Apache Con North America 2011

Becoming a content-driven, modular application
A Case Study

Brett Porter, MaestroDev

brett@maestrodev.com

Presented by

The Apache Software Foundation
http://www.apache.org/

Produced by

STONE CIRCLE PRODUCTIONS, LLC

Brett Porter

Brett Porter

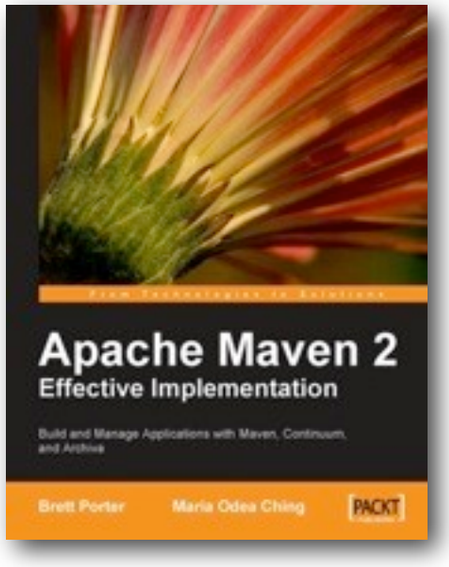
- Apache Software Foundation
  - Apache Maven, Archiva, Continuum, NPanday, Infrastructure, others
  - Member and Director

Brett Porter

- Apache Software Foundation
  - Apache Maven, Archiva, Continuum, NPanday, Infrastructure, others
  - Member and Director
- MaestroDev
  - CTO
  - Directing Maestro development

Brett Porter

- Apache Software Foundation
  - Apache Maven, Archiva, Continuum, NPanday, Infrastructure, others
  - Member and Director
- MaestroDev
  - CTO
  - Directing Maestro development
- Co-author
  - Apache Maven 2: Effective Implementation
  - Better Builds with Maven



Content-related Experience

Content-related Experience

- This page left intentionally blank

A Practical Example

- This is about a change we made in our project
- May apply to you if you have a similar challenge
- Only certain types of applications fit as a content application

A Practical Example

- This is about a change we made in our project
- May apply to you if you have a similar challenge
- Only certain types of applications fit as a content application

- Apache Archiva
  - You can check out the code for yourself, it’s open source
  - http://svn.apache.org/repos/asf/archiva/trunk/

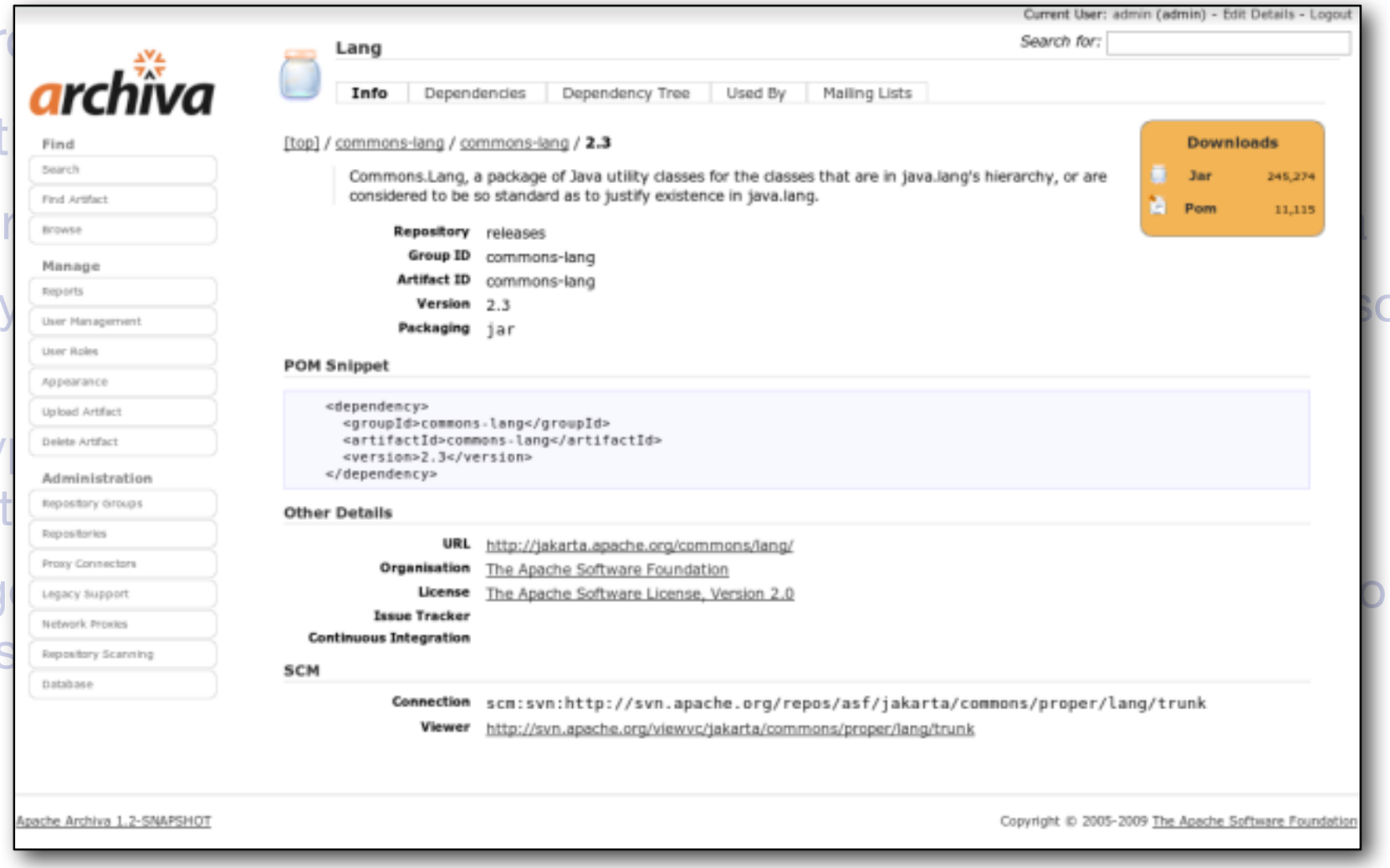
Friday, 11 November 11

Archiva: Some Background

- Repository Manager for Maven (and other similar tools)
- Naturally hierarchical content based on the Maven repository format
- Basically an artifact file server with a custom interface
  - rule-based retrieval and management of artifacts and associated metadata
  - access directly over HTTP and WebDAV, a user-driven web interface, and some web services
  - artifacts are typically binaries, ranging from small to multi-gigabyte, with information attached from the Maven POM or other sources
  - typically a large number of files, and rapid turnover as new are added and older development snapshots are purged

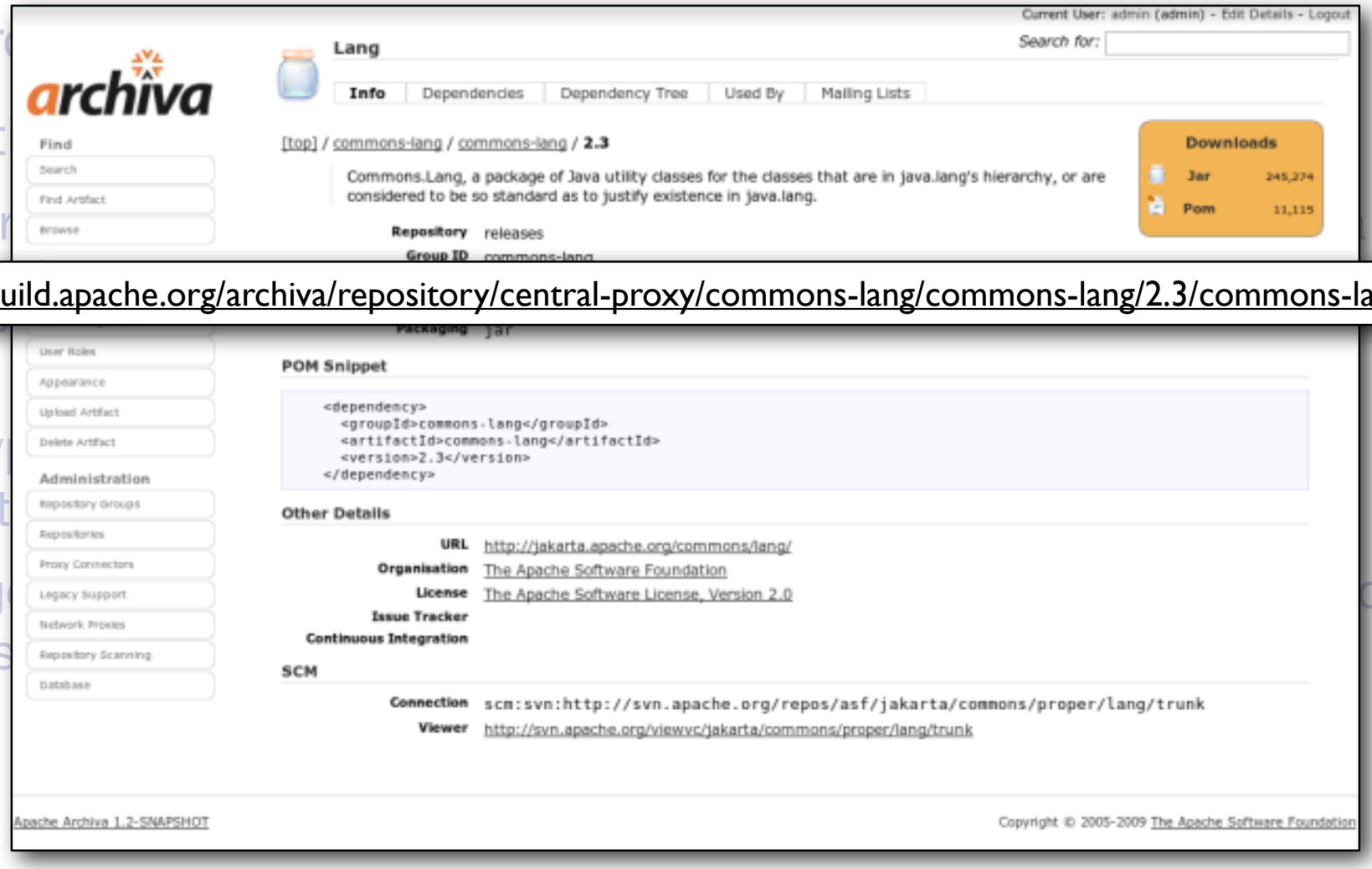
Archiva: Some Background

- Repository Manager for Maven (and other similar tools)
- Naturally hierar
- Basically an art
  - rule-based retr
  - access directly some web services
  - artifacts are ty information att
  - typically a larg older development s



Archiva: Some Background

- Repository Manager for Maven (and other similar tools)
- Naturally hierar
- Basically an art
  - rule-based retr
  - access
    web services
  - artifacts are ty
    information att
  - typically a larg
    development s



http://vmbuild.apache.org/archiva/repository/central-proxy/commons-lang/commons-lang/2.3/commons-lang-2.3.jar

older

Other Repository Managers

Other Repository Managers

- Uses Lucene and flat files for metadata
- Has an established anti-database stance
- Focuses on “self-healing” metadata to ensure integrity

Other Repository Managers

Nexus

- Uses Lucene and flat files for metadata
- Has an established anti-database stance
- Focuses on “self-healing” metadata to ensure integrity

artifactory

- Initially used JCR to store everything, including binary artifact data
- Claimed benefits of integrity
- Had reputation for wedging the database
- Harder to import/export the content
- Now seems to support a filesystem-only repository

Archiva History

- Around in part since March 2005
  - converting Maven 1 to Maven 2 repositories
  - relied heavily on scanning the repository on the filesystem and pulling out metadata
- Grew into a repository manager application as a Maven subproject
  - promoted to top level project Apache Archiva in March 2008
- Architecture was using Lucene as a “database”, but stored everything in its original form

Archiva 1.0 Architecture

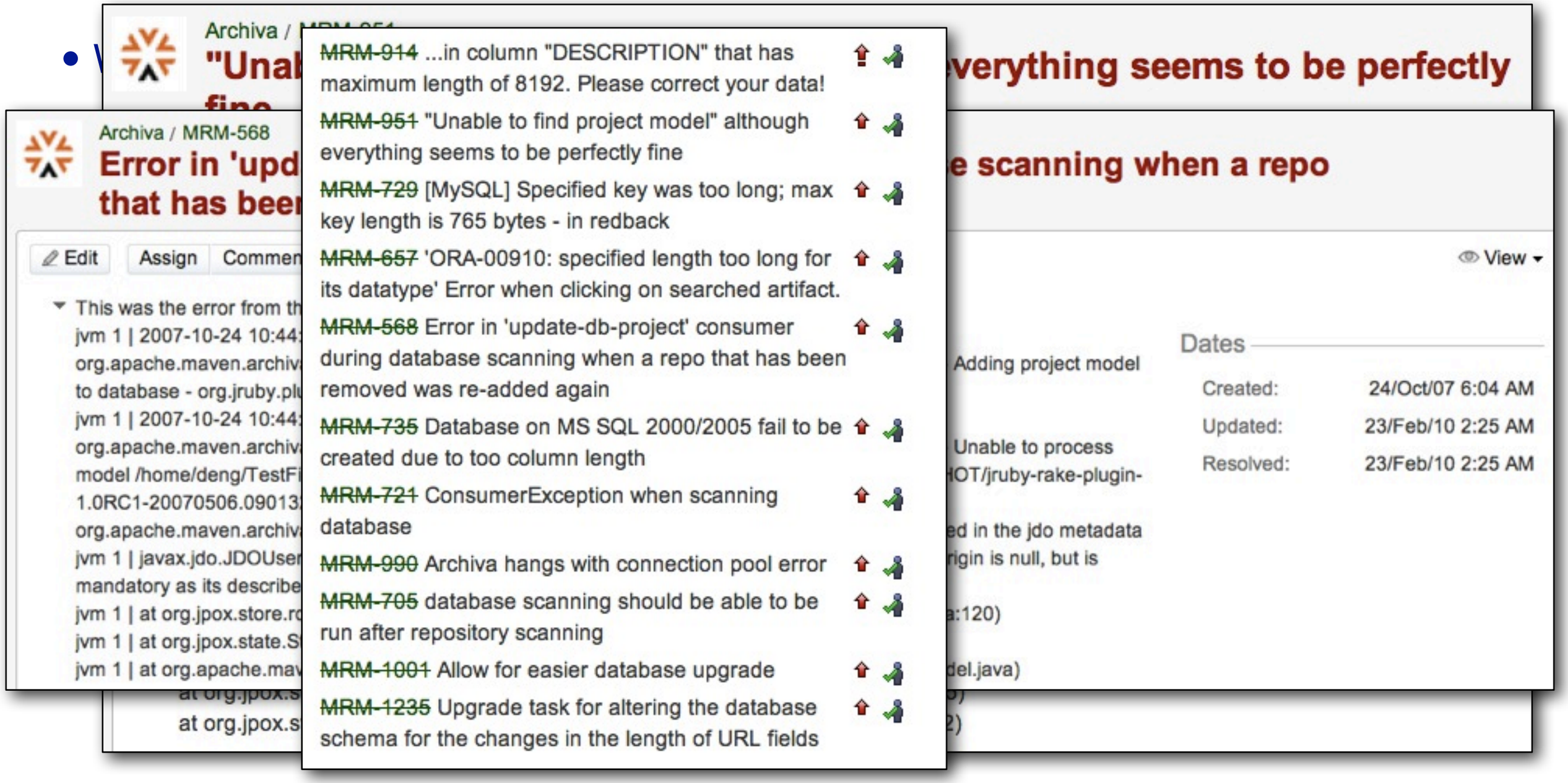
- Leap forward in functionality
- All of storage was re-done, partly using database
  - to be able to query easily and use persistence APIs
  - to do two phase scanning

This Didn’t Work Out So Well

- We used JDO 2 (JPOX) - not a great deal of resources for it
- Two-phase scanning could get out of sync
- Fell into the classic trap - only one person knew how it worked
- A few problems started to crop up
  - database exceptions deep in the stack that were hard to deal with
  - performance concerns
  - memory consumption (particularly with embedded database)
  - lack of extensibility for metadata
  - configuration for initial set up was not necessarily out of the box

This Didn't Work Out So Well

- W



Motivation for Change

- The architecture was holding it back
- Wanted to implement extensible metadata for artifacts

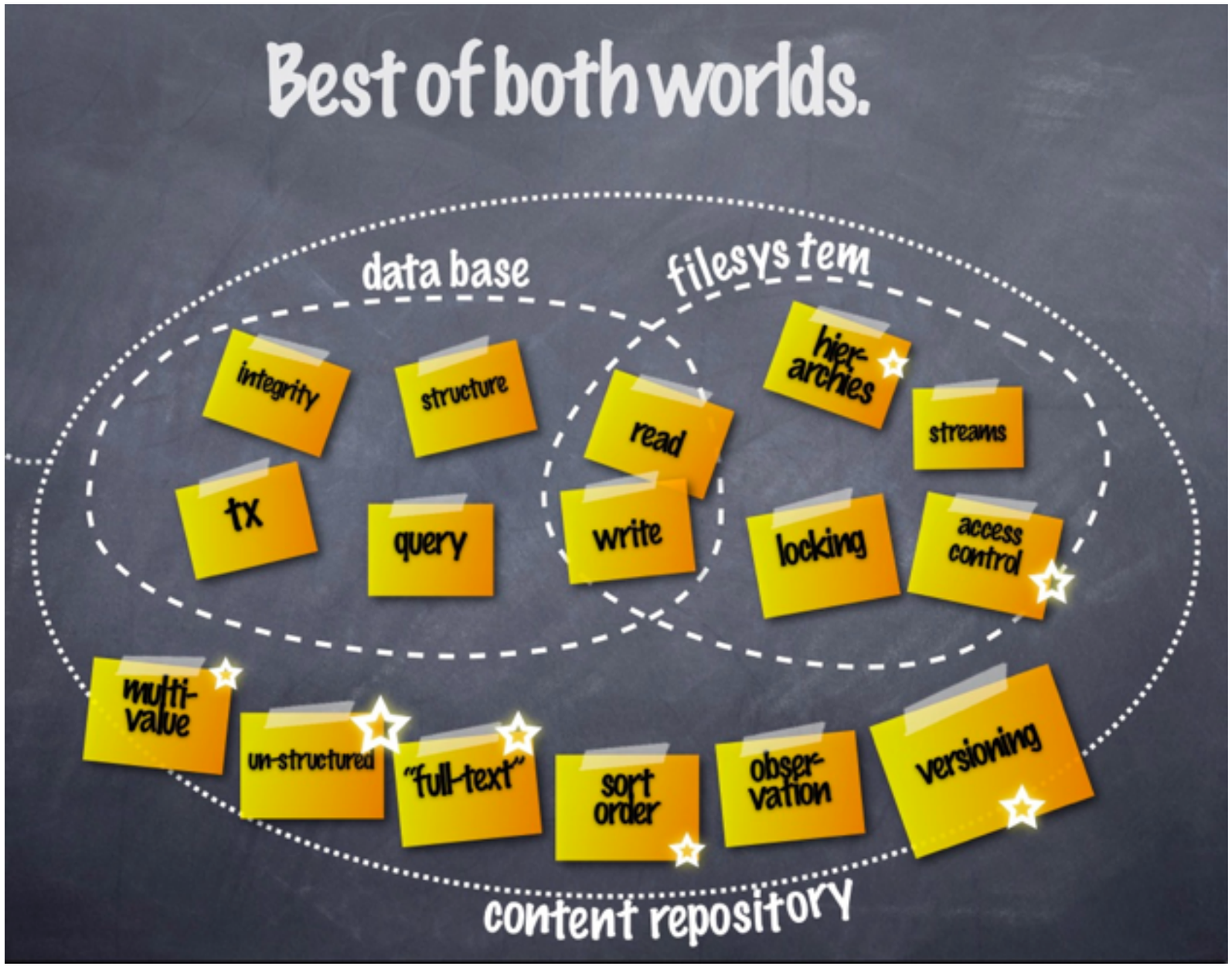
Back to the Future

- Blend up the strengths of the original architecture, the newer feature set, and the added metadata
- Allow use of unstructured data
- Improve or solve the problems we'd seen
- Remove the database altogether
- Separate the metadata from the storage
- Pick up other improvements on the way, like lazy-loading content inside artifacts and proxying remote repositories
- Move toward a defined target architecture, and keep it working along the way
- Make the application modular: reusable and extensible
- Add a plugin architecture

Friday, 11 November 11

Content vs. Database

- http://java.dzone.com/articles/java-content-repository-best



Hierarchical vs. Relational

- Hierarchy familiar for XML, DOM, filesystems
- How much structure is known in advance?
- What type of queries are needed?
  - hierarchy good at locating content, but not based on joined data
- Databases are not as good at transitive retrieval, or navigation / traversal of data

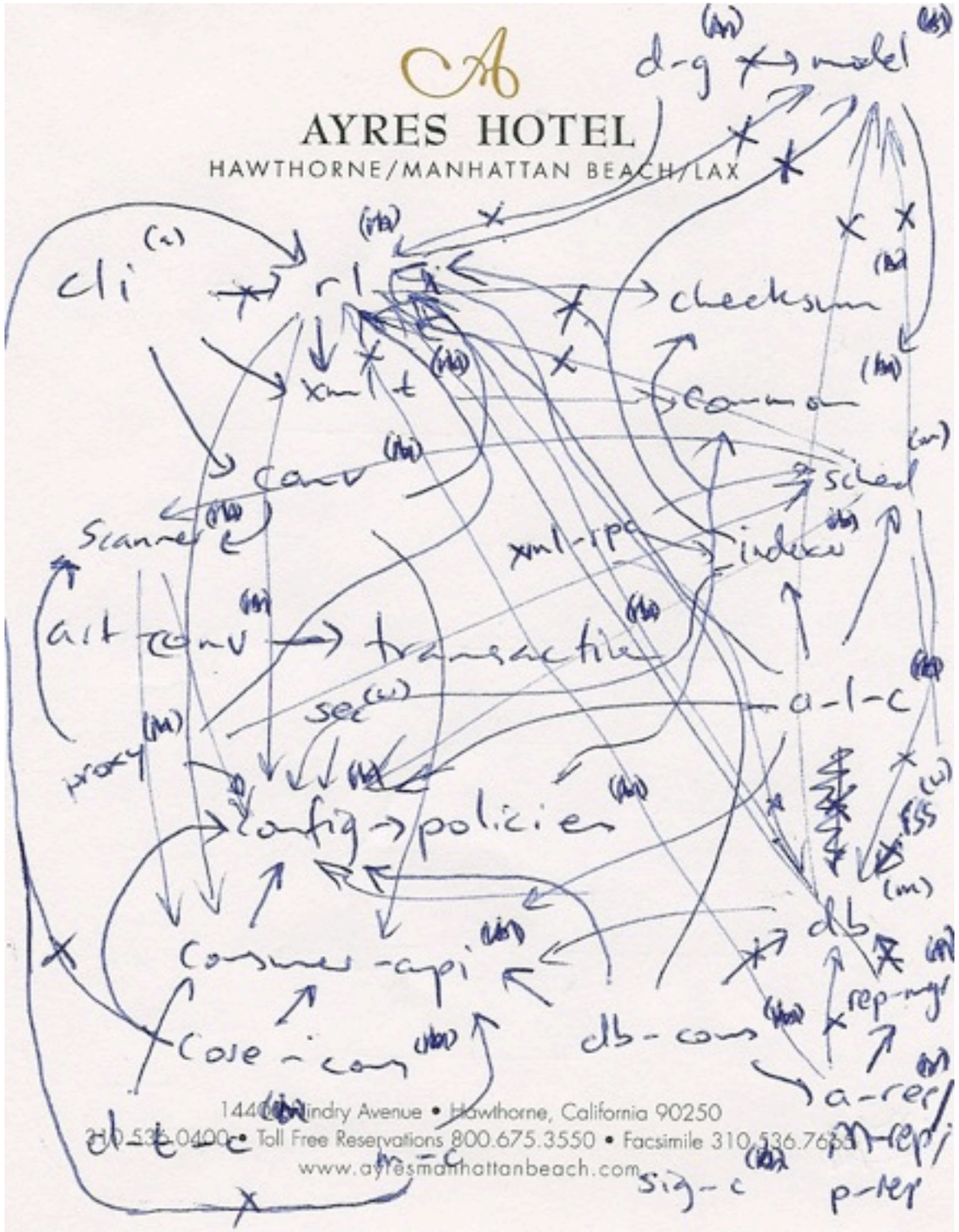
- Archiva is hierarchical
  - filesystem-like structure
  - POM inheritance & dependency relationships

Archiva: First Steps

- Reviewed the architecture

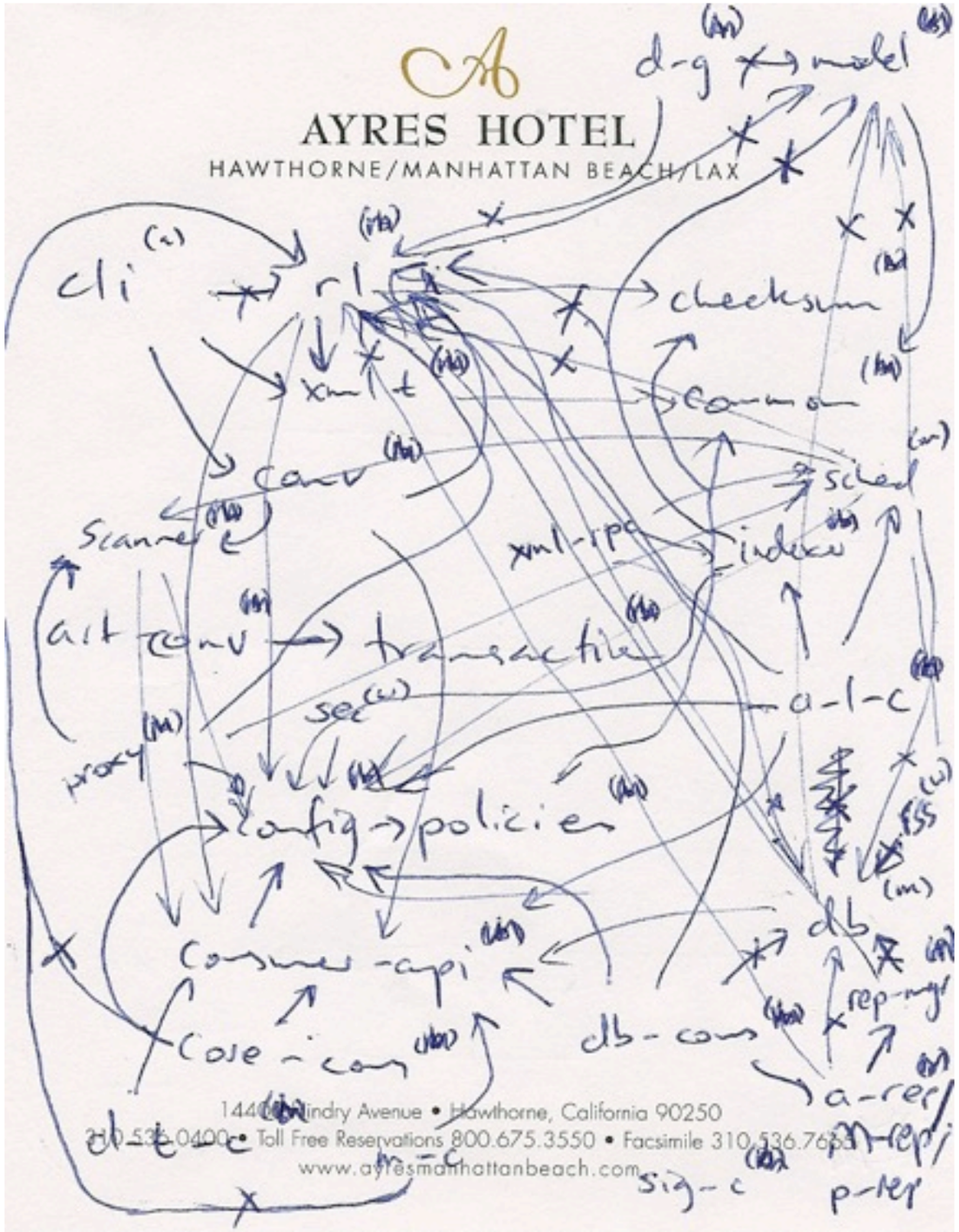
Archiva: First Steps

- Reviewed the architecture



Archiva: First Steps

- Reviewed the architecture

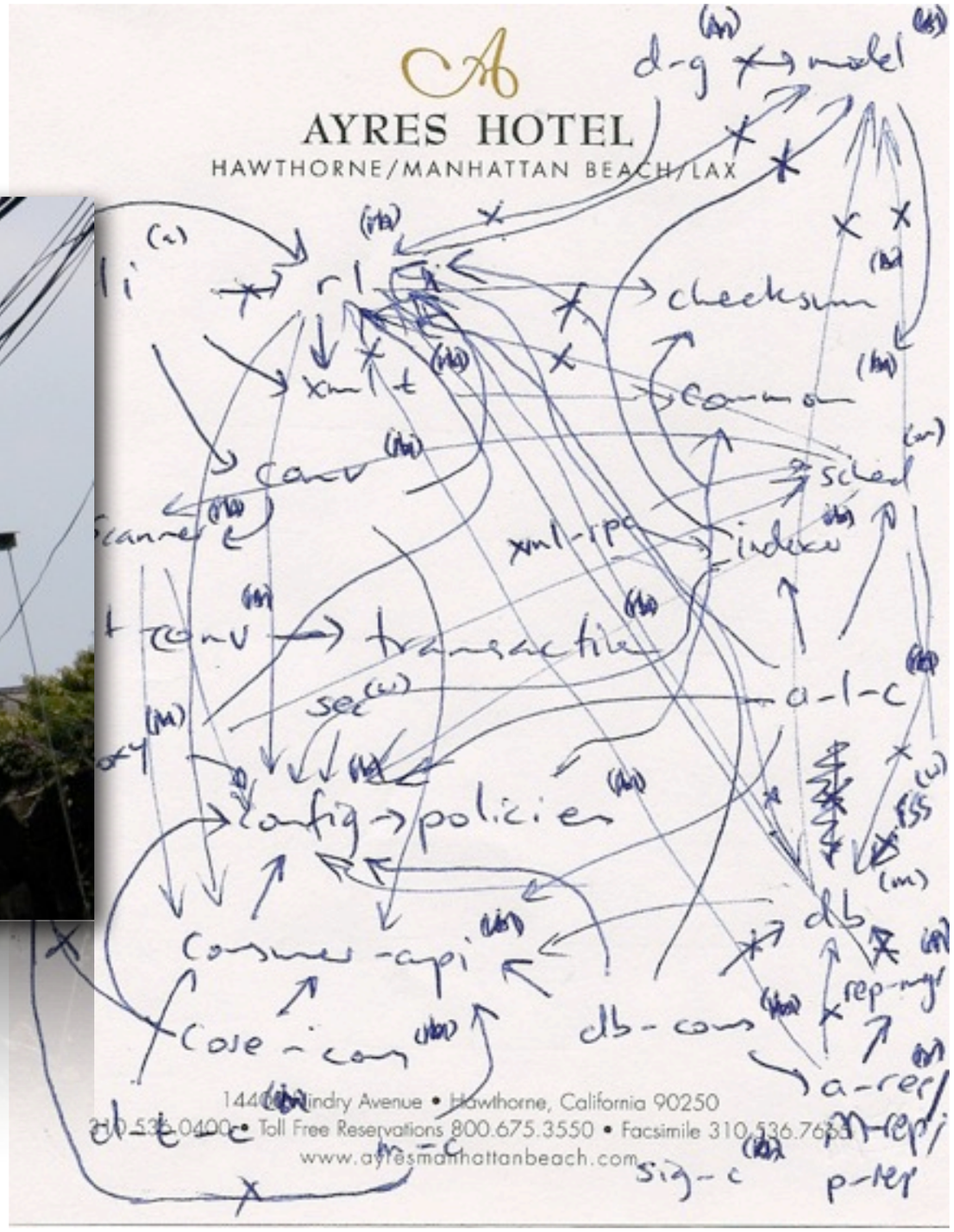


YIKES!

Archiva: First Steps

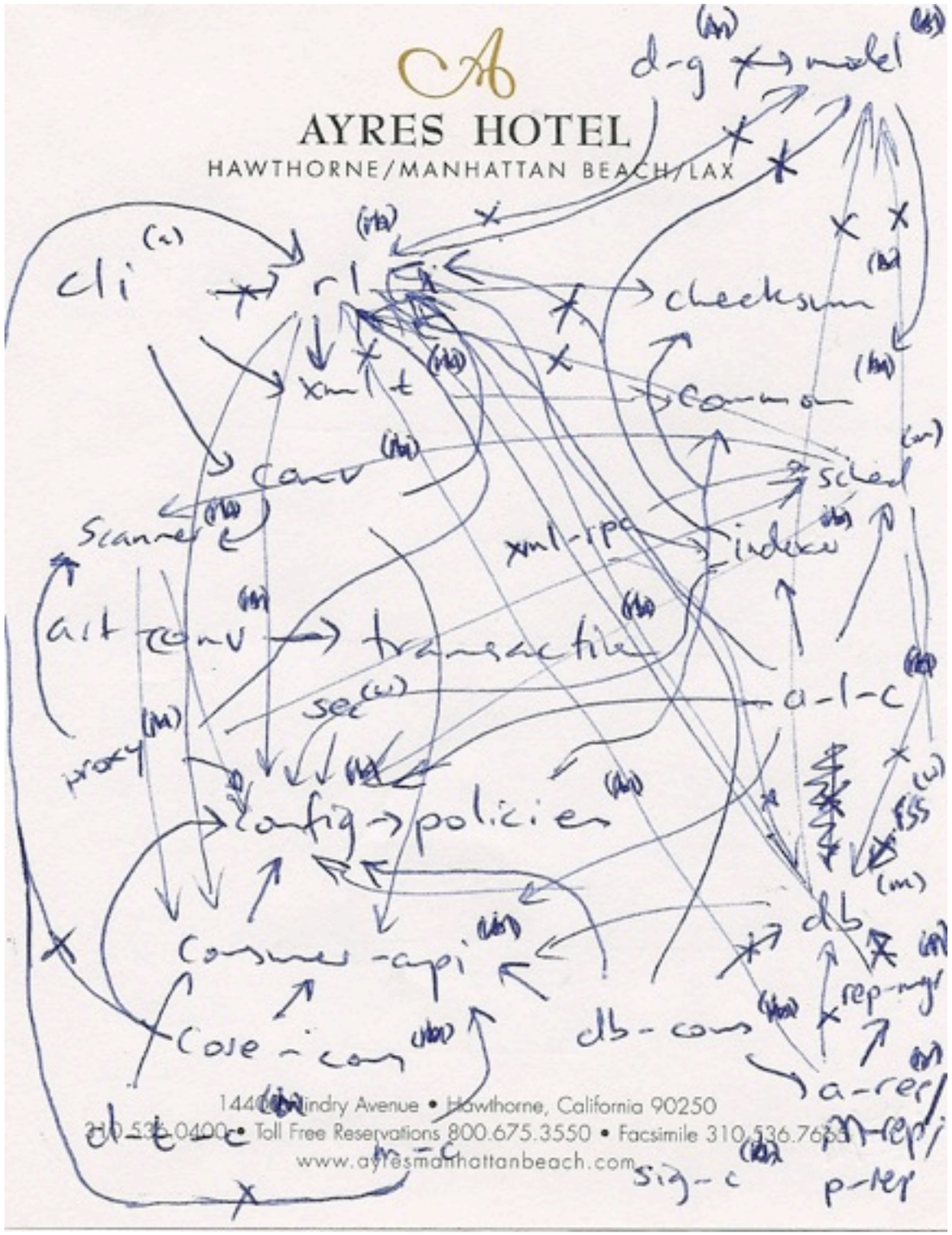
- Reviewed the architecture

YIKES!



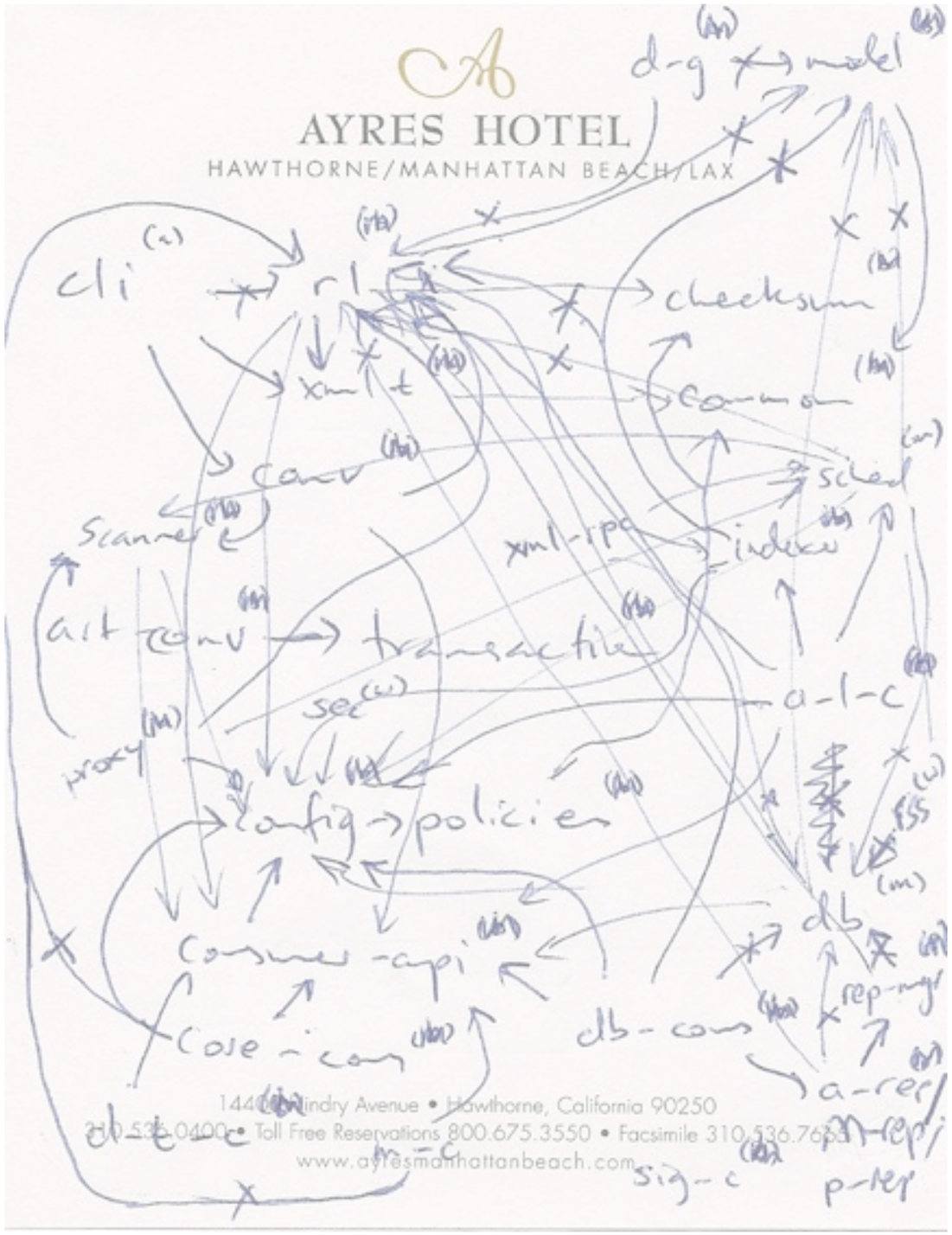
Target Restructuring

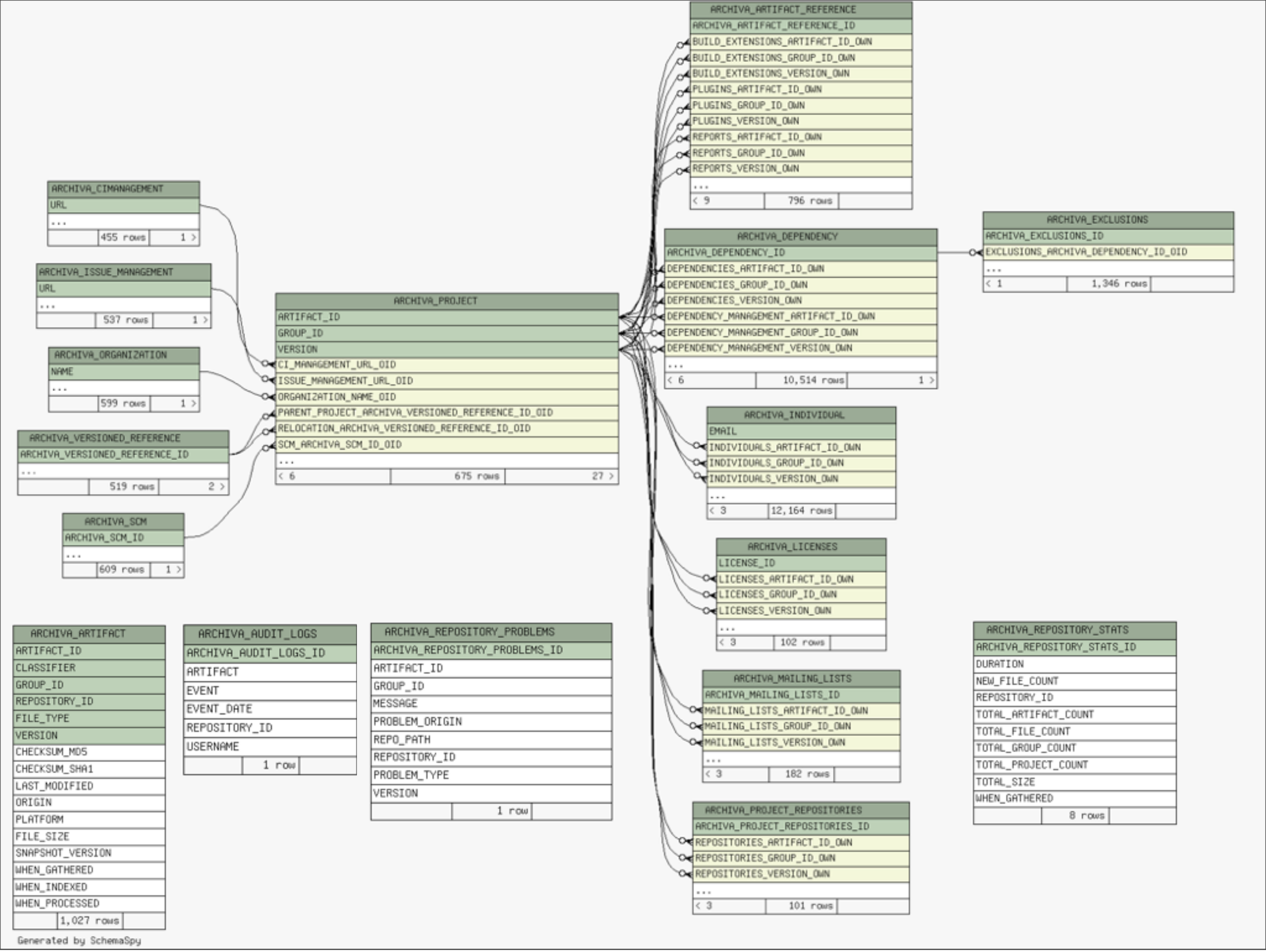
- Technically unrelated to the effort at hand
- Trying to remove the database showed how pervasive it was
  - model (JDO annotated classes) and database module were everywhere
  - repository-layer was the culprit
  - other modules grew up out of what was available
- Started to build the right abstraction
  - directly replace some uses
  - others were redirected via the old code



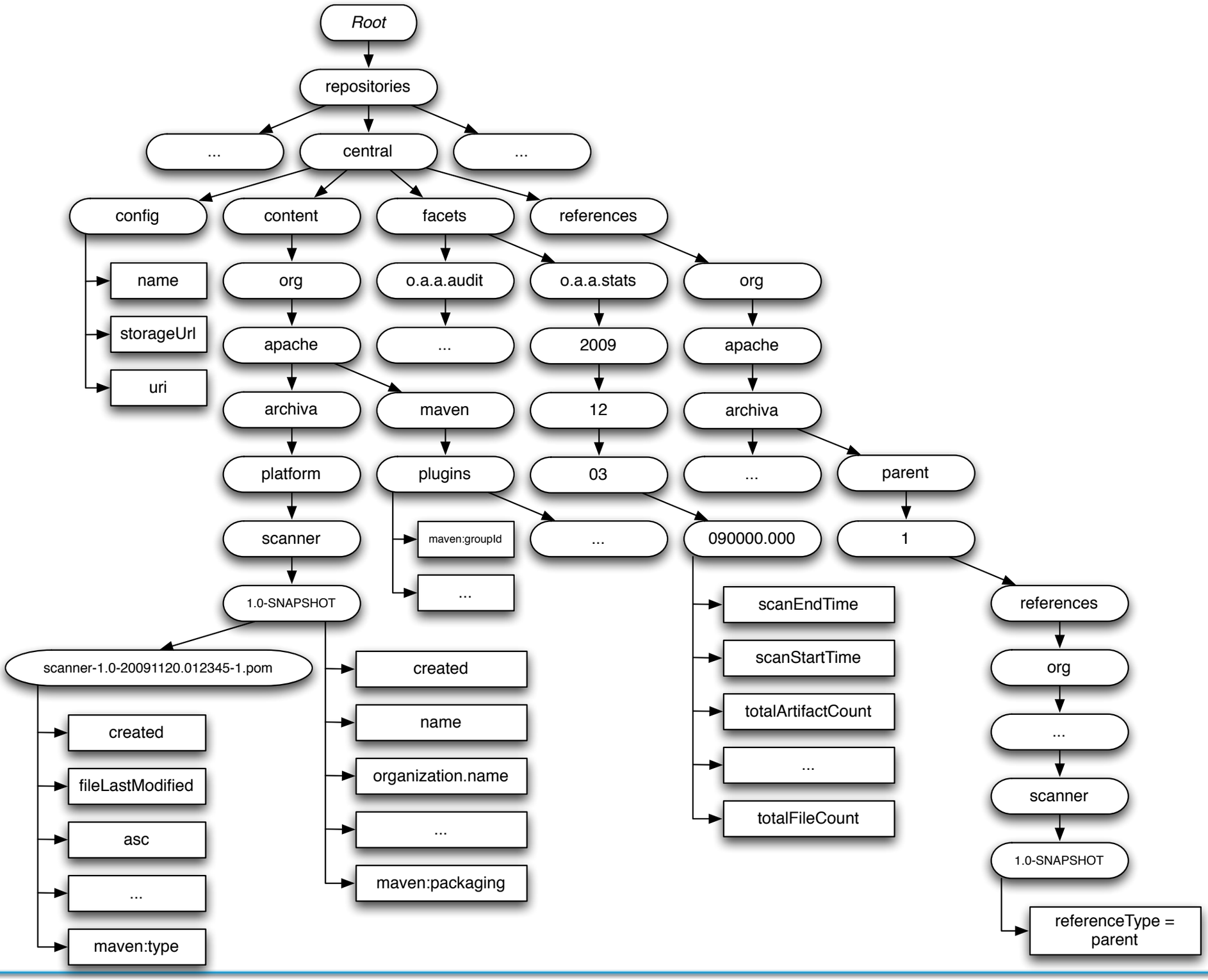
Target Restructuring

- Technically unrelated to the effort at hand
- Trying to remove the database showed how pervasive it was
  - model (JDO annotated classes) and database module were everywhere
  - repository-layer was the culprit
  - other modules grew up out of what was available
- Started to build the right abstraction
  - directly replace some uses
  - others were redirected via the old code




ARCHIVA_ARTIFACT_REFERENCE
ARCHIVA_ARTIFACT_REFERENCE_ID
BUILD_EXTENSIONS_ARTIFACT_ID_OWN
BUILD_EXTENSIONS_GROUP_ID_OWN
BUILD_EXTENSIONS_VERSION_OWN
PLUGINS_ARTIFACT_ID_OWN
PLUGINS_GROUP_ID_OWN
PLUGINS_VERSION_OWN
REPORTS_ARTIFACT_ID_OWN
REPORTS_GROUP_ID_OWN
REPORTS_VERSION_OWN
...
< 9
796 rows
ARCHIVA_CIMANAGEMENT
URL
...
455 rows
1 >
ARCHIVA_ISSUE_MANAGEMENT
URL
...
537 rows
1 >
ARCHIVA_ORGANIZATION
NAME
...
599 rows
1 >
ARCHIVA_VERSIONED_REFERENCE
ARCHIVA_VERSIONED_REFERENCE_ID
...
519 rows
2 >
ARCHIVA_SCM
ARCHIVA_SCM_ID
...
609 rows
1 >
ARCHIVA_PROJECT
ARTIFACT_ID
GROUP_ID
VERSION
CI_MANAGEMENT_URL_OID
ISSUE_MANAGEMENT_URL_OID
ORGANIZATION_NAME_OID
PARENT_PROJECT_ARCHIVA_VERSIONED_REFERENCE_ID_OID
RELOCATION_ARCHIVA_VERSIONED_REFERENCE_ID_OID
SCM_ARCHIVA_SCM_ID_OID
...
< 6
675 rows
27 >
ARCHIVA_DEPENDENCY
ARCHIVA_DEPENDENCY_ID
DEPENDENCIES_ARTIFACT_ID_OWN
DEPENDENCIES_GROUP_ID_OWN
DEPENDENCIES_VERSION_OWN
DEPENDENCY_MANAGEMENT_ARTIFACT_ID_OWN
DEPENDENCY_MANAGEMENT_GROUP_ID_OWN
DEPENDENCY_MANAGEMENT_VERSION_OWN
...
< 6
10,514 rows
1 >
ARCHIVA_EXCLUSIONS
ARCHIVA_EXCLUSIONS_ID
EXCLUSIONS_ARCHIVA_DEPENDENCY_ID_OID
...
< 1
1,346 rows
ARCHIVA_INDIVIDUAL
EMAIL
INDIVIDUALS_ARTIFACT_ID_OWN
INDIVIDUALS_GROUP_ID_OWN
INDIVIDUALS_VERSION_OWN
...
< 3
12,164 rows
ARCHIVA_LICENSES
LICENSE_ID
LICENSES_ARTIFACT_ID_OWN
LICENSES_GROUP_ID_OWN
LICENSES_VERSION_OWN
...
< 3
102 rows
ARCHIVA_MAILING_LISTS
ARCHIVA_MAILING_LISTS_ID
MAILING_LISTS_ARTIFACT_ID_OWN
MAILING_LISTS_GROUP_ID_OWN
MAILING_LISTS_VERSION_OWN
...
< 3
182 rows
ARCHIVA_PROJECT_REPOSITORIES
ARCHIVA_PROJECT_REPOSITORIES_ID
REPOSITORIES_ARTIFACT_ID_OWN
REPOSITORIES_GROUP_ID_OWN
REPOSITORIES_VERSION_OWN
...
< 3
101 rows
ARCHIVA_ARTIFACT
ARTIFACT_ID
CLASSIFIER
GROUP_ID
REPOSITORY_ID
FILE_TYPE
VERSION
CHECKSUM_MD5
CHECKSUM_SHA1
LAST_MODIFIED
ORIGIN
PLATFORM
FILE_SIZE
SNAPSHOT_VERSION
WHEN_GATHERED
WHEN_INDEXED
WHEN_PROCESSED
1,027 rows
ARCHIVA_AUDIT_LOGS
ARCHIVA_AUDIT_LOGS_ID
ARTIFACT
EVENT
EVENT_DATE
REPOSITORY_ID
USERNAME
1 row
ARCHIVA_REPOSITORY_PROBLEMS
ARCHIVA_REPOSITORY_PROBLEMS_ID
ARTIFACT_ID
GROUP_ID
MESSAGE
PROBLEM_ORIGIN
REPO_PATH
REPOSITORY_ID
PROBLEM_TYPE
VERSION
1 row
ARCHIVA_REPOSITORY_STATS
ARCHIVA_REPOSITORY_STATS_ID
DURATION
NEW_FILE_COUNT
REPOSITORY_ID
TOTAL_ARTIFACT_COUNT
TOTAL_FILE_COUNT
TOTAL_GROUP_COUNT
TOTAL_PROJECT_COUNT
TOTAL_SIZE
WHEN_GATHERED
8 rows
Generated by SchemaSpy

Design a Content Model

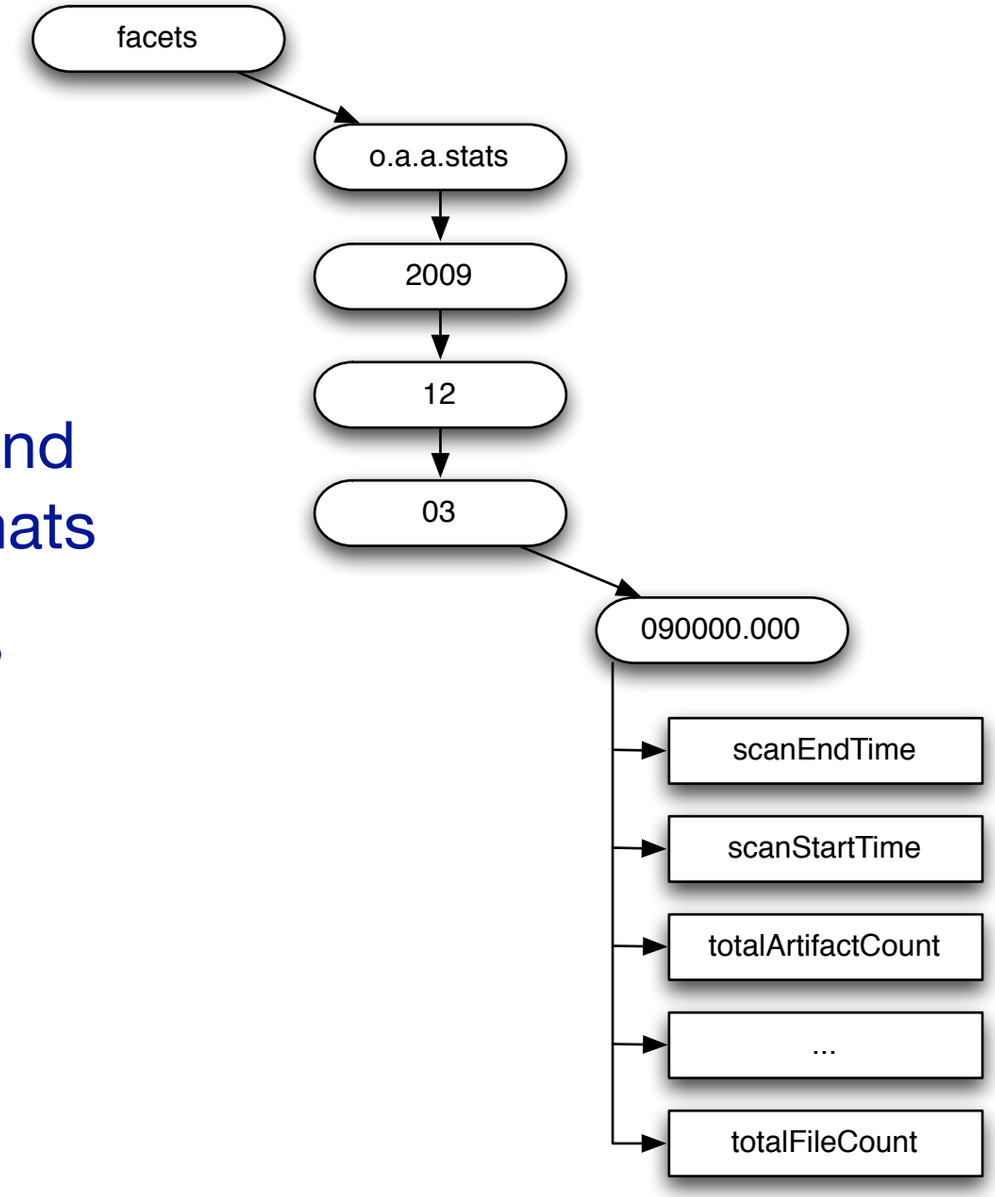


Hierarchy

- Natural benefits for us due to the repository structure
- Can traverse to a part of a group without dealing with substrings
- Layout not identical to a Maven repository, and will be translated from a variety of input formats
- Unstructured data lends itself well to plugins and arbitrary metadata

Hierarchy

- Natural benefits for us due to the repository structure
- Can traverse to a part of a group without dealing with substrings
- Layout not identical to a Maven repository, and will be translated from a variety of input formats
- Unstructured data lends itself well to plugins and arbitrary metadata



David’s Model

- Helpful reference
  - http://wiki.apache.org/jackrabbit/DavidsModel
- Rule #1: Data First, Structure Later. Maybe.
- Rule #2: Drive the content hierarchy, don't let it happen.
- Rule #3: Workspaces are for clone(), merge() and update().
- Rule #4: Beware of Same Name Siblings.
- Rule #5: References considered harmful.
- Rule #6: Files are Files are Files.
- Rule #7: ID's are evil.

Faceted Metadata

- Designed faceted metadata model and corresponding API
- Assuming a hierarchical content model, though not yet using JCR
- For most compatibility with existing code, fully mapped object model

Repository API

- Content access mechanism
- Individual coordinate paths passed in (group, artifact, version)
  - no need to construct the strings, avoids layout being spread out
- Resolvers to fill in metadata or obtain artifacts
  - layered access
  - track completeness
  - proxying remotely
- Resolver is a generally useful pattern if you need to load the data from an external source on the fly
- Metadata vs. Storage

Metadata Persistence

- To get it working, a simple hand-rolled file-based implementation
- With everything working, now saw what we could achieve with JCR

JCR - Java Content Repository

- Using Jackrabbit
- Very simple translation to the JCR API
- Initial memory usage is much higher
- Performance was still better than others
- Switched to file-based persistence of the content repository
  - <PersistenceManager class = "org.apache.jackrabbit.core.persistence.bundle.BundleFsPersistenceManager"/>
  - Yet to be proven at scale in Archiva, potentially not as performant
- Not using OCM (Object Content Mapping - similar to ORM but for content repositories)

Filling Metadata

```
ProjectVersionMetadata versionMetadata =
  new ProjectVersionMetadata();

try
{
  Node root = session.getRootNode();

  Node node = root.getNode(
    "repositories/" + repositoryId + "/content/" +
    namespace + "/" + projectId + "/" + projectVersion );

  versionMetadata.setId( projectVersion );
  versionMetadata.setName(
    node.hasProperty( "name" ) ?
      node.getProperty( “name” ) :
      null );
...
```

Adding Metadata

```
try
{
  Node root = session.getRootNode();

  Node node = root.getNode(
    "repositories/" + repositoryId + "/content/" +
    namespace + "/" + projectId );

  Node versionNode = node.addNode(
    versionMetadata.getId() );

  versionNode.setProperty( "name",
    versionMetadata.getName() );
  versionNode.setProperty( "description",
    versionMetadata.getDescription() );
...
```

Querying

```
SELECT * FROM [archiva:artifact] AS artifact WHERE
ISDESCENDANTNODE(artifact,'/repositories/central/content/’)
AND ([sha1] = $checksum OR [md5] = $checksum)
```

```
Query query =
  session.getWorkspace().getQueryManager().createQuery( q,
    Query.JCR_SQL2 );

ValueFactory valueFactory = session.getValueFactory();
query.bindValue( "checksum", valueFactory.createValue( checksum ) );

QueryResult result = query.execute();
artifacts = new ArrayList<ArtifactMetadata>();
for ( Node n : JcrUtils.getNodes( result ) ) {
  artifacts.add( getArtifactFromNode( repositoryId, n ) );
}
```

What about Modularisation?

- Modularisation was key to our changes
- Can be done no matter what technologies you use
- Planning for OSGi, but other priorities have come up on the way

Where the Changes Helped

- Scanning vs. On-demand
- Dependency structure more performant and reliable
- Database was removed
  - whole class of exceptions just disappeared
  - previously unreliable operations like reverse dependency tree fixed
  - memory usage reduced
  - configuration simplified
- Metadata is more extensible
  - generic metadata plugin
  - new plugins contribute metadata without changing code or schema

Challenges

- Initially query by artifact properties
  - e.g. how to find an artifact with a given checksum
  - starting to use JCR 2.0 queries which are being much more effective
- Correctly configuring Jackrabbit

Opportunities

- Exposing JCR API directly to Archiva plugins
- Integration of existing WebDAV access
- Security access directly integrated into JCR
- JCR event model
- JCR version control
- OSGi
- Sling
- General design - lots more to do!

Tips

- Review whether data is hierarchical or structure derived from the data
- Centralise access, but don't overdo the abstraction
- Align content model to natural usage
  - try not to deal with constructing and parsing paths
  - deal with content directly rather than translating to objects
- Huge value in having automated unit tests to keep it working as you make significant changes

Archiva: Help Wanted

- Looking for developers to get involved
- dev@archiva.apache.org

Thanks!

- http://archiva.apache.org/
- http://archiva.apache.org/ref/1.4-M1/
- http://wiki.apache.org/jackrabbit/DavidsModel
- http://www.scribd.com/doc/11163161/JCR-or-RDBMS-why-when-how
- http://wiki.apache.org/jackrabbit/JcrLinks
- http://jackrabbit.apache.org/
- http://maven.apache.org/