The pain of versions

The
Subversion Repository Search Engine
(SupoSE)

Web Site:
www.soebes.com
Blog:
blog.soebes.com
Email:
info@soebes.com

Dipl.Ing.(FH) Karl Heinz Marbaise

Agenda

1. The Fundamental Idea

- We would like to search for different items within Subversion repositories.
  - Why and How?

1. The Fundamental Idea

- We don't know the particular revision number
- We don't know the range of time
- We don't know which file etc.
- We don't know in which file in which revision etc.
- ...

1. The Fundamental Idea

- But what we know…
  - It must be in the Repository

    Somewhere ;-)

1. SupoSE was Born...

- The Subversion Repository Search Engine....
  - SupoSE for short....

2. The Requirements

- In which Revision the Ticket #76 has been solved ?
  - You have to search within the log messages of all revisions.

Note: This only works if you put in the needed information into the log message.

2. The Requirements

- Which Tags or Branches did or do exist within the current project?
  - Search for directories in all folders and revisions.
    - This needed to find deleted folders (e.g. Tags or Branches) as well.

2. The Requirements

- In which documents did we used the term(s) “…” ?
  - Search within the contents of the versioned items in all revisions and all folders (branches/tags/trunk).

2. The Requirements

- In which file did we used the method “executeTestXYZ” ?
  - Search within the contents based on context sensitive informations (parsed files of particular type).
  - For example Java, Perl, Python, Ruby ….files.

2. The Requirements

- Where do we used the property name „xyz...“?
  - Search for property names

- Which files/revisions etc. do have the property “xyz...” with the particular value “content”?
  - Search for particular property values

2. The Requirements

- The search process shouldn't be limited to a single Repository.
  - In usual industrial setup's you will find multiple Subversion Repositories.

3. Ideas

- If we would scan the whole Repository every time we do a query it would be:
  - to slow....
  - it will produce a high load on the repository server.

So this is no option.

3. Ideas

- We have basicly two phases:
  - Initial Phase
    - Reading the content from the Repository and indexing it.
  - Update Phase
    - Read the changed/added contents of the Repository and indexing it.

3. Ideas

- We need to do a full-text search:
  - Many search engines working this way.
  - e.g. The Eclipse Search works the same way...
  - And many others too...

3. Ideas

- How could we update the index?
  - Using Hook scripts to update the indexed informations
    - Pro:
      - Only if something changes

3. Ideas

- How could we update the index?
  - Using Hook scripts to update the indexed informations
    - Con:
      - Slow down commit performance
      - Need to change the Repositories
      - May be we don't have access to repository server.

3. Ideas

- Indexing the Repositories based on the existing access permission of SVN users.
    - Pro:
        - No need to change the repositories.
    - Con:
        - Not everything can be indexed.
        - Performance

3. Ideas

- Scan the repositories based on file://// access.
    - Pro:
        - Very fast
        - No need for authorization
            - We can scan everything

3. Ideas

- Scan the repositories based on file://// access.
    - Con:
        - Installation on the SVN Repository server
        - Load of the SVN Server (peek load for the initial phases).

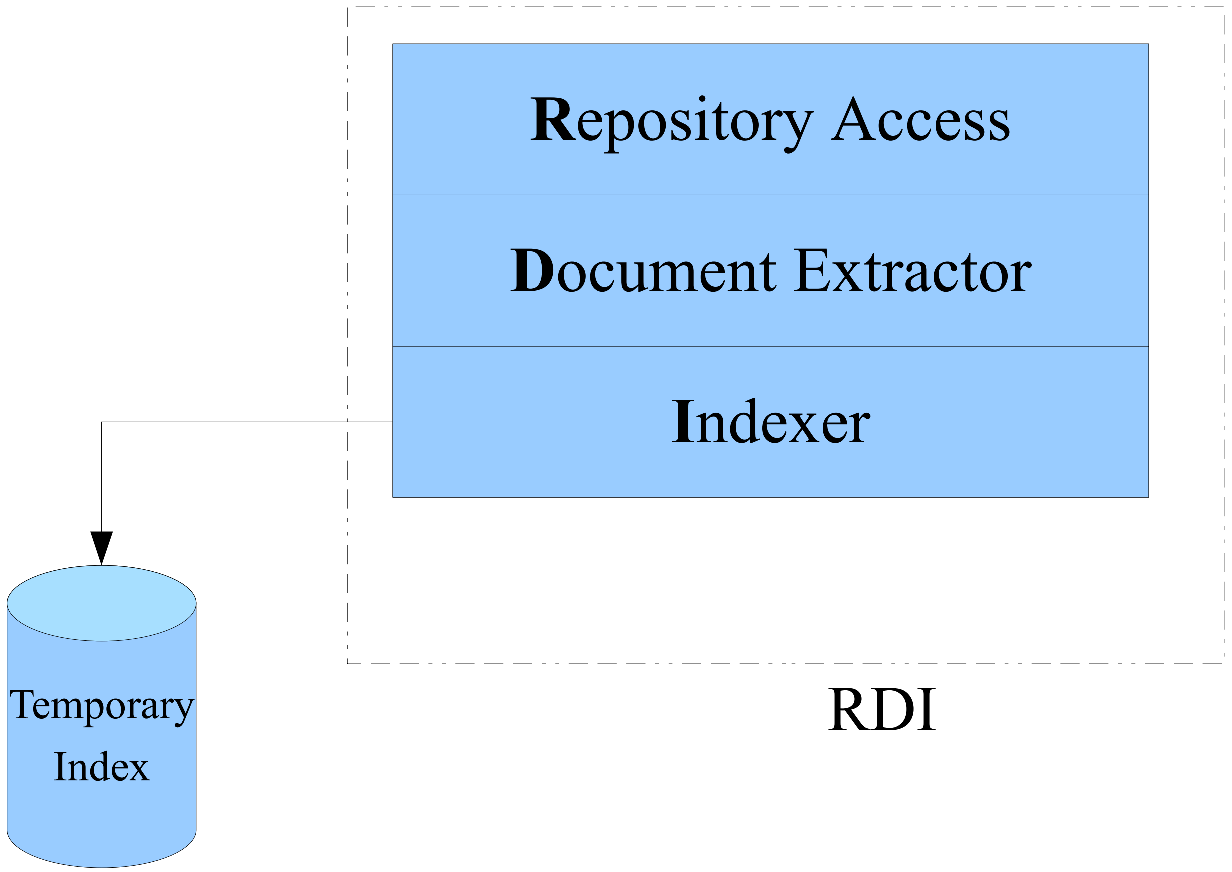
4. Basic Concepts

- Scan the repositories and indexing the information we need.
  - Use the file:/// protocol to access the Repository as preferable method.
  - Use other protocols (http, https or svn) if needed.

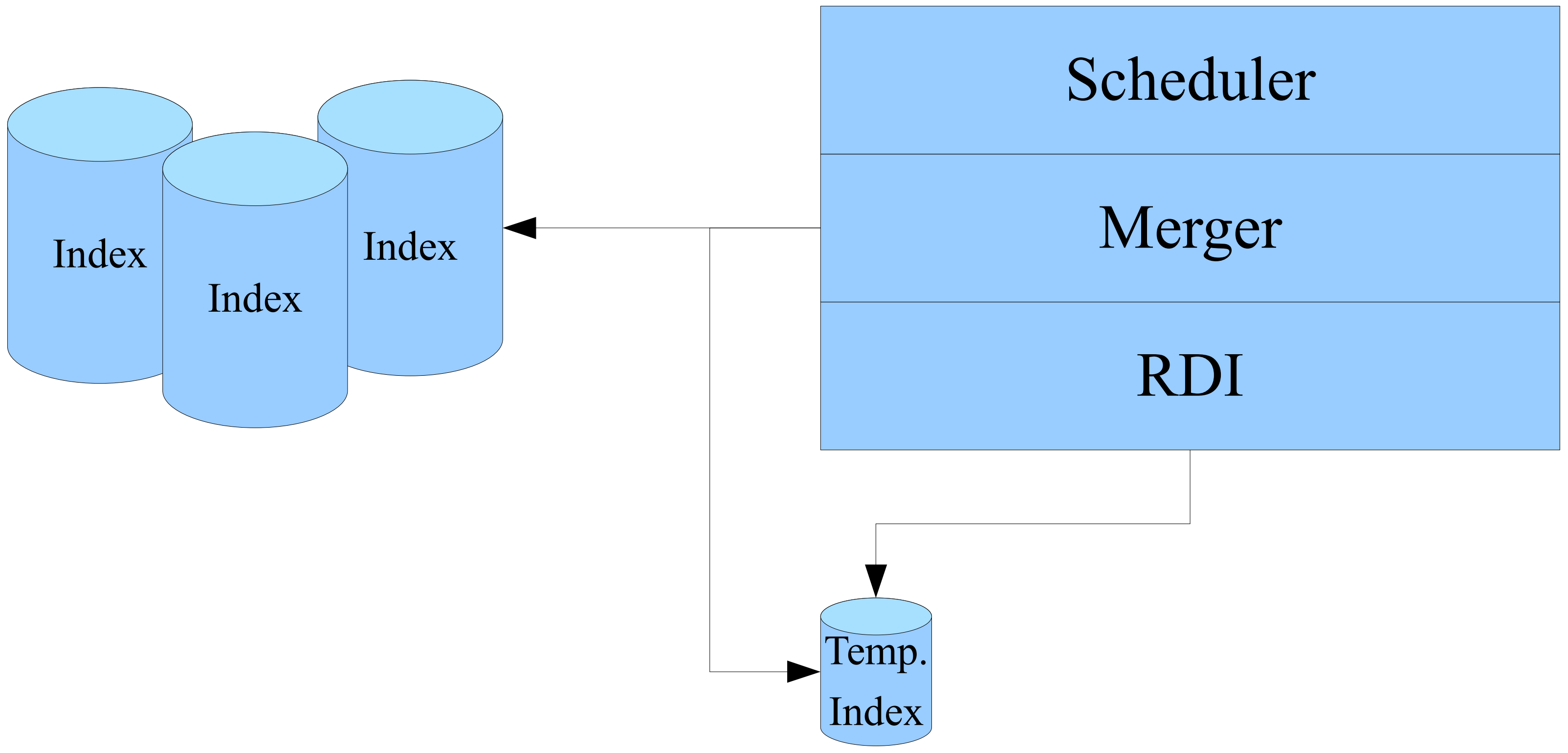
4. Basic Concepts

- Scan on a scheduled base for example daily or hourly etc.
- Should be made configurable.

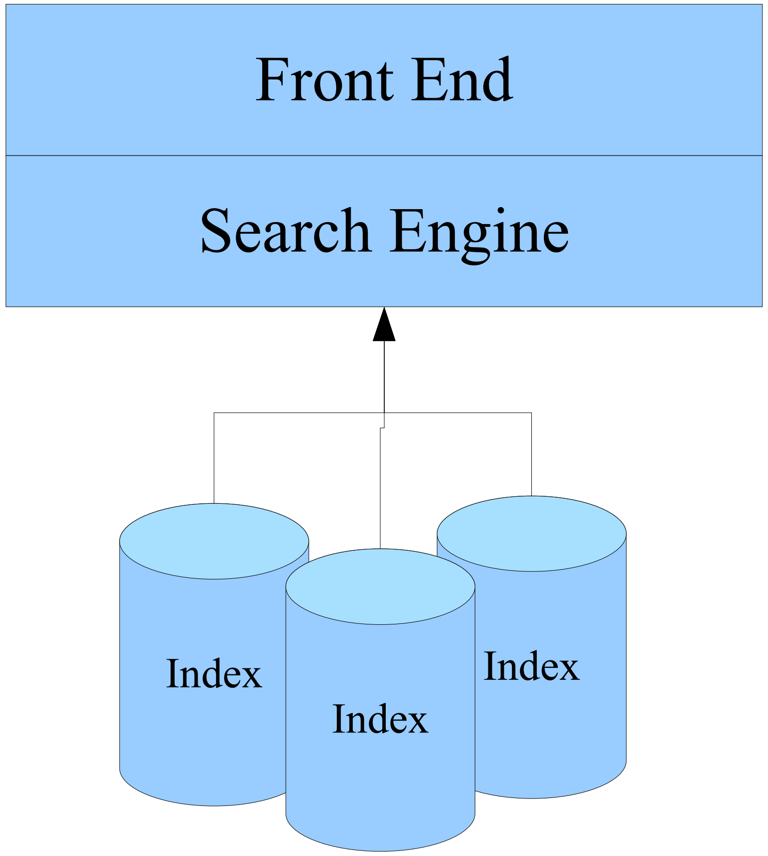
5. Basic Architecture



5. Basic Architecture



5. Basic Architecture



6. The components

- Accessing the Subversion Repository via Java only
  - SVNKit
- Full Text searching capabilities
  - Apache Lucene
- Scheduled running of Jobs
  - Quartz Framework

6. The components

- Access and extract information from Office files etc.
  - POI Framework
  - currently working on PoC for changing to Apache Tika Framework.

6. The components

- Parsing of different Languages (like Java)
  - ANTLR 3.0

7. Open Questions

- Security for the indexed results
  - Authorization of the Search Engine
- What about restrictions for the search results ?
- What about property changes?
  - How do we get informed about them? Hook Scripts ?

7. Open Questions

- What if a repository has path-based authorization and what will happen if this has been changed?
  - What about the already indexed informations?
  - What about the search result?

8. Roadmap

- Bug Fixing ...
- Bug Fixing ...
- ....

8. Roadmap

- Proof of Concept (PoC)
  - Integration of the Tika Framework for extracting the information from different file types (Office and friends etc.)
- Improve/simplify search for daily usage
- Improve/simplify configuration of the indexing processes

8. Roadmap

- GUI
  - May be Web based or Swing or...
- May be PlugIn's
  - trac, Eclipse, Redmine etc.
- Enhance documentation (DocBook Maven?)
- Enhance Command line interface
  - Better output etc. (e.g. sorting)

8. Roadmap

- Interface to make the connection of other applications possible
  - SOAP/RPC ?
  - others ?
- Make the scheduled part runnable in JBoss/Jonas etc. ?
- Performance ?
- Clustering?

9. Current State

- Currently Command Line Based only.
- Indexing of single or multiple (scheduled) repositories working
  - Results can be stored into different destination indexes, can be configured but there seemed to be Bugs in there ;-(
- Searching currently only via command line or via Luke (Swing)

A. Examples

- Scanning of a single Repository

```
supose
    scan
    --url file:///path/to/repos
    --index index.Repos
```

A. Examples

- Scanning of a single Repository

  supose
      scheduled
      --config ...(CHECK THIS)

A. Examples

- Which tags existing in SupoSE Repository?

```
supose
    search
    --index index.Supose
    --query "+path:/tags/*"
```

A. Examples

- The output of the query before:

```
 1. R:14 F:/tags/RELEASE-0.1.0 K:A
 2. R:29 F:/tags/RELEASE-0.2.0 K:A
 3. R:48 F:/tags/RELEASE-0.2.0.1 K:A
 4. R:70 F:/tags/R_0.3.0.0RC1 K:A
 5. R:76 F:/tags/R_0.3.0.0RC2 K:A
 6. R:91 F:/tags/0.4.0.0RC1 K:A
 7. R:93 F:/tags/R_0.4.0.0RC1 K:A
 8. R:111 F:/tags/R_0.4.0RC2 K:A
 9. R:112 F:/tags/R_0.4.0.0RC2 K:A
10. R:115 F:/tags/R_0.4.0.0RC2 K:A
11. R:92 F:/tags/0.4.0.0RC1 K:D
12. R:112 F:/tags/R_0.4.0RC2 K:D
13. R:114 F:/tags/R_0.4.0.0RC2 K:D
```

A. Examples

- Do exist Word files in this repository?

```
supose
    search
    --index index.Supose
    --query "+filename:/*.doc"
```

A. Examples

- What is part of revision 100 of the particular repository?

```
supose
    search
    --index index.Supose
    --query "+revision:100"
```

B. Performance

- Currently the scan of the SupoSE repository itself (with 112 Revisions) via http:// (Internet)
  - This has taken ca. 25 Minutes ;-(

- A scan of Repository (2.8 GiBi) with 12168 Revisions via file:/// protocol took ca. 37 minutes.

On-line Sources I

- [1] Homepage SupoSE
  - http://supose.soebes.de
- [2] SVNKit pure Java Subversion Library
  - http://www.svnkit.com
- [3] POI Framework
  - http://poi.apache.org
- [4] ANTLR
  - http://www.antlr.org

On-line Sources II

- [5] Lucene Framework
  - http://lucene.apache.org
- [6] Tike Framework
  - http://incubator.apache.org/tika

Questions?

subconf2008@soebes.com

- Thank you for your attention.