

# Comparing LDAP Hierarchical Directories to Relational Databases

A Technical White Paper  
April 2008

## Abstract

This White Paper provides a comparison of the technical differences between Lightweight Directory Access Protocol (LDAP) based hierarchal directories and Relational Database Management System (RDBMS). The White Paper examines which system is more appropriate for certain business operations, and how the two systems can work together.

## Table of Contents

<b>Executive Summary</b> .....	<b>4</b>
<b>Introduction</b> .....	<b>5</b>
<b>LDAP Directory Services Overview</b> .....	<b>6</b>
<b>Relational Database Management System (RDBMS) Overview</b> .....	<b>8</b>
<b>Directory Services Strengths</b> .....	<b>9</b>
<b>RDBMS Strengths</b> .....	<b>12</b>
<b>When to use LDAP vs. RDBMS (and when to use both)</b> .....	<b>13</b>
When directory services are recommended.....	13
When an RDBMS is recommended.....	13
When Directory Services could be used instead of an RDBMS.....	14
<b>Examples of Directory Enabled Applications</b> .....	<b>14</b>
Using directory services for Cellular Phone portals.....	14
Using Directory Services for Hierarchical Multi-media Management System.....	15
Using Directory Services for On-Line Documentation System.....	15
<b>Where directory services and an RDBMS can work together</b> .....	<b>15</b>
<b>Summary</b> .....	<b>16</b>
<b>About Sun Java System Directory Server Enterprise Edition</b> .....	<b>17</b>
<b>Sun Directory Server</b> .....	<b>17</b>
<b>Sun Directory Proxy Server</b> .....	<b>19</b>
<b>Sun Identity Synchronization for Windows</b> .....	<b>20</b>
<b>Sun Directory Editor</b> .....	<b>21</b>
<b>Sun Directory Server Resource Kit</b> .....	<b>21</b>



## Executive Summary

It is important to understand the technical differences between LDAP based hierarchal directories and a relational database management system (RDBMS) and when one system is more appropriate for a business operations than the other system. Both systems provide mechanisms for storing, managing, and accessing data. Sun Java™ System Directory Server is an LDAP based hierarchal directory and is mostly associated as a data storage system for user information. But certainly, you can use Sun Java System Directory Server as a data storage system for many other types of data in today's world.

In the past, LDAP based directory servers had been touted as mainly read only services because write performance was very slow in the frequently changing data situations (static vs dynamic data). With respect to Sun Java System Directory Server, this argument is outdated as significant write performance improvements are made. This is the same as the argument used to be against databases where the databases were not very good if you had a lot of read (search) requests since they were not designed for highly repetitive read access and thus a directory server was a much better solution. RDBMS systems have also improved in this area with technologies like Oracle's Times Ten solution which is an in memory database solution.

The playing field has somewhat been leveled between LDAP and RDBMS where the argument is less about read or write performance today but focuses more on the data structure of hierarchical and relational models. There are still some key factors on the performance side that influence your decision but it is not as cut and dry as it used to be. One constant factor will always be true and that is if you have data that has to be represented in a relational model then you are best off using an RDBMS. If you have data that can be represented in a hierarchical model then it is more than likely that a LDAP (hierarchical) based directory server is your better solution.

Both Sun Java System Directory Server and an RDBMS have their respective strengths and sometimes it is easy to know which system to use. Some applications are a natural fit for a directory, and some for a database. On the other hand, there are other environments where either directory or database, or both, could be utilized. As the number of different applications grows, there is a corresponding growth in the gray area for choosing which system to use.

The following table highlights the strengths of the Sun Java System Directory Server and strengths of an RDBMS. Prioritizing requirements and comparing them against Sun Java System Directory Server and RDBMS's respective strengths is a good way to start evaluating which system is right for the job.

Sun Java System Directory Server Strengths	RDBMS Strengths
Tuned for <b>high read</b> performance	Tuned for <b>write</b> performance
Best used where <b>read</b> operations occur over 60% of the time (write performance is now much closer to RDBMS systems)	Best used where <b>write</b> operations occur over 40% of the time or when transactions during updates are required.
When it comes to performance both Sun Java System Directory Server and many RDBMS systems have improved their read and write performance over the last 3-5 years. The discussion is more around relational or hierarchical data models instead of read or write performance since both the solutions work in a wide variety of use cases.	
Hierarchical data model	Relational data model
Performs high-speed searches on hierarchical data	Performs complex searches on relational data
Data replication with "loosely-coupled" updates to replicas. No limit to the number of masters	Support for synchronous replication transactions with rollbacks (not all solutions).
Application transparent distributed data for extremely large and geographical deployments	Applications can perform record locking
Data shared by many applications using the standards-based access protocol LDAP. <ul style="list-style-type: none"> <li>With Directory Server Enterprise Edition (DSEE) 6 you can access data in RDBMS systems using LDAP. The LDAP calls are translated into SQL calls.</li> </ul>	Support for SQL <ul style="list-style-type: none"> <li>Some RDBMS systems provide LDAP front end interfaces. These differ and some translate LDAP to SQL while others will use LDAP to native database calls.</li> </ul>

Both Sun Java System Directory Server and most RDBMS systems are mature and robust data storage systems. When used in the appropriate environment, both Sun Java System Directory Server and an RDBMS will prove their value for managing the volumes of data today and in the future. It is a matter of selecting the correct data storage system that best meets the needs for a specific environment.

## Introduction

In the age of e-business and Web service solution architectures, it's become clear that managing data is a complicated task. Two different models for managing data are the hierarchical model (such as the one used by directories) where data can be represented under a tree structure and the relational model (such as the one used by databases) where data can be represented as tables with rows and columns.

Databases are great for storing large volumes of loosely related data, and running reports or doing number crunching. It's a general-purpose tool for handling lists of data, such as a catalog or a mailing list. However, it takes effort for users and applications to see the relationships between lists, which can be a big problem as the system grows. A database is like a Swiss Army knife, with some time and some effort, it can accomplish a great

deal of work. But it's not always the best tool for the job, since it is designed for doing a large number of tasks but not excel at any one of them.

LDAP directories take a different approach. A directory excels at managing large volumes of data that has some relationship with one another. For instance, a directory is the right tool for managing identity, policy, and security in the enterprise, because each requires defining users in relation to their role in the company and to its applications. It's optimized to make everything manageable from the top down, so that a change that affects a group of users takes place across the board.

Directory services also provide a way to address an entry so that you can find the same thing over and over again. You can specify where to find an object, so that you don't need to do a search every time you need to locate the critical data. Directory services act as a repository for a variety of critical enterprise resources. Some of the things a directory store includes are users and group information, access rights, policy, and security. Directories power web-enabled applications and significantly reduces development time, reduce administrative costs, tighten access control, delegate administration and improve security. All of this means a direct savings to the bottom line, central control over identity, and less stress on making enterprise applications work together as a single piece of fabric.

The growing relationship complexity between users, the role they have in an enterprise and the enormous amount of fragmented information are driving the need for directories and hierarchical ways of managing data.

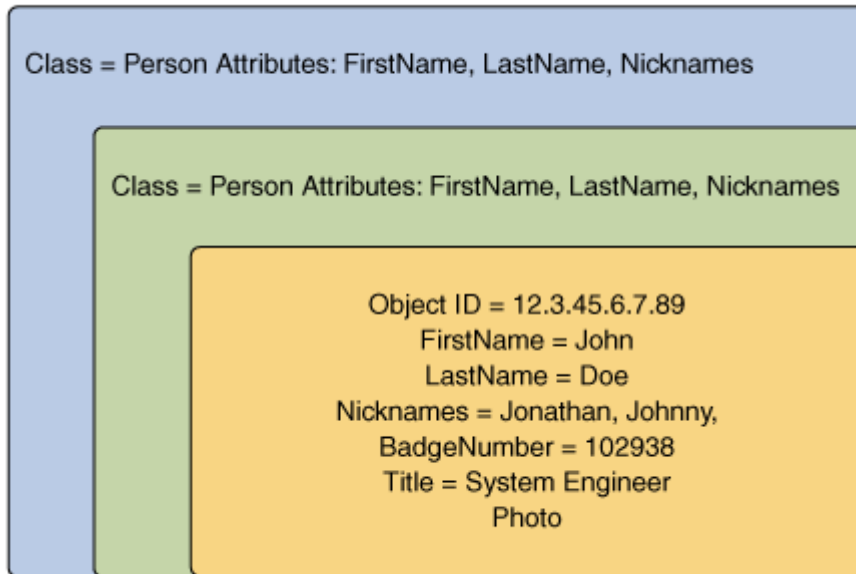
## LDAP Directory Services Overview

In our examples, we will look at Sun Java System Directory Server, which provides *LDAP-based* directory service. The server includes two major parts: a front-end that is responsible for directory service network communications using *LDAP* and a back-end that is responsible for storing the directory information, that is, data store management).

LDAP is an open standard running over TCP/IP that communicates client requests to a directory server. LDAP defines the access protocol to the data. Some of the things that LDAP can do include authentication, searching a directory, modifying elements of the directory, and compare items that are in the directory tree. LDAP does *not* define how the data is stored on the server, but rather how clients access the data and how to expect to receive the results.

The data store for a directory holds a collection of "objects" (also called "entries" in the tree). This is somewhat akin to the idea of a "record" or "row" in an RDBMS. Sun Java System Directory Server uses a data store tuned for use within an environment that requires high performance LDAP service. It is geared to support high-performance for search requests by building indexes oriented towards directory entries. In addition, Sun Java System Directory Server caches the most heavily accessed data in memory for rapid response to popular requests.

Each object stores its information in attributes (which is akin to a "field" or "column" in an RDBMS). An attribute simply consists of a name, and its associated value(s), for example **BadgeNumber=102938**. Directory services also support multi-value attributes, which means that a given attribute may contain one or more values. For example, the attribute **Nicknames** can be assigned multiple values: **Nicknames=Jonathan, Johnny, codehead**. Attributes may have values that are text or binary, a binary can be used to store a photo for example.

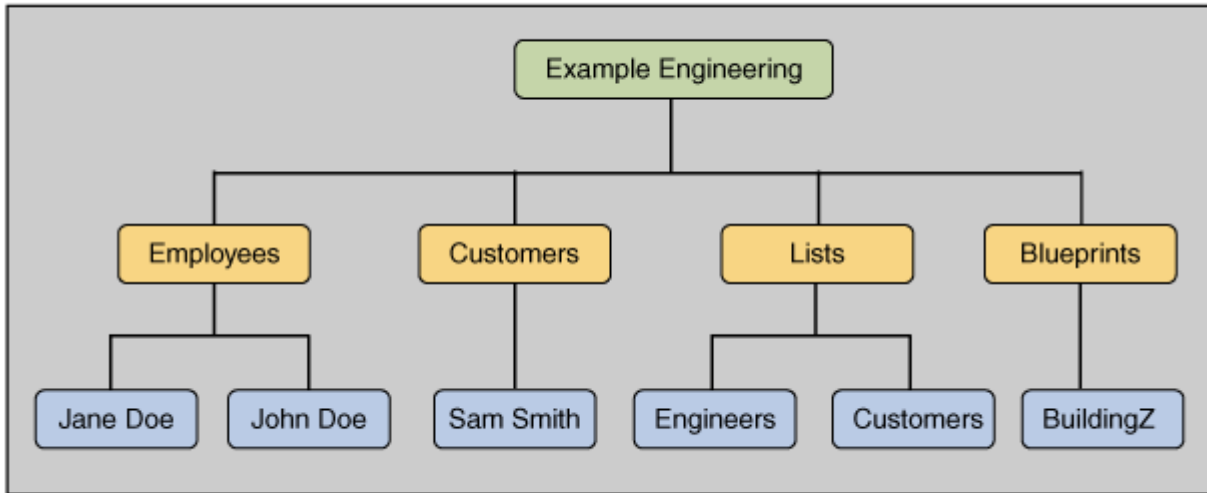


To simplify the task of defining attributes that make up an object, multiple attributes are grouped together in a class. So instead of trying to manage attribute definitions by adding them one at a time to an object, an administrator simply groups together a set of attributes as a class, then defines the object as belonging to that particular class. One class may not be enough, so each object may be assigned multiple classes to get the right mix of attributes for a given task.

The data storage model of directory services is hierarchical, which means it's shaped like an upside-down tree. The tree has a root at the top, and the tree contains definitions below it in order to define the shape of the tree that stores the objects. The upside down tree is similar to the way you look at a file system, which starts with a definition of the very top of the file system, and branches out into folders to organize and store files.

The hierarchical tree structure of directory services helps applications and users find data even when they are not sure of what they are looking for or from where to start searching. That's because no matter what type of information you are searching for in the directory, you can start your search high in the tree for broader scope or down in the tree's branches for more granularity. To do a similar task in an RDBMS, there is a basic assumption that you know what you are looking for and know which server and table you need to search.

In addition, in a database, the search would need to traverse through multiple linked tables to do the same kind of tree traversal that is an inherent ability associated with directory services.



In this example, the Example Engineering Company starts with a root called "Example Engineering". Underneath it, there are four organizational units: **Employees**, **Customers**, **Lists**, and **Blueprints**. The **Employees** include two employee objects, Jane Doe and John Doe. Example Engineering added one of their customers to their tree, and list Sam Smith as part of their **Customers** branch of their tree. Example Engineering created two groups under the **Lists** object, "Engineers" and "Customers", to help manage their anticipated growth.

Note that although there is already an entry in the tree called "Customers", the one under groups can be defined separately, to interact with different applications or store different objects. Groups can be static where the actual members of the group are added, or the groups can be dynamic by building elements from other parts of the tree. Jane Doe could have a dynamic group that lists all the customers who have an attribute such as "Dept=Marketing". The final organizational unit of Example Engineering is **Blueprints**. If Jane and John are storing blueprints electronically, such as a JPEG file for example, then these files could be quite large. Example Engineering wants to use their directory as a way to rapidly locate blueprints, instead of storing the blueprint itself. To accomplish this, Example Engineering uses an index to locate the blueprints, such as a URL.

## Relational Database Management System (RDBMS) Overview

Let's take a look at databases and how it handles data. There are three general-purpose categories for databases: An RDBMS stores data items in tables. Tables consist of columns and rows. The row holds the item and the columns hold values of the item. A very simple employee table may look something like this:

User ID	FirstName	LastName	BadgeNumber	Title
123.45.6789	John	Smith	3928	Engineer
001.02.0003	William	Jones	4958	VP
987.65.4321	Scott	Hart	9586	CEO

The data storage model of an RDBMS is based on relations or tables. Whereas a directory services will group data items (objects/entries) into organizational units, an RDBMS groups data items (rows/records) into various tables.

One of the strengths of an RDBMS is its ability to allow an application to lock rows, make changes to data in a row's column, and treat the set of changes as a single transaction. Using the table above as an example, when John Smith wants to purchase the Sun Network book, the book purchasing application can lock the corresponding Books row while it checks the Quantity and checks John Smith's available funds. If there are enough Sun Network books and John has the funds then the transaction can be committed - Quantity is decremented by one, John's funds reduced, and an entry made in the Purchases table. If Mr. Smith is lacking funds or the book Quantity is 0 then the transaction can be rolled-back with no changes being made.

In general, the applications use databases when there are the following requirements:

1. **Transactional data** - Traditional applications that process data, such as a financial application or a order processing application, handle data in a very basic manner. It looks up records, makes modifications, and makes sure that it's safe to store the modified record. A transactional database does this very well, and it's found in applications such as e-commerce systems, ERP, CRM, HR, A/P, A/R, payroll, and other line of business enterprise systems.
2. **Relational Data** - Companies with large volumes of data usually want to do some type of processing on it to recognize trends, identify issues, and mine the relationships. It all boils down to a question of "how can I build a report on all of this information?" When you start to think about the big iron relational databases, it's probably doing some kind of processing and data mining, with applications such as data warehousing, data mining, data analysis, business intelligence, and portal systems.

## Directory Services Strengths

- **Read Performance:** One of the key strengths of directory services is its read performance. It quickly locates objects in the directory tree and responds to client requests.
- **Hierarchical:** Many data models are best represented in a hierarchical or tree structure. Directory services uses a hierarchical data model but also provides **organizational units** as a way to group objects/entries, and use top down definitions to manage those objects.
- **Simple Searches :** Simple searches are read operations looking for a particular attribute value. An example of a simple search is searching for all employees whose last name contains "Smith". The hierarchical tree structure of a directory also helps locate data without the need to understand the layout of the data (schema). A search can be initiated for any attribute that is assigned a certain value at any point in the directory. Due to its high read performance, and the hierarchical data structure, simple searches are a strength of directory services.
- **Complex Searches:** While simple searches are the easiest way to find entries, in some cases you need to

find entries or a set of entries that match a set of criteria. In this case you can use complex searches such as looking for customers that live in a specific state and city and drive a blue car (&(state=TX) (city=Austin) (carcolor=blue)). This ability to create searches across multiple attributes in a users entry allows you to locate a collection of users very quickly. The key to these searches and their speed is indexing so if all of the attributes of a search are indexed properly then the results should be returned very fast. In cases where these complex searches may be very unpredictable and across many different attributes, the results will take longer to return.

- **Data Sharing:** Many applications need access to similar information. Examples of shared information may include:
  - User authentication and authorization
  - User profile information, product information, building information, etc
  - Equipment information
  - Configuration settings, application information
  - XML and XML-related (DTD, schemas, UDDI, etc.) files
  - Images (gifs, jpegs, etc).

Directory services provide a common place to store this type of information and a common protocol and API to access the information. This reduces or even eliminates the need for every application to have yet another subset of information. Directory services are a fast, highly available, distributed host for many forms of shared data.

- **Data access over the network:** LDAP is designed as a standardized data access protocol over TCP/IP and is well positioned in a networked environment.
- **Data Replication:** Replication is the mechanism by which data is automatically copied from one directory to another. Any directory tree or subtree (stored in its own database) can be copied between servers. The directory services system that holds the master copy of the information will automatically copy any updates to all replicas.
- **Change Schema over protocol:** You can modify the schema of Sun Java System Directory Server using the standard LDAP protocol. This allows you to use any tool that uses LDAP to manage the schema since no special tools or syntax is required. When there are 100's of applications that need access to the data it is likely that there will be frequent schema changes required to adapt to the changing needs of the applications or requirements of new applications coming onboard. For example, you may create an objectclass named "blog" with the OID (Object ID) 1.3.6.1.4.1.26922.1.1. OIDs are also hierarchal; breaking apart the OID, "1.3.6.1.4.1" identifies the object is owned by a private enterprise, "26922" identifies the specific enterprise and the trailing suffix "1.1" is completely defined by the enterprise. An enterprise, having defined a new globally unique object, may then publish a detailed description of the new objectclass for reference by consumers. Clearly, identifying schema elements with collision proof ids versus names is a distinct advantage.
- **Globally Unique Schema:** All LDAP schema has it's own object identifier (OID) within the directory server. This OID can also be registered with IANA (internet Assigned Numbers Authority) to ensure that your schema objects are truly globally unique and that no one else can use that schema object for other purposes other than what you have registered it for. This ensures that you do not have data naming

collisions across applications or departments/enterprises since each object has a precise and traceable object identifier (OID).

- **Standard Drivers:** When using LDAP based directory servers there actually are no drivers required to communicate with the servers since everything is over the standard LDAPv3 protocol, which all LDAP based directory servers support. This means that you can talk to many different directory servers from various vendors with the exact same LDAP language and syntax. This simplifies application development since there is no need to modify the application when it needs to talk to different LDAP data sources.
- **LDAP Controls:** Controls provide a mechanism for both the client and server to send additional instructions or information with an LDAP operation. An LDAP server will use controls to send information back to the client during certain operations. An example is most LDAP directory servers have password controls that return information to the client when it is authenticating a user. This information from the server can then inform the client that their password is about to expire, has expired or their account is locked. Clients can also use LDAP controls and one specific control would be the Persistent Search control. This control notifies the search requester if the data in the directory has changed for a given set of entries that the client is interested in monitoring. An example of this would be if an application stores it's configuration data in the directory it can then monitor that configuration data for any changes and then read the changed configuration data again.
- **Fault tolerance/Failover:** By replicating directory trees to multiple servers, directory services remain available even if some hardware, software, or network problem prevents client applications from accessing a particular directory services system. Clients are referred to another directory server for read and write operations. Note that write failover requires a multi-master replication environment.
- **Load balancing:** By replicating the directory tree across servers, the access load on any given machine can be reduced, thereby improving service response time.
- **Higher performance/Reduced costs:** By replicating directory entries to a location close to the applications, directory response times can be improved avoiding unnecessary network delays and network usage
- **Local data management:** Replication allows local data ownership and management while sharing it with other directory services systems.
- **Standard Data Distribution:** Distributed data is accomplished using referrals and/or chaining. A referral is a piece of information returned by a server that tells a client application which directory services system to contact for a specific piece of information. This redirection mechanism occurs when a client application requests a directory entry that does not exist on the local server.
- **Enhanced Data Distribution:** The Sun Java System Directory Serve Enterprise Edition also has the ability to distribute data in a different way than using referrals inside of the directory server itself. This enhanced data distribution model uses the Directory Proxy server to create a distributed set of entries across many directory servers. Distribution algorithms in the Directory Proxy server decide how the data will be distributed across the backend directory servers and can be based on first or last name, user id or

other attributes in their entry. There are many ways to define how the distribution model distributes the data and provides applications with a single view of the distributed data so they are not aware of this complexity.

- Allows horizontal scaling of flat Directory Information Tree (DIT) across many backend servers
  - Provides increased performance through parallel operations and better data maintenance procedures
  - Designed for extra-large deployments (more than 20 million entries), but can be used in smaller deployments
- **Chaining:** Chaining is a method for relaying requests to another server. This method is implemented through the database link. During chaining, a server receives a request from a client application for data that it does not contain. The server then contacts other servers on behalf of the client application and returns the results to the client application.
  - **Directory services - Managing real world objects:** Objects such as tools, vehicles, and equipment can be managed using directory services. Applications could be developed for the check-in/check-out process and directory services could be the data store for these objects. With the ability to be distributed, instances of directory services could be located across an enterprise's network.

## RDBMS Strengths

- **Write Performance:** Writes, updates, changes, modifications, adds - if over 40% or more of these types of operations are performed on a data store then there is a good chance the data should be in a RDBMS. A RDBMS system is tuned to provide about the same performance for both read and write operations.
- **Relational:** The relational data model places data items in relations or tables. Most transaction-oriented environments map to a relational data model.
- **Complex Searches:** Complex searches (in contrast to Simple Searches above) are best done using a RDBMS. For example, a search that wants all customers with the following criteria is best done with an RDBMS:
  - Over 30 but under 50
  - Who like beer and chips
  - Purchase magazines on fishing and hunting
  - Own a boat, pickup truck, and wading boots
  - Have a \$5,000 line of credit on their credit card
  - Enjoy watching cooking shows

Note that this is similar to the complex searches that we mentioned in the LDAP directory section but is across many different types of data that you may not store in the directory. Data mining applications are better served by relational data models as the relationships can be made more complex to meet deep reporting capabilities.

- **Transactions with Rollbacks:** Any environment, where changes to data items must be treated as a set, and where the changes can be reset if not all conditions are met, require transaction support with

rollback capability. Rollbacks provide the ability to bring the database back to a known state by reversing the process for a change. Most industry-leading RDBMS systems provide transaction support with rollbacks.

- **Application-based Record Locking:** Coupled with the transaction capability of an RDBMS, is the ability for an application to lock a record. In an environment where more than one application could be attempting to change a record at the same time, record locking becomes very important. Although directory services can protect against multiple threads when updating data, it does not provide the same kind of record-locking features that are found in a database.
- **SQL:** The Structured Query Language (SQL) is a language used to search for an extract information from a RDBMS. Corresponding functionality is available in directory services using the LDAP URL mechanism. Directory services can be queried using a LDAP URL using an application and/or a browser. SQL language benefits of less standardization than LDAP v3 and variation exists from vendor to vendor reducing application portability.

## When to use LDAP vs. RDBMS (and when to use both)

There are many places where directory services are the only correct solution and times where only a RDBMS makes sense. Examples of these environments will be defined first. There are other environments where either directory services or a RDBMS can be successfully implemented. And there are environments that are best served by implementing both directory services and an RDBMS.

### When directory services are recommended

Managing user information and providing a way to manage the user's authentication and authorization is an ideal use of directory services. In this environment:

- Data is being read
- Data is being shared by many applications
- Data can be defined in a hierarchal model (relationship can be defined by inheritance)
- Data is being read at a very high rate (1000s-10,000s per second)
- Data is updated in an infrequent to moderate level

It is also desired to allow update privileges of certain attributes to certain users or groups of users. Directory services have its roots in user management and continue to excel for managing both internal users and external business partners and customers. Its hierarchal model allows security rules to be inherited top down from the root, avoiding any unprotected data.

### When an RDBMS is recommended

A transactional environment, where a number of data items must all be modified as a set, is the ideal use of a RDBMS. In this environment:

- Data is being updated very often
- Modification to data items must be treated as a single set or transaction
- Data must be defined in a relational model (relationship cannot be defined by inheritance)

- Data items (rows) can be locked by an application

In a RDBMS, if any issue is raised in modifying any of the data items in the set then all the data items can be reset, or rolled back, to their state before the transaction started.

### When Directory Services could be used instead of an RDBMS

There are many cases where directory services could be used as the data store instead of an RDBMS. Using the Strengths table above, if the requirements for the data store are split between directory services and an RDBMS then directory services may be a better solution. Directories generally have a smaller system footprint than a fully transactional RDBMS, are usually less expensive to acquire, and can have lower maintenance costs than a RDBMS. Experience has also shown that directory services are easier to understand, implement, and extend. Developers can use the C, Java, and PERL SDKs to create applications that are LDAP-enabled.

### Examples of Directory Enabled Applications

#### Using directory services for Cellular Phone portals

The number of people using Cellular (cell) phones today is increasing at an incredible rate and so are the functions and applications that are available on their cell phones. In order for cell phone operators to be able to keep up with the demands of these new users and applications that need to have a highly scalable and robust directory to service their clients. Most cell phone companies today allow their customers to manage their cell phones using their web portals as well so it is important that information about the cell phone user and their personal preferences is easily available at all times.

Using Directory Services for both the web portal and cell phone preferences can provide a very rich and simplistic experience for customers when using their cell phones. When the customer first purchases their cell phone they are going to initially start adding people to their phone book. This would be made a lot easier if they could do this easily using a web interface on a portal and then have that phonebook information downloaded (or synced) with the phone itself. This information would live in directory server as entries under their main directory entry. Since the directory server is already designed for storing user information, all the required attributes are already part of the directory server so no customization is required.

In addition to storing phone numbers, users will also likely start adding services such email access and other services to enhance their phone. This preference information can be easily stored in the directory server for that user and each time they go to access the service, the cell phone would access this information in the directory server for that service for that user. This could enable capabilities such as single sign on to many different applications on their phone instead of the user having to sign into each on separately. Again a user could have the option of maintaining this information through the cell phone providers web portal for easier administration.

There are many examples in this area where directories can provide a very robust solution in the place of the RDBMS systems and usually in a much simpler manner. In cases where the number of users accessing the data is extremely large you need to be able to guarantee that access to the data will be extremely fast and this is where directories do perform the best.

## Using Directory Services for Hierarchical Multi-media Management System

Text documents can be searched using a string but how can a search be performed against files such as image or media files? Directory services could be implemented to build indexes, organization and search facilities for non-text files such as:

- Images - BMP, GIF, TIFF, JPEG, etc.
- Media - Quicktime, Flash, MPEG, etc.
- Component files - Java components (EJB, JAR, EAR, WAR, JSP), EXE, etc.
- Audio - WAV, AVI, AIFF, GSM, MIDI, AU, etc.

Each directory services entry for these types of files would include a link to the actual file and attributes about the file such as:

- Date created
- Creator's ID or email
- Version number
- Subject matter
- Description
- Actual file location (i.e. `http://....` or `ftp://....` )

A LDAP search could be performed looking for any files that contain "subject matter=BuildingZ". The list returned may include image files (a picture of the completed building) and a Quicktime file showing a video about the building. The search would also return the URL links to the actual image and Quicktime files that the user could click on to access.

## Using Directory Services for On-Line Documentation System

Assume a large extranet audience that needs to search online documentation. The documentation consists of text, images, multi-media, etc. The audience needs to search the documentation, regardless of the type, and view the documentation using a web browser. The documentation creators are an internal group consisting of engineers and technical writers. The environment can be further complicated by requiring that there may be certain pieces of the documentation that only specific people can update and only specific people can view. Directory services, not a RDBMS, may be the correct data storage management system.

Directory services could have read-only servers that have replication agreements with an directory services master system. The directory services master system supports both reads and writes. Entries in directory services can be assigned access control right down to the attribute level. This provides control for both the users who can read a specific entry but also the users who can update a particular attribute. In this environment directory services would not store the documentation but provide a URL link to the actual file(s).

## Where directory services and an RDBMS can work together

There are environments where both directory services and an RDBMS, working together, provide the best solution. In most programming models, the user interface, such as a web form, is built independently from the business logic, such as the servlet. In the day-to-day world there is a natural separation of browsing a store's shelves and purchasing an item.

Searching for items, looking the item over and selecting the item (placing it in our shopping basket) does not involve actually purchasing the item. Searching, browsing and selecting is a function best served by directory services. To purchase your items you take your shopping basket to the register and hand over your credit card. This is where the business transaction takes place and this function is best served using an RDBMS. In this case the RDBMS will need to regularly update directory services with an inventory count so a customer can see if a particular item is still available.

Another option to having updates going from the RDBMS system to the directory server is to use Virtual Directory technologies that allow applications to access data via LDAP in other data sources. In this case you could use the Sun Virtual Directory to retrieve the inventory counts for an item from the RDBMS system instead of synchronizing the data. This allows the application to continue talking LDAP until the customer decides to checkout and purchase the items. During the purchase you do not have to worry about updating many data sources, just the RDBMS and this information will be reflected the next time a customer looks at that item.

Integration between directory services and an RDBMS can be done in a number of ways besides using Virtual Directories as well. One of the criteria to use on selecting an integration method is how quickly a change in one data store must be reflected in the other data store. The following, listed in order of the fastest update to the slowest, are ways to keep data synchronized:

- Programmatically - update all data stores as part of a transaction
- Use a Virtual Directory technology
- Use an Enterprise Application Integration (EAI) suite
- Bulk load of data from one data store to the other data store

If the existing data store service is provided by an RDBMS, and integration is required between directory services and the RDBMS, then investigating the use of Virtual Directory technologies should be considered. The question that must be asked is: "does it make sense to continue investing in RDBMS expansion or does it make more sense to invest in directory services integration?" It may be considerably less expensive, in the long run, to off-load the RDBMS functions (i.e. reading, searching, data distribution, etc.) that are better served by directory services.

## Summary

Both directory services and an RDBMS provide mechanisms for storing, managing, and accessing data. Both directory services and an RDBMS have their respective strengths. There are certain environments that, because of the requirements, essentially mandate either directory services or a RDBMS. There are other environments where either, or both, directory services and a RDBMS could be utilized. In cases where either directory services or a RDBMS could be used, directory services may prove to be easier and less expensive to deploy. When used in their appropriate environments, both directory services and a RDBMS will prove their value for managing the volumes of data today and in the future.

## About Sun Java System Directory Server Enterprise Edition

With the advent of web services, there's a growing need to store information in hierarchical tables. Web services need to access broad categories of information, and locate it by searching and traversing structures until it can find the information that it needs for a particular task. One approach that some people have taken is to use a database, but databases need to convert its table-based relational nature to fit hierarchies, some of which could be changing dynamically.

One of the strengths of a directory, on the other hand, is that it is in its nature to use a hierarchical structure. This is a clear case where the lines between a database and directory blur when applying it towards a particular task, and we will likely see more of a shift from databases to directories as web services grow in prominence. Web services use directories as an organization's central repository for identities, roles and policies. A Web services directory stores organized information about users, customers, partners, suppliers, and even information on how to find other web services and applications. Using a directory, it's simple for administrators to manage who your users are and define what they can do. Users can use the directory to find other information, and use it to keep track of the information that personalizes their user experience.

It's what developers and architects building e-business and Web service solutions say they need to get the job done right. Web services are shifting the role of user data into a broader context. Web services expand the traditional definitions of identity, because now it's more than just "who we are" and "what we do". It's now about managing location, providing applications with context on what you're doing, storing preferences and personalizing the experience, and defining how we personally relate to one another and other web applications. By leveraging Sun Java System Directory Server Enterprise Edition as an end-to-end integrated identity infrastructure, IT organizations gain consistency in how the organization looks at customers, partners, suppliers, system-to-system relationships, and more. So solving identity challenges today not only saves time and money today, it will also play a crucial role to staying ahead of the competition when building Web services down the road.

## Sun Directory Server

The Directory Server component of Directory Server Enterprise Edition (DSEE) provides the most scalable, high-performance LDAP data store for identity information in the industry and serves as the foundation for the new generation of e-business applications and Web services.

**Service Manageability:** The Directory Server provides a comprehensive set of management tools for administering the server as well as the service.

- The Directory Server delivers up-to-date, consistent, and always-available identity data - and offers a central point of control for managing the service. A Web-based centralized GUI-based administration console can be used to configure and manage multiple Directory Servers leveraging Directory Server Enterprise Edition management framework. The interface includes all the tools required for effective day-to-day server administration and service from configuration to monitoring. Directory servers can be grouped in set of servers which enables configuration changes to all, some or individual servers. In

addition, new command line utilities for almost all configuration or administration actions can be performed dynamically while the servers are running. This also provides Directory administrators the tools for or advanced scripting and powerful management capabilities.

- These management features mean that most management operations that would typically be performed while the directory is offline - such as backup, bulk import, and re-indexing - can instead be performed while it is online, thus maximizing availability.
- Management flexibility makes it much simpler to deploy the directory service into many different environments. If data centers are outsourced to third-party companies or operated on a "lights out" basis that requires remote management, the command line utilities make it as easy to manage the service as if it were in a local data center.

**Availability:** The Directory Server natively supports a variety of access protocols and offers a highly flexible and scalable replication environment ensuring availability in distributed environments.

- The Directory Server supports the LDAP v2 and v3 protocols and the Directory Service Markup Language (DSML) v2 natively for standards-based access. LDAP and DSML over HTTP/Simple Object Access Protocol (SOAP) protocols enable clients anywhere on a network to securely search and update directory data objects, receive changes made by other applications, and authenticate users or applications – even through firewalls.
- To ensure that there is no single point of failure for applications using the aforementioned protocols to access identity data, Directory Server supports an unlimited number of masters or read-only servers. Special features of the replication protocol allow for optimizations when replicating data over high-latency networks.

**Security:** The Directory Server provides many advanced security features to achieve compliance with information security policies and to ensure that only those with proper authorization have access to the information.

- Macro-level and dynamic Access Control Instructions (ACIs) make access definable at the lowest level of data - an attribute. They make it possible to define access control policies once and then re-use them across the directory tree. Macro ACIs can be used to optimize the number of ACIs in the directory and thereby reduce the complexity of the security framework.
- Along with ACIs, role-based access provides a simpler way to provide access based on information in a user's entry. Roles are defined and administered like groups, but they provide more efficient grouping mechanisms for applications. Roles can be used in ACIs to control access to data. They can also be used by Class of Service (CoS) to define "virtual" attributes for an entry, reducing storage requirements on entries and allowing a single change to update an unlimited number of related entries.
- Directory Server supports a means for determining what access a user has on a set of information. By using the Get Effective Rights control, administrators who maintain access policies for the directory service can tighten security by auditing the permissions of directory users and applications. This capability can also be used to build applications with adaptive interfaces, based on the user's rights.
- Directory Server supports encryption mechanisms to protect data on the disk and during transfer through communications channels. Combined with support for fractional replication and data-hiding based on access, this can be used to comply with European Union and other international privacy regulations.
- To guard against unauthorized access to user accounts that can be used to obtain identity information, the Directory Server supports multiple password policies that can be defined on a per-user basis or

targeted to certain groups. These policies help to ensure users are changing passwords on a regular basis and that anyone attempting to hack into an account is effectively blocked.

**Scalability:** The Directory Server provides both vertical and horizontal growth without major deployment redesign. This level of scalability becomes increasingly critical as deployment grows.

- The Directory Server is the highest-performing LDAP directory server in the market today, with the ability to provide sustained search performance of over 10,000 entries per second on a single machine and horizontal scalability to tens of thousands of searches per second.
- The requirement to store and update information constantly is increasing with the expansion of use across the organization. Update performance of directory server has been seen near the 1,000 per second range on multi-million entry deployments, allowing for near relational database-write performance.
- As the industry's only 64-bit, enterprise-class directory with linear CPU scalability to 18 CPUs, the Directory Server allows access to maximum memory capacity and delivers high performance accommodating extremely large directories on a single system for maximum hardware benefit.
- Advanced replication mechanism with unlimited number of masters, highly-available change log, prioritized replication and global account lockout let you deploy your service in widely distributed environment to match your geographical constraints.

## Sun Directory Proxy Server

The Directory Proxy Server component of Directory Server Enterprise Edition (DSEE) is an LDAP application layer protocol gateway designed to deliver high availability, virtual directory, data distribution, enhanced directory access control, and schema compatibility.

**Service Manageability:** Same web-based administration console as the Directory Server.

- Directory proxy servers can be grouped in a set of servers to enable broad configuration changes to all, some or individual servers
- New advanced Command Line Interface enables advanced scripting and provides powerful management capabilities

**Virtual Directory:** Allows quicker integration of new or existing data sources by transparently aggregating data from multiple directories and databases.

- Data in many data sources can be managed through one interface, avoiding synchronization and data ownership political issues
- Applications can take advantage of new and interesting identity data immediately without costly changes
- Spans across multiple applications, allowing applications with different data structure requirements to take advantage of the same physical data

**Data Distribution:** Includes a data distribution feature to support massive deployments where data can be

spread across many servers.

- Allows horizontal scaling of flat Directory Information Tree (DIT) across many backend servers
- Provides increased performance through parallel operations and better data maintenance procedures
- Designed for extra-large deployments (more than 20 million entries), but can be used in smaller deployments

**Availability:** Ensures that systems always have access to the data they need when they need it through configurable load balancing and failover/failback.

- Works with the Directory Server to protect against denial of service attacks and thus enables around-the-clock reliability
- Automatically routes requests appropriately through a referral mechanism and provides secure firewall-like services for the Directory Server
- Detects outages and routes traffic around affected areas, effectively load-balancing requests across systems; when the affected areas are restored to operation, the Proxy Server detects it automatically
- Routes requests to the most appropriate server based on type of operation

**Security:** The Directory Proxy Server accommodates large numbers of users accessing the directory and minimizes the security risks associated with providing this level of access.

- Security features make it possible to determine where a request is coming from, whether it is allowed, and what type of authentication is required for it
- Uses the concept of groups to define how to identify an LDAP client and what restrictions to enforce on clients that match a particular group
- Can configure a fine-grained access control policy on LDAP directories to protect private directory information from unauthorized access while still making it safe to publish public information
- Can be configured to prevent certain kinds of operations typically performed by web trawlers and robots in search of information.

## Sun Identity Synchronization for Windows

The Identity Synchronization for Windows component of [Directory Server Enterprise Edition \(DSEE\)](#) allows seamless integration with the Windows desktop environment.

**Interoperability:** Identity Synchronization for Windows provides basic synchronization of identity data between DSEE and Microsoft Active Directory.

- The synchronization of key identity data (including users, passwords, and domain global distribution and domain global security static groups) eliminates the need for users to modify passwords several times to accommodate different applications' authentication mechanisms
- The use of a non-intrusive implementation for synchronizing key identity data eliminates the time-consuming and maintenance-intensive need to install a client component on Windows Active Directory servers. This also addresses political issues related to integration of information often owned by separate

groups

- Identity Synchronization for Windows enables users to change passwords and other identity data in either the Windows environment or the web-based application environment and maintain synchronization between Active Directory and the Directory Server
- Disabling of accounts can be synchronized between Active Directory and the Directory Server, which ensures conformance of access policies to applications and data between the Microsoft Windows desktops and Web-based applications

## Sun Directory Editor

The Directory Editor component of [Directory Server Enterprise Edition \(DSEE\)](#) is a J2EE-based Web application for efficient, cost-effective management of directory data.

**Manageability:** Directory Editor allows users to manage identity data within the directory service.

- Simplifies and streamlines everyday management tasks by allowing administrators to create a forms-based Web interface for users
- Supports extensive customization, branding, and embedding for the interface. Customization is done using a form-based interface for configuration, rather than writing code

To ensure data security and privacy, built-in authorization controls limit visibility of menus and actions, ensuring that users only see what they are authorized to see within the application

## Sun Directory Server Resource Kit

The Directory Server Resource Kit component of [Directory Server Enterprise Edition \(DSEE\)](#) provides tools and APIs for deploying, accessing, tuning, and maintaining the Directory Server Enterprise Edition. These utilities will help implement and maintain more robust LDAP-based solutions.

- LDAP Software Development Kits (SDKs) for C and Java programming languages help streamline the writing of client applications for DSEE. SDKs include a collection of high-level APIs for the rapid development of new applications that take advantage of DSEE features.
- Performance testing and capacity planning tools help deployment engineers and system administrators measure performance and perform capacity-planning installations of DSEE
- Debugging and maintenance tools help with DSEE troubleshooting as well as daily maintenance
- Deployment utilities and tools make it easy to roll out new installations of DSEE and to migrate to new releases
- LDAP productivity tools include sample LDAP applications that were developed using DSEE