Last Used Ext BlockId..... 43,145
Last Used Block..... 296
```

The items reported are as follows:

- Unformatted Blocks - The number of blocks that are allocated to the table below the high water mark, but have not been used. Add unformatted and unused blocks together to get a total count of blocks allocated to the table but never used to hold data in an ASSM object.
- FS1 Blocks-FS4 Blocks - Formatted blocks with data. The ranges of numbers after their name represent the emptiness of each block. For example, (0-25) is the count of blocks that are between 0 and 25 percent empty.

- Full Blocks – The number of blocks that are so full that they are no longer candidates for future inserts.
- Total Blocks, Total Bytes, Total Mbytes - The total amount of space allocated to the segment measured in database blocks, bytes, and megabytes.
- Unused Blocks, Unused Bytes – Represents a portion of the amount of space never used. These are blocks allocated to the segment but are currently above the high water mark of the segment
- Last Used Ext FileId – The file ID of the file that contains the last extent that contains data.
- Last Used Ext BlockId – The block ID of the beginning of the last extent; the block ID within the last-used file.
- Last Used Block – The block ID offset of the last block used in the last extent.

The code for creating the SHOW_SPACE procedure can be downloaded from the Apress website (www.apress.com). This utility is a simple layer on top of the DBMS_SPACE API in the database.

Big_Table

For examples throughout this book, I use a table called BIG_TABLE. Depending on which system I use, this table has between one record and four million records and varies in size from 200MB to 800MB. In all cases, the table structure is the same. To create BIG_TABLE, I wrote a script that does the following:

- Creates an empty table based on ALL_OBJECTS. This dictionary view is used to populate the BIG_TABLE.
- Makes this table NOLOGGING. This is optional. I did it for performance. Using NOLOGGING mode for a test table is safe; you won't use it in a production system, so features like Oracle Data Guard will not be enabled.
- Populates the table by seeding it with the contents of ALL_OBJECTS and then iteratively inserting into itself, approximately doubling its size on each iteration. Additionally it creates a primary key constraint on the table and gathers statistics.

To build the BIG_TABLE table, you can run the following script at the SQL*Plus prompt and pass in the number of rows you want in the table. The script will stop when it hits that number of rows.

```
create table big_table
as
select rownum id, OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID,
DATA_OBJECT_ID, OBJECT_TYPE, CREATED, LAST_DDL_TIME, TIMESTAMP,
STATUS, TEMPORARY, GENERATED, SECONDARY, NAMESPACE, EDITION_NAME
from all_objects
where 1=0
/

alter table big_table nologging;

declare
  l_cnt number;
  l_rows number := &numrows;
```

```

begin
  insert /*+ append */
  into big_table
  select rownum id, OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID,
  DATA_OBJECT_ID, OBJECT_TYPE, CREATED, LAST_DDL_TIME, TIMESTAMP,
  STATUS, TEMPORARY, GENERATED, SECONDARY, NAMESPACE, EDITION_NAME
  from all_objects
  where rownum <= &numrows;
--
  l_cnt := sql%rowcount;
  commit;
  while (l_cnt < l_rows)
  loop
    insert /*+ APPEND */ into big_table
    select rownum+l_cnt, OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID,
    DATA_OBJECT_ID, OBJECT_TYPE, CREATED, LAST_DDL_TIME, TIMESTAMP,
    STATUS, TEMPORARY, GENERATED, SECONDARY, NAMESPACE, EDITION_NAME
    from big_table a
    where rownum <= l_rows-l_cnt;
    l_cnt := l_cnt + sql%rowcount;
    commit;
  end loop;
end;
/

alter table big_table add constraint
big_table_pk primary key(id);

exec dbms_stats.gather_table_stats( user, 'BIG_TABLE', estimate_percent=> 1);

```

I estimated baseline statistics on the table. The index associated with the primary key will have statistics computed automatically when it is created.

Coding Conventions

The one coding convention I use in this book that I would like to point out is how I name variables in PL/SQL code. For example, consider a package body like this:

```

create or replace package body my_pkg
as
  g_variable varchar2(25);

  procedure p( p_variable in varchar2 )
  is
    l_variable varchar2(25);
  begin
    null;
  end;
end;
/

```

Here I have three variables: a global package variable, `G_VARIABLE`; a formal parameter to the procedure, `P_VARIABLE`; and a local variable, `L_VARIABLE`. I name my variables after the scope they are contained in. All globals begin with `G_`, parameters with `P_`, and local variables with `L_`. The main reason for this is to distinguish PL/SQL variables from columns in a database table. I just always name my variables after the scope. That way, I can easily distinguish parameters from local variables and global variables, in addition to removing any ambiguity with respect to column names and variable names.