NoSQL²: Store LDAP Data in HBase

Stefan Seelmann
seelmann@apache.org

Apache Directory

About me

- Stefan Seelmann
- Freelancer
  - Software development with Java
  - LDAP, Identity Management
- Open Source developer
  - Apache Directory project
  - DataNucleus LDAP store

Apache Directory

Agenda

- Apache Directory project
- LDAP is NoSQL
- Motivation for HBase backend
- Schema Design: how LDAP data fits into Hbase
  - LDAP Information Model
  - LDAP Naming Model
  - LDAP Functional Model
- Why HBase? And why not?
- Status, Future
- Demo

Apache Directory Server

- Directory Server in Java
- Protocols: LDAP, DNS, DHCP, NTP, Kerberos, Change Password
- X.500 ACI, Triggers, Stored Procedures
- Open Group certified LDAPv3 server
- In progress: ChangeLog, Replication (RFC 4533), Configuration in DIT

Apache Directory Studio

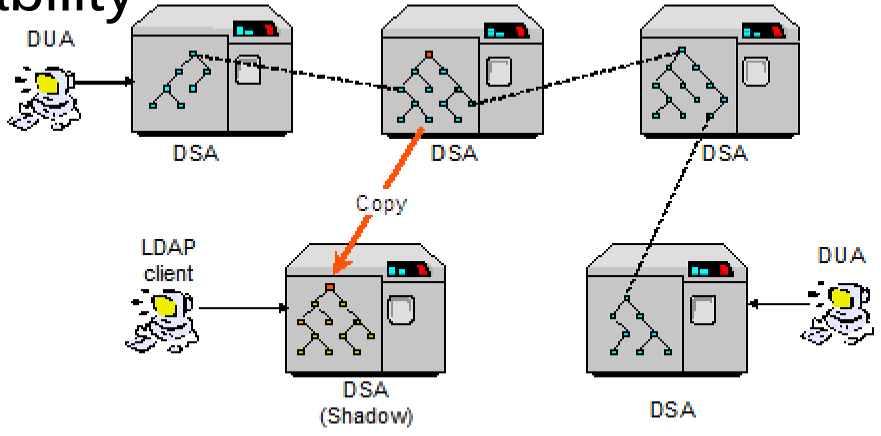
- Directory client platform
- Eclipse based
  - Integration into Eclipse
  - Application for Linux (x32/x64), Mac OS X, Windows
- Features:
  - LDAP Browser
  - LDIF Editor
  - Schema Editor
  - Integrated ApacheDS
  - Configuration for ApacheDS

eclipse COMMUNITY AWARDS 2009 WINNER

Apache Directory

LDAP is NoSQL

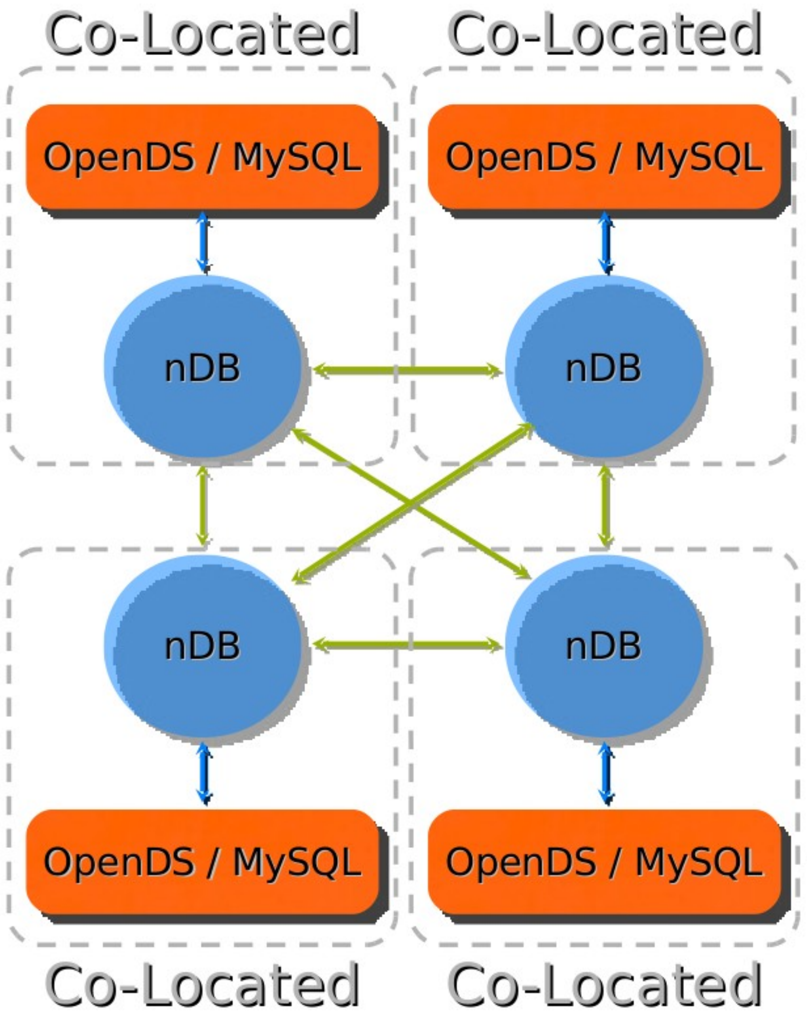
- Classical Backends
  - B+tree based: BerkleyDB, FLAIM, JET Blue, JDBM
  - Oracle and IBM use their RDBMS
- Replication
  - Proprietary protocols, now RFC 4533 (content sync)
  - multi-master for high availability
  - master-slave for read-scalability
  - „eventually consistent“
- Partitioning
  - distributed tree
  - for write-scalability



Apache Directory

Motivation

- LDAPCon2009: OpenLDAP and OpenDS projects presented MySQL NDB backend
  - using relations
  - max. DN depth
  - fixed value size
  - no substring matching



http://www.symas.com/ldapcon2009/papers/NDB_ldapcon2009.pdf

Apache Directory

Information Model (1)

- Entry consists of a set of attributes
- Attribute description and one or many values
- Schema:
  - object classes
  - attribute types
  - syntaxes
  - matching rules

LDIF notation

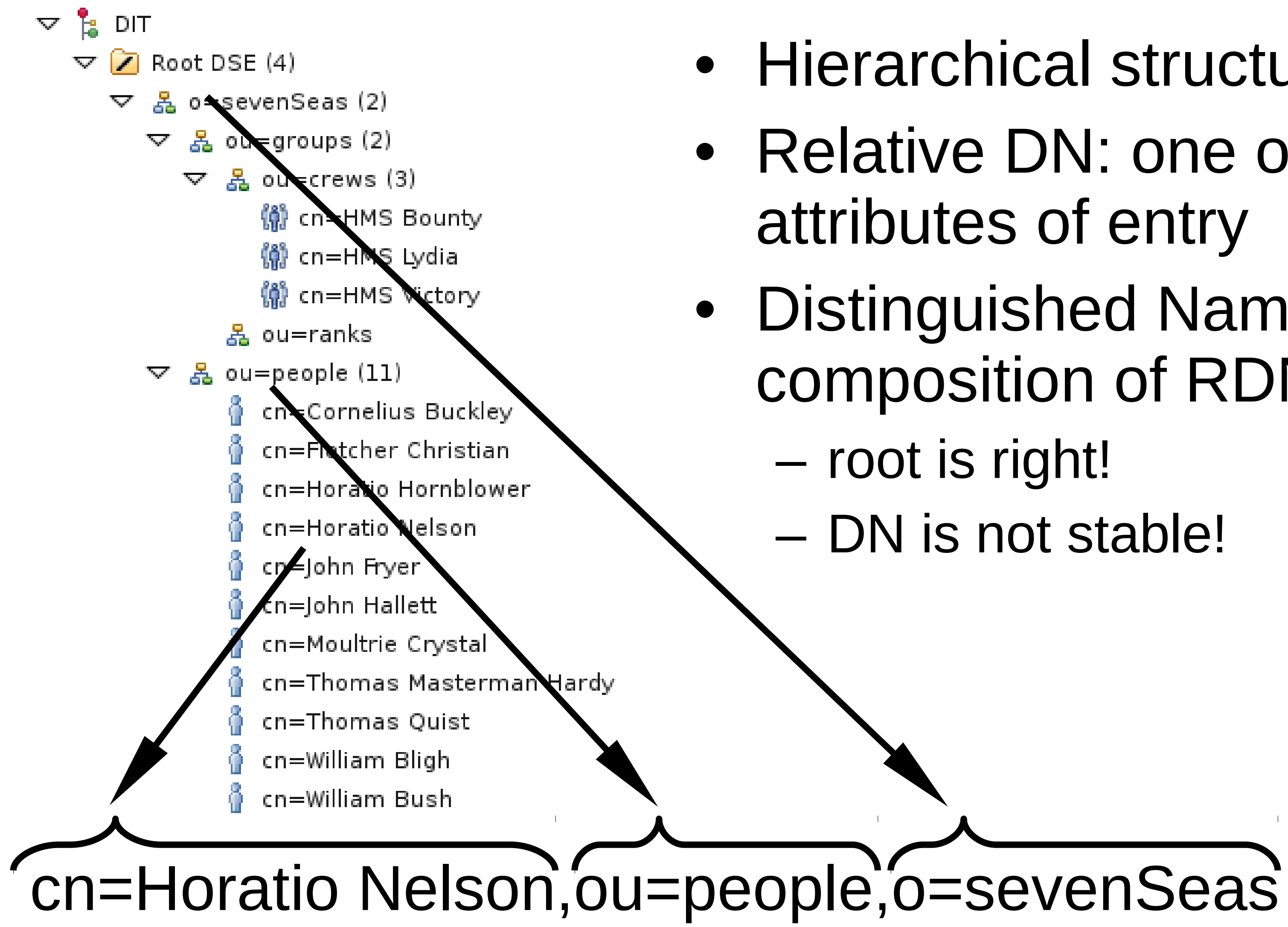
```
...
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Horatio Nelson
uid: hnelson
mail: hnelson@example.com
userPassword:: e1NIQX01ZW42RzZNZXpScm...
jpegPhoto:: /9j/4AAQSkZJRgABAQAAAQABA....
entryUUID: 44444444-4444-4444-4444-444444444444
createTimestamp: 20100506180000Z
...
```

Information Model (2)

- Table „master“
- Column family „upAttributes“ for attributes
- One row per entry, entryUUID as row key
- Additional index for multi-valued attributes

```
---------------------------------------------------------------
| master       | treeInfo | upAttributes                      |
---------------------------------------------------------------
| 44444444-... | ...      | objectClass0 -> top               |
|              |          | objectClass1 -> person            |
|              |          | objectClass2 -> organizationalPerson |
|              |          | objectClass3 -> inetOrgPerson     |
|              |          | cn0 -> Horatio Nelson             |
|              |          | uid0 -> hnelson                   |
|              |          | mail0 -> hnelson@example.com      |
|              |          | userPassword0 -> <bytes>          |
|              |          | jpegPhoto0 -> <bytes>             |
|              |          | entryUUID0 -> 44444444-4444-4444-4444-444444444444 |
|              |          | createTimestamp -> 20100506180000Z |
|              |          | ...                               |
---------------------------------------------------------------
```

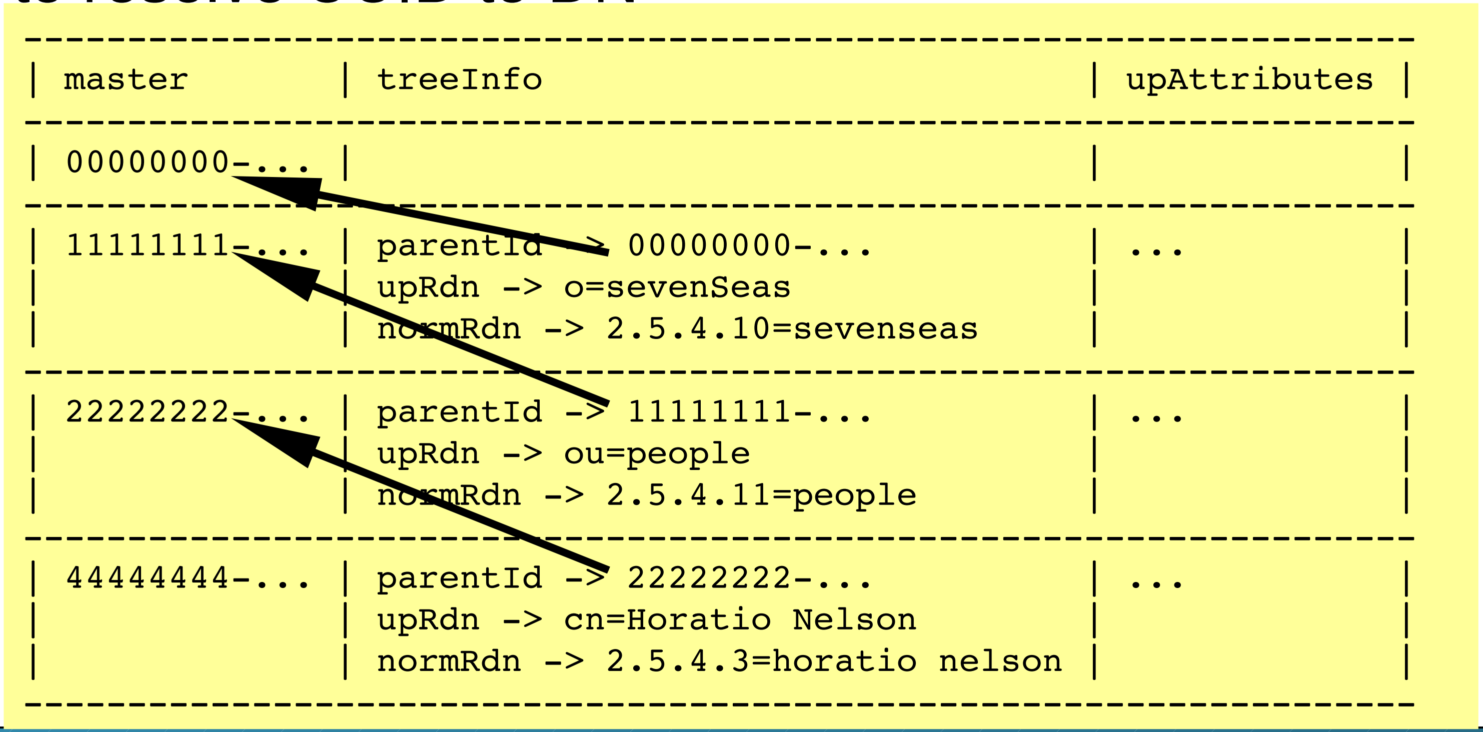
Naming Model (1)



- Hierarchical structure
- Relative DN: one or multiple attributes of entry
- Distinguished Name: composition of RDNs till root
  - root is right!
  - DN is not stable!

Naming Model (2)

- Table "master"
  - column family „treeInfo“: parentId, upRdn, normRdn
  - don't store DN, only RDN and pointer to parent
  - used to resolve UUID to DN



```
-------------------------------------------------------------------
| master        | treeInfo                          | upAttributes |
-------------------------------------------------------------------
| 00000000-...  |                                   |              |
-------------------------------------------------------------------
| 11111111-...  | parentId -> 00000000-...          | ...          |
|               | upRdn -> o=sevenSeas              |              |
|               | normRdn -> 2.5.4.10=sevenseas     |              |
-------------------------------------------------------------------
| 22222222-...  | parentId -> 11111111-...          | ...          |
|               | upRdn -> ou=people                |              |
|               | normRdn -> 2.5.4.11=people        |              |
-------------------------------------------------------------------
| 44444444-...  | parentId -> 22222222-...          | ...          |
|               | upRdn -> cn=Horatio Nelson        |              |
|               | normRdn -> 2.5.4.3=horatio nelson |              |
-------------------------------------------------------------------
```

Naming Model (3)

- Table "tree"
  - column family „treeInfo“
  - used to resolve DN to UUID

```
-----------------------------------------------------------------------------
| tree                                  | treeInfo             | normAttributes |
-----------------------------------------------------------------------------
| 00000000-...,2.5.4.10=sevenseas       | id -> 11111111-...   | ...            |
|                                       | ...                  |                |
-----------------------------------------------------------------------------
| 11111111-...,2.5.4.11=people          | id -> 22222222-...   | ...            |
|                                       | ...                  |                |
-----------------------------------------------------------------------------
| 22222222-...,2.5.4.3=horatio nelson   | id -> 44444444-...   | ...            |
|                                       | ...                  |                |
-----------------------------------------------------------------------------
```

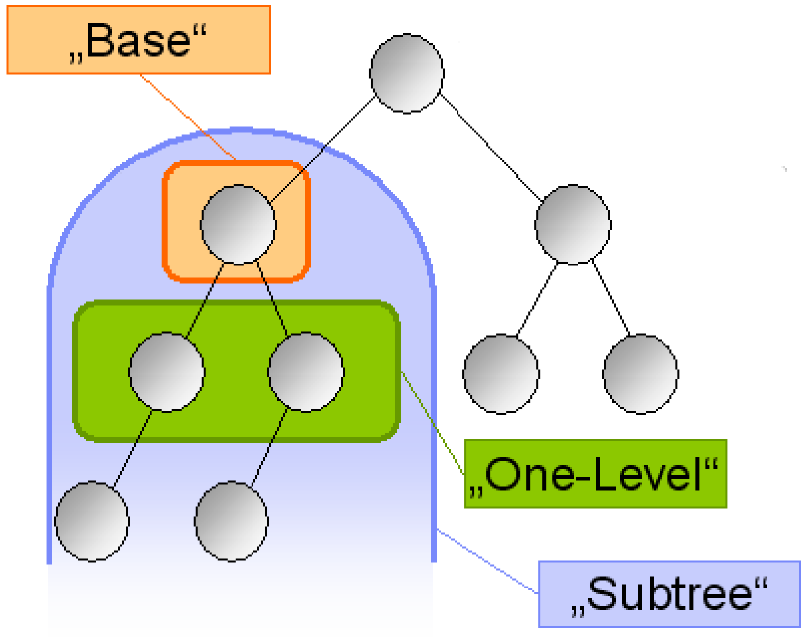
- Fast enough for flat trees
- Cache branch entries, with TTL

Functional Model - Write

- add entry
  - put()
- modify entry
  - get() + apply modifications + checkAndPut()
  - null out deleted attributes, for audit log
- delete entry
  - null out attributes
  - no real delete, just mark as deleted, for audit log
- moddn (move, rename)
  - just update pointers to parent

Functional Model - Search (1)

- Parameters
  - base DN
  - scope: base, one, sub
  - filter
    - equal, substring, presense
    - less than eq, greater than eq
    - and, or, not
    - (uid=hnelson)
    - (&(objectClass=person)(mail=*@example.com)(modifyTimestamp>=20100101)(!(uid=*)))
- Procedure:
  - get each entry in scope and check if filter matches



Functional Model - Search (2)

- For luck HBase has scanner with filters
- Normalized attributes in „tree“ table
  - qualifier is attribute description + value
  - SingleColumnQualiferFilter
  - one-level and sub-level count

```
-----------------------------------------------------------------------------------------------------
| tree                  | treeInfo             | normAttributes                                      |
-----------------------------------------------------------------------------------------------------
| 22222222-...,2.5.4.3  | id -> 44444444-...   | 0.9.2342.19200300.100.1.1=hnelson -> 0              |
|   =horatio nelson     | oneLevelCount -> 0   | 0.9.2342.19200300.100.1.3=hnelson@example.com -> 0  |
|                       | subLevelCount -> 0   | 2.5.4.0=inetorgperson -> 3                          |
|                       | status -> e          | 2.5.4.0=organizationalperson -> 2                   |
|                       |                      | 2.5.4.0=person -> 1                                 |
|                       |                      | 2.5.4.0=top -> 0                                    |
|                       |                      | 2.5.4.3=horatio nelson -> 0                         |
|                       |                      | 2.5.4.35=<bytes> -> 0                               |
|                       |                      | ...                                                 |
-----------------------------------------------------------------------------------------------------
```

Functional Model - Search (3)

- HBase scanner:
  - base -> UUID as scanner start and stop row
  - filter -> converted to scanner filter
  - scope sub -> recursive scans if one-level count > 0
- Better, but still not optimal
  - too large scan range, e.g. for (uid=foobar)

```
-----------------------------------------------------------------------------------------------
| tree                   | treeInfo             | normAttributes                              |
-----------------------------------------------------------------------------------------------
| 22222222-...,2.5.4.3   | id -> 44444444-...   | 0.9.2342.19200300.100.1.1=hnelson -> 0      |
|   =horatio nelson      | oneLevelCount -> 0   | 0.9.2342.19200300.100.1.3=hnelson@example.com -> 0 |
|                        | subLevelCount -> 0   | 2.5.4.0=inetorgperson -> 3                  |
|                        | status -> e          | 2.5.4.0=organizationalperson -> 2           |
|                        |                      | 2.5.4.0=person -> 1                         |
|                        |                      | 2.5.4.0=top -> 0                            |
|                        |                      | 2.5.4.3=horatio nelson -> 0                 |
|                        |                      | 2.5.4.35=<bytes> -> 0                       |
|                        |                      | ...                                         |
-----------------------------------------------------------------------------------------------
```

Apache Directory

Functional Model - Search (4)

- ApacheDS XDBM search engine
  - for B+Tree like backends
  - uses index tables
  - search path optimization, based on candidate count
- Two type of attribute index tables
  - column based
  - row based
- Additional: presence
  - entryUUID as row key
  - check with exists()

```
-----------------------------------
| index_givenname | info          |
-----------------------------------
| *44444444-...   | status -> e |
-----------------------------------
| *77777777-...   | status -> e |
-----------------------------------
```

Apache Directory

Functional Model - Search (5)

- Column based index tables
  - for attributes with (almost) unique values, e.g. uid, mail
  - value as row key, entryUUID as column qualifier
  - eq: just a get to retrieve all entryUUIDs, count is cheap
  - substr: scan, regex filter
    - additional start and stop row for substring initial filters
  - gte: scan, start row
  - lte: scan, stop row

```
----------------------------------------
| index_givenname | info               |
----------------------------------------
| =cornelius      | 77777777-... -> e |
----------------------------------------
| =horatio        | 44444444-... -> e |
|                 | CCCCCCCC-... -> e |
----------------------------------------
| =john           | 99999999-... -> e |
|                 | DDDDDDDD-... -> e |
----------------------------------------
| =william        | AAAAAAAA-... -> e |
----------------------------------------
```

Functional Model - Search (6)

- Row based index tables
  - for often used attribute values, e.g. objectClass:person
  - row key composed of value and entryUUID
    - no fixed length!
  - eq: scan with start and stop row, count more expensive
  - substr: scan, regex filter
    - additional start and stop row for initial filter
  - gte: scan, start row
  - lte: scan, stop row

```
-----------------------------------------------
| index_objectClass                | info      |
-----------------------------------------------
| =inetorgperson<\00>44444444-... | status -> e |
-----------------------------------------------
| =inetorgperson<\00>99999999-... | status -> e |
-----------------------------------------------
| =person<\00>44444444-...        | status -> e |
-----------------------------------------------
| =person<\00>99999999-...        | status -> e |
-----------------------------------------------
```

Apache Directory

Why HBase?

- Java API, Thrift, REST, ...
- Embeddable for integration tests
- Soon in Maven repo
- Map/Reduce Jobs
  - bulk import job: write master table only
  - index job: creates tree and index tables
    - also useful for rebuilding indices
  - backup/restore
  - mass modifications (@sun.com -> @oracle.com)
  - analysis, data migration

Apache Directory

Why HBase?

- Built-in replication
- Scalability
- Strong consistency
- Sparse, everything is a byte[]
- Versions for audit log
- Transactions
- Atomic increment/decrement/checkAndPut
- Scanner with ranges and filter
- Apache License
- Great community

Who not HBase?

- Heavy to setup
- Real tests require real hardware
- Exceptions in logs
- Security
  – no trusted connection between client and server
  – data isn't stored encryted
  – no authn/authz

Apache Directory

Status

- All LDAP operations work
  - add, modify, delete, moddn, search
  - ApacheDS integration test pass
- Performance
  - Great improvement from HBase 0.20.3 -> 0.20.4
  - only tested pseudo-distributed and on VMware
- TODOs
  - make ApacheDS ready for distributed backends
    - event notification from HBase?
  - caching
  - schema and config partition

Demo

- HBase 0.20.4-RC5, standalone
- HBase Explorer 0.2.1
- ApacheDS trunk
  - 3 partitions:
    - o=hbase
    - o=sevenSeas
    - dc=example,dc=com
- Apache Directory Studio Plugin 1.5.3

Resources

- Apache Directory Project
  - http://directory.apache.org/
- Wiki page
  - https://cwiki.apache.org/confluence/display/DIRxSBOX/HBase+Prototype