LDAP

Lightweight Directory Access Protocol

Bernhard R. Fiser, 0109815
b.fiser@abenteuerland.at

2002/12/10, Version 1.3

2002/10/28, Version 1.2
2002/10/25, Version 1.1

Inhaltsverzeichnis

1. Abstract

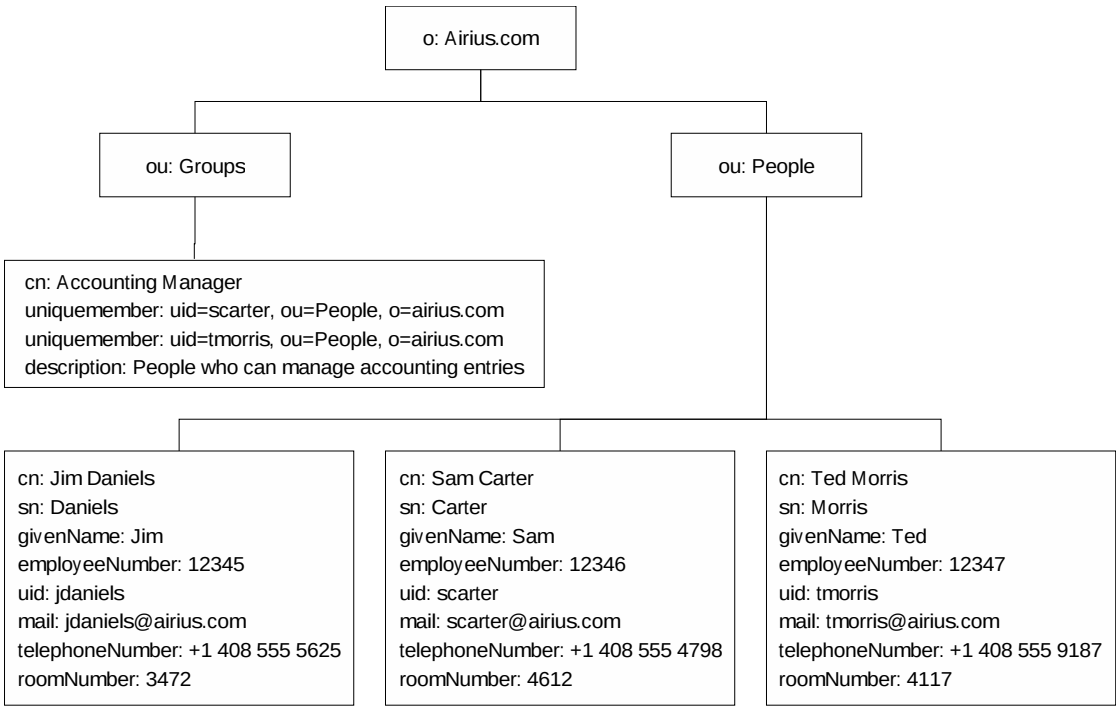
Directories und LDAP gewinnen heute zutage immer mehr an Bedeutung, und sind in IT Fachkreisen nicht mehr wegzudenken. Da LDAP zum Basiswissen jedes sich in der IT−Branche bewegenden Individuums gehören sollte gibt dieses Dokument einen groben Überblick über das LDAP Konzept und die Struktur, sowie eine Einführung in die Suche in Directories zum besseren Konzeptverständnis. Um das fachliche Knowhow abzurunden, werden auch der geschichtliche Hintergrund und einige Anwendungen beleuchtet.

2. Einführung

Ein Directory (Verzeichnis) ist ein Service, worin es möglich ist nach bestimmten strukturierten Daten zu suchen. Obwohl sich das nun nach einer "normalen" relationalen Datenbank anhört, besteht doch ein wesentlicher Unterschied (Ein LDAP Directory verwendet intern üblicherweise eine relationale Datenbank um die Directoryinhalte zu speichern.). Ein Directory ist genau beschrieben durch die Art wie User oder Programme mit dem Directory interagieren. Das bedeutet, dass das Protokoll und die Programmierschnittstelle (API) genau definiert sind.

Abbildung 1 zeigt ein Beispiel über die Art und Weise wie Daten in einem Directory gespeichert werden. In diesem Beispiel werden alle Informationen zu einer Person in einem einzelnen Objekt gespeichert. Die Struktur des Baumes definiert die Beziehungen − die Relationen − zwischen den Objekten. Die Eigenschaften selbst, die ein bestimmtes Objekt im Directory charakterisieren, sind als Attribute gespeichert, deren Bedeutung selbst wiederum in den Schemas festgelegt ist.

Abbildung 1. LDAP Directory Baum.



Als Beispiel möchte ich an dieser Stelle das häufig verwendete Attribut cn erwähnen, was für ***c**ommon **n**ame* steht und soviel bedeutet wie 'allgemeine Bezeichnung'.

In Abbildung 2 sieht man nun, wie die gleiche Information in einer relationalen Datenbank abgelegt werden würde. Hier wird die Personeninformation in einer Tabelle abgelegt, die Informationen über die Organisationseinheiten in einer weiteren Tabelle und in einer Dritten die Information über die Organisation selbst.

Abbildung 2. Darstellung der Abbildung 1 als Relationenschema.

User table

ID	EMP#	FULL NAME	LAST	FIRST	uid	PHONE	EMAIL	ou
1	12345	Jim Daniels	Daniels	Jim	jdaniels	+14085555625	jdaniels@airius.com	1
2	12346	Sam Carter	Carter	Sam	scarter	+14085554798	scarter@airius.com	1
3	12347	Ted Morris	Morris	Ted	tmorris	+14085559187	tmorris@airius.com	1

Organizational Unit table

ID	NAME	ORGANIZATION
1	People	1
2	Groups	1
3	Accounting	2

Organization table

ID	NAME
1	Airius.com
2	acme.com

Die Hauptunterschiede [1] zwischen einer relationalen Datenbank und einem Directory sind in der Folge aufgelistet:

- Directories sind hauptsächlich für Umgebungen konzipiert, in denen viele Clients gleichzeitig viele Suchoperationen, das heißt lesende Zugriffe, und wenig Schreibzugriffe durchführen. Als Konsequenz daraus folgt, dass relationale Datenbanken bei vielen Schreibzugriffen üblicherweise eine bessere Performance aufweisen.
- Directories unterstützen normalerweise keine komplexeren, relationale Abfragen wie zum Beispiel table joins.
- Directories unterstützen keine Transactions über mehrere Operationen um die Datenintegrität zu gewährleisten.
- Directories haben wesentlich bessere und flexibler Suchfunktionen für Textsuche und unscharfe Suche.
- Directories sind mit sogenannten Schemas ausgestattet, die bei der Installation mitgeliefert werden und weltweit standardisiert sind. Bei relationalen Datenbanken hingegen muss man Relationsschemas selbst definieren, weiters sind diese nicht standardisiert, und auch nicht die Wertebereiche der Attribute.
- Directories werden der Redundanz und Performance wegen öfters weitreichend repliziert im Gegensatz zu Datenbanken.
- Directoryprotokolle, wie beispielsweise LDAP, sind meistens wesentlich besser für WAN (Wide Area Network) geeignet, wie beispielsweise das Internet oder große Firmennetzwerke.

- Directories sind meistens wesentlich einfacher zu konfigurieren als kommerzielle Datenbanken und kosten auch wesentlich weniger, was vielfach auf die oben angeführten Punkte zurück geführt werden kann.

3. Geschichtliche Entwicklung

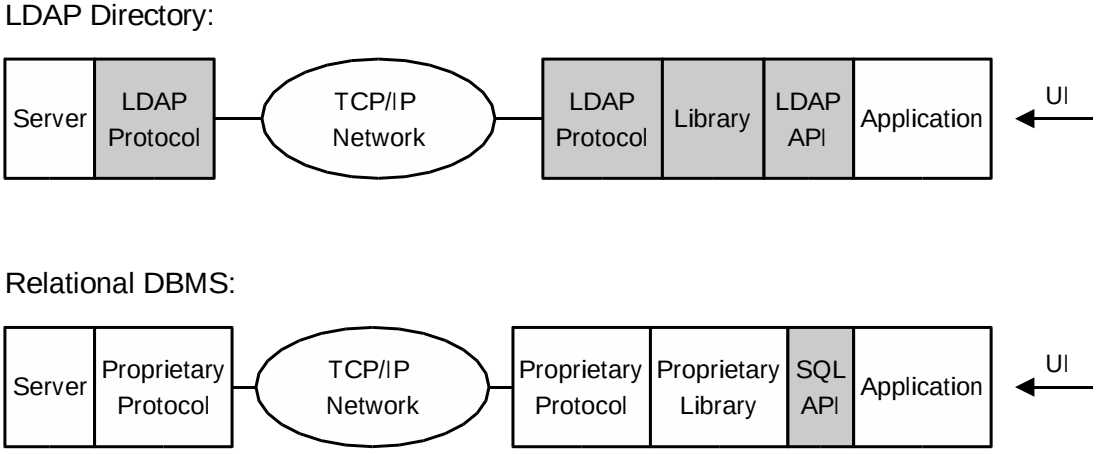
[1] Das Xerox Clearinghouse war das erste großräumig verteilte Directory Service für die gemeinsame Nutzung von User Account Informationen. Das Clearinghouse basierte auf den Forschungen rund um das verteilte Computersystem *Grapevine* im Palo Alto Research Center von Xerox in den frühen 80er Jahren. An diesem Netzwerk nahmen bis zu 1500 Stationen teil. Obwohl damals Pioniersarbeit geleistet wurde, konnte sich das System aufgrund des proprietären Protokolls nicht durchsetzen.
Internetdirectories entstanden mit der Einführung des Internet und der gleichzeitigen Verfügbarkeit des sehr mächtigen Betriebssystems UNIX. Eines dieser Services ist das Domain Name Service (DNS) welches für die Übersetzung von Internet Domain Names wie `www.myserver.com` in eine IP−Adresse verantwortlich ist. Ein anderes ist das *whois* Service, eine Datenbank für Domainname Registrierung.
NIS und NIS+ (Network Information Service) sind Services die von Sun Microsystems entwickelt wurden, und dienen dazu um ein einheitliches Login auf unterschiedlichen Systemen zu ermöglichen. Obwohl aus Kompatibilitätsgründen diese Services auf UNIX−Systemen weiterhin unterstützt werden, geht der Trend Richtung LDAP.
NDS (Novell Directory Service), ein high sophisticated, skalierbares und hierarchisches Service, wurde von Novell 1993 eingeführt, und bietet einen Ersatz für die user und network database, die sogenannte *Bindery.* 1997 bot Novell auch LDAP Clients die Möglichkeit auf das NDS zu zugreifen. Derzeit ist Novell beschäftigt das NDS völlig neu zu schreiben und durch LDAP zu ersetzen.
Active Directory wurde von Microsoft mit Windows 2000 eingeführt um das total veraltete Windows NT Domainkonzept zu ersetzen. Obwohl es ursprünglich hauptsächlich für das Management von Arbeitsstationen gedacht war, kann es trotzdem für andere Anwendungen auch verwendet werden, da es vollen Zugriff für LDAP Clients bietet.
X.500, the "heavyweight" directory service ist ein Standard, der aufgrund der starken Ausbereitung der verteilten Systeme durch die ITU (International Telecommunication Union, früher CCITT) und die ISO (International Organization for Standarization) entwickelt wurde. 1988 wurde das erste Dokument erzeugt. Verschiedene Umstände machten es sehr kompliziert Client Applikation für den Zugriff auf X.500 Directories zu entwickeln, und so wurde es abgewandelt und auf das im Internet verwendete Transportprotokoll TCP/IP umgesetzt, LDAPv1 im Jahr 1993. Obwohl dieses Protokoll noch nicht besonders erfolgreich war, wurde es weiterentwickelt, und so entstand 1995 LDAPv2, welches den absoluten Durchbruch schaffte.
Die zukünftige Entwicklung von LDAP ist für einige Themenbereiche schon vorherzusehen, da einige wesentliche, wünschenswerte Funktionen noch nicht standardisiert sind, dazu gehören *Authentication, Replication, Accesscontrol, Dynamic attributes* und *Transactions.*

4. Das LDAP Protokoll

[2] Das derzeit letztgültige, standardisierte Protokoll ist LDAP Version 3, üblicherweise notiert als LDAPv3. Es ist eine sehr ausführliche Definition und umfasst die auf TCP/IP aufsetzende Protokollschicht, die Syntaxdefinitionen der Attribute, die Darstellung der Distinguished Names (DN), die Definition der Suchfilter, das LDAP URL−Format, die Definition der Authentication Methoden, eine Erweiterung zur Verwendung von Transport Layer Security (TLS) und eine Zusammenfassung der Schemas zur Verwendung mit LDAPv3.

Durch diese Definitionen ist eine problemlose Interkommunikation zwischen Servern, Clients und Usern möglich, das einer einfachen, unkomplizierten und portablen Lösung der heutigen Zeit entspricht. Im Gegensatz zu einer relationalen Datenbank, bei der im Wesentlichen nur eine hochsprachliche Kommunikation (SQL [3]) definiert ist, nicht jedoch die restlichen Randbedingungen. Abbildung 3 zeigt nun schematisch die wesentlichen Vorteile davon, wie weitreichend diese Definitionen sind, im Gegensatz zu relationalen Datenbanken.

Abbildung 3. Vergleich der Protokolldefinition zwischen LDAPv3 Directories und relationalen Datenbanken.

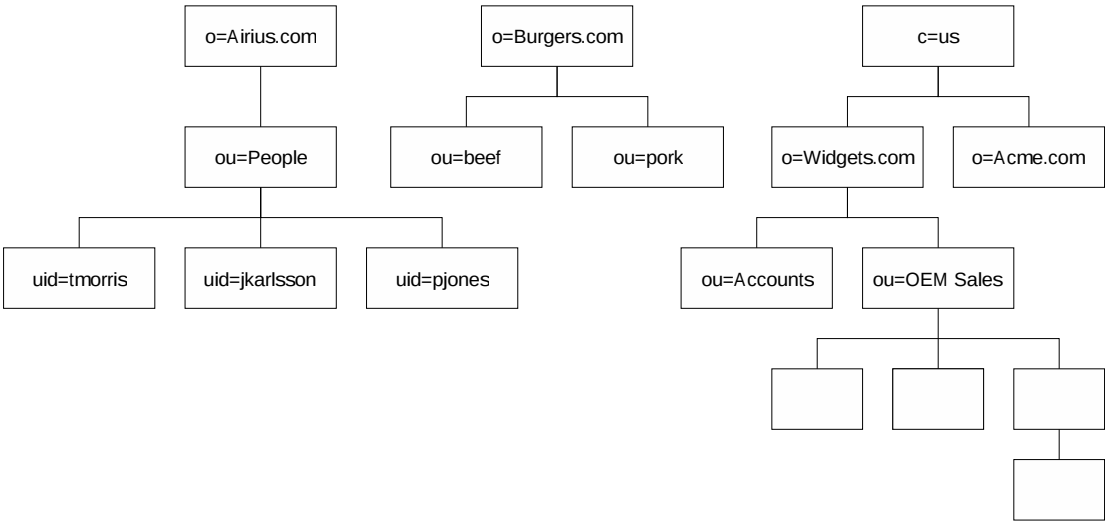


Zusätzlich zu dieser Protokolldefinition gibt es ein Datenformat, das sogenannte LDIF Format [6] (LDAP Data Interchange Format). Mit LDIF kann man einzelne Einträge, Zweige oder das ganze Directory eines LDAP Servers exportieren oder importieren. Es ist ein Textformat mit normalen ASCII Zeichensatz, kann daher auch manuell mit einem Texteditor erstellt und bearbeitet werden, und kann auch dazu verwendet werden Daten einfach in ein Directory aufzunehmen. Viele Emailclients beispielsweise unterstützen auch den Import und Export des Adressbuchs im LDIF Format. Man könnte dadurch zum Beispiel das Adressbuch der Firma in der man beschäftigt ist aus dem LDAP Directory des Unternehmens exportieren und im Homeoffice im Emailclient wieder importieren.

5. Struktur, Schemas und Attribute

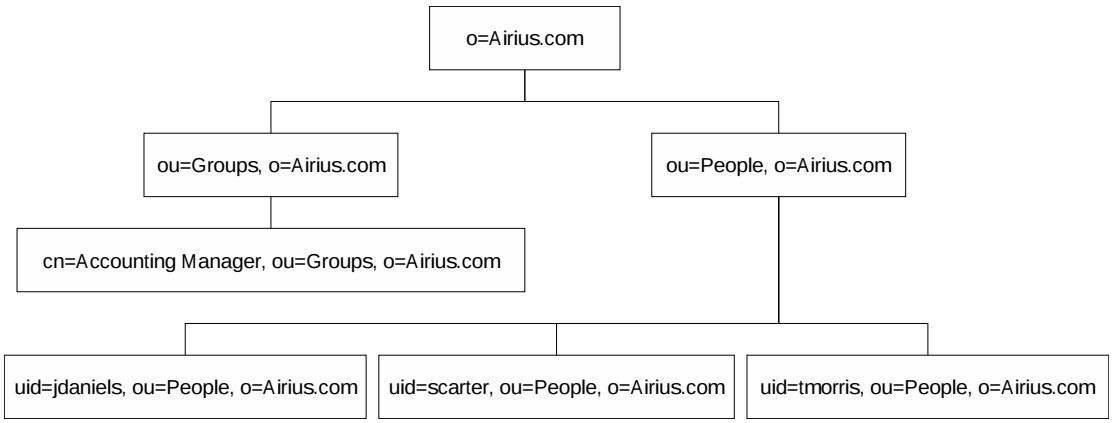
Das Directory ist streng hierarchisch organisiert, die Daten werden als Entries (auch als Einträge oder Objekt bezeichnet) gespeichert. Jeder Eintrag hat einen sogenannten distinguished name (DN), was in etwa einem primary Key in einer relationalen Datenbank entspricht, welcher im Unterschied dazu aber im gesamten Directory eindeutig ist, der primary Key aber nur eine eindeutige Bestimmung innerhalb einer Table zulässt. Ganz oben im Baum sind die sogenannten root Einträge (Wurzeleinträge), die auch als suffixes oder namimg contexts bezeichnet werden. Abbildung 4 zeigt ein Directory mit drei root Einträgen. Jeder einzelne Eintrag im Directory unterscheidet sich von seinen Nachbareinträgen durch den relativen distinguished name (RDN, relative distinguished name).

Abbildung 4. Directory Baum mit mehreren naming contexts.



Der RDN besteht aus einem Attribute des Eintrages gefolgt durch ein Gleichheitszeichen (=) und einem Wert des Attributes (zum Beispiel uid=mmueller oder ou=Technik). Den DN (distinguished name) kann man nun zusammen setzen, indem man den RDN eines Directory Eintrages nimmt, den Baum nach oben wandert und immer den nächsten RDN durch Beistrich getrennt anhängt. Umgekehrt kann man aus dem DN die RDN's der Einträge erzeugen, indem man die DN in lauter Einzelteile zerteilt. Abbildung 5 zeigt einen Directory Baum mit Einträgen und deren DN's. Zu beachten ist, dass der RDN immer auch als Attribut des selben Typs und Wertes im zugehörigen Directory Eintrag existieren muss. Es darf keinen Eintrag im gesamten Directory geben, dessen RDN nicht auch als Attribut abgebildet ist. Versucht man einen derartigen Eintrag zu erstellen, so sollte der LDAP Server sofort mit einer Fehlermeldung antworten und die Eintragung nicht zulassen.

Abbildung 5. Ein einfacher Directory Baum mit DN's.



Jeder Eintrag im Directory verfügt weiters über verschiedene Attribute, zumindest jedoch das Attribut **`objectClass`**, welches immer den Typ des Eintrags definiert. Die `objectClass` definiert außerdem weitere Attribute, die im Zusammenhang mit diesem Eintrag verbunden sein müssen oder können. Das wiederum ist im Schema definiert. Es ist möglich zu einem Eintrag auch mehrere `objectClass's` zu kombinieren, mit anderen Worten kann man einen ganz bestimmten Eintrag mit Informationen erweitern. Das ist eine Option, die bei relationalen Datenbanken gar nicht, oder nur mit viel Aufwand abgebildet werden kann.

Ein wichtiges Merkmal aller Definitionen, das heißt Schemas, Klassen und Attribute, ist, dass jedes dieser Elemente eindeutig durch eine sogenannte `OID` (object identifier) weltweit identifiziert und zugeordnet werden kann. Die `OID` wird durch numerische durch Punkte getrennte Zeichenketten repräsentiert und auch in anderen Protokollen, beispielsweise dem SNMP, verwendet. Die in Abbildung 6 gezeigte `objectClass top` wird durch die `OID` 2.5.6.0 eindeutig repräsentiert. `OID`'s können bei internationalen Organisationen wie ITU, ISO oder IANA registriert werden. Weitere, detailiertere Informationen zu `OID`'s finden sich in [8].

Als Beispiel sind hier die `objectClass top`, `objectClass person`, `objectClass organizationalPerson` und die `objectClass inetOrgPerson` im Format der Schemadefinition LDAPv3 abgebildet [4]:

Abbildung 6. Schemadefinitionen der `objectClass` `top`, `person`, `organizationalPerson` und `inetOrgPerson`.

```
objectclass ( 2.5.6.0
      NAME 'top'
      ABSTRACT
      MUST objectClass )

objectclass ( 2.5.6.6
      NAME 'person'
      SUP top STRUCTURAL
      MUST ( sn $ cn )
      MAY (
            userPassword $ telephoneNumber $
            seeAlso $ description ) )

objectclass ( 2.5.6.7
      NAME 'organizationalPerson'
      SUP person STRUCTURAL
      MAY (
            title $ x121Address $ registeredAddress $
            destinationIndicator $
            preferredDeliveryMethod $ telexNumber $
            teletexTerminalIdentifier $
            telephoneNumber $ internationaliSDNNumber $
            facsimileTelephoneNumber $ street $
            postOfficeBox $ postalCode $
            postalAddress $ physicalDeliveryOfficeName $
            ou $ st $ l ) )

objectclass ( 2.16.840.1.113730.3.2.2
      NAME 'inetOrgPerson'
      DESC 'RFC2798: Internet Organizational Person'
      SUP organizationalPerson STRUCTURAL
      MAY (
            audio $ businessCategory $ carLicense $
            departmentNumber $ displayName $ employeeNumber $
            employeeType $ givenName $ homePhone $
            homePostalAddress $ initials $ jpegPhoto $
            labeledURI $ mail $ manager $ mobile $ o $ pager $
            photo $ roomNumber $ secretary $ uid $
            userCertificate $ x500uniqueIdentifier $
            preferredLanguage $ userSMIMECertificate $
            userPKCS12 ))
```

Die `objectClass organizationalPerson` stellte eine Menge an Attributen zur Verfügung um eine Person eines Unternehmens darzustellen. Was hier deutlich erkennbar ist, ist dass die `objectClass organizationalPerson` zwingend die `objectClass person`, und diese wiederum die `objectClass top` bedingt. Die Attribute sind entweder als `MUST`, das heißt sie müssen zwingend vorhanden sein, oder als `MAY`, also als optionale Attribute, definiert. Die Attribute selbst sind die Felder, die die eigentlichen Daten eines Directoryeintrages enthalten. Sie charakterisieren einen ganz bestimmten Eintrag. Die Bedeutung und die Wertebereiche der Attribute sind genau beschrieben, das heißt beispielsweise `givenName` enthält den Vornamen einer Person und ist vom Typ `caseIgnoreString`. In dieser exakten Definition der Attribute und ihrer Werte liegt der enorme Vorteil in der Verwendung eines Directories. Sie repräsentieren eine einheitliche Datenstruktur auf die von verschiedensten Programmen und Systemen aus zugegriffen und deren Semantik eindeutig interpretiert werden kann. Die genauen Formate können genauso

wie die genauen Schema Definitionen in [2] nachgelesen werden, es sind hier in Abbildung 7 jedoch zwei Beispiele zur näheren Erläuterung angeführt.

Abbildung 7. Attributdefinitionen fuer attributetype `name` und `commonName`.

```
attributetype ( 2.5.4.41
      NAME 'name'
      EQUALITY caseIgnoreMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

attributetype ( 2.5.4.3
      NAME ( 'cn' 'commonName' )
      SUP name )
```

Diese Definitionen können nun mit konkreten Einträgen befüllt werden und danach von verschiedenen Applikationen wieder abgerufen werden. Ein konkreter Eintrag in einem Directory könnte nun wie in Abbildung 8 dargestellt aussehen.

Abbildung 8. Konkretes Beispiel eines `inetOrgPerson` Eintrages.

```
dn: uid=scarter, ou=People, o=airius.com
cn: Sam Carter
sn: Carter
givenName: Sam
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
ou=Accounting
ou=People
l: Sunnyvale
uid: scarter
userPassword: 12345678
mail: scarter@airius.com
telephoneNumber: +1 234 567
```

Bei der Konzeption der Struktur eines LDAP Directories kann es sehr hilfreich sein einen Überblick über die vorhandenen Schemas zu haben. Es ist eine Vielzahl unterschiedlicher Schemas zu allen Fachbereichen bereits standardisiert, oder zur Standardisierung vorgeschlagen, und es ist zu empfehlen vor der Erstellung eigener, proprietärer Schemas genau zu recherchieren, ob nicht eines der bereits vorhandenen verwendet werden kann. Eine sehr gute und laufend aktualisierte Übersicht über die vorhandenen Schemas findet sich unter [5].

6. Datensuche im Directory

LDAP verfügt über mächtige Suchfunktionen, mit denen es möglich ist mit Hilfe verschiedenster Kriterien im Directory nach Einträgen zu suchen. Durch die hierarchische Struktur und die damit zwingend verbundene Relation der Einträge zueinander, ist es möglich den gesamten Datenbestand mit einer einzigen, einfachen

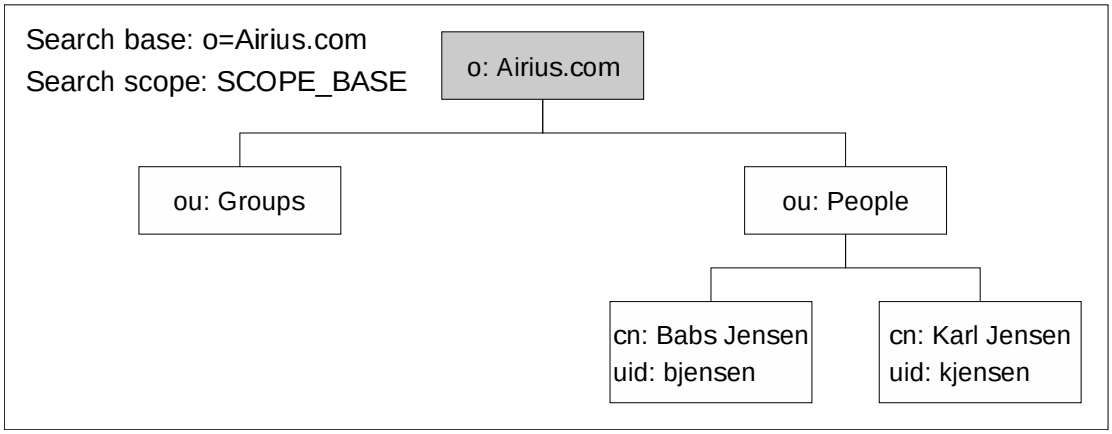
Operation abzusuchen. Ganz anders müsste man in einer relationalen Datenbank eine SELECT−String formulieren, in dem man die Relationen der Tables zueinander in der WHERE−Bedingung ausformulieren müsste. Das kann in einer großen Datenbank sehr schwierig bis unmöglich sein.
Die Suche im Directory benötigt und stützt sich im wesentlich auf die vier Parameter Base DN, Scope, Filter und Attribute.

Die Base DN (Suchbasis) gibt an von welchem Eintrag aus gesucht werden soll. Nachdem das Directory als Baum organisiert ist, wird von dieser Basis aus noch unten verzweigend gesucht.

Der Scope (Sichtweise) gibt an wie weit in die Tiefe des Directories gesucht werden soll. Im LDAP Protokoll werden dabei folgende drei Möglichkeiten unterschieden:

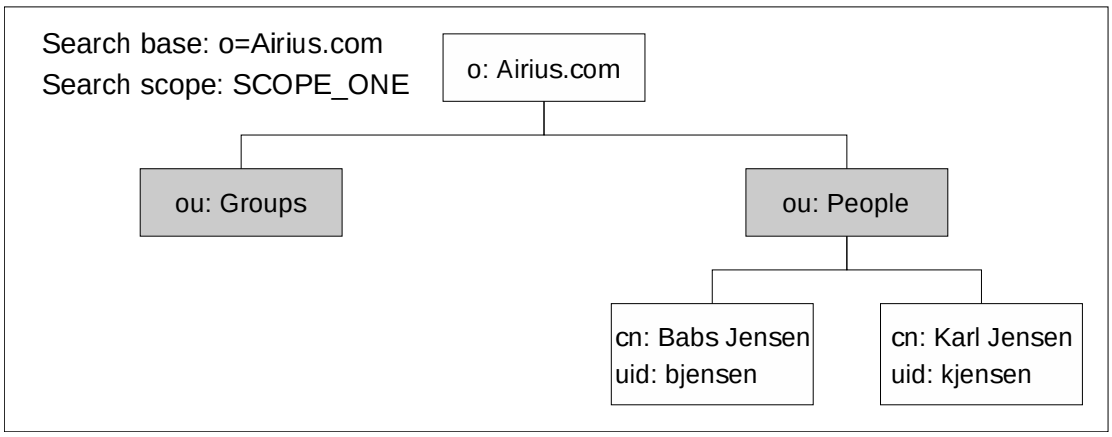
- **BASE**, gibt an, dass nur die Stelle, die durch die Base DN selbst repräsentiert wird durchsucht werden soll. Mit anderen Worten wird nur genau ein Eintrag, der der Base DN, untersucht und zurückgeliefert, sofern die weiteren Suchkriterien erfüllt sind. Abbildung 9 zeigt grau unterlegt welche Einträge mit Scope BASE durchsucht werden.

Abbildung 9. Suche mit Scope BASE.



- **ONE**, gibt an, daß alle Einträge, die genau ein Ebene unterhalb der Base DN im Baum, nicht jedoch der Eintrag, der durch die Base DN selbst representiert wird, durchsucht werden. Abbildung 10 zeigt, welche Einträge mit Scope ONE durchsucht werden.

Abbildung 10. Suche mit Scope ONE.



- **SUBTREE**, gibt an, dass alle Einträge unterhalb im Directory Baum, einschließlich des durch den Base DN repräsentierten, durchsucht werden.

Der Suchfilter ist durch eine Zeichenkette definiert. Filter werden durch Klammern und eine Kombination aus den Zeichen &, | und !, welche die logischen Operationen AND, OR und NOT repräsentieren. Will man beispielsweise nach allen Personen suchen, die mit "tony" beginnen, würde man folgenden Filter verwenden:

```
(&(objectClass=person)(cn=tony*))
```

Dieser Filter nimmt alle Einträge, die die `objectClass person` enthalten und im `commonName` tony* enthalten ist. Das Zeichen * dient als Wildcard und ersetzt null bis beliebig viele Zeichen. Durch die Einschränkung auf `objectClass person`, werden alle anderen Einträge, die keine Person sind, bei der Suche nicht berücksichtigt. Das Attribut `commonName` könnte auch in anderen Einträgen vorkommen. Das Attribute `commonName` ist wie die meisten anderen LDAP Attribute case−insensitive, wodurch die Suche mit `cn=tony*`, `cn=Tony*` oder `cn=TONY*` immer zum gleichen Suchergebnis führen wird. Suchfilter können außerdem beliebig tief verschachtelt werden, wie das folgende Beispiel zeigt:

```
(&(objectClass=person)(|(cn=sam carter)(cn=tony*)))
```

Mit diesem Filter werden alle Personen gefunden, die entweder mit "tony" beginnen oder "sam carter" heißen.

Die Attribute, die nach Suchoperation zurückgegeben werden sollen, sind ebenfalls anzugeben. Nachdem jeder Eintrag eine Vielzahl von Attributen enthalten kann und diese außerdem von einem Eintrag zum nächsten variieren können, ist es essentiell die gewünschten Attribute anzugeben. Es verringert einerseits die Datenmengen, die über das Netzwerk transportiert werden, außerdem wird die programmtechnische Auswertung des Ergebnisses vereinfacht, da man nicht zusätzlich noch eine lokale Filterung benötigt und nicht über viele, unnötige Attribute iterieren muss.

7. Accesscontrol

Sicherheit und Zugriffsrechte sind heute ein sehr heikler Punkt und sollten immer mit äußerster Sorgfalt überlegt und eingesetzt werden. Obgleich Sicherheit unbequem ist und daher von vielen Personen in einem Unternehmen als Behinderung angesehen wird, sollte man trotzdem keinesfalls die Sicherheitsspielregeln "aufweichen". Das kann manchmal verheerende Folgen haben, wie ein stunden− oder tagelanger Stillstand eines oder mehrere Systeme, die vielleicht lebensnotwendig für ein Unternehmen sind. Sicherheitsspielregeln, wenn sie gut überlegt sind, sind keine Behinderung. Der Mensch ist außerdem erziehungsfähig. Man fährt bei Nebel auf einer Autobahn nicht mit 150 Stundenkilometern, auch wenn man bei Sonnenschein gewohnt ist dort so schnell zu fahren.

Es ist im LDAPv3 derzeit noch kein Standard für eine einheitliche Kontrolle und Konfiguration von Accesscontrol vorhanden, eine Arbeitsgruppe innerhalb der IETF (Internet Engineering Task Force) ist jedoch bereits im Erarbeiten eines Drafts. Nichts desto trotz stellen alle LDAP Server, wenn auch nicht einheitlich konfigurierbar, folgende Zugriffsmöglichkeiten zur Verfügung, die auf Basis eines DN’s zugewiesen werden können, das heißt, es ist möglich gewissen Personen, oder Gruppen von Personen, die im Directory selbst definiert sind, gewisse Rechte auf Directoryebenen und Zweige zu geben:

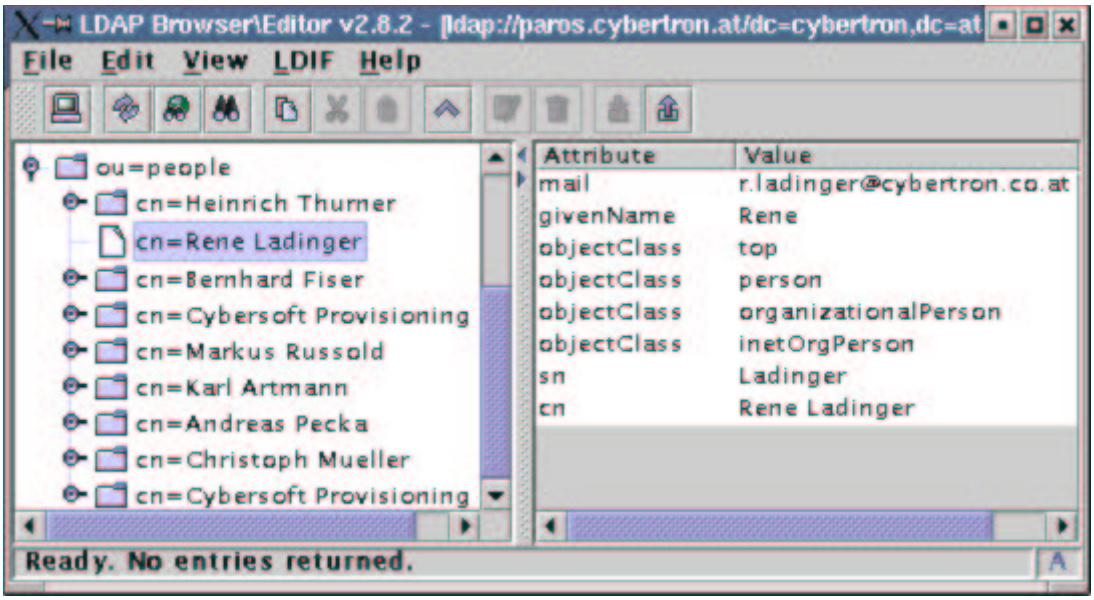
- **read**, Leserecht.
- **write**, Schreibrecht. Darin impliziert ist hinzufügen (add), bearbeiten (modify) und löschen (delete).
- **search**, Suchrecht. Das erlaubt einem Client nach einem Eintrag zu suchen. Um die gefundenen Einträge zu lesen benötigt er auch noch das Leserecht.
- **compare** erlaubt es dem Client einen Eintrag mit einem anderen zu vergleichen.
- **selfwrite** erlaubt dem Client seine eigenen Daten zu bearbeiten.
- **add** erlaubt das Hinzufügen von Einträgen.
- **proxy** erlaubt einem User stellvertretend für einen anderen User Operationen auszuführen.

8. Clients

Es existiert eine Vielzahl verschiedener Programme, die die Fähigkeit haben, auf ein Directory zuzugreifen. Das können einerseits spezielle LDAP Browser sein, die direkt der Datenmanipulation von Directories dienen, andererseits verschiedene Clientprogramme mit speziellen Ausrichtungen wie beispielsweise Emailprogramme die auf ein "Adressbuch" in einem LDAP−Directory zugreifen können. Zu diesen gehören unter anderen eine Vielzahl von Freeware Programmen aber auch die klassischen, kommerziellen Softwarepakete wie Netscape 4.x und höher und Microsoft Outlook. Weiters basiert das Windows Domainkonzept (nicht zu verwechseln mit einer Internet DNS Domain), welches von Microsoft mit Windows 2000 eingeführt wurde, mit dem sogenannten *Active Directory* ebenfalls auf LDAP, und nicht mehr dem alten, sehr umständlichen, Windows NT Domainkonzept. Obwohl mit einem LDAPv3 Browser auf die Basisfunktionen zugegriffen werden kann, hat sich Microsoft, leider nicht an alle Standards gehalten. Auch mit

Webbrowser wie beispielsweise dem *Konqueror* aus dem Open Source Projekt KDE oder dem *Netscape 4.x* und höher kann man auf Directories mit Hilfe des LDAP URL−Formats [2] auf Directories zugreifen. Im *Konqueror* sind alle Funktionen vollständig implementiert, da dieser im Hintergrund über das command line interface *ldapsearch* auf das Directory zugreift. Im Netscape wurde die Funktion integriert, jedoch nur rudimentär. Als LDAP Browser möchte ich den LDAP Browser V2.8 [7] von Jarek Gawor der Universität von Chicago vorstellen, ein sehr brauchbares Managementtool im Umgang mit Directories, welches in Java programmiert wurde. Es sind alle Funktionen wie Suche nach Einträgen, Einträge löschen, hinzufügen, bearbeiten und sogar Eingabemasken für gewisse Schemas implementiert. Das Programm hat außerdem ein LDIF Interface für den Import und Export von Daten aus und in ein Directory. LDAP Protokollstandards V2, V3 und SSL wird unterstützt. Abbildung 11 zeigt einen Screenshot eines Directories mit LDAP Browser V2.8.

Abbildung 11. Screenshot LDAP Browser V2.8



9. Literaturverzeichnis

[1]
LDAP, Programming with Java.
R. Weltman, T. Dahbura. November 2000.

[2]
RFC3377, Lightweight Directory Access Protocol (v3)
Technical Specification. J. Hodges, R. Morgan. September 2002.

[3]
ISO/IEC 9075, Information Technology − Database Languages − SQL.
International Organization for Standardization.1999.

[4]
RFC2256, A Summary of the X.500(96) User Schema for use with LDAPv3.
M. Wahl. December 1997.

[5]
http://ldap.akbkhome.com/
Ein Verzeichnis standardisierter LDAP Schemas.

[6]
RFC2849, The LDAP Data Interchange Format (LDIF).
Technical Specification. G. Good. June 2000.

[7]
LDAP Browser V2.8.2, http://www.iit.edu/~gawojar/ldap/
Jarek Gawor, University of Chicago. April 2001.

[8]
http://www.alvestrand.no/objectid/
Ein Verzeichnis aller standardisierter OID's.