LDAP搜索底层逻辑

一，ldap基本模型

DIT ： ldap的树结构，每个树上的节点叫entry

DN（Distinguished
Name)：从特定节点到树的根的直接下级的节点的路径序列为DN，区别于其他节点的名字，相当于节点的绝对路径，由不同部分的rdn组成

RDN（Relative Distinguished Name)：区别于同层节点的名字叫RDN，相当于相对路径

Attribute：每个节点属性都有相应的value，存放在文件中

objectClass：对象类（ObjectClass）是属性的集合，LDAP预想了很多人员组织机构中常见的对象，并将其封装成对象类。

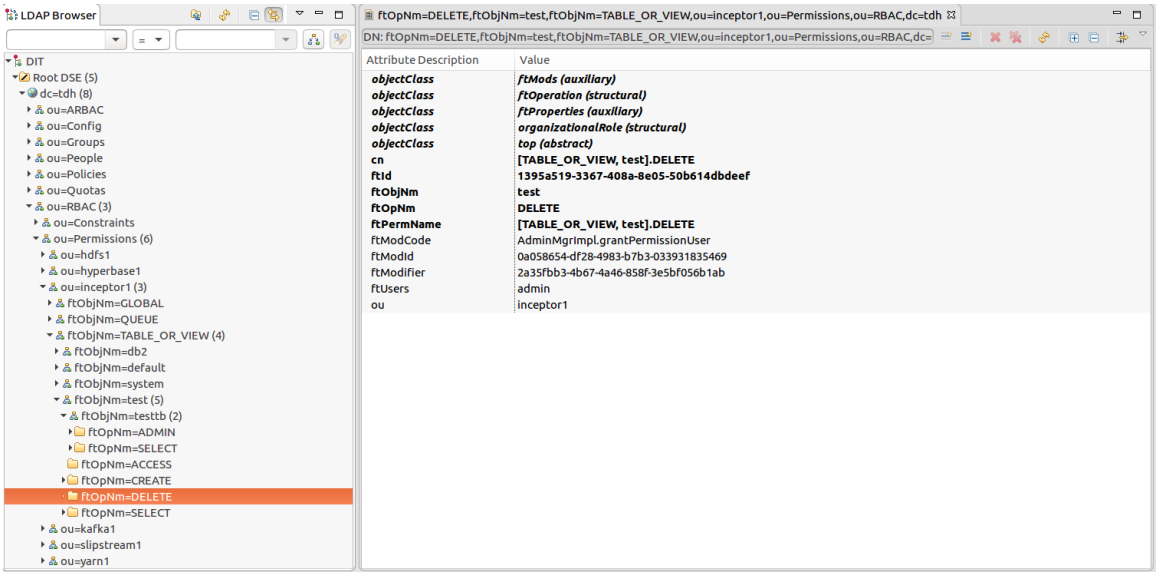
schema：对象类、属性类型、语法分别约定了条目、属性、值。这些构成了模式(Schema)，模式中的每一个元素都有唯一的OID编号

Cursor：游标，指向查找的目前节点，有向前插入和向后插入两种方式，通过此游标可以存/取/删除数据项

search scope：查询范围，默认有三种。baseObject：作用域仅限于由命名的条目；singleLevel：baseobject的直接子节点一层；wholeSubtree： baseObject及其所有子树集合

search filter：分10种操作情况，分为相应的过滤器
and/or/not/equalityMatch/substrings/greaterOrEqual/lessOrEqual/present/approcMarch/extensibleMatch

```
filterextensible     = ( attr [dnattrs]
                         [matchingrule] COLON EQUALS assertionvalue )
                       / ( [dnattrs]
                           matchingrule COLON EQUALS assertionvalue )
```



数据在ldap的存储形式

二，底层搜索操作

执行搜索权限时的操作

客户端通过guardian插件，经过guardian接入apacheds，通过传入的perm数据会在后端提取信息并通过这些信息（datasource和component）得到一个自己具有唯一可区别的名称Dn(getDn)，对应于ladp目录树的一个节点，保存在permObjDn中。

根据perm数据的prefix，substring，action形成一个过滤器字符串存在filterbuf中，scope和cookie单独作为过滤条件传到search部分。

于此同时拉取存在缓存中的该用户对应的role和group权限，一并合并到filterbuf字段中。

将所有的过滤信息构建一个searchcursor，每次把符合条件的当前节点解析到perm中，然后pop出去，进行下一轮查找，最后形成一个permlist

有继承的情况会包装成一个组，然后照上面的逻辑再进行一次perm的查询过滤。

ldap把请求信息封装成一个searchRequest，具体结构见链接：https://tools.ietf.org/html/rfc4511

然后经过拦截器，apacheds总共有14个拦截器，search操作一共7个，每个Interceptor的调用会依据它的声明顺序依次执行。

然后接入ldap底层的lmdb，ldap的底层数据库用的是LMDB搜索操作，先通过优化器优化查询语句（通过scan
count字段，把索引关联比较少的查询节点放在前面）然后根据属性的匹配原则来定义evaluator，在cursor移动时执行设定的evaluate操作，实现节点过滤。最后根据节点的dn长度和接收请求的sortkey来确定返回顺序。

LMDB的全称是LightningMemory-Mapped
Database，闪电般的内存映射数据库。它文件结构简单，一个文件夹，里面一个数据文件，一个锁文件。数据随意复制，随意传输。它的访问简单，不需要运行单独的数据库管理进程，只要在访问数据的代码里引用LMDB库，访问时给文件路径即可。基于文件映射IO和B+树的key-value接口。通过mvcc事务处理保证读取内容不被修改，进行目录访问。

lmdb使用mmap,
同时在创建env对象时，数据库已经被整个映射进整个进程空间，因此系统在映射时，会给数据库文件保留全部地址空间，从而在根据上述算法获取真实数据库，系统触发缺页错误，进而从数据文件中获取整个页面内容。cursor对象是进行所有数据库操作的对象，读写都是基于游标进行。进行读写操作时，首先需要根据条件确定页面位置，从而获得一个游标，应用程序根据游标对象操作数据库。搜索阶段分为页搜索和cursor搜索，时从B-Tree根节点检索，根据key的值，从根节点开始遍历子树获取每一层对应的page，在page之内检索key，再根据B-Tree查找方法确定下一层子节点的page，层层遍历，从而最终确定key的位置或者判断
B-Tree中没有对应的key。同时将页面存放到cursor页堆栈中。这样cursor将可以重用对应的页面，为后续进行更新等操作提供便利。

lmdb的结构与底层搜索参照：http://wiki.dreamrunner.org/public_html/C-C++/Library-Notes/LMDB.html

参考文献： Chu H. MDB: A memory-mapped database and backend for OpenLDAP[C]//Proceedings of the 3rd International Conference on LDAP, Heidelberg, Germany. 2011: 35.