

# LDAP Schema Design - case study

## ABSTRACT

This document describes about the design of LDAP schema by taking a case study. Explains about LDAP, object class, attributes and a case study with college administration information. Case study mentioned in this document is tested in OpenLDAP ver 2.0.18 stable version.

## SCOPE

Scope of this document is limited to explaining how to design LDAP schema with a case study. This document expects users with basic knowledge of LDAP. For more information about LDAP please refer [www.openldap.org](http://www.openldap.org)

## INTRODUCTION

### *What is LDAP?*

LDAP stands for Lightweight Directory Access Protocol. This is a lightweight protocol for accessing directory services. This runs over TCP/IP or connection oriented transfer services. The LDAP model is based on entries. Entry will have set of attributes. It will be uniquely identified by Distinguished Name (DN). Attribute can have multiple values also. Entry can be numeric, string, binary etc.. The arrangement of the LDAP is like hierarchical tree. Each node in a tree should be uniquely identified with DN. Entry will have objectclass which describes the attributes of object classes and it can have multiple objectclasses.

Following are the steps for defining new schema:

1. obtain Object Identifier
2. define custom attribute types
3. define custom object classes

Refer [www.openldap.org](http://www.openldap.org) for more information, this section is brief from OpenLDAP documentation.

### *Object Identifiers*

Each schema element is identified by a globally unique Object Identifier (OID). OIDs are also used to identify other objects. They are commonly found in protocols described by ASN.1. In particular, they are heavily used by the Simple Network Management Protocol (SNMP). As OIDs are hierarchical, your organization can obtain one OID and branch it as needed. For example, if your organization were assigned OID 1.1, you could branch the tree as follows:

OID	Assignment
1.1	Organization's OID
1.1.1	SNMP Elements
1.1.2	LDAP Elements
1.1.2.1	AttributeTypes
1.1.2.1.1	myAttribute
1.1.2.2	ObjectClasses
1.1.2.2.1	myObjectClass

You are, of course, free to design a hierarchy suitable to your organizational needs under your organization's OID. No matter what hierarchy you choose, you should maintain a registry of assignments you make.

For more information about Object Identifiers (and a listing service) see <http://www.alvestrand.no/harald/objectid/>.

*Under no circumstances should you hijack OID namespace!*

To obtain a registered OID at *no cost*, apply for an OID under the [Internet Assigned Numbers Authority](#) (IANA) maintained *Private Enterprise* arc. Any private enterprise (organization) may request an OID to be assigned under this arc. Just fill out the [IANA](#) form at <http://www.iana.org/cgi-bin/enterprise.pl> and your official OID will be sent to you usually within a few days. Your base OID will be something like 1.3.6.1.4.1.x where x is an integer.

---

**Note:** Don't let the "MIB/SNMP" statement on the IANA page confuse you. OIDs obtained using this form may be used for any purpose including identifying LDAP schema elements.

---

Alternatively, OID name space may be available from a national authority (e.g., ANSI, BSI).

For private experiments, OIDs under 1.1 may be used. The OID 1.1 arc is regarded as dead name space.

## Attribute Type Specification

The *attributetype* directive is used to define a new attribute type. The directive uses the same Attribute Type Description used by the attributeTypes attribute found in the subschema subentry, e.g.:

```
attributetype <Attribute Type Description>
```

where Attribute Type Description is defined by the following BNF:

```
AttributeTypeDescription = "(" whsp
    numericoid whsp                ; AttributeType identifier
    [ "NAME" qdescrs ]             ; name used in AttributeType
    [ "DESC" qdstring ]           ; description
    [ "OBSOLETE" whsp ]           ;
    [ "SUP" woid ]                 ; derived from this other
                                   ; AttributeType
    [ "EQUALITY" woid              ; Matching Rule name
    [ "ORDERING" woid              ; Matching Rule name
    [ "SUBSTR" woid ]              ; Matching Rule name
    [ "SYNTAX" whsp noidlen whsp ] ; Syntax OID
    [ "SINGLE-VALUE" whsp ]        ; default multi-valued
    [ "COLLECTIVE" whsp ]         ; default not collective
    [ "NO-USER-MODIFICATION" whsp ; default user modifiable
    [ "USAGE" whsp AttributeUsage ; default userApplications
    whsp ")"

AttributeUsage =
    "userApplications" /
    "directoryOperation" /
    "distributedOperation" / ; DSA-shared
    "dSAOperation"         ; DSA-specific, value depends on
server
```

where whsp is a space ( ' '), numericoid is a globally unique OID in dotted-decimal form (e.g. 1.1.1.0), qdescrs is one or more names, woid is either the name or OID optionally followed by a length specifier (e.g {10}).

For example, the attribute types name and cn are defined in core.schema as:

```
attributeType ( 2.5.4.41 NAME 'name'
    DESC 'name(s) associated with the object'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
attributeType ( 2.5.4.3 NAME ( 'cn' $ 'commonName' )
    DESC 'common name(s) associated with the object'
    SUP name )
```

Notice that each defines the attribute's OID, provides a short name, and a brief description. Each name is an alias for the OID. *slapd(8)* returns the first listed name when returning results.

The first attribute, `name`, holds values of `directoryString` (UTF-8 encoded Unicode) syntax. The syntax is specified by OID (1.3.6.1.4.1.1466.115.121.1.15 identifies the `directoryString` syntax). A length recommendation of 32768 is specified. Servers should support values of this length, but may support longer values. The field does NOT specify a size constraint, so is ignored on servers (such as *slapd*) which don't impose such size limits. In addition, the equality and substring matching uses case ignore rules. Below are tables listing commonly used syntax and matching rules.

Name	OID	Description
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean value
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 string
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII string
Integer	1.3.6.1.4.1.1466.115.121.1.27	integer
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID
Numeric String	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
Octet String	1.3.6.1.4.1.1466.115.121.1.40	arbitrary octets
Printable String	1.3.6.1.4.1.1466.115.121.1.44	printable string

Name	Type	Description
booleanMatch	equality	boolean
octetStringMatch	equality	octet string
objectIdentifierMatch	equality	OID
distinguishedNameMatch	equality	DN
uniqueMemberMatch	equality	Name with optional UID
numericStringMatch	equality	numerical
numericStringOrderingMatch	ordering	numerical
numericStringSubstringsMatch	substrings	numerical
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
caseIgnoreIA5Match	equality	case insensitive, space insensitive
caseIgnoreIA5OrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreIA5SubstringsMatch	substrings	case insensitive, space insensitive
caseExactIA5Match	equality	case sensitive, space insensitive
caseExactIA5OrderingMatch	ordering	case sensitive, space insensitive
caseExactIA5SubstringsMatch	substrings	case sensitive, space insensitive

The second attribute, *cn*, is a subtype of *name* hence it inherits the syntax, matching rules, and usage of *name*. *commonName* is an alternative name.

Neither attribute is restricted to a single value. Both are meant for usage by user applications. Neither is obsolete nor collective.

## ***Object Class Specification***

The *objectclasses* directive is used to define a new object class. The directive uses the same Object Class Description used by the *objectClasses* attribute found in the subschema subentry, e.g.:

```
objectclass <Object Class Description>
```

where Object Class Description is defined by the following BNF:

```
ObjectClassDescription = "(" whsp
    numericoid whsp      ; ObjectClass identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    [ "SUP" oids ]       ; Superior ObjectClasses
    [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                        ; default structural
    [ "MUST" oids ]      ; AttributeTypes
    [ "MAY" oids ]       ; AttributeTypes
    whsp ")"
```

where *whsp* is a space (' '), *numericoid* is a globally unique OID in numeric form (e.g. 1.1.0), *qdescrs* is one or more names, and *oids* is one or more names and/or OIDs.

## LDIF Format

The LDAP Data Interchange Format (LDIF) is used to represent LDAP entries in a simple text format. This section provides a brief description of the LDIF entry format

The basic form of an entry is:

```
# comment
dn: <distinguished name>
<attrdesc>: <attrvalue>
<attrdesc>: <attrvalue>
...
```

Lines starting with a '#' character are comments. An attribute description may be a simple attribute type like `cn` or `objectClass` or `1.2.3` (an OID associated with an attribute type) or may include options such as `cn;lang_en_US` or `userCertificate;binary`.

A line may be continued by starting the next line with a *single* space or tab character. For example:

```
dn: branch=Electronics Communication,dept=Civil,o=
  college
branch: Electronics
  Communication
```

is equivalent to:

```
dn: branch=Electronics Communication,dept=Civil,o=college
branch: Electronics Communication
```

Multiple attribute values are specified on separate lines. e.g.,

```
BoardMember: abc
BoardMember: cde
```

If an <attrvalue> contains non-printing characters or begins with a space, a colon (':'), or a less than ('<'), the <attrdesc> is followed by a double colon and the base64 encoding of the value. For example, the value " begins with a space" would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

You can also specify a URL containing the attribute value. For example, the following specifies the `jpegPhoto` value should be obtained from the file `/path/to/file.jpeg`.

```
cn:< file:///path/to/file.jpeg
```

Multiple entries within the same LDIF file are separated by blank lines. Here's an example of an LDIF file containing three entries.

```

# Barbara's Entry
dn: cn=Barbara J Jensen,dc=example,dc=com
cn: Barbara J Jensen
cn: Babs Jensen
objectClass: person
sn: Jensen

# Bjorn's Entry
dn: cn=Bjorn J Jensen,dc=example,dc=com
cn: Bjorn J Jensen
cn: Bjorn Jensen
objectClass: person
sn: Jensen
# Base64 encoded JPEG photo
jpegPhoto:: /9j/4AAQSkZJRgABAAAAQABAAD/2wBDABALD
A4MChAODQ4SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQ
ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG

# Jennifer's Entry
dn: cn=Jennifer J Jensen,dc=example,dc=com
cn: Jennifer J Jensen
cn: Jennifer Jensen
objectClass: person
sn: Jensen
# JPEG photo from file
jpegPhoto:< file:///path/to/file.jpeg

```

Notice that the jpegPhoto in Bjorn's entry is base 64 encoded and the jpegPhoto in Jennifer's entry is obtained from the location indicated by the URL.

---

**Note:** Trailing spaces are not trimmed from values in an LDIF file. Nor are multiple internal spaces compressed. If you don't want them in your data, don't put them there.

## COLLEGE LDAP Schema - CASE STUDY

### *Description*

This document describes about design of ldap schema for a college administration department. College will have various departments like mechanical, chemical, Electronics, civil etc.. Each department will of various branches. For example electronics department can have Electronics & Communication Engineering, Electronics & Electrical Engineering, Electronics & Instrumentation Engineering, Computer Science Engineering, Computer Science & Information Technology, Electronics & Computer Science etc.. Each branch will have in charge, number of students, teaching and non teaching staff. Each student will be attending set of subjects for an academic year. Each teaching staff & non teaching staff will have salary details. Framing the requirements into hierarchical structure, following LDAP tree depicts the tree structure.

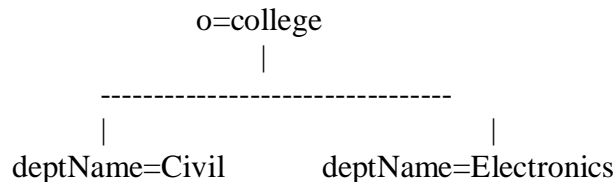
## LDAP Schema design

From the above description, the objectclasses identified are COLLEGE, DEPARTMENT, BRANCH, ACADEMIC, STUDENT, SUBJECTS, STAFF and SALARY and each object class consists of following attributes.

COLLEGE	DEPARTMENT	BRANCH	ACADEMIC	STUDENT
collegeName	deptName	branchName	academicYear	studentName
collegePrincipal	deptHead	BranchHead		studentNumber
collegePresident		branchNumberOfStudents		studentAddress
collegeSecretary				studentAcademic
collegeBoardMember				studentCourse
				studentStatus
				studentYear

SUBJECTS	STAFF	SALARY
subjectName	staffName	salaryBASIC
subjectMarks	staffID	salaryDA
subjectCode	staffAddress	salaryHRA
	staffQualification	salaryAllowance
	staffDesignation	salaryTotal
	staffBranch	
	staffType	

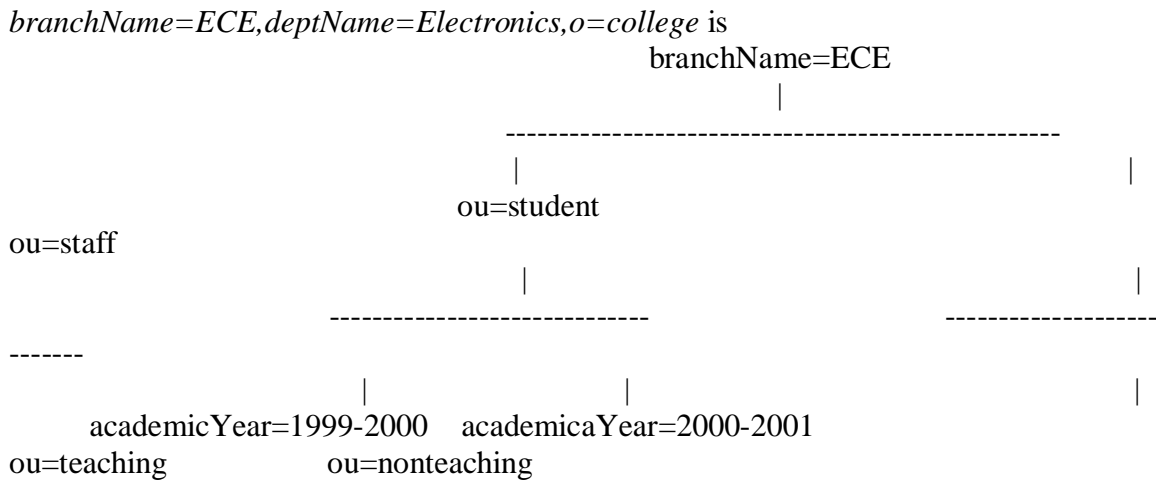
The top node is *college* with unique DN as *o=college*. College will have different departments like Civil, Electronics, Mechanical etc.. The number of child nodes depends on the number of departments in the college. Let us take, college has two departments Civil and Electronics. Then the *o=college* parent node will have two child nodes *deptName=Civil* and *deptName=Electronics*. Each node can be identified as *deptName=Civil,o=college* and *deptName=Electronics,o=college*. If searched with DN *deptName=Electronics,o=college*, attributes at this entry will be retrieved. The subtree is



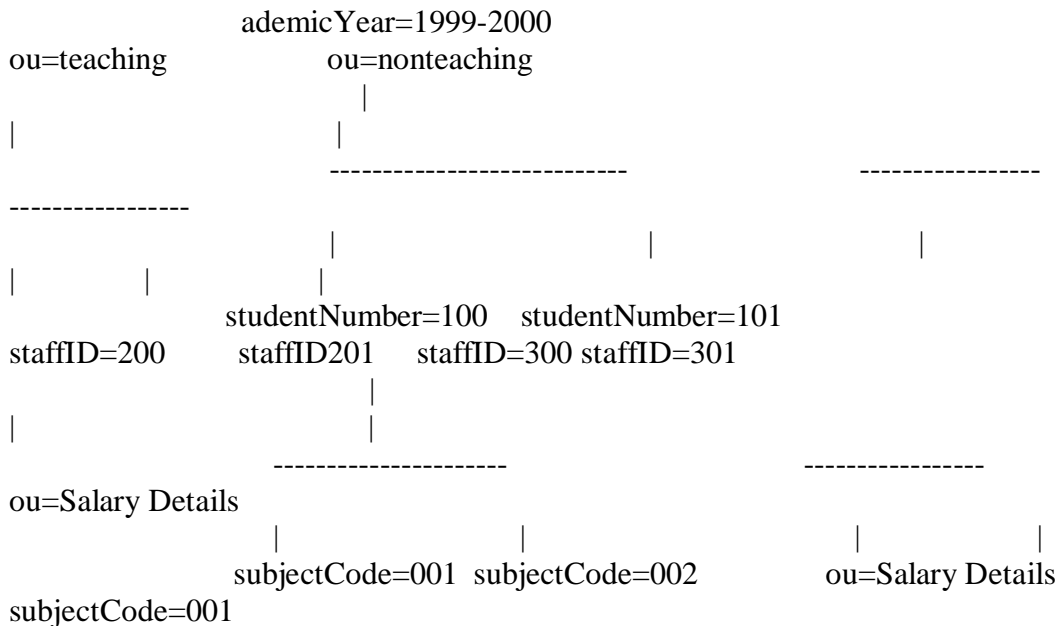
Consider the department Electronics has three branches ECE, EEE & CSE. The DN for each branch will be *branchName=ECE,deptName=Electronics,o=college*,

*branchName=EEE,deptName=Electronics,o=college* and  
*branchName=CSE,deptName=Electronics,o=college*.

Now take the branch ECE, each branch will have students and staff. Each student belongs to a particular year and each staff belongs to either teaching or non-teaching staff. So consider the sub tree under the dn

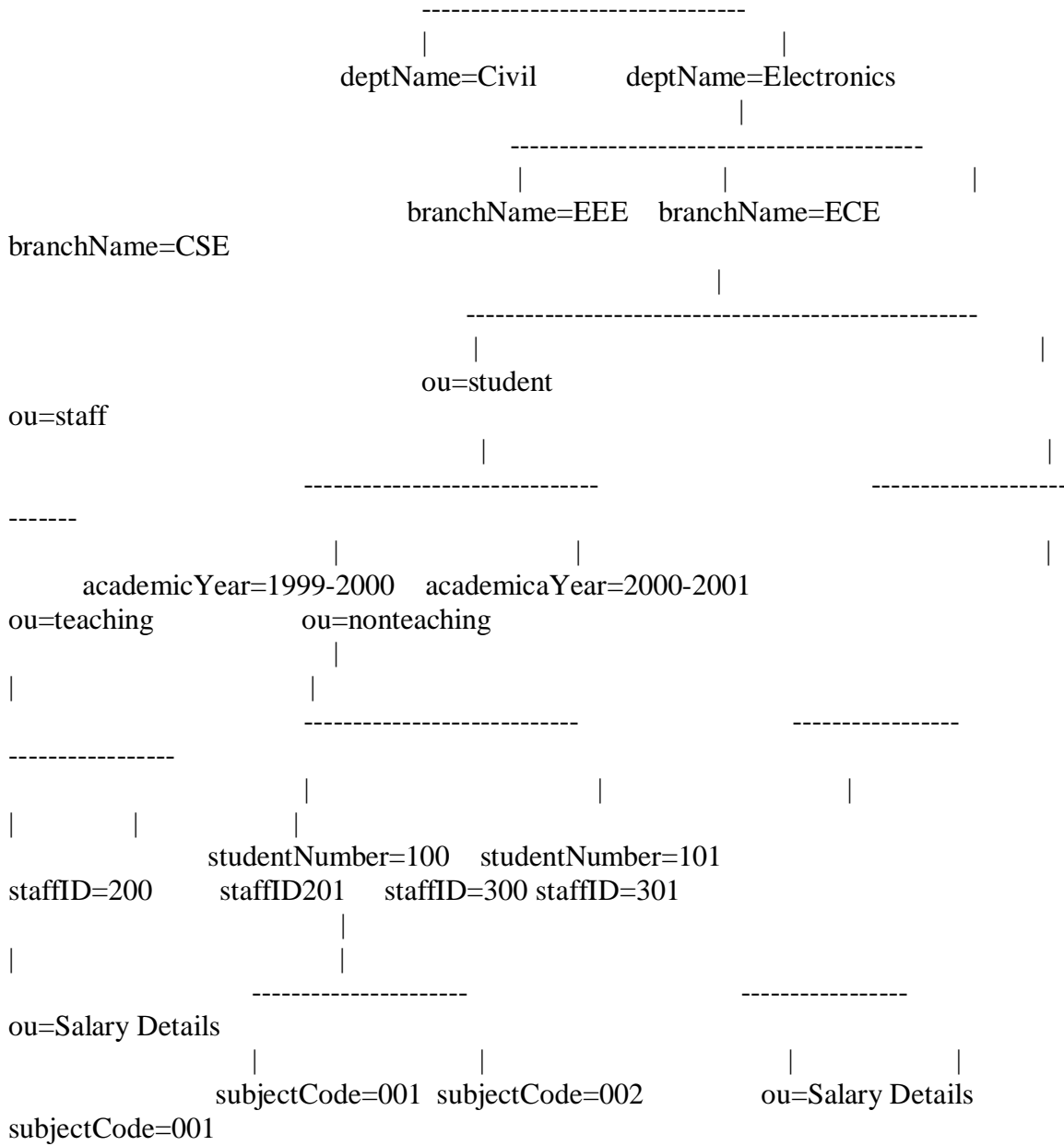


Each academicYear consists of students and each student will be studying the subjects, each staff will have salary details and subjects he/she is handling. The subtrees for academicYear, teaching and nonteaching are



### **COLLEGE LDAP Tree**

o=college  
|



## **Attributes**

This section contains attribute file for College LDAP tree

```

attributetype ( 1.3.6.1.4.1.2.1 NAME 'collegeName'
  DESC 'college name'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15

```

SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.2 NAME 'collegePrincipal'  
   DESC 'college principal name'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.3 NAME 'collegePresident'  
   DESC 'college president'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.4 NAME 'collegeSecretary'  
   DESC 'college secretary name'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.5 NAME 'collegeBoardMember'  
   DESC 'board member name'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   )  
 attributetype ( 1.3.6.1.4.1.2.6 NAME 'deptName'  
   DESC 'description'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.7 NAME 'deptHead'  
   DESC 'description'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.8 NAME 'branchName'  
   DESC 'board member name'  
   EQUALITY caseIgnoreMatch  
   SUBSTR caseIgnoreSubstringsMatch  
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
   SINGLE-VALUE )  
 attributetype ( 1.3.6.1.4.1.2.9 NAME 'branchHead'  
   DESC 'description'  
   EQUALITY caseIgnoreMatch

SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.10 NAME 'branchNumberOfStudents'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
  
attributetype ( 1.3.6.1.4.1.2.11 NAME 'studentName'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.12 NAME 'studentNumber'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)  
attributetype ( 1.3.6.1.4.1.2.13 NAME 'studentAddress'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.14 NAME 'studentAcademicYear'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.15 NAME 'studentCourse'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE)  
attributetype ( 1.3.6.1.4.1.2.16 NAME 'studentStatus'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.17 NAME 'studentYear'

DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.18 NAME 'subjectName'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.19 NAME 'subjectMarks'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
  
attributetype ( 1.3.6.1.4.1.2.20 NAME 'subjectCode'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
  
attributetype ( 1.3.6.1.4.1.2.21 NAME 'staffName'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.22 NAME 'staffID'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.23 NAME 'staffAddress'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.24 NAME 'staffQualification'  
DESC 'description'  
EQUALITY caseIgnoreMatch

SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.25 NAME 'staffDesignation'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.26 NAME 'staffDepartment'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.27 NAME 'staffBranch'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.28 NAME 'staffType'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.29 NAME 'salaryBasic'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.30 NAME 'salaryDA'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.31 NAME 'salaryHRA'  
DESC 'description'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
SINGLE-VALUE )  
attributetype ( 1.3.6.1.4.1.2.32 NAME 'salaryAllowance'

```

DESC 'description'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )
attributetype ( 1.3.6.1.4.1.2.33 NAME 'salaryTotal'
DESC 'description'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )
attributetype ( 1.3.6.1.4.1.2.34 NAME 'academicYear'
DESC 'description'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )

```

## ***Object class***

This section gives objectclass details for College LDAP tree.

```

objectclass ( 1.3.6.1.4.1.1.1 NAME 'objCollege'
SUP top STRUCTURAL
DESC 'object class'
MUST (collegeName $ collegePrincipal $ collegePresident $ collegeSecretary
)
MAY (collegeBoardMember ) )
objectclass ( 1.3.6.1.4.1.1.2 NAME 'objDepartment'
SUP top STRUCTURAL
DESC 'object class'
MUST (deptName $ deptHead ) )
objectclass ( 1.3.6.1.4.1.1.3 NAME 'objBranch'
SUP top STRUCTURAL
DESC 'object class'
MUST (branchHead $ branchName $ branchNumberOfStudents ) )
objectclass ( 1.3.6.1.4.1.1.4 NAME 'objStudent'
DESC 'object class'
SUP top STRUCTURAL
MUST ( studentName $ studentNumber $ studentAddress $
studentAcademicYear $ studentCourse $
studentStatus $ studentYear ) )
objectclass ( 1.3.6.1.4.1.1.5 NAME 'objSubject'

```

```
DESC 'object class'  
SUP top STRUCTURAL  
MUST ( subjectCode $subjectName $ subjectMarks ) )
```

```
objectclass ( 1.3.6.1.4.1.1.6 NAME 'objStaff'  
DESC 'object class'  
SUP top STRUCTURAL  
MUST ( staffName $ staffID $ staffAddress $ staffQualification  
$ staffDepartment $ staffBranch $ staffType) )
```

```
objectclass ( 1.3.6.1.4.1.1.7 NAME 'objSalary'  
SUP top STRUCTURAL  
DESC 'object class'  
MUST ( salaryBASIC $ salaryDA $ salaryHRA $ salaryAllowance $  
salaryTotal) )
```

```
objectclass ( 1.3.6.1.4.1.1.8 NAME 'objAcademic'  
DESC 'object class'  
SUP top STRUCTURAL  
MUST ( academicYear ) )
```

### ***Sample LDIF File***

This section gives sample ldif file for creating LDAP tree,

```
dn: o=college  
o: college  
objectclass: organization  
objectclass: objCollege  
objectclass: top  
Description: top object class  
collegeName: ABC Engineering college  
collegePrincipal: John Peter  
collegePresident: Samual A  
collegeSecretary: Benson K  
collegeBoardMember: Jennifer King  
collegeBoardMember: George K Danial
```

```
dn: deptName=Civil,o=college  
deptName: Civil  
objectclass: objDepartment  
objectclass: top  
deptHead: abc
```

```
dn: deptName=Electronics,o=college  
deptName: Civil  
objectclass: objDepartment
```

objectclass: top  
deptHead=defg

dn: branchName=ECE,deptName=Electronics,o=college  
branchName: ECE  
objectclass: top  
objectclass: objBranch  
branchHead: kkk  
branchNumberOfStudents: 50

dn: ou=student,branchName=ECE, deptName=Electronics,o=college  
ou: student  
objectclass: top  
objectclass: organizationalUnit

dn: ou=staff,branchName=ECE, deptName=Electronics,o=college  
ou: staff  
objectclass: top  
objectclass: organizationalUnit

dn: academicYear=1999-2000,ou=student,branchName=ECE,  
deptName=Electronics,o=college  
academicYear: 1999-2000  
objectclass: top  
objectclass: objAcademic

dn: academicYear=2000-2001,ou=student,branchName=ECE,  
deptName=Electronics,o=college  
academicYear: 2000-2001  
objectclass: top  
objectclass: objAcademic

dn: studentNumber=100,academicYear=1999-2000,ou=student,branchName=ECE,  
deptName=Electronics,o=college  
studentNumber: 100  
objectclass: top  
objectclass: objStudent  
studentName: abc  
studentAddress: abc  
studentAcademicYear: 1999-2000  
studentCourse: BS  
studentStatus: Active  
studentYear: 1999

dn: studentNumber=101,academicYear=1999-2000,ou=student,branchName=ECE,  
deptName=Electronics,o=college

studentNumber: 101  
objectclass: top  
objectclass: objStudent  
studentName: def  
studentAddress: def  
studentAcademicYear: 1999-2000  
studentCourse: BS  
studentStatus: Active  
studentYear: 1999

dn: subjectCode=001,studentNumber=101,academicYear=1999-2000,ou=student,branchName=ECE, deptName=Electronics,o=college  
subjectCode: 001  
objectclass: top  
objectclass: objSubject  
subjectName: Mathematics-1  
subjectMarks: 90

dn: subjectCode=002,studentNumber=101,academicYear=1999-2000,ou=student,branchName=ECE, deptName=Electronics,o=college  
subjectCode: 002  
objectclass: top  
objectclass: objSubject  
subjectName: Electrical Technology  
subjectMarks: 85

dn: ou=teaching,ou=staff,branchName=ECE, deptName=Electronics,o=college  
ou: teaching  
objectclass: top  
objectclass: organizationalUnit

dn: ou=nonteaching,ou=staff,branchName=ECE, deptName=Electronics,o=college  
ou: nonteaching  
objectclass: top  
objectclass: organizationalUnit

dn: staffID=001,ou=teaching,ou=staff,branchName=ECE, deptName=Electronics,o=college  
staffID: 001  
objectclass: top  
objectclass: objStaff  
staffName: abc

staffAddress: sdfasjflasd  
staffQualification: MS, Phd  
staffDepartment: Electronics

staffBranch: ECE  
staffType: teaching

dn: staffID=002,ou=teaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
staffID: 002  
objectclass: top  
objectclass: objStaff  
staffName: def  
staffAddress: saas  
staffQualification: MS, Phd  
staffDepartment: Electronics  
staffBranch: ECE  
staffType: teaching

dn: staffID=101,ou=nonteaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
staffID: 101  
objectclass: top  
objectclass: objStaff  
staffName: abc  
staffAddress: sdfasjflasd  
staffQualification: BS  
staffDepartment: Electronics  
staffBranch: ECE  
staffType: nonteaching

dn: staffID=102,ou=nonteaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
staffID: 102  
objectclass: top  
objectclass: objStaff  
staffName: def  
staffAddress: saas  
staffQualification: B.Com  
staffDepartment: Electronics  
staffBranch: ECE  
staffType: nonteaching

dn: ou=Salary Details,staffID=002,ou=teaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
ou: Salary Details  
objectclass: top  
objectclass: objSalary  
objectclass: organizationalUnit  
salaryBasic: 10000

salaryDA: 20000  
salaryHRA: 4000  
salaryAllowance: 6000  
salaryTotal: 40000

dn: subjectCode=001,staffID=002,ou=teaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
subjectCode: 001  
objectclass: top  
objectclass: objSubject  
subjectName: Electrician-1  
subjectMarks: 100

dn: ou=Salary Details,staffID=102,ou=nonteaching,ou=staff,branchName=ECE,  
deptName=Electronics,o=college  
ou: Salary Details  
objectclass: top  
objectclass: objSalary  
objectclass: organizationalUnit  
salaryBasic: 1000  
salaryDA: 2000  
salaryHRA: 400  
salaryAllowance: 600  
salaryTotal: 4000

## **CONCLUSION**

Copy the ldap object class file and ldap attribute file and run LDAP server. Add the ldif file into LDAP server and check the LDAP tree. Above files are tested using OpenLDAP v2.0.18 stable version. This is only an approach for designing LDAP schema. This can be tuned/modified to requirements of a college.

## **REFERENCES**

[1] [www.openldap.org](http://www.openldap.org) describes about OpenSource LDAP(OpenLDAP)

## **ACRONYMS**

LDAP	Lightweight Directory Access Protocol
WWW	World Wide Web
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Security Layer
HTML	HyperText Markup Language
TCP/IP	Transmission Control Protocol/Internet Protocol
SSL	Secure Socket Layer