Database Modeling MSc

dr. Kovács László, 2020

2. LDAP database, LDAP API

Essential knowledge units

1. LDAP model, LDAP database
2. Architecture of LDAP database
3. Elements of LDAP schema
4. LDIF schema language
5. LDIF nodeset expressions
6. LDIF access control commands
7. OpenLDAP commands
8. LDAP API architecture, classes
9. Query and update in Java LDAP API
10. Conversion of ER model into LDIF schema

Practical objectives

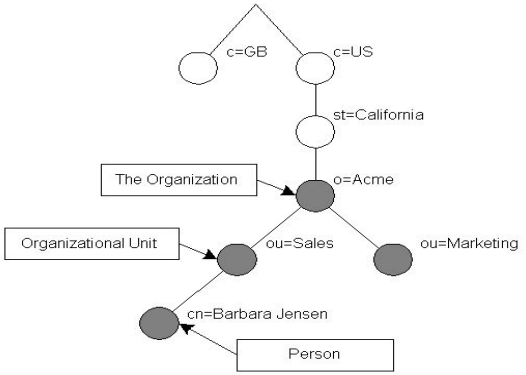
1. development of LDAP schema
2. usage of LDAP clients
3. development of LDIF schemas
4. programming Java LDAP API
5. development of LDAP DB from ER

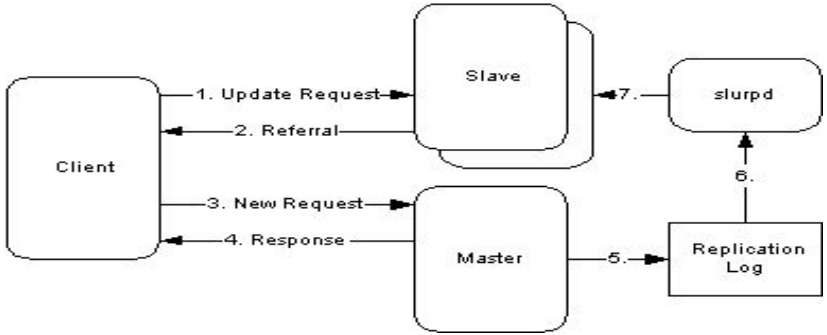
development: JDeveloper, OpenLDAP

1. LDAP model, LDAP database

Lightweight Directory Access Protocol

It is based on the X.500 protocol(1992)
Directory service (RFC1777) (no DBMS)
Hierarchical data model
Simple data structure
Optimized for Read
No transaction management
Client - server structure
Distributed data storage
ACL based security
Simple operations



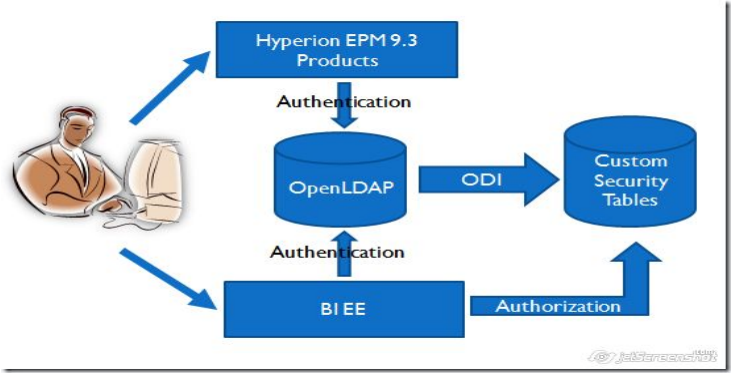


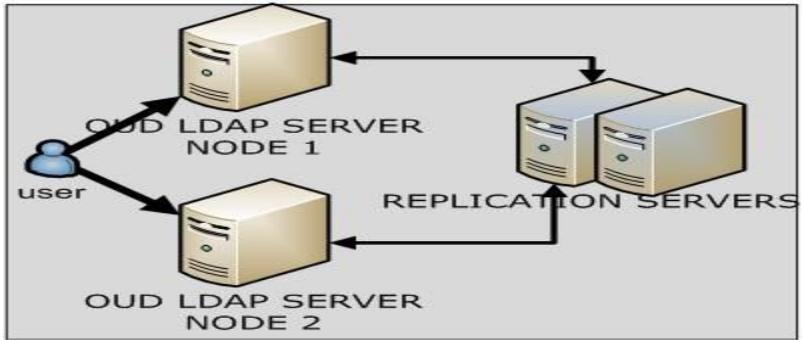
1.b LDAP model, LDAP database

Application areas:

- common directory information base
- centralised user id / password management
- resource repository
- employee repository
- configuration repository
- device catalog

data repository: it stores typed and ordered information about objects





1.c LDAP model, LDAP database

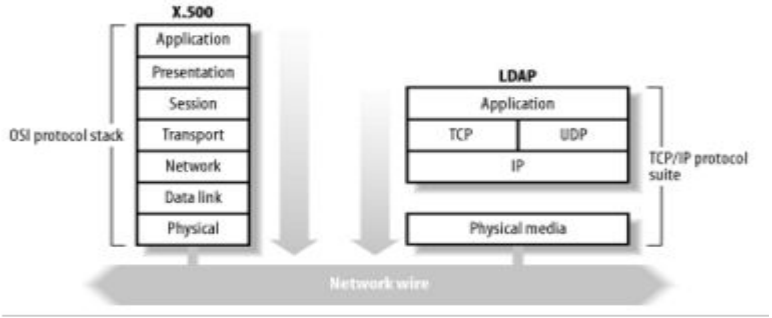
LDAP:

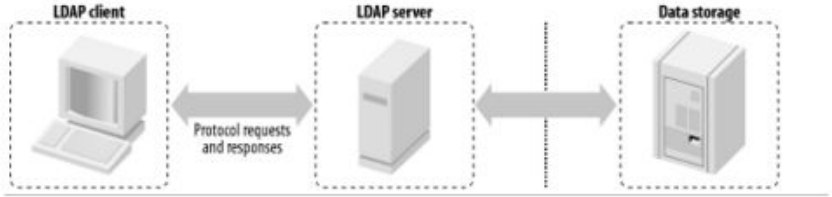
it is based on a light network suite

It uses a separate layer for schema and physical storage

Components of the LDAP model:

- informational
- naming
- functional
- security
- LDIF language
- external utilities





2. Architecture of LDAP database

Client - multi Server structure

One logical server

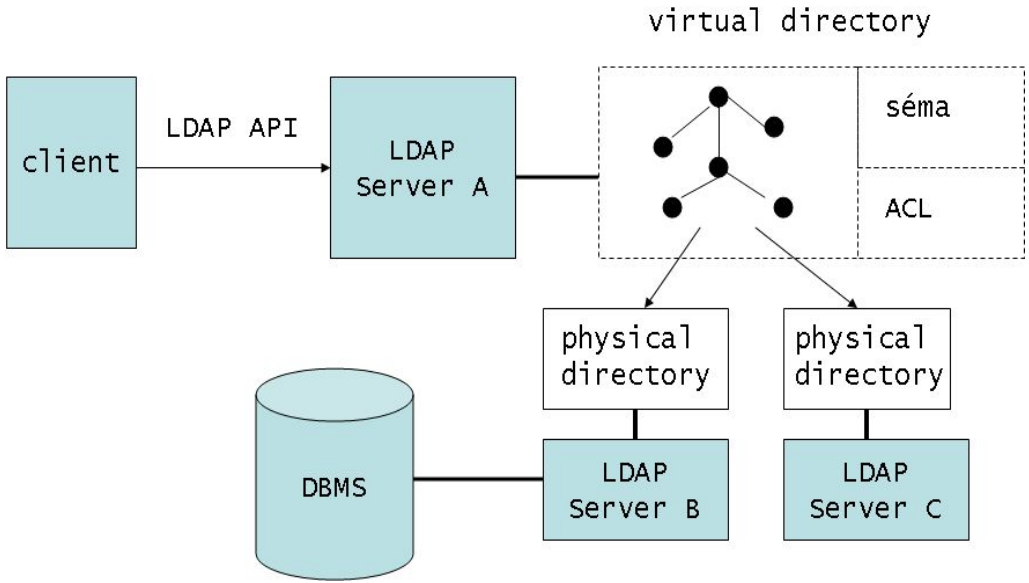
Virtual object repository

Separate physical databases

Distributed data storage

Supports different types of data storages

Schema support



2.b Architecture of LDAP database

The connection is message-based

Separate messages (stateless)

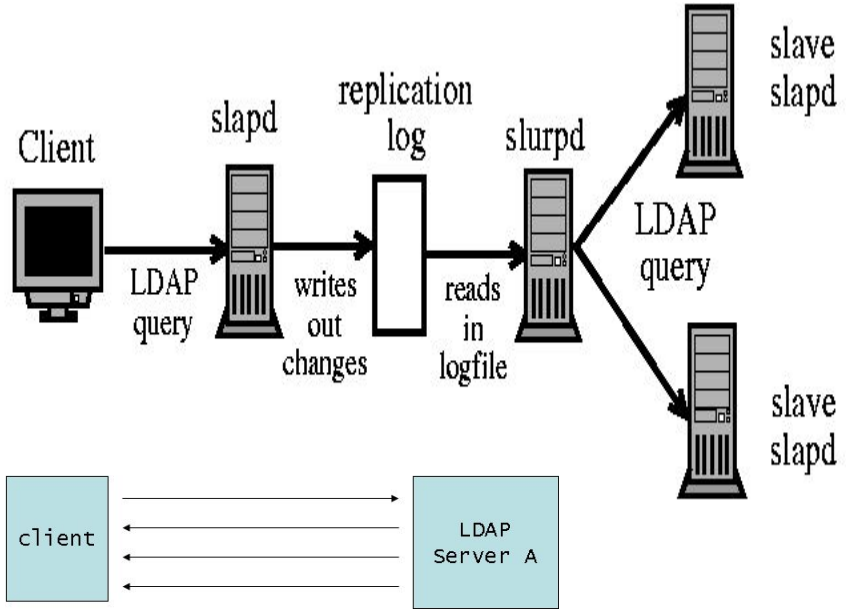
several parallel messages from the same client

different types of messages

DCL: connect, bind, unbind, abandon

DML: add, delete, modify

DQL: search, compare



3. Elements of LDAP schema

LDAP schema:

very flexible hierarchy structure

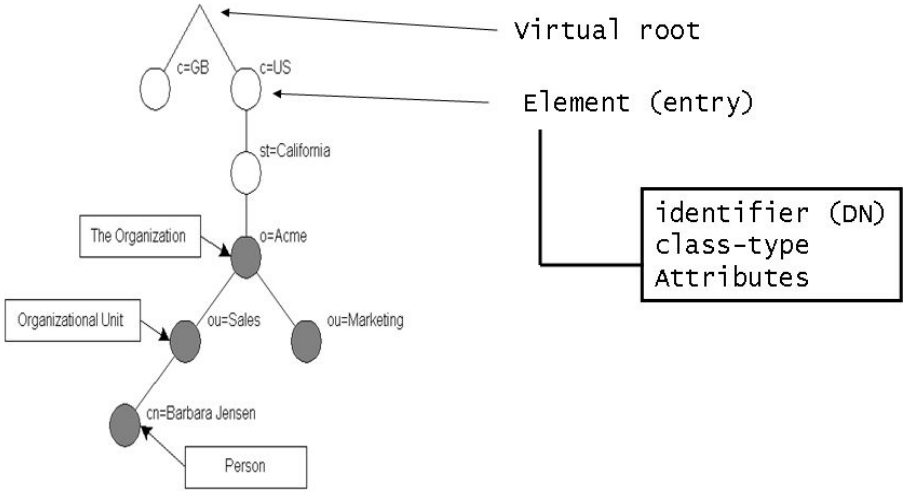
a node is a simple record having several (attribute + value) pairs

there is a key attribute (+ value)

simple schema control

identification of node: access path

DN: distinct node name
(reverse access path)
RDN: relative DN



dn: dc=iit,dc=uni-miskolc,dc=hu
dn: o=ab,dc=iit,dc=uni-miskolc,dc=hu

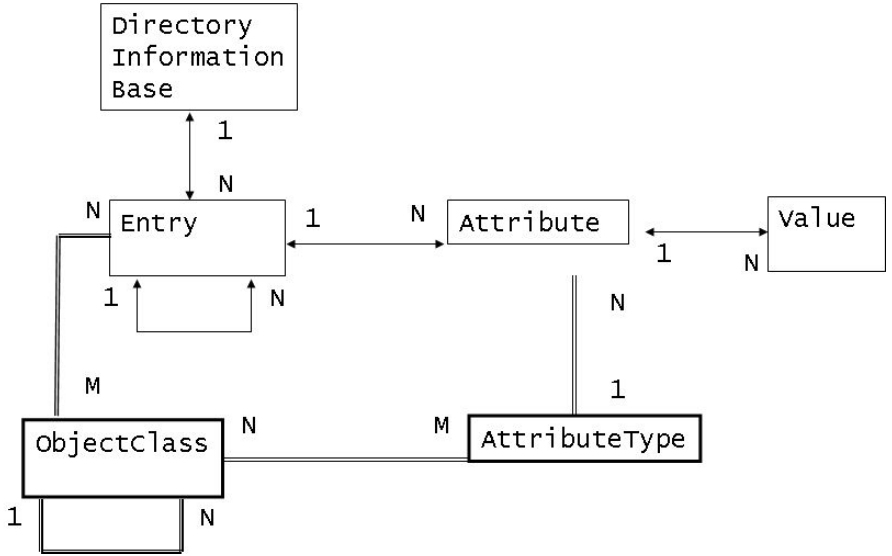
3.b Elements of LDAP schema

Logical and physical structure

Logical part:
- hierarchy
- node (entry)
- attribute + value

schema: object
- objectclass
- attributetype
- no constraint on child schema

Physical part:
- text file
- databases



3.c Elements of LDAP schema

Special kinds of entries: Referral and Alias

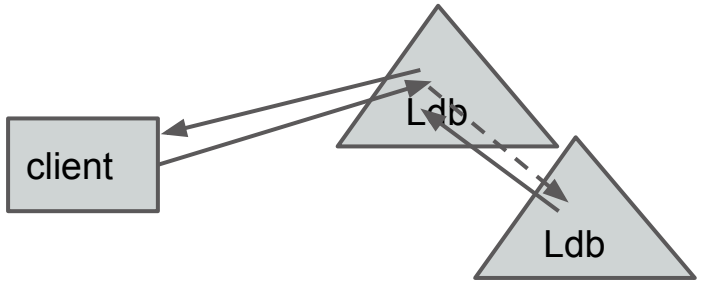
Referral:

the entry contains a reference to another entry point (at the local or remote server)
Inter-LDAP server Jump

the reference is resolved by the client or by the server (chaining)

Alias:

a link to another entry within the same entry tree




```
dn: o=grommets,dc=example,dc=com
objectClass: referral
objectClass: extensibleObject
o: grommets
ref: ldap://ldap2.example.com/o=grommets,dc=net

dn: uid=jschmidt,ou=people,dc=example,dc=de
objectClass: alias
objectClass: extensibleObject
uid: jschmidt
aliasedObjectName:
uid=jschmidt,ou=people,dc=example,dc=kr
```

4. LDIF schema language

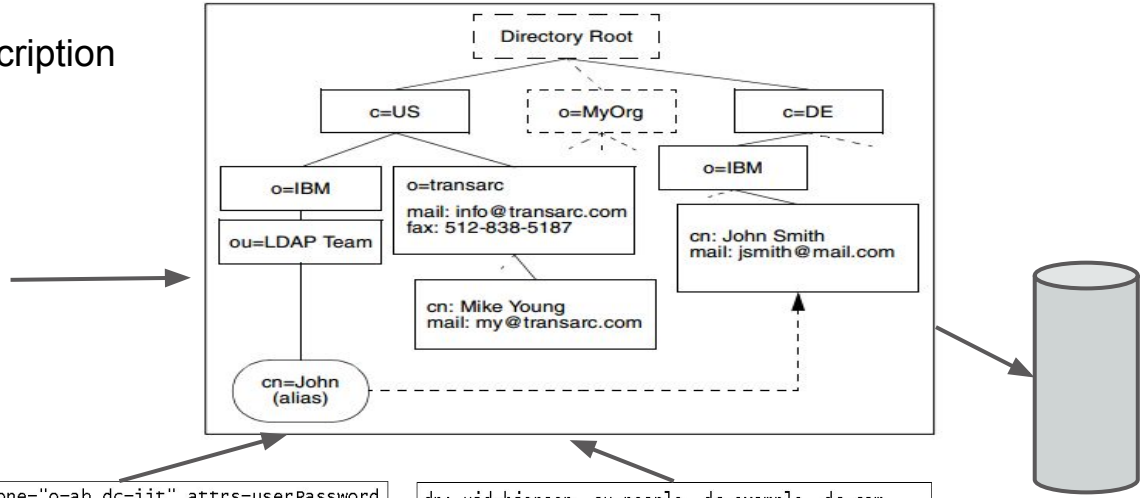
LDIF : schema and command description language for LDAP

schema description

ACL commands

DML commands

Query commands




```
access to dn.one="o=ab,dc=iit" attrs=userPassword
  by self write
  by anonymous auth
  by dn.base="dc=iit" write
  by * none
access to dn.subtree="sn=KOL,o=ab,dc=iit"
       by dn.base="sn=KOL,o=ab,dc=iit" write
access to dn.subtree="sn=BRJ,o=ab,dc=iit"
       by dn.base="sn=BRJ,o=ab,dc=iit" write
access to *
       by self write
       by users read
       by anonymous auth
```

```
dn: uid=bjensen, ou=people, dc=example, dc=com
objectclass: top
objectclass: person
cn: Barbara Jensen
givenName: Barbara
sn: Jensen
uid: bjensen
mail: bjensen@example.com
description: Manager, Switching Products Division
```

4.b LDIF schema language

Attribute type specification:

- OID
- name
- data type
- single value or multi valued
- value constraint
- description
- parent attribute type (inheritance)
- equality mode
- ordering mode

```
attributetype (
        2.5.4.6
        NAME ( 'c' 'countryName' )
        DESC 'RFC2256: ISO-3166 country 2-letter code'
        SUP name
        SINGLE-VALUE )

attributetype ( 2.5.4.16 NAME 'postalAddress'
        DESC 'RFC2256: postal address'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )
```

Value being represented	H-R	OBJECT IDENTIFIER
ACI Item	N	1.3.6.1.4.1.1466.115.121.1.1
Access Point	Y	1.3.6.1.4.1.1466.115.121.1.2
Attribute Type Description	Y	1.3.6.1.4.1.1466.115.121.1.3
Audio	N	1.3.6.1.4.1.1466.115.121.1.4
Binary	N	1.3.6.1.4.1.1466.115.121.1.5
Bit String	Y	1.3.6.1.4.1.1466.115.121.1.6
Boolean	Y	1.3.6.1.4.1.1466.115.121.1.7
Certificate	N	1.3.6.1.4.1.1466.115.121.1.8
Certificate List	N	1.3.6.1.4.1.1466.115.121.1.9
Certificate Pair	N	1.3.6.1.4.1.1466.115.121.1.10
Country String	Y	1.3.6.1.4.1.1466.115.121.1.11

4.c LDIF schema language

Object class definition:

- name
- OID
- parent type (inheritance)
- category (structural, abstract,..)
- mandatory attributes
- optional attributes
- description

```
objectclass ( 2.5.6.2 NAME 'country'
        DESC 'RFC2256: a country'
        SUP top STRUCTURAL
        MUST c
        MAY ( searchGuide $ description ) )
```

```
attributetype ( 12.12.12.7 NAME 'db'
        DESC 'darabszam'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27{128} )
attributetype ( 12.12.12.8 NAME 'lakcim'
        DESC 'lakcim'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )
##
objectclass ( 12.12.13.1 NAME 'kiado'
        DESC 'kiado'
        SUP top STRUCTURAL
        MUST ( adoszam $ nev)
        MAY ( varos ))
objectclass ( 12.12.13.2 NAME 'konyv'
        DESC 'konyv'
        SUP top STRUCTURAL
        MUST ( iksbn $ cim)
        MAY ( ar ))
objectclass ( 12.12.13.3 NAME 'ugyfel'
        DESC 'vevo ugyfel'
        SUP top STRUCTURAL
        MUST ( nev $ lakcim $ db)
```

5. LDIF nodeset expressions

Components of a selection expression

base: the root of the search subtree

scope: subset of the subtree

scope mode:

LDAP_SCOPE_BASE
LDAP_SCOPE_ONE
LDAP_SCOPE_SUBTREE

filter:

condition on parameter values

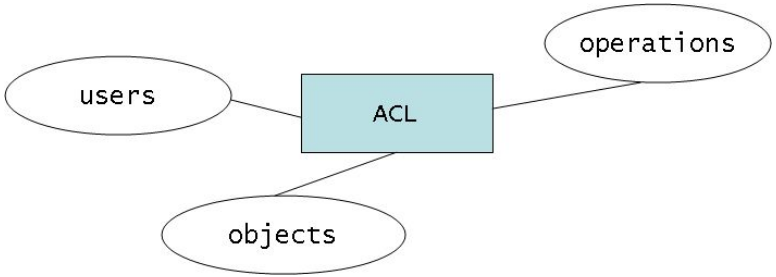
```
sample LDAP tree:
       0: o=suffix
       1: cn=Manager,o=suffix
       2: ou=people,o=suffix
       3: uid=kdz,ou=people,o=suffix
       4: cn=addresses,uid=kdz,ou=people,o=suffix
       5: uid=hyc,ou=people,o=suffix
```

```
Nodeset expressions:
       dn.base="ou=people,o=suffix"      ➔ 2
       dn.one="ou=people,o=suffix"       ➔ 3,5
       dn.subtree="ou=people,o=suffix"   ➔ 2,3,4,5
```

6. LDIF access control commands

The access control is a crucial component of the LDAP repository

there is a root who can access every data items.
ACL -based access control



ACL entry in configuration file :

TO nodes BY user operation

identification of the users:

* | self | anonymous | users | dn.regex = DN

operation codes:

none | auth | search | read | write

6.b LDIF access control commands

identification of the objects:

* | dn.base=DN | dn.one=DN | dn.subtree=DN | dn.regex = DN
filter=kif
attrs=tul.lista

standard users:
A user corresponds to an LDAP entry
this entry has a field userPassword

The user name is equal to the DN of the related entry

```
to dn.one="ou=people,o=suffix" filter=(objectClass=person)
```

```
to *   by self write
       by anonymous auth
       by * read
```

```
to dn.subtree="dc=example,dc=com" attr=homePhone
       by self write
       by dn.children=dc=example,dc=com" search
```

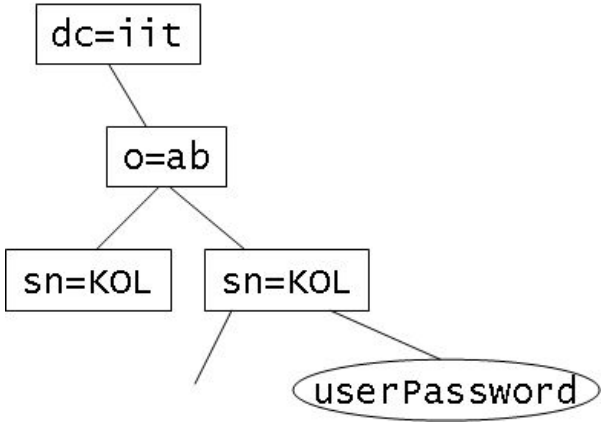
```
Konfigurációs bejegyzés:
  access to dn.base="" by * read
  access to dn.base="cn=Subschema" by * read
  access to *
       by self write
       by users read
       by anonymous auth
```

6.c LDIF access control commands

basic control operations:

1. connect / anonymous
   can read only the roor

2. connect / bind:KOL
   can read everything (except password)
   it can modify only its subtree



Login:
dn: sn=KOL,o=ab,dc=iit
pwd: KOL

Admin parameters are given in the configuration file
rootdn DN
rootpw PWD

6.d LDIF access control commands

The order of the ACL rules has an important role

The first rule in the sequence is executed

```
access to dn.one="o=ab,dc=iit" attrs=userPassword
  by self write
  by anonymous auth
  by dn.base="dc=iit" write
  by * none
access to dn.subtree="sn=KOL,o=ab,dc=iit"
       by dn.base="sn=KOL,o=ab,dc=iit" write
access to dn.subtree="sn=BRJ,o=ab,dc=iit"
       by dn.base="sn=BRJ,o=ab,dc=iit" write
access to *
       by self write
       by users read
       by anonymous auth
```

7. OpenLDAP commands

insert a new entry into the LDAP tree:

```
ldapadd -h ldap.example.com -D "cn=directory manager"
        -w secret -f updates.ldif
```

add(DN,attvallist)
DN: entry DN
Attvallist: the description of the attributes

query in the LDAP tree:

```
ldapsearch -h ldap.example.com -s base -b
"uid=bjensen,ou=people,dc=example,dc=com" "(objectclass=*)"
```

search(base-DN,scope, attlist, filter, limits)
base-DN: root of the subtree
scope: search level (base, onelevel, sub)
attlist: required attributes
filter: filter condition
limits: constraints on execution time and
size of the result

7.b OpenLDAP commands

LDAP Servers:

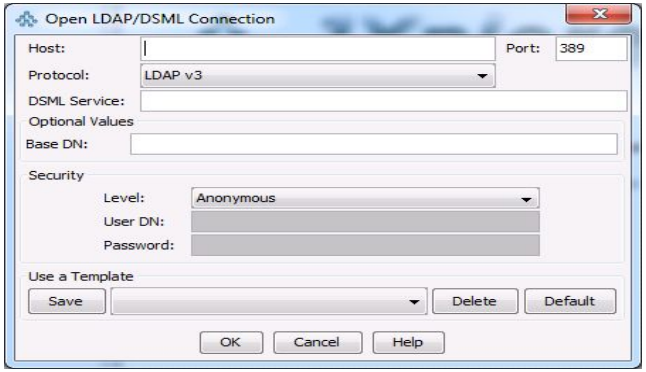
Apache Directory Server (Apache Software Foundation)

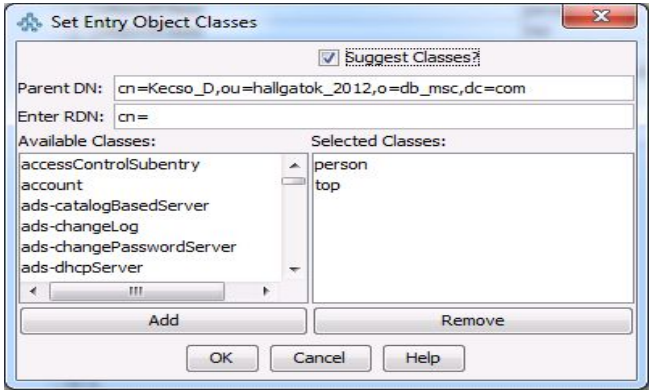
OpenLDAP (Kurt Zeilenga and others (based on Slapd))

Mandriva Directory Server

Oracle Internet Directory (Oracle)

Active Directory (Microsoft)





Ldap Clients:

Apache Directory Server/Studio - an LDAP browser and directory client

JXplorer - a Java-based browser that runs in any operating environment.

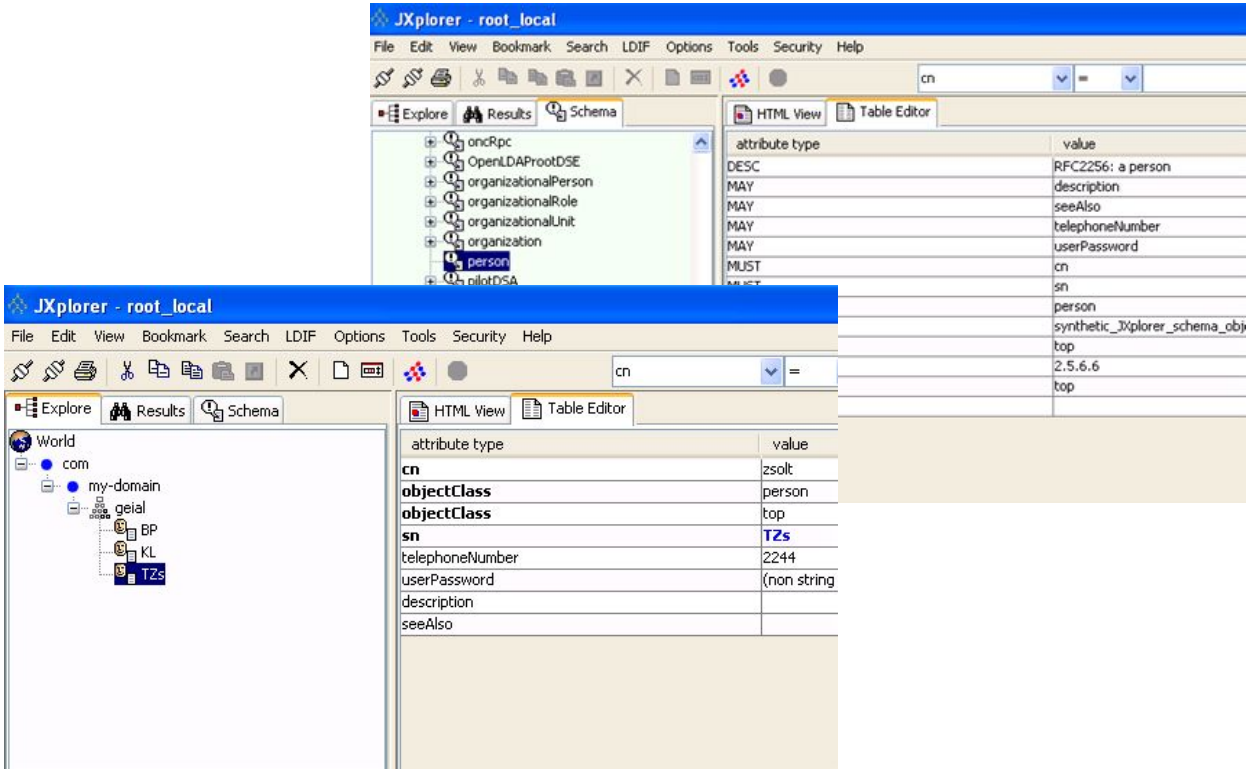
phpLDAPadmin - a web-based LDAP administration tool

Active Directory Explorer - a freeware LDAP client tool from Microsoft

7.c OpenLDAP commands

JXplorer client:

- LDAP add/delete/copy/modify
- tree copy, move and delete
- Drag-n-drop editing
- Complex searching
- UI for search filter construction
- SSL/TLS support
- SASL Authentication
- Full i18n support
- Hungarian, French and German
- Traditional and Simplified Chinese
- Unicode Support
- UTF8 allowed in DNs
- Schema support
- Supports complex DNs
- Paged results
- Extensive Help System



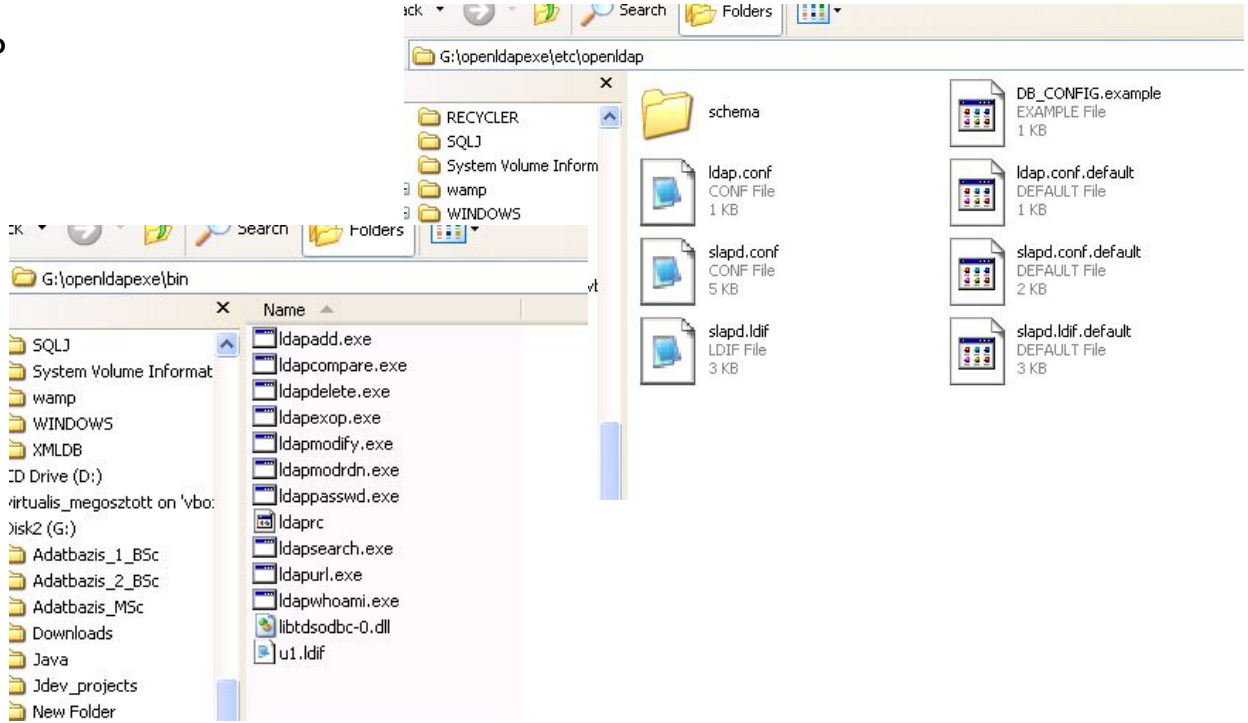
8. LDAP API architecture, classes

Installation of OpenLDAP

different data source option

config files:
- slapd.conf
- lapd.conf

schema files



8.b LDAP API architecture, classes

SLAPD.configuration file:

```
include         ../etc/openldap/schema/core.schema
include         ../etc/openldap/schema/cosine.schema
logfile         ../var/log/openldap.log
moduleload      back_bdb.la

database        bdb
suffix          "dc=my-domain,dc=com"
rootdn          "cn=Manager,dc=my-domain,dc=com"
rootpw          {SSHA}ELYEVFER2s6yqWugfOzoIpJZJ09cO6dN
directory       ../var/openldap-data

access to attrs=userPassword
        by dn="cn=Manager,dc=my-domain,dc=com" write
        by anonymous auth
        by * none
```

9. Query and update in Java LDAP API

Java LDAP API : Java classes to perform operations in LDAP repository

class library:

```
import com.novell.ldap.*
```

connection class:

```
LDAPConnection lc = new LDAPConnection();
```

connect / disconnect

```
lc.connect(host, port);
lc.disconnect();
```

bind:

```
lc.bind(ldapversion, loginDN, password);
```

9.b Query and update in Java LDAP API

constraints on the operations:

```
LDAPConstraints cons = lc.getConstraints();
cons.setTimeLimit(6000);
```

search operations:

```
String searchBase = "cn=TZs,o=ME";
int searchScope = LDAPConnection.SCOPE_BASE;
String searchFilter = "(year=10)";

LDAPSearchResults searchResults =
lc.search( searchBase,searchScope,searchFilter,...);
```

9.b Query and update in Java LDAP API

```
while ( searchResults.hasMore()) {
      LDAPEntry nextEntry = null;
      try {
         nextEntry = searchResults.next();
         System.out.println("\n" + nextEntry.getDN());
         LDAPAttributeSet attributeSet = nextEntry.getAttributeSet();
         Iterator allAttributes = attributeSet.iterator();
         while(allAttributes.hasNext()) {
            LDAPAttribute attribute =
                     (LDAPAttribute)allAttributes.next();
            String attributeName = attribute.getName();
            if (attributeName.compareTo("sn")==0) {
               Enumeration allValues = attribute.getStringValues();
               String attributevalue = (String) allValues.nextElement();
               System.out.println("    " + attributeName + "  :: " + attributevalue);
            }
```

9.c Query and update in Java LDAP API

DML operations:

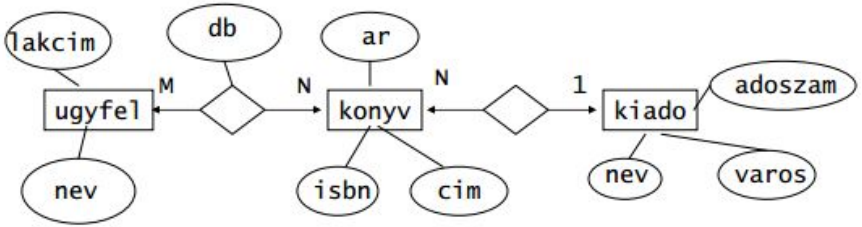
```
lc.add(entry);
```

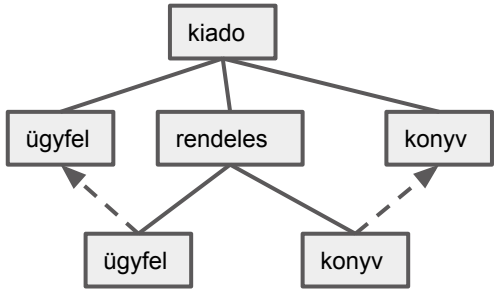
```
LDAPAttribute attribute = null;
String cn = "Toth Anna";
attribute = new LDAPAttribute("cn", cn);
attributeSet.add(attribute);

String dn = "cn=TothA," + containerName;
LDAPEntry newEntry = new LDAPEntry(dn, attributeSet);

lc.add(newEntry);
```

10. Conversion of ER into LDIF schema

Steps of implementation:

- convert ER into a hierarchical model (1:N relationships)
- develop the attribute and objectclass definitions
- create LDIF files to implement the repository





Examples

```
#attributetype ( 2.5.4.1 NAME ( 'aliasedObjectName' 'aliasedEntryName' )
#               DESC 'RFC2256: name of aliased object'
#               EQUALITY distinguishedNameMatch
#               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE )

attributetype ( 2.5.4.2 NAME 'knowledgeInformation'
                DESC 'RFC2256: knowledge information'
                EQUALITY caseIgnoreMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

# system schema
#attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' )
#               DESC 'RFC2256: common name(s) for which the entity is known by'
#               SUP name )
```

Examples

```
import com.novell.ldap.*;
import java.io.UnsupportedEncodingException;

public class LdapJ {
    public LdapJ() {
        super();
            }
    public static void main(String[] args) {
        LdapJ ldapJ = new LdapJ();
            ldapJ.ldapproba();
            }
    public void ldapproba() {
        int ldapPort = LDAPConnection.DEFAULT_PORT;
        int ldapVersion = LDAPConnection.LDAP_V3;
        String ldapHost = "localhost";
        String loginDN = "cn=Manager,dc=my-domain,dc=com";
        String password = "secret";
        String objectDN = "sn=BP,ou=geial,dc=my-domain,dc=com";
        String testPassword = "BP1";
        LDAPConnection lc = new LDAPConnection();
```

Examples

```
try {
    // connect to the server
    lc.connect( ldapHost, ldapPort );
    System.out.println("S1");
    // authenticate to the server
    lc.bind( ldapVersion, loginDN, password.getBytes("UTF8") );

    System.out.println("S2");

     LDAPAttribute attr = new LDAPAttribute(
                                "userPassword", testPassword );
    System.out.println("S3");
    boolean correct = lc.compare( objectDN, attr );
    System.out.println("S4");

    System.out.println( correct ? "The password is correct.":
                               "The password is incorrect.\n");
```

Examples

```
        // disconnect with the server
            lc.disconnect();
         }
         catch( LDAPException e ) {
            if ( e.getResultCode() == LDAPException.NO_SUCH_OBJECT ) {
               System.err.println( "Error: No such entry" );
            } else if ( e.getResultCode() ==
                                 LDAPException.NO_SUCH_ATTRIBUTE ) {
               System.err.println( "Error: No such attribute" );
            } else {
               System.err.println( "Error: " + e.toString() );
            }
         }
         catch( UnsupportedEncodingException e ) {
                  System.out.println( "Error: " + e.toString() );
         }
         System.exit(0);
            }
}
```