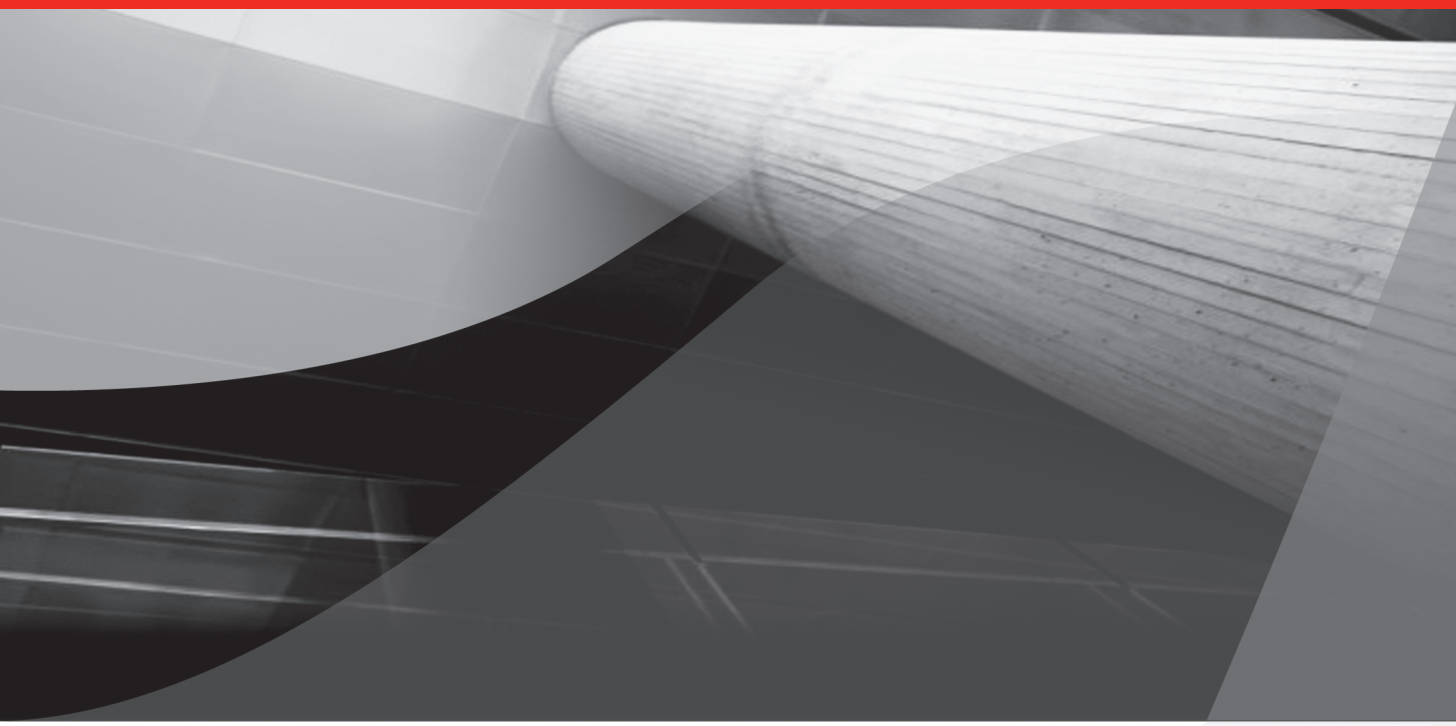
ORACLE



Oracle Database 11*g* & MySQL 5.6 Developer Handbook

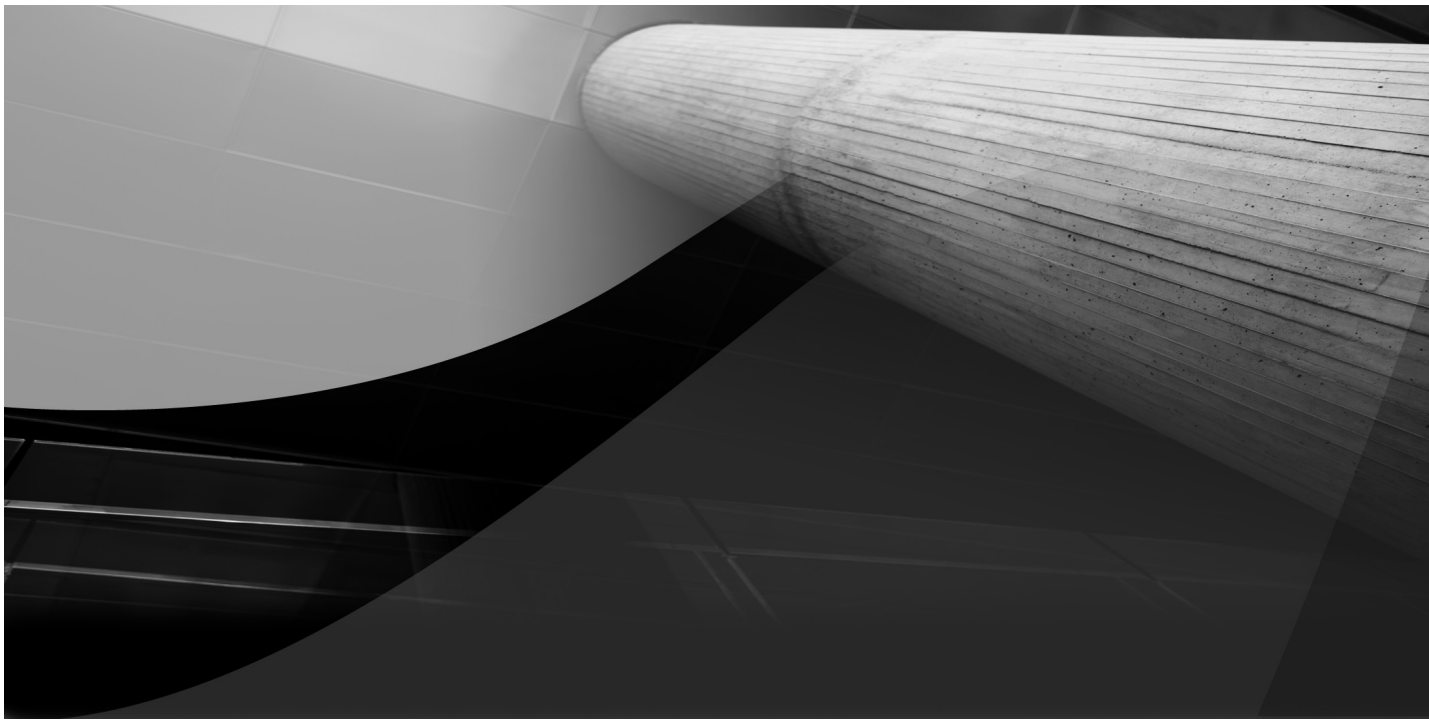
ORACLE DATABASE

MySQL

Design and Deploy Highly Portable Database Applications

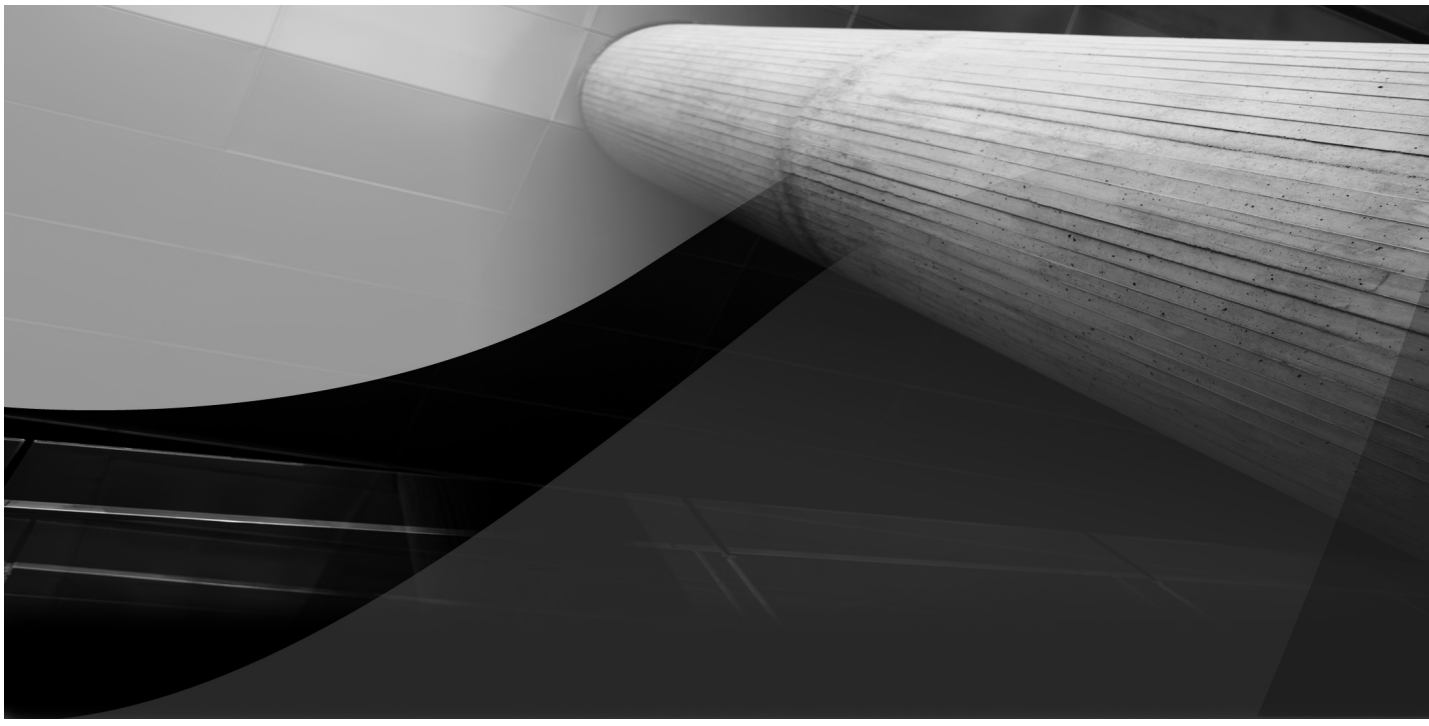
Michael McLaughlin
Oracle ACE

ORIGINAL • AUTHENTIC
Oracle Press
ONLY FROM McGRAW-HILL



PART I

Development Components



CHAPTER 1

Architecture

atabase architecture has two parts: One part qualifies how the database server works; the other part explains how the developer interacts with the database server. The latter part is *client-server computing*.

The client software provides the interface to the engine, like the steering wheel, brakes, and dashboard of a car. The engine is the server software. Server software includes an engine that stores and processes data, a transmission that governs transactions, and an enhanced odometer that logs what the system does to files. It also includes tires, body parts, seat cushions, and bumpers, which are support programs that manage the system's content integrity.

This chapter discusses client-server architecture for the Oracle and MySQL databases. It describes how they work together in general and explains how their client and server components work together. The chapter covers the following:

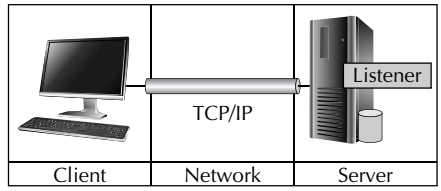
- General client-server computing model
- Oracle Database 11*g*
  - Client software: SQL*Plus
  - Oracle server software
- Oracle MySQL 5.6
  - Client software: MySQL Monitor
  - MySQL server software

These two databases have many similarities and some big differences. Any comparative comments are included here only once, where they make sense. Rather than repeat them in subsequent sections, you're asked to refer back to the prior comment. These comments should help you develop a solid understanding about how both databases work and the differences between the two.

General Client-Server Computing Model

Let's start with a brief discussion of how client and server software works before we examine how the Oracle or MySQL databases work. Client-server computing is a model whereby two computers share resources across a network. The sharing pattern involves a client that originates requests and a server that receives them. Like a mailbox in front of your home, a recipient in this model is a running process that actively listens for incoming requests and forwards them to the database. That process is called the *listener*.

The following illustration shows a view of how the two software components work in client-server computing. Oracle and MySQL implement these concepts differently, as you will see.



The client software component provides an interactive and batch interface that lets you and your programs interact with the database. Both MySQL and Oracle provide client interface software to manipulate database structures, and both query and modify data. These are command-line interfaces. Oracle implements its client as Oracle SQL*Plus and MySQL Server implements its client as MySQL Monitor.

Command-line tools are necessary in this computing model but can be discouraging to most beginning developers, because it takes time to learn how to use them. You can find instructions on using command-line tools in Chapter 2. Developers expect friendlier tools, known as *Computer-Aided Software Engineering (CASE)* tools, which generally provide a graphical user interface (GUI) to command-line–based client software.

Oracle SQL Developer and Oracle MySQL Workbench are the current vendor development tools. Many others exist, such as Quest's Toad. Although it was tempting to throw in a chapter on using these GUI CASE tools, the interface could change between writing and publishing this book. Because writing SQL and stored programs prepares you to develop database-centric applications, this book focuses on the command-line tools that you'll likely use in writing your code.

The Oracle and MySQL server-side software supports a relational database management system (RDBMS). You install local copies of both the client- and server-side software when you install either database product on any platform. As mentioned, the client-side software is a command-line console. It submits requests to the server-side engine and the engine returns result sets or acknowledgments of success or failure.

These requests are written in SQL (pronounced *sequel* by IBM engineers) statements. SQL stands for Structured Query Language and originally stood for Structured English Query Language (SEQUEL). SEQUEL as a name ran afoul of an existing British trademark and was shortened by IBM to SQL. Although SQL is often labeled as a nonprocedural programming language, that's *technically* inaccurate. Nonprocedural languages are typically event-driven languages, such as Java.

Instead, *SQL is a set-based declarative language*. Declarative programming languages let developers state what a program should do without qualifying how it will accomplish this. Declarative languages are much like an automatic transmission in a car. High-level instructions map to detailed activities hidden from the driver, such as accelerating and decelerating without bothering to change gears.

Like the throttle or gas pedal, SQL statements submit requests to database engines. The engine receives the request, determines the sequence of actions required to accomplish the task, and performs the task. Internally, the engines support *imperative languages*. The imperative languages change the state of variables and sets of variables for any assigned task.

SQL lets you interact with data, but it also lets you define and configure data structures without dealing with the specific mechanics of operation. The SQL statement engine processes all SQL statements. All means *all*, with no exceptions. SQL statements are *events* and fall into categories: they can be *Data Definition Language (DDL)*, *Data Manipulation Language (DML)*, *Data Control Language (DCL)*, and *Transaction Control Language (TCL)*.

Although there are many variations of how you use SQL commands, only 16 basic commands exist. The DDL commands let you create and modify structures in the database via `CREATE`, `ALTER`, `DROP`, `RENAME`, `TRUNCATE`, and `COMMENT` statements. DML commands let you query, add, modify, or remove data from structures via `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements. You also have a hybrid `MERGE` statement in the DML family of commands, which lets you insert or update rows based on logic you embed in the statement. When you transact across more than a single table, you use TCL commands `SAVEPOINT`, `ROLLBACK`, and `COMMIT`. Lastly the `GRANT` and `REVOKE` DCL commands let you give and retrieve privileges to act in the database.

Procedural Language Extensions

PL/SQL (Procedural Language/SQL) was defined by Oracle before database standards were provided for stored programs. They were added because SQL by itself couldn't do everything required of database-centric applications. MySQL 5.0 and newer versions implement SQL/PSM (SQL/Persistent Stored Module). This implementation owes its approach to the ISO/IEC 9075-4:2003 (more or less the ANSI SQL:2003) standard.

These are both procedural languages, which means they are imperative programming languages that use procedures or subroutines. The strength of both implementations is their recursive ability to allow SQL to call stored programs and stored programs to run SQL statements.

Essential elements of modern databases exist because of procedural language extensions, such as database triggers. Triggers are event-driven components defined by SQL and implemented by PL/SQL or SQL/PSM.

Set-based declarative languages such as SQL don't accomplish all that databases need to do. As a result of that failure, most commercial databases have implemented procedural programming language extensions called *modularized* imperative programming languages. *Modularized* means that they include subroutines and *procedural* comes from the label of subroutines as functions or procedures. Although they are subroutines, both require different semantics and interface paradigms.

Server-side software provides the management infrastructure for databases. It provides the SQL statement processor, which parses and runs statements. It also provides a subordinate engine for processing blocks of SQL, which may be a SQL statement, a set of SQL statements, or blocks of Oracle PL/SQL or MySQL SQL/PSM code.

As a developer, you must understand how the client software speaks with the server. It isn't adequate, in and of itself, to say *client-server computing is where two computers share resources across the network*. The client sends requests to the server and the server replies to the client through process communication. Process communication can occur through either an operating system pipe or a network socket. The former works only on a single machine, while the latter works on a single machine or set of machines connected by a network.

Beyond resolving their network addresses, the server software needs to start a process that *listens* for incoming requests. As mentioned, these types of processes are called *listeners*.

Although listener processes are part of the server software, their implementations often differ. Some of the differences are tied to how they implement user security.

A *schema* in an Oracle database is a discrete work area and equivalent to a database in the MySQL database except for one difference: the relationship between the user and work area. A *user* is synonymous with a schema in Oracle Database 11*g*, and there's a one-to-one map between user and schema. This makes an Oracle schema a private work area of a designated user.

MySQL users are separate from databases and the super user must grant privileges on a database to the user. A user has equivalent permissions to an Oracle user in their schema when the `root` user grants all privileges on that database to the user.

A data *repository* is a database or database instance. Database management systems create and maintain databases, just like Microsoft Word creates and maintains Word documents. The principal difference between a simple Word document and a database lies in the *data catalog*. The data catalog is a set of two-dimensional tables that define all data types, structures, and

Process Communication

An operating system pipe is an operator that lets the standard out (`stdout`) from one program feed into the standard in (`stdin`) of another program. You can use the pipe (`|`) operator to pipeline communication between two programs, or you can define a named pipe, which is a specialized file or FIFO (first in, first out) queue in Linux. That's why developers call them local sockets or Inter-Process Communication (IPC).

Network sockets are like named pipes. They take a request from a client program and forward it to the server program. Network sockets typically use the Transmission Control Protocol (TCP) to communicate between computers. Internet Protocol (IP) addresses support this type of connection between computers; that's why developers call network sockets TCP/IP communication.

Unlike named pipes, network sockets don't define an intermediary, because the network TCP/IP stack provides the communication plumbing between client and server machines. Network sockets require a program that's awake and ready to receive requests from client software. These awake and ready programs are listeners.

A listener program listens on an *ephemeral* (fancy word for *short-lived*) port. The word *ephemeral* in this context actually means a temporary address. Ports act like post office boxes to the server. When you start a program listening on a port, it acts like a FIFO queue, processing requests as they arrive.

processes. A data catalog is defined when you initially create a database, and it contains the default values that become building blocks for your database. As you use these building blocks, they write more data into the data catalog. The data catalog is a repository for *metadata*—data about data.

The building blocks also define basic data types and the rules for creating and running stored programs. You implement these as PL/SQL or SQL/PSM programs. You also have an option of

Database or Schema?

Is a discrete work area different when we call it a database or schema? The short answer is *yes*! They're not the same thing.

The long answer is more difficult, because *schema* is, since MySQL 5.0.2, an alias for *database* in Oracle MySQL 5.6. That might give you the impression that they're exactly the same, but they're not.

A MySQL database is a discrete work area. It is not directly and exclusively tied to a user as the owner of the work area. For example, a super user (`root`) in a MySQL database must grant privileges to a user to work in a database. Users by default have no designated or default work area. In fact, many users may be granted access to work in the same database.

On the other hand, an Oracle schema is a discrete work area owned by a single user account. When a super user (typically `SYS` or `SYSTEM`) creates users, it also creates the discrete work area owned by the user. Together, the user and work area create a schema in an Oracle database. However, any user may grant privileges to another user to work in their schema, and super users may do likewise.

creating anonymous or named block programs in PL/SQL but not SQL/PSM. You'll read more about PL/SQL in Chapter 13.

The next sections show you how the client, listener, and sever software work together and what they do in the two RDBMSs. Examples in Chapter 2 will show you how to interact through the SQL command-line interface of both databases. The following section describes the implementation of the database server architectures.

Oracle Database 11*g*

The Oracle Database 11*g* architecture has three components: the client, server, and listener software. Here we'll define, demonstrate, and explore the client and server software components of the Oracle database.

Next, we'll cover the Oracle Database 11*g* client, server, and listener software. These subsections describe the following:

- The general purpose of the command-line client software
- How to start, stop and configure the listener
- How to find and use the data dictionary

The data dictionary, or data catalog, keeps track of all structures and data. Ultimately, each data structure is stored by a *unique identifier (UID)* that is numeric. Structures also have a label or alias assigned to them, such as a table or column name. Using SQL, you can change that alias at any time, but changes to the table or column name won't alter the UID of those structures.

Client Software: SQL*Plus

SQL*Plus is the client software for Oracle. It was originally written as an interactive and batch development environment and SQL report writer in the 1980s. As a result, the client software also includes a set of well-designed formatting extensions to SQL. These extensions let you format, aggregate, and manipulate breaks for output from queries.

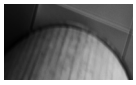
Converting from MySQL

Keep in mind the following when you're converting from a MySQL database to an Oracle database:

- Oracle SQL*Plus replaces the MySQL Monitor.
- The Oracle listener is a separately configurable server-side component, and a single listener can support multiple listening ports.
- Users map to a single schema, and they are synonymous for the purpose of definer rights models, covered in Chapter 3.
- All database users have permissions to connect via IPC or TCP/IP sockets because that's how they're configured in the default `listener.ora` and `sqlnet.ora` files.
- Grants of privileges to schema are effectively grants of permissions to users.

Advanced Friendly Interface
SQL*Plus was originally labeled as the Advanced Friendly Interface (AFI). That should debunk the rumor that the temporary `afiedt.buf` buffer file stood for *a file editor buffer*. Although you might not see it as *advanced* or *friendly* by today's standards, it certainly was back in the day.

Chapter 2 shows you how to use the SQL*Plus client software. Specifically, it shows you how to connect, write log files, save statements, edit statements from the SQL prompt, abort statements, call script files, and call and run statements.



NOTE
*You can find more information on SQL*Plus in the SQL*Plus User's Guide and Reference, Oracle documentation that is downloadable from Oracle's web site.*

Oracle 11*g* Server Software

This section covers the basics of how the Oracle Database 11*g* works. It also describes how users, scheme (schemas), and privileges work. These components are important because they support the design and development of database-centric applications.

Figure 1-1 shows a conceptual view of how the Oracle Database 11*g* client and server software interact. It also shows a breakaway view of the database session. Every session includes at a minimum one SQL*Plus environment, a SQL statement engine, and a PL/SQL engine that all interact with each other.

The inputs are straightforward in the figure because they are SQL statements that you enter or send to the SQL*Plus client software. Client software can process these statements interactively or through batch submission. You create a batch operation by grouping a series of SQL statements into a single file. This type of file is a *script* or *batch* file. There are also batch operations or programs.

Batch files should be rerunnable programs. Rerunnable programs don't raise exceptions or errors unless failures are critical. That means they conditionally process SQL statements that add, drop, or alter structure, and they insert, update, and delete data with statements that conform to the definitions of tables. The command-line syntax for running batch programs is shown in the "Batch Submission" section of Chapter 2.

At the bottom right in Figure 1-1, you can see external input and output files. These may be plain text or program units. When they're program units, they present alternative ways for submitting SQL statements to the database. They may use the Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC) libraries.

At the top of Figure 1-1, outputs are returned to the console. The output results can support interactive statements through session variables. Alternatively, output results may simply write a log file. At least, that's the case when the *SQL statement engine* returns outputs to the calling session. There's a *buffer* between the SQL*Plus environment and the PL/SQL engine. That buffer

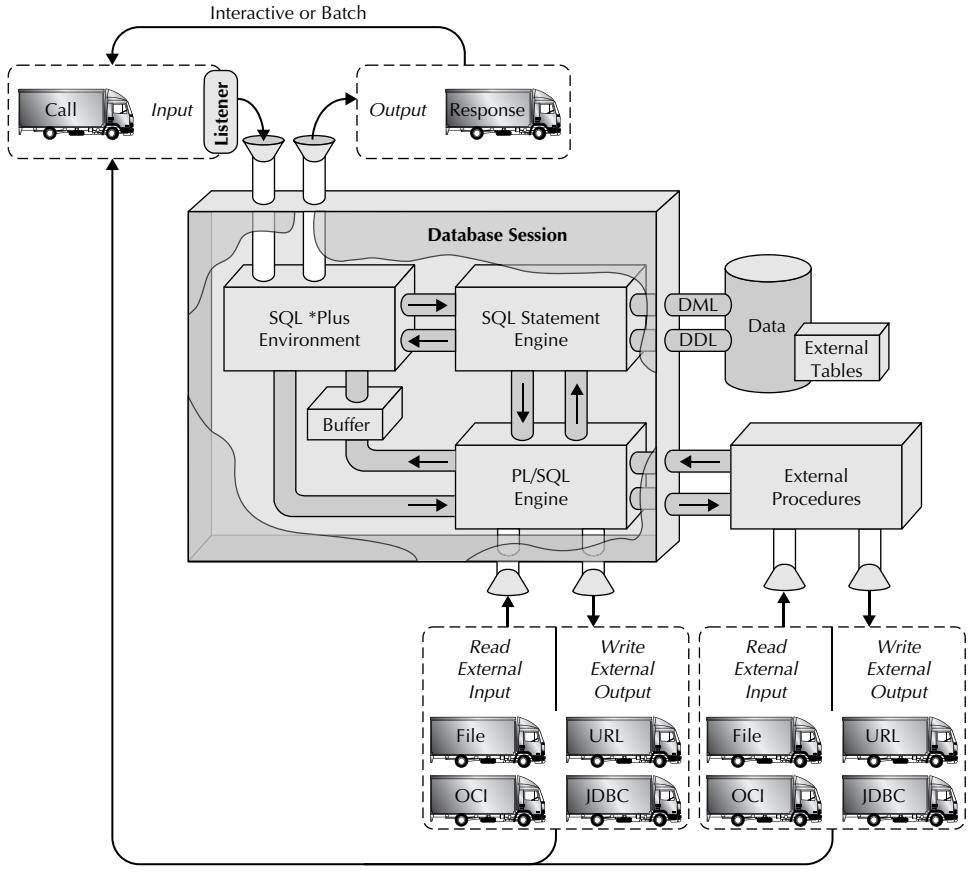


FIGURE 1-1. *Oracle Database 11g processing architecture*

is closed by default, which means comments echoed out by PL/SQL programs are blocked from display. You'll learn how to open the buffer and capture results in Chapter 2.

Reading and writing files appears straightforward, but it isn't. You have the option of reading external file data through (a) externally organized tables (covered in Chapter 6), (b) the Oracle's `UTL_FILE` package, (c) Java libraries inside the database, or (d) external procedures. Check the *Oracle Database 11g PL/SQL Packages and Types Reference* for the `UTL_FILE` package. You can find examples on embedded Java in Chapter 15 of *Oracle Database 11g PL/SQL Programming*.

The `UTL_FILE` package and Java libraries are the most secure approaches to reading and writing external files, because grants to these external locations are stored within the data catalog. External procedures call external C-callable libraries. These procedures pose a security risk to the database simply because anyone with access to the file system may access these libraries. These libraries can contain code that lets you establish a network socket for outgoing transmissions. As a result of that risk, many DBAs choose not to grant the `CREATE LIBRARY` privilege to protect against a security breach of confidential company and customer data.

Oracle Database Management System

The Oracle database management system comprises two major components: the data repository and a set of programs. The set of programs represent the library of code and APIs that implement Oracle Database 11*g*. The set of programs also lets you start a database instance. They allocate a shared memory realm where other programs process SQL statements. This shared memory realm is the active database instance.

In addition to implementing an RDBMS, the set of programs provides background processes that run the components that support the database. They're also programs that provide APIs for integration, backup and recovery resources, and configuration utilities and wizards. Some background processes run by default when you start an Oracle instance, while others require an administrator to configure and start them separately.

By way of an example, the optional *Archiver (ARCn)* process is critical to recovering databases. When an Oracle database is in archive mode, the Archiver mirrors writes to the redo log file. The Archiver writes those changes to the archive log files as the database switches from one redo log file to another. You can configure a number of other optional background processes. Refer to the *Oracle Database Administrator's Guide 11*g for more information on optional processes.

Together, the set of programs also provides a Multiversion Concurrency Control (MVCC), which ensures that one user won't inadvertently destroy another user's change. It does this by guaranteeing all changes to data are ACID-compliant (as described in the following discussion), which ensures the integrity of concurrent operations on data transactions.

ACID-compliant transactions meet four conditions: atomicity, consistency, isolation, and durability:

- *Atomic* means they complete or fail while undoing any partial changes.
- *Consistent* means they change from one state to another the same way whether or not the change is made through parallel or serial actions.
- *Isolated* means partial changes are never seen by other users or processes in the concurrent system.
- *Durable* means they are written to disk and made permanent when completed.

Object Relational Database Management System

Oracle is more than an ordinary RDBMS. The Oracle database is actually an object relational database management system (ORDBMS) because it supports collapsed objects inside the data repository. A *collapsed object* (also a flattened object) is a text string representing an object type and a call parameter list to the object, such as this:

```
some_object_type(call_param1, call_param2, call_param3)
```

The database holds the definitions of object types in the program source component of the database catalog. You can query its contents from the `USER_SOURCE` view (provided it's not *wrapped*, or obfuscated, to hide the source). Likewise, you can discover its constructors, functions, and procedures by describing it. Chapter 2 shows you how to describe tables. All you need to do is substitute the object name for the table name to make it work for object types.

Oracle manages ACID-compliant transactions by writing them to disk first, as redo log or redo and archive log files. Then it writes them to the database. This multiple step process with logs ensures that Oracle's buffer cache (part of the instance memory) isn't lost from any completed transaction. Log writes occur before acknowledgement of the transaction process occurs.

Oracle has five core background processes. Together, they ensure the successful operation and maintenance of an Oracle database instance. They are:

- **Process Monitor (PMON)** Cleans up the instance after failed processes by rolling back transactions, releasing database locks and resources, and restarting deceased processes.
- **System Monitor (SMON)** Manages system recovery by opening the database, rolling forward changes from the online redo log files, and rolling back uncommitted transactions. SMON also coalesces free space and deallocates temporary segments.
- **Database Writer (DBWR)** Writes data to files when any of the following occur: checkpoints are reached, dirty buffers reach their threshold or no buffers are free, timeouts occur, Real Application Cluster (RAC) ping requests are made, tablespaces are placed in `OFFLINE` or `READ ONLY` state, tables are dropped or truncated, and tablespaces begin backup processing.
- **Log Writer (LGWR)** Writes at user commits or at 3-second intervals, whichever comes first; when one-third of the log is full or 1 MB of redo instructions exists; and before the DBWR writes.
- **Checkpoint (CKPT)** Signals the DBWR at checkpoints and updates the file header information for database and control files at checkpoints. The checkpoint process synchronizes buffer cache information from the shared memory segment with database disks.

If you're a MySQL regular, you'll notice Oracle has included several additional processes for managing different responsibilities of the database. Although MySQL doesn't implement these processes directly, the responsibilities held by theses processes are delegated to the MySQL engines. MySQL 5.6 implements the InnoDB engine as the default engine, which provides services similar to those of an Oracle database.

Oracle Work Areas

The work area for an application is where you create and maintain tables, and insert, update, and delete data. The work area is a *database* in MySQL and a *schema* in Oracle Database 11*g*. An Oracle database instance often has many schemas. Although they can exist to support discrete applications, they often support designs known as *definer* and *invoker rights* models.

The definer rights model qualifies who defines or owns an object in the database, such as a table or view. The invoker rights model qualifies who may invoke a stored program. In a nasty twist with invoker rights programs, the invoker must have a local place to store any data inserted or updated and likewise a local place from which to delete data. Unfortunately, this generally results in duplicating tables in schemas granted invoker right privileges on stored programs.

Grants to the tables owned by the schema that defined the stored programs are impractical. If the schema needs to provide access to the programs and local tables, it uses the default definer rights model. The principal advantage of invoker rights is that they distribute the data in the tables.

The definer rights model in Oracle holds that the user/schema is always the definer; they cannot be disassociated. Although the super users `SYS` and `SYSTEM` hold a specialized privilege to `CREATE ANY TABLE`, users may create tables only in their own schema by default. Naturally, super users may grant super privileges to other users, but that is a bad idea unless those accounts are crafted to administer the database. Definer rights models are discussed further in Chapter 3.

The next section builds on your basic understanding of how a database management system works. It describes the Oracle Listener service and some basic configuration elements.

Listener

The Oracle Listener is a separate configurable component for any Oracle database. A single listener can support multiple databases.

The listener can be configured to listen for local traffic only, such as IPCs. It is generally configured to support IPCs and network sockets across TCP/IP. The default configuration of the listener traditionally mixes the IPC traffic and TCP/IP traffic through the same net service name. This type of configuration causes external procedures to fail. You must create distinct listeners when you implement external procedures to avoid failure of the procedures at runtime. The instructions to modify the listener file for external procedures can be found in Chapter 13 of *Oracle Database 11g PL/SQL Programming*. You can also find them in the *Oracle Database Advanced Application Developer's Guide, 11g Release*.

You typically configure two files, `listener.ora` and `tnsnames.ora`, to change how the Oracle Listener works. Sometimes you need to modify the `sqlnet.ora` file, too. These files can be found in the `network/admin` subdirectory of the Oracle home.

The `listener.ora` file contains the configuration instructions for starting and running the Oracle Listener process. The `tnsnames.ora` file contains the mapping between net service names, such as `orcl` or `xe`, and fully qualified Transparent Network Substrate (TNS) addresses. Aliases simplify the parameters a user needs to provide to connect the client to the server. The `sqlnet.ora` file provides configuration instructions for tracing connections, network domain limitations, and encryption.

The default `listener.ora` file for Oracle Database 11*g* should look more or less like this:

```
-- This is an example of a default listener.ora file.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = hostname)(PORT = port_number))
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = oracle_product_home_directory)
      (PROGRAM = extproc)
    )
  )
```

The `listener.ora` file is taken from the standard installation of a sample Oracle Database 11*g* installation. The default name of the listener is `LISTENER` and it holds references to the `hostname` and listening port. The `SID_LIST_{listener_name}` uses `LISTENER`. For every listener, there is an `SID_LIST_{listener_name}` because this list holds the Oracle Server home directory and any libraries used by external programs. Only the actual hostname, port number, and Oracle product home directory have been removed and replaced with generic comments. Notice that both an IPC and a TCP address exist in the `DESCRIPTION` section of the listener. This type of configuration file lets a single listener handle both protocols. It's convenient and it's the default file with the sample databases.

Oracle Listeners let users connect through either a local IPC or TCP/IP socket to a database instance. They do this by referencing a TNS alias for the listener. TNS implements the session, presentation, and application layers of the Open Systems Interconnection (OSI) model. Moreover, it maps a net service name to a physical address. Definitions for how this map works are found in the `listener.ora` file. The settings are configurable by the database administrator (DBA).

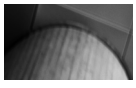
A default `tnsnames.ora` file for the same release should look similar to this:

```
-- This is an example of a default tnsnames.ora file.

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = hostname)(PORT = port_number))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )

EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
    (CONNECT_DATA =
      (SID = PLSExtProc)
      (PRESENTATION = RO)
    )
  )
```

Look at the entry in the `tnsnames.ora` file and you can see that `ORCL` is the net service name for a connection, which is an alias for the full connect string. Because an Oracle Listener can connect to multiple databases, a `SERVICE_NAME = ORCL` value is also included. The `SERVICE_NAME` is an alias to an instance or several instances when you work with clustered databases and is separate from the net service name. You should also know that the NETCA (Oracle Net Services Configuration) utility provides a GUI interface tool and wizard for most network configuration steps.



TIP
In Linux or UNIX, the net service name is case-sensitive.

You could change the net service name from ORCL to a name more appropriate for your system (RUMBA in the example below), stop and start the listener, and then connect using this:

```
sqlplus some_user@RUMBA
```

Here's the command-line syntax to start the listener:

```
lsnrctl start
```

Getting the listener's status is straightforward: replace the word start with status, like this:

```
lsnrctl status
```

You stop the listener with this:

```
lsnrctl stop
```

The lsnrctl command works at the command line in Windows or Linux. The listener is also a service in Microsoft Windows. You can start or stop the listener at the command line or through the service, because the service calls the same command.

If you can't reach the Oracle Listener, you can use an Oracle utility to check the network. The tnsping utility is similar to the ping utility. Assuming the preceding tnsnames.ora file and net service name ORCL, you can check whether the listener is reachable and available using this syntax:

```
tnsping orcl
```

You would see the following type of reply:

```
TNS Ping Utility for 64-bit Windows: Version 11.1.0.7.0 - Production on ...
Copyright (c) 1997, 2008, Oracle.  All rights reserved.
Used parameter files:
C:\app\McLaughlin-7x64\product\11.1.0\db_1\network\admin\sqlnet.ora
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST =
McLaughlin7x64)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_
NAME = orcl)))
OK (10 msec)
```

Potential Laptop Configuration Issue

A failure can occur when you install Oracle on any computer without a static IP address. It shows itself when you can't connect through a network socket but can connect through IPC. This problem is most likely caused by Oracle substituting the IP address at the time of installation for a hostname value in both the listener.ora and tnsnames.ora files.

Make sure that the hostname matches your machine's hostname, not the assigned IP address. You can do this by checking the hosts file, found in the C:\WINDOWS\System32\drivers\etc directory on Windows and in the /etc directory on Linux or UNIX. You may need to disable Microsoft's User Account Controls (UAC) to access the hosts file on Windows Vista and Windows 7.

The Oracle Listener is a convenient architecture, because it provides a number of configuration options. However, if your database is running when the listener is down, or vice versa, you won't be able to connect to the database through the network layer and you'll have two components to check and potentially fix.

Oracle's security model allows connections from all authorized users. It doesn't check their network origin unless you configure advanced networking options. Chapter 6 shows steps that let you configure network access rights in the `sqlnet.ora` file. That's because the Oracle Listener doesn't disambiguate `localhost` and network access across network sockets. This differs from the MySQL security model. By default, Oracle supports IPC communication when you opt not to include the net service name in your call to SQL*Plus, which is also known as a connect string. A connect string is a string used to identify and connect to a remote database. Connect strings are defined in a local `tnsnames.ora` file, Oracle Names Server or OID directory.

The biggest difference between how Listeners work is their role in the server software. Oracle Database 11*g* implements a single listener process that can listen for multiple database instances, whereas Oracle MySQL 5.6 (and its predecessors) has a listener process for each MySQL installation. The former scales better but the latter is simpler to administer for novices.

Oracle Data Dictionary

The Oracle data dictionary is stored in the `SYS` schema. It is a complex design, and you should not change data there without the explicit guidance and direction of Oracle Support. The dictionary houses the structure, definition, declaration, and access control attributes for all the objects in the database.

Names of data objects (tables) in the data dictionary end with a dollar (`$`) sign. Oracle does not publish a way to map these tables, and it does not provide an Entity Relation Diagram (ERD) or map of the data dictionary. Your sole interface to these objects should be SQL commands.

Oracle does provide administrative views that let you examine objects you've created and built-in objects in the database. These views start with `ALL_`, `DBA_`, `USER_`, and denote access levels to the internally defined objects. You can find a comprehensive list of these administrative views in the Oracle Database Reference 11*g* online. Many chapters show you how to use these administrative views to solve particular problems.

Oracle MySQL 5.6

Like Oracle Database 11*g*, the Oracle MySQL 5.6 architecture has two direct components: client and server software. Notably different from Oracle Database, MySQL embeds the listener software inside the server component. This means when you start the MySQL daemon, `mysqld` process, you start the database server and listener.

The following sections define, demonstrate, and explore the client and server software components of the MySQL database. These sections cover the Oracle MySQL 5.6 client, server, and listener software. They describe the general purpose of the command-line client software; how to start, stop, and configure the listener; and how to find and use the data dictionary.

Client Software: The MySQL Monitor

The *MySQL Monitor* is the command-line SQL interface for the MySQL 5.6 database. MySQL Monitor processes SQL statements, which it does well.

Converting from Oracle

Keep in mind the following when you're converting from an Oracle database to a MySQL database:

- Oracle MySQL Monitor replaces Oracle SQL*Plus and doesn't offer any formatting features or extended SQL aggregation behaviors.
- The Oracle MySQL listener is part of the database server. You must have a unique listener for each installation of the MySQL database. All you can configure is the listening port.
- Oracle MySQL users are distinct from any schema or database.
- MySQL users connect via TCP/IP sockets when defined by a domain, a partial domain, or as anywhere (`%`), unless you shut down network connections by configuring the `my.ini` file.
- Grants of privileges are made to users and never to databases.

There are two big differences between the MySQL Monitor and Oracle SQL*Plus environments. One is that the error messages are much less meaningful and in many cases not actionable. The second difference is that MySQL Monitor processes SQL and SQL/PSM call statements only. SQL report writing is left to other tools that you can purchase or implement.

Errors are less meaningful in MySQL, because MySQL provides two levels of error handling: error and warning messages. You generally need to check the warning messages in MySQL before you start solving problems. Chapter 2 shows you how to use the MySQL Monitor.

MySQL Server Software

Like the server section for the Oracle database, this section covers the Oracle MySQL database. You'll learn about the organization of users and databases, as well as the security privileges that link them together.

The `root` user is the super user in a MySQL Database. Likewise, `SYS` is the super user in Oracle and equivalent to the `root` user in a MySQL database.

Note that a database in MySQL is a *logical* repository, not a *physical* one. This means that you create and maintain things in a logical work area, or database. Database engines manage the physical storage of data.

Figure 1-2 shows a conceptual view of how the MySQL client and server software interact. It also shows a breakaway view of a session, which includes how the MySQL Monitor, SQL statement engine, and SQL/PSM engine interact with each other. Note that MySQL doesn't include the equivalent of Oracle Database 11*g* external procedures.

Reading and writing files is performed by SQL statements processed in the MySQL Monitor. This differs from Oracle Database 11*g*, where they're managed by the PL/SQL engine. You can input or output files, and you can choose how the outputted files are formatted. Beginning with Oracle MySQL 5.6, you can natively read and write XML files.

SQL/PSM stored programs can return data to the console when you select the information inside stored procedures. The data selected can be as simple as debugging comments or as complex as large return sets from complex queries, but only stored procedures can return the data to the

console. Functions cannot return data because they run as contained units. All their internal behaviors are encapsulated, and they can't echo content to the current session. Functions return only values defined by their return types. Chapters 13 and 14 show how you work with functions and procedures.

MySQL Database Management System

The MySQL database management system has three major components. Two are similar to the Oracle database's data repository and its set of programs, but they're not exactly the same.

The MySQL data repository isn't a single entity as in Oracle. It is a series of repositories organized by the logical database that owns the table definitions. Likewise, the set of programs is more complex. Some programs exist at the MySQL level and key transactional programs exist at the engine level. Moreover, these sets of programs mimic the set of programs in an Oracle database.

Together they maintain Multiversion Concurrency Control (MVCC) in most cases. MVCC prevents one user from inadvertently destroying another user's work. These sets of programs fail to maintain MVCC rules when some of the data is stored and managed by one processing engine

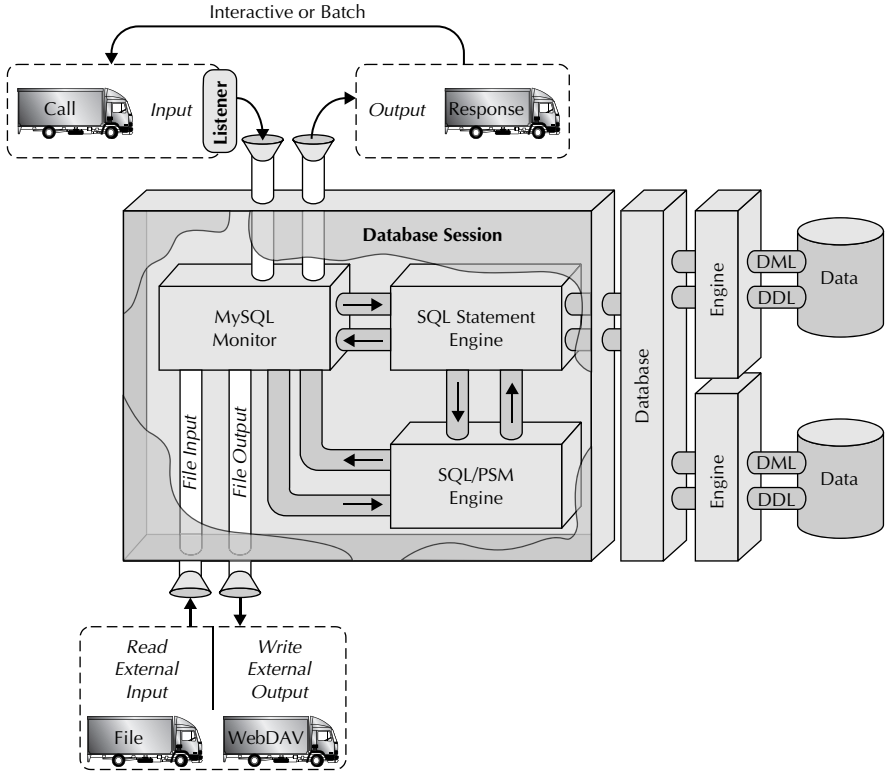


FIGURE 1-2. *MySQL database processing architecture*

and the rest is managed by another processing engine. Unfortunately, only the default InnoDB engine supports the X/Open XA distributed transaction model.

Distributed transactions require agreement between all parties before writing data permanently. One party may write data with the mistaken belief that the other parties' data is in agreement. The result is that the transaction state is incomplete as a whole. This possibility breaks the guarantee of ACID-compliant transactions in a distributed transaction. The behavior causes data inconsistencies between data stored in two or more repositories when they're managed by different MySQL engines.

Database users with the create table privilege may create tables in one database that use different engines. This introduces a major design and management issue, however, because tables defined in an engine are stored in that engine's repository. There is no way to guarantee ACID-compliant transactions across different repositories, because the X/Open XA protocol is deployed only by one engine, and both engines are needed to support the XA protocol. At present, the best thing to do for transactional data is store the tables exclusively in the InnoDB engine.

For example, suppose a table that contains rows of data for orders uses one engine, while another table containing binary images of scanned shipping labels uses another. It's presently impossible to update both tables with a guarantee that both will be changed. That's because this involves a distributed transaction, and agreement to write may result in one completed update without the other.

You should implement transactional data only in a single engine. That engine should support the full spectrum of transactional mechanics.

Engines

As mentioned, Oracle MySQL supports multiple engines. Each engine has distinct features but *only* InnoDB enforces referential integrity and the full X/Open XA protocol for distributed transactions.

InnoDB is the *default* engine for Oracle MySQL 5.6 for all but the `mysql` and `information_schema` databases. These two databases store and manage data using the MyISAM database engine. MyISAM was the default engine for earlier versions of the MySQL database. You can find the available engines in any release by typing the following from the command line:

```
help engines
```

Engines own the processes governing the integrity of data in their repositories. They also determine the physical structure of data and provide APIs that map those physical data structures to the logical structures of the data dictionary or catalog.

InnoDB maintains logs much like Oracle Database 11*g*. It uses those logs to ensure the integrity of transactions and to limit the cost of committing transactions. The engine does this by logging transactions rather than flushing the buffer pool after every commit. The InnoDB log management shifts random I/O into sequential I/O and provides effective transaction management, including *savepoint* and *rollback* behaviors. Chapter 4 qualifies the definition and use of savepoints and rollbacks in transaction management.

Chapter 6 shows you how to define tables that use the default or an override engine. Oracle MySQL Database 5.6 uses the following engines:

- **InnoDB** A transaction safe or ACID-compliant database engine
- **MyISAM** A non-transaction safe database engine, and previous default engine for earlier releases of the MySQL database
- **Memory** An in-memory database engine that stores all data in physical or virtual memory

- **Merge** A non-transaction database engine that lets you group identical MyISAM tables as one entity; key infrastructure element of many Very Large Databases (VLDB)
- **Archive** An engine that lets you archive seldom-used data
- **Federated** An engine that allows you to link several MySQL servers into one logical database from many physical servers; often used when servicing data marts
- **CSV** A non-transactional database engine that lets you store data in a comma-separated values (CSV) format, which makes interchanges easier when using other CSV-enabled tools
- **Blackhole** A non-transactional database that accepts but doesn't store data; used as an engine for external replication of the database
- **Example** A stub provided to developers to help them understand how to link engines into MySQL databases

Like Oracle, the InnoDB engine transforms MySQL 5.6 into a transaction database. InnoDB provides ACID-compliant transactions, row-level locking, Oracle-style nonlocking reads, and referential integrity through foreign keys. Changing the MySQL 5.6 default InnoDB engine means you switch from a transactional engine to a non-transactional engine.

Listener

The listener is started by the MySQL daemon (`mysqld`) when that process starts. The MySQL listener is a subordinate process of the server process, and it runs on the `[mysqld]` port number you specify in the `mysql.ini` configuration file. By default, the port number for the listener is 3306.

You can verify whether or not it's running by checking the Windows Services (`services.msc`), or you can check at the MS-DOS command line with the following syntax:

```
netstat -a | findstr /C:LISTENING | findstr /C:3306
```

It should report something similar to this:

```
  TCP    0.0.0.0:3306           McLaughlin7x64:0       LISTENING
```

The port number will change after you modify the `my.ini` file and reboot the MySQL service. Any changes made in the `my.ini` file are visible only after you restart the service (Windows) or process (Linux).

MySQL's security model does distinguish users according to their point of origin. Users defined within MySQL have a three-part authentication token: user name, password, and host. Users have three possible host values:

- **`localhost`** Can connect only from the same machine as the MySQL server
- **IP, subdomain, or domain** Can connect only from an IP or partially qualified subdomain or domain address
- **`%`** Can connect from anywhere; basically a *wildcard* host operator across the network

MySQL databases generally let you connect through a TCP/IP socket, but you can define the `mysql.ini` file to allow IPC communication. You disable TCP communication when you enable

IPC communication in MySQL. As a result, you usually use IPC communication only while performing system maintenance.

MySQL Data Dictionary

The MySQL data dictionary is stored in the `mysql` database. A view-only copy is also stored in the `information_schema` database. It is a fairly straightforward set of tables. Although you can make changes to the MySQL data dictionary, you should be very careful about what you do. If you're using the Community Edition, you're responsible for any changes you make that lead to catastrophic failures. If you're using the Enterprise Edition of MySQL, you should not change data without the explicit guidance and direction of Oracle Support.

All objects in the `mysql` database are tables. The equivalent objects in the `information_schema` database are non-updateable views. MySQL doesn't provide administrative views as does Oracle Database 11*g*.

Summary

There are many similarities and notable differences between the Oracle Database 11*g* and Oracle MySQL 5.5 databases. Primarily, Oracle databases map each user account to a given schema of the same name, while MySQL databases grant permissions to user accounts to access independent databases. MySQL databases can store data in one or more repositories, which may not be transaction safe.

Transactions across repositories in MySQL databases aren't ACID-compliant unless both engines support the X/Open XA distributed transaction protocol. At this time, only InnoDB is transaction safe and implements X/Open XA transactions. On the other hand, Oracle Database 11*g* stores data in a single transaction-safe repository.

The two-tier client-server model works in both databases. The features of client-side SQL*Plus and MySQL Monitor software are different from the perspective of available features. Servers differ in how they enforce ACID-compliance. MySQL's ability to log and maintain files differs by the engine you use. InnoDB is the default engine for MySQL 5.5, and it runs very much like Oracle by maintaining sequential log files.

Mastery Check

The mastery check is a series of true or false and multiple choice questions that let you confirm how well you understand the material in the chapter. You may check the Appendix for answers to these questions.

1. **True** ☐ **False** ☐ Databases implement a two-tier client-server model.
2. **True** ☐ **False** ☐ Client software provides only an interactive method for running SQL statements.
3. **True** ☐ **False** ☐ SQL statements can be grouped into categories.
4. **True** ☐ **False** ☐ You must interact with all databases across a network socket.
5. **True** ☐ **False** ☐ Distributed transactions occur by default in MySQL when you store data in two or more tables using different engines.
6. **True** ☐ **False** ☐ You can connect to MySQL database only through a network socket.

7. **True** ☐ **False** ☐ You can connect to an Oracle database only through a network socket.
8. **True** ☐ **False** ☐ The MVCC model for Oracle Database 11*g* maps with some small exceptions to the MVCC model of a MySQL database transactional engine such as InnoDB.
9. **True** ☐ **False** ☐ Any Oracle user is synonymous with the schema of the same name.
10. **True** ☐ **False** ☐ Sequential log files are maintained by programs in MySQL databases as they are for Oracle databases, notwithstanding the engine of implementation.
11. Which of the following isn't a valid engine in a MySQL database instance?
    - **A.** Blackhole
    - **B.** CSV
    - **C.** TSV
    - **D.** Memory
    - **E.** MyISAM
12. Which of the following isn't a valid SQL category?
    - **A.** Data Query Language (DQL)
    - **B.** Data Control Language (DCL)
    - **C.** Data Manipulation Language (DML)
    - **D.** Data Definition Language (DDL)
    - **E.** Transaction Control Language (TCL)
13. Which of the following isn't a core background process for an Oracle Database 11*g* server?
    - **A.** ARCn
    - **B.** PMON
    - **C.** SMON
    - **D.** Checkpoint
    - **E.** LGWR
14. Which of the following isn't a property of an ACID-compliant transaction?
    - **A.** Atomic
    - **B.** Consistent
    - **C.** Isolated
    - **D.** Durable
    - **E.** None of the above

15. Which users have the same role and responsibility in Oracle Database 11*g* and MySQL 5.6, except for mapping to the data catalog?

- **A.** The Oracle `SYSTEM` and MySQL `root` users
- **B.** The Oracle `SYS` and MySQL `root`
- **C.** The Oracle `SYS`, `SYSTEM`, and MySQL `root`
- **D.** The Oracle `SYS` and MySQL `super`
- **E.** The Oracle `SYSTEM` and MySQL `super`