

THE EXPERT'S VOICE® IN ORACLE

# Pro Oracle SQL

*Exploit the full power of SQL and  
supporting features in Oracle Database*



Karen Morton, Kerry Osborne, Robyn Sands  
Riyaj Shamsudeen, and Jared Still

Apress®

# Pro Oracle SQL



Karen Morton

Kerry Osborne

Robyn Sands

Riyaj Shamsudeen

Jared Still

Apress®

## **Pro Oracle SQL**

Copyright © 2010 by Karen Morton, Kerry Osborne, Robyn Sands, Riyaj Shamsudeen, and Jared Still

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-3228-5

ISBN-13 (electronic): 978-1-4302-3229-2

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

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.

President and Publisher: Paul Manning

Lead Editor: Jonathan Gennick

Technical Reviewers: Christopher Beck, Iggy Fernandez, and Bernard Lopuz

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Coordinating Editor: Anita Castro

Copy Editor: Mary Behr

Compositor: Lynn L'Heureux

Indexer: Julie Grady

Artist: April Milne

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://www.springeronline.com).

For information on translations, please e-mail [rights@apress.com](mailto:rights@apress.com), or visit [www.apress.com](http://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/info/bulksales](http://www.apress.com/info/bulksales).

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

# Contents at a Glance

■ About the Authors .....	xv
■ About the Technical Reviewer .....	xvii
■ Introduction .....	xix
■ Chapter 1: Core SQL .....	1
■ Chapter 2: SQL Execution .....	29
■ Chapter 3: Access and Join Methods .....	61
■ Chapter 4: SQL is About Sets .....	105
■ Chapter 5: It's About the Question .....	129
■ Chapter 6: SQL Execution Plans .....	153
■ Chapter 7: Advanced Grouping .....	191
■ Chapter 8: Analytic Functions .....	221
■ Chapter 9: The Model Clause .....	251
■ Chapter 10: Subquery Factoring .....	283
■ Chapter 11: Semi-joins and Anti-joins .....	325
■ Chapter 12: Indexes .....	373
■ Chapter 13: Beyond the SELECT .....	403
■ Chapter 14: Transaction Processing .....	433
■ Chapter 15: Testing and Quality Assurance .....	465
■ Chapter 16: Plan Stability and Control .....	497
■ Index .....	565

# Contents

■ <b>About the Authors .....</b>	<b>xv</b>
■ <b>About the Technical Reviewer.....</b>	<b>xvii</b>
■ <b>Introduction.....</b>	<b>xix</b>
■ <b>Chapter 1: Core SQL.....</b>	<b>1</b>
The SQL Language .....	1
Interfacing to the Database .....	2
Review of SQL*Plus .....	3
Connect to a Database .....	3
Configuring the SQL*Plus environment.....	4
Executing Commands .....	6
The Five Core SQL Statements .....	8
The SELECT Statement.....	9
The FROM Clause .....	10
The WHERE Clause .....	11
The GROUP BY Clause.....	12
The HAVING Clause .....	13
The SELECT List.....	13
The ORDER BY Clause.....	14
The INSERT Statement .....	15
Single-table Inserts .....	15
Multi-table Inserts .....	16
The UPDATE Statement .....	18
The DELETE Statement.....	22
The MERGE Statement .....	24
Summary .....	27

<b>Chapter 2: SQL Execution .....</b>	<b>29</b>
Oracle Architecture Basics .....	29
SGA – The Shared Pool .....	31
The Library Cache .....	32
Identical Statements.....	33
SGA – The Buffer Cache .....	37
Query Transformation.....	39
View Merging .....	40
Subquery Unnesting .....	44
Predicate Pushing .....	47
Query Rewrite with Materialized Views .....	50
Determining the Execution Plan .....	52
Executing the Plan and Fetching Rows .....	56
SQL Execution – Putting It All Together.....	59
Summary .....	60
 <b>Chapter 3: Access and Join Methods.....</b>	 <b>61</b>
Full Scan Access Methods.....	61
How Full Scan Operations are Chosen .....	62
Full Scans and Throwaway.....	65
Full Scans and Multiblock Reads.....	66
Full Scans and the Highwater Mark.....	67
Index Scan Access Methods .....	73
Index Structure .....	74
Index Scan Types.....	75
Index Unique Scan .....	78
Index range scan.....	79
Index Full Scan .....	81
Index Skip Scan .....	85
Index Fast Full Scan .....	87
Join Methods .....	88
Nested Loops Joins.....	89
Sort-Merge Joins .....	92

Hash Joins .....	94
Cartesian Joins .....	96
Outer Joins .....	98
Summary .....	104
<b>Chapter 4: SQL is About Sets.....</b>	<b>105</b>
Thinking in Sets .....	105
Moving from Procedural to Set-based Thinking .....	106
Procedural vs Set-based Thinking: An Example .....	111
Set Operations .....	113
UNION and UNION ALL .....	114
MINUS .....	117
INTERSECT.....	119
Sets and Nulls .....	119
NULLs and Unintuitive Results.....	120
NULL Behavior in Set Operations.....	123
NULLs and GROUP BY and ORDER BY.....	124
NULLs and Aggregate Functions.....	126
Summary .....	127
<b>Chapter 5: It's About the Question .....</b>	<b>129</b>
Asking Good Questions.....	129
The Purpose of Questions.....	130
Categories of Questions.....	130
Questions about the Question .....	133
Questions about Data .....	135
Building Logical Expressions .....	141
Summary .....	154
<b>Chapter 6: SQL Execution Plans .....</b>	<b>153</b>
Explain Plans.....	153
Using Explain Plan.....	153
Understanding How EXPLAIN PLAN can Miss the Mark.....	160
Reading the Plan.....	163

Execution Plans .....	166
Viewing Recently Generated SQL.....	166
Viewing the Associated Execution Plan.....	166
Collecting the Plan Statistics .....	168
Identifying SQL Statements for Later Plan Retrieval .....	171
Understanding DBMS_XPLAN in Detail .....	174
Using Plan Information for Solving Problems .....	180
Summary .....	189
 <b>■ Chapter 7: Advanced Grouping.....</b>	<b>191</b>
Basic GROUP BY Usage .....	192
HAVING Clause .....	195
“New” GROUP BY Functionality .....	197
CUBE Extension to GROUP BY .....	197
Putting CUBE To Work .....	201
Eliminate NULLs with the GROUPING() Function .....	207
Extending Reports with GROUPING() .....	209
Extending Reports With GROUPING_ID().....	210
GROUPING SETS and ROLLUP().....	214
GROUP BY Restrictions .....	217
Summary .....	220
 <b>■ Chapter 8: Analytic Functions .....</b>	<b>221</b>
Example Data .....	221
Anatomy of Analytic Functions.....	222
List of Functions.....	223
Aggregation Functions.....	224
Aggregate Function Over An Entire Partition .....	225
Granular Window Specifications .....	226
Default Window Specification .....	227
Lead and Lag .....	227
Syntax and Ordering .....	227
Example 1: Returning a Value from Prior Row .....	227
Understanding that Offset is in Rows.....	228
Example 2: Returning a Value from an Upcoming Row .....	229



First_value & Last_value .....	230
Example: First_value to Calculate Maximum .....	231
Example: Last_value to Calculate Minimum .....	231
Other Analytic Functions.....	232
Nth_value (11gR2) .....	232
Rank.....	234
Dense_rank .....	235
Row_number .....	236
Ratio_to_report.....	237
Percent_rank .....	238
Percentile_cont.....	238
Percentile_disc .....	240
NTILE.....	241
Stddev .....	242
Listagg.....	243
Performance Tuning .....	243
Execution Plans.....	244
Predicates.....	244
Indexes .....	246
Advanced topics.....	246
Dynamic SQL .....	247
Nesting Analytic Functions .....	248
Parallelism.....	249
PGA size .....	250
Organizational Behavior.....	250
Summary .....	250
<b>■ Chapter 9: The Model Clause.....</b>	<b>251</b>
Spreadsheets .....	251
Inter-Row Referencing via the Model clause.....	252
Example Data.....	252
Anatomy of a Model Clause.....	253
Rules .....	254
Positional and Symbolic References .....	255
Positional Notation .....	256

Symbolic Notation .....	257
FOR Loops .....	258
Returning Updated Rows .....	258
Evaluation Order .....	260
Row Evaluation Order .....	260
Rule Evaluation Order .....	262
Aggregation .....	263
Iteration .....	264
An Example .....	265
PRESENTV and NULLs .....	266
Lookup Tables .....	267
NULLs .....	269
Performance Tuning with the Model Clause .....	271
Execution Plans .....	271
Predicate Pushing .....	274
Materialized Views .....	276
Parallelism .....	277
Partitioning in Model Clause Execution .....	278
Indexes .....	280
Subquery Factoring .....	281
Summary .....	282
<b>■ Chapter 10: Subquery Factoring .....</b>	<b>283</b>
Standard Usage .....	283
Optimizing SQL .....	286
Testing Execution Plans .....	286
Testing Over Multiple Executions .....	290
Testing the Effects of Query Changes .....	293
Seizing Other Optimization Opportunities .....	296
Applying Subquery Factoring to PL/SQL .....	301
Recursive Subqueries .....	304
A CONNECT BY Example .....	304
The Example Using an RSF .....	306
Restrictions on RSF .....	307
Differences from CONNECT BY .....	308

Duplicating CONNECT BY Functionality .....	309
The LEVEL Pseudocolumn .....	309
The SYS_CONNECT_BY_PATH Function .....	311
The CONNECT_BY_ROOT Operator.....	313
The CONNECT_BY_ISCYCLE Pseudocolumn and NOCYCLE Parameter .....	316
The CONNECT_BY_ISLEAF Pseudocolumn .....	319
Summary .....	324
<b>■ Chapter 11: Semi-joins and Anti-joins .....</b>	<b>325</b>
Semi-joins.....	325
Semi-join Plans .....	334
Controlling Semi-join Plans .....	339
Controlling Semi-join Plans Using Hints.....	339
Controlling Semi-join Plans at the Instance Level .....	342
Semi-join Restrictions .....	345
Semi-join Requirements .....	347
Anti-joins .....	347
Anti-join Plans.....	353
Controlling Anti-join Plans .....	363
Controlling Anti-join Plans Using Hints.....	363
Controlling Anti-join Plans at the Instance Level .....	364
Anti-join Restrictions .....	367
Anti-join Requirements.....	371
Summary .....	371
<b>■ Chapter 12: Indexes .....</b>	<b>373</b>
Understanding Indexes .....	374
When to use Indexes.....	374
Choice of Columns .....	376
The Null Issue .....	378
Index Structural Types.....	379
B-tree indexes .....	379
Bitmap Indexes .....	380
Index Organized Tables .....	381

Partitioned Indexes.....	383
Local Indexes.....	384
Global Indexes .....	385
Hash Partitioning vs. Range Partitioning .....	386
Solutions to Match Application Characteristics.....	390
Compressed Indexes.....	390
Function Based Indexes .....	392
Reverse Key Indexes .....	395
Descending Indexes .....	396
Solutions to Management Problems .....	397
Invisible Indexes.....	397
Virtual Indexes .....	399
Bitmap Join Indexes.....	400
Summary .....	402
<b>■ Chapter 13: Beyond the SELECT .....</b>	<b>403</b>
INSERT .....	403
Direct Path Inserts .....	403
Multi-Table Inserts .....	406
Conditional Insert .....	407
DML Error Logging .....	409
UPDATE.....	417
DELETE .....	424
MERGE.....	428
Syntax and Usage .....	428
Performance Comparison.....	432
Summary .....	435
<b>■ Chapter 14: Transaction Processing .....</b>	<b>433</b>
What is a Transaction? .....	433
ACID Properties of a Transaction .....	434
Transaction Isolation Levels .....	435
Multi-Version Read Consistency .....	437

Transaction Control Statements.....	438
Commit.....	438
Savepoint.....	438
Rollback.....	438
Set Transaction .....	438
Set Constraints .....	439
Grouping Operations into Transactions .....	439
The Order Entry Schema .....	440
The Active Transaction .....	447
Using Savepoints.....	449
Serializing Transactions .....	452
Isolating Transactions .....	455
Autonomous Transactions .....	458
Summary .....	463
<b>■ Chapter 15: Testing and Quality Assurance.....</b>	<b>465</b>
Test Cases .....	466
Testing Methods.....	467
Unit Tests.....	468
Regression Tests.....	472
Schema Changes .....	472
Repeating the Unit Tests .....	476
Execution Plan Comparison .....	478
Instrumentation .....	484
Adding Instrumentation to Code.....	484
Testing for Performance .....	488
Testing to Destruction .....	490
Troubleshooting through Instrumentation .....	491
Summary .....	495
<b>■ Chapter 16: Plan Stability and Control .....</b>	<b>497</b>
Plan Instability: Understanding The Problem .....	497
Changes to Statistics .....	498
Changes to the Environment .....	500

Changes to the SQL .....	502
Bind Variable Peeking .....	502
Identifying Plan Instability .....	505
Capturing Data on Currently-Running Queries .....	505
Reviewing the History of a Statement's Performance.....	506
Aggregating Statistics by Plan .....	508
Looking for Statistical Variance by Plan .....	509
Checking for Variations Around a Point in Time .....	511
Plan Control: Solving the Problem.....	513
Modifying Query Structure.....	513
Making Appropriate Use of Literals.....	514
Giving the Optimizer some Hints.....	514
Plan Control: Without Access to the Code) .....	522
Option 1: Change the Statistics .....	523
Option 2: Change Database Parameters .....	525
Option 3: Add or Remove Access Paths .....	525
Option 4: Apply Hint Based Plan-Control Mechanisms .....	526
Outlines .....	526
SQL Profiles .....	538
SQL Plan Baselines .....	555
Hint Based Plan Control Mechanisms Wrap Up .....	562
Conclusion .....	562
 <b>■ Index.....</b>	 <b>565</b>

# About the Authors



■ **KAREN MORTON** is a consultant and educator specializing in application optimization in both shoulder-to-shoulder consulting engagements and classroom settings. She is a Senior DBA Performance and Tuning Specialist for Fidelity Information Services. For over 20 years, Karen has worked in information technology. Starting as a mainframe programmer and developer, she has been a DBA, a data architect, and now is a researcher, educator, and consultant. Having used Oracle since the early 90s, she began teaching others how to use Oracle over a decade ago.

Karen is a frequent speaker at conferences and user groups, an Oracle ACE, and a member of the OakTable network (an informal association of “Oracle scientists” that are well known throughout the Oracle community). She blogs at [karenmorton.blogspot.com](http://karenmorton.blogspot.com).



■ **KERRY OSBORNE** began working with Oracle (version 2) in 1982. He has worked as both a developer and a DBA. For the past several years, he has been focused on understanding Oracle internals and solving performance problems. He is an OakTable member and is the author of an upcoming Apress book on Exadata. Kerry is a frequent speaker at Oracle conferences. Mr. Osborne is also a co-founder of Enkitec, an Oracle-focused consulting company headquartered in Dallas, Texas. He blogs at [kerryosborne.oracle-guy.com](http://kerryosborne.oracle-guy.com).



■ **ROBYN SANDS** is a software engineer for Cisco Systems, where she designs and develops embedded Oracle database products for Cisco customers. She has been working with Oracle since 1996, and has extensive experience in application development, large system implementations, and performance measurement. Robyn began her work career in industrial and quality engineering, and has combined her prior education and experience with her love of data by searching for new ways to build database systems with consistent performance and minimal maintenance requirements. She is a member of the OakTable network and a co-author of *Expert Oracle Practices: Oracle Database Administration from the Oak Table* (Apress, 2010). Robyn occasionally posts random blog entries at [adhdoddba.blogspot.com](http://adhdoddba.blogspot.com).



■ **RIYAJ SHAMSUDEEN** is the principal DBA and President of OraInternals, a performance/recovery/E-Business consulting company. He specializes in RAC, performance tuning, and database internals. He frequently blogs about these technology areas in his blog [orainternals.wordpress.com](http://orainternals.wordpress.com). He is also a regular presenter in US and international conferences. He is a proud member of OakTable network and an Oracle ACE. He has 19 years of experience using Oracle technology products and 18 years as an Oracle DBA/Oracle Applications DBA.



■ **JARED STILL** has been wrangling Oracle databases for longer than he cares to remember. During that time he has learned enough about SQL to realize that there will always be more to learn about SQL. He believes that everyone that queries an Oracle database should gain enough mastery of the SQL language that writing effective queries should become second nature. His participation as a co-author of *Pro Oracle SQL* is one way to help others achieve that goal. When Jared isn't managing databases, he likes to tinker with and race fast cars.



# About the Technical Reviewers



■ **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 Master Principal Technologist focusing on core database technologies. He is a co-inventor of two U.S. Patents on software methodologies that were the basis for what is now known as Oracle Application Express. Chris has reviewed other Oracle books including *Expert One-On-One* (Peer Information, 2001) and *Expert Oracle Database Architecture* (Apress, 2005), both by Tom Kyte, 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 4 children; when not spending time with them, he can usually be found wasting time playing video games or watching Serie A football.



■ **IGGY FERNANDEZ** has a rich history of working with Oracle Database in many capacities. He is the author of *Beginning Oracle Database 11g Administration* (Apress, 2009) and the editor of the *NoCOUG Journal*. He writes a regular column called “The SQL Corner” for the *NoCOUG Journal* and regularly speaks on SQL topics at Oracle conferences. He has a lot of opinions but is willing to change them when confronted with fresh facts. His favorite quote is “*A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines. Speak what you think now in hard words, and tomorrow speak what tomorrow thinks in hard words again, though it contradict everything you said today.*” (Ralph Waldo Emerson, *Self-Reliance and Other Essays*.)



■ **BERNARD LOPUZ** has been a senior technical support analyst at Oracle Corporation since 2001, and he is an Oracle Certified Professional (OCP). Before he became an Oracle DBA, he was a programmer developing Unisys Linc and Oracle applications, as well as interactive voice response (IVR) applications such as telephone banking voice-processing applications. Bernard was coauthor of the *Linux Recipes for Oracle DBAs* (Apress, 2008) and technical reviewer of two other books, namely, *Oracle RMAN Recipes* (Apress, 2007) and *Pro Oracle Database 11g Administration* (Apress, 2010). He has a bachelor's degree in computer engineering from the Mapúa Institute of Technology in Manila, Philippines. Bernard was born in Iligan,

Philippines, and now resides in Ottawa, Canada, with his wife, Leizle, and daughters, Juliet and Carol. Aside from tinkering with computers, Bernard is a soccer and basketball fanatic.

# Acknowledgments

I want to thank my fellow authors for all their hard work. This book is the result of many hours of your personal time and I appreciate every minute you spent to produce this excellent work. I'd also like to thank my family who graciously supported me during the long hours I had my nose stuck in my computer working. Your encouragement to take on this project was the main reason I decided to do so. Thanks for always believing in me.

Karen Morton

I'd like to dedicate this work to my family. My wife Jill and my kids Jordan, Jacob, Noah, and Lindsey have put up with me while I sat around with a far off look in my eyes (usually I wondering why my Mac wouldn't apply the right fonts to my examples). Seriously though, anyone who writes a book sacrifices a lot of time to do so. Writers undertake these projects for various reasons, but it is their choice. The people that care about them, though, also end up sacrificing a lot, through no fault of their own. So I am thankful for my family and the patience they have shown with me and for even occasionally pretending to be mildly interested in what I writing about.

Kerry Osborne  
*Enkitec*

blog: [kerryosborne.oracle-guy.com](http://kerryosborne.oracle-guy.com)

Thank you to the Oracle community in general and the OakTable network specifically for all the support and encourage over the years. Your examples motivated me to continue to learn, and the information you shared made it possible.

Robyn Sands

I dedicate this book to my lovely wife Nisha Riyaj.

Riyaj Shamsudeen

My portion of this book is dedicated to my wife Carla. She patiently tolerated the many late nights I spent in the home office, creating example SQL queries and writing text to explain them. Without her support I just couldn't do this.

I have spent a large portion of my DBA career working as a lone DBA, without a team of DBA peers to call on when needed. The online Oracle communities in their many forms have filled that void nicely. In particular, I would like to acknowledge those that participate in the Oracle-L mailing list at [www.freelists.org/list/oracle-l](http://www.freelists.org/list/oracle-l). Though this is now considered an old fashioned form of social media, the members of the Oracle-L community are quite knowledgeable and always willing to share their expertise. Much of what I have learned has been through participation in this forum.

Jared Still

# CHAPTER 1



## Core SQL

### Karen Morton

Whether you're relatively new to writing SQL or you've been writing it for years, learning to write "good" SQL is a process that requires a strong foundation knowledge of core syntax and concepts. This chapter provides a review of the core concepts of the SQL language and its capabilities along with descriptions of the common SQL commands with which you should already be familiar. For those of you who have worked with SQL previously and have a good grasp on the basics, this will be a brief refresher, and it will prepare you for the more detailed treatment of SQL we'll cover in the chapters ahead. If you're a new SQL user, you may want to read *Beginning Oracle SQL* first to make sure you're comfortable with the basics. Either way, Chapter 1 is intended to "level set" you with a whirlwind tour of the five core SQL statements and provide a quick review of the tool we'll be using to execute SQL, SQL\*Plus.

### The SQL Language

The SQL language was originally developed in the 1970s by IBM and called Structured English QUery Language, or SEQUEL. The language was based on the model for relational database management systems (RDBMS) developed by E.F. Codd in 1969. The acronym was later shortened to SQL due to a trademark dispute. In 1986, ANSI adopted SQL as a standard, and in 1987, ISO did so as well. A piece of not-so-common knowledge is that the official pronunciation of the language was declared to be "ess queue ell" by ANSI. Most people, including me, still use the *sequel* pronunciation just because it flows a bit easier linguistically.

The purpose of SQL is to simply provide an interface to the database, in our case, Oracle. Every SQL statement is a command, or instruction, to the database. It differs from other programming languages like C and Java in that it is intended to process data in sets, not individual rows. The language also doesn't require that you provide instructions on how to navigate to the data—that happens transparently under the covers. But, as you'll see in the chapters ahead, knowing about your data and how and where it is stored can be very important if you want to write efficient SQL in Oracle.

While there are minor differences in how vendors (like Oracle, IBM and Microsoft) implement the core functionality of SQL, the skills you learn in one database will transfer to another. You will be able to use basically the same SQL statements to query, insert, update, and delete data and create, alter, and drop objects regardless of the database vendor.

Although SQL is the standard for use with various RDBMS, it is not particularly relational in practice. I'll expand on this a bit later in the book; I would also recommend that you read C.J. Date's book entitled *SQL and Relational Theory* for a more detailed review. Keep in mind that the SQL language doesn't always follow the relational model precisely—it doesn't implement some elements of the relational model at all while implementing other elements improperly. The fact remains that

since SQL is based on this model you must not only understand SQL but you must understand the relational model as well in order to write SQL as correctly and efficiently as possible.

## Interfacing to the Database

Numerous ways have been developed over the years for transmitting SQL to a database and getting results back. The native interface to the Oracle database is the Oracle Call Interface (OCI). The OCI powers the queries that are sent by the Oracle kernel internally to the database. You use the OCI anytime you use one of Oracle's tools like SQL\*Plus or SQL Developer. Various other Oracle tools like SQL\*Loader, Data Pump, and Real Application Testing (RAT) use OCI as well as language specific interfaces such as Oracle JDBC-OCI, ODP.Net, Oracle Precompilers, Oracle ODBC, and the Oracle C++ Call Interface (OCCI) drivers.

When you use programming languages like COBOL or C, the statements you write are known as Embedded SQL statements and are preprocessed by a SQL preprocessor before the application program is compiled. Listing 1-1 shows an example of a SQL statement that could be used within a C/C++ block.

**Listing 1-1.** *Embedded SQL Statement Used Within C/C++ Block*

```
{
    int a;
    /* ... */
    EXEC SQL SELECT salary INTO :a
           FROM hr.employees
           WHERE employee_id = 108;
    /* ... */
    printf("The salary is %d\n", a);
    /* ... */
}
```

Other tools, like SQL\*Plus and SQL Developer, are interactive tools. You enter and execute commands, and the output is displayed back to you. Interactive tools don't require you to explicitly compile your code before running it. You simply enter the command you wish to execute. Listing 1-2 shows an example of using SQL\*Plus to execute a statement.

**Listing 1-2.** *Using SQL\*Plus to Execute a SQL Statement*

```
SQL> select salary
2  from hr.employees
3  where employee_id = 108;

-----
SALARY
-----
12000
```

In this book, we'll use SQL\*Plus for our example listings just for consistency, but keep in mind that whatever method or tool you use to enter and execute SQL statements, everything ultimately goes

through the OCI. The bottom line is that the tool you use doesn't matter, the native interface is the same for all.

## Review of SQL\*Plus

SQL\*Plus is a command line tool provided with every Oracle installation regardless of platform (Windows, Unix). It is used to enter and execute SQL commands and to display the resulting output in a text-only environment. The tool allows you to enter and edit commands, save and execute commands individually or via script files, and display the output in nicely formatted report form. To start SQL\*Plus you simply start `sqlplus` from your host's command prompt.

## Connect to a Database

There are multiple ways to connect to a database from SQL\*Plus. Before you can connect however, you will likely need to have entries for the databases you will need to connect to entered in the `$ORACLE_HOME/network/admin/tnsnames.ora` file. Two common ways are to supply your connection information when you start SQL\*Plus, as shown in Listing 1-3; the other is to use the SQL\*Plus connect command after SQL\*Plus starts, as shown in Listing 1-4.

### **Listing 1-3.** *Connecting to SQL\*Plus from the Windows Command Prompt*

```
E:\pro_oracle_sql>sqlplus hr@ora11r2

SQL*Plus: Release 11.2.0.1.0 - Production on Sun Jun 6 11:22:24 2010

Copyright (c) 1982, 2010, Oracle. All rights reserved.
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining and Real Appliation Testing
options

SQL>
```

To start SQL\*Plus without being prompted to login to a database, you can start SQL\*Plus with the `/nolog` option.

### **Listing 1-4.** *Connecting to SQL\*Plus and Logging into the Database from The SQL> Prompt*

```
E:\pro_oracle_sql>sqlplus /nolog

SQL*Plus: Release 11.2.0.1.0 - Production on Sun Jun 6 11:22:24 2010

Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect hr@ora11r2
Enter password:
Connected.
SQL>
```

## Configuring the SQL\*Plus environment

SQL\*Plus has numerous commands that allow you to customize the working environment and display options. Listing 1-5 shows the SQL\*Plus commands available after entering the SQL\*Plus help index command at the SQL> prompt.

**Listing 1-5.** *SQL\*Plus Command List*

```
SQL> help index
```

Enter Help [topic] for help.

@	COPY	PAUSE	SHUTDOWN
@@	DEFINE	PRINT	SPOOL
/	DEL	PROMPT	SQLPLUS
ACCEPT	DESCRIBE	QUIT	START
APPEND	DISCONNECT	RECOVER	STARTUP
ARCHIVE LOG	EDIT	REMARK	STORE
ATTRIBUTE	EXECUTE	REPFOOTER	TIMING
BREAK	EXIT	REPHEADER	TTITLE
BTITLE	GET	RESERVED WORDS (SQL)	UNDEFINE
CHANGE	HELP	RESERVED WORDS (PL/SQL)	VARIABLE
CLEAR	HOST	RUN	WHENEVER OSERROR
COLUMN	INPUT	SAVE	WHENEVER SQLERROR
COMPUTE	LIST	SET	XQUERY
CONNECT	PASSWORD	SHOW	

The set command is the primary command used for customizing your environment settings. Listing 1-6 shows the help text for the set command.

**Listing 1-6.** *SQL\*Plus SET Command*

```
SQL> help set
```

```
SET
---
```

Sets a system variable to alter the SQL\*Plus environment settings for your current session. For example, to:

- set the display width for data
- customize HTML formatting
- enable or disable printing of column headings
- set the number of lines per page

```
SET system_variable value
```



where `system_variable` and `value` represent one of the following clauses:

APPI[NFO] {OFF ON text}	NEWP[AGE] {1 n NONE}
ARRAY[SIZE] {15 n}	NULL text
AUTO[COMMIT] {OFF ON IMM[EDIATE] n}	NUMF[ORMAT] format
AUTOP[RINT] {OFF ON}	NUM[WIDTH] {10 n}
AUTORECOVERY {OFF ON}	PAGES[IZE] {14 n}
AUTOT[RACE] {OFF ON TRACE[ONLY]}	PAU[SE] {OFF ON text}
[EXP[LAIN]] [STAT[ISTICS]]	RECSEP {WR[APPED] EA[CH] OFF}
BLO[CKTERMINATOR] {. c ON OFF}	RECSEPCHAR {_ c}
CMDS[EP] {; c OFF ON}	SERVEROUT[PUT] {ON OFF}
COLSEP {_ text}	[SIZE {n   UNLIMITED}]
CON[CAT] {. c ON OFF}	[FOR[MAT] {WRA[PPED]
COPYC[OMMIT] {0 n}	WOR[D_WAPPED]
COPYTYPECHECK {ON OFF}	TRU[NCATED]}}
DEF[INE] {& c ON OFF}	SHIFT[INOUT] {VIS[IBLE]
DESCRIBE [DEPTH {1 n ALL}]	INV[ISIBLE]]}
[LINENUM {OFF ON}] [INDENT {OFF ON}]	SHOW[MODE] {OFF ON}
ECHO {OFF ON}	SQLBL[ANKLINES] {OFF ON}
EDITF[ILE] file_name[.ext]	SQLC[ASE] {MIX[ED]
EMB[EDDED] {OFF ON}	LO[WER]   UP[PER]}
ERRORL[OGGING] {ON OFF}	SQLCO[NTINUE] {>   text}
[TABLE [schema.]tablename]	SQLN[UMBER] {ON OFF}
[TRUNCATE] [IDENTIFIER identifier]	SQLPLUSCOMPAT[IBILITY]
	{x.y[z]}
ESC[APE] {\\ c OFF ON}	SQLPRE[FIX] {# c}
ESCCHAR {@ ? % \$ OFF}	SQLP[ROMPT] {SQL> text}
EXITC[OMMIT] {ON OFF}	SQLT[ERMINATOR] {; c ON OFF}
FEED[BACK] {6 n ON OFF}	SUF[FIX] {SQL text}
FLAGGER {OFF ENTRY INTERMED[IATE] FULL} TAB {ON OFF}	
FLU[SH] {ON OFF}	TERM[OUT] {ON OFF}
HEA[DING] {ON OFF}	TI[ME] {OFF ON}
HEADS[EP] {  c ON OFF}	TIMI[NG] {OFF ON}
INSTANCE [instance_path LOCAL]	TRIM[OUT] {ON OFF}
LIN[ESIZE] {80 n}	TRIMS[POOL] {OFF ON}
LOBOF[FSET] {1 n}	UND[ERLINE] {- c ON OFF}
LOGSOURCE [pathname]	VER[IFY] {ON OFF}
LONG {80 n}	WRA[P] {ON OFF}
LONGC[HUNKSIZE] {80 n}	XQUERY {BASEURI text
MARK[UP] HTML [OFF ON]	ORDERING{UNORDERED
[HEAD text] [BODY text] [TABLE text]	ORDERED DEFAULT}
[ENTMAP {ON OFF}]	NODE{BYVALUE BYREFERENCE
[SPOOL {OFF ON}]	DEFAULT}
[PRE[FORMAT] {OFF ON}]	CONTEXT text}
SQL>	

Given the number of commands available, you can easily customize your environment to best suit you. One thing to keep in mind is that the set commands aren't retained by SQL\*Plus when you exit/close the tool. Instead of typing in each of the set commands you want to apply each time you use SQL\*Plus, you can create a file named `login.sql`. There are actually two files which SQL\*Plus reads by default each time you start it. The first is `glogin.sql` and it can be found in the directory `$ORACLE_HOME/sqlplus/admin`. If this file is found, it is read and the statements it contains are executed. This will allow you to store the SQL\*Plus commands and SQL statements that customize your experience across SQL\*Plus sessions.

After reading `glogin.sql`, SQL\*Plus looks for the `login.sql` file. This file must exist in either the directory from which SQL\*Plus was started or in a directory included in the path the environment variable `SQLPATH` points to. Any commands in `login.sql` will take precedence over those in `glogin.sql`. Since 10g, Oracle reads both `glogin.sql` and `login.sql` each time you either start SQL\*Plus or execute the `connect` command from within SQL\*Plus. Prior to 10g, the `login.sql` script was only executed when SQL\*Plus started. The contents of a common `login.sql` file are shown in Listing 1-7.

**Listing 1-7.** *A Common login.sql File*

```
SET LINES 3000
Sets width of display line (default 80 characters)
SET PAGES 1000
Sets number of lines per page (default 14 lines)
SET TIMING ON
Sets display of elapsed time (default OFF)
SET NULL <null>
Sets display of nulls to show <null> (default empty)
SET SQLPROMPT '&_user@&_connect_identifier> '
Sets the prompt to show connected user and instance
```

Note the use of the variables `_user` and `_connect_identifier` in the `SET SQLPROMPT` command. These are two examples of predefined variables. You may use any of the following predefined variables in your `login.sql` file or in any other script file you may create:

- `_connect_identifier`
- `_date`
- `_editor` (This variable specifies the editor which is started when you use the `edit` command.)
- `_o_version`
- `_o_release`
- `_privilege`
- `_sqlplus_release`
- `_user`

## Executing Commands

There are two types of commands that can be executed within SQL\*Plus: SQL statements and SQL\*Plus commands. The SQL\*Plus commands shown in Listing 1-5 and 1-6 are specific to SQL\*Plus and can be used for customizing the environment and executing commands that are specific to SQL\*Plus, like

DESCRIBE and CONNECT. Executing a SQL\*Plus command requires only that you type the command at the prompt and hit Enter. The command is automatically executed. On the other hand, in order to execute SQL statements, you must use a special character to indicate you wish to execute the entered command. You may use either a semi-colon (;) or a forward slash (/). A semi-colon may be placed directly at the end of the typed command or on a following blank line. The forward slash must be placed on a blank line in order to be recognized. Listing 1-8 shows how these two execution characters are used.

**Listing 1-8.** *Execution Character Usage*

```
SQL>select empno, deptno from scott.emp where ename = 'SMITH' ;
      EMPNO      DEPTNO
-----
      7369        20
SQL>select empno, deptno from scott.emp where ename = 'SMITH'
2  ;
      EMPNO      DEPTNO
-----
      7369        20
SQL>select empno, deptno from scott.emp where ename = 'SMITH'
2  /
      EMPNO      DEPTNO
-----
      7369        20
SQL>select empno, deptno from scott.emp where ename = 'SMITH'
2
SQL>/
      EMPNO      DEPTNO
-----
      7369        20
SQL>select empno, deptno from scott.emp where ename = 'SMITH' /
2
SQL>l
1* select empno, deptno from scott.emp where ename = 'SMITH' /
SQL>/
select empno, deptno from scott.emp where ename = 'SMITH' /
*
```

ERROR at line 1:  
ORA-00936: missing expression

Notice the fifth example that puts the / at the end of the statement. The cursor moves to a new line instead of executing the command immediately. Then, if you press Enter again, the statement is entered into the SQL\*Plus buffer but not executed. In order to view the contents of the SQL\*Plus buffer, the list command is used (abbreviated to only l). If you then attempt to execute the statement in the buffer using /, which is how the / command is intended to be used, you get an error. That's because you had typed in the / on the end of the SQL statement line originally. The / is not a valid SQL command and thus causes an error when the statement attempts to execute.

Another way to execute commands is to place them in a file. You can produce these files with the text editor of your choice outside of SQL\*Plus or you may invoke an editor directly from SQL\*Plus using the EDIT command. The EDIT command will either open a named file or create a file if it doesn't exist. The file must be in the default directory or you must specify the full path. To set the editor to one of your choice, you simply set the predefined `_editor` variable using the following command: `define _editor='<full path>/myeditor.exe'`. Files with the extension of `.sql` will execute without having to include the extension and can be ran using either the `@` or `START` command. Listing 1-9 shows the use of both commands.

**Listing 1-9.** *Executing .sql Script Files*

```
SQL> @list_depts
      DEPTNO DNAME          LOC
-----
      10 ACCOUNTING      NEW YORK
      20 RESEARCH        DALLAS
      30 SALES            CHICAGO
      40 OPERATIONS      BOSTON

SQL>
SQL> start list_depts
      DEPTNO DNAME          LOC
-----
      10 ACCOUNTING      NEW YORK
      20 RESEARCH        DALLAS
      30 SALES            CHICAGO
      40 OPERATIONS      BOSTON

SQL>
SQL> 1* select * from scott.dept
SQL>
```

SQL\*Plus has many features and options—way too many to cover here. For what we'll need in this book, this brief overview should suffice. However, the Oracle documentation provides guides for SQL\*Plus usage and there are numerous books, including *Beginning Oracle SQL*, that go into more depth if you're interested.

## The Five Core SQL Statements

The SQL language contains many different statements. In your professional career you may end up using just a small percentage of what is available to you. But isn't that the case with almost any product you use? I once heard a statistic quoted stating that most people use 20 percent or less of the functionality available in the software products or programming languages they regularly use. I don't know if that's actually true or not, but in my experience, it seems fairly accurate. I have found the same basic SQL statement formats in use within most applications for almost 20 years. Very few people ever use everything SQL has to offer—and often improperly implement those they do use frequently. Obviously, we will not be able to cover all the statements and their options found in the SQL language. This book is intended to provide you deeper insight into the most commonly used statements and help you learn to apply them more effectively.

In this book, we will examine five of the most frequently used SQL statements. These statements are SELECT, INSERT, UPDATE, DELETE, and MERGE. Although we'll address each of these core statements in some fashion, the focus will be primarily on the SELECT statement. Developing a good command of these five statements will provide a strong foundation for your day-to-day work with the SQL language.

## The SELECT Statement

The SELECT statement is used to retrieve data from one or more tables or other database objects. You should already be familiar with the basics of the SELECT statement so instead of reviewing the statement from that beginner point of view, I wanted to review how a SELECT statement processes logically. You should have already learned the basic clauses that form a common SELECT statement, but in order to build the foundation mindset you'll need to write well-formed and efficient SQL consistently, you need to understand how SQL processes.

How a query statement is processed logically may be quite different from its actual physical processing. The Oracle cost-based optimizer (CBO) is responsible for generating the actual execution plan for a query and we will cover what the optimizer does, how it does it, and why in the chapters ahead. For now, note that the optimizer will determine how to access tables and in which order to process them, and how to join multiple tables and apply filters. The logical order of query processing occurs in a very specific order. However, the steps the optimizer chooses for the physical execution plan can end up actually processing the query in a very different order. Listing 1-10 shows a query stub containing the main clauses of a SELECT statement with step numbers assigned to each clause in the order it is logically processed.

**Listing 1-10.** *Logical Query Processing Order*

```

5      SELECT  <column list>
1      FROM      <source object list>
1.1    FROM      <left source object> <join type>
          JOIN <right source object> ON <on predicates>
2      WHERE     <where predicates>
3      GROUP BY  <group by expression(s)>
4      HAVING    <having predicates>
6      ORDER BY  <order by list>
```

You should notice right away that SQL differs from other programming languages in that the first written statement (the SELECT) is not the first line of code that is processed; the FROM clause is processed first. Note that I have shown two different FROM clauses in this listing. The one marked as 1.1 is provided to show the difference when ANSI syntax is used. It may be helpful to imagine that each step in the processing order creates a temporary dataset. As each step is processed, the dataset is manipulated until a final result is formulated. It is this final result set of data that the query returns to the caller.

In order to walk through each part of the SELECT statement in more detail, you'll use the query in Listing 1-11 that returns a result set containing a list of female customers that have placed more than four orders.

**Listing 1-11.** *Female Customers Who Have Placed More Than Four Orders*

```
SQL> select c.customer_id, count(o.order_id) as orders_ct
  2   from oe.customers c
  3   join oe.orders o
  4   on c.customer_id = o.customer_id
  5   where c.gender = 'F'
  6   group by c.customer_id
  7   having count(o.order_id) > 4
  8   order by orders_ct, c.customer_id
  9   /
```

CUSTOMER_ID	ORDERS_CT
146	5
147	5

## The FROM Clause

The FROM clause lists the source objects from which data is selected. This clause can contain tables, views, materialized views, partitions or subpartitions, or may specify a subquery that identifies objects. If multiple source objects are used, this logical processing phase also applies each join type and ON predicates (shown as step 1.1). You'll cover join types in more detail later but note that as joins are processed, they occur in the following order:

1. Cross join, also called a Cartesian product
2. Inner join
3. Outer join

In the example query in Listing 1-11, the FROM clause lists two tables: customers and orders. They are joined on the customer\_id column. So, when this information is processed, the initial dataset that will be produced by the FROM clause will include rows where the customer\_id matches in both tables. The result set will contain 105 rows at this point. To verify this is true, simply execute only the first four lines of the example query as shown in Listing 1-12.

**Listing 1-12.** *Partial Query Execution Through the FROM Clause Only*

```
SQL> select c.customer_id cust_id, o.order_id ord_id, c.gender
  2   from oe.customers c
  3   join oe.orders o
  4   on c.customer_id = o.customer_id;
```

CUST_ID	ORD_ID	G	CUST_ID	ORD_ID	G	CUST_ID	ORD_ID	G
147	2450	F	101	2430	M	109	2394	M
147	2425	F	101	2413	M	116	2453	M
147	2385	F	101	2447	M	116	2428	M
147	2366	F	101	2458	M	116	2369	M
147	2396	F	102	2431	M	116	2436	M