HowTo ASIX LDAP

Lightweight Directory Access Protocol

Curs 2020 -2021

Descripció dels aprenentatges

Bàsic

1. Instal·lar una base de dades LDAP (de cara a barraca!)
    a. Esborrar la BD d'exemple tant la configuració com la base de dades.
    b. Crear un fitxer de configuració slapd.conf al nostre gust per generar la base de dades edt.og (dc=edt,dc=org). Redefinir la propietat dels elements (de la configuració i de les dades) assignant ldap.ldap.
    c. Injectar dades (populate) a baix nivell amb slapadd i amb el servidor amb ldapaddd. Problema de permisos (propietari i grup amb les accions a baix nivell).
    d. Consultar les dades a baix nivell amb slapcat. Les dades del motor de base de dades (el dimoni) i les de la base de dades dc=edt,dc=org.
2. Consulta / manipulació de les dades.
    a. Consultes ldapsearch. Base search, scope, atributs, filtres i operadors lògics.
    b. Afegir i eliminar dades amb ldapadd i ldapdelete.
    c. Actualitacions amb ldapmodify. Tractament de fitxers ldif. Modificació dels rdn amb fitxers ldif i ldapmodrdn.
    d. Ordres client ldapwhoai, ldapcompare, ldapuri, etc.
3. Creació de múltiples bases de dades.
    a. Crear múltiples bases de dades eliminant la configuració actual. generar les BD edt.org, m06.cat i exemple.com. Omplir amb dades (populate). Examinar la configuració i els directoris de dades.
    b. Consulta de dades a baix nivell amb slapcat i a alt nivell amb ldapsearch.
    c. Persistència de les dades de l'antiga base de dades.
    d. Entendre la distincio entre directori de configuracio i directoris de dades. Persistència de dades.
4. Ordres de servidor slapd.
    a. Repàs a totes les ordres de servidor: slapadd, slapcat, slappasswd, etc.
    b. Generar passwords amb slappasswd.
    c. Definició i regeneració d'index. Examinar-ne els fitxers.
5. Tractament de ACLs.
    a. Modificar els passwords dels usuaris amb ldappasswd.
    b. Modificar passwords dels usuaris amb ldapmodify.
    c. Definició de acls. Tractament de les acls i les seves regles de funcionament. Valor per defcete, combinació de regles globals i locals, clàusules what, by, acces.
    d. Examinar els valors de les acls consultant la configuració de la base de dades amb slapcat. Consultar les acls de les entitats i els atributs amb ldapacl.
    e. Modificar les acls directament modificant el motor de base de dades cn=config, amb ordres ldapmodify.

6. Configuració dinàmica del servidor.
    a. Accés al motor de la base de dades: l'element cn=config.
    b. Modificar dinàmicament els valors de rootdn i rootpw.
7. Miscel·lània.
    a. Consulta dels logs amb journalctl.
    b. Eina gràfica gq.
    c. Estudi dels schema, objectes, atributs i regles.

Intermig

8. Disseny d'un Schema propi.
    a. Definició d'atributs propis
    b. Crear Objectes Sctructurals.
    c. Crear Objectes Auxiliars.
    d. Crear un Schema propi.
    e. Implementar un Schema propi i fer el populate i l'explotació de dades
9. LDAP distribuït.
    a. Ldap distribuït: arbres i subarbres en servidors diferents.
    b. Creació de referrals bàsics.
    c. Cració de referrals amb resolució.
10. Conexions segures amb TLS
    a. Obtenció de certificats digitals.
    b. Implementació de connexions LDAP segures amb TLS.

Avançat:

11. Underconstruction!!
    a. Redundacia: productors i consumidors.
    b. Backups de les configuracions i les dades.
    c. Overlays.
12. Implementació amb altres àmbits:
    a. Apache LDAP
    b. Samba LDAP

Documentació / Recursos

Recursos:

- ❏ **web** de @edt ASIX M06-ASO, del mòdul. N'hi ha prou de posar a google ASIX-M06.
- ❏ **github edtasixm06** amb tota l'estructura per generar imatges automatitzades docker, pas a pas els servidors (i altres) que hem anat fabricant. Posar al google github edtasixm06.
- ❏ **dockerhub images**. Imatges de docker amb els exemples de sercidors, clients, etc que hem fabricat al mòdul. Posar al google dockerhub edtasixm06.

HowTo/exercicis

Documentació:

- objectius-ldap
- activitats_asix_m06_uf1_nf1_2015-2016 (fitxers exemples: dades config)

- HowTo-ASIX-draft-presentacio-Docker.pdf
- M06-UF1NF1-Introduccio_a_LDAP
- HowTo-ASIX_LDAP_2016-2017

- Documentació: Mastering Openldap
- Fitxers d'exemples per crear BD LDAP
- Documentació: Howto-ASIX-2_LDAP

Containers:

Curs 2020-2021

- edtasixm06/ldap20:base
- edtasixm06/ldap20:editat
- edtasixm06/ldap20:acl
- edtasixm06/ldap20:schema
- edtasixm06/ldap20:entrypoint

Github

- https://github.com/edtasixm06/ldap20.git

Conceptes generals de LDAP

Consultar el document de presentació pdf "M06-UF1NF1-Introduccio_a_LDAP".
Conceptes clau de configuració:

- El fitxer **slapd.conf** conté la configuració per generar la base de dades, però és un sistema deprecated.
- El directori **/etc/openldap/slapd.d** és on es desa la configuració en format d'estructura de directoris ldiff. Es transforma amb slaptest del fitxer de configuració al directori.
- El directori **/var/lib/ldap** és on es desa per defecte la informació, les dades de la base de dades. Si hi ha vàries bases de dades cal un directori diferent per a cada una.
- Es pot consultar (**slapcat**) i inserir dades massivament en offline (**slapadd**) amb el servei apagat.
- Per usar ordres LDAP client cal que el servei estigui engegat, i cal connectar al servidor **localhost** (a l'aula per defecte les ordres client consulten al servidor d'informàtica).

Concepte claus de funcionament:

- Cal assegurar-se que els permisos del directori /etc/openldap/slapd.d (i subarbre) i els de /var/lib/ldap (i subarbre) siguin propietat de l'usuari i grup ldap, és a dir **ldap.ldap**.
- Les dades es poden assignar a la base de dades amb ordres de servidor (slapadd) i amb ordres de client (ldapadd), que tenen característiques diferents.
- Sempre que es modifiqui la configuració del directori /etc/openldap/slapd.d o de les dades de /var/lib ldap usant l'ordre slapadd **cal** tornar a assignar la propietat d'usuari i grup a ldap.ldap.
  És a dir, si es realitzen accions de root sobre la configuració o les dades (amb el servei apagat) cal restaurar el proipetari i grup a ldap.ldap.
- La càrrega de dades intensiva (inicial per exemple) es fa accedint directament al backend (amb el servei offline) usant l'ordre slapadd.
- Amb el servei engegat es poden afegir dades a la base de dades amb l'ordre LDAP client **ldapadd**. La comunicació és del client al servidor usant el protocol **LDAP**.
- Assegurar-se que les ordres client consulten al servidor LDAP apropiat, si no s'indica a les aules d'informàtica es consulta al servidor d'informàtica (gandhi). Es recomana usar l'opció "**-h localhost:389**" per forçar l'ordre a consultar el propi servidor.

Instal·lació i creació d'una BD (edt.org)

L'objectiu d'aquest apartat es instal·lar el servidor, eliminar la BD per defecte que incorpora i generar-ne una corresponent a **"edt.org"**, que s'identifica com a **"dc=edt,dc=org"**. Carregar les dades en offline (servidor apagat) de la organització edt.org. Restaurar el propietari i grup a ldap.ldap per poder engegar el servei slapd. Un cop engegat el servei carregar les dades d'usuaris de edt.org i fer-ne consultes ldapsearch.

Descripció del procediment d'instal·lació:
Part-A

- Instal·lar els paquets de openldap-clients i openldap-servers
- Esborrar la base de dades d'exemple que porta incorporada (myexample.com) i el directori de dades.
- Afegir el fitxer DB_CONFIG (que tristament hem esborrat) al direcori /var/lib/ldap. El trobareu en el paquet openldap-servers.
- Verificar el fitxer de dades slapd.conf i generar a partir d'aquest fitxer el directori de configuració /etc/openldap/slapd.d. Verificar el directori creat i observar l'estructura de dades de directoris ldiff que s'ha generat.
  nota podeu consultar el fitxer de configuració a l'apartat de configuració.
  nota en usar l'ordre slaptest mostra warnings perquè verifica l'existència de la base de dades i les dades quan encara no existeixen. Un cop feta l'ordre es pot usar **slaptest -u** i si no mostra errors ni warnings és pot continuar.

- Fer un dump de la base de dades generada observant la base de dades {1}, el monitor {2} (si s'ha configurat) i el propi servei del dimoni slapd {0}.
- *En aquest punt s'ha generat la configuració i l'estructura per a una BD anomenada **dc=edt,dc=org**, pero no conté cap element, cap entrada, ni tan sols el node arrel. S'ha preparat tot, però encara no hi ha cap dada*.

Part-B

- Carregar les dades de la organització "edt.org". Es carrega el node arrel i tres ou (unitats d'organitzacio), però no es carreguen dades d'usuaris (es farà posteriorment).
  Aquestes dades es carregen amb el servei apagat usant slapadd que accedeix directament a baix nivell als fitxers del backend.
  Observar el directori de dades /var/lib/ldap.
- Restaurar els permisos apropiats de propietari i grup a ldap.ldap del directori de configuració (/etc/openldap/slapd.d) i del directori de dades (/var/lib/ldap).
- Engegar el servei slapd i verificar que està en marxa.
- Consultar el contingut de la base de dades "dc=edt,dc=org" de l'organització "edt.org". Usar tant la comanda client ldapsearch com la comanda servidor slapcat.

Part-C

- Afegir dades a la base de dades amb l'ordre client ldapadd. Concretament s'afegiran usuaris a la "ou=usuaris".
- Verificar les dades llistant-les amb consultes client ldasearch, i també fer un volcat de les dades amb slapcat.

** no copieu aquestes linies d'ordres amb copy+paste a la consola perquè la codificació de caràcters genera guins i espais incorrectes. Sembla que estigui bé però no ho està!

Part-A

```
# yum -y install openldap-servers openldap-clients
# rm -rf /etc/openldap/slapd.d/*
# rm -rf /var/lib/ldap/*
# cp  /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG

# slaptest -v  -f slapd-edt.org.conf
# slaptest -v  -f slapd-edt.org.conf  -F /etc/openldap/slapd.d
# slaptest -v  -f slapd-edt.org.conf  -F /etc/openldap/slapd.d  -u
# slaptest -v -F /etc/openldap/slapd.d  -u

# slapcat
# slapcat -n1
# slapcat -n0
```

Part-B

```
# slapdd -F /etc/openldap/slapd.d  -l organitzacio_edt.org.ldiff
# ls /var/lib/ldap

# chown -R ldap.ldap /etc/openldap/slapd.d
# chown -R ldap.ldap /var/lib/ldap

# systemctl start slapd
# systemctl status slapd

# ldapsearch -x -LLL -h localhost:389 -b 'dc=edt,dc=org'
# slapcat
# slapcat  -n1 | less
```

Part-C

```
# ldapadd -x -h localhost:389 -D 'cn=Manager,dc=edt,dc=org'  -w secret
-f usuaris_edt.org.ldif

# ldapsearch -x -LLL -h localhost:389 -b 'dc=edt,dc=org'
# slapcat
# slapcat  -n1 | lesss
```

Ara la base de dades ja conté un node arrel (edt.org), els nodes d'unitat d'organitzacio ou (maquines, clients, productes i usuaris) i un conjunt d'usuaris. Usant ordres client LDAP es poden consultar i actualitzar les dades de la BD.

Consultes ldsapsearch

Atenció, cal indicar el servidor a consultar amb l'opció **-h localhost:389**.
Elements de les consultes:

- base de la consulta (en SQL sería la cláusula FROM)
- indicar els camps a llistar (en SQL seria els camps del select) . Hi ha els operadors 1.1, * i +. Diferència entre atributs d'usuari i operacionals.
- establir condicions de filtrat (en SQL sería la cláusula where)
- scope o àmbit de les consultes: base, one, sub, children
- operadors: and &, or |, not ! i els relacionals habituals.

Opcions generals de les consultes:

-x per indicar que el tràfic LDAP no es xifrat, és en text pla
-LLL per obtenir respostes 'planes', sense capçaleres ni comentaris, només dades de resposta.
-h host:port per indicar el host i el port al que es vol consultar. Atenció que cal indicar-ho perquè el client ldap per defecte a les aules d'informàtica està configurat per accedir al servidor d'informàtica (gandhi).

Exemple (zoom) de consulta amb filtres and i or:

' (& (| (cn=* Mas) (cn=* Pou)) (gidNumber=600)) '

```
# ldapsearch -x  -h localhost:389 -b 'dc=edt,dc=org'
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org'

# ldapsearch -x -h localhost:389 -LLL -b 'ou=productes,dc=edt,dc=org'
# ldapsearch -x -h localhost:389 -LLL -b 'ou=productes,dc=edt,dc=org' 1.1
# ldapsearch -x -h localhost:389 -LLL -b 'ou=productes,dc=edt,dc=org' *
# ldapsearch -x -h localhost:389 -LLL -b 'ou=productes,dc=edt,dc=org' +

# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' dn cn mail
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' dn cn mail uid uidNumber

# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' 'cn=Pere Pou'  *
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' 'cn=Pere Pou'  +
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' 'cn=Pere Pou'  dn cn mail
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' 'cn=* Pou'  dn cn gidNumber

# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' 'gidNumber=600'  dn cn mail
uidNumber gidNumber
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org' '(|(cn=* Mas)(gidNumber=600))'
dn cn mail uidNumber gidNumber

# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org'  -s base
```

```
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org'  -s one
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org'  -s children
# ldapsearch -x -h localhost:389 -LLL -b 'dc=edt,dc=org'  -s sub

# ldapsearch -x -h localhost:389 -LLL -b 'ou=usuaris,dc=edt,dc=org'  -s base
# ldapsearch -x -h localhost:389 -LLL -b 'ou=usuaris,dc=edt,dc=org'  -s one
# ldapsearch -x -h localhost:389 -LLL -b 'ou=usuaris,dc=edt,dc=org'  -s children
# ldapsearch -x -h localhost:389 -LLL -b 'ou=usuaris,dc=edt,dc=org'  -s sub
```

Configuració slapd.conf

Descripció general

Exemple de fitxer de condifuracio slapd.d que crea la base de dades corresponent a la organització **"edt.org"**. En el protocol LDAP aquesta base de dades s'identifica com a **"dc=edt,dc=org"**.

En aquest fitxer podem observar clarament les seccions:

- include on es carregen els schema de dades necessaris per la BD. per exemple podem observar que es carrega l'schema de *inetorgperson*.
- directives globals del servei slapd com per exemple el *bind* per indicar que usa el protocol ldapv2.
- definició de la base de dades de l'organització "edt.org".
- [opcional] definició d'índex
- [opcional] definició d'altres bases de dades
- [opcional] activació o no de la funció de monitoritzar el servei slapd.

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/duaconf.schema
include         /etc/openldap/schema/dyngroup.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/java.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/openldap.schema
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema

# -- Global Directives
-----------------------------------------------------------------------------------
# Allow LDAPv2 client connections.  This is NOT the default.
allow bind_v2
pidfile        /var/run/openldap/slapd.pid
#argsfile    /var/run/openldap/slapd.args

# --Databse dc=edt,dc=org ----------------------------------------------------------------------------
database bdb
suffix "dc=edt,dc=org"
```

```
rootdn "cn=Manager,dc=edt,dc=org"
rootpw secret
directory /var/lib/ldap

# Indices to maintain for this database
index objectClass                       eq,pres

# ACLs for this database
access to *
    by self write
    by * read
# --end Database ------------------------------------------------------------------------------------

# -- Database monitor ---------------------------------------------------------------------------
database monitor
# allow only rootdn to read the monitor
access to *
          by dn.exact="cn=Manager,dc=edt,dc=org" read
          by * none
# --end Database ------------------------------------------------------------------------------------
```

La part clau del fitxer de configuració slapd.conf és la definicio de les bases de dades a emmagatzemar, hi haurà una secció database per a cada una d'elles.
nota a vegades és fàcil confondre's i escriure les ACL al final del document fora del bloc de la base de dades on realment es pretenia configurar aquestes ACLs.

Els elements principals a definir són:

```
database bdb
suffix "dc=edt,dc=org"
rootdn "cn=Manager,dc=edt,dc=org"
rootpw secret
directory /var/lib/ldap
index objectClass  eq,pres
```

- Es defineix una base de dades de tipus bdb (Berkeley database). Aquest es el backend on es desen les dades. LDAP no descriu com es desen les dades i es poden fer servir diversos backends diferents (postgres, el filesystem, etc).
- La directiva sufix defineix el nom de la base de dades. Si l'organització es "edt.org" el dn (Distinguished Name) és "dc=edt,dc=org".
- Cal definir un usuari administrador o root de la base de dades. Atenció, no és un usuari de dins de la base de dades es un usuari extern que es defineix aquí, al fitxer de configuració.
  Es defineix en dos passos, la directiva que defineix l'usuari administrador del la base de dades s'anomena rootdn, i en aquest cas és un usuari (no real ni del LDAP) anomenat "cn=Manager,dc=edt,dc=org".
  També cal assignar un password a aquest usuari, el password es defineix aqui amb la directiva rootpw i se li assigna "secret" com a password.
  En aquest cas el passwd es en text pla però es poden generar passwords xifrats,

consultar l'apartat pertinent.
En resum, es crea un usuari virtual (ni del sistema ni de la base de dades) amb drets totals d'administració d'aquesta base de dades.

- Les dades de la base de dades es desen al directori indicat per la directiva directory. Generalment es el directori /var/lib/ldap.
- Finalment cal indicar per quins elements es vol que el servei indexi les dades de la base de dades. Aqui es defineix un index per als objectClass.

Definició de ACLs:

```
access to *
   by self write
   by * read
```

Es poden definir múltiples ACLs per als elements de la base de dades fins i tot a nivell d'atributs. **Sempre** rootdn pot modificar (accedir i modificar) qualsevol contingut, per alguna cosa es root!.

El tractament de ACLs es similar als altres àmbits informàtics on es poden aplicar i com es habitual, pot acabar essent bastant complicat. En la forma més simple son fàcils d'entendre.

La ACL d'exemple indica que:

- qualsevol usuari pot modificar les seves pròpies dades.
- qualsevol usuari pot veure les dades dels altres.

Evidentment **no** és una ACL molt assenyada, però ens servirà per començar a remenar…

Reflexions sobre les dades i les ACLs:

- Un usuari pot modificar *totes* les seves dades? Puc passar de profe a Conseller d'Ensenyament amb un simple ldapmodify? Modificar la nòmina?
- Els altres usuaris quines dades poden observar? El password dels altres? L'adreça?
- Hi ha un camp que tot usuari ha de poder modificar per ell mateix, el seu propi password.

nota a vegades és fàcil confondre's i escriure les ACL al final del document fora del bloc de la base de dades on realment es pretenia configurar aquestes ACLs.

Configuració de múltiples bd

Aquest és un exemple on es configuren tres bases de dades diferents corresponents a les organitzacions "edt.org", "example.com" i "m06.cat". Cada base de dades es desa en un directori diferent dins de /var/lib utilitzant un nom tipus /var/lib/ldap.edt.org.

```
include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/duaconf.schema
include         /etc/openldap/schema/dyngroup.schema
```

```
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/java.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/openldap.schema
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema
# Allow LDAPv2 client connections.  This is NOT the default.
allow bind_v2
pidfile         /var/run/openldap/slapd.pid
#argsfile    /var/run/openldap/slapd.args

# --Databse dc=example,dc=com ----------------------------------------------------------------
database bdb
suffix "dc=example,dc=com"
rootdn "cn=Manager,dc=example,dc=com"
rootpw secret
directory /var/lib/ldap.example.com
index objectClass  eq,pres
# --end Database -----------------------------------------------------------------------------

# --Databse dc=edt,dc=org --------------------------------------------------------------------
database bdb
suffix "dc=edt,dc=org"
rootdn "cn=Manager,dc=edt,dc=org"
rootpw jupiter
directory /var/lib/ldap.edt.org
index objectClass  eq,pres
# --end Database -----------------------------------------------------------------------------

# --Databse dc=m06,dc=cat --------------------------------------------------------------------
database bdb
suffix "dc=m06,dc=cat"
rootdn "cn=Manager,dc=m06,dc=cat"
rootpw jupiter
directory /var/lib/ldap.m06.cat
index objectClass  eq,pres
# --end Database -----------------------------------------------------------------------------

# --Database Monitor -------------------------------------------------------------------------
database monitor
# allow only rootdn to read the monitor
access to *
        by dn.exact="cn=Manager,dc=example,dc=com" read
        by dn.exact="cn=Manager,dc=edt,dc=org" read
        by dn.exact="cn=Manager,dc=m06,dc=cat" read
        by * none
# --end Database -----------------------------------------------------------------------------
```

Generar password de rootdn

Per generar passwords de servidor es pot usar l'ordre de servidor slappasswd que permet interactivament generar un password en el format de xifrat indicat. Per met xifrar en varis formats com per exemple MD6 i SHA1.

```
# slappasswd -h {md5}
New password:
Re-enter new password: jupiter
{MD5}J6UUjqD73a4i2QK+qaGVMQ==

# slappasswd -h {SHA}
New password:
Re-enter new password: jupiter
{SHA}ovf8ta/reYP/u2zj0afpHt8yE1A=

# slappasswd -h {CRYPT}
New password:
Re-enter new password: jupiter
{CRYPT}ZMcZ4/x5x0KIo
```

Un cop generat el password de rootdn amb el xifrat (o text pla) a usar, es copia a la directiva de configuració rootpw de la base de dades, per exemple:

```
rootdn "cn=Manager,dc=edt,dc=org"
rootpw {MD5}J6UUjqD73a4i2QK+qaGVMQ==
```

Ordres client LDAP

ldapsearch (tractada anteriorment)
ldapadd
ldapdelete
ldapmodify
ldapmodrdn
ldappasswd
ldapwhoami
ldapurl
ldapcompare
getent

Podeu consultar l'ajuda de les ordres client ldap consultant les seves pàgines man:

```
#  man ldap
ldapadd         ldap.conf       ldapexop        ldapmodrdn   ldapsearch   ldapwhoami
ldapcompare  ldapdelete     ldapmodify    ldappasswd    ldapurl
```

Afegir i eliminar: ldapadd, ldapdelete

```
# afegir
# ldapadd -x -D cn=Manager,dc=edt,dc=org -w secret -f /tmp/ldap/usuaris1.ldiff

# esborrar
# ldapdelete -x -D cn=Manager,dc=edt,dc=org -w secret "cn=Pau
Pou,ou=usuaris,dc=edt,dc=org"

# oju!
# ldapdelete -x -r -D cn=Manager,dc=edt,dc=org -w secret "ou=usuaris,dc=edt,dc=org"
```

Modificar: ldapmodify, ldapmodrdn

```
# modificar
# ldapmodify -x -D cn=Manager,dc=edt,dc=org -w secret -f modificacions-ldif
```

Exemples de fitxer de modificacions tipus LDIF per a l'ordre *ldamodify*. Consultar **man (5) ldif**: changetype: <[modify|add|delete|modrdn]>

```
dn: cn=Pau Pou,ou=usuaris,dc=edt,dc=org
changetype: modify
replace: mail
mail: modme@example.com
-
replace: homephone
homephone: 111-222-333
-
delete: description
-
add:description
description: nova descripció per a l'usuari Pau

dn: cn=Anna Pou,ou=usuaris,dc=edt,dc=org
changetype: delete

dn: cn=Anna Pou,ou=usuaris,dc=edt,dc=org
changetype: add
objectClass: posixAccount
objectClass: inetOrgPerson
cn: Anna Pou
cn: Anita Pou
sn: Pou
mail: anna@edt.org
ou: Alumnes
uid: anna
uidNumber: 5002
gidNumber: 600
homeDirectory: /tmp/home/anna
userPassword::
e1NTSEF9Qm00QjNCdS9mdUg2QmJ5OWxneGZGQXdMWXJLMFJiT3E=
description: modified description
homePhone: 93-222-333

dn: cn=Anna Pou,ou=usuaris,dc=edt,dc=org
changetype: modify
replace: homePhone
homePhone: 93-123-456

dn: cn=Anna Pou,ou=usuaris,dc=edt,dc=org
changetype: modify
delete: mail

dn: cn=Pau Pou,ou=usuaris,dc=edt,dc=org
changetype: modrdn
newrdn: cn=Pau Maria Pou
deleteoldrdn: 0
#newsuperior: ou=usuaris,dc=edt,dc=org
```

```
# modificar rdn
# ldapmodrdn -x -D cn=manager,dc=edt,dc=org -w secret 'cn=Anna
Puig,ou=usuaris,dc=edt,dc=org' 'cn=Annita Puig'
# ldapsearch -x -b dc=edt,dc=org '(cn=Annita Puig)'
```

Miscel·lània: ldapcompare, ldapwhoami, getent

```
# comparar
# ldapcompare -x "cn=Annita Puig,ou=usuaris,dc=edt,dc=org" mail:anna@edt.org
# ldapcompare -x "cn=Annita Puig,ou=usuaris,dc=edt,dc=org" mail:annita@edt.org

# identitat
# ldapwhoami -x
# ldapwhoami -x -D "cn=Manager,dc=edt,dc=org"
# ldapwhoami -x -D "cn=Manager,dchan=edt,dc=org" -w secret

# ldapwhoami -x -D cn=Manager,dc=edt,dc=org -W
Enter LDAP Password:
dn:cn=Manager,dc=edt,dc=org

# Si no s'indica l'opció “-x” intenta comunicació xifrada SASL (no usar...)
# ldapwhoami
SASL/DIGEST-MD5 authentication started
Please enter your password:
ldap_sasl_interactive_bind_s: Invalid credentials (49)
        additional info: SASL(-13): user not found: no secret in database

# ldapwhoami -x
anonymous

# verificar si l'usuari ldap forma part dels usuaris del sistema integrats als del /etc/passwd
# getent passwd | grep anna
# getent passwd
```

Modificar el password: ldappasswd

```
# modifica el passwd:
# ldappasswd -v -x -D ‘cn=Manger,dc=edt,dc=org’ -w secret  ‘cn=Anna
Pou,ou=usuaris,dc=edt,dc=org’

# ldappasswd -v -x -D ‘cn=Anna Pou,dc=edt,dc=org’ -w secret  ‘cn=Anna
Pou,ou=usuaris,dc=edt,dc=org’

# ldappasswd -v -x  ‘cn=Anna Pou,ou=usuaris,dc=edt,dc=org’
```

Ordres LDAP de servidor: slapd

El servidor slapd s'acompanya d'un conjunt d'ordres de baix nivell que actuen directament sobre la base de dades amb el servei apagat, es tracta de les ordres:

slaptest
slapadd
slapcat
slappasswd
slapindex
slapacl
slapauth
slapdn
slapschema

slaptest

```
# slaptest -v -f /etc/openldap/slapd.conf #(test slapd.conf file)
# slaptest -v -F /etc/openldap/slapd.d #(test slapd.d directory)
# slaptest -v -F /etc/openldap/slapd.d.web #(test slapd.d-web anoder-directory)

# slaptest -v -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d -u
#(generate slapd.d tree and no check DB, doesen't exist)
# slaptest -v -F /etc/openldap/slapd.d -n0 -l file-slapd.ldif #(generate DB using .ldif)
```

slapcat

```
# slapcat -v -n0      #lists cn=config DB
# slapcat  -n nº      #lists nº DataBase. Shuld contain entrys.
# slapcat             #lists default DB (usually the first suffix defined)
# slapcat   -n1       #lists DB {1} usually the defaut
```

slapadd

```
# slapdd -F /etc/openldap/slapd.d  -l organitzacio_edt.org.ldiff
```

slappasswd

```
# slappasswd -h {md5}
New password:
Re-enter new password: jupiter
{MD5}J6UUjqD73a4i2QK+qaGVMQ==

# slappasswd -h {SHA}
New password:
Re-enter new password: jupiter
{SHA}ovf8ta/reYP/u2zj0afpHt8yE1A=

# slappasswd -h {CRYPT}
New password:
Re-enter new password: jupiter
{CRYPT}ZMcZ4/x5x0KIo
```

slapindex

Al directori de dades /var/lib/ldap hi ha les dades de la base de dades i també els índex que s'han generat. Els índex a realitzar es descriuen en la directiva índex de fitxer de configuració slapd.

```
database bdb
suffix "dc=edt,dc=org"
rootdn "cn=Manager,dc=edt,dc=org"
rootpw jupiter
directory /var/lib/ldap
index objectClass eq,pres
index cn,sn,mail  eq,pres
access to * by self write by * read
```

Podeu consultar la documentació de la directiva *index* en la pàgina de manual **man (5) slapd-bdb**.

index {<attrlist>|default} [pres,eq,approx,sub,<special>]

Specify the indexes to maintain for the given attribute (or list of attributes). Some attributes only support a subset of indexes. If only an <attr> is given, the indices specified for default are maintained. Note that setting a default does not imply that all attributes will be indexed. Also, for best performance, an **eq** index should always be configured for the **objectClass** attribute.

A number of special index parameters may be specified. The index type **sub** can be decomposed into **subinitial**, **subany**, and **subfinal** indices. The special type **nolang**

may be specified to disallow use of this index by language subtypes. The special type **nosubtypes** may be specified to disallow use of this index by named subtypes.

Note: changing index settings in slapd.conf(5) requires rebuilding indices, see slapindex(8); changing index settings dynamically by LDAPModifying "cn=config" automatically causes rebuilding of the indices online in a background task.

Si es llista el contingut del directori /var/lib/ldap es poden veure fitxers de índex. Aquests fitxers es poden regenerar (degut per exemple a que s'ha fet una càrrega massiva via slapadd o degut a que s'han corromput). Es poden regenerar tots o un índex en concret.

```
# ls /var/lib/ldap
alock  cn.bdb  __db.001  __db.002  __db.003  DB_CONFIG  dn2id.bdb  id2entry.bdb
log.0000000001  objectClass.bdb  sn.bdb mail.bdb

# file /var/lib/ldap/*
/var/lib/ldap/alock:              data
/var/lib/ldap/cn.bdb:             Berkeley DB (Btree, version 9, native byte-order)
/var/lib/ldap/__db.001:           Applesoft BASIC program data
/var/lib/ldap/__db.002:           data
/var/lib/ldap/__db.003:           data
/var/lib/ldap/DB_CONFIG:          ASCII text
/var/lib/ldap/dn2id.bdb:          Berkeley DB (Btree, version 9, native byte-order)
/var/lib/ldap/id2entry.bdb:       Berkeley DB (Btree, version 9, native byte-order)
/var/lib/ldap/log.0000000001:  Berkeley DB (Log, version 18, native byte-order)
/var/lib/ldap/objectClass.bdb: Berkeley DB (Btree, version 9, native byte-order)
/var/lib/ldap/sn.bdb:             Berkeley DB (Btree, version 9, native byte-order)
/var/lib/ldap/mail.bdb:           Berkeley DB (Btree, version 9, native byte-order)

# slapindex  -v
indexing id=00000001
indexing id=00000002

# slapindex  mail
```

slapacl

extret del man slapacl

slapacl - Check access to a list of attributes.

slapacl is used to check the behavior of slapd(8) by verifying access to directory data according to the access control list directives defined in its configuration. It opens the slapd.conf(5) configuration file or the slapd-config(5) backend, reads in the access/olcAccess directives, and then parses the attr list given on the command-line; if none is given, access to the entry pseudo-attribute is tested.

slapauth

extret del man slapauth

slapauth - Check a list of string-represented IDs for LDAP authc/authz

Slapauth is used to check the behavior of the slapd in mapping identities for authentication and authorization purposes, as specified in slapd.conf(5). It opens the slapd.conf(5) configuration file or the slapd-config(5) backend, reads in the authz-policy/olcAuthzPolicy and authz-regexp/olcAuthzRegexp directives, and then parses the ID list given on the command-line.

slapdn

extret del man slapdn

slapdn - Check a list of string-represented LDAP DNs based on schema syntax

Slapdn is used to check the conformance of a DN based on the schema defined in slapd(8) and that loaded via slapd.conf(5). It opens the slapd.conf(5) configuration file or the slapd-config (5) backend, reads in the schema definitions, and then parses the DN list given on the command-line.

slapschema

extret del man slapschema

slapschema - SLAPD in-database schema checking utility

Slapschema is used to check schema compliance of the contents of a slapd(8) database. It opens the given data- base determined by the database number or suffix and checks the compliance of its contents with the corresponding schema. Errors are written to standard output or the specified file. Databases configured as subordinate of this one are also output, unless -g is specified.

Administrators may need to modify existing schema items, including adding new required attributes to object-Classes, removing existing required or allowed attributes from objectClasses, entirely removing objectClasses, or any other change that may result in making perfectly valid entries no longer compliant with the modified schema. The execution of the slapschema tool after modifying the schema can point out inconsistencies that would otherwise surface only when inconsistent entries need to be modified.

The entry records are checked in database order, not superior first order. The entry records will be checked considering all (user and operational) attributes stored in the database. Dynamically generated attributes (such as subschemaSubentry) will not be considered.

Organització de les pàgines man del servidor

Podeu consultar l'ajuda de les ordres client ldap consultant les seves pàgines man:

```
# man slap
```

slapacl	*slapd-config*	*slapdn*	*slapd-sock*	*slapo-dyngroup*	*slapo-sock*
slapadd	*slapd-dnssrv*	*slapd-ndb*	*slapd-sql*	*slapo-dynlist*	*slapo-sssvlv*
slapauth	*slapd-hdb*	*slapd-null*	*slapindex*	*slapo-memberof*	*slapo-syncprov*
slapcat	*slapd-ldap*	*slapd.overlays*	*slapo-accesslog*	*slapo-pbind*	*slapo-translucent*
slapd	*slapd-ldbm*	*slapd-passwd*	*slapo-auditlog*	*slapo-pcache*	*slapo-unique*
slapd.access	*slapd-ldif*	*slapd-perl*	*slapo-chain*	*slapo-ppolicy*	*slapo-valsort*
slapd.backends	*slapd-mdb*	*slapd.plugin*	*slapo-collect*	*slapo-refint*	*slappasswd*
slapd-bdb	*slapd-meta*	*slapd-relay*	*slapo-constraint*	*slapo-retcode*	*slapschema*
slapd.conf	*slapd-monitor*	*slapd-shell*	*slapo-dds*	*slapo-rwm*	*slaptest*

Observeu que estan organitzades segons es tracti de:

- ❏ Odres de servidor.
- ❏ El dimoni slapd.
- ❏ Descripció de access, backends, overlay i plugins (slapd.xxx).
- ❏ Descripció de varis temes (slapd-xxx).

Configuració de ACLs

Es poden definir múltiples ACLs per als elements de la base de dades fins i tot a nivell d'atributs. **Sempre** rootdn pot modificar (accedir i modificar) qualsevol contingut, per alguna cosa es root!.

El tractament de ACLs es similar als altres àmbits informàtics on es poden aplicar i com es habitual, pot acabar essent bastant complicat. En la forma més simple son fàcils d'entendre.

Reflexions sobre les dades i les ACLs:

- Un usuari pot modificar *totes* les seves dades? Puc passar de profe a Conseller d'Ensenyament amb un simple ldapmodify? Modificar la nòmina?
- Els altres usuaris quines dades poden observar? El password dels altres? L'adreça?
- Hi ha un camp que tot usuari ha de poder modificar per ell mateix, el seu propi password.

nota a vegades és fàcil confondre's i escriure les ACL al final del document fora del bloc de la base de dades on realment es pretenia configurar aquestes ACLs.

Conceptes generals

*[consultar la documentació de: man **slapd.access**, man **slapd.conf** i el **capítol 8** ACLs Administració LDAP]*

En la configuració de slapd.d es poden especificar ACLs globals per a totes les bases de dades, es la secció de definició de directives globals.

Per a cada base de dades es poden descriure ACLs específiques per a cada base de dades. **UII** a posar-les al lloc pertinent dins de cada bloc database.

Es genera una llista de regles ACL concatenant al final de les regles específiques de cada BD les regles globals (que tenen menys precedència que les específiques de la BD).

Les regles s'apliquen de manera seqüencial començant per la primera i anar descendint. Si una regla fa match al *what* ja no s'examinen més regles. Com a totes les ACL de tota la vida! Per tant l'ordre en que s'escriuen és determinant.

A més a més recordar que:

- L'usuari rootdn te drets totals per a fer-ho tot. Independentment de les ACLs.

- L'usuari *anonymous* no pot realitzar actualitzacions (encara que les ACLs ho permetessin).

- Per defecte si no s'ha indicat cap ACL es permet la lectura de tot per a tothom i només rootdn pot realitzar actualitzacions.
  access to * by * read

- Si hi ha alguna ACL definida (sigui global, local a la BD o totes dues) s'afegeix al final implícitament una regla que denega qualsevol tipus d'accés.
  access to * by * none

- Un cop s'està processant una entrada ACL que ha fet match (s'ha seleccionat el *what*) es comproven els operadors *by*, implícitament al final es denega qualsevol access.
  by * none

- els drets són acumulatius, evidentment si es té el dret de write implica el dret de read, search, etc.

Exemple-01:

access to * by * read

La ACL d'exemple indica que:

- qualsevol usuari, anònim o autenticat pot veure totes les dades de la BD.
- únicament rootdn pot realitzar actualitzacions.

Exemple-02:

access to * by * write

La ACL d'exemple indica que:

- qualsevol usuari pot modificar qualsevol de les dades de la BD, tant les seves com les dels altres.
- tot i que la ACL permet drets de write a tothom (anònims i usuaris autenticats) cal recordar que els usuaris anònims no poden realitzar actualitzacions.
- els drets són acumulatius, evidentment si es té el dret de write implica el dret de read, search, etc. És a dir, tothom pot observar totes les dades de la base de dades (també els usuaris anònims).
- evidentment rootdn pot realitzar actualitzacions.

Exemple-03:

access to * by self write by * read

La ACL d'exemple indica que:

- qualsevol usuari, anònim o autenticat pot veure totes les dades de la BD.
- un usuari pot modificar totes les seves pròpies dades.

- rootdn pot realitzar tot tipus d'actualitzacions.

Exemple-04:

```
access to attrs=homePhone by * read
access to * by * write
```

La ACL d'exemple indica que:

- tots els usuaris poden modificar qualsevol dada de la BD excepte el camp homePhone que es únicament de lectura.
- si s'està processant el camp homePhone fa *match* la primera regla i s'assigna dret de read i ja no es continua avaluant les següents regles.
- si s'està avaluant un camp/element que no és homePhone no fa *match* la primera regla i si que s'aplica la segona regla (amb l'operador *).
- rootdn pot realitzar tot tipus d'actualitzacions.

Exemple-05 ok:

```
access to attrs=homePhone
        by dn.exact="cn=Anna Pou,ou=usuaris,dc=edt,dc=org" write
        by * read
access to * by * write
```

La ACL d'exemple indica que:

- tots els usuaris poden modificar qualsevol dada de la BD excepte el camp homePhone que es únicament de lectura.
- excepte la usuària "Anna pou" que sí que té el dret de modificar els homePhone.
- si s'està processant el camp homePhone fa *match* la primera regla i llavors es valida qui és l'usuari, si és "Anna Pou" pot modificar-lo, si és un altre usuari (o anònim) només pot fer read.
- rootdn pot realitzar tot tipus d'actualitzacions.

Exemple-05 ko(malament):

```
access to attrs=homePhone
        by dn.exact="cn=Anna Pou,ou=usuaris,dc=edt,dc=org' write
access to attrs=homePhone
        by * read
access to * by * write
```

La ACL d'exemple és incorrete, perquè:

- si s'està processant el camp homePhone fa *match* amb la primera clausula what. Llavors passa a avaluar-se l'operador *by*.
- si l'usuari es "Anna Pou" s'aplica el write.
- però que passa si no és "Anna Pou"? **NO** es passa a la següent clàusula what sinó que es mira si hi ha un altre *by* (dins el mateix what) com que no hi és implícitament és com si hi ha un by * none.

- En realitat la ACL incloent els valors implícits és com si diu:

```
access to attrs=homePhone
        by dn.exact=”cn=Anna Pou,ou=usuaris,dc=edt,dc=org’ write
        by * none
access to attrs=homePhone by * read
access to * by * write
access to * by * none
```

- on podem observar en vermell les definicions implícites. la primera d’elles es la que fa que si no es tracta de la “Anna Pou” ningú pugui accedir al homePhone.
- fixeu-vos també que a la segona clàusula what no s’hi entrarà mai! Igual que a la última regla implícita (en vermell) que tampoc s’hi entrarà mai.

Exemple-06:

```
access to attrs=homePhone
        by dn.exact=”cn=Anna Pou,ou=usuaris,dc=edt,dc=org” write
        by dn.exact=”cn=Admin System,ou=usuaris,dc=edt,dc=org”  write
        by * read
access to *
   by dn.exact=”cn=Admin System,ou=usuaris,dc=edt,dc=org”  write
   by self write
   by * read
```

La ACL d’exemple indica que:

- el homePhone només pot ser actualitzat per “Anna Pou” i per “Admin System” (de tothom!). Tots els altres usuaris i l’usuari anònim només en poden fer read.
- tots els altres camps poden ser modificats per l’usuari “Admin System” i per el propi usuari (self). Els altres usuaris poden fer read de les dades però no actualitzar dades d’altres usuaris.
- rootdn pot realitzar tot tipus d’actualitzacions.

Exemple-07:

Tot usuari es pot modificar el seu propi password i tothom pot veure totes les dades de tothom, excepte els altres passwords.

```
access to attrs=userPasswordl
        by self write
         by * auth
        [ by * none ]
access to *
        by * read
```

Podeu trobar més exercicis a practica_ldap_acls.pdf

RDN i userPassword

L'atribut que s'utilitzi com a RDN per identificar els dn dels usuaris i l'atribut userPassword tenen una consideració especial (que potser han de tenir també altres atributs). Això és degut a que són utilitzats per a fer el BIND dels usuaris.

Anem a pams, quan un usuari realitza una acció identificant-se amb el seu dn i password, el client ldap primer es connecta al servidor com a usuari anònim. Llavors intenta validar el dn amb el password. Un cop validat realitza l'acció encomanada com a tal usuari.
És a dir, primerament hi ha l'autenticació de l'usuari, que com a "***anonymous***" ha de poder accedir al *dn* i al *userPassword* per poder fer l'autenticació.
Si algun d'aquests atributs és none no es podrà fer l'autenticació.
Cal almenys assignar un acces de **auth** per a l'atribut userPaaswd.
Cal almenys assignar un accés de **auth/search** als altres atributs necessaris per accedir a l'element (elements del dn, en especial el RDN).

Conclusió:

- Après que el to * ha d'anar al final o emmascara tots els atres to.
- Après que no es poden posar dos cops un mateix atribut perquè no més entrerà al primer.
- Un cop entrat a un to hi ha una llista de by, si no hi ha el * aquest serà none.
- L'atribut userPassword i el que s'utilitza com a RDN han de tenir permís de auth/search si no no es podran fer els bind d'autenticació.

Synopsis ACLs

Ldap Documentation ACLs

If no access controls are present, the default policy allows anyone and everyone to read anything but restricts updates to rootdn. (e.g.,"**access to * by * read**").

When dealing with an access list, because the global access list is effectively appended to each per-database list, if the resulting list is non-empty then the access list will end with an implicit **access to * by * none** directive. If there are no access directives applicable to a backend, then a default read is used.

Be warned: the **rootdn** can always read and write EVERYTHING!

access to <what> [by <who> [<access>] [<control>]]+

Grant access (specified by <access>) to a set of entries and/or attributes (specified by <what>) by one or more requestors (specified by <who>).

The field <what> specifies the entity the access control directive applies to. It can have the forms:

dn[.<dnstyle>]=<dnpattern>
filter=<ldapfilter>

attrs=<attrlist>[val[/matchingRule][.<attrstyle>]=<attrval>]

Te field <who> indicates whom the access rules apply to. Multiple <who> statements can appear in an access control statement, indicating the different access privileges to the same resource that apply to different accessee. It can have the forms:

*
anonymous
users
self[.<selfstyle>]
dn[.<dnstyle>[,<modifier>]]=<DN>

THE <ACCESS> FIELD

The optional field <access> ::= [[real]self]{<level>|<priv>} determines the access level or the specific access privileges the who field will have. Its component are defined as

<level> ::= none|disclose|auth|compare|search|read|{write|add|delete}|manage
<priv> ::= {=|+|-}{0|d|x|c|s|r|{w|a|z}|m}+

The modifier **self** allows special operations like having a certain access level or privilege only in case the operation involves the name of the user that's requesting the access.

The level access model relies on an incremental interpretation of the access privileges. The possible levels are **none**, **disclose**, **auth**, **compare**, **search**, **read**, **write**, and **manage**. **Each access level implies all the preceding ones**, thus manage grants all access including administrative access. The write access is actually the combination of add and delete, which respectively restrict the write privilege to add or delete the specified <what>.

The **none** access level disallows all access including disclosure on error.
The **disclose** access level allows disclosure of information on error.
The **auth** access level means that one is allowed access to an attribute to perform authentication/authorization operations (e.g. bind) with no other access. This is useful to grant unauthenticated clients the least possible access level to critical resources, like passwords.

Operation requeriments

Operations require different privileges on different portions of entries. The following summary applies to primary database backends such as the BDB and HDB backends. Requirements for other backends may (and often do) differ.

The **add** operation requires add **(=a)** privileges on the pseudo-attribute entry of the entry being added, and add (=a) privileges on the pseudo-attribute children of the entry's parent. When adding the suffix entry of a database, add access to children of the empty DN ("") is required. Also if Add content ACL checking has been configured on the database (see the slapd.conf(5) or slapd-config(5) manual page), add (=a) will be required on all of the attributes being added.

The **bind** operation, when credentials are stored in the directory, requires auth **(=x)** privileges on the attribute the credentials are stored in (usually userPassword).

The **compare** operation requires compare **(=c)** privileges on the attribute that is being compared.

The **delete** operation requires delete **(=z)** privileges on the pseudo-attribute entry of the entry being deleted, and delete **(=d)** privileges on the children pseudo-attribute of the entry's parent.

The **modify** operation requires write **(=w)** privileges on the attributes being modified. In detail, add (=a) is required to add new values,delete (=z) is required to delete existing values, and both delete and add **(=az)**, or write (=w), are required to replace existing values.

The **modrdn** operation requires write (=w) privileges on the pseudo-attribute entry of the entry whose relative DN is being modified, delete (=z) privileges on the pseudo-attribute children of the old entry's parents, add (=a) privileges on the pseudo-attribute children of the new entry's parents, and add (=a) privileges on the attributes that are present in the new relative DN. Delete (=z) privileges are also required on the attributes that are present in the old relative DN if deleteoldrdn is set to 1.

The **search** operation, requires search **(=s)** privileges on the entry pseudo-attribute of the searchBase (NOTE: this was introduced with OpenLDAP 2.4). Then, for each entry, it requires search (=s) privileges on the attributes that are defined in the filter. The resulting entries are finally tested for read (=r) privileges on the pseudo-attribute entry (for read access to the entry itself) and for read (=r) access on each value of each attribute that is requested. Also, for each referral object used in generating continuation references, the operation requires read (=r) access on the pseudo-attribute entry (for read access to the referral object itself), as well as read (=r) access to the attribute holding the referral information (generally the ref attribute).

Exemples de consulta de permisos

L'ordre slapacl permet consultar els permisos assignats a una entrada de la base de dades, una entity i també els permisos assignats als atributs. També permet consultar si un usuari concret té o no determinats drets.

Observar que tota entitat consta de dos elements anomenats entity i children que determinen els permisos de la 'caixa' de l'entitat i els permisos dels 'fills' de l'entitat. Es pot pensar en un sistema equivalent al dels permisos de fitxers i de directoris.

```
# slapacl -b 'ou=usuaris,dc=edt,dc=org'
entry: read(=rscxd)
children: read(=rscxd)
ou=usuaris: read(=rscxd)
description=Container per usuaris del sistema linux: read(=rscxd)
```

```
objectClass=organizationalUnit: read(=rscxd)
…

# slapacl -b 'cn=Anna Pou,ou=usuaris,dc=edt,dc=org'
entry: read(=rscxd)
children: read(=rscxd)
objectClass=posixAccount: read(=rscxd)
objectClass=inetOrgPerson: read(=rscxd)
cn=Anna Pou: read(=rscxd)
cn=Anita Pou: read(=rscxd)
sn=Pou: read(=rscxd)
mail=anna@edt.org: read(=rscxd)
ou=Alumnes: read(=rscxd)
uid=anna: read(=rscxd)
uidNumber=5002: read(=rscxd)
gidNumber=600: read(=rscxd)
homeDirectory=/tmp/home/anna: read(=rscxd)
userPassword=****: read(=rscxd)
...

# slapacl -b 'cn=Anna Pou,ou=usuaris,dc=edt,dc=org' 'mail'
mail: read(=rscxd)

# slapacl -b 'cn=Anna Pou,ou=usuaris,dc=edt,dc=org' 'mail/read'
read access to mail: ALLOWED
# slapacl -b 'cn=Anna Pou,ou=usuaris,dc=edt,dc=org' 'mail/write'
write access to mail: DENIED
# slapacl -b 'cn=Anna Pou,ou=usuaris,dc=edt,dc=org' 'mail/add'
add access to mail: DENIED

# slapacl -D 'cn=Marta Mas,ou=usuaris,dc=edt,dc=org' -b 'cn=Marta
Mas,ou=usuaris,dc=edt,dc=org' 'userPassword'
authcDN: "cn=marta mas,ou=usuaris,dc=edt,dc=org"
userPassword: write(=wrscxd)

 slapacl -D 'cn=Marta Mas,ou=usuaris,dc=edt,dc=org' -b 'cn=Marta
Mas,ou=usuaris,dc=edt,dc=org' 'userPassword/read'
authcDN: "cn=marta mas,ou=usuaris,dc=edt,dc=org"
read access to userPassword: ALLOWED

# slapacl -b 'cn=Marta Mas,ou=usuaris,dc=edt,dc=org' 'userPassword/read'
read access to userPassword: DENIED
```

Configuració dinàmica del servidor

Actualitzacions de la configuració del servidor 'en calent'

El servei LDAP i la configuració del servidor, del propi dimoni funcionen com una base de dades LDAP. Totes les opcions de configuració del servidor es poden consultar i actualitzar amb ordres ldap.

Recapitulem, el servidor s'ha configurat amb les directives del fitxer de configuració slapd.conf amb el que s'ha generat una estructura de directoris slapd.d. Aquesta estructura de directoris té forma d'entrades LDIF.

A més a més en engegar el servidor i crear la base de dades de l'organització "edt.org" també s'han generat automàticament altres bases de dades, corresponents al propi servidor, al frontend, al les BD que es creen i al monitor (si s'ha activat).

La base de dades config es crea sempre (tant si s'indica en el fitxer de configuració com si no) i representa tota la configuració del servidor slapd. Es poden fer consultes i actualitzacions ldap sobre aquesta base de dades i això generarà canvis 'en calent' en la configuració del servidor slapd.

****atenció:****
Per poder accedir a la base de dades "cn=config" corresponent a la configuració del dimoni del servei slapd cal definir un rootdn i un rootpw per a l'entrada database config. Aquesta entrada és única. L'exemple mostra què cal incloure de nou en la configuració de slapd.conf. Caldrà eliminar i **regenerar** de nou la configuració, **no** les dades.

part afegida al slapd.conf

```
database config
rootdn "cn=Sysadmin,cn=config"
rootpw {SSHA}vwpQxtzc7yLsGg8K7fm02p2Fhox/PFP4
# el passwd es syskey
```

```
# systemctl  stop slapd
# rm -rf /etc/openldap/slapd.d/*
# slaptest -f slapd-cn=configl-edt.org.conf -F /etc/openldap/slapd.d/
config file testing succeeded
# chmod -R ldap.ldap /etc/openldap/slapd.d/
# systemctl  start slapd
```

Ara autenticats amb l'usuari "cn=Sysadmin,cn=config" (atenció a no posar el dc=edt,dc=org!) es poden realitzar consultes ldap a la configuració del servidor. Podem llistar totes les entrades LDAP igual que si es realitzés l'ordre "*slapcat -n0*". També llistar les dades per a una BD concreta (per exemple la de "dc=edt,dc=org". I fins i tot mostrar els valors concrets d'una directiva de configuració (un atribut en la BD ldap).

```
# slapcat -n0
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b 'cn=config'
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b 'cn=config' dn

# configuració de la bd{1} edt.org
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b
'olcDatabase={1}bdb,cn=config'

# configuració de les ACLs
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b
'olcDatabase={1}bdb,cn=config'  olcAccess

# configuració dels índex
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b
'olcDatabase={1}bdb,cn=config'  olcDbIndex
```

També es poden fer actualitzacions a la base de dades "cn=config" usant les mateixes eines de ldap (ldapmodify, ldapadd,etc). Cal autenticar-se com a rootdn per poder realitzar les actualitzacions. A continuació es pot veure un exemple de fitxer LDIF usat per a la modificació de la configuració del servidor:

```
dn: olcDatabase={1}bdb,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: to * by * read
```

El fitxer entry.ldif anterior elimina totes les ACLs de la base de dades {1} que correspon a "dc=edt,dc=org" i a continuació estableix una única ACL amb drets de lectura per a tothom.

```
# slapcat -n0 | grep olcAccess

# ldapmodify -x -h localhost -D 'cn=Sysadmin,cn=config' -w syskey -f entry.ldif

# slapcat -n0 | grep olcAccess
# ldapsearch -x -h localhost -LLL -D 'cn=Sysadmin,cn=config' -w syskey -b
'olcDatabase={1}bdb,cn=config'  olcAccess
```

Instal·lació d'un servei (repàs general)

Passos a realitzar:

- instal·lar el servei.
- observar el contingut dels paquets instal·lats.
- identificar els directoris de configuració, dades, etc.
- identificar el nom del servei i saber fer: start, stop, status, enable, disable.
- identificar el fitxer de configuració.
- identificar el fitxer del PID del servei i observar-lo en l'arbre de processos. Llistar els serveis actius (list-units) i observar que en forma part.
- Identificar el directori de logs del servei i observar el tipus de logs que genera.
- Identificar el fitxer de lock del subsystem que impedeix a altres serveis LDAP engegar-se o usar recursos apropiats.

Aprofitant que estem parlant d'un repàs general és el moment per tornar a fer un repàs de la presentació de LDAP, en aquest moment hauriem d'entendre totes les explicacions exceptuant el concepte d'schema:

M06-UF1NF1-Introduccio_a_LDAP

Configuració LDAP Client

Observar el contingut del fitxer de configuració client */etc/openldap/ldap.conf*.

Exemple del fitxer client /etc/openldap/ldap.conf per defecte que s'instal·la.

```
# cat /etc/openldap/ldap.conf
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE	dc=example,dc=com
#URI	ldap://ldap.example.com ldap://ldap-master.example.com:666

#SIZELIMIT	12
#TIMELIMIT	15
#DEREF		never

TLS_CACERTDIR	/etc/openldap/certs

# Turning this off breaks GSSAPI used with krb5 when rdns = false
SASL_NOCANON	on
```

Exemple de fitxer client /etc/openldap/ldap.conf d'un host de les aules d'informàtica:

```
# cat /etc/openldap/ldap.conf
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE	dc=example,dc=com
#URI	ldap://ldap.example.com ldap://ldap-master.example.com:666

#SIZELIMIT	12
#TIMELIMIT	15
#DEREF		never

TLS_CACERTDIR /etc/openldap/certs

# Turning this off breaks GSSAPI used with krb5 when rdns = false
SASL_NOCANON	on
```

URI ldap://ldap/
BASE dc=escoladeltreball,dc=org

Eines gràfiques: gq i phpldapadmin

Eina gràfica: gq

Utilitzar l'eina gràfica gq per consultar la base de dades ldap. Examinar les entitats i els seus atributs. En especial identificar els atributs requerits i opcionals.

El gq **falla molt sovint**, però és realment pràctic per **examinar els objectclass i els seus atributs**, ajuda molt en la creació de nous schema (consultant la part de schema).

```
# dnf -y install gq
$ gq &
```

Un cop instal·lat el procediment és:

- ❏ configurar un "***server***". Una connexió indicant un nom de connexió i contra quin host i port cal connectar. Opcionalment es pot especificar un usuari (bindDN) i contrasenya de connexió, però en el nostre cas no cal perquè realitzarem un accés *anonymous*.
- ❏ Un cop configurat es pot accedir via "***browse***" al la connexió configurada. En fer-ho sol·licitarà usuari i passwd però es pot prémer enter per establir la connexió anònima.
  Amb aquesta opció es pot navegar per les entitats del DIT, clicqueu en el triangle desplegable de cada entitat. Malauradament la versió actual *'peta'* si intenteu accedir al contingut concret d'una entitat.
- ❏ Utilitzem gq per poder navegar per l'estructura "***schema***". En aquesta navegació es poden observar els objectClass, els attribyteTypes i altres elements que defineixen els tipus de dades que hi ha carregats al servidor.

Editar el server amb el que contactar:

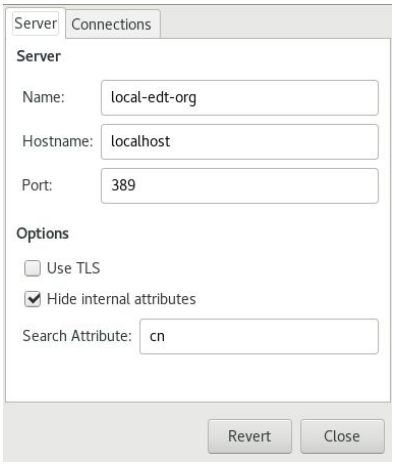
Menú File:
- Preferences
  - Servers
    - edit:
      - name: *edt.org* (o el nom identificatiu de la connexió que es desitgi)
      - hostname: 172.17.0.2 (si és un container docker) indicar el nom de host o adreça IP del servidor ldap.
      - *<**nota**>: es penja en accedir als objectes, però és útil per examinar els schema*

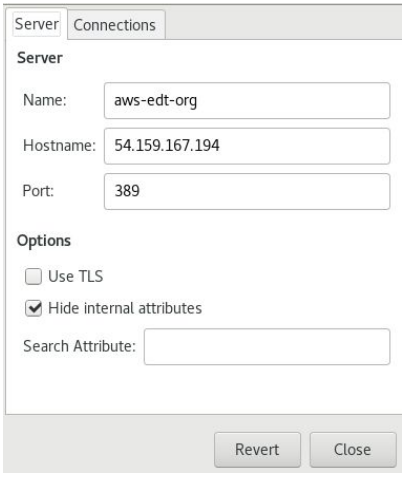
- Browse:
  - seleccionar la connexió edt.org
  - passwd: usar <enter> si fem un bind anònim.
  - podem obserbar tot l'arbre DIT de dc=edt,dc=org

- Schema
  - podem observar tots els schema carregats
  - també els atributs, objectes, matching rules i syntaxes.

Exemple de configuració de l'accés a un servidor (primera imatge):

- es defineix el nom de connexió "local-edt-ldap".
- s'indica que connectarà al localhost port 389 (on hi ha propagat el port ldap del container).
- no s'utilitza una connexió de tràfic segur TLS.
- en la pestanya Connections no s'ha indicat res. La connexió es farà com a usuari anònim.





Podem definir tantes connexions a servidor com ens calguin. Podem accedir a altres servidors ldap remots indicant un nou server i configurant-ne l'accés. En la segona imatge del quadre anterior es pot observar:

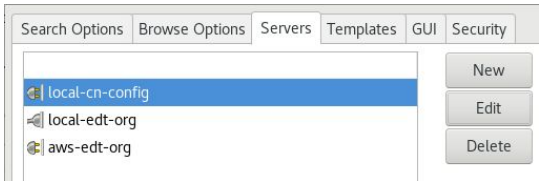
- es defineix una connexió anomenada aws-edt-org
- aquesta connexió s'estableix contra un servidor ldap que s'està executant en una AMI de AWS EC2 (Amazon Cloud Computing) a l'adreça IP indicada.
- en la pestanya Connections no s'ha indicat res. La connexió es farà com a usuari anònim.

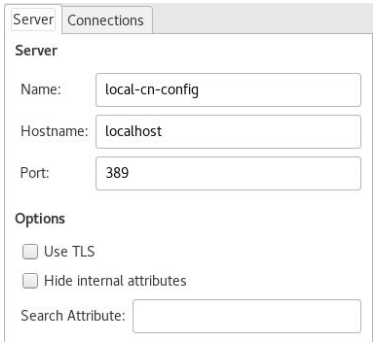
De fet es poden crear tantes connexions com es desitgi. En el nostre cas podem fer connexions a altres hosts de l'aula, a altres hosts AWS dels companys de classe, al container directament usant la seva adreça IP de la xarxa interna local (si estem al host amfitrió atacant directament per exemple 172.18.0.02), etc.

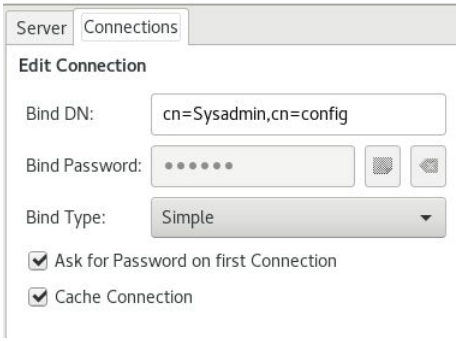
Però no ens hem de limitar a consultar la base de dades dc=edt,dc=org, podem consultar també les bases de dades:

- dc=edt,dc=org
- cn=config (si no hi ha una ACL amb read per tothom caldrà autenticar com a cn=Sysadmin,cn=config i el password pertinent).
- podem consultar els valors de la base de dades monitor.

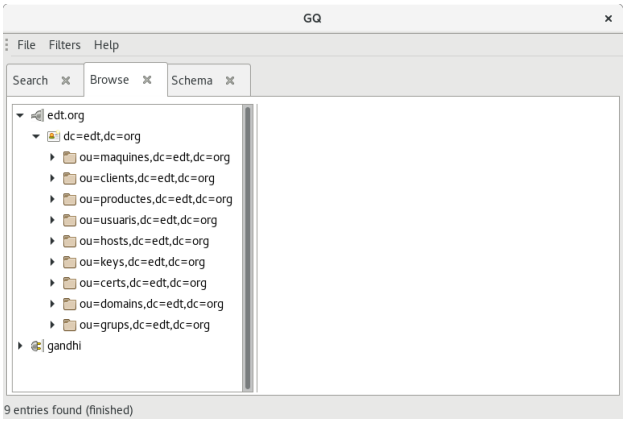
Observem per exemple la configuració per accedir a cn=config:







Exemple de llistat de dades de dc=edt,Dc=org:



Eina gràfics: phpldapadmin

Per utilitzar des d'un navegador web l'eina gràfica phpldapadmin cal instal·lar i configurar phpldapadmin i també apache (servei httpd).

Podeu consultar documentació de com configurar phpldapadmin en el “chaper 8 LDAP and the Web” apartat “phpLDAPadmin” del llibre “Mastering OpenLDAP. Configuring, Securing and Integrating Directory Services” de l’editorial Pack Publishing.

En principi phpldapadmin està configurat per ser accedit **localment**, incorpora un servidor http i és a través del port 80 local via **localhost/phpldapadmin** que podrem accedir a l’administració web del ldap.

Si volem permetre l’accés remot a la seu web de phpldapadmin caldrà configurar el servidor web Apache per tal de permetre connexions remotes (per defecte està configurat per permetre només l’accés local al port 80)..

****nota**** Si volem per poder engegar el container amb el servidor ldapserver però atacar al port 389 del host i no de la ip local del container cal mapejar al host el port ldap:

$ docker run **-p 389:389** -d edtasixm06/ldap20:schema

******nota***** Genereu el container amb Fedora-27 o baralleu-vos amb la nova versió 7.4 de PHP!*

Procediment de treball:

- instal·lar el software
- configurar el phpldapadmin perquè accedeixi al repositori de dades ldap (al nostre container ldapserver).
- configurar apache (el servidor web) perquè la web de phpldapadmin sigui accesible externament i no únicament localment.
- accedir a la pàgina web d’administració ldap.

Instal·lar el software:

```
# dnf install phpldapadmin php-xml httpd
```

Configurar el servei phpldapadmin:

```
# vim /etc/phpldapadmin/config.php

/*********************************************
 * Define your LDAP servers in this section  *
 *********************************************/
...
$servers->newServer('ldap_pla');
$servers->setValue('server','name','Local edt.org');
$servers->setValue('server','host','172.17.0.2');
$servers->setValue('server','port',389);
$servers->setValue('server','base',array('dc=edt,dc=org'));
$servers->setValue('login','auth_type','session');
$servers->setValue('server','tls',false)

/*********************************
```

```
 *      SASL Authentication         *
 ************************************/
// $servers->setValue('appearance','password_hash','ssha');
    // comentada perquè generava un error
//$servers->setValue('login','attr','uid');
    // comentada perquè generava un error
```

Configurar Apache per permetre l'accés remot

```
# vim /etc/httpd/config.d/phpldapadmin.conf

Alias /phpldapadmin /usr/share/phpldapadmin/htdocs
Alias /ldapadmin /usr/share/phpldapadmin/htdocs
<Directory /usr/share/phpldapadmin/htdocs>
  <IfModule mod_authz_core.c>
        Require all granted
  </IfModule>
  <IfModule !mod_authz_core.c>
        Allow from *
  </IfModule>
</Directory>
```

Configuració extra actual:

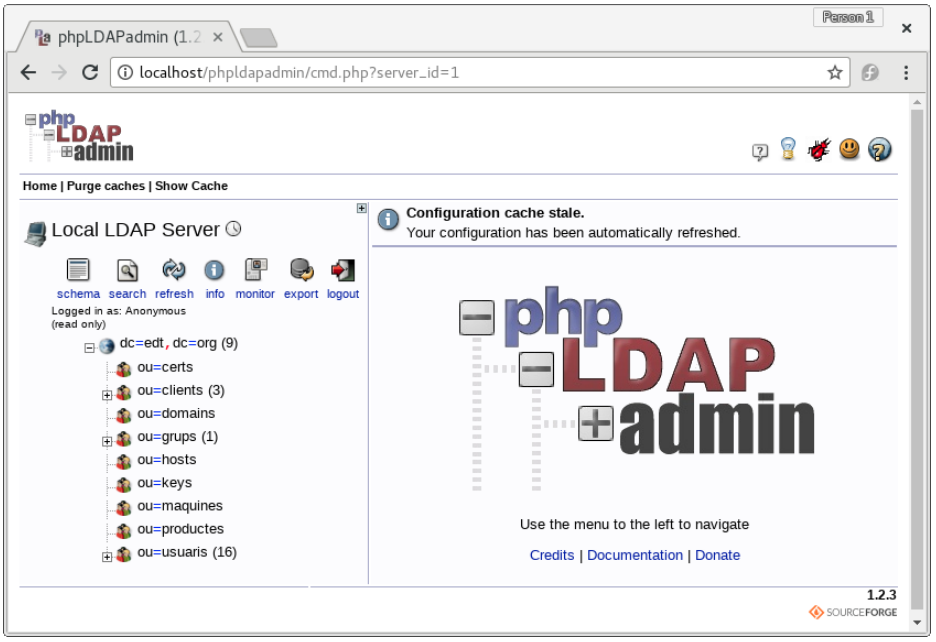
```
<només per a php 7.4, no cal per a f27 > # mkdir /run/php-fpm
```

Iniciar el servei

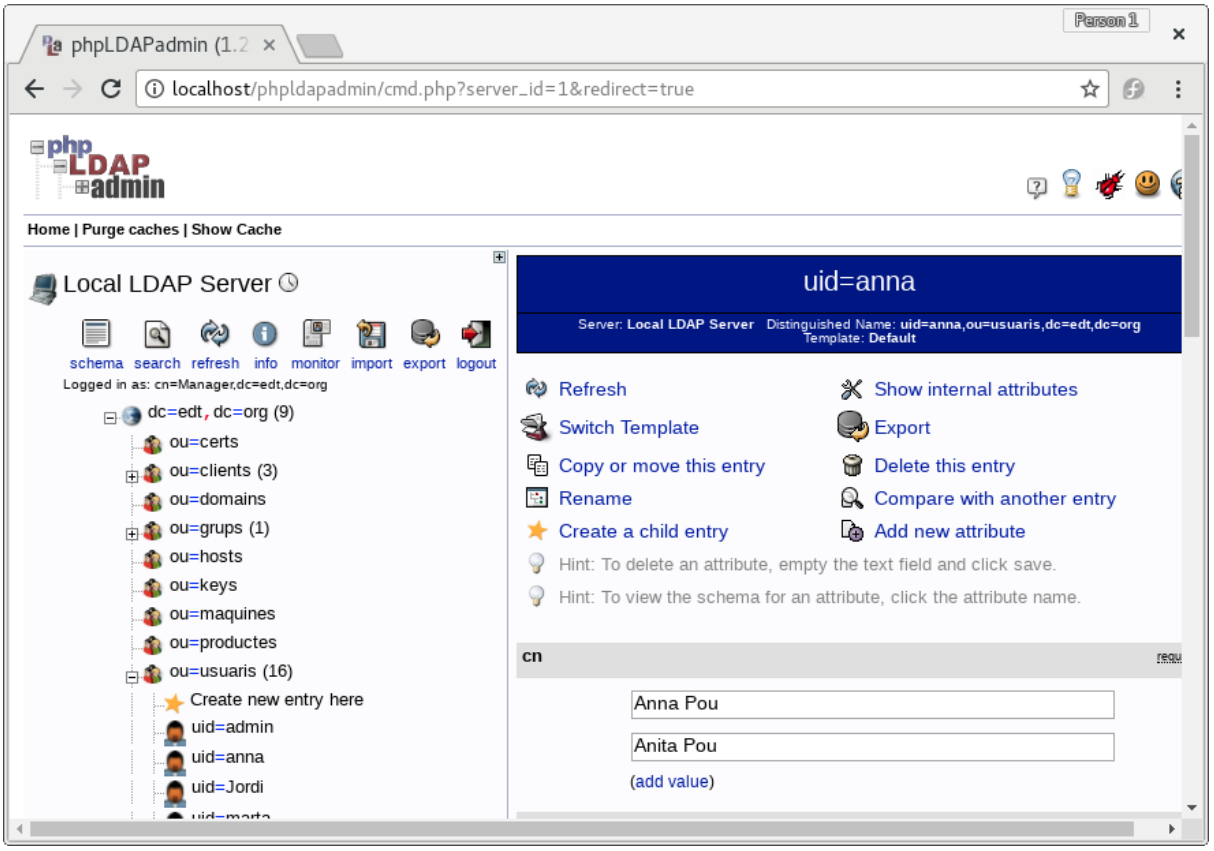
```
# /sbin/php-fpm
# /sbin/httpd
# https -S
```

Exemple de connexió com a usuari anònim.
Per accedir a la web d'administració de phpldapadmin cal contactar al port 80 del host on s'està executant el servei web Apache que conté aquesta web.
Si ho fem localment caldrà indicar la URL: ***localhost/phpldapadmin***
Si ho fem remotament caldrà indicar la URL: http://172.17.0.3/phpldapadmin suposant que aquesta és l'adreça IP on s'executa el servei.

Exemple d'accés local:



Com a ***anonymous*** simplement podem consultar informació (si les ACLs ho permeten). Podem identificar-nos com un usuari en l'apartat de login. Si fem el Bind com a **rootDN** es pot administrar tota la base de dades.



Observeu l'estrella amb l'opció "Create New user" i les opcions Copy, Rename, etc que apareixen.

Generar accés a dues bases de dades

Igual que s'ha configurat l'accés per a la base de dades edt.org podem generar també un altre accés per a la base de dades de configuració cn=config.

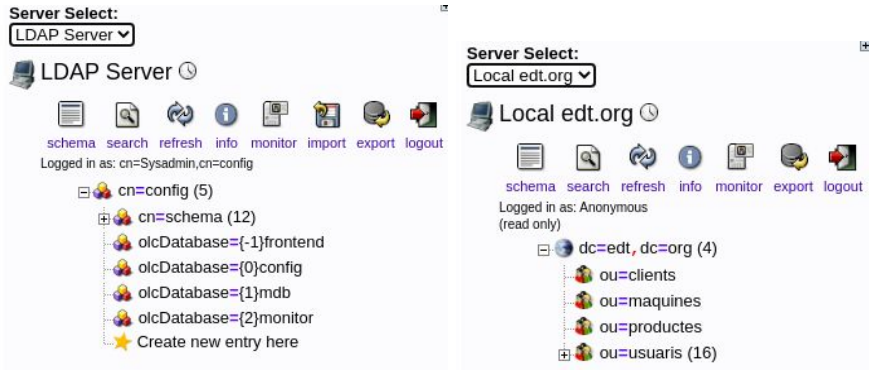
- ❏ dc=edt,dc=org (amb accés anònim)
- ❏ cn=config (amb accés autenticat)

Generar a la part final del fitxer de configuració una nova entrada de v¡base de dades ldap. Atenció, assegureu-vos que el codi no està comentat (moveu avall el /* de comentari!).

```
/*********************************************************************
 * If you want to configure additional LDAP servers, do so below.        *
 * Remove the commented lines and use this section as a template for all  *
 * your other LDAP servers.                                               *
 *********************************************************************/

$servers->newServer('ldap_pla');
$servers->setValue('server','name','LDAP Server');
$servers->setValue('server','host','ldap.edt.org');
$servers->setValue('server','port',389);
$servers->setValue('server','base',array('cn=config'));
$servers->setValue('login','auth_type','cookie');
//$servers->setValue('login','bind_id','');
//$servers->setValue('login','bind_pass','');
$servers->setValue('server','tls',false);
/*
```

Ara podem observar que tenim un desplegable per seleccionar la base de dades amb la que connectar. Recordeu que per accedir a cn=config segurament cal autenticació.



Schema: creació objectes/atributs

En aquest apartat es mostraran exemples de creació d'atributs, objectes i esquemes propis. Consultar la documentació de la guia d'administració ldap disponible localment (file:///usr/share/doc/openldap-servers/guide.html) o a la seu web de openldap (http://www.openldap.org/doc/admin24/). Consultar el capítol (13) "Schema specificaction".

També es recomana consultar la documentació de la web www.ldap.com, en especial l'apartat Understandig LDAP schema.

Una altra manera de practicar i aprendre el funcionament dels atributs, objectes i schema es navegar per aquests elements a la pestanya Schema del visor gràfic gq.

Es treballaran els següents exemples:

- Implementar els exemples del a documentació de openldap, creant atributs i objectes propis. Carregar l'Schema a la base de dades i carregar dades.
- Implementar un schema propi amb dades de futbolistes. Creant l'objecte futbolista i els seus atributs. Aquest objecte serà derivat de inetOrgperson. Carregar l'Schema a la base de dades i carregar dades.
- Modificar l'exemple anterior fent de l'objecte un objecte structural que no deriva de cap altre objecte, és a dir, derivat de TOP. Carregar l'Schema a la base de dades i carregar dades.
- Fer un tercer exemple amb futbolistes on l'objecte futbolista és de tipus auxiliar. Això implica que cada entitat futbolista de la base de dades ha d'anar associada a un objecte structural (s'utilitzarà inetorgPerson). Carregar l'Schema a la base de dades i carregar dades.

Conceptes generals de Schema

En aquest apartat s'intentarà descriure el concepte de Schema comparant-lo amb la definició d'una base de dades relacional com per exemple la base de dades training usada a classe al mòdul de bases de dades.

Bases de dades relacionals

Una base de dades relacional es composa de múltiples taules, cada taula conté registres d'informació que contenen per a cada element un conjunt de dades, els camps. En bases de

dades relacions cada taula ha de ser homogènia, la informació de cada registre que la composa conté els mateixos camps. Dins d'una taula un registre no pot sre que contingui uns camps i un altre registre uns altres camps.

Així per exemple, la taula oficinas conté a cada registre la descripció d'una oficina. Quan es defineix la taula oficinas cal indicar quins seran els camps que la componen i per cada camp cal indicar el nom del camp i el tipus de valor que contindrà (i altres resctriccions).

Exemple de definició de la base de dades oficinas:

```
CREATE TABLE "oficinas" (
        "oficina" int2 NOT NULL,
        "ciudad" character varying(15) NOT NULL,
        "region" character varying(10) NOT NULL,
        "dir" int2,
        "objetivo" numeric(9,2),
        "ventas" numeric(9,2) NOT NULL,
        primary key (oficina)
);
```

En l'exemple anterior podem observar que:

- es defineix el format que tindrà una taula anomenada oficinas, on cada oficina contindrà la informació dels camps que es detallen.
- El camp 'oficina' és de tipus enter. Concretament veiem que es defineix que el primer camp s'anomena 'oficina', és de tipus 'int2' i no pot ser un null. És a dir, s'indica el nom, el tipus de dades a contenir i les restriccions.
- Observem per exemple que el camp 'region' és de tipus text de 10 caràcters.
- En canvi el camp 'ventas' és un camp numèric amb decimals.

En resum, es crea una estructura de dades que anomenem ***taula*** i es defineix quina informació contindrà, cada bocí d'informació és un ***camp*** del qual cal indicar el ***nom***, ***tipus*** de dada i altres ***restriccions***.

Base de dades LDAP

La base de dades LDAP s'estructura en forme d'arbre ***DIT Directory Information Tree***. És un arbre on cada element és una ***Entity***. Cada entitat està formada per un o més ***objectClass***. Aquests es composen d'***atributs*** que tenen un nom identificatiu, un tipus de dada i poden tenir restriccions.

Fent la analogia amb les bases de dades relacionals podem dir que en el DIT no tenim taules perquè cada element emmagatzemat, cada *entity*, pot ser del tipus que sigui (podem desar oficinas, pedidos, productos, etc tot barrejat en un sol DIT).

Cada *entitat* està formada per almenys un *objectClass*, però poden ser-ne varis. Observem per exemple que els usuaris (cn=Pere Pou,ou=usuaris,dc=edt,dc=org) estan formats per dos objectClass: posixAccount i inetOrgPerson. En canvi per exemple l'entitat productes

(ou=productes) està formada només per l'objetClass organizationalUnit. També l'entitat dc=edt,dc=org està formada per dos objectClass anomenats dcObject i organization.

Podríem dir que cada entitat, potser millor pensat dir cada objectClass, és com una taula d'una base de dades relacional: ***table ~ entity***

Cada objectClass es composa d'***atributs*** que emmagatzemen informació (igual que els camps en les BDR). Per exemple un objectClass d'un posixAccount (un compte de sistema Linux/Unix) ha de contenir el uid, uidNumber, gidNumber, etc. Cada un d'aquests elements és un atribut.

Podem dir amb tota certesa que un atribut és l'equivalent a un camp d'una base de dades relacional: ***field ~ attribute***

objectClass

Observem per exemple la definició de l'objecte ***posixAccount***:

```
olcObjectClasses: {0}( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
  DESC 'Abstraction  of an account with POSIX attributes'
  SUP top
  AUXILIARY
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description
   ) )
```

De la mateixa manera que en la definició d'una taula d'una base de dades relacional cal dir quins són els camps que la componen (indicant nom, tipus i restriccions per a cada camp), en definir un objectClass cal indicar quins són els atributs que el componen i quins són obligatoris i quins opcionals.

Analitzem la definició de l'objectClass posixAccount

- Primerament s'indica el OID o identificador únic.
- S'indica el nom de l'objecte amb NAME: posixAccount
- S'indica una descripció amb DESC.
- Aquest objecte no és derivat de cap altre objecte, és un objecte fil de TOP. És a dir, un objecte arrel que no deriva de cap altre.
- Els objectes poden ser *Structural* o *Auxiliary*, en aquest cas és Auxiliary.
- MUST conté la llista d'atributs (separats per $) dels atributs que són obligatoris.
- MAY conté la llista d'atributs (separats per $) dels atributs que són obligatoris.
- easy peacy!

La definició indica que un posixAccount és un objecte auxiliar derivat de TOP que conté els atributs obligatoris cn, uid, uidNumber, gidNumber i homeDirectory. Els atributs userPassword, loginShell, gecos i description són opcionals.

De quin tipus són aquests camps? pot un uidNumber prendre el valor ‘patata’? Pot un homeDirectori valdre ‘345’?.

attributeTypes

En la definició d’una taula de base de dades relacional per a cada camp s’indica el seu nom, el tipus de dada que pot contenir i altres requeriments (per exemple si pot ser null, repetit, etc). On es defineix això en les bases de dades LDAP?

La definició de l’objecte posixAccount indica de quins atributs es composa i els anomena, però no estan definits dins del objectClass sinó que cada atribut es defineix a part per ell mateix.

Exemples de definició d’atributs:

```
olcAttributeTypes: ( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
    DESC 'RFC2307: An inte ger uniquely identifying a user in an administrative domain'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )
```

```
olcAttributeTypes: {0}( 1.3.6.1.1.1.1.2 NAME 'gecos'
    DESC 'The GECOS field; the common name'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
    SINGLE-VALUE )

olcAttributeTypes: {1}( 1.3.6.1.1.1.1.3 NAME 'homeDirectory'
    DESC 'The absolute path to the home directory'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
    SINGLE-VALUE )

olcAttributeTypes: {2}( 1.3.6.1.1.1.1.4 NAME 'loginShell'
    DESC 'The path to the login shell' EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
    SINGLE-VALUE )
```

Observer el primer exemple correspondent a la definició de l’atribut ***uidNumber***:

- S’indica el ***OID*** d’aquest atribut, un valor de Object ID únic per a tot objective.
- Amb *NAME* s’indica el nom de l’atribut.
- Amb *DESC* es fa la descripció de què és i per a què serveix l’atribut.
- Un uidNumber és d’un valor numèric i es defineix amb EQUALITY que per comparar dos uidNumbers cal usar la **regla *integerMatch***. És la regla que defineix com comparar dos integers.
- *ORDERING* indica quina regla cal usar per ordenar uidNumbers, sabem que són números integer i que per tant es vol aplicar una **regla** que ordeni enters, ***integerOrdeningMatch***.
- Amb *SYNTAX* es defineix quin tipus de dada és aquest atribut. Encara que no ho sembla a simple vista aquí se li està dient que aquest atribut és un enter. Sembla

mentida però en lloc de dir 'integer' el que es fa és posar el OID que indica que el tipus de dada que conté és un integer, en aquest cas el OID ***1.3.6.1.4.1.1466.115.121.1.27***.

- Igual que passa amb els camps de les bases de dades relacionals podem posar restriccions als atributs, per exemple si són únics o si es poden repetir. Amb *SINGLE-VALUE* s'indica que aquest és un camp que no admet duplicats (de fet que és un camp clau).

Amb ***SYNTAX*** es fa l'equivalent a quan en un camp d'una base de dades relacional escrivim de quin tipus és, si varchar, decimal, integer, etc. L'inconvenient és que en lloc d'usar aquestes paraules que ja tenim clares com a informàtics s'utilitzen codis OID que identifiquen als tipus de dades.

Així el més usual és que per a atributs numèrics indiquem les propietats de: EQUALITY i ORDERING. Per a atributs de cadena és molt usual indicar les propietats de: EQUALITY i SUBSTR. I sempre caldrà indicar amb SYNTAX de quin tipus és

Així doncs per definir un *objectClass* cal definir els *attributeTypes* que el composen o bé usar attributeTypes que ja estiguin definits en altres *schema*. Per definir un attributeTypes cal indicar la *SYNTAX* (el tipus de dades) i les *regles* de comportament. Aquestes regles i tipus es poden definir també o usar de les que ja hi ha predefinides en altres schema.

Quan es defineix un atribut es pot fer de nou, indicant-ho tot o es pot fer 'derivant' l'atribut d'un de ja existent. Observem l'exemple de l'atribut *ou organizationalUnit*:

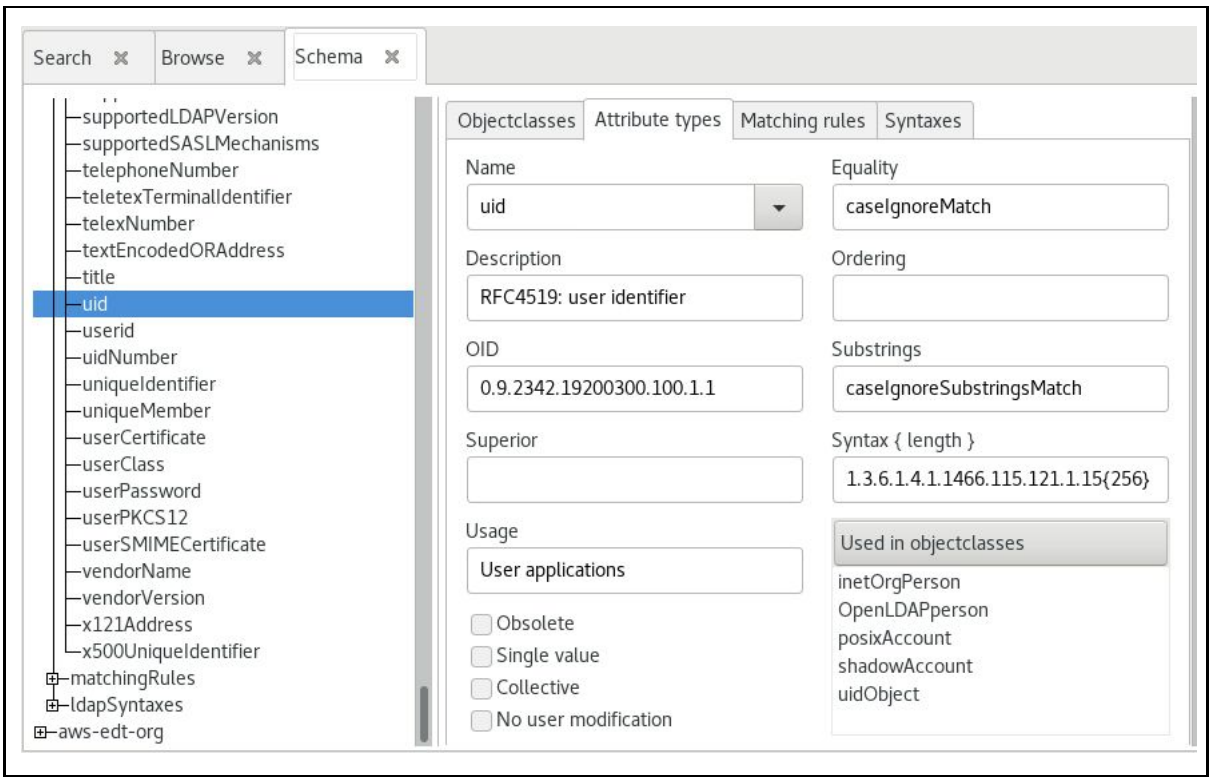
```
olcAttributeTypes: {8}( 2.5.4.11 NAME ( 'ou' 'organizationalUnitName' )
   DESC  'RFC2256: organizational unit this object belongs to'
   SUP name )
```

Podem veure que:

- Name defineix dos noms tots dos vàlids, ou i organizationalUnit. Es poden usar indistintament.
- No s'indica la SINTAX ni el EQUALITY, etc sinó que s'indica que ***SUP*** és name. És a dir, que és un atribut derivat de *name* i que n'hereta totes les propietats.

nota podeu consultar objectClass, atributs, etc per exemple a la web: ***ldapwiki.com***.

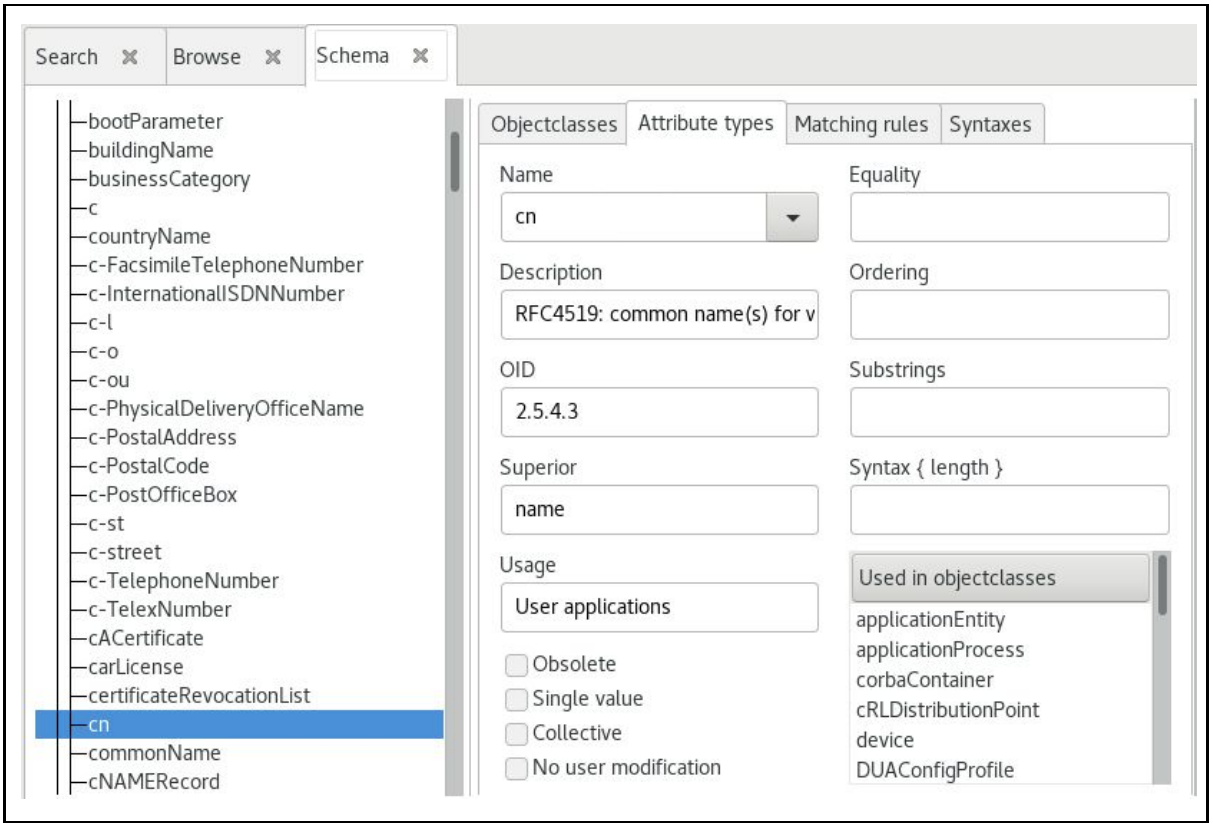
Observeu per exemple la definició del attributeTypes *uid* (el login):



Podem veure que:

- El nom uid està en un desplegable perquè tant pot ser uid com userid.
- Correspon a la definició del stàndard RFC4519* que descriu què és un user identifier.
- OID que identifica de manera única en l'espai de noms LDAP aquest objecte.
- La sintàxi correspon a un string (acaba en 15) i està limitada a {256} caràcters.
- Les regles que s'apliquen són de comparar caràcters ignorant la diferència de majúscules/minúscules.
- Es llista els objectClass que l'utilitzen.

Observeu per exemple la definició del attributeTypes ***cn common name***:



Podem veure que:

- Aquest atribut és un derivat de l'atribut *name* del qual n'hereta totes les propietats.

ldapSyntaxes

De quins tipus pot ser un atribut? En bases de dades relacionals coneixem els tipus de dades clàssics boolean, integer, varchar, decimal, etc. Quins són els tipus de dades LDAP?

La resposta és aquests i tots els que es vulguin crear. Evidentment hi ha un llistat bàsic de tipus de dades ja predefinits en els schema que inclouen booleans, enters, cadenes, cadenes IA5, imatges, audio, binaris, etc. De fet n0hi ha tants i amb tants dígits de OID que són un mal de cap!

Un llistat d'exemple dels més típics és (extret de ldapwiki)

- Bit String
- Boolean
- Country String
- Delivery Method
- Directory String
- DIT Content Rule Description
- DIT Structure Rule Description
- DN
- Enhanced Guide
- Facsimile Telephone Number
- Fax
- Generalized Time
- Guide LDAPSyntax
- IA5 String
- Integer
- JPEG
- Numeric String

- Octet String
- Postal Address
- Printable String
- Telephone Number
- Teletex Terminal Identifier
- Telex Number
- UTC Time

Si ho preferim podem veure el llistat d'alguns dels OID usuals (extret de OpenLdap Documentation):

Name	OID	Description
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean value
directoryString	1.3.6.1.4.1.1466.115.121.1.15	Unicode (UTF-8) string
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	LDAP DN
integer	1.3.6.1.4.1.1466.115.121.1.27	integer
numericString	1.3.6.1.4.1.1466.115.121.1.36	numeric string
OID	1.3.6.1.4.1.1466.115.121.1.38	object identifier
octetString	1.3.6.1.4.1.1466.115.121.1.40	arbitrary octets

Així per exemple per dir:

- audio → 1.3.6.1.4.1.1466.115.121.1.4
- binari → 1.3.6.1.4.1.1466.115.121.1.5
- boolean → 1.3.6.1.4.1.1466.115.121.1.7
- country string → 1.3.6.1.4.1.1466.115.121.1.11
- distinguished name → 1.3.6.1.4.1.1466.115.121.1.12
- directory string → 1.3.6.1.4.1.1466.115.121.1.15
- IA5 string → 1.3.6.1.4.1.1466.115.121.1.26
- integer → 1.3.6.1.4.1.1466.115.121.1.27
- JPEG → 1.3.6.1.4.1.1466.115.121.1.28
- octet string → 1.3.6.1.4.1.1466.115.121.1.40
- telephone number → 1.3.6.1.4.1.1466.115.121.1.50

****nota**** podeu també consultar els deplegables de schema de l'aplicació gq.

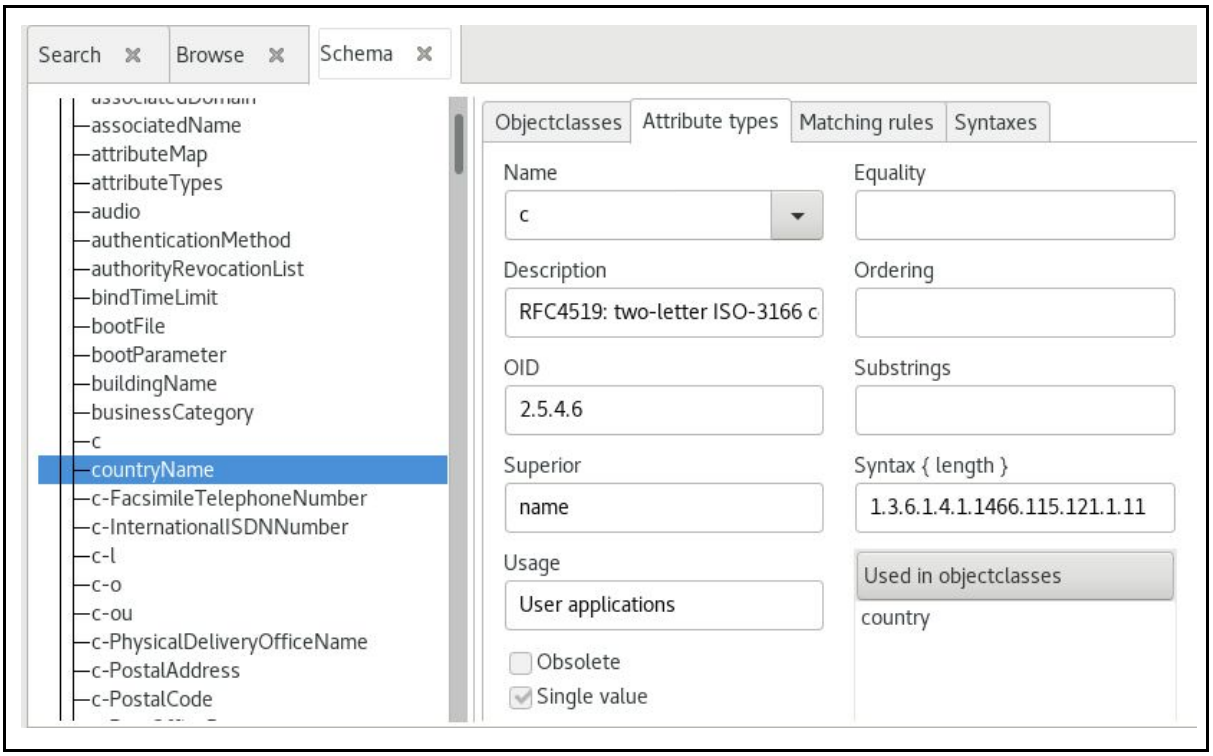
Particularització de tipus de dades

Fixeu-vos en el cas per exemple de country string (1.3.6.1.4.1.1466.115.121.1.11). És un string per identificar un país, el tipic *ct* per catalunya o *pl* per palestina, per exemple. No es tracta doncs de un string qualsevol sinó de una ***particularització*** del tipus String que defineix que han de ser codis de dos lletres per identificar països.

Un altre exemple és el telephone number (1.3.6.1.4.1.1466.115.121.1.50) que no és un valor numèric sinó una particularització del tipus String indicant el format que han de tenir els números de telèfon.

Un exemple a considerar són els strings IA5, els strings que contenen únicament caràcters del codi alfabètic internacional IA5 (consulteu a la *wikipèdia en IA5*).

Observeu per exemple l definició del attributeTypes *country name*:



Definició de tipus de dades propis

De la mateixa manera que es poden crear objectClass i attributeTypes es poden crear ldapSyntaxes propis. És a dir, tipus de dades definits per l'usuari. Per exemple es podria definit un tipus de dada anomenat nomModul que seria un string que conté el nom d'una assignatura; o per exemple numModul que seria un valor tipus "m06" que descriu el identificador d'una assignatura. De fet es pot definir el tipus de dada que es vulgui!

matchingRules

Imaginem el següent cas: creem un objectClass persona amb els camps nom, edat, pes, alçada, adreça, títols acadèmics, etc. Podem comparar dues persones? com les comparem, qui tingui més gran… l'edat? el pes? o segons la moda d'ara qui tingui menys pes?.

A programació s'estudien els criteris de comparació i el concepte de comparadors. Per comparar dues cadenes o dos enters no hi ha cap problema, tothom sap fer-ho!. El sistema sap com fer-ho. Però si en python creem dos objectes persona, com els comparem?. Si volem comparar-los per cognom caldrà definir un comparador a tal efecte. De fet sabem que per ordenar generalment ens cal usar (almenys per desempatar) un camp clau. Per exemple es poden comprar per nom (nom i cognoms) i desempatar per dni. És a dir, caldria fer un comparador seguint aquest criteri.

Els tipus de dades clàssics que existeixen a tots els llenguatges tenen definits mecanismes molt clars de comparació (EQUALITY) d'ordenació (ORDERING) i de tractament de substrings (SUBSTR).

Però si en l'apartat anterior hem vist que es poden inventar tipus de dades nous caldrà també definir en aquests tipus de dades com aplicar aquestes regles anomenades ***matchingRules***.

En general els tipus de dades que s'utilitzen acaben essent derivats numeros o strings i les regles que s'apliquen són regles típiques per a aquests tipus de dades.

Així per exemple podem consultar algunes de les matchinRules a OpenLdap Documentation:

Name	Type	Description
booleanMatch	equality	boolean
caseIgnoreMatch	equality	case insensitive, space insensitive
caseIgnoreOrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreSubstringsMatch	substrings	case insensitive, space insensitive
caseExactMatch	equality	case sensitive, space insensitive
caseExactOrderingMatch	ordering	case sensitive, space insensitive
caseExactSubstringsMatch	substrings	case sensitive, space insensitive
distinguishedNameMatch	equality	distinguished name
integerMatch	equality	integer
integerOrderingMatch	ordering	integer
numericStringMatch	equality	numerical
numericStringOrderingMatch	ordering	numerical
numericStringSubstringsMatch	substrings	numerical
octetStringMatch	equality	octet string
octetStringOrderingMatch	ordering	octet string
octetStringSubstringsMatch ordering	octet st	ring
objectIdentiferMatch	equality	object identifier

Podem observar que hi ha les típiques regles de:

- booleanMatch
- caseIgnoreMatch
- caseExactMatch
- integerMatch
- integerOrderingMatch
- numericStringMatch
- octetStringMatch

Amb aquestes regles predefinides en general ja en tindrem prou ***en aquest curs NO generarem regles pròpies***. però hem de saber que existeix la possibilitat de definir-ne de noves i incorporar-ne de noves procedents d'altres schema.

Pensem per exemple en un attributeType de tipus JPEG que conté una imatge. Potser volem poder establir comparadors del tipus quina és la resolució, si la resolució és la mateixa és que fan match, si no no. O ordenar segons la resolució. I fins i tot comparar imatges si realment tenen la mateixa imatge de fons amb reconeixement d'imatges?... ufff!!!

Conclusions finals

Les dades que fins ara s'han estat utilitzant de la base de dades dc=edt,dc=org contenient objectes del tipus organization, organizationUnit, posixAccount, inetOrgPerson. Cada un d'aquest objectes consta d'atributs que són alguns obligatòris i d'altres opcionas. Cada atribut es defineix amb un nom, un tipus de dada (syntax) i requeriments a complir (single value per exemple). De tipus de dades n'hi ha molts de predefinits però s'en poden crear de nous. Cada tipus de dada pot especificar quines són les regles que defineixen el seu comportament en l'àmbit de la comparació, l'ordenació i els substrings.

Així en els fitxers de schema hi trobarem les definicions de:

- objectClass
- attributeTypes
- ldapSyntaxes
- matchingRules

Definició d'un objectClass

En crear nous objectClass cal tenir present aquests conceptes teòrics:

- Tipus: estructural / auxiliar / abstract (no el treballem)
- Superior: TOP / un altre objecte

Tipus: Structural / Auxiliary

> Un objectClass ***structural*** és un objecte que té sentit per ell mateix, que una entitat pot estar formada simplement per aquest objecte. Recordem que tota entitat ha d'estar formada per almenys un objectClass structural.
>
> Una possible definició és que és un objecte de que és vàlid per si mateix.
>
> Com que això costa una mica d'entendre si no es tenen coneixements de POO Programació Orientada a Objectes, anem a esperar a fer la definició dels objectes auxiliary per veure si s'acalreix això!

Un objectClass ***auxiliary*** és un objecte que no pot existir com a entitat per si mateix i que per tant ha d'anar associat a altres objectes. És Com un complement, un afegit que poden tenir (o no) altres objectes.

Les entitats estan formades d'almenys un objecte estructural i poden tenir cap o varis objectes auxiliary.

Pensem per exemple en el joc de la wii. L'aparell i els comandament són elements estructurals del joc, si no els tens no tens la wii. En canvi el volant per conduir és un element auxiliary, podem tenir-lo, però no és imprescindible. L'element de per si no serveix de res si no va associat als elements estructurals de l'aparell i comandaments.

Penseu en vosaltres mateixos, com a objectClass sou persones Person (o qualsevol dels seus derivats… sense ofendre!). Però a part podem ser socis d'un club de futbol, d'un centre esportiu, de la biblioteca, algun cavernicola d'un videoclub, etc. Podem pensar que cada un d'aquestes altres activitats corresponen al altres objectClass que són auxiliary que podem afegir en l'entitat d'usuaris que té a Person com a objectClass structural.

Superior: TOP / altre-objectClass

Un objectClass diem que deriva de ***TOP*** si no rep herència de ningú, si no deriva de cap altre objectClass.

Un objectClass que sigui un derivat d'un altre objecte diem que aquest altre objecte és el seu ***superior***. L'element derivat rep en herència tots els atributs (attributeTypes, ldapSyntaxes i matchingRules).

Per exemple si es crea un objectClass *hisx2InetOrgPerson* derivat de *inetOrgPerson* heretarà tots els atributs de inetOrgPerson. Això inclou els atributs MUST cn i sn definits a *Person*.

Investigar els schema carregats

Quins són els schema carregats a la configuració: ***slapd.conf***

```
include     /etc/openldap/schema/corba.schema
include     /etc/openldap/schema/core.schema
include     /etc/openldap/schema/cosine.schema
include     /etc/openldap/schema/duaconf.schema
include     /etc/openldap/schema/dyngroup.schema
include     /etc/openldap/schema/inetorgperson.schema
include     /etc/openldap/schema/java.schema
include     /etc/openldap/schema/misc.schema
include     /etc/openldap/schema/nis.schema
include     /etc/openldap/schema/openldap.schema
include     /etc/openldap/schema/ppolicy.schema
include     /etc/openldap/schema/collective.schema
```

La ruta on es troben els fitxers de schema és:

- ❑ */etc/openldap/schema*

Observar que un cop generada la confiuració amb ***slaptest*** aquests schema estan en el directori ***/etc/openldap/slapd.d***:

```
# tree /etc/openldap/slapd.d/
/etc/openldap/slapd.d/
├── cn=config
│   ├── cn=schema
│   │   ├── cn={0}corba.ldif
│   │   ├── cn={10}ppolicy.ldif
│   │   ├── cn={11}collective.ldif
│   │   ├── cn={1}core.ldif
│   │   ├── cn={2}cosine.ldif
│   │   ├── cn={3}duaconf.ldif
│   │   ├── cn={4}dyngroup.ldif
│   │   ├── cn={5}inetorgperson.ldif
│   │   ├── cn={6}java.ldif
│   │   ├── cn={7}misc.ldif
│   │   ├── cn={8}nis.ldif
│   │   └── cn={9}openldap.ldif
│   ├── cn=schema.ldif
│   ├── olcDatabase={-1}frontend.ldif
│   ├── olcDatabase={0}config.ldif
│   ├── olcDatabase={1}mdb.ldif
│   └── olcDatabase={2}monitor.ldif
└── cn=config.ldif
```

Llistat de schema predefinits (extret de OpenLdap Documentation):

File	Description
core.schema	OpenLDAP *core* (required)
cosine.schema	Cosine and Internet X.500 (useful)
inetorgperson.schema	InetOrgPerson (useful)
misc.schema	Assorted (experimental)
nis.schema	Network Information Services (FYI)
openldap.schema	OpenLDAP Project (experimental)

Un cop carregada la configuració dinàmica observar en el bolcat de la base de dades zero (cn=config) les configuracions carregades:

```
# slapcat -n0 | grep -A 3 posixAccount
olcObjectClasses: {0}( 1.3.6.1.1.1.2.0 NAME 'posixAccount' DESC 'Abstraction
  of an account with POSIX attributes' SUP top AUXILIARY MUST ( cn $ uid $ u
 idNumber $ gidNumber $ homeDirectory ) MAY ( userPassword $ loginShell $ ge
 cos $ description ) )
```

nota observeu que s'ha usat l'opció -A del grep per mostrar les 3 següents línies d'on ha trobat la ocirrència

Buscar quin schema té el objectClass ***posixAccount***

```
# cd /etc/openldap/slapd.d/cn=config/cn=schema

# grep posixAccount *
cn={8}nis.ldif:olcObjectClasses: {0}( 1.3.6.1.1.1.2.0 NAME 'posixAccount' DESC 'Abstraction
```

Observem que el fitxer d'schema ***nis.ldif*** és el que conté la definició de què és un objectClass *posixAccount*.

Busquem ara on podem trobar la definició de què és un atribut ***uidNumber***:

```
# slapcat -n0 | grep -A 3 uidNumber | more
olcAttributeTypes: ( 1.3.6.1.1.1.1.0 NAME 'uidNumber' DESC 'RFC2307: An inte
 ger uniquely identifying a user in an administrative domain' EQUALITY integ
 erMatch ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )
```

Observem els atributs ***gecos***, ***homeDirectory*** i ***loginShell***:

```
# head -n14 /etc/openldap/slapd.d/cn\=config/cn\=schema/cn\=\{8\}nis.ldif
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 34738d11
dn: cn={8}nis
objectClass: olcSchemaConfig
cn: {8}nis
olcAttributeTypes: {0}( 1.3.6.1.1.1.1.2 NAME 'gecos' DESC 'The GECOS field;
 the common name' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substrings
 Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
olcAttributeTypes: {1}( 1.3.6.1.1.1.1.3 NAME 'homeDirectory' DESC 'The absol
 ute path to the home directory' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4
 .1.1466.115.121.1.26 SINGLE-VALUE )
olcAttributeTypes: {2}( 1.3.6.1.1.1.1.4 NAME 'loginShell' DESC 'The path to
 the login shell' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121
 .1.26 SINGLE-VALUE )
```

Observar per exemple la definició del attributeTypes ***sn***:

```
olcAttributeTypes: {1}( 2.5.4.4 NAME ( 'sn' 'surname' )
    DESC 'RFC2256: last (family) name(s) for which the entity is known by'
    SUP name )
```

Observar per exemple la definició del attributeTypes ***ou***:

```
olcAttributeTypes: {8}( 2.5.4.11 NAME ( 'ou' 'organizationalUnitName' )
    DESC  'RFC2256: organizational unit this object belongs to'
    SUP name )
```

Carregueu en un editor de text (vim) un dels fitxers de schema, per exemple include /etc/openldap/schema/nis.schema i examineu-ne el contingut.

Exemples de creació d'un Schema

En aquest apartat veurem com crear un schema propi. Primerament es mostra l'exemple que podeu trobar a la documentació web de *OpenLdap Documentation* (chapter 13). Tot seguit es crea un exemple propi per inventar un esquema de *futbolistes*.

Exemple OpenLdap Documetation

Llistat d'exemple de l'schema que implementa els exemples de la documentació openldap:

```
attributetype ( 1.1.2.1.1 NAME 'x-my-UniqueName'
    DESC 'unique name with my organization'
            SUP name )
```

```
#
attributetype ( 1.1.2.1.2 NAME 'x-my-UniqueOrg'
    DESC 'unique name of my organization'
    EQUALITY caseIgnoreMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE )
#
attributetype ( 1.1.2.1.3 NAME 'x-my-Photo'
    DESC 'a photo (application defined format)'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
    SINGLE-VALUE )
#
attributetype ( 1.1.2.1.4 NAME 'x-my-PhotoURI'
    DESC 'URI and optional label referring to a photo'
    SUP labeledURI )

# Exemple objecte
objectclass ( 1.1.2.2.1 NAME 'x-my-data'
    DESC 'nou tipus objecte my-data'
    AUXILIARY
    MUST description
    MAY  ( x-my-UniqueOrg $ x-my-Photo $ x-my-PhotoURI $ countryName $
                localityName ) )

# Exemple objecte
objectclass ( 1.1.2.2.2 NAME 'x-my-Person'
    DESC 'nou tipus objecte my-person'
    SUP inetOrgPerson
    STRUCTURAL
    MUST description
    MAY ( x-my-UniqueName $ countryName $ localityName $ streetAddress $
              stateOrProvinceName  ) )
```

Llistat d'un fitxer LDIF amb dades injectables a la base de dades dc=edt,dc=org d'entitats que utilitzen l'schema definit anteriorment:

```
dn: o=depinf,ou=maquines,dc=edt,dc=org
objectclass: organization
objectclass: x-my-data
o: depinf
description: organitzacio depinf amb un objectclass compost de organization i x-my-data
 camp obligatoris o i description
countryName: uk
localityName: barcelona

dn: cn=Persona01,ou=maquines,dc=edt,dc=org
objectclass: x-my-person
cn: Persona01
sn: paio01
description: objecte x-my-data derivat de inetorgpersona  els camps must son cn i sn
 derivats de Person com a camp must del derivat hi ha la desciption
countryName: uk
localityName: barcelona
```

Finalment observar com s'ha modificat el fitxer de configuració slapd.conf per incloure aquest schema, verificar-lo i regenerar la configuració

```
# cat slapd.conf
…
include                 /var/tmp/m06/schema/mysch01.schema

# slaptest -f slapd.conf
# slatest -f slapd.conf -F /etc/openldap/slapd.d
```

```
# slapcat -n0 | grep dn:
```

Carregar i comprovar les dades de la BD:

```
# ldapadd -xvh localhost:386 -D cn=Manager,dc=edt,dc=org' -w secret -f dades-ldif

# slapcat | grep dn:
```

Cas pràctic: futbolista.schema

En aquest apartat s'implementarà un cas pràctic creant un schema propi anomenat futbolista.schema que permet emmagatzemar informació de futbolistes. Un futbolista entemem que es descriurà per el nom de l'equip, el dorsal, la seva pàgina web, una foto i l'estat de lesionat (si/no).

Descripció dels *atributs* del ObjectClass *Futbolista*

- *equip* atribut de text que conté el nom de l'equip al que pertany. Ha de ser un nom únic (potser en Figo opina diferent…)
- *dorsal* atribut numèric amb el valor del dorsal que utilitza. Ha de ser un valor únic.
- *webpage* atribut amb una URL de la seva pàgina web personal de merchandising. Pot ser multi-valued perquè pot tenir més d'una pàgina web.
- *photo* attribut amb una foto JPEG del futbolista. Pot ser multi-value.
- *lesionat* atribut que indica l'estat de lesionat o no del futbolista, un boolean.

Descripció del *objectClass Futbolista*

- MUST equip. Considerem que l'atribut equip és obligatori
- MAY tots els altres atributs són opcionals.

Passes a fer:

- Crear un fitxer futbolista.schema amb la definició dels atributs i del objectClass.
- Incorporar al fitxer de configuració *slapd.d* que carregui aquest esquema amb la directiva *include*.
- Generar de nou la configuració del servidor amb *slaptest*. Si no hi ha errors és que el fitxer afegit de futbolista.schema és correcte.
- Crear un fitxer *ldif* amb dades a inserir de futbolistes complint els requisits de must i may i del tipus de dades de cada atribut.

Casos a implementar:

A) *Estructural* derivat de *inetOrgPerson*
   Crear un objectClass Futbolista que sigui estructural i derivat de inetOrgPerson. Això implica que a més a més dels atributs que tindrà com a futbolista hereterà els de inetOrgPerson (on cn i sn són obligatoris).

B) *Estructural* derivat de *TOP*

Crear un objectClass Futbolista que sigui estructural i derivat de TOP, és a dir, de ningú. Ara ja no herata cap atribut de inetOrgPerson de manera que les entitats no poden tenir els atributs cn, sn etc que tenien en l'exemple anterior.

Apareix el problema del dn. Com s'identifica un futbolista si no hi ha cn? Això exigeix modificar l'schema i afegir un nou atribut amb el nom del futbolista, que es únic i actúa de RDN.

C) *Auxiliary* derivat de TOP

Es crearà un objectClass auxiliary. Es pot decidir si es fa derivat de TOP o de inetOrgPerson o d'un altre objectClass, però en aquest exemple es farà derivat de TOP.

En ser auxiliary caldrà que una entitat (per exemple usuari) tingui a un altre objecte structural (per exemple inietOrgPerson). Així per exemple els futbolistes en realitat seran entitats inetOrgPerson i x-Futbolista (els dos objectClass).

Cas-A: Derivat de inetOrgPerson

Implementar les dades de un futbolista (equip, dorsal, webpage, foto, lesinat) creant un nou objecte x-Futbolista estructural derivat de inetOrgPerson. Això implica que a més a més dels atributs que tindrà com a futbolista hereterà els de inetOrgPerson (on cn i sn són obligatoris).

Exemple del fitxer futbolista-A.schema

```
attributetype ( 1.1.2.1.1 NAME 'x-equip'
   EQUALITY caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   SINGLE-VALUE )

attributetype ( 1.1.2.1.2 NAME 'x-dorsal'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   SINGLE-VALUE )

attributetype ( 1.1.2.1.4 NAME 'x-webpage'
   DESC 'MOD uirlabel for webpage'
   EQUALITY caseExactMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributetype ( 1.1.2.1.5 NAME 'x-photo'
   DESC 'fotosssss'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.40)

attributetype ( 1.1.2.1.6 NAME 'x-lesionat'
            DESC 'lesionat     TRUE'
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
            SINGLE-VALUE)

objectclass ( 1.1.2.2.1 NAME 'x-Futbolista'
   DESC 'Futboleros'
   SUP inetOrgPerson
   STRUCTURAL
   MUST x-equip
```

```
MAY (x-dorsal $ x-webpage $ x-photo $ x-lesionat) )
```

Exemple de fitxer LDIF amb dades de futbolistes. Utilitzeu un parell d'imatges JPEG petites, per exemple de icones i les inseriu també de contingut com a photo del futbolista:

```
dn: cn=Vladimir remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolista
cn: vladimir remar
cn: VLaD ThE IMPaLer
sn: vladi
x-equip: los pimientos
x-dorsal: 7
x-webpage: www.vladimir.remar
x-photo:< file://opt/docker/myphoto1.jpg
x-lesionat: FALSE
homephone: 555-212-2220
mail: vladimir.remar@gmail.com

dn: cn=renzo remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolista
cn: renzo remar
cn: RARS0
sn: renzo
x-equip: los pimientos
x-dorsal: 10
x-webpage: www.renzo.remar
x-photo:< file://opt/docker/myphoto2.jpg
x-lesionat: TRUE
homephone: 555-212-2221
mail: renzo.remar@gmail.com
```

Cas-B: Structural standalone object

Modificació de l'exemple anterior per generar un objecte x-Futbolista que sigui estructural pero no derivi de cap altre objecte. Bé, de fet si deriva de TOP que és una classe abstracta de la qual deriven tots.

Ara ja no hereta els atributs i propietats de inetOrgPerson com en el cas anterior. Això obliga a afegir un camp *nom de futbolista* per poder-lo identificar apropiadament i usar-lo en el dn.

Observar que els camps cn, sn, homePhone i mail usats en l'exemple anterior ja no són necessaris. De fet no es poden posar perquè no pertanyen al objectClass x-Futbolista.

Exemple de fitxer schema futbolista-B.schema:

```
attributetype ( 1.1.2.1.7 NAME ( 'x-nom' 'lonom' )
            EQUALITY caseIgnoreMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE )

attributetype ( 1.1.2.1.1 NAME 'x-equip'
   EQUALITY caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   SINGLE-VALUE )
```

```
attributetype ( 1.1.2.1.2 NAME 'x-dorsal'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )

attributetype ( 1.1.2.1.4 NAME 'x-webpage'
    DESC 'MOD uirlabel for webpage'
    EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributetype ( 1.1.2.1.5 NAME 'x-photo'
    DESC 'fotosssss'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.40)

attributetype ( 1.1.2.1.6 NAME 'x-lesionat'
    DESC 'lesionat     TRUE'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
    SINGLE-VALUE)

objectclass ( 1.1.2.2.1 NAME 'x-Futbolistes'
    DESC 'Futboleros'
    SUP TOP
    STRUCTURAL
    MUST ( lonom $ x-equip )
    MAY (x-dorsal $ x-webpage $ x-photo $ x-lesionat) )
```

Llistat de dades LDIF:

```
dn: lonom=Vladimir remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolista
lonom: vladimir remar
#cn: VLaD ThE IMPaLer
#sn: vladi
x-equip: los pimientos
x-dorsal: 7
x-webpage: www.vladimir.remar
x-photo<:file:///var/tmp/myphoto.jpg
x-lesionat: FALSE
#homephone: 555-212-2220
#mail: vladimir.remar@gmail.com

dn: lonom=renzo remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolista
lonom: renzo remar
#cn: RARS0
#sn: renzo
x-equip: los pimientos
x-dorsal: 10
x-webpage: www.renzo.remar
x-photo:< file://opt/docker/myphoto2.jpg
x-lesionat: TRUE
#homephone: 555-212-2221
#mail: renzo.remar@gmail.com
```

Cas-C: Objecte Auxiliary

Modificar els exemples anteriors per generar un objecte x-Futbolistes que sigui Auxiliary i que es pugui usar en entitats que combinin les dades de l'objecte structural *inetOrgPerson* i x-Futbolista.

Ara ja torna a no caldre el nom (x-nom) o almenys no cal que sigui obligatori ja que aquesta informació és als camps cn i sn. Observar les dades de un element inserit llistat amb slapcat.

Exemple de fitxer futbolista-C.schema

```
attributetype ( 1.1.2.1.7 NAME ( 'x-nom' 'lonom')
            EQUALITY caseIgnoreMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE )

attributetype ( 1.1.2.1.1 NAME 'x-equip'
   EQUALITY caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   SINGLE-VALUE )

attributetype ( 1.1.2.1.2 NAME 'x-dorsal'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   SINGLE-VALUE )

attributetype ( 1.1.2.1.4 NAME 'x-webpage'
   DESC 'MOD uirlabel for webpage'
   EQUALITY caseExactMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributetype ( 1.1.2.1.5 NAME 'x-photo'
   DESC 'fotosssss'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.40)

attributetype ( 1.1.2.1.6 NAME 'x-lesionat'
   DESC 'lesionat     TRUE'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
   SINGLE-VALUE)

objectclass ( 1.1.2.2.1 NAME 'x-Futbolistes'
   DESC 'Futboleros'
   SUP TOP
   AUXILIARY
   MUST x-equip
   MAY (x-nom $ x-dorsal $ x-webpage $ x-photo $ x-lesionat) )
```

Llistat de les dades LDIF:

```
dn: cn=Vladimir remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolistes
objectclass: inetOrgPerson
cn: vladimir remar
cn: VLaD ThE IMPaLer
sn: vladi
x-equip: los pimientos
x-dorsal: 7
x-webpage: www.vladimir.remar
x-photo:< file:///opt/docker/myphoto.jpg
x-lesionat: FALSE
homephone: 555-212-2220
mail: vladimir.remar@gmail.com

dn: cn=renzo remar,ou=Productes,dc=edt,dc=org
objectclass: x-Futbolistes
objectclass: inetOrgPerson
cn: renzo remar
cn: RARS0
sn: renzo
x-equip: los pimientos
x-dorsal: 10
```

x-webpage: www.renzo.remar
x-photo:< file:///opt/docker/myphoto2.jpg
x-lesionat: TRUE
homephone: 555-212-2221
mail: renzo.remar@gmail.com

Permanència i Backup de les dades

Un cop tenim un servidor LDAP en funcionament és important garantir que les dades que emmagatzema perduren i que la configuració actual també està correctament desada per si cal tornar a instal·lar de nou el servidor.

Des del punt de vista de les dades cal tenir en ment que n'hi ha de dos tipus diferents:

- **Les dades de configuració**: Són les dades de ***/etc/openldap/slapd.d***. Si bé és cert que el fitxer de configuració és el slapd.conf sabem que la configuració s'ho pogut modificar amb actualitzacions dinàmiques via cn=config (tal i com fem en els exemples de acls). De manera que la configuració real del servidor és la que està al directori de configuració o la que reflecteix el bolcat de ***slapcat -n0***.

- **Les dades de la BD**: Són les dades que emmagatzema el servidor LDAP, el DIT. es troben usualment al directori ***/var/lib/ldap***.Es poden bolcar amb **slapcat**.

Estratègies:
- generar tar/gz dels directoris de dades i de configuració (per separat)
- generar bolcats ldif de la configuració i de les dades (per separat)
- usar volums docker i fer-ne backups.

Estratègies de backup / restore de la configuració

ldif

El mecanisme més senzill és fer un bolcat de la configuració (base de dades -n 0) en format ldif. I quan es vol restaurar carregar a la base de dades de configuració (-n 0) aquest fitxer ldif.

[root@ldap docker]# slapcat -n0 -l config.edt.org.ldif
rm -rf /etc/openldap/slapd.d/* # rm -rf /var/lib/ldap/*
[root@ldap docker]# slapadd -n0 -F /etc/openldap/slapd.d -l config.edt.org.ldif

Observem els passos realitzats:
- fer un bolcat amb *slapcat -n0 -l <file>* generant config.edt.org.ldif que conté tota la configuració dinàmica actual.

- esborrem la configuració tenint el servei aturat. Podem decidir si esborrar també les dades o no. Recordeu que si es modifica la configuració però es mantenen les dades la nova configuració ha de ser compatible amb les dades emmagatzemades.
- inserim de nou la configuració amb *slapadd -n0 -F /etc/openldap/slapd.d -l <file>*.

tar.gz

Generar un tar.gz del directori de configuració /etc/openldap/slapd.d. I quan calgui restaurar la configuració extreure el tar generant de nou el directori.

```
[root@ldap docker]# tar cvzf config.edt.org.tgz /etc/openldap/slapd.d/
```

```
rm -rf /etc/openldap/slapd.d/*
```

```
[root@ldap docker]# tar  xvzf config.edt.org.tgz  -C /
```

copia del directori

S volem fer proves de configuracions diferents preservant la configuració actual també es pot fer una copia del directori simplement amb un altre nom. Quan calgui restaurar-la es torna a posar el nom /etc/openldap/slapd.

```
[root@ldap docker]# mv /etc/openldap/slapd.d/ /etc/openldap/slapd.d-$(date +"%Y%m%d")
[root@ldap docker]# mkdir /etc/openldap/slapd.d
[root@ldap docker]# chown -R ldap.ldap /etc/openldap/slapd.d
```

```
[root@ldap docker]# ll /etc/openldap/
total 24
drwxr-xr-x 1 root root 4096 Sep 15 12:33 certs
-rw-r--r-- 1 root root  121 Sep 15 12:33 check_password.conf
-rw-r--r-- 1 root root  855 Nov  3 12:32 ldap.conf
drwxr-xr-x 2 root root 4096 Oct 26 15:44 schema
drwxr-xr-x 2 ldap ldap 4096 Nov  3 12:40 slapd.d
drwxr-x--- 3 ldap ldap 4096 Nov  3 12:32 slapd.d-20201103
```

Estratègies de backup / restore de les dades

ldif

Igual que s'ha vist que podem fer amb la configuració es pot fer un bolcat de totes les dades amb ***slapcat*** i restaurar-les de nou amb ***slapadd***.

```
[root@ldap docker]# slapcat -l dades.edt.org-ldif
[root@ldap docker]# rm -rf /var/lib/ldap/*
[root@ldap docker]# slapadd  -l dades.edt.org-ldif
```

tar.gz

També es pot generar una copia de seguretat de les dades amb tar/gz desant en un fitxer tot el contingut del directori de dades ***/var/lib/ldap***.

```
[root@ldap docker]# tar -cvPzf dades.edt.org.tgz /var/lib/ldap
/var/lib/ldap/
/var/lib/ldap/lock.mdb
/var/lib/ldap/data.mdb
/var/lib/ldap/DB_CONFIG
[root@ldap docker]# rm -rf /var/lib/ldap/*
[root@ldap docker]# tar xvzPf dades.edt.org.tgz
/var/lib/ldap/
/var/lib/ldap/lock.mdb
/var/lib/ldap/data.mdb
/var/lib/ldap/DB_CONFIG
```

copia del directori

Igual que es pot fer amb la configuració el directoris de dades de lap també es pot copiar tal qual, per exemple per fer proves i restaurar-lo després.

```
[root@ldap docker]# mv /var/lib/ldap/ /var/lib/ldap-$(date +%Y%m%d)
[root@ldap docker]# mkdir /var/lib/ldap
[root@ldap docker]# chown -R ldap.ldap /var/lib/ldap
```

Utilització de Volums de Docker

Amb Docker podem crear ***volums*** de dades que proporcionen permanència, perdurabilitat, a les dades, encara que el container finalitzi i s'elimini les dades ***perduren***.

Si es vol que la configuració del servidor LDAP i les dades emmagatzemades del DIT siguin permanents es poden generar dos volums, un per la configuració i un per les dades. D'aquesta manera les dades queden emmagatzemades als volums amb independència de l'existència dels containers.

Crear els dos volums:

```
$ docker volume ls
DRIVER          VOLUME NAME

$ docker volume create ldap-data
```

```
ldap-data

$ docker volume create ldap-config
ldap-config

$ docker volume ls
DRIVER          VOLUME NAME
local           ldap-config
local           ldap-data
```

Engegar el container usant els volumes:

```
$ docker run --rm --name ldap.edt.org -h ldap.edt.org --net 2hisix -p 389:389 -v ldap-config:/etc/openldap/slapd.d -v ldap-data:/var/lib/ldap -d edtasixm06/ldap20:schema
426022f4a0efaba6831a4c8d05b1457359497b18a3dae99955eb73bfcf8292cc
```

Observar amb inspect la configuració de un volum

```
$ docker volume inspect ldap-data
[
    {
        "CreatedAt": "2020-11-03T14:14:17+01:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/ldap-data/_data",
        "Name": "ldap-data",
        "Options": {},
        "Scope": "local"
    }
]
```

Atenció tal i com està configurat el container ara s'esborra tota la base de dades i la configuració. Caldria eliminar de install.sh aquesta part. Per fer-ho ben fet ens caldrà generar un Entrypoint que permet decidir que volem fer en iniciar (construir tot de cop, usar les dades que ja hi ha, etc).

Entrypoint per decidir: initdb / initdbedt / start

Un cop s'ha aconseguit la permanència de les dades usant docker volumes ara cal repensar el funcionament de la imatge docker ldap actual, que cada cop que es crea de nou esborra del tot el contingut de la configuració i de les dades. Utilitzant ***Entrypoint*** es pot modificar l'script d'inici de manera que segons l'argument rebut faci:

- ***initdbedt***: el que s'ha fet fins ara, eliminar la configuració i les dades, generar una nova configuració amb slalptest basada en slapd.conf i carregar les dades de l'organitzaió edt.org.

- ***initdb***: igual que l'anterior però no carrega les dades de l'organització edt.org. No carrega cap dada.

- ***start***: no esborra ni la configuració ni les dades, simplement en crear el container engega el servei slapd preservant la configuració i les dades pre-existents (dels volumnes).

Exemple de Dockerfile:

```
# ldapserver
FROM fedora:32
LABEL version="1.0"
LABEL author="@edt ASIX-M06"
LABEL subject="ldapserver"
RUN dnf -y install openldap-servers openldap-clients
RUN mkdir /opt/docker
COPY * /opt/docker/
RUN chmod +x /opt/docker/startup.sh
WORKDIR /opt/docker
ENTRYPOINT [ "/bin/bash", "/opt/docker/startup.sh" ]
EXPOSE 389
```

Exemple de fitxer startup.sh

```
#! /bin/bash
ulimit -n 1024

function initdbedt(){
  rm -rf /etc/openldap/slapd.d/*
  rm -rf /var/lib/ldap/*
  cp /opt/docker/DB_CONFIG /var/lib/ldap/.
  slaptest -f /opt/docker/slapd.conf -F /etc/openldap/slapd.d
  slapadd -F /etc/openldap/slapd.d -l /opt/docker/edt.org.ldif
  chown -R ldap.ldap /etc/openldap/slapd.d
  chown -R ldap.ldap /var/lib/ldap
  cp /opt/docker/ldap.conf /etc/openldap/ldap.conf
  /sbin/slapd -d0 -h "ldap:// ldaps:// ldapi://"
}

function initdb(){
  rm -rf /etc/openldap/slapd.d/*
  rm -rf /var/lib/ldap/*
  cp /opt/docker/DB_CONFIG /var/lib/ldap/.
  slaptest -f /opt/docker/slapd.conf -F /etc/openldap/slapd.d
  chown -R ldap.ldap /etc/openldap/slapd.d
  chown -R ldap.ldap /var/lib/ldap
  cp /opt/docker/ldap.conf /etc/openldap/ldap.conf
  /sbin/slapd -d0 -h "ldap:// ldaps:// ldapi://"
}

function start(){
  /sbin/slapd -d0 -h "ldap:// ldaps:// ldapi://"
}

echo "startup: $1, $*"
case $1 in
  initdbedt)
    echo "initdbedt"
    initdbedt;;
  initdb)
    echo "initdb"
    initdb;;
  start)
    echo "start"
    start;;
  *)
    echo "altre"
    start;;
esac
#bash /opt/docker/install.sh
#/sbin/slapd -d0 ldap:// ldaps:// ldapi://
```

Engeguem-ho tot!

Un cop generades imatges de ldap i de phpldapadmin podem engegar-ho tot de cop i verificar que hi ha accés tant amb les ordres client tipus ldapsearch com gràficament amb phpldapadmin.

Podem engegar els containers:

- manualment.
- amb un script.
- usant *docker-compose*.

Exemple d'escript *run.sh*

```
$ docker run --rm --name ldap.edt.org -h ldap.edt.org --net 2hisix -p 389:389 -v
ldap-config:/etc/openldap/slapd.d -v ldap-data:/var/lib/ldap -d edtasixm06/ldap20:latest

$ docker run --rm  --name phpldapadmin.edt.org -h phpldapadmin.edt.org --net 2hisix -p
80:80 -d edtasixm06/phpldapadmin:20
```

Exemple *docker-compose.yml* amb *ldap* - *phpldapadmin*

```
version: "2"
services:
  ldap:
    image: edtasixm06/ldap20:latest
    container_name: ldap.edt.org
    hostname: ldap.edt.org
    ports:
      - "389:389"
      - "636:636"
    volumes:
      - "ldap-data:/var/lib/ldap"
      - "ldap-config:/etc/openldap/slapd.d"
    networks:
      - 2hisix
  ssh:
    image: edtasixm06/phpldapadmin:20
    container_name: phpldapadmin.edt.org
    hostname: phpldapadmin.edt.org
    ports:
     - "80:80"
    networks:
     - 2hisix
networks:
  2hisix:
volumes:
  ldap-data:
  ldap-config:
```

Exemple *docker-compose.yml* amb *ldap* - *phpldapadmin* - *portainer*

```
version: "2"
services:
```

```
  ldap:
    image: edtasixm06/ldap20:latest
    container_name: ldap.edt.org
    hostname: ldap.edt.org
    ports:
      - "389:389"
      - "636:636"
    volumes:
      - "ldap-data:/var/lib/ldap"
      - "ldap-config:/etc/openldap/slapd.d"
    networks:
     - 2hisix
  php:
    image: edtasixm06/phpldapadmin:20
    container_name: phpldapadmin.edt.org
    hostname: phpldapadmin.edt.org
    ports:
     - "80:80"
    networks:
     - 2hisix
  portainer:
    image: portainer/portainer
    container_name: portainer.edt.org
    hostname: portainer.edt.org
    ports:
     - "9000:9000"
    volumes:
     - /var/run/docker.sock:/var/run/docker.sock
networks:
  2hisix:
volumes:
  ldap-data:
  ldap-config:
```

Més enllà!

Altres activitats a realitzar per a l'aprenentatge de LDAP són:

- ❏ Monitoritzar el tràfic d'una consulta/actualització LDAP amb wireshark.
- ❏ Monitoritzar el rendiment del servidor amb la BD Monitor. Mostrar les dades usant Grafana.
- ❏ LDAP amb autenticació SASL. Tràfic segur sobre TLS.
- ❏ Servidors de rèplica: productor consumidor.
- ❏ Servei delegat: arbre d'entitats amb diversos servidors.
- ❏ Autenthicació amb SASL (PLAIN, CRAM, DIGEST, EXTERNAL i GSSAPI)
- ❏ Authenticació SASL External amb certificat digital del client.

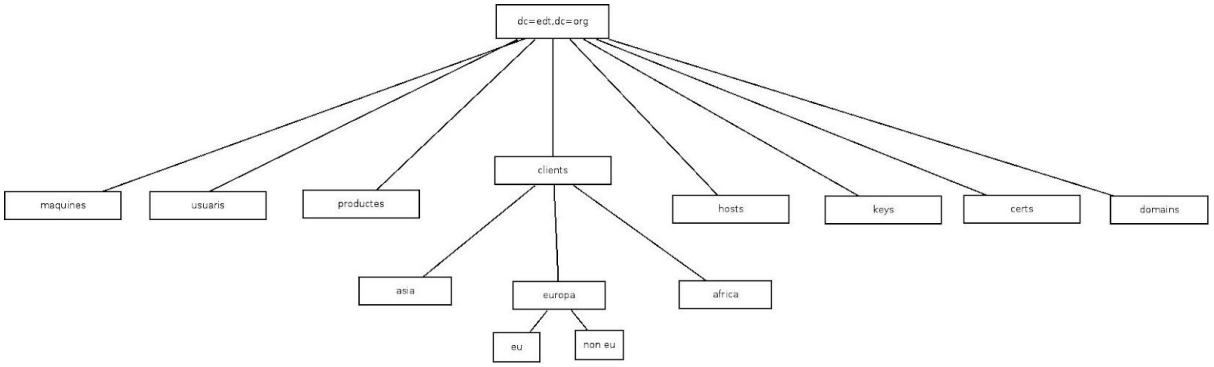
Apèndix: exemples usats

Base de dades

dc=edt,dc=org
    rootdn=Manager
    rootpwd=secret

cn=config
    rootdn=Sysadmin
    rootpw=syskey

Dades DB



dc=edt,dc=org

1) maquines clients productes
2) usuaris
3) hosts keys certs domains asia africa
   europa:
      eu noneu
4) grups
      cup admin alumnes profes asiaafrica europa
5) referral de subtree