



CHAPTER 1

Oracle Database 11g Architecture Options



Oracle Database 11g is a significant upgrade from prior releases of Oracle. New features give developers, database administrators, and end users greater control over the storage, processing, and retrieval of their data. In this chapter, you will see highlights of the Oracle Database 11g architecture. You will see detailed discussions of new features such as SQL replay, change management, and result caching in later chapters. The goal of this chapter is to present a high-level overview of the capabilities you can feature in your Oracle applications and provide an introduction to the chapters that describe them.

This book is divided into eight major sections.

In Part I, “Critical Database Concepts,” you will see an overview of Oracle Database 11g’s options, how to install the Oracle software, how to create or upgrade a database, and advice on planning your application implementation. These chapters establish the common vocabulary that both end users and developers can use to coherently and intelligently share concepts and ensure the success of any development effort. This introductory chapter and Chapter 4 are intended for both developers and end users of Oracle; Chapters 2 and 3 are intended for database administrators.

Part II, “SQL and SQL*Plus,” teaches the theory and techniques of relational database systems and applications, including SQL (Structured Query Language) and SQL*Plus. The section begins with relatively few assumptions about data-processing knowledge on the part of the reader and then advances, step by step, through some very deep issues and complex techniques. The method very consciously uses clear, conversational English, with unique and interesting examples, and strictly avoids the use of undefined terms or jargon. This section is aimed primarily at developers and end users who are new to Oracle or need a quick review of certain Oracle features. It moves step by step through the basic capabilities of SQL and Oracle’s interactive query facility, SQL*Plus. When you’ve completed this section, you should have a thorough understanding of all SQL keywords, functions, and operators. Within an Oracle database, you should be able to produce complex queries, create tables, and insert, update, and delete data.

Part III, “Beyond the Basics,” covers advanced options, including virtual private databases, Data Pump, replication, text indexing, external tables, change replay, and the use of the flashback options for developers and database administrators. Most of the features described in this section will not be directly used by end users, but the applications they use can be based on these features.

Part IV, “PL/SQL,” provides coverage of PL/SQL. The topics include a review of PL/SQL structures, plus triggers, stored procedures, and packages. Both standard and native dynamic PL/SQL is covered.

Part V, “Object-Relational Databases,” provides extensive coverage of object-oriented features such as abstract datatypes, methods, object views, object tables, nested tables, varying arrays, and large objects.

Part VI, “Java in Oracle,” provides coverage of the Java features in the Oracle database. This section includes an overview of Java syntax as well as chapters on JDBC and Java stored procedures.

Part VII, “Hitchhiker’s Guides,” provides an overview of the Real Application Cluster and grid architecture available in Oracle Database 11g as well as case studies in using Oracle’s tuning tools, the new features such as the client-side cache, an overview of database administration, and a high-level description of the use of XML in Oracle.

Part VIII, “Alphabetical Reference,” is a reference for the Oracle server—a book unto itself. Reading the introductory pages to this reference will make its use much more effective and understandable. This section contains references for most major Oracle commands, keywords, products, features, and functions, with extensive cross-referencing of topics. The reference is

intended for use by both developers and users of Oracle but assumes some familiarity with the products. To make the most productive use of any of the entries, it's worthwhile to read the introductory pages of the reference. These pages explain in greater detail what is and is not included and how to read the entries.

On the Downloads page at www.oraclepressbooks.com, you will find the table-creation statements and row insertions for all the tables used in this book. For anyone learning Oracle, having these tables available on your own Oracle ID, or on a practice ID, will make trying or expanding on the examples very easy.

Databases and Instances

An Oracle database is a collection of data in one or more files. The database contains physical and logical structures. In the course of developing an application, you create structures such as tables and indexes to store rows and speed their retrieval. You can create synonyms for the object names, view objects in different databases (across database links), and restrict access to the objects. You can even use *external tables* to access files outside the database as if the rows in the files were rows in tables. In this book, you will see how to create these objects and develop applications based on them.

An Oracle *instance* comprises a memory area called the *System Global Area (SGA)* and the background processes that interact between the SGA and the database files on disk. In an Oracle Real Application Cluster (RAC), more than one instance will use the same database (see Chapter 50). The instances generally are on separate servers connected by a high-speed interconnect.

Inside the Database

Within the Oracle database, the basic structure is a table. Oracle Database 11g supports many types of tables, including the following:

- **Relational tables** Using the Oracle-supplied datatypes (see “Datatypes” in the Alphabetical Reference), you can create tables to store the rows inserted and manipulated by your applications. Tables have column definitions, and you can add or drop columns as the application requirements change. Tables are created via the **create table** command.
- **Object-relational tables** To take advantage of features such as type inheritance, you can use Oracle’s object-relational capabilities. You can define your own datatypes and then use them as the basis for column definitions, object tables, nested tables, varying arrays, and more. See Part V of this book.
- **Index-organized tables** You can create a table that stores its data within an index structure, allowing the data to be sorted within the table. See Chapter 17.
- **External tables** Data stored in flat files may be treated as a table that users can query directly and join to other tables in queries. You can use external tables to access large volumes of data without ever loading them into your database. See Chapter 28. Note that Oracle also supports BFILE datatypes, a pointer to an external binary file. Before creating a BFILE or an external table, you must create a directory alias within Oracle (via the **create directory** command) pointing to the physical location of the file. See Chapter 40 for details on BFILES and other large object datatypes.

6 Part I: Critical Database Concepts

- **Partitioned tables** You can divide a table into multiple partitions, which allows you to separately manage each part of the table. You can add new partitions to a table, split existing partitions, and administer a partition apart from the other partitions of the table. Partitioning may simplify or improve the performance of maintenance activities and user queries. You can partition tables on ranges of values, on lists of values, on hashes of column values, or on combinations of those options. See Chapter 18.
- **Materialized views** A materialized view is a replica of data retrieved by a query. User queries may be redirected to the materialized views to avoid large tables during execution—the optimizer will rewrite the queries automatically. You can establish and manage refresh schedules to keep the data in the materialized views fresh enough for the business needs. See Chapter 26.
- **Temporary tables** You can use the **create global temporary table** command to create a table in which multiple users can insert rows. Each user sees only his or her rows in the table. See Chapter 14.
- **Clustered tables** If two tables are commonly queried together, you can physically store them together via a structure called a *cluster*. See Chapter 17.
- **Dropped tables** You can quickly recover dropped tables via the **flashback table to before drop** command. You can flash back multiple tables at once or flash back the entire database to a prior point in time. Oracle supports *flashback queries*, which return earlier versions of rows from an existing table.

To support access to tables, you can use views that perform joins and aggregations, limit the rows returned, or alter the columns displayed. Views may be read-only or updatable, and they can reference local or remote tables. Remote tables can be accessed via database links. You can use synonyms to mask the physical location of the tables. See Chapter 25 for details on database links, and Chapter 17 for details on views.

To tune the accesses to these tables, Oracle supports many types of indexes, including the following:

- **B*-tree indexes** A B*-tree index is the standard type of index available in Oracle, and it's very useful for selecting rows that meet an equivalence criteria or a range criteria. Indexes are created via the **create index** command.
- **Bitmap indexes** For columns that have few unique values, a bitmap index may be able to improve query performance. Bitmap indexes should only be used when the data is batch loaded (as in many data warehousing or reporting applications).
- **Reverse key indexes** If there are I/O contention issues during the inserts of sequential values, Oracle can dynamically reverse the indexed values prior to storing them.
- **Function-based indexes** Instead of indexing a column, such as Name, you can index a function-based column, such as **UPPER(Name)**. The function-based index gives the Oracle optimizer additional options when selecting an execution path.
- **Partitioned indexes** You can partition indexes to support partitioned tables or to simplify the index management. Index partitions can be local to table partitions or may globally apply to all rows in the table.

- **Text indexes** You can index text values to support enhanced searching capabilities, such as expanding word stems or searching for phrases. Text indexes are sets of tables and indexes maintained by Oracle to support complex text-searching requirements. Oracle Database 11g offers enhancements to text indexes that simplify their administration and maintenance.

See Chapters 17 and 46 for further details on the index types listed here (excluding text indexes). For text indexes, see Chapter 27.

Storing the Data

All of these logical structures in the database must be stored somewhere in the database. Oracle maintains a data dictionary (see Chapter 45) that records *metadata* about each object—the object owner, a definition, related privileges, and so on. For objects that require physical storage space of their own, Oracle will allocate space within a *tablespace*.

Tablespaces

A tablespace consists of one or more datafiles; a datafile can be a part of one and only one tablespace. Oracle Database 11g creates at least two tablespaces for each database—SYSTEM and SYSAUX—to support its internal management needs. You can use Oracle managed files (OMF) to simplify the creation and maintenance of datafiles.

You can create a special kind of tablespace, called a *bigfile* tablespace, that can be many thousands of terabytes in size. Along with OMF, the management of bigfiles makes tablespace management completely transparent to the DBA; the DBA can manage the tablespace as a unit without worrying about the size and structure of the underlying datafiles.

If a tablespace is designated as a *temporary* tablespace, the tablespace itself is permanent; only the segments saved in the tablespace are temporary. Oracle uses temporary tablespaces to support sorting operations such as index creations and join processing. Temporary segments should not be stored in the same tablespaces as permanent objects.

Tablespaces can be either *dictionary managed* or *locally managed*. In a dictionary-managed tablespace, space management is recorded in the data dictionary. In a locally managed tablespace (the default), Oracle maintains a bitmap in each datafile of the tablespace to track space availability. Only quotas are managed in the data dictionary, dramatically reducing the contention for data dictionary tables.

Automated Storage Management

Automatic storage management (ASM) automates the layout of datafiles and other operating system–level files used by the database, by distributing them across all available disks. When new disks are added to the ASM instance, the database files are automatically redistributed across all disks in the defined disk group for optimal performance. The multiplexing features of an ASM instance minimize the possibility of data loss and are generally more effective than a manual scheme that places critical files and backups on different physical drives. See Chapter 51.

Automatic Undo Management

To support your transactions, Oracle can dynamically create and manage *undo segments*, which help maintain prior images of the changed blocks and rows. Users who have previously queried the rows you are changing will still see the rows as they existed when their queries began. Automatic Undo Management (AUM) allows Oracle to manage the undo segments directly with no database administrator intervention required. The use of AUM also simplifies the use of

flashback queries. You can execute *flashback version queries* to see the different versions of a row as it changed during a specified time interval. See Chapters 29 and 30 for further details on the use of undo segments, flashback queries, and flashback version queries.

Dropped Data

The *recycle bin* concept introduced with Oracle Database 10g impacts the space requirements for your tablespaces and datafiles. The default behavior for the drop of a table is for the table to retain its space allocation; you can see its space usage via the RECYCLEBIN data dictionary view. If you create and drop a table twice, there will be two copies of the table in the recycle bin. Although this architecture greatly simplifies recoveries of accidentally dropped tables, it may considerably increase the space used in your database. Use the **purge** command to remove old entries from your recycle bin. See the Alphabetical Reference for the syntax of the **purge** command.

Guarding the Data

You can fully control the access to your data. You can grant other users privileges to perform specific functions (such as **select**, **insert**, and so on) on your objects. You can pass along the right to execute further grants. You can grant privileges to roles, which are then granted to users, grouping privileges into manageable sets.

Oracle supports a very detailed level of privileges; you can control which rows are accessible and, during auditing, which rows trigger audit events to be recorded. When you use the Virtual Private Database (VPD) option, users' queries of tables are always limited regardless of the method by which they access the tables.

You can enable column masking for sensitive data, and you can encrypt the data as it is stored on disk. See Chapter 20 for details on the implementation of VPD.

In addition to securing access to the data, you can audit activities in the database. Auditable events include privileged actions (such as creating users), changes to data structures, and access of specific rows and tables.

Programmatic Structures

Oracle supports a wide array of programmatic access methods. The SQL language, described in detail throughout this book, is key to any application development effort. Other access methods include the following:

- **PL/SQL** As described in Part IV of this book, PL/SQL is a critical component of most application implementations. You can use PL/SQL to create stored procedures and functions, and you can call your functions within queries. Procedures and functions can be collected into packages. You can also create triggers, telling the database what steps to take when different events occur within the database. Triggers may occur during database events (such as database startup), changes to structures (such as attempts to drop tables), or changes to rows. In each case, you will use PL/SQL to control the behavior of the database or application when the triggering event occurs.
- **Dynamic SQL** You can generate SQL at run time and pass it to procedures that execute it via dynamic SQL. See Chapter 36.
- **SQL*Plus** As shown throughout this book, SQL*Plus provides a simple interface to the Oracle database. SQL*Plus can support rudimentary reporting requirements, but it is

better known for its support of scripting. It provides a consistent interface for retrieving data from the data dictionary and creating database objects.

- **Java and JDBC** As shown in Part VI of this book, Oracle's support for Java and JDBC allows you to use Java in place of PL/SQL for many operations. You can even write Java-based stored procedures. Oracle's Java offerings have been expanded and enhanced with each new release.
- **XML** As described in Chapter 52, you can use Oracle's XML interfaces and XML types to support inserting and retrieving data via XML.
- **Object-oriented SQL and PL/SQL** You can use Oracle to create and access object-oriented structures, including user-defined datatypes, methods, large objects (LOBs), object tables, and nested tables. See Part V.
- **Data Pump** Data Pump Import and Data Pump Export, both introduced in Oracle Database 10g, greatly enhance the manageability and performance of the earlier Import and Export utilities. You can use Data Pump to quickly extract data and move it to different databases while altering the schema and changing the rows. See Chapter 24 for details on the use of Data Pump.
- **SQL*Loader** You can use SQL*Loader to quickly load flat files into Oracle tables. A single flat file can be loaded into multiple tables during the same load, and loads can be parallelized. See Chapter 23.
- **External programs and procedures** You can embed SQL within external programs, or you can create procedural libraries that are later linked to Oracle. See Chapter 35.
- **UTL_MAIL** A package introduced in Oracle Database 10g, UTL_MAIL allows a PL/SQL application developer to send e-mails without having to know how to use the underlying SMTP protocol stack.

Choosing Architectures and Options

Oracle provides a full array of tools for developing applications based on Oracle Database 11g. Many of the features introduced with Oracle Database 11g will be available to you regardless of the application architecture you select.

If you have previously implemented applications in earlier versions of Oracle, you should review your database to identify areas where new features will benefit your application. For example, if you have previously implemented materialized views, you may be able to take advantage of new features that expand the possibilities for incremental ("fast") refreshes of the materialized views. Oracle provides a set of procedures that help you manage your materialized view refresh schedule. For example, you can execute a procedure that will generate a description of your refresh possibilities and the configuration issues (if any) that prevent you from using the fastest options possible. You can use another Oracle-provided procedure to generate recommendations for tuning materialized view structures based on a provided set of sample queries.

Some of the new features may contain small changes that can have a dramatic impact on your application or your coding approach. For example, you can use the change replay features to capture commands executed on one database and replay them on another. Some of the significant

10 Part I: Critical Database Concepts

new features include “invisible” indexes, simplified table maintenance, and editioned objects. You should evaluate your previous architecture decisions in light of the new features available.

In the next several chapters, you will see how to install Oracle Database 11g and how to upgrade to Oracle Database 11g from prior releases. Following those chapters, you will see an overview of application planning, followed by many chapters on the use of SQL, PL/SQL, Java, object-oriented features, and XML to get the most out of your Oracle database. Your application architecture may change over time as the business process changes. During those changes you should be sure to review the latest features to determine how your application can best exploit them for functionality and performance.