Introduction to the Oracle Server Technologies

CERTIFICATION OBJECTIVES

1.01 Position the Oracle Product Family
1.02 Explain Relational Structures
1.03 Understand the SQL Language
1.04 Appreciate the DBA’s Role
✓ Two-Minute Drill
Q&A Self Test
Chapter 1: Introduction to the Oracle Server Technologies

This chapter describes the entire Oracle product family, the concepts behind relational databases, the SQL language, and the role of the database administrator (DBA) in the IT environment. The content is not directly tested in the OCP examination, but it is assumed knowledge that provides an essential background to the study of Oracle database administration. It also gives an idea of the scope of a DBA’s work. The DBA is often expected to know everything about everything. Well, that isn’t possible—but at least he/she should be aware of the whole environment and how the parts link together.

The Oracle server technologies product set is more than a database. There are also the Oracle Application Server and the Oracle Enterprise Manager. Taken together, these are the server technologies that make up the Oracle Grid offering. Grid computing is an emerging environment for managing the complete IT environment and for providing resources to users on demand.

Supplied with the server technologies are the development tools: third-generation languages (3GLs) and rapid application development tools. It is also possible to write software in a wide range of third-party application development environments. Oracle Corporation has a total commitment to supporting international standards for data processing, which means that it is perfectly possible to run third-party products on the Oracle technology stack.

The final part of the Oracle product set is the applications, primarily the Oracle E-Business Suite and the Oracle Collaboration Suite. These are applications written with the Oracle development tools and running on the Oracle servers that can fulfill the needs of virtually any organization for business data processing.

The Oracle database is a relational database management system (RDBMS) with object extensions. Data to be managed by an RDBMS should be normalized—converted into two-dimensional tables. Structured query language (SQL, pronounced “sequel”) is an international standard for managing data stored in relational databases. Oracle Database 11g offers an implementation of SQL that is generally compliant with the current standard, which is SQL-2003. Full details of the compliance can be found in Appendix B of the “SQL Reference,” which is part of the Oracle database documentation set.

The DBA’s job is to administer the database. But this tautological definition misses a huge amount. The DBA has a pivotal role in an organization’s management and knowledge structure; he/she will usually be expected to be competent in all the topics dealt with here.

This chapter consists of summarized descriptions of the Oracle product family, the concepts behind the relational paradigm and the normalization of data into relational structures, the SQL language, and the role of the DBA.
CERTIFICATION OBJECTIVE 1.01

Position the Oracle Product Family

There are many products developed and marketed by Oracle Corporation. Acquisitions made in recent years have expanded the list substantially. The core products can be grouped as follows:

The server technologies
- The Oracle database
- The Oracle Application Server
- The Oracle Enterprise Manager

The development tools
- The languages
- The Oracle Developer Suite

The applications
- Oracle E-Business Suite
- Oracle Collaboration Suite

The Server Technologies

There is a family of products that make up the Oracle server technology stack. Taken together, they aim to deliver the components of Oracle’s Grid computing architecture.

Oracle Corporation’s Grid offering is not always the same as the Grid defined by others. Oracle does tend to emphasize the server technology side of Grid computing.

The concept underlying the Grid is virtualization: end users ask for a service, but they neither know nor need to know the source of that service.

The three server technologies that Oracle provides to help deliver the Grid are
- The Oracle Database Server
- The Oracle Application Server
- The Oracle Enterprise Manager
Chapter 1: Introduction to the Oracle Server Technologies

The database is the main concern here: it is the repository for data and the engine that manages access to the data. The Oracle Application Server runs software on behalf of end users: it generates the user interfaces in the form of windows displayed in users’ browsers, and submits calls for data retrieval and modification to the database for execution. The Oracle Enterprise Manager is an administration tool for monitoring, managing, and tuning the Oracle processes and also (through plug-ins) many third-party products.

The Oracle Database Server
Chapter 2 deals with the architecture of the Oracle Database Server in detail. For now, all that is necessary is to know that it is a repository for data without effective limits on size, and that is accessed by end users either directly through client-server tools and applications or indirectly through applications running on an application server.

The choice between a two-tier client-server architecture and an architecture consisting of three or more tiers is irrelevant to the database: it can execute SQL invoked by client software running on a PC local to the end user as efficiently as it can execute SQL submitted by an application server running on a middle-tier server on behalf of a remote client using a browser.

The Oracle Application Server
With the emergence of the Web as the de facto standard platform for delivering applications to end users has come the need for application servers. An application server replaces the client-side software traditionally installed on end-user computers: it runs applications centrally, presenting them to users in windows displayed locally in web browsers. The applications make use of data stored in one or more database servers.

Oracle Application Server is a platform for developing, deploying, and managing web applications. A web application can be defined as any application with which users communicate via HTTP. Web applications usually run in at least three tiers: a database tier manages access to the data, the client tier (often implemented as a web browser) handles the local window management for communications with the users, and an application tier in the middle executes the program logic that generates the user interface and the SQL calls to the database.

Web applications can be developed with a number of technologies, predominant among which today is Java. Applications written in Java should conform to the J2EE (Java 2 Enterprise Edition) standard, which defines how such applications
should be packaged and deployed. J2EE and related standards are controlled by Sun Microsystems and accepted by virtually all software developers. Oracle Application Server is a J2EE-compliant application server. Oracle’s implementation of the standards allows for automatic load balancing and fault tolerance across multiple application servers on multiple machines though J2EE clustering. Clustering virtualizes the provision of the application service: users ask for an application, which might be available from a number of locations, and the cluster works out from where any one session or request can best be serviced. If one location fails, others will take up the load, and more resources can be made available to an application as necessary. The ability to separate the request for a service from the location of its provision and to add or remove J2EE servers from a cluster dynamically is a major part of the Oracle Application Server’s contribution to the Grid.

It is important to note that Oracle’s commitment to international standards is total. Applications running in the Oracle Application Server environment can connect to any databases for which there are Java-compliant drivers: it is not necessary to use an Oracle database. Applications developed with the Oracle Application Server toolkits could be deployed to a third-party J2EE-compliant application server. However, the Oracle product set is particularly powerful and will often be the best choice.

The simplest processing model of web applications is three tier: a client tier that manages the user interface, a middle tier that generates the interface and issues SQL statements to the data tier, and a data tier that manages the data itself. In the Oracle environment, the client tier will be a browser (such as Mozilla or Microsoft Internet Explorer) that handles local window management, handles the keyboard input, and tracks mouse movements. The middle tier will be an Oracle Application Server running the software (probably written in Java) that is generating the windows sent to the client tier for display, and the SQL statements sent to the data tier for execution. The data tier will be an Oracle server: an instance and a database. In this three-tier environment, there are two types of session: end-user sessions from the client tier to the middle tier, and database sessions from the middle tier to the data tier. The end-user sessions will be established with HTTP. The database sessions are client-server sessions consisting of a user process and a server process, as described in the preceding section.

It is possible for an application to use a one-for-one mapping of end-user session to database session: each user, from his/her browser, will establish a session against the application server, and the application server will then establish a session against the database server on the user’s behalf. However, this model has been proven to be very inefficient when compared to the connection pooling model. With connection
pooling, the application server establishes a relatively small number of persistent
database sessions and makes them available on demand (queuing requests if necessary)
to a relatively large number of end-user sessions against the application server.

From the point of view of the database, it makes no difference whether a SQL
statement comes from a client-side process such as SQL*Plus or Microsoft Access or
from a pooled session to an application server. In the former case, the user process
all happens on one machine; in the latter, the user process has been divided into
two tiers: an applications tier that generates the user interface and a client tier that
displays it. But the database tier really doesn’t care.

DBAs often find themselves pressed into service as Application Server
administrators. Be prepared for this. There is a separate OCP curriculum for
Application Server, for which it may well be worth studying.

Oracle Enterprise Manager

The increasing size and complexity of IT installations makes management of each
given component a challenging task. This is hardly surprising: no one ever said that
managing a powerful environment should necessarily be simple. Management tools
can make the task easier, and the management staff more productive.

Oracle Enterprise Manager comes in three forms:

- Database Control
- Application Server Control
- Grid Control

Oracle Enterprise Manager Database Control is a graphical tool for managing
one database, which may be a Real Application Clusters (RAC) clustered database.
RAC databases are covered in more advanced books; they are mentioned here
because they can be managed through the tool. Database Control has facilities for
real-time management and monitoring, for running scheduled jobs such as backup
operations, and for reporting alert conditions interactively and through e-mail. An
RAC database will have a Database Control process running on each node where
there is a database instance; these processes communicate with each other, so that
each has a complete picture of the state of the RAC.

Oracle Enterprise Manager Application Server Control is a graphical tool for
managing one application server instance, or a group of instances. The grouping
technology is dependent on the version. Up to and including Oracle Application
Server 10g release 2, multiple application servers were managed as a farm, with
a metadata repository (typically residing in an Oracle database) as the central management point. This is an excellent management model and offers superb capabilities for deploying and maintaining applications, but it is proprietary to Oracle. From Application Server 10g release 3 onward, the technology is based on J2EE clustering, which is not proprietary to Oracle.

Both Database Control and Application Server Control consist of a Java process running on the server machine, which listens for HTTP and HTTPS connection requests. Administrators connect to these processes from a browser. Database Control then connects to the local database server, and Application Server Control connects to the local application server. An advantage of using browser access is that remote management should be no problem. Most, though naturally not all, sites will permit incoming HTTP or HTTPS connections through their firewalls.

Oracle Enterprise Manager Grid Control globalizes the management environment. A management repository (residing in an Oracle database) and one or more management servers manage the complete environment: all the databases and application servers, wherever they may be. Grid Control can also manage the nodes, or machines, on which the servers run, and (through plug-ins) a wide range of third-party products. Each managed node runs an agent process, which is responsible for monitoring the managed targets on the node: executing jobs against them and reporting status, activity levels, and alert conditions back to the management server(s).

Grid Control gives a holistic view of the environment, and if well configured makes administration staff far more productive than without. It becomes possible for one administrator to manage effectively hundreds or thousands of targets. All communications are over HTTP or HTTPS. Provided the corporate firewalls are configured to permit these protocols, it becomes possible for the administration staff to connect to the management servers from any location that has a browser. Furthermore, the communications between the agents running on the various nodes and the management server(s) are also HTTP or HTTPS, so an organization’s complete environment—even if spread over many geographically separate sites, using the Internet for inter-site communications—can be managed as a whole.

The core functionality of Oracle Enterprise Manager (OEM) is available for no additional charge—it is bundled with the database or application server license. There are, however, additional packs that are separately licensed. The packs consist for the most part of wizards that make the work of monitoring, tuning, and general management easier. Some DBAs make extensive use of Oracle Enterprise Manager and rely on the packs. Others (perhaps the older ones) still prefer to work with command-line utilities such as SQL*Plus. There is little doubt that using Oracle Enterprise Manager can make a DBA far more productive, but it is by no means compulsory.
Chapter 1: Introduction to the Oracle Server Technologies

Grid Computing

Critical to the concept of Grid computing is virtualization. This means that at all levels there is a layer of abstraction between what is requested and what is provided. In the Oracle Grid environment, end users ask for an application service and let the Grid work out which clustered J2EE application server can best provide it. Application servers ask for database service from an RAC database and let the Grid work out from which RAC instance the data service can best be provided. Within the Grid there is a mapping of possible services to available service providers, and algorithms for assigning the workload and resources appropriately. The result is that end users have neither the need nor the capacity to know from where their computing resources are actually being provided. The analogy often drawn is with delivery of domestic electricity: it is supplied on demand, and the home owner has no way of telling which power station is currently supplying him/her.

The Grid is not exclusive to Oracle. At the physical level, some operating system and hardware vendors are providing Grid-like capabilities. These include the ability to partition servers into virtual machines, and dynamically add or remove CPU(s) and RAM from the virtual machines according to demand. This is conceptually similar to Oracle’s approach of dynamically assigning application server and database server resources to logical services. There is no reason why the two approaches cannot be combined. Both are working toward the same goal and can work together. The result should be an environment where adequate resources are always available on demand, without facing the issues of excess capacity at some times and underperformance at others. It should also be possible to design a Grid environment with no single point of failure, thus achieving the goal of 100 percent uptime, which is being demanded by many users.

The Development Tools and Languages

The Oracle server technologies include various facilities for developing applications, some existing within the database, others external to it.
Within the database, it is possible to use three languages:

- SQL
- PL/SQL
- Java

Applications running externally to the database can be written in a wide variety of 3GLs (notably Java) or in the tools shipped with Oracle Application Server. Predominant among these are

- Forms
- Reports
- XML Publisher
- Discoverer

There is also a wide variety of third-party tools and environments that can be used for developing applications that will connect to an Oracle database; in particular .NET from Microsoft, for which Oracle provides a comprehensive developers’ toolkit.

**Languages Internal to the Database**

The one language that is unavoidable is SQL. SQL is used for data access, but it cannot be used just on its own for developing complete applications. It has no real facilities for developing user interfaces, and it also lacks the procedural structures needed for manipulating rows individually. The other two languages available within the database fill these gaps. They are PL/SQL and Java. PL/SQL is a 3GL proprietary to Oracle. It has the usual procedural constructs (such as conditional branching based on if-then-else and iterative looping) and facilities for user interface design. In the PL/SQL code, one can embed calls to SQL. Thus, a PL/SQL application might use SQL to retrieve one or more rows from the database, then perform various actions based on their content, and then issue more SQL to write rows back to the database. Java offers a similar capability to embed SQL calls within the Java code. This is industry-standard technology: any Java programmer should be able write code that will work with an Oracle database (or indeed with any other Java-compliant database.)

All Oracle DBAs must be fully competent with SQL and PL/SQL. This is assumed, and required, knowledge.

Knowledge of Java is not assumed and indeed is rarely required. A main reason for this is that bespoke Java applications are now rarely run within the database.
Chapter 1: Introduction to the Oracle Server Technologies

Early releases of Oracle’s application server could not run some of the industry-standard Java application components, such as Java servlets and Enterprise JavaBeans (EJBs). To get around this serious divergence from standards, Oracle implemented a Java engine within the database that did conform to the standards. However, from Oracle Application Server release 9i, it has been possible to run servlets and EJBs where they should be run: on the application server middle tier. Because of this, it is becoming less common to run Java within the database.

Some Oracle components, such as Intermedia and Text, are written in Java, and for this reason a DBA will usually need to enable Java in the database, but he/she will not be expected to tune or debug these. The DBA is, however, likely to spend a large amount of time tuning and debugging SQL and PL/SQL. Oracle’s model for the division of responsibility here is clear: the DBA identifies code with problems and passes it to the developers for fixing. But in many cases, the developers lack the skills (or perhaps the inclination) to do this and the DBA has to fill this role. He/she will also often become a teacher: spreading knowledge of better techniques among the developers.

All DBAs must be fully competent with SQL and with PL/SQL. Knowledge of Java and other languages is not usually required but is often helpful.

Languages External to the Database

Other languages are available for developing client-server applications that run externally to the database. The most commonly used are C and Java, but it is possible to use most of the mainstream 3GLs. For most languages, Oracle provides the OCI (Oracle Call Interface) libraries that let code written in these languages connect to an Oracle database and invoke SQL commands.

Applications written in C or other procedural languages make use of the OCI libraries to establish sessions against the database server. These libraries are proprietary to Oracle. This means that any code using them will be specifically written for Oracle, and would have to be substantially rewritten before it could run against any other database. Java applications can avoid this problem. Oracle provides database connectivity for both thick and thin Java clients.

A thick Java client is Oracle aware. It uses the supplied OCI class library to connect to the database. This means that the application can make use of all the database’s capabilities, including features that are unique to the Oracle environment. Java-thick client applications can exploit the database to the full. But they can never work with a third-party product, and they require the OCI client software to be installed.
A thin Java client is not aware of the database against which it is running: it works with a virtual database defined according to the Java standard, and it lets the container within which it is running map this virtual database onto the Oracle database. This gives the application portability across database versions and providers: a thin Java client application could be deployed in a non-Oracle environment without any changes. But any Oracle features that are not part of the Java database connectivity standard will not be available.

The choice between thick and thin Java clients should be made by a team of informed individuals and influenced by a number of factors including performance, the need for Oracle-specific features, corporate standards, application portability, programmer productivity. Oracle's JDeveloper tool can be used to develop both thick- and thin-client Java applications.

Oracle Developer Suite
Many organizations will not want to use a 3GL to develop database applications. Oracle Corporation provides rapid application development tools as part of the Oracle Developer Suite. These can make programmers far more productive than if they were working with a 3GL. Like the languages, all these application development tools end up doing the same thing: constructing SQL statements that are sent to the database server for execution.

Oracle Forms Developer builds applications that run on an Oracle Application Server middle tier and display in a Java applet on the user's terminal. The entry point to Forms applications is through a browser. Forms applications are generally much faster to develop than applications written in, say, Java. An advantage for end users is that the client-side intelligence provided by the applet means that Forms applications can have a better user interface than applications with user interfaces generated by Java servlets. Servlet applications are restricted by the limitations of HTML tags, whereas the Forms applet can generate any user interface gadget that may be required. For example, there is no HTML tag for a rolling combo box, but these are no problem for Forms.

Oracle Reports is a tool for generating and formatting reports, either on demand or according to a schedule. Completed reports can be cached for distribution. As with Forms, programmer productivity can be higher than when working with a 3GL. An Oracle Reports application does impose restrictions on the client machine because of the possible output formats; these are commonly HTML or PDF. The restrictions come from the need for the client to be able to display these formats, and the need for the developer to be aware of the clients' capabilities. XML Publisher
Chapter 1: Introduction to the Oracle Server Technologies

avoids these issues by formatting a report’s output as XML tags. Any client can request an XML Publisher report and (provided it has an XML parser) display the results. This is the key to distributing reports over wireless protocols to any device, such as a cellular telephone.

Oracle Discoverer is an end-user tool for report generation. Oracle Reports and XML Publisher need a programmer to design the report. A well-designed report can be highly customizable by the end user through use of parameters supplied at request time, but a programmer is still needed to design the report definition. Oracle Discoverer empowers end users to develop reports themselves. Once Oracle Discoverer, which runs on an Oracle Application Server middle tier, has been appropriately configured, no more programmer input is needed: the end users do all the development. Discoverer can add immense value for end users, while freeing up programming staff for real development work.

The Oracle Applications

The number of Oracle applications products has increased substantially in recent years due to a large number of corporate acquisitions, but two applications remain predominant. The Oracle E-Business Suite is a comprehensive suite of applications based around an accounting engine and Oracle Collaboration Suite is a set of office automation tools.

The Oracle E-Business Suite, based around a core of financial applications, includes facilities for accounting, human resources, manufacturing, customer relationship management, customer services, and much more. All the components share a common data model. The current release has a user interface written with Oracle Developer Forms and Java, depending on which tool is most suitable for the various modules and the expected users, running on Oracle Application Server. There is a large amount of PL/SQL in the database to enable the business functions. Future releases will merge the functionality of other products acquired recently (such as the Siebel and Peoplesoft applications) into a common Java-based interface.

The Oracle Collaboration Suite includes (among other things) servers for e-mail, diary management, voicemail and fax, web conferencing, and (perhaps most impressive) file serving. There is complete integration between the various components. The applications run on Oracle Application Servers, and can be accessed through a web interface from browsers or made available on mobile wireless devices, such as cellular phones.
**EXERCISE 1-1**

Investigate DBMSs in Your Environment

This is a paper-based exercise, with no specific solution.

Identify the applications, application servers, and databases used in your environment. Then, concentrating on the databases, try to get a feeling for how big and busy they are. Consider the number of users, the volatility of the data, and the data volumes. Finally, consider how critical they are to the organization: how much downtime or data loss can be tolerated for each application and database? Is it possible to put a financial figure on this?

The result of this study should give an idea of how critical the DBA’s role is.

---

**CERTIFICATION OBJECTIVE 1.02**

**Explain Relational Structures**

Critical to an understanding of SQL is an understanding of the relational paradigm, and the ability to normalize data into relational structures. Normalization is the work of systems analysts, as they model business data into a form suitable for storing in relational tables. It is a science that can be studied for years, and there are many schools of thought that have developed their own methods and notations.

**Rows and Tables**

Using the relational paradigm, data is stored in two-dimensional tables. A table consists of a number of rows, each consisting of a set of columns. Within a table, all the rows have the same column structure, though it is possible that in some rows some columns may have nothing in them. An example of a table would be a list of one’s employees, each employee being represented by one row. The columns might be employee number, name, and a code for the department in which he/she works. Any employees not currently assigned to a department would have that column blank. Another table could represent the departments: one row per department, with columns for the department’s code and the department’s name.
Chapter 1: Introduction to the Oracle Server Technologies

A note on terminology: what Oracle refers to as a table may also be called a relation or an entity. Rows are sometimes called records or tuples, and columns may be called attributes or fields. The number of rows in the table is the cardinality of the tuples.

Relational tables conform to certain rules that constrain and define the data. At the column level, each column must be of a certain data type, such as numeric, date-time, or character. The “character” data type is the most general, in that it can accept anything. At the row level, usually each row must have some uniquely identifying characteristic: this could be the value of one column, such as the employee number and department number in the examples just given, that cannot be repeated in different rows. There may also be rules that define links between the tables, such as a rule that every employee must be assigned a department code that can be matched to a row in the departments table. Following are examples of the tabulated data definitions.

Departments table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPTNO</td>
<td>Department number</td>
<td>Numeric</td>
<td>2</td>
</tr>
<tr>
<td>DNAME</td>
<td>Department name</td>
<td>Character</td>
<td>14</td>
</tr>
</tbody>
</table>

Employees table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPNO</td>
<td>Employee number</td>
<td>Numeric</td>
<td>4</td>
</tr>
<tr>
<td>ENAME</td>
<td>Employee name</td>
<td>Character</td>
<td>10</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>Department number</td>
<td>Numeric</td>
<td>2</td>
</tr>
</tbody>
</table>

The tables could contain the rows shown next.

Departments:

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>DNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ACCOUNTING</td>
</tr>
<tr>
<td>20</td>
<td>RESEARCH</td>
</tr>
<tr>
<td>30</td>
<td>SALES</td>
</tr>
<tr>
<td>40</td>
<td>OPERATIONS</td>
</tr>
</tbody>
</table>
Employees:

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>DEPTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7369</td>
<td>SMITH</td>
<td>20</td>
</tr>
<tr>
<td>7499</td>
<td>ALLEN</td>
<td>30</td>
</tr>
<tr>
<td>7521</td>
<td>WARD</td>
<td>30</td>
</tr>
<tr>
<td>7566</td>
<td>JONES</td>
<td>20</td>
</tr>
<tr>
<td>7654</td>
<td>MARTIN</td>
<td>30</td>
</tr>
<tr>
<td>7698</td>
<td>BLAKE</td>
<td>30</td>
</tr>
<tr>
<td>7782</td>
<td>CLARK</td>
<td>10</td>
</tr>
<tr>
<td>7788</td>
<td>SCOTT</td>
<td>20</td>
</tr>
</tbody>
</table>

Looking at the tables, the two-dimensional structure is clear. Each row is of fixed length, each column is of fixed length (padded with spaces when necessary), and the rows are delimited with a new line. The rows have been stored in code order, but this would be a matter of chance, not design; relational tables do not impose any particular ordering on their rows. Department number 10 has one employee, and department number 40 has none. Changes to data are usually very efficient with the relational model. New employees can be appended to the employees table, or they can be moved from one department to another simply by changing the DEPTNO value in their row.

Consider an alternative structure, where the data is stored according to the hierarchical paradigm. The hierarchical model was developed before the relational model, for technology reasons. In the early days of computing, storage devices lacked the capability for maintaining the many separate files that were needed for the many relational tables. Note that this problem is avoided in the Oracle database by abstracting the physical storage (files) from the logical storage (tables); there is no direct connection between tables and files, and certainly not a one-to-one mapping. In effect, many tables can be stored in a very few files.

A hierarchical structure stores all related data in one unit. For example, the record for a department would include all that department’s employees. The hierarchical paradigm can be very fast and very space efficient. One file access may be all that is needed to retrieve all the data needed to satisfy a query. The employees and departments listed previously could be stored hierarchically as follows:

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>DEPARTMENT</th>
<th>EMPNO</th>
<th>ENAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ACCOUNTING</td>
<td>7782</td>
<td>CLARK</td>
</tr>
<tr>
<td>20</td>
<td>RESEARCH</td>
<td>7369</td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7566</td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7788</td>
<td>SCOTT</td>
</tr>
<tr>
<td>30</td>
<td>SALES</td>
<td>7499</td>
<td>ALLEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7521</td>
<td>WARD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7654</td>
<td>MARTIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7698</td>
<td>BLAKE</td>
</tr>
<tr>
<td>40</td>
<td>OPERATIONS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1: Introduction to the Oracle Server Technologies

In this example layout, the rows and columns are of variable length. Columns are delimited with a comma, rows with a new line. Data retrieval is typically very efficient if the query can navigate the hierarchy: if one knows an employee's department, the employee can be found quickly. If one doesn't, the retrieval may be slow. Changes to data can be a problem if the change necessitates movement. For example, to move employee 7566, JONES, from RESEARCH to SALES would involve considerable effort on the part of the database because the move has to be implemented as a removal from one line and an insertion into another. Note that in this example, while it is possible to have a department with no employees (the OPERATIONS department), it is absolutely impossible to have an employee without a department: there is nowhere to put her.

The relational paradigm is highly efficient in many respects for many types of data, but it is not appropriate for all applications. As a general rule, a relational analysis should be the first approach taken when modeling a system. Only if it proves inappropriate should one resort to non-relational structures. Applications where the relational model has proven highly effective include virtually all online transaction processing (OLTP) systems and decision support systems (DSSs). The relational paradigm can be demanding in its hardware requirements and in the skill needed to develop applications around it, but if the data fits, it has proved to be the most versatile model. The problems arise from the need to maintain indexes that give the versatility of access of maintain the links between tables, and the space requirements of maintaining multiple copies of the indexed data in the indexes themselves and in the tables in which the columns reside. Nonetheless, relational design is in most circumstances the optimal model.

A number of software publishers have produced database management systems that conform (with varying degrees of accuracy) to the relational paradigm; Oracle is only one. IBM was perhaps the first company to commit major resources to it, but its product (which later developed into DB2) was not ported to non-IBM platforms for many years. Microsoft's SQL Server is another relational database that has been limited by the platforms on which it runs. Oracle databases, by contrast, have always been ported to every major platform from the first release. It may be this that gave Oracle the edge in the RDBMS market place.

A note on terminology: confusion can arise when discussing relational databases with people used to working with Microsoft products. SQL is a language and SQL Server is a database—but in the Microsoft world, the term “SQL” is often used to refer to either.
Data Normalization

The process of modeling data into relational tables is known as normalization. There are commonly said to be three levels of normalization: the first, second, and third normal forms. There are higher levels of normalization: fourth and fifth normal forms are well defined, but any normal data analyst (and certainly any normal human being) will not need to be concerned with them. It is possible for a SQL application to address un-normalized data, but this will usually be dreadfully inefficient because that is not what the language is designed to do. In most cases, data stored in a relational database and accessed with SQL should be normalized to the third normal form.

As an example of normalization, consider a table called BOOKS storing details of books, authors, and publishers, using the ISBN number as the primary key. A primary key is the one attribute that can uniquely identify a record. These are two typical entries:

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Title</th>
<th>Authors</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Oracle 11g SQL Fundamentals 1 Exam Guide</td>
<td>John Watson, Roopesh Ramklass</td>
<td>McGraw-Hill, Spear Street, San Francisco, CA</td>
</tr>
<tr>
<td>67890</td>
<td>Oracle 11g New Features Exam Guide</td>
<td>Sam Alapati</td>
<td>McGraw-Hill, Spear Street, San Francisco, CA</td>
</tr>
</tbody>
</table>

Storing the data in this table gives rise to several anomalies. First, here is the insertion anomaly: it is impossible to enter details of authors who are not yet published, because there will be no ISBN number under which to store them. Second, a book cannot be deleted without losing the details of the publisher: a deletion anomaly. Third, if a publisher’s address changes, it will be necessary to update the rows for every book he/she has published: an update anomaly. Furthermore, it will be very difficult to identify every book written by one author. The fact that a book may have several authors means that the “author” field must be multivalued, and a search will have to search all the values. Related to this is the problem of having to restructure the table of a book comes along with more authors than the original design can handle. Also, the storage is very inefficient due to replication of address details across rows, and the possibility of error as this data is repeatedly entered is high. Normalization should solve all these issues.
Chapter 1: Introduction to the Oracle Server Technologies

The first normal form is to remove the repeating groups. In this case, the multiple authors: pull them out into a separate table called AUTHORS. The data structures will now look like this:

**BOOKS**

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Title</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Oracle 11g SQL Fundamentals 1 Exam Guide</td>
<td>McGraw-Hill, Spear Street, San Francisco, CA</td>
</tr>
<tr>
<td>67890</td>
<td>Oracle 11g New Features Exam Guide</td>
<td>McGraw-Hill, Spear Street, San Francisco, CA</td>
</tr>
</tbody>
</table>

**AUTHORS**

<table>
<thead>
<tr>
<th>Name</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Watson</td>
<td>12345</td>
</tr>
<tr>
<td>Roopesh Ramklass</td>
<td>12345</td>
</tr>
<tr>
<td>Sam Alapati</td>
<td>67890</td>
</tr>
</tbody>
</table>

One row in the BOOKS table is now linked to two rows in the AUTHORS table. This solves the insertion anomaly (there is no reason not to insert as many unpublished authors as necessary), the retrieval problem of identifying all the books by one author (one can search the AUTHORS table on her name), and the problem of a fixed maximum number of authors for any one book (simply insert as many or as few AUTHORS as are needed).

This is the first normal form: no repeating groups.

The second normal form removes columns from the table that are not dependent on the primary key. In this example, that is the publisher's address details: these depend on the publisher, not the ISBN. The BOOKS table and a new PUBLISHERS table will then look like this:

**BOOKS**

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Title</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Oracle 11g OCP SQL Fundamentals 1 Exam Guide</td>
<td>McGraw-Hill</td>
</tr>
<tr>
<td>67890</td>
<td>Oracle 11g New Features Exam Guide</td>
<td>McGraw-Hill</td>
</tr>
</tbody>
</table>
PUBLISHERS

<table>
<thead>
<tr>
<th>Publisher</th>
<th>Street</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>McGraw-Hill</td>
<td>Spear Street</td>
<td>San Francisco</td>
<td>California</td>
</tr>
</tbody>
</table>

All the books published by one publisher will now point to a single record in PUBLISHERS. This solves the problem of storing the address many times, and the consequent update anomalies and also the data consistency errors caused by inaccurate multiple entries.

Third normal form removes all columns that are interdependent. In the PUBLISHERS table, this means the address columns: the street exists in only one city, and the city can be in only one state; one column should do, not three. This could be achieved by adding an address code, pointing to a separate address table:

PUBLISHERS

<table>
<thead>
<tr>
<th>Publisher</th>
<th>Address Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>McGraw-Hill</td>
<td>123</td>
</tr>
</tbody>
</table>

ADDRESSES

<table>
<thead>
<tr>
<th>Address Code</th>
<th>Street</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Spear Street</td>
<td>San Francisco</td>
<td>California</td>
</tr>
</tbody>
</table>

One characteristic of normalized data that should be emphasized now is the use of primary keys and foreign keys. A primary key is the unique identifier of a row in a table, either one column or a concatenation of several columns (known as a composite key). Every table should have a primary key defined.

Note that the Oracle database deviates from this standard: it is possible to define tables without a primary key—though this is usually not a good idea, and some other RDBMSs do not permit it.

A foreign key is a column (or a concatenation of several columns) that can be used to identify a related row in another table. A foreign key in one table will match a primary key in another table. This is the basis of the many-to-one relationship. A many-to-one relationship is a connection between two tables, where many rows in one table refer to a single row in another table. This is sometimes called a parent-child relationship.
Chapter 1: Introduction to the Oracle Server Technologies

relationship: one parent can have many children. In the books example so far, the keys are as follows:

<table>
<thead>
<tr>
<th>Table</th>
<th>Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOKS</td>
<td>Primary key: ISBN</td>
</tr>
<tr>
<td></td>
<td>Foreign key: Publisher</td>
</tr>
<tr>
<td>AUTHORS</td>
<td>Primary key: Name + ISBN</td>
</tr>
<tr>
<td></td>
<td>Foreign key: ISBN</td>
</tr>
<tr>
<td>PUBLISHERS</td>
<td>Primary key: Publisher</td>
</tr>
<tr>
<td></td>
<td>Foreign key: Address code</td>
</tr>
<tr>
<td>ADDRESSES</td>
<td>Primary key: Address code</td>
</tr>
</tbody>
</table>

These keys define relationships such as that one book can have several authors. There are various standards for documenting normalized data structures, developed by different organizations as structured formal methods. Generally speaking, it really doesn’t matter which method one uses as long everyone reading the documents understands it. Part of the documentation will always include a listing of the attributes that make up each entity (also known as the columns that make up each table) and an entity-relationship diagram representing graphically the foreign-to-primary key connections. A widely used standard is that primary keys columns should be identified with a hash (#); foreign key columns with a backslash (\); mandatory columns (that cannot be left empty) with an asterisk (*); optional columns with a lowercase o. The books tables can now be described as follows:

Table BOOKS

<table>
<thead>
<tr>
<th></th>
<th>ISBN</th>
<th>Primary key, required</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Title</td>
<td>Optional</td>
</tr>
<tr>
<td>o</td>
<td>Publisher</td>
<td>Foreign key, link to the PUBLISHERS table</td>
</tr>
</tbody>
</table>

Table AUTHORS

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Together with the ISBN, the primary key</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>ISBN</td>
<td>Part of the primary key, and a foreign key to the BOOKS table</td>
</tr>
<tr>
<td>#o</td>
<td>ISBN</td>
<td>Optional, because some authors may not yet be published</td>
</tr>
</tbody>
</table>
Table PUBLISHERS

<table>
<thead>
<tr>
<th>#</th>
<th>Publisher</th>
<th>Primary key</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Address code</td>
<td>foreign key, link to the ADDRESSES table</td>
</tr>
</tbody>
</table>

Table ADDRESSES

<table>
<thead>
<tr>
<th>#</th>
<th>Address code</th>
<th>Primary key</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Street</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>City</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>State</td>
<td></td>
</tr>
</tbody>
</table>

The second necessary part of documenting the normalized data model is the entity-relationship diagram (ERD). This represents the connections between the tables graphically. There are different standards for these; Figure 1-1 shows the entity-relationship diagram for the books example using a very simple notation limited to showing the direction of the one-to-many relationships. It can be seen that one BOOK can have multiple AUTHORS, one PUBLISHER can publish many books, and so on. More complex notations can be used to show whether the link is required or optional, information which will match that given in the table columns listings previously.

This is a very simple example of normalization and is not in fact complete. If one author were to write several books, this would require multiple values in the ISBN column of the AUTHORS table. That would be a repeating group, which would have to be removed because repeating groups break the rule for first normal form. A major exercise with data normalization is ensuring that the structures can handle all possibilities. Tables in a real-world application may have hundreds of columns and dozens of foreign keys.

Errors in relational analysis can be disastrous for an application. It is very difficult (and expensive) to correct any errors later. By contrast, errors made during the programming stage of development can usually be fixed comparatively quickly and cheaply.
CERTIFICATION OBJECTIVE 1.03

Understand the SQL Language

SQL is defined, developed, and controlled by international bodies. Oracle Corporation does not have to conform to the SQL standard but chooses to do so. The language itself can be thought as being very simple (there are only sixteen commands), but in practice SQL coding can be phenomenally complicated.

What follow are the sixteen SQL commands, separated into commonly used groups.

The Data Manipulation Language (DML) commands:
- SELECT
- INSERT
- UPDATE
- DELETE
- MERGE

The Data Definition Language (DDL) commands:
- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT

The Data Control Language (DCL) commands:
- GRANT
- REVOKE

The Transaction Control Language (TCL) commands:
- COMMIT
- ROLLBACK
- SAVEPOINT
SQL is a set-oriented language, whereas most 3GLs are procedural languages. Programmers working in procedural languages specify what to do with data, one row at a time. Programmers working in a set-oriented language say what they want to do to a group (a set) of rows, and let the database work out how to do it to however many rows are in the set.

Procedural languages are usually less efficient than set-oriented languages at managing data, both as regards development and as regards execution. A procedural routine for looping through a group of rows and updating them one by one will involve many lines of code, whereas SQL might do the whole operation with one command: programmer productivity increases. During program execution, procedural code gives the database no options; it must run the code as it has been written. With SQL, the programmer states what he/she wants to do, but not how to do it: the database has the freedom to work out how best to carry out the operation. This will usually give better results.

Where SQL fails is that it is purely a data access language. Most applications will need procedural constructs, such as flow control: conditional branching and iteration. They will also usually need screen control, user interface facilities, and variables. SQL has none of these. SQL is a set-oriented language capable of nothing other than data access. For application development, one will therefore need a procedural language that can invoke SQL calls. It is therefore necessary for SQL to work with a procedural language.

Consider an application that prompts a user for a name, retrieves all the people with that name from a table, prompts the user to choose one of them, and then deletes the chosen person. The procedural language will draw a screen and generate a prompt for a name. The user will enter the name. The procedural language will construct a SQL SELECT statement using the name and submit the statement through a database session to the database server for execution. The server will return a set of rows (all the people with that name) to the procedural language, which will format the set for display to the user, and prompt her to choose one (or more) of them. The identifier for the chosen person (or people) will then be used to construct a SQL DELETE statement for the server to execute. If the identifier is a unique identifier (the primary key), then the set of rows to be deleted will be a set of just one row; if the identifier is nonunique, then the set selected for deletion would be larger. The procedural code will know nothing about the likely size of the sets retrieved or deleted, and the programmer will not know whether the row(s) to be deleted were located by scanning the entire table, or by direct access after searching an index.
Chapter 1: Introduction to the Oracle Server Technologies

CERTIFICATION OBJECTIVE 1.04

Appreciate the DBA’s Role

“Database administrator” is often not a very precisely defined job. In many organizations, end users expect the DBA to be an expert on all aspects of the IT environment. If the database crashes, they call the DBA. Fair enough. But they may also call her if the network fails, if the servers crash, or if the application has a bug. Some DBAs believe that end users will call them if the plumbing gets blocked. Perhaps this is because virtually any failure in the IT environment results in end users being unable to use the database, so the DBA is the natural first point of contact.

Apart from these excessive expectations, the DBA is often the only person in the IT management structure who has a holistic view of the environment. Oracle Enterprise Manager Grid Control is a tool specifically intended to aid the DBA with this: it presents a complete picture of the performance and availability of all the databases, application servers, and server machines in the organization. A vital part of the DBA’s job is to identify where in a complex information technology environment a failure or a performance issue has occurred, and work with the appropriate support group to fix it. Duties more specifically related to database administration include

- **Sizing applications and server hardware** Accurate forecasts of the necessary main memory, disk space, and CPUs that are needed to ensure that applications will run well without demanding unnecessary resources are an important part of maintaining performance without using excessive budget.

- **Oracle software installation and maintenance** This is a non-trivial task in organizations with many servers. Software installations must be kept up-to-date with critical patches (for security, for instance), and maintenance patches should be applied as they are issued, but before any such update is done to live systems, the DBA must ensure that it is adequately tested.

- **Database physical design** There will usually be many ways to configure the physical storage of a database, some of which may have a large impact on the performance of the system and its manageability. The DBA must also be aware of the impact of different storage structures on devices such as disc and tape systems.

- **Monitoring and tuning performance** This is a continuous activity for production systems. A good DBA will be able to anticipate performance issues and fix them before they arise.
Appreciate the DBA’s Role

- Assisting developers with application design and tuning SQL  Some DBAs spend eight hours a day tuning SQL. Perhaps this should be the work of the programmers, but at the very least the DBA must identify the problem areas they should be addressing.

- Liaising with vendors, end users, developers, senior management, and other support groups  As the technician with the most complete picture of the environment, the DBA must take a leading role in coordinating planning and action by all parties involved in the IT environment.

- Backup, restore, and recovery  Perhaps the most important part of the job. The DBA must establish routines that will ensure that agreed targets for uptime and data loss (perhaps as demanding as 100 percent and zero respectively) can be met in the face of any possible problem. There is no right or wrong here, only conformance (or lack thereof) to the agreed targets.

- User and security management  Another critical part of the job. As with uptime and data loss, there is no right and wrong in security—only conformance with agreed standards. The DBA must set up procedures that will ensure conformance, and monitor their effectiveness.

Some DBAs believe that they are doing their job perfectly if no one knows they are there. There is a certain amount of truth in this. Database administration is to a large extent support work, and if the work is done well enough with sufficient proactive planning and preventive maintenance, there will never be a reason for users to report a problem. Usually this ideal can’t be reached, and a large amount of time will be spent working reactively with different people in different groups to solve issues.

The wide scope of the DBA role requires continual study and personal development, study of the Oracle database itself and also of related technologies. It also requires the inclination to educate and to spread knowledge. This can be the most rewarding part of the job.

CERTIFICATION SUMMARY

This chapter summarized some of the knowledge that is assumed before beginning to study for the OCP examinations: the position of the Oracle database in the Oracle product family, the principles of relational databases and data normalization, the SQL language, and other application development tools. Finally, it considers the DBA’s role in the IT environment. A very extensive role indeed.
Chapter 1: Introduction to the Oracle Server Technologies

TWO-MINUTE DRILL

Position the Oracle Product Family
- The Oracle database stores and manages access to user data.
- The Oracle Application Server runs applications that connect users to the database.
- Oracle Enterprise Manager is a tool for managing databases, application servers, and if desired, the entire computing environment.
- Languages built into the database for application development are SQL, PL/SQL, and Java.

Explain Relational Structures
- Data must be normalized into two-dimensional tables.
- Tables are linked through primary and foreign keys.
- Entity-relationship diagrams represent the tables graphically.

Understand the SQL Language
- SQL is a set-oriented language.
- The DML commands are SELECT, INSERT, UPDATE, DELETE, and MERGE.
- The DDL commands are CREATE, ALTER, DROP, RENAME, TRUNCATE, and COMMENT.
- The DCL commands are GRANT and REVOKE.
- The TCL commands are COMMIT, ROLLBACK, and SAVEPOINT.

Appreciate the DBA’s Role
- Sizing applications and server hardware.
- Oracle software installation and maintenance.
- Database physical design.
- Monitoring and tuning performance.
- Assisting developers with application design and tuning SQL.
- Liaising with vendors, end users, developers, senior management, and other support groups.
- Backup, restore, and recovery.
- User and security management.
SELF TEST

Position the Oracle Product Family
1. Which of these languages can run within the database? (Choose all correct answers.)
   A. C
   B. Java
   C. PL/SQL
   D. SQL
   E. Any other language, if it is linked with the OCI libraries
2. In a web application, on which tier does the application software run? (Choose the best answer.)
   A. In the web browser on the client tier
   B. On the middle tier
   C. Within a Java-enabled database

Explain Relational Structures
3. For what data storage paradigm must data be normalized? (Choose the best answer.)
   A. Hierarchical databases
   B. Network databases
   C. Object-oriented databases
   D. Relational databases
4. What type of relationship should be avoided when normalizing data? (Choose the best answer.)
   A. One-to-many
   B. Many-to-one
   C. Many-to-many
   D. One-to-one

Understand the SQL Language
5. SQL cannot do everything. What functions require another language? (Choose all correct answers.)
   A. User interface design
   B. Branching structures such as IF...THEN...ELSE
   C. Operations that affect many rows at once
   D. Table creation and deletion
6. Which of these is not a SQL command? (Choose the best answer.)
   A. MERGE
   B. UPSERT
   C. COMMENT
   D. SAVEPOINT
   E. All the above are SQL commands

Appreciate the DBA’s Role

7. With regard to ensuring that data will not be lost and that security will not be compromised, what would not normally be part of the DBA’s duties? (Choose the best answers.)
   A. Designing backup routines
   B. Setting up disk mirroring
   C. Creating and dropping database objects
   D. Testing restore and recovery strategies

LAB QUESTION

A novice DBA needs a system on which to study the Oracle database, and to practice the skills needed to reach the OCA level of competence. Ideally, he/she will have

- A machine with a graphics monitor, 1 GB RAM, and 5 GB free disk space
- The Database 11g installation software, on DVD or downloaded
- The SQL Developer software
- An IP address: either fixed, DHCP, or loopback
- An operating system such as Windows XP or Linux
- Access to the Oracle database documentation
- If permitted, an account on Metalink

Assemble these items.
SELF TEST ANSWERS

Position the Oracle Product Family

1. ☑️ B, C, and D. SQL, PL/SQL, and Java are implemented internally to the database.
   ✗️ A is wrong because C is not available internally (though with OCI, it can of course be used for writing user processes). E is wrong because OCI is used for writing user processes that run externally to the database.

2. ☑️ B. In a three-tier web application, the application software resides on the middle tier.
   ✗️ A is wrong because the client tier manages only the user interface. C is wrong because the database tier manages only the data access.

Explain Relational Structures

3. ☑️ D. Normalization is the process of converting data to two-dimensional relational tables.
   ✗️ A, B, and C. Hierarchical, network, and object-oriented databases do not implement normalized data structures.

4. ☑️ C. Many-to-many relationships should be resolved into one-to-many relationships, by inserting another entity between the two.
   ✗️ A, B, and D. These are all acceptable in the third normal form model.

Understand the SQL Language

5. ☑️ A and B. SQL has no facilities for user interface design or for flow control.
   ✗️ C is wrong because operations such as this are inherent in SQL’s set orientation. D is wrong because DDL commands are a part of SQL.

6. ☑️ B. The UPSERT command does not exist—though some developers refer to a MERGE operation as an UPSERT
   ✗️ A, C, D, and E. MERGE, COMMENT, and SAVEPOINT are all SQL commands.

Appreciate the DBA’s Role

7. ☑️ B. Disk administration is usually the responsibility of the system administrators, not the database administrator.
   ✗️ A, C, and D. These are typically all part of the DBA’s work.

LAB ANSWER

Having assembled the necessary items, you are ready to proceed with your studies. Enjoy.