

# CHAPTER 1

## Oracle Development Overview



This chapter introduces you to Oracle PL/SQL development. Because this is a workbook, we want to keep it short and to the point. If you'd like more detail, try reading the companion book, *Oracle Database 11g PL/SQL Programming*.

We'll cover the following in this chapter:

- History and background
- Architecture

Development examples in the book use SQL\*Plus because it's the lowest common denominator when it comes to Oracle development. We'd argue that while tools are great, they're also dangerous. Their greatness lies in simplifying tasks and disclosing metadata that might be hidden for months or years without the tool. Their weakness is more subtle. Tools provide opportunities to solve problems when we may not quite understand the problem or solution. Occasionally, the solutions we choose with the tool may be suboptimal or wrong. A solid understanding of Oracle basics lets you use any tool effectively.

## History and Background

This is the short version. The idea of relational databases offered a business opportunity, and Larry Ellison saw that opportunity; with a few friends, he formed the Software Development Laboratories (SDL). That company morphed into Relational Software, Inc. (RSI), and subsequently became Oracle Corporation. Oracle then captured the majority of the relational database market.

The concept of a relational database is complex. More or less, the idea of a relational database is to (a) store information about how data is stored or structured, (b) store the data, and (c) access and manage both the structure and data through a common language. SQL, *Structured Query Language*, is that language.

Oracle innovated beyond the limited semantics of SQL and created its dialect. Oracle developers provided an if-then-else semantic through the `DECODE` statement and hierarchical queries through the `CONNECT BY` semantic. These concepts set Oracle apart from the competition. ANSI-92 adopted the if-then-else semantic, but implemented it as a case-when-else semantic. Hierarchical queries remain an Oracle dialect standalone.

In the late 1980s, Oracle saw the need for a procedural extension to SQL and created PL/SQL (Procedural Language/Structured Query Language). It was and remains innovative. Perhaps the most important aspect of PL/SQL is that you can call SQL statements from inside it, and you can call PL/SQL from SQL. This feature was maligned for many years by Oracle's competition. People still shy away from PL/SQL to stay *database agnostic*, which is a fancy way to say they want SQL solutions that are easily portable to other platforms. However, major competitors have begun to add stored procedures to their products.

Figure 1-1 shows a timeline that covers the evolution of PL/SQL in the Oracle database. We find it interesting that Oracle has provided 11 major feature upgrades during the 25 year history of the language. You'll note that

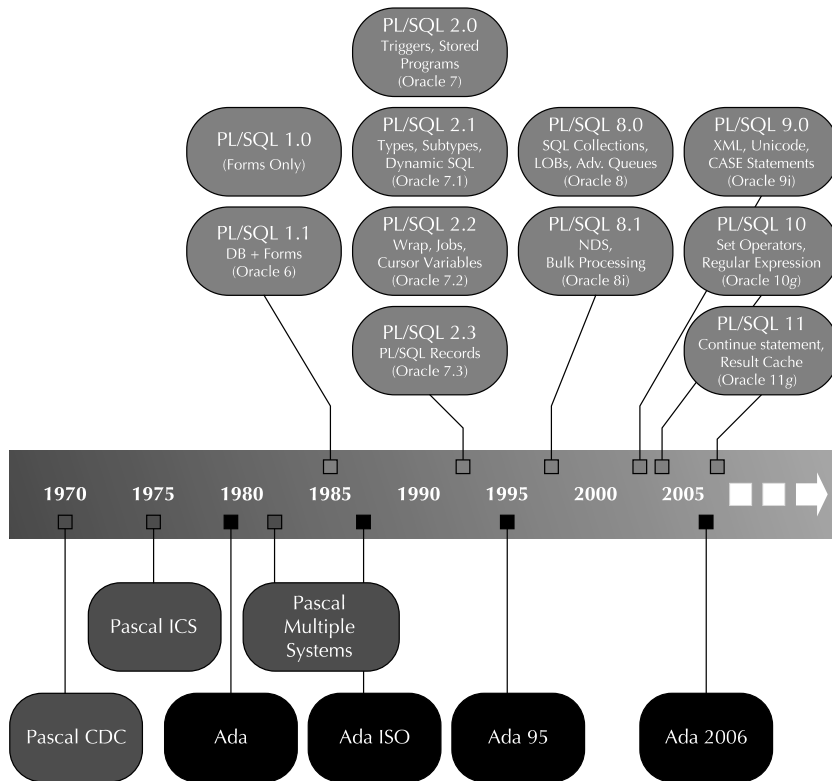


FIGURE 1-1. PL/SQL language timeline

Pascal is all but dead and gone, and Ada has only four upgrades in the same space of time. The only other language showing such feature investment is Java.

We conclude from years of experience with the product and other databases, that Oracle made the right call. PL/SQL is an extremely valuable and powerful tool for leveraging the database server. The ability to exploit it is critical to developing dynamic and effective database-centric applications.

# Architecture

The architecture of a database has many levels. We'll use a car as an analogy while discussing database architecture. The database administrator (DBA) works with the engine. The database developer sometimes works with the engine and drives the car the rest of the time. In this section, we explain how that happens.

Before we explain how to drive the "Oracle car," we'll give you a quick tour of the factory that produces the car, for two reasons: First, you need to understand some terminology if you're new to the Oracle database. Second, the same SQL that manufactures the database lets you "drive" the database. Likewise, SQL actually runs beneath the wizards that Oracle provides.

An Oracle database is composed of a series of files, processes, and a single database catalog. You create a database by using a tool, such as the Database Configuration Assistant (executable name is `dbca` in all operating systems). The Database Configuration Assistant is one of the programs that you install on the server tier when you're installing the Oracle product. Collectively, these programs are called a Database Management System (DBMS). The Database Configuration Assistant is a wizard that simplifies how you create an Oracle database.

When you use a wizard to create a database, it creates the necessary files, processes, and database catalog. The catalog is a set of tables that knows everything about the structures and algorithms of the database. You probably have heard it called metadata, or data about data.

Discussing the concept of metadata can overwhelm new users. We don't want to overwhelm you here, but you should know that great power comes with an understanding of metadata! Metadata is nothing more than a bunch of tables that define what you can store, manipulate, and access in a database. An Oracle database is also known as a *database instance*. More or less, the DBMS creates databases like factories create cars. Oracle can create more than one database instance on any server provided there's enough memory and disk space.

## The Listener

The Oracle *listener* is a background process that listens for incoming requests to the database. It listens on a single port. The listener routes requests to any database defined in its `listener.ora` configuration file. It knows which database you want based on the network alias you provide. The following shows how you connect to an Oracle Database 11g database, assuming you're using the default sample database instance:

```
# sqlplus plsql/plsql@orcl
```

The connection calls the `sqlplus` command line utility with a PLSQL user name, PLSQL user password (yes, it's trivial), and after the `@` symbol an `orcl` network alias. The network alias maps to a database through the `tnsnames.ora` file, which is part of the client software installed on the server.

The listener configuration file can support multiple databases, unlike MySQL or SQL Server, which require a listener for each database. There's great power in separating the listener from the `oracle` daemon (that's an Oracle service on Windows, and *daemon* is old English for *demon*). Oracle can also link code to one Oracle home and then use a bequeath process to connect to another Oracle home, as is done in the Oracle e-Business Suite.

The easiest analogy for a DBMS would be a word-processing program, such as Word, WordPerfect, or Pages. After installing these programs on your computer, they become factories that let you produce documents. We could probably call them document management systems (DMSs), but they're not quite that capable. They do let you create and manipulate documents. In short, they provide a user interface (UI) that lets you create and edit documents. This is like the steering wheel, accelerator, brakes, and dashboard that let you drive a car.

Oracle also provides a UI, known as SQL\*Plus. Oracle actually called its SQL\*Plus command line interface the *Advanced Friendly Interface (AFI)*, as still evidenced by the default temporary file buffer, `afiedt.buf`. As experienced users, we can testify that it isn't that advanced by today's standards, nor is it that friendly. At least that's true until you try the command line interfaces of MySQL and SQL Server. After using either, you'd probably

## 8 Oracle Database 11g PL/SQL Programming Workbook

conclude as we do that the SQL\*Plus UI is both advanced and friendly by comparison.

The command line is the basic UI, but most users adopt tools, such as Quest's Toad (expensive) or Oracle's SQL\*Developer (free). They're not that difficult to use once you understand the basics of how connections work, which we cover in this chapter.

The command line is an essential tool when you write production code. Production code must be rerunnable, which means you can run the command when it has already been run before. This means you package a set of related SQL and/or PL/SQL commands into a file, which promotes the file to a script. You run the script file from the command line or from another script that calls scripts. This is why we'll show you how to use the command line in the "Two-Tier Model" section a bit later.

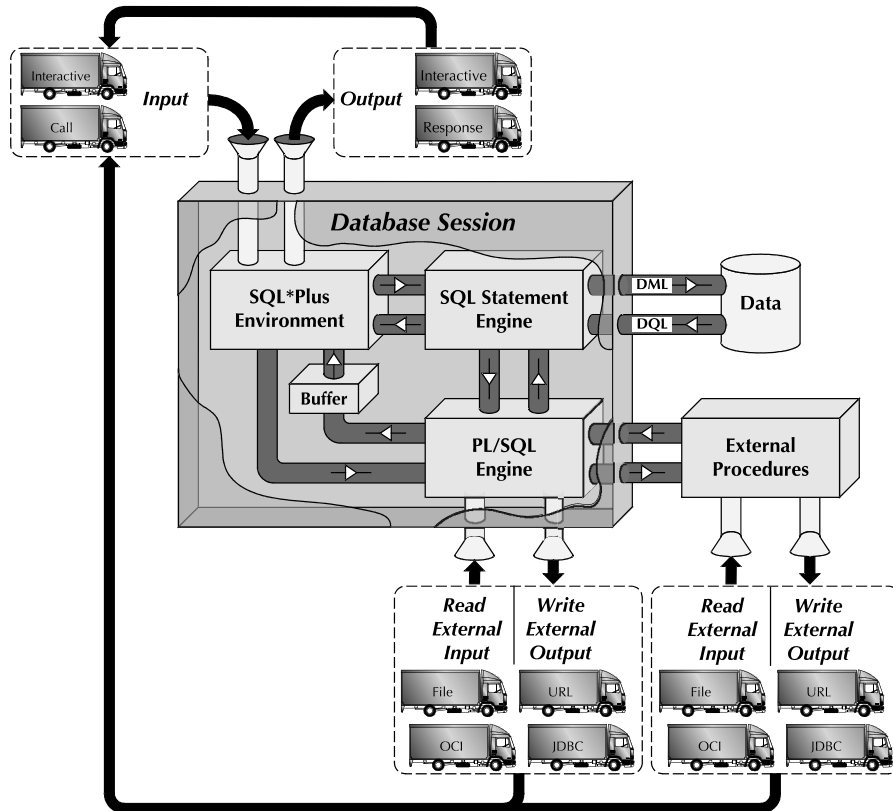
The basic architecture that provides the command line UI also supports graphical user interfaces (GUIs). That architecture is a request and acknowledgment model. You submit a request in the form of a SQL statement or an anonymous PL/SQL block, and you receive a set of data or acknowledgment that your statement or block ran. You may receive a runtime error notification when either fails. You may also receive a "no rows returned" message when no matching data is found by a query. The set of data returned is formally known as an *aggregate table*, a fancy description for a set of records.

There is only one significant difference between the SQL statement and PL/SQL block. An anonymous PL/SQL block doesn't return any data unless you open a door in the SQL\*Plus environment. You can do this with the statement

```
SQL> SET SERVEROUTPUT ON SIZE 1000000
```

This opens the buffer door (see Figure 1-2), as seen on the acknowledgment channel from the PL/SQL engine to the SQL\*Plus environment. Most GUI tools manage this for you when you run PL/SQL interactively. You need to know how to open that door when you write rerunnable production code, because you may use PL/SQL in the code and may want to output data from PL/SQL to a log file.

Figure 1-2 shows the Oracle processing architecture—or how you operate the car. Notice that all input goes in through the SQL\*Plus environment and all results or notifications return through the same environment.



**FIGURE 1-2.** Database processing architecture

The SQL statement engine processes all SQL statements. All means *all*—with no exceptions that we know about. SQL statements fall into categories. SQL statements alone interact with the data, structures, and permissions of databases and control transaction scope. SQL statements are grouped as *Data Definition Language (DDL)*, *Data Manipulation Language (DML)*, *Data Control Language (DCL)*, and *Transaction Control Language (TCL)*. While there are many variations of how you use SQL commands, only 16 basic commands are used—at least you should think of them this way for Oracle certification purposes. An excellent reference on that topic is *OCA Oracle Database 11g: SQL Fundamentals I (Exam 1Z0-051)* by Oracle Press.

Outside of the certification process, a `SELECT` statement is sometimes placed in a *Data Query Language (DQL)* category. The argument goes that it only queries the data, rather than manipulate it. The problem with the argument, however, is that when you append a `FOR UPDATE` clause to a `SELECT` statement, it locks rows in a transactional model, at least in Oracle. That sure seems like manipulation to us, but you've now read both sides of the story.

DDL statements are `CREATE`, `ALTER`, `DROP`, `RENAME`, `TRUNCATE`, and `COMMENT`. They allow you to create, alter, drop, rename, truncate, and comment tables and other objects. DML statements are `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `MERGE`. They let you query, add, change, and remove data from the database. DCL statements are `GRANT` and `REVOKE`. They let you grant and revoke privileges and groups of privileges known as *roles*. TCL statements are `COMMIT`, `ROLLBACK`, and `SAVEPOINT`. They let you control when to make data permanent or undo temporary changes.

A SQL statement can call a named PL/SQL program unit, and a PL/SQL block can call a SQL statement. A named PL/SQL program unit is a function or procedure stored in the database catalog. A PL/SQL call to a SQL statement can include only SQL data types and named PL/SQL program units stored in the database catalog. That means it can't call a locally defined function inside a SQL statement. Procedures can't be called inside a SQL statement directly; they must be contained inside a stored function. The reason that you can't call a local function inside a SQL statement is because the SQL engine doesn't have access to a local function.

In the next two sections, we discuss the connection mechanism for Oracle databases. We explain the basics of the *two-tier* computing model and the more complex *three-tier* model. They're essential to your understanding how to use SQL or PL/SQL.

## Two-Tier Model

All databases adopt a two-tier model: the engine and the interface. We call these components the *client* and *server*. The client is the interface that lets us issue SQL commands and in Oracle lets you call PL/SQL blocks. The server is the database engine.

A typical installation of the Oracle database installs both the client and server on the database server. That's because the mechanic (or DBA) who maintains the engine uses the same interface to manage many of the parts. Other server-side utilities let the DBA manage part replacement when

the server is shut down. This is similar to how you'd replace parts in an engine—you'd shut off the engine before taking it apart to replace something.

Our focus in this workbook is the interface to the running engine. We use the database server copy of the client software when we drive the database from the local server. Sometimes we want to drive the database remotely from a laptop. We have several ways to accomplish that: We can install a copy of the Oracle Client software on a remote machine, or we can use a tool such as SQL\*Developer to connect across the network using the Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) protocols.

We'll cover these communication methods in the following subsections. We begin with a standalone server because it has the fewest parts. The Oracle client separate from the server is next, because it's a natural progression from the standalone server. After laying the groundwork, we'll show you the JDBC and ODBC possibilities in a two-tier model. They come last, because they depend on either the standalone server or a local installation of the Oracle Client software.

## Oracle Standalone Communication

The basic communication model requires that a process on the local server acts as the client, and a background process (the Oracle daemon or service) acts as the server. The client process connects to the server process.

You can create a connection like this when you have access to the `sqlplus` executable and a copy of a valid `tnsnames.ora` file. This is true, unless you're the user who installed the Oracle database. That user has special permissions and can connect without going through the Oracle network layer. You find the `sqlplus` executable in the `$ORACLE_HOME/bin` on Linux, Mac OS X, or UNIX. On Windows, you find it in the `%ORACLE_HOME%\bin` directory.

SQL\*Plus is an interactive terminal that lets you connect, disconnect, or reconnect to various user accounts. A user account is a database, also known as a *user* or *schema*. Users are therefore scheme, or in some documentation, schemas. Scheme are individual databases in the scope of an Oracle database instance.

As with a terminal shell, you can set environment variables in SQL\*Plus. You can see your options by typing the following at a command prompt:

```
SQL> SHOW ALL
```

## 12 Oracle Database 11g PL/SQL Programming Workbook

If you're on the server, you can connect locally without the network. This is important, because occasionally the network layer can get trashed by user configuration errors. A local connection uses an IPC (Internal Process Communication) channel to the Oracle daemon, but you can make that connection only when you're the user who installed the database.



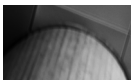
### TIP

*On Windows, you must be the Administrator account that installed the product, not just a user with Administrator rights, to connect as the local user.*

Local connections are there for the privileged user—SYS. However, you can use them to connect to any user when the network is nonoperative or the listener is simply stopped. You connect locally by using the following syntax:

```
# sqlplus system/manager
```

An alternative configuration is available through the Oracle Client component. You access it by leveraging the local `tnsnames.ora` file, or you manually provide the string that a network alias would represent. This type of connection goes through the network layer, which may be as simple as the computer's loopback, or through the DNS server to your local machine. The latter occurs when the `tnsnames.ora` file contains a reference to the hostname or IP address.



### TIP

*In some networks, outgoing calls are blocked. This means you can connect through the loopback only to a server running on your laptop. You may need to edit your `tnsnames.ora` file manually to make this change.*

You connect through the network by using the following syntax:

```
# sqlplus plsql/plsql@orcl
```

## Best Practice

Always use network alias and `tnsnames.ora` file resolution of network aliases. Strongly consider setting a `$TNS_ADMIN` or `%TNS_ADMIN%` environment variable to point to your correct `tnsnames.ora` file location.

The connection calls the `sqlplus` command line utility with a `PLSQL` user name, `PLSQL` user password (yes, it's trivial), and after the `@` symbol an `orcl` network alias. The network alias maps to a database through the `tnsnames.ora` file, which is part of the client software installed on the server. We recommend that you define the `$TNS_ADMIN` or `%TNS_ADMIN%` environment variable to ensure that you're always pointing to the correct `tnsnames.ora` file. Many developers have chewed up hours trying to figure out why they can't connect because they skipped this step.

The alternative without a `tnsnames.ora` file requires that you provide the complete Oracle Transparent Network Substrate (TNS) string:

```
C:\>sqlplus plsql/plsql@"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=hostname)
(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=orcl)))"
```

When you connect to the database with either the network alias or complete string, you form a socket communication between the client and server software. A socket provides state-aware communication, which changes to data during your session and isn't permanent until you commit changes. A failure to commit changes rolls them back. That means they're undone. You'll learn more on the nature of transaction management in Chapter 3.

### NOTE

*The Oracle TNS resolution can't contain tabs or line returns when you use the TNS string.*

You can write the session activity to a relative or local file provided you have the appropriate operating system privileges. A relative file is located in the directory where you call the `sqlplus` program. A local file is written to a fully qualified filename, which means that it includes a directory path from a *logical drive* on Windows or *mount point* on Linux, Mac OS X, or UNIX, and a filename.

## 14 Oracle Database 11g PL/SQL Programming Workbook

You open the file with

```
SQL> SPOOL C:\Data\SomeDirectory\SomeFileName.txt
```

You can run files with the @ symbol before the filename, like this:

```
SQL> @C:\Data\SomeDirectory\SomeScript.sql
```

If you've enabled the `SPOOL` before the script file runs and the script file enables the `SPOOL`, the script's log file will cancel your session's log file.

A socket connection is also known as a pessimistic connection because you control the behavior of the server from the client. This type of connection requires a connection that traverses the physical machine's loopback. This means your connection goes from your terminal session to the database daemon via the loopback. As mentioned, it may go through the DNS server when a `hostname` or IP lookup is required.

This type of communication is like driving your car in the brickyard (Indianapolis Raceway). It's on a predefined track or server.

### Oracle Client Communication

The Oracle Client software is a set of libraries that you can install on a Linux, Mac OS X, UNIX, or Windows operating system. Once they're configured, the libraries let you create a socket between your physical (or virtual) machine and a remote server.

The client software lets you create a socket with the server. Sockets between machines typically require software that can exchange messages and maintain a connection. Oracle Client software lets you launch a local client SQL\*Plus session. Through the session, you're able to establish a socket with the server and have an interactive session with the database. It provides the same control that you enjoyed from a local connection on the server.

This means you'll need a local copy of the `tnsnames.ora` file, or the fully qualified string, to connect to the remote database. You can't connect without a network alias in this configuration.

Everything else works exactly as it does in the Oracle standalone communication. The log files, script files, and connection libraries reside on the client, not the server. Client components on the server still mirror those on your client machine. You don't use the remote Oracle Client software; as a result, the Oracle Client software may be a different version than the database version. We'd recommend that you keep the versions as close as possible, and that you use Oracle's certification page for clarity on which ones to deploy in your configuration.

## Best Practice

Try to synchronize your Oracle Client software deployment with the Oracle server, and stay in compliance with the certification matrix provided by Oracle Support Services.

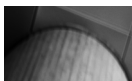
Flip back to the “Oracle Standalone Communication” section for connection syntax. As it does in the preceding section, this socket provides state-aware communication. Again, changes to data during your session aren’t permanent until you commit them. A failure to commit them rolls the changes back.

This type of communication is like driving your car on designated raceways. It requires preinstallation of software.

## JDBC Communication

JDBC changes between releases, so you should ensure that you have the right JDBC files for deployment. JDBC contains a series of class files that let you create a connection with an Oracle database. You need to place those class files inside your class path to make a connection with the database. JDBC connections are two-tier connections, and they’re like the prior communication models because they’re also pessimistic connections.

You can find the JDBC file in the `$ORACLE_HOME/jdbc/lib` on Linux, Mac OS X, or UNIX. Alternatively, you’ll find the executable in the `%ORACLE_HOME%\jdbc\lib` directory on Windows. You generally want to use the version that is consistent with the database. Java 5 and Java 6 are consistent with Oracle 11g, but you should use Java 6, which means the `ojdbc6.jar` for Western European solutions or `ojdbc6_g.jar` for globalization solutions. Equivalent files for Java 5 are deployed with Oracle Database 11g. You can learn more about configuring the JDBC for Oracle 11g in Appendix D of *Oracle Database 11g PL/SQL Programming*.



### NOTE

*Java’s license agreement is something you should read because you must be using a version within one release of the current deployment release.*

## 16 Oracle Database 11g PL/SQL Programming Workbook

You put the Oracle JDBC file into your `$CLASSPATH` environment variable on Linux, Mac OS X, or UNIX, like this,

```
# export set CLASSPATH=$ORACLE_HOME/jdbc/lib/ojdbc6.jar:.$CLASSPATH
```

or into the `%CLASSPATH%` on Windows, like this:

```
C:\> SET CLASSPATH=%ORACLE_HOME%\jdbc\lib\ojdbc6.jar;.;%CLASSPATH%
```

As described in the previous two sections, the JDBC lets you form a socket, which provides state-aware communication. When you make changes through Java, they're not permanent until you commit them. Without a commit, the changes are undone and rolled back. (See Chapter 3.)

This type of communication is like driving on the local roads. You need software to make it work, but it's free, flexible, and easy to deploy.

### ODBC Communication

ODBC depends on a shared library that works much like the JDBC library. The library is often language-dependent and always platform-dependent. A Dynamic Link Library (DLL) is required for Windows, and a shared object library is required for Linux, Mac OS X, and UNIX.

The ODBC also forms a socket and all rules that apply about transactions for the previous two-tier methods apply here. You generally must purchase the ODBC driver files. You can build your own from the stubs that Oracle provides, but be warned that it's not that straightforward.

Microsoft Office 2007 ships with an ODBC DLL for the Windows platform, but Microsoft Office 2008 for the Mac OS X platform doesn't ship with a DLL. While we can't make a recommendation (legal risks, and so on), you can find ODBC libraries through Actual Technologies or OpenLink Software that are rumored to work for Mac OS X.

This type of communication is like driving on a toll road. You must have the appropriate software and prepay the fees. However, once you've prepaid you can even do a bit of off-road hot-dogging.

### Three-Tier Model

All databases support a three-tier model, because it's really just a middleware solution. The middleware can be a multithreaded JServlet, Apache module, or general software appliance. The middleware generally creates a pool of connections to the Oracle database and shares the connections with requests made by other clients.

Typically in a three-tier model, the client-to-middleware communication doesn't enjoy a state-aware connection. In fact, it's often stateless through the Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure (HTTP/HTTPS) protocols.

This shift in communication semantics means changes are automatic and permanent when they occur. If you submit a data change via an `INSERT`, `UPDATE`, or `DELETE` statement across HTTP/HTTPS and receive acknowledgment of success, that change is permanent. This is known as an optimistic processing model. It alone is a reason for stored procedures that manage transactions across multiple tables in any database.

The exception to an optimistic process occurs when the middleware maintains a lock on the data and manages your transaction scope for you. This type of implementation is done for you by default in Oracle Application Express (APEX), because Oracle APEX maintains a transaction and persistent object state for your activities. The mechanics of how this works would require a chapter of its own. Suffice it to say, this is a possible architecture for your internally developed applications.

## Downloadable Code

Because we haven't included any examples in this chapter, there aren't any bundled files.

## Summary

This chapter provided a tour of the Oracle development environment for client- and server-side PL/SQL development. You should be positioned to understand, work, and experiment with the examples from the other chapters.

## Best Practice Review

- Always use network alias and `tnsnames.ora` file resolution of network aliases. Strongly consider setting a `$TNS_ADMIN` or `%TNS_ADMIN%` environment variable to point to the correct `tnsnames.ora` file.
- Try to synchronize your Oracle Client software deployment with the Oracle server, and stay in compliance with the certification matrix provided by Oracle Support Services.

## 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 Appendix E for answers to these questions.

1.  **True**    **False**   You use DDL statements to create tables.
2.  **True**    **False**   You use DML statements to manipulate data.
3.  **True**    **False**   You use DCL statements to grant or revoke privileges.
4.  **True**    **False**   You use TCL statements to control timestamps.
5.  **True**    **False**   A `SELECT` statement is a DQL statement, not a DML statement, in all cases.
6.  **True**    **False**   The PL/SQL version numbers have always been synchronized with the Oracle Database release numbers.
7.  **True**    **False**   You can't connect without a network alias.
8.  **True**    **False**   You must have a local copy of a `tnsnames.ora` file when you use Oracle Client software.
9.  **True**    **False**   Oracle, unlike MySQL, includes its listener service as a separate process from its database server-side daemon.
10.  **True**    **False**   You can spool individual script files without interfering with a session spool file.
11. Which are valid two-tier communication types?
  - A. Oracle standalone communication
  - B. Oracle client communication
  - C. JDBC communication
  - D. All of the above
  - E. Only B and C

12. Which of the following is a valid JDBC driver for Oracle 11g?
  - A. The `ojdbc6_g.jar` file
  - B. The `ojdbc6.jar` file
  - C. The `ojdbc5.jar` file
  - D. All of the above
  - E. Only A and B
13. Which of the following does Oracle deploy with each release of the database server?
  - A. The Oracle Client software
  - B. The JDBC driver files
  - C. The ODBC driver files
  - D. All of the above
  - E. Only A and B
14. Where does the state-aware connection exist in a three-tier model?
  - A. Between the JServlet middle-tier and the database
  - B. Between the client and the database
  - C. Between the Apache server and the database
  - D. All of the above
  - E. Only B and C
15. Which tier can create pooled connections to share?
  - A. The client
  - B. The server
  - C. The middle-tier
  - D. None of the above
  - E. Only A and B

