Understanding and Deploying LDAP Directory Services
- 1 -
Directory Services Overview

The fact that you have picked up this book and started to read it suggests that you have some idea what a directory service is and what it can do for you. This chapter assumes you have an everyday understanding of directories and expands on that notion to answer three simple but important questions:

- **What is a directory?** In brief, a directory is a specialized database. In this chapter you’ll learn what makes a directory specialized, what separates it from a traditional database, the defining characteristics of a directory, and why they are important.

- **What can a directory do for you?** Directories can do many things, and you probably chose this book with some particular set of problems in mind that you’d like a directory to help you solve. We’ll take you through the basic uses of a directory, many of which may have already occurred to you, as well as covering some more-advanced uses that may be new to you.

- **What isn’t a directory?** The answer to this question is sometimes even more important when defining a successful directory environment than learning what a directory is. In this chapter you’ll learn what separates a directory from a file system, a Web server, and other things you have deployed on your network. The distinctions drawn here are crucial to narrowing the task of designing your directory service.

This chapter aims to answer each of these questions in detail, formalizing the answers to give you a common understanding of the task before you: designing a directory service. You’ll learn why directories are important, the scope of a directory solution, and what they can do for you. Armed with this knowledge, you’ll be ready to read the rest of the book, which deals with the details of understanding, designing, deploying, maintaining, and finally making use of your very own directory service.

**Directory Service Defined**
We will use many terms throughout this book that may be new to you. A directory service is the collection of software, hardware, processes, policies, and administrative procedures involved in making the information in your directory available to the users of your directory. Your directory service includes at least the following components:

- Information contained in the directory
- Software servers holding this information
o Software clients acting on behalf of users or other entities accessing this information

o The hardware on which these clients and servers run

o The supporting software, such as operating systems and device drivers

o The network infrastructure connecting clients to servers and servers to each other

o The policies governing who can access and update the directory, what can be stored in it, and so on

o The procedures by which the directory service is maintained and monitored

o The software used to maintain and monitor the directory service

As you can see, it’s quite a list! Some of these components are depicted in Figure 1.1. Generally, we will use the term directory as a synonym for directory service. It’s important to keep in mind that your directory is a sophisticated system of components that work together to provide a service. Concentrating exclusively on one set of components without thinking about the others is sure to lead to trouble.

**FIGURE 1.1 Directory system components.**

**What Is a Directory?**

Most people are familiar with various kinds of directories, whether they realize it or not. Directories are part of our everyday lives. Everyday examples of directories we encounter include the phone book and yellow pages, *TV Guide*, shopping catalogs, the library card catalog, and others. We refer to these directories as *everyday directories*, or sometimes *offline directories*.

Using these examples as a guide, it’s clear that directories help people find things by describing and organizing the items to be found. Information in such directories ranges from phone numbers to television shows, from consumer goods to reference material, and more.

Directories in the computer and networking world are similar in many ways, but with some important differences. We call these directories *online directories*. Online directories differ from offline directories in the following ways:

- Online directories are *dynamic*.

- Online directories are *flexible*.

- Online directories can be made *secure*.

- Online directories can be *personalized*.

These differences are explored in the sections that follow. It’s also important to understand that there
are different kinds of directories. We expand on this notion more in Chapter 2, "A Brief History of Directories." We’ll give a brief categorization here in order to frame the rest of our discussion. We divide directories into the following categories:

- **NOS-based directories.** Directories such as Novell’s NDS, Microsoft’s Active Directory, and Banyan’s StreetTalk Directory are based on a network operating system (NOS). NOS-based directories such as these are developed specifically to serve the needs of a network operating system.

- **Application-specific directories.** These directories come bundled with or embedded into an application. Examples are the Lotus Notes name and address book, the Microsoft Exchange directory, and Novell’s GroupWise directory.

- **Purpose-specific directories.** These directories are not tied to an application, but are designed for a narrowly defined purpose and are not extensible. An example is the Internet’s Domain Name System (DNS).

- **General-purpose, standards-based directories.** These directories are developed to serve the needs of a wide variety of applications. Examples include the LDAP directories we focus on in this book and X.500-based directories.

In this chapter we will make reference to all four types of directories. Our focus is squarely on the general-purpose type of directory, however.

**Directories Are Dynamic**

The everyday directories you are familiar with are relatively static; that is, they do not change very often. For example, the phone book comes once a year; you have to call information to get more up-to-date information. A new *TV Guide* is produced every week, but still your favorite show is preempted without notice more often than you’d like. The shopping catalogs you receive in the mail are updated only several times a year, at most; also, they do not contain such useful information as which items are in stock in which colors and sizes. Why? Because that information changes so often that by the time the catalog got to you, it would be out-of-date.

By contrast, online directories have the capacity to be kept much more up-to-date. This feature is not always used, of course. Directories are usually only as up-to-date as their administrators choose to keep them. Sometimes administrative procedures are put in place to update the directory automatically. Often, online directories are much better if they are their own ultimate authority for the information they hold. As soon as information changes, it can be updated in the directory and made available to users.

It’s easy to see how this online update capability can be used to make directories more accurate, resulting in a more useful directory. This kind of improvement is incremental. But online updates have the potential to produce more revolutionary improvements, too. These improvements open the door to brand new directory applications that have no offline analogy.

For example, consider a directory that contains up-to-date information on who’s employed at your organization. Such a directory could be consulted by an automated card reader to authorize access to buildings and rooms at your company. In this case, access could be revoked easily and instantly,
simply by making a change to the directory.

As another example, consider a directory containing location information that is updated as you move from office to office, from hotel room to hotel room, and to other locations. This directory could be consulted to route your phone calls, faxes, and messages to you wherever you are. Traditional paper directories could never be used for such a purpose. However, the very nature of this application requires very frequent updates of the information.

This superior update capacity of online directories not only tends to keep information more up-to-date, it also can be used to distribute the update responsibility. The closer information is to its source, the more accurate and timely the information is likely to be. There are at least three reasons for this:

- The source of the information is, by definition, the most accurate.
- Extra delay and opportunity for error between the source and the directory are eliminated if the source makes the update itself.
- Depending on the information and the application, the source is likely to be the party most motivated to maintain the information correctly.

To illustrate, consider the location directory example described previously. The source is the user (you) and the information is your current location. Who knows better than you where you are? (One would hope you know that best!) Which is the more accurate path for an update to be received on: directly from you or from your administrative assistant (your typing skills notwithstanding)? Suppose the update came from a directory administrator typing in information reported by your assistant relayed from you? At each step, opportunity for error is introduced, and the accuracy is further decreased. Finally, who is most motivated to have accurate information about you in the directory? Again, it is likely to be you, the source, because you do not get your phone calls, faxes, and mail unless the information is accurate. Of course, this example assumes that you are responsible enough to want the information to be accurate and that you have the tools and expertise to make it happen.

**Directories Are Flexible**

Another important difference between static, everyday directories and online directories is that online directories offer far greater flexibility. This flexibility has two aspects:

- Online directories are flexible in the types of information they can store.
- Online directories are flexible in the ways that information can be organized and searched.

**Flexible Content**

Offline directories are static in terms of their content. By that we mean that offline directories contain a very restricted and seldom extended set of information. For example, if you wanted to know something beyond the phone number, address, and name information provided by your phone book, you are probably out of luck. But there is a whole host of other useful information you might like to have. Fax number, mobile phone number, pager number, email address, even a picture or short biographical sketch, to name a few, are all items in the same category as the traditional phone
information. But these items are seldom, if ever, included.

By contrast, online directories can easily be extended with new types of information. The cost of additions like these are huge with printed directories but relatively small with online directories. A printed directory would need to be redesigned, reprinted, and redistributed. The cost of this is enormous. The cost of printing the previous directory cannot be leveraged much at all.

Online directories, however, are typically designed to be extended without a redesign. There is no need for reprinting because changes are reflected automatically and immediately. Nor is there a need to redistribute the directory because clients access the directory online and do not keep their own copy. Some clients may cache or replicate portions of the data, but these copies can be updated automatically.

Extending a printed directory in this way is usually done only if a large majority of the users of the directory is clamoring for the information. This is the case because of simple economic and practical reasons. First, as a producer of a printed directory, you could not afford to double or triple the size of your directory to include more information without a compelling reason; doing so would double or triple your cost in producing the directory. Also, from a practical standpoint, the directory itself could become unwieldy and inconvenient for the very customers you are trying to serve.

An online directory, on the other hand, can be extended without incurring such costs. Adding a new data item used by only a small proportion of your users suddenly becomes cost-effective. The cost is incremental to the cost of providing the basic service. It may only involve adding some more disk space to your system and marginally increasing backup time, management, and support costs. No inconvenience is experienced by users of your service, however, because they need not even see the additional information. Those customers who want the new information can easily get it. An economic incentive exists as well: You could charge extra for these premium directory services.

Flexible Organization

The second way online directories provide more flexibility is in how they let you organize your data. Let’s continue with our phone book example. The phone book contains name, phone number, and address information, organized to facilitate searching by name. If you wanted to search by phone number or by address, you would find it difficult, to say the least.

Other specialized directories that are organized to facilitate these kinds of searches may exist, but there is no guarantee of consistency with differently organized directories. Your phone book organized by name might be more or less up-to-date than your special phone book organized by phone number. Such directories contain duplicate information, which often leads to inconsistencies and out-of-date information. Also, such directories are usually not readily available, and they are usually expensive. The types of data organization that can be supported are limited. They are also limited by the nature of the medium on which the directories are distributed (e.g., paper) and by the capabilities of their end users (people without specialized training, perhaps).

By contrast, online directories can support several kinds of data organization simultaneously. The online analogy to your printed phone book can easily let you search by name, phone number, address, or other information. Furthermore, online directories can provide more-advanced types of searches that would be difficult or impossible to provide in printed form.
For example, if you are not sure of the spelling of a name, an online directory can let you search for names that sound like the one you provide. It can also provide searches based on common misspellings, substrings of names, and other variations. These different kinds of searches can be performed simultaneously or in some defined order (for example, an exact search first, then a sounds-like search, then a substring search, and so on) until a match is found. This kind of power in searching is key to providing users with the kind of "do what I mean" behavior they often desire.

**Directories Can Be Secure**

Offline directories offer little, if any, security. The phone book, for example, is public. Your company’s printed internal phone book may have "do not distribute outside the company" stamped on it in big red letters, but this kind of security is advisory at best. This lack of security reduces the number of applications that can be served by an offline directory. It also forces users to make difficult choices, if any choice is available to them at all. Most people are familiar with unlisted phone numbers, a service most phone companies offer for a premium fee. Opting out of the directory makes your number unavailable to telemarketers and other annoying callers. However, it also makes your number unavailable to people you probably want to have it.

The root of this problem is the lack of any security in an offline directory. Its information is accessible to anybody with access to the directory, or information can be left out of the directory and accessible to nobody. This is a natural consequence of the methods used to distribute and access offline directories. Distribution is often very wide, and everybody gets his or her own copy. The access method consists of flipping through pages or calling a public number, such as 411. None of these methods provide any way of determining who is accessing the directory and, therefore, what information they should have access to.

Online directories can solve these problems. Online directories centralize information, allowing access to that information to be controlled. Clients accessing the directory can be identified through a process called **authentication**. The directory can use the identity established in conjunction with access control lists (ACLs) and other information to make decisions about which clients have access to what information in the directory.

Returning to our phone book analogy, consider how security features such as ACLs would change the situation. You could be listed in the directory, but your information would be accessible only to a subset of directory clients. You might be able to specify this subset as a list of friends. You might be able to specify it via some criteria, such as "anyone who lives on my block." You could allow your information to be available to everyone except a list of people you specify. The possibilities go on, and the results are quite powerful.

It’s important to realize that even this level of powerful and flexible security is not a panacea. For example, ACLs can be effectively, if somewhat awkwardly, defeated by a trusted user copying confidential information off of his or her screen and distributing it outside the company. Still, online directories have security capabilities that are far more advanced than those of offline directories.

**Directories Can Be Personalized**

Another difference between printed directories and online directories is the degree to which each can be personalized. There are two aspects to this personalization:
- Personalized delivery of service to users of the directory
- Personalized treatment of information contained in the directory

*TV Guide* and the phone book are personalized on a regional basis. But everyone gets the same LL Bean catalog and accesses the same card catalog at the library. Furthermore, everyone within the same region gets the same phone book or *TV Guide*. It would be nice to get catalogs tailored to your specific interests, a phone book organized to do searches in the way you prefer, or a card catalog that remembers the kinds of books you like. This is the first aspect of personalization: the ability to deliver information tailored to your needs as an information consumer.

The second aspect of personalization concerns your ability to determine who has access to information about you and other things. This is your ability to tailor the directory to your needs as an information provider. In offline directories, as we saw previously, you have only two broad choices about the accessibility of directory information about yourself: You can either be included in the directory or not—with no in-between. Furthermore, many directories do not even provide you with this choice. Trying to get yourself unlisted can be a frustrating and time-consuming experience.

Online directories offer both of these features. The mechanism for doing so is rooted in the directory security capabilities described previously. By identifying users who access the directory and storing profile information about them, an online directory can easily provide personalized views of the directory to different users. For example, an online catalog can show you the types of products you are most likely to be interested in. This personalized service could be based on interests explicitly declared by you. It could also be based on your previous interactions with the service.

From a user’s perspective, personalization of this kind is great because it gives the user a more desirable service. The user does not need to wade through information that is of less interest just to get to the information the user does consider interesting. From a service provider’s perspective, personalization of this kind is great because it provides a more desirable service to the service provider’s users. It also allows the service provider to better target all kinds of special services. For example, the service provider can provide information about promotions and sales, new product offerings, and advertisements, all tailored to a user’s preferences.

**Directory Described**

So far we’ve been relying primarily on a common-sense understanding of the word *directory* in our discussion. We’ve used everyday printed directories that you are probably familiar with to explain what online directories are and how they differ from those offline. Now it’s time to glean from our previous discussion the defining characteristics of online directories. The definition we will give is not a formal or mathematical one. Instead, we will expound on a list of characteristics that online directories share.

**Design Center Defined**

We use the term *design center* to refer to the defining set of assumptions, constraints, or criteria driving the design or implementation of a system. When designing or implementing a system, you have to make all kinds of decisions about what’s important, what’s not, what the system must do well, and what it can afford to do less well. A system’s design center is an expression of the focus the designer or implementer had...
when making these decisions. Design center is a concept that applies to software and other systems and products as well.

For example, suppose you were going to design and implement a vehicle for yourself. Aside from needing a few common characteristics that essentially boil down to a wheeled, motorized conveyance, you have a lot of flexibility. A designer who has a large family might design a station wagon or van. His design center might be focused on large passenger capacity. Another designer with a lot of stuff to haul around might design a truck. Her design center might be focused on cargo capacity. Another driving enthusiast designer might focus on performance.

Software and service design centers work in similar ways with the following questions. Does the software system or service need to serve a large community or a small one? Is the community technically sophisticated or inexperienced? Is performance a critical feature of the system? Is security? The answers to these questions and others drive the focus of the design and implementation efforts and ultimately determine the character of the system.

A directory can be thought of as a specialized database. It is interesting to compare databases and directories, because the differences have more to do with environment and design center than with anything fundamental. The comparison is also interesting because most people generally have a better understanding of what a database is and does than they do a directory. The differences between a database and a directory fall into the following broad categories:

- **Read-to-write ratio.** Directories typically have a higher read-to-write ratio than databases.
- **Extensibility.** Directories are typically more easily extended than databases.
- **Distribution scale.** Directories are usually more widely distributed than databases.
- **Replication scale.** Directories are often replicated on a higher scale than databases.
- **Performance.** Directories usually have very different performance characteristics than databases.
- **Standards.** Support for standards is important in directories, less so in databases.

Each of these points is explained in the remainder of this section.

**Read-to-Write Ratio**

One defining characteristic of a directory is that it is typically read or searched far more often than it is written or updated. This is quite often not the case with a database. A database might be used to record transaction information or other auditing data that is read only under exceptional, or at least infrequent, conditions. For example, such data might be read only once a month to produce a summary report, or once a year when an internal audit is conducted.

Information in a directory, on the other hand, is usually read many more times than it is written. In fact, it is not unusual for a piece of directory information to be read 1,000 to 10,000 times more often than it is written. If you think about the types of information usually stored in a directory, this makes sense. Information about people, for example, changes relatively infrequently, especially compared to
the number of times the information needs to be accessed. How often do you change phone numbers compared with the number of times somebody calls you? How often do you change addresses compared with the number of times you receive mail?

Data with this "often read, seldom written" characteristic is not restricted to information about people. Catalog data, most location information, configuration information, network routing information, reference information, and many other types of information are all read far more often than they are written. The domain of applications that can be served by a directory is quite large. For some applications, the information is never updated online; instead, it is updated only periodically via some batch process initiated by an administrator.

Why is this characteristic important? It sets a design center for directory implementations. Implementers can make important, simplifying design decisions based on this characteristic. Directory implementations can be highly optimized for the types of operations that will be performed most often. If one operation is performed 10,000 times more often than another, it’s a good idea to spend more time making that operation perform quickly. Contrast this with databases, which must be optimized for write and read operations. This kind of optimization has implications on other directory features--such as replication--which we will discuss later.

**Information Extensibility**

Another important, defining characteristic of a directory is that it supports information extensibility. The term *directory schema* refers to the types of information that can be stored, the rules that information must obey, and the way that information behaves.

Directories are not limited to a fixed set of schema that can be stored and retrieved. This information can be extended in response to new needs and new applications. A directory usually comes with a useful set of predefined types of information that can be stored, but many installations have special requirements that dictate the extension of this predefined set. Your organization may have special attributes you want to store, including, for example, employee status for people or the building location code for a printer. Sometimes these new attributes may even define new kinds of behavior from an existing attribute.

Although databases are used to store many kinds of information organized in all kinds of ways, they are usually constrained in the types of information that can be stored. It is rare to find a database that allows you to introduce a new, primitive data type with new semantics.

**Data Distribution**

Distribution of data is another area in which directories differ from databases. Data distribution refers to the placement of information in servers throughout your network. Data can be centralized in a single server, as shown in Figure 1.2, or data can be distributed among several servers, as shown in Figure 1.3.

**FIGURE 1.2** Centralized directory data held in a single server.

**FIGURE 1.3** Distributed directory data held in three servers.

Although you can find databases that allow limited distribution of data, the scale of the distribution is
quite different. The typical relational database allows you to store one table over here and another table over there. This distribution is usually limited to a few sites. The ability to make queries that involve both of these sites exists, but performance is often a problem. This causes the distribution features to be rarely used.

Data distribution is a fundamental factor in the design of directories. Part of the directory’s purpose is to allow data to be distributed across different parts of your network. This capability is aimed at addressing environments where authority and administration must be distributed. An example of an organization needing this kind of distribution is one with offices in several countries around the world. Each office wants to have authority over its own directory; thus, the country-specific directories must appear to the outside world as a single, logical directory for the organization as a whole.

Another example in which data distribution is important is in support of large-scale directories. As your directory gets bigger, at some point the tactic of buying a bigger server with more disk and memory and CPU horsepower produces diminishing returns.

A better approach may be to construct your directory from a set of smaller machines that work together to provide the overall service. This solution is cheaper in many cases. It has the advantage of harnessing the parallel processing power of all the machines holding the directory. It also has certain attractive practical implications on the performance of some system administration functions, such as performing backups, recovering from disasters, and so on. Consider a directory distributed across ten small machines: Backing up or recovering one of the small machines is easier than backing up or recovering a single large machine.

**Data Replication**

Closely related to data distribution is the topic of replication. Replication is the process of maintaining multiple copies of directory data at different locations. There are a number of reasons to do this:

- **Reliability.** In case one copy of the directory is down, others can be accessed.
- **Availability.** Clients are more likely to find an available replica, even if part of the network has failed.
- **Locality.** Clients get better and more reliable performance from a directory the closer they are to it.
- **Performance.** More queries can be handled as additional replicas are added.

More-detailed information on replication can be found in Chapter 10, "Replication Design."

Databases sometimes support replication, but typically they do so only for a very small number of replicas. Historically, performance has been a big problem with database replication implementations. This is partly because database replication is almost always strongly consistent; that is, all copies of the data must be in sync at all times.

Directory replication, on the other hand, is almost always loosely consistent. This means that
temporary inconsistencies in the data contained in different replicas are acceptable. This characteristic has important implications for the number of replicas that directories can support and the physical distribution of those replicas across the network.

As we shall see later, performance is an important directory characteristic. One good way of helping to ensure great performance is to make sure that each user of the directory has a copy of it close by. There are two reasons for this:

- Moving directory data close to the clients accessing it cuts down on the network latency of directory requests.
- The total number of directory queries processed by the system as a whole can be increased. As the number of replicas increases, so does the number of queries that can be handled. If one directory server can handle a million queries per day, adding another server could increase the capacity of the system to two million queries per day.

Availability of the directory is also a key factor. Directories tend to be used by many different applications for such fundamental purposes as authentication, access control, and configuration management. The directory must always be available to these applications if they are to function at all.

It is important to note that availability is not the same thing as reliability. A reliable directory may have redundant hardware and an uninterruptible power source. Such a directory may almost never go down, but that does not mean that it is always available to the clients that need to access it. For example, entire networks between clients and servers might go down. From the client’s perspective, this causes the same problem as the directory going down.

You could try to solve this problem by building into your network the same kind of hardware reliability that is available for servers. Redundancy, uninterruptible power, and other techniques are all valuable, although not always practical. The other approach is to replicate your directory data to bring the data closer to the clients needing access to it. This helps to mitigate network problems that might otherwise prevent clients from accessing the directory. A sample unreplicated scenario is shown in Figure 1.4, and a sample replicated scenario is shown in Figure 1.5.

**FIGURE 1.4** An unreplicated directory service with data held by only one server.

**FIGURE 1.5** A replicated directory service with data held by three servers.

There are several implications of these facts on directory replication. Directories are replicated on a far greater scale than databases. It is not unusual for a directory replica to be maintained on each subnet in your network to minimize latency and increase availability. In some cases, a replica might be maintained on each machine, which can lead to literally hundreds or thousands of replicas. These replicas may be many network hops away from the central directory. They may even be connected over links that are only up intermittently. These kinds of replication requirements set directories apart from databases.

**Performance**

As mentioned previously, high performance is another characteristic that differentiates directories
from databases. Database performance is typically measured in terms of the number of transactions that can be handled per second. This is also an important measure of directory performance, but the requirements on a directory are far more stringent than on most database systems.

A typical large database system might handle hundreds of transactions every second. The aggregate directory performance required by a typical large directory system may be thousands or tens of thousands of queries per second. These queries are usually simpler than the complex transactions handled by databases. As described earlier, the read-to-write ratio is typically much higher on a directory than on a database. Therefore, update performance is not as critical for directories as for databases. As we shall see later, though, it is important nonetheless.

Some of the directory’s increased performance requirements are caused by the wide variety of applications that use the directory. Whereas a database may be designed and deployed with a single or a small set of driving applications in mind, directories are often deployed as an infrastructure component that will be used by an unknown but continually increasing number of applications developed across your company, and even across the Internet at large. Access to the directory is distributed, as is the development of the applications causing this access. This means that you, as the directory administrator, often do not have control over the kinds of queries your directory must answer. Therefore, it is important that your directory be flexible and capable of good performance regardless of the types of queries it must respond to.

Another root of directory performance requirements is the types of applications that typically access the directory. Applications access the directory for many different purposes. If your directory is used by your email software to route email, for example, one or more directory lookups are required for each piece of mail. Depending on the volume of mail your site processes, this can be a significant load on the directory.

There are many more examples that require high performance. If your directory is used by Web application software as an authentication database, it is accessed each time a user launches a new application. If your directory is used by these applications to store user preference and other information needed to provide location independence, even more directory accesses are called for. If your directory is used to store configuration and access control information for your Web, mail, and other servers, there is a potential directory access each time those services are accessed by clients. If you have a large user population, this quickly adds up to a lot of traffic. In these environments, using directory locality to minimize network latency is critical to providing adequate performance.

As you can see, directories are at the center of a lot of things that cause performance requirements to increase quickly. Of course, client-side caching can and should be used to minimize the number of times the directory itself is accessed, but even these techniques can only slow the flow of directory queries. High performance is still one of the most important characteristics of a directory.

Earlier we stated that the read-to-write ratio for directories is very high. The natural conclusion you could draw from this is that write performance is not nearly as important as read and search performance. Although this is true in a way, the scale of data handled by many directories makes write performance an important factor as well. And, as we described earlier, the capacity for online updating is one of the key enablers of some exciting new online directory applications. Clearly, the ability to update is important, and it must function at a certain level of performance.

For example, consider a directory with a million entries. This may seem like a lot, but this is not
unreasonable for a very large corporation (after you’re finished adding entries for all users, groups, network devices, external partners, customers, and other things). If each entry changes only once a month on average, that is a million updates per month, 250,000 updates per week, almost 36,000 updates per day, or around 1,500 updates per hour. That’s quite a few updates! And the peak number of updates that must be handled is much higher because user-initiated changes are usually made during business hours. Administrator-initiated changes may need to be saved up and applied in a batch during limited off-peak hours, further increasing performance requirements.

**Standards and Interoperability**

The last important factor sets directories apart from databases is standards. The database world has various pseudo-standards, from the relational model itself to SQL. These pseudo-standards make it easier to migrate from one database system to another. They also make it so that when you’ve learned the concepts behind one vendor’s system, you can easily apply that knowledge to come up to speed on another’s quickly. These standards do not provide real interoperability, however. In the directory world, because applications from any vendor must be able to use the directory, real interoperable standards are critical.

This is where LDAP comes in. LDAP provides the standard models and protocols used in today’s modern directories. LDAP makes it possible for a client developed by Microsoft to work with a server developed by Netscape, and vice versa. LDAP also makes it possible for you to develop applications that can be used with any directory. In the database world, an Oracle application cannot be used with an Informix database. An Informix application cannot be used with a Sybase database. This kind of interoperability, lacking in databases, is important to directories for two reasons:

- It allows the decoupling of directory clients from directory servers.
- It allows the decoupling of the development process from a decision about a particular directory vendor.

Before LDAP came along, each application that needed a directory usually came with its own directory built right in. This may seem a convenient solution at first glance, but consider what things are like when you’ve installed your 24th application and, therefore, your 24th directory. Each user in your organization who requires access to these applications needs an entry in each directory—a lot of duplicate information to maintain. This is one of the primary sources of headaches for system administrators and increased costs for IT organizations. This situation is illustrated in Figure 1.6.

**FIGURE 1.6 Application-specific directories cause duplicate information and system administration headaches.**

Application developers everywhere can write applications using the standard directory tools of their choice. These applications will run with any LDAP-compliant directory, which essentially turns the directory into a piece of network infrastructure. This dramatically increases the number of applications that can and will be written to take advantage of the directory. It also frees you from having to rely on a single vendor for your directory solution. These same advantages are what drove the success of other Internet protocols, such as HTTP (for the Web), IMAP (for accessing email), and even TCP/IP itself. A standards-based directory infrastructure is illustrated in Figure 1.7.

**FIGURE 1.7 A standards-based, general-purpose application directory eliminates information
Directory Description Summary

Here is a reasonably concise description to summarize a directory: It is a specialized database that is read or searched far more often than it is written to. A directory usually supports storing a wide variety of information and provides a mechanism to extend the types of information that can be stored. Directories can be centralized or distributed. They are often distributed in large scale, both in how and where information is distributed. Directories are usually replicated so that they are highly available to the clients accessing them. The scale of directory replication often involves hundreds, if not thousands, of replicas. Replication also helps increase directory performance, which is important to providing applications with a fast, reliable infrastructure component that can be used with confidence. Finally, with LDAP, directories have become standardized. This allows applications and servers from different vendors to be developed, sold, and deployed independently.

What Can a Directory Do for You?

We’ve already talked about some applications of offline directory services that you deal with every day. Also, we’ve talked about how these same applications can be improved, and sometimes revolutionized, by the timely update capabilities, extensible nature, and personalization possibilities of online directories. Now it’s time to turn our attention fully to this exciting aspect of online directories: the types of new applications that can be developed to take advantage of all the capabilities of the online directory.

Finding Things

As mentioned at the beginning of this chapter, most everyday directories are aimed at the problem of finding things. Finding the right book in the library or the telephone number of your friend or a business you want to contact; the right style, size, and color of shirt you want to order from a catalog; or the time and channel of the television show you want to watch tonight--directories organize information in this way so that it is easy to find what you are looking for. Finding things is also an important application for online directories. And as we will see, online directories have capabilities that allow far more advanced ways of finding things.

Perhaps the most basic and best understood application of an online directory is the online analogy to a phone book. Directories in this category start on one end with the large Internetwide or countrywide directories provided by Internet directory service providers, such as Four11 (www.four11.com), BigFoot (www.bigfoot.com), WhoWhere? (www.whowhere.com), and AnyWho (www.anywho.com). On the other end of the spectrum, you can apply directory technology to create a local phone book for your organization. The service provided by these directories is remarkably similar to the analogous paper directories, with some important differences.

Searching Versus Browsing
Directories usually support both searching and browsing, but you must take into account their differences. Thinking about the offline analogies again, it becomes clear that both searching and browsing are needed. When you are looking for the phone number of Pete’s Pizza delivery service, you know what you want and you want to go right to the answer. Phone
books keep alphabetized listings to make this easy. This is *directory searching*.
On the other hand, when you are in the mood to buy some new clothes, it is seldom the case that you know exactly what you want. You probably have some general characteristics in mind (you know what you like and are in the mood for). It is unlikely that you know the make and model of the shirt you want to buy. You want to flip through the pages of a catalog, looking at pictures of clothes until you come to what you want. This is *browsing*.

Browsing and searching are not mutually exclusive. Often, you perform some kind of search to narrow down the choices, which you then browse. Sometimes the reverse is true: You browse some possibilities to find the place where you want to search. The important point to remember is that browsing and searching are complementary. Some applications require one, some the other. And some applications require both, either used together or at different times.

First, online directories let you greatly increase the scale and coverage of the directory. The BigFoot directory, for example, contains essentially every phone book in North America. Try keeping that many phone books in your house for quick and easy reference! More importantly, online directories allow you to organize information, even in an application as simple as a phone book, in new and exciting ways. For example, you can use your directory to search by name, phone number, address, and other information not even contained in a traditional phone book. You can also perform new kinds of searches in case you are unable to correctly spell the name you are looking for--if you think the phone number you have might be off by a digit, or if you have only part of a name. You can also browse the contents of the directory in different ways.

This flexibility of searching and browsing methods is a direct result of the directory’s ability to organize information in different ways simultaneously. This kind of flexibility is important in a wide variety of applications. Consider, for example, a network printing application. This application allows users to choose the desired attributes of a printer. These attributes might be location, color or black-and-white, print quality, size or type of paper, single- versus double-sided capability, and others. Typical interaction with the user might involve the user selecting the most important attributes (for example, color and print quality). The user can then choose from a list of printers, perhaps choosing one based on location.

As a final example of finding things, consider a client in search of a network application service. This service might be a database service, an employee benefit enrollment service, a stock reporting or trading service, an online dating service, a discipline-specific research service, or just about anything. As long as these services register themselves with the directory, users or other application programs can later query the directory to locate the services. The directory’s online update capability allows users to find the services they need even when those services change locations. The service simply updates the directory with its new location.

**Managing Things**

One application topic that sets online directories apart from their offline counterparts is information management. Online directories provide a big piece of the management puzzle in a distributed environment, providing a central repository for objects that need to be managed. The example cited most often in this category is the expensive task of user and group management--one of the most important management services a directory can provide. A centralized directory can help reduce costs of this service substantially.
The N+1 Directory Problem

Until recently, just about every application you deployed came with its own incompatible directory. Mail applications such as cc:Mail, groupware applications such as Lotus Notes and Microsoft Exchange, and even earlier versions of Netscape’s Commerce Web server all came with their own directory. UNIX, too, was not immune, with applications such as sendmail using the /etc/aliases database. Although this situation is convenient if you are installing only one application, it quickly becomes unmanageable as the number of directories in your environment increases.

Imagine that you have a user needing access to all the applications you have installed on your network. The user must be added to all the application-specific directories. If any information about the user changes, it must be changed in all these directories. This is known as the N+1 directory problem; that is, each new application adds one more directory to manage. This is quite a troublesome and costly problem for administrators everywhere (refer to Figure 1.6).

Consider how this situation changes when you install a centralized directory. In this case, you need to enter or change user information in only one place. As new applications are added to the network, they use the centralized directory instead of their own. Of course, the flaw in this beautiful plan is clear: All applications must be rewritten to use a centralized directory, at least as an option, rather than their own proprietary directories. These changes are taking place, although the pace at which they are occurring may not be as fast as you’d like. The improved situation was depicted in Figure 1.7.

Another advantage of storing user and group information in a centralized directory is enjoyed by application developers. Instead of having to develop a proprietary directory service integrated with each application, application developers can use what’s already provided and focus their activities on making a better application.

Another important application of directories to the management problem relates to the task of configuration management. Historically, the configuration for an application has been kept in a set of files or an OS-specific repository such as the Microsoft Windows Registry. This solution is simple to implement and works just fine for many applications. But network applications often have different needs, especially when deployed in large numbers.

Consider, for example, a network of Web servers. There may be hundreds of them within a large corporation. If they share one or more configuration items, imagine the onerous task of changing the configuration in all of them. With the configuration file approach, you would need to visit each Web server and edit its configuration file. With hundreds of servers, this task would indeed be tedious. You could develop an ad hoc configuration management system based on tools, such as rdist and rsync, but this system itself needs to be managed.

Now consider the implications of these servers storing their configuration information in the centralized directory. First, remote management of the server becomes possible; the server’s configuration can be accessed from anywhere on the network. Second, the configuration can be shared among servers, making cluster management of servers possible. In our earlier example, the hundreds of Web servers could have their configuration changed quickly and easily from a central location. An example of this application is shown in Figure 1.8.

**FIGURE 1.8 Using a directory for centralized configuration management.**
A similar and potentially more powerful application of directory-based management is in the area of user configuration and preferences. Like application configuration, user preferences have historically been maintained in configuration files or local databases, such as the Microsoft Windows Registry, dot files in home directories on UNIX, and Macintosh preferences files. In an environment where centralized user configuration management is important (and there are many such environments), this proves to be a rather inconvenient solution. However, by having applications store and read this information from the directory, thousands of user configurations can be changed at one time. Imagine being able to change a piece of configuration one night, and the next day when your users arrive and start their applications, the configuration is seamlessly changed.

When this approach is combined with the storage of per-user state information in the directory, location independence can be achieved. This is an important feature in many environments. Consider a user who accesses applications from a machine at home and one at work. The user probably has her preferences set up just the way she prefers. Perhaps she has a personal email address book she needs to access from each location, or personal bookmarks in her Web browser. Storing these things in a directory and retrieving them from the network allows the user to have the same environment both at home and at work.

Similar requirements exist in shop floor situations or on university campuses, where kiosk access is provided to users without a dedicated machine of their own. (We use the terms "nomadic" or "roaming" to describe these types of users.) IS professionals and others who find themselves in front of many different machines in the course of a normal day can also benefit from location independence enabled by a directory. This application is illustrated in Figure 1.9.

FIGURE 1.9 Using the directory to provide location independence.

Lightweight Database Applications

In the first section of this chapter we described a directory by comparing it to a database. In the next section we will go into more detail about the differences between the two.

Directories are good at storing and retrieving relatively small pieces of information. The fact that LDAP directories provide a standard protocol and API access to this information is also attractive. Prior to LDAP’s arrival, application developers with a need to maintain a simple database for their application had just a couple choices:

- **Build a proprietary special-purpose database.** This approach means extra work for the application developer and the application administrator who must install and maintain the database. It also means a lot of wasted time solving problems that have already been solved.

- **Embed a commercial database package.** This approach is more attractive because it generally involves less work for the application developer and avoids duplicating existing efforts. However, it can still be inconvenient for users and administrators who have to install and maintain the database. Embedding a full-fledged database such as Oracle is overkill and comes with a high price tag. Other simpler, embedded databases exist, but they are usually proprietary and hard to integrate.

Embedding a simple directory server can often solve the same problems with some distinct
advantages. Directories are generally smaller, less complex applications than full-fledged databases. LDAP directories provide a standard access protocol, meaning that your application can work with other directories, and other applications can work with your directory. This helps to avoid the N+1 directory problem we described earlier.

LDAP directories also provide de facto (soon-to-be-official) standard APIs you can use to access the directory, making your application more portable. This reduces the amount of work required for application programmers to embed directory services in an application.

One example of an embedded database like this can be found in Netscape’s Communicator Web and mail client. The client maintains several local databases containing user bookmarks, local address book information, security information, and other things. These all used to be stored in separate proprietary databases, meaning that the information could not be shared and could not be accessed from anywhere but the local machine. Introducing a directory server to store the information solves both of these problems.

Another example of an embedded database is found in the Netscape Certificate Server. This server maintains a database of certificates it has issued to users. Version 1.0 of the product stored these certificates in an Informix database. This proved to be unwieldy, hard to manage and install, and generally a poor choice. The next version of the Certificate Server used an embedded directory server to manage this database of information. This directory-based solution is much easier to install and manage, easier to develop with, and it is much more flexible.

Security Applications

Another interesting directory application is security. Of particular interest are public key-based security systems. We cover these systems in more detail in Chapter 11, "Privacy and Security Design." For now, it’s enough to know that the public key infrastructure (PKI) required to provide security like this is hard to manage. PKI requires a way to distribute security objects such as certificates to clients and servers, to keep those security objects up-to-date, to revoke them when they are compromised, and to handle other functions. Without a directory, this task is difficult to manage.

Directories help solve two problems that certificates and PKI in general introduce:

- **The PKI life cycle management problem.** This refers to how certificates are created, maintained, and destroyed. Without a directory, this process is up to you to manage. If you are lucky, you might have some proprietary software acting on your behalf that manages the process for you.

- **The certificate location problem.** This refers to how you find the certificates of the people with whom you want to communicate securely, and how people who wish to communicate with you find your certificate. Without a directory, you and your friends might have to resort to calling each other on the phone and reading off strings of hexadecimal (base 16) digits to each other.

Introducing a directory helps solve these two problems. The directory acts as a central point of administration throughout the PKI life cycle. Your various security objects can even live in the directory, where you or an administrator you trust can maintain them. When you need a new certificate, one can easily be issued to you through the directory. When your certificate needs to be revoked, again the directory provides the means to do so quickly and efficiently--and the means by
which other parties can be notified of the change.

The directory also helps locate the certificates of others. In this application, the certificate or other security information is just another attribute of a user’s directory entry. A certificate can be retrieved from the directory as conveniently and efficiently as a name, phone number, or email address. A more detailed and formal description of these problems and processes is presented in Chapter 11.

**What a Directory Is Not**

In our experience, people sometimes have one of two extreme reactions upon learning about the wonders of directories. One reaction is negative: "What good is this directory really going to do? Why can’t I just use X to fill that same need? I don’t see how this directory is going to be as reliable and perform as well as you say, and I am hesitant to make my application depend on it!" This reaction can usually be overcome by additional education, helping the skeptic see a directory in action, and better explaining all the great things a directory can do for the person.

The other reaction is equally misguided, but can be harder to overcome: "This directory is great! I bet there’s nothing it can’t do. I can now use the directory instead of my file system, Web server, FTP server, and a host of other things. Finally, a handy, all-in-one tool that chops, slices, dices, makes homemade desserts, but best of all, it cleans itself!" The enthusiasm of this reaction is to be admired, but there is danger here, too. There is an old saying that states that if the only tool you have is a hammer, every problem begins to look like a nail. Needless to say, not all problems are nails, and trying to treat them as such tends to lead to poor carpentry.

A directory is no different—like any other technology, it is best suited to solving one set of problems. A directory can be usefully applied to many other problems as well, though with somewhat less satisfaction. A directory can be abused by trying to make it solve problems that it was not intended to solve.

In this section we examine this area more closely, explaining some of the many applications for which directories are *not* well suited. We will explain why directories are different from the following network services that may look similar on the surface:

- Databases
- File systems
- Web servers
- FTP servers
- DNS servers

We will explain how your directory can complement all these services and why each one fills a valuable niche in your computing infrastructure. We will conclude this section with a summary of when to use a directory to store information and when to use something else.

**Database Comparison**
In the previous section we described what a directory is, and we spent quite a bit of time comparing directories to databases. We will not repeat that comparison here, but we will highlight a few important points. First, recall that directories are typically read-focused rather than write-focused. So if your application is writing large volumes of data, say for recording logs or merchandise transactions, you should choose a database over a directory.

Second, directories support a relatively simple transaction model. Directory transactions involve only a single operation and a single directory entry. Databases, on the other hand, are designed to handle large and diverse transactions, spanning multiple data items and many operations. If your application requires this kind of heavy-duty transaction support, again a database is a better answer than a directory. On the other hand, if you have an application that does not have these requirements but instead wants to read and occasionally write information to a network-accessible database in simple transactions, a directory server may well be a good, cost-effective, much simpler choice than using something like an Oracle database.

**File System Comparison**

A directory makes a poor file system. Files have characteristics that are different from directory information. Files are often very large, containing many megabytes or even gigabytes of information, whereas directories are optimized for storing and retrieving relatively small pieces of information.

Although there is often no restriction on the size of a directory entry or attribute, the question is one of design center. Some files are written far more often than they are read, such as log files and files used to hold a database. Directories, as you’ll recall, are optimized for read over write. Some files, of course, are read far more often than they are written. Application binaries are a good example of files in this category, but the size of these files is often larger than should be stored in a directory.

Applications often access files in chunks, especially if the file is large. File systems support calls--such as seek(), and read() and write()--can be used to write only a portion of a larger file for this very purpose. Directories do not provide support for this kind of random access. Instead, a directory entry is split up into attributes. You can retrieve attributes separately, but you usually have to retrieve all of any attribute you ask for. There is no way to retrieve only part of a value, starting at some byte offset. File systems, on the other hand, are not good at storing attribute-based information. They do not typically support any kind of searching capability like directories do.

**Web Comparison**

Since the World Wide Web burst onto the computing scene a few years ago, Web servers have become ubiquitous. Chances are good that your organization, depending on its size, runs anywhere from one to several hundred. Web servers have certain similarities to file systems. They are made to serve clients accessing documents or files that can vary greatly in size. Although Web server documents usually share the typical read-many-write-few characteristic of directory information, directories are not well suited to the task of delivering to clients multimegabyte JPEG images or Java applications.

Web servers also often serve as a springboard to a development platform for Web applications. These platforms range from simple CGI (common gateway interface) to more-complex platforms, such as the one found in the Netscape Application Server. Directories typically do not provide this kind of
flexibility in application development, even when they do provide directory application development platform services, such as those provided by the Netscape Directory Server.

Directories are optimized for providing sophisticated searching capabilities of the data they hold. Web servers can be used to develop similar search applications, but such applications are not standards based. Web servers are tuned to providing GUI-style interfaces on applications; they are not tuned to providing generic application access to directory data. If you have a specific database of information you want to make available to users, a Web server might be a good choice. If you have information you want to make available to a wide variety of applications, a directory server is a better choice.

**FTP Comparison**

An argument similar to the previous one for Web servers applies to File Transfer Protocol (FTP) servers. One could argue that the days of FTP servers are numbered now that Web servers have arrived, but they are hardly threatened by the arrival of directory services. Again, the main differentiating factor is the size of the data and the type of client that needs access to it. Another important point is that FTP is a very simple protocol, tuned to do one thing and do it well. If all you want to do is create a means to transfer files from one place to another, the extra directory infrastructure needed to perform replication, searching, updating, and so on is overkill.

On the other hand, if your application involves more than simple retrieval and storage of information, a directory is a more appropriate choice. Unlike directories, FTP provides no capability for search, no attribute-based information model, and no incremental update capability.

**DNS Comparison**

DNS, the Internet’s Domain Name System, translates host names, such as home.netscape.com, into IP addresses. Host names are good for users wanting to remember how to connect to their favorite Internet service. IP addresses are required by the networking infrastructure underlying the Internet that is responsible for actually making the user’s connection happen. The DNS and a typical directory like LDAP have certain similarities, such as providing access to a hierarchical distributed database. But there are some important differences that set the two apart.

DNS is highly optimized for its main purpose; most directories are meant to be more general purpose. The DNS has a specialized, fixed set of schemas; directories allow schemas to be extended. DNS does not allow updates of its information; directories do. DNS can be accessed over the UDP connectionless transport; directories are usually connection-based.

---

**NOTE**

Some recent work on dynamic DNS by the Internet Engineering Task Force (IETF) is aimed at providing update capabilities. Proposed IETF work on connectionless LDAP is aimed at providing LDAP access over UDP. These efforts and others may bring DNS and LDAP closer together. But for now, the best argument for not trying to replace the DNS with LDAP is that the DNS seems to be working just fine. If it’s not broken, don’t fix it!

---

**The Complementary Directory**
There are certain similarities between directories and many of the services just listed. Directories tend to complement most of these services. We will now take a moment to explore this notion of a complementary directory, what it means, and how it can create synergy among all the services in your enterprise.

A good example of the complementary directory is how it relates to a Web server. Here the directory has a number of important supporting roles to play.

First, the directory can serve as an authentication database. When clients authenticate to the Web server, their credentials are checked against the directory. When the Web server needs to make an access control decision, the directory can again be consulted to determine group membership and other information pertinent to the decision. The value of the directory in this role is especially apparent when you consider an environment that is running many Web servers, all of which need access to the same authentication database. By sharing a directory, the Web servers reduce the user management problem to a manageable level. Other services can benefit from this type of use as well.

Second, the directory can serve as a network-accessible storage device for information about configuration, access control, user preferences and profiles, and other things. The value of the directory in this role is twofold:

- If any of this information is to be shared across servers, the directory can act as a central repository, eliminating redundant administration much as it does for user and group information. Imagine the convenience of being able to change a shared configuration item on a hundred Web servers with a single change to the directory, as opposed to visiting each Web server and making the change redundantly or maintaining a separate, ad hoc system to manage configuration.

- The directory can provide a standard, network-accessible way to administer all this information. This opens up a whole new set of possibilities for standardized management tools and common administration frameworks.

As a third example of its complementary nature, the directory can be used to help organize and access information contained in the Web server itself. Today, Web search engines exist to catalog and organize Web-based content. But as anyone who has spent much time using these services can attest, you often spend more time wading through irrelevant matches to your query than reading the actual information desired. The problem is simply that Web content lacks structure and, therefore, is hard to organize and search in an automated way. With the advent of XML (Extensible Markup Language), this lack of structure is beginning to change. This is where directories come in.

Directories are great at organizing and providing subsequent access to information. Imagine today’s Web search engines driven by directories: Using the directory query language and typed information structure to more accurately specify the information you are looking for, the directory could return a much more focused set of results. Keep in mind that in this scenario the content itself is still stored in a Web server and that free-text searches would still be conducted as they are today. The directory’s value is to provide some structure on top of this arrangement and also provide a precise, yet flexible, query mechanism.

Another good example of a directory complementing existing services is found in examining file
systems and FTP servers. In this case a directory can be used not to hold the contents of files, but rather to hold meta-information about those files, their locations, who owns them, and other things that might be useful in locating them. Most importantly, the directory can hold the information that a file system or FTP client needs to access files contained in that service. An FTP server could also use a directory server for authentication purposes.

This idea of using directories to organize and search for information that you really want to access is a common theme. Directories often don’t hold the content for which you are searching, but they can hold the location of that information along with other attributes that can help you find what you are looking for.

The dividing line between what should be held in a directory and what should be held elsewhere is not always clear. In fact, sometimes it can be quite difficult to tell. General guidelines are that the more information there is, the less likely it should be put in a directory. The more frequently the information changes, the less likely it should be maintained in a directory. The less structured a piece of information is, the less benefit you will likely derive from placing it in a directory. However, the more often a piece of information is shared, the more benefit you will likely derive from placing it in a directory.

**Directory Services Overview Checklist**

Here’s a brief summary of things to think about when deciding whether to use a directory or some other piece of technology for storing information:

- *Size of the information.* Directories are best at storing relatively small pieces of information, not multimegabyte files. Directories are good at storing pointers to large things, but not the large things themselves.

- *Character of the information.* Directories typically have an attribute-based information model. If you can express your information naturally in this form, a directory might be a good choice. If you can’t, consider using a database, file system, or other approach.

- *Read-to-write ratio.* Directories are best for information that is read far more often than it is written. If the information is to be written more frequently, a database or file system might be a more appropriate choice.

- *Search capability.* Directories are made to search the information they contain. If your application requires this, a directory might be a good choice.

- *Standards-based access.* If you need standards-based access to your information, a directory is a good choice.

Keeping these principles in mind when you are deciding what to use for your application will keep you out of trouble. By this time, you should have a good understanding of what a directory is, what a directory is not, and how a directory relates to other services in your network. Hopefully, this will help you avoid the "hammer syndrome," where every problem looks like a nail you can use your directory hammer to solve.
Further Reading


Looking Ahead

This chapter gives you an overview of directories and directory services. Starting from a layman’s understanding of everyday, offline directories, we’ve explained analogous online directories and the features that enable them to be used for new types of applications. By now, you should have a good idea what a directory is and what it can do for you. Equally important, you should have a good idea of what a directory is not, and what you should not try to use it for.

In Chapter 2 we will look at directories from a historical perspective. Then in Chapter 3 we will introduce LDAP more formally in preparation for in-depth coverage of directory design.