Contents at a Glance

About the Authors
Acknowledgments
Introduction

Chapter 1: Getting Started
Chapter 2: Locking and Issues
Chapter 3: Lock Types
Chapter 4: Concurrency and Multiversioning
Chapter 5: Transactions
Chapter 6: Redo and Undo
Chapter 7: Investigating Redo
Chapter 8: Investigating Undo

Index
Contents

About the Authors .............................................................................................................. ix
Acknowledgments ................................................................................................................ xi
Introduction .......................................................................................................................... xiii

■ Chapter 1: Getting Started ............................................................................................... 1
  Background ......................................................................................................................... 1
    Locking ............................................................................................................................. 1
    Concurrency Control ....................................................................................................... 2
    Multiversioning ............................................................................................................... 3
    Transactions .................................................................................................................... 4
    Redo and Undo ................................................................................................................. 4
  Summary ............................................................................................................................ 5

■ Chapter 2: Locking and Issues .......................................................................................... 7
  What Are Locks? ................................................................................................................ 7
  Lost Updates ..................................................................................................................... 10
  Pessimistic Locking ......................................................................................................... 10
  Optimistic Locking ........................................................................................................... 12
    Optimistic Locking Using a Version Column ............................................................... 13
    Optimistic Locking Using a Checksum ....................................................................... 16
  Optimistic or Pessimistic Locking? ............................................................................... 19
  Blocking .......................................................................................................................... 19
    Blocked Inserts ............................................................................................................ 20
    Blocked Merges, Updates, and Deletes ....................................................................... 22
## CONTENTS

- Deadlocks .................................................................................................................. 22
- Lock Escalation ........................................................................................................... 26
- Summary ....................................................................................................................... 27

### Chapter 3: Lock Types ............................................................................................. 29

- DML Locks .................................................................................................................. 29
  - TX (Transaction) Locks ............................................................................................. 29
  - TM (DML Enqueue) Locks ....................................................................................... 38
- DDL Locks .................................................................................................................. 40
- Latches ......................................................................................................................... 44
  - Latch “Spinning” ..................................................................................................... 45
  - Measuring the Cost of Latching a Shared Resource .............................................. 45
- Mutexes ....................................................................................................................... 53
- Manual Locking and User-Defined Locks ................................................................. 54
  - Manual Locking ...................................................................................................... 54
  - Creating Your Own Locks ....................................................................................... 54
- Summary ....................................................................................................................... 55

### Chapter 4: Concurrency and Multiversioning .......................................................... 57

- What Are Concurrency Controls? .............................................................................. 57
- Transaction Isolation Levels ....................................................................................... 58
  - READ UNCOMMITTED ............................................................................................ 59
  - READ COMMITTED ................................................................................................ 61
  - REPEATABLE READ ............................................................................................... 62
  - SERIALIZABLE ...................................................................................................... 65
  - READ ONLY .......................................................................................................... 67
- Implications of Multiversion Read Consistency ......................................................... 67
  - A Common Data Warehousing Technique That Fails .......................................... 68
  - An Explanation for Higher Than Expected I/O on Hot Tables .............................. 69
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Consistency</td>
<td>71</td>
</tr>
<tr>
<td>Consistent Reads and Current Reads</td>
<td>72</td>
</tr>
<tr>
<td>Seeing a Restart</td>
<td>74</td>
</tr>
<tr>
<td>Why Is a Restart Important to Us?</td>
<td>77</td>
</tr>
<tr>
<td>Summary</td>
<td>78</td>
</tr>
<tr>
<td>Chapter 5: Transactions</td>
<td>79</td>
</tr>
<tr>
<td>Transaction Control Statements</td>
<td>79</td>
</tr>
<tr>
<td>Atomicity</td>
<td>81</td>
</tr>
<tr>
<td>Statement-Level Atomicity</td>
<td>81</td>
</tr>
<tr>
<td>Procedure-Level Atomicity</td>
<td>83</td>
</tr>
<tr>
<td>Transaction-Level Atomicity</td>
<td>87</td>
</tr>
<tr>
<td>DDL and Atomicity</td>
<td>87</td>
</tr>
<tr>
<td>Durability</td>
<td>87</td>
</tr>
<tr>
<td>WRITE Extensions to COMMIT</td>
<td>87</td>
</tr>
<tr>
<td>COMMITS in a Nondistributed PL/SQL Block</td>
<td>89</td>
</tr>
<tr>
<td>Integrity Constraints and Transactions</td>
<td>90</td>
</tr>
<tr>
<td>IMMEDIATE Constraints</td>
<td>90</td>
</tr>
<tr>
<td>DEFERRABLE Constraints and Cascading Updates</td>
<td>91</td>
</tr>
<tr>
<td>Bad Transaction Habits</td>
<td>95</td>
</tr>
<tr>
<td>Committing in a Loop</td>
<td>95</td>
</tr>
<tr>
<td>Using Autocommit</td>
<td>101</td>
</tr>
<tr>
<td>Distributed Transactions</td>
<td>102</td>
</tr>
<tr>
<td>Autonomous Transactions</td>
<td>103</td>
</tr>
<tr>
<td>How Autonomous Transactions Work</td>
<td>104</td>
</tr>
<tr>
<td>When to Use Autonomous Transactions</td>
<td>105</td>
</tr>
<tr>
<td>Summary</td>
<td>108</td>
</tr>
</tbody>
</table>
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](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.

I’m **Darl Kuhn**, a DBA/developer working for Oracle. I also teach Oracle classes at Regis University in Denver, Colorado, and I am 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.
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.

Second, at Oracle, I work with the best and brightest people I have ever known, and they all have contributed in one way or another.

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. 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 like this one. And I’d like to thank Heidi, Lisa, and Brandi; without their support, I could not have successfully participated.

—Darl Kuhn
I’ve been asked many times, “What is the key to building highly concurrent and scalable database applications?” Invariably my response is “Begin with the basics, start with thoroughly understanding how Oracle manages transactions.”

When designing and creating database applications, understanding how the underlying database manages transactions will enable you to make intelligent architectural decisions that result in highly concurrent and scalable applications. Without knowledge of how the database handles transactions, you’ll invariably make poor design choices and end up with code that will never perform well. If you’re going to be building systems that use an Oracle database, it’s critical that you understand Oracle’s transaction management architecture.

Who This Book Is For

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 Oracle handles transaction management. As the title suggests, *Oracle Database Transactions and Locking Revealed* focuses on the core database topics of how transactions work, as well as locking. Related to those topics are Oracle’s use of redo and undo. I’ll explain what each of these are and why it is important for you to know about these features.

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 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.

Whether you want to type the code or not, all the source code for this book is available in the Source Code section of the Apress web site (www.apress.com). If you like to type 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
- Creating the BIG_TABLE table
- 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 set up whatever user you want to run the examples in this book. I picked the username 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

**Note** In Oracle 10g and above, you must install the demonstration subdirectories from the installation media. I have reproduced the necessary components of `demobld.sql` as well.

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:

```sql
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;
```

This finishes off the installation of the demonstration schema. 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.

Creating the Schema Without the Script

In the event you do not have access to `demobld.sql`, the following is sufficient to run the examples in this book:

```sql
CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
 ENAME VARCHAR2(10),
 JOB VARCHAR2(9),
 MGR NUMBER(4),
 HIREDATE DATE,
 SAL NUMBER(7, 2),
 COMM NUMBER(7, 2),
 DEPTNO NUMBER(2)
);
```
INSERT INTO EMP VALUES (7369, 'SMITH', 'CLERK', 7902, TO_DATE('17-DEC-1980', 'DD-MON-YYYY'), 800, NULL, 20);
INSERT INTO EMP VALUES (7499, 'ALLEN', 'SALESMAN', 7698, TO_DATE('20-FEB-1981', 'DD-MON-YYYY'), 1600, 300, 30);
INSERT INTO EMP VALUES (7521, 'WARD', 'SALESMAN', 7698, TO_DATE('22-FEB-1981', 'DD-MON-YYYY'), 1250, 500, 30);
INSERT INTO EMP VALUES (7566, 'JONES', 'MANAGER', 7839, TO_DATE('2-APR-1981', 'DD-MON-YYYY'), 2975, NULL, 20);
INSERT INTO EMP VALUES (7654, 'MARTIN', 'SALESMAN', 7698, TO_DATE('28-SEP-1981', 'DD-MON-YYYY'), 1250, 1400, 30);
INSERT INTO EMP VALUES (7698, 'BLAKE', 'MANAGER', 7839, TO_DATE('1-MAY-1981', 'DD-MON-YYYY'), 2850, NULL, 30);
INSERT INTO EMP VALUES (7782, 'CLARK', 'MANAGER', 7839, TO_DATE('9-JUN-1981', 'DD-MON-YYYY'), 2450, NULL, 10);
INSERT INTO EMP VALUES (7788, 'SCOTT', 'ANALYST', 7566, TO_DATE('09-DEC-1982', 'DD-MON-YYYY'), 3000, NULL, 20);
INSERT INTO EMP VALUES (7839, 'KING', 'PRESIDENT', NULL, TO_DATE('17-NOV-1981', 'DD-MON-YYYY'), 5000, NULL, 10);
INSERT INTO EMP VALUES (7844, 'TURNER', 'SALESMAN', 7698, TO_DATE('8-SEP-1981', 'DD-MON-YYYY'), 1500, 0, 30);
INSERT INTO EMP VALUES (7876, 'ADAMS', 'CLERK', 7788, TO_DATE('12-JAN-1983', 'DD-MON-YYYY'), 1100, NULL, 20);
INSERT INTO EMP VALUES (7900, 'JAMES', 'CLERK', 7698, TO_DATE('3-DEC-1981', 'DD-MON-YYYY'), 950, NULL, 30);
INSERT INTO EMP VALUES (7902, 'FORD', 'ANALYST', 7566, TO_DATE('3-DEC-1981', 'DD-MON-YYYY'), 3000, NULL, 20);
INSERT INTO EMP VALUES (7934, 'MILLER', 'CLERK', 7782, TO_DATE('23-JAN-1982', 'DD-MON-YYYY'), 1300, NULL, 10);

CREATE TABLE DEPT
(DEPTNO NUMBER(2),
 DNAME VARCHAR2(14),
 LOC VARCHAR2(13)
);

INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');

If you create the schema by executing the preceding commands, do remember to go back to the previous subsection and execute the commands to create the constraints.

Setting Your SQL*Plus 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 following SQL*Plus command must be issued:

SQL> set serveroutput on
If you are like me, typing this in each and every time would quickly get tiresome. Fortunately, SQL*Plus allows us to set up 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 as follows:

```
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.> '
```

An annotated version of this file is as follows:

- `define _editor=vi`: Set up the default editor SQL*Plus would use. You may set that to be your favorite text editor (not a word processor) such as Notepad or emacs.
- `set serveroutput on size unlimited`: Enable DBMS_OUTPUT to be on by default (hence you don’t have to type `set serveroutput on` every time). Also set the default buffer size to be as large as possible.
- `set trimspool on`: When spooling text, lines will be blank-trimmed and not fixed width. If this is set off (the default), spooled lines will be as wide as your linesize setting
- `set long 5000`: Sets the default number of bytes displayed when selecting LONG and CLOB columns.
- `set linesize 100`: Sets the width of the lines displayed by SQL*Plus to be 100 characters.
- `set pagesize 9999`: Sets the pagesize, which controls how frequently SQL*Plus prints out headings, to a big number (we get one set of headings per page).
- `column plan_plus_exp format a80`: Sets the default width of the explain plan output we receive with AUTOTRACE. a80 is generally wide enough to hold the full plan.

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, so I know who I am as well as where I am:

```
EODA@ORA12CR1>
```

### 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.

Note If you’re using a very old version of Oracle, you can manually create the PLAN_TABLE by executing the

$ORACLE_HOME/rdbms/admin/utlxplan.sql script.

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.

You can control the report by setting the AUTOTRACE system variable.

- SET AUTOTRACE OFF: No AUTOTRACE report is generated. This is the default.
- SET AUTOTRACE ON EXPLAIN: The AUTOTRACE report shows only the optimizer
  execution path.
- SET AUTOTRACE ON STATISTICS: The AUTOTRACE report shows only the SQL statement
  execution statistics.
- SET AUTOTRACE ON: The AUTOTRACE report includes both the optimizer execution path and
  the SQL statement execution statistics.
- SET AUTOTRACE TRACEONLY: Like SET AUTOTRACE ON, but suppresses the printing of the user’s
  query output, if any.
- SET AUTOTRACE TRACEONLY EXPLAIN: Like SET AUTOTRACE ON, but suppresses the printing of
  the user’s query output (if any), and also suppresses the execution statistics.

Setting Up StatsPack

StatsPack is designed to be installed when connected as SYS (CONNECT/AS SYSDBA) 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 the following three pieces of information before running the spcreate.sql script:

- 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

SQL*Plus: Release 12.1.0.1.0 Production on Fri May 23 15:45:05 2014
Copyright (c) 1982, 2013, Oracle. All rights reserved.
Connected to:  
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SYS@ORA12CR1> @spcreate

Choose the PERFSTAT user's password
-----------------------------------
Not specifying a password will result in the installation FAILING
Enter value for perfstat_password:
... <output omitted for brevity> ...
```

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.

---

**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.
- Creates a primary key constraint on the table.
- 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.

```sql
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:

```sql
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;
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. For example, a procedure such as the following would always print out every row in the `EMP` table where `ENAME` is not null:

```sql
create procedure p( ENAME in varchar2 )
as
begin
  for x in ( select * from emp where ename = ENAME ) loop
    dbms_output.put_line( x.empno );
  end loop;
end;
```

SQL sees `ename = ENAME` and compares the `ENAME` column to itself (of course). We could use `ename = P.ENAME`; that is, qualify the reference to the PL/SQL variable with the procedure name; but this is too easy to forget, leading to errors.

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.