TNTBase - Versioned Storage for XML

Vyacheslav Zholudev

Jacobs University Bremen

KWARC – Knowledge Adaptation and Reasoning for Content

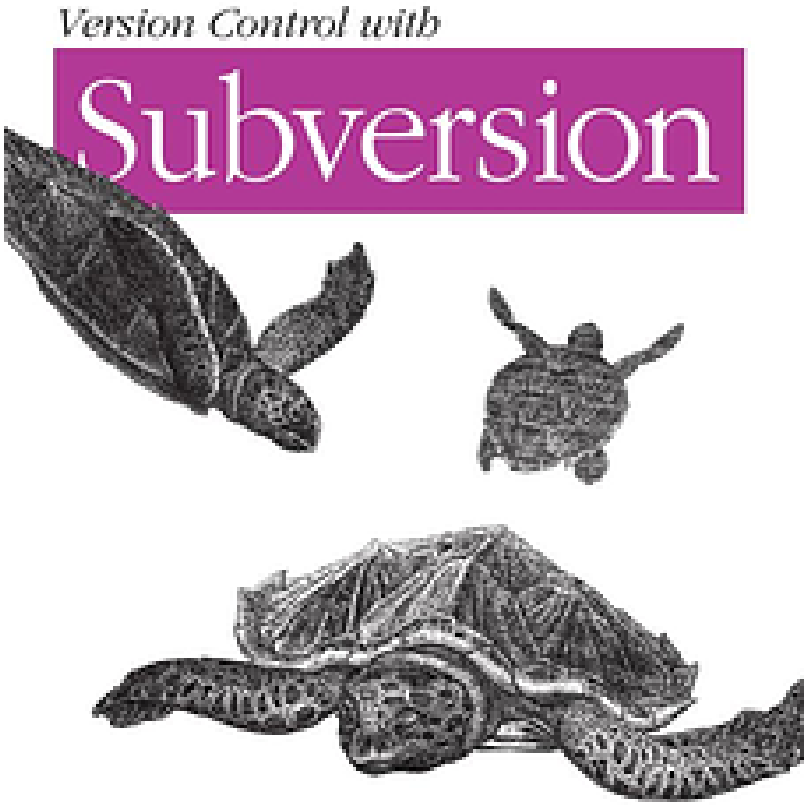
August 12, 2009

Motivation

- Deep Web for XML is emerging
- Infrastructure for *managing changes* in Deep Web for XML is currently frugal
- Version control systems are *deeply text-based*
- *No built-in support* for XML fragment techniques (XPath, XQuery, XML indexing)
- *No powerful versioning* in XML-native databases

Running example

- SVN Book sources:
  `http://svnbook.googlecode.com/svn/`



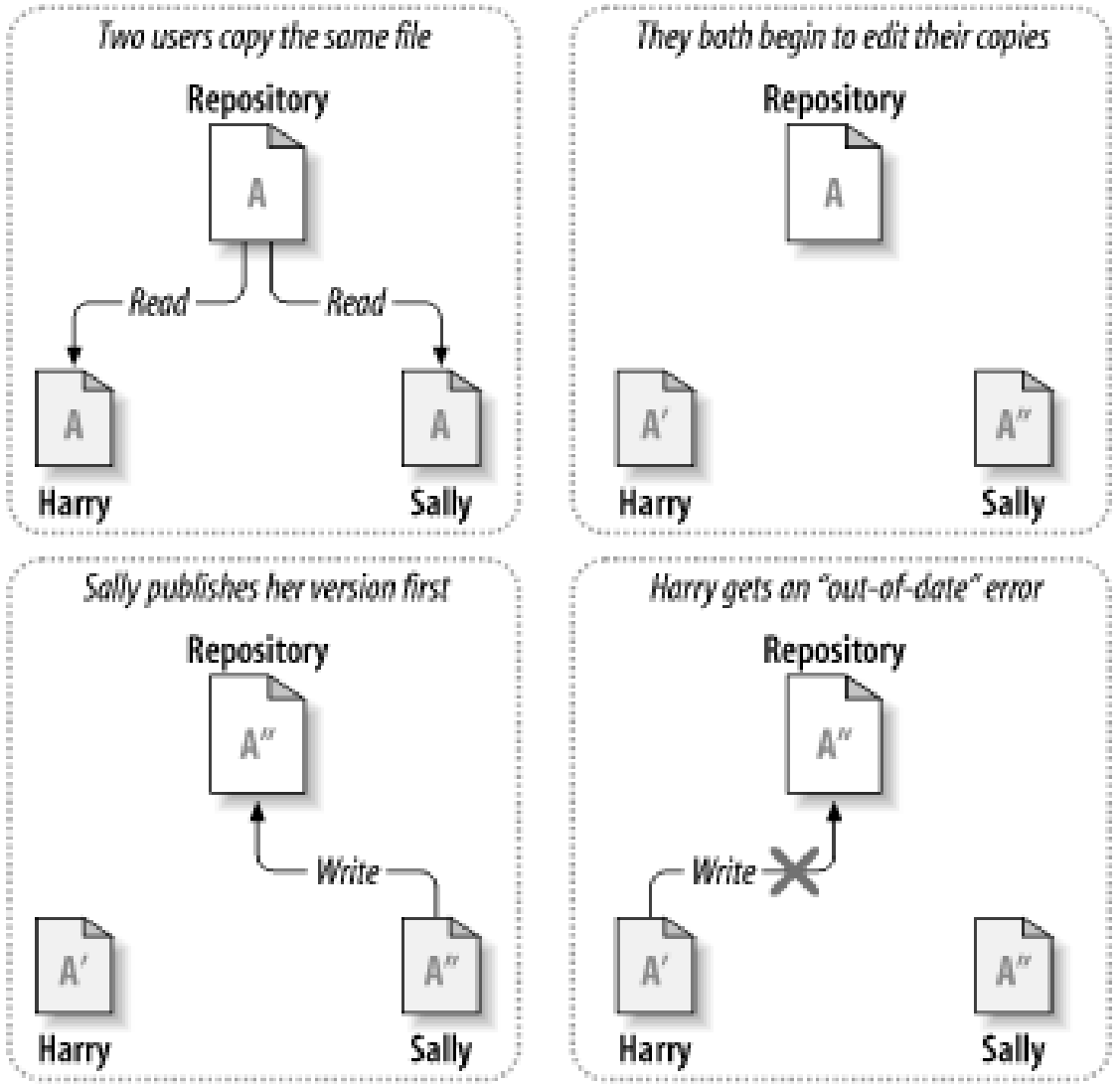
Our Goal: Support your Workflows

- Provide a basis for XML-based management systems (e.g. DocBook-based systems)
- Collaborative editing
- Document history support (versioning)
- XML-fragment access techniques
- Make possible workflows and globally distributed project teams as we know them from Open Source community

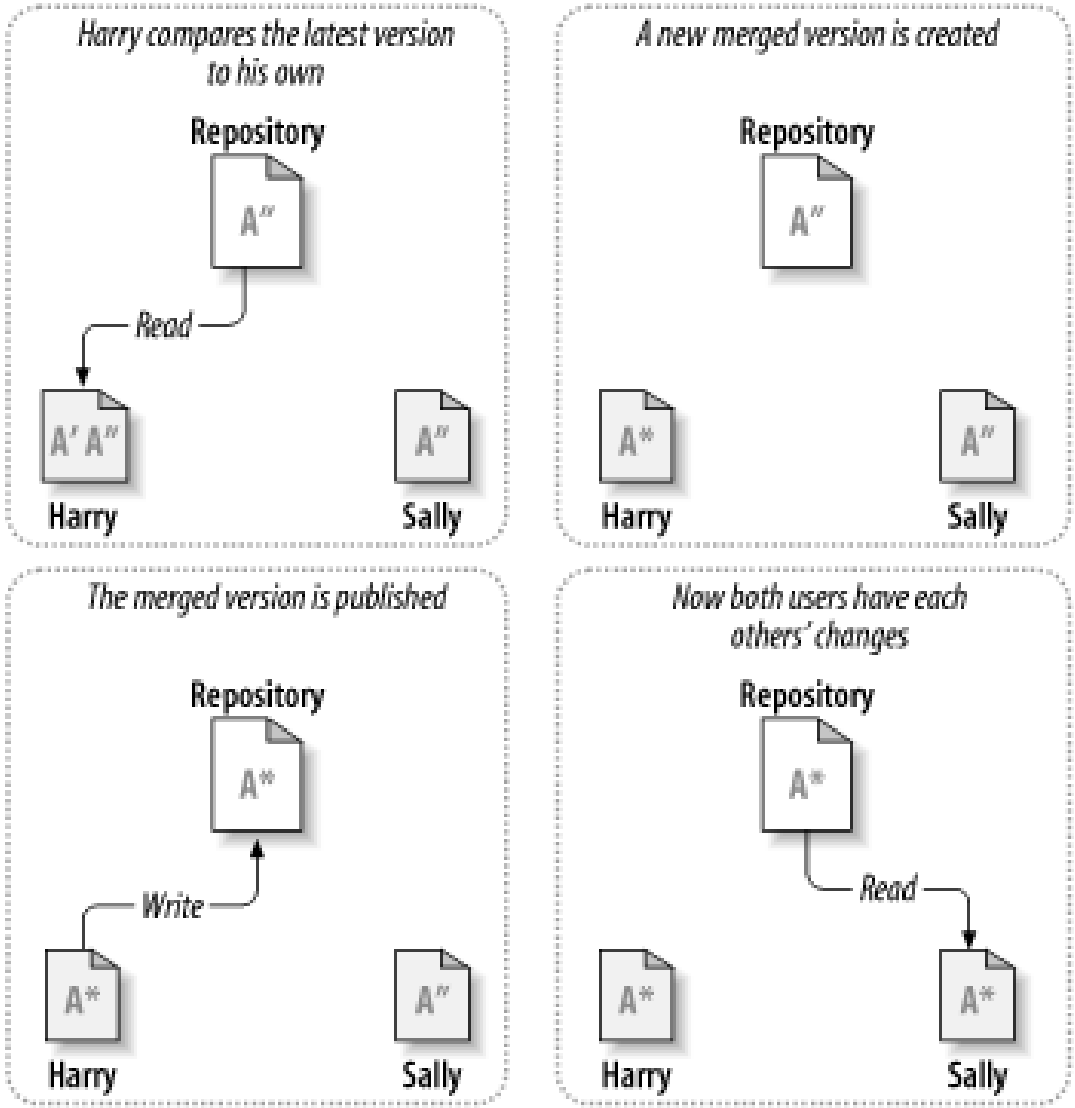
State of the Art

1. Subversion - a Version Control System
2. Berkeley DB XML - an XML-native Database

Subversion = Repository + Working Copies



Subversion = Repository + Working Copies

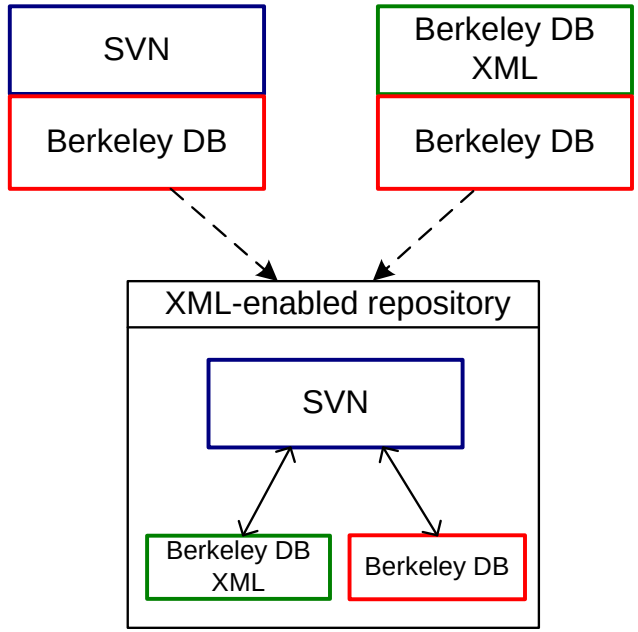


Berkeley DB XML

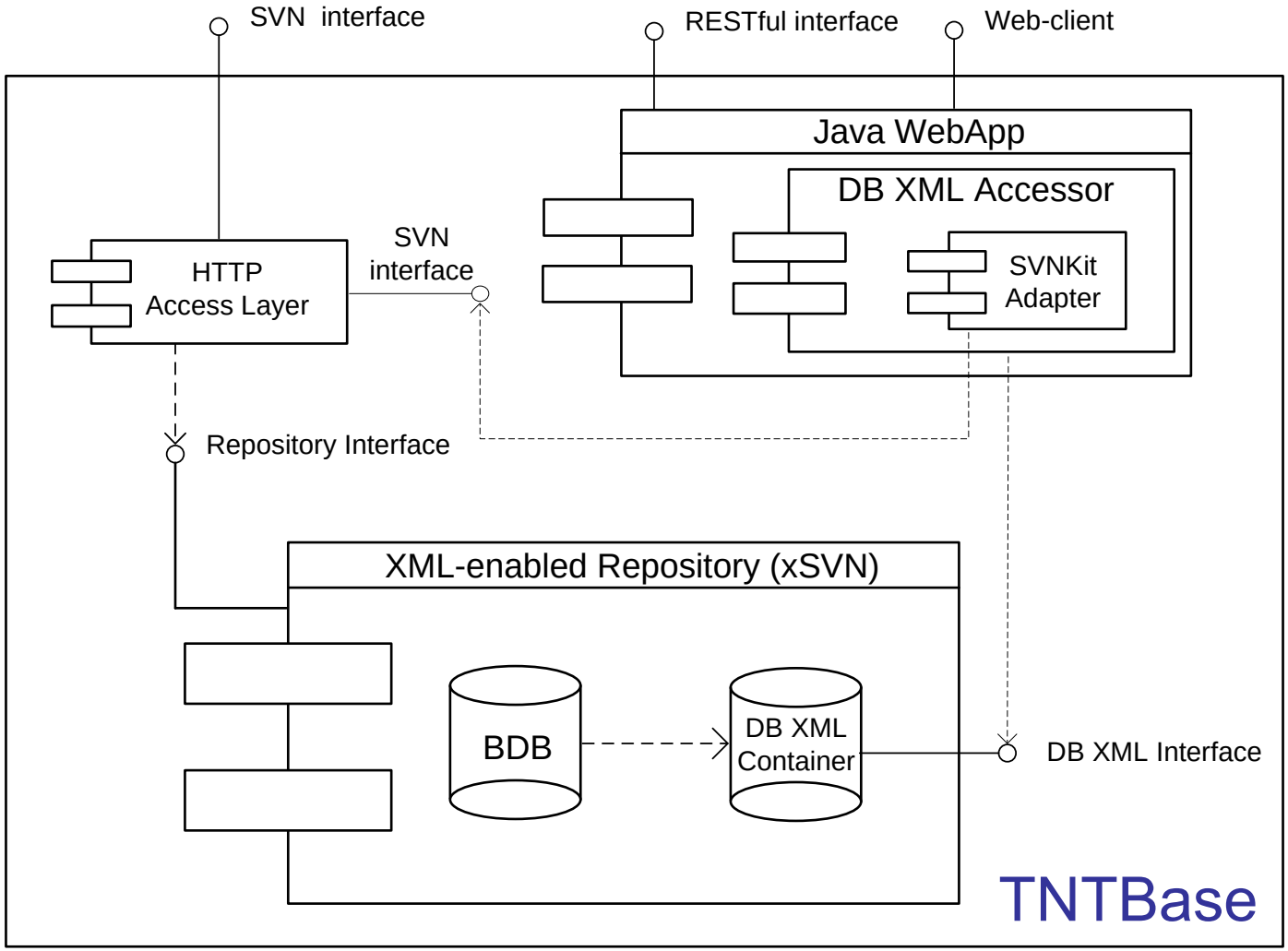
- XML-native embeddable database
- Built on top of Berkeley DB
- XQuery support (including XQuery Update facilities)
- XML indexing
- Support for multi-threaded and multi-process environments
- Transaction support with different levels of isolation

The Idea

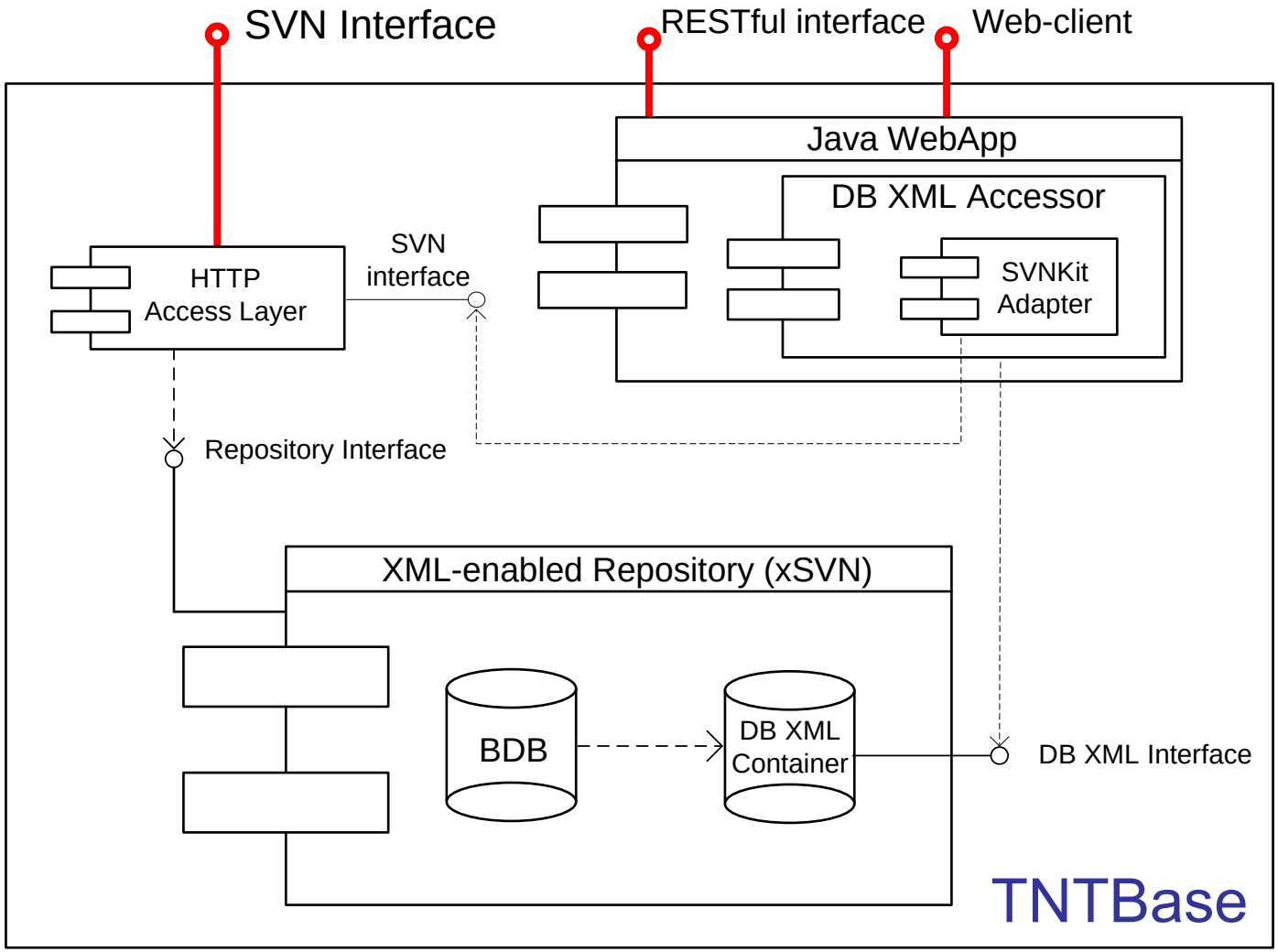
- SVN and Berkeley DB XML utilize Berkeley DB
- XML-content → Berkeley DB XML
- Non-XML content → Berkeley DB
- Advantage: Retain SVN algorithms unchanged



TNTBase architecture

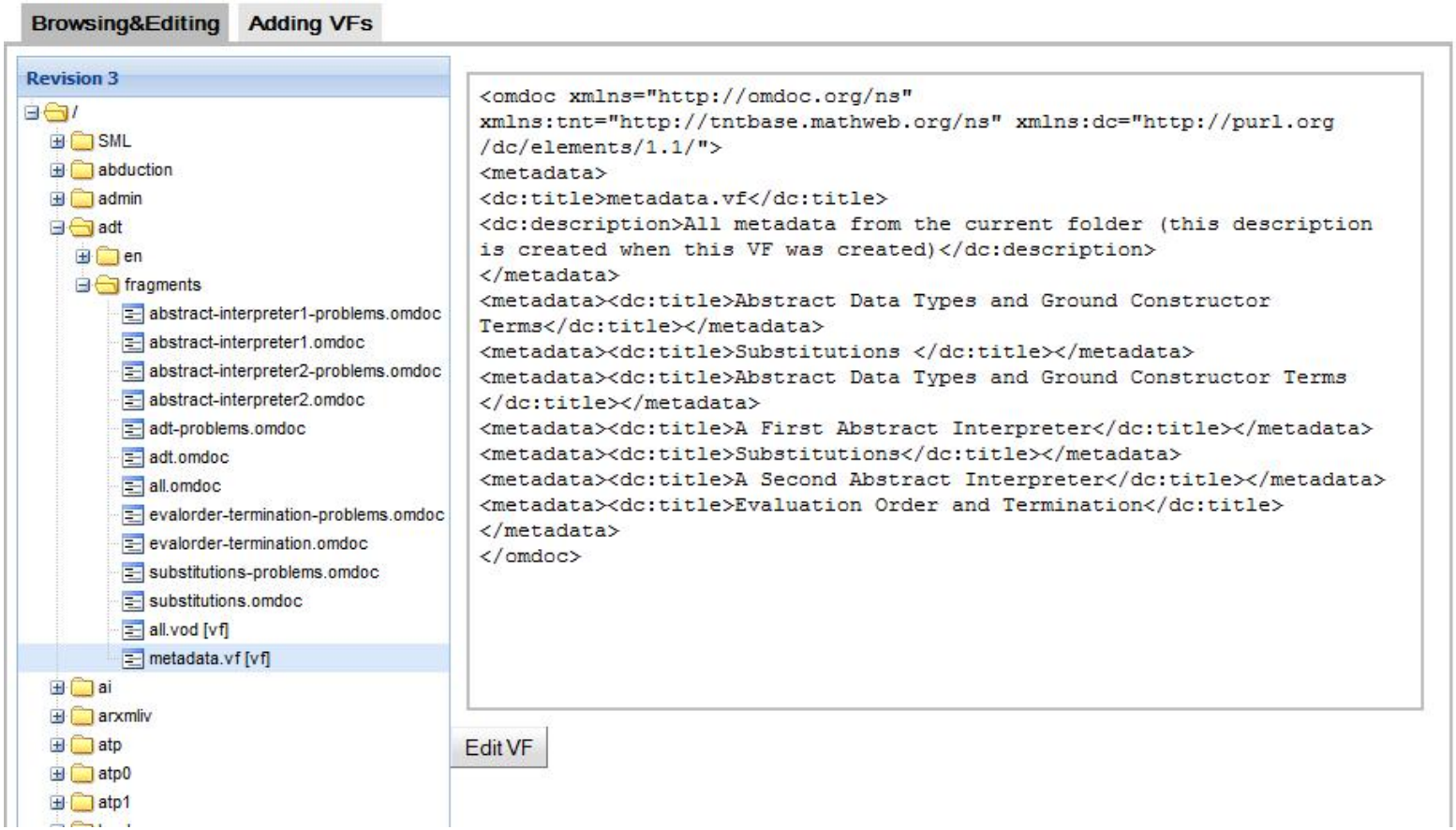


TNTBase architecture

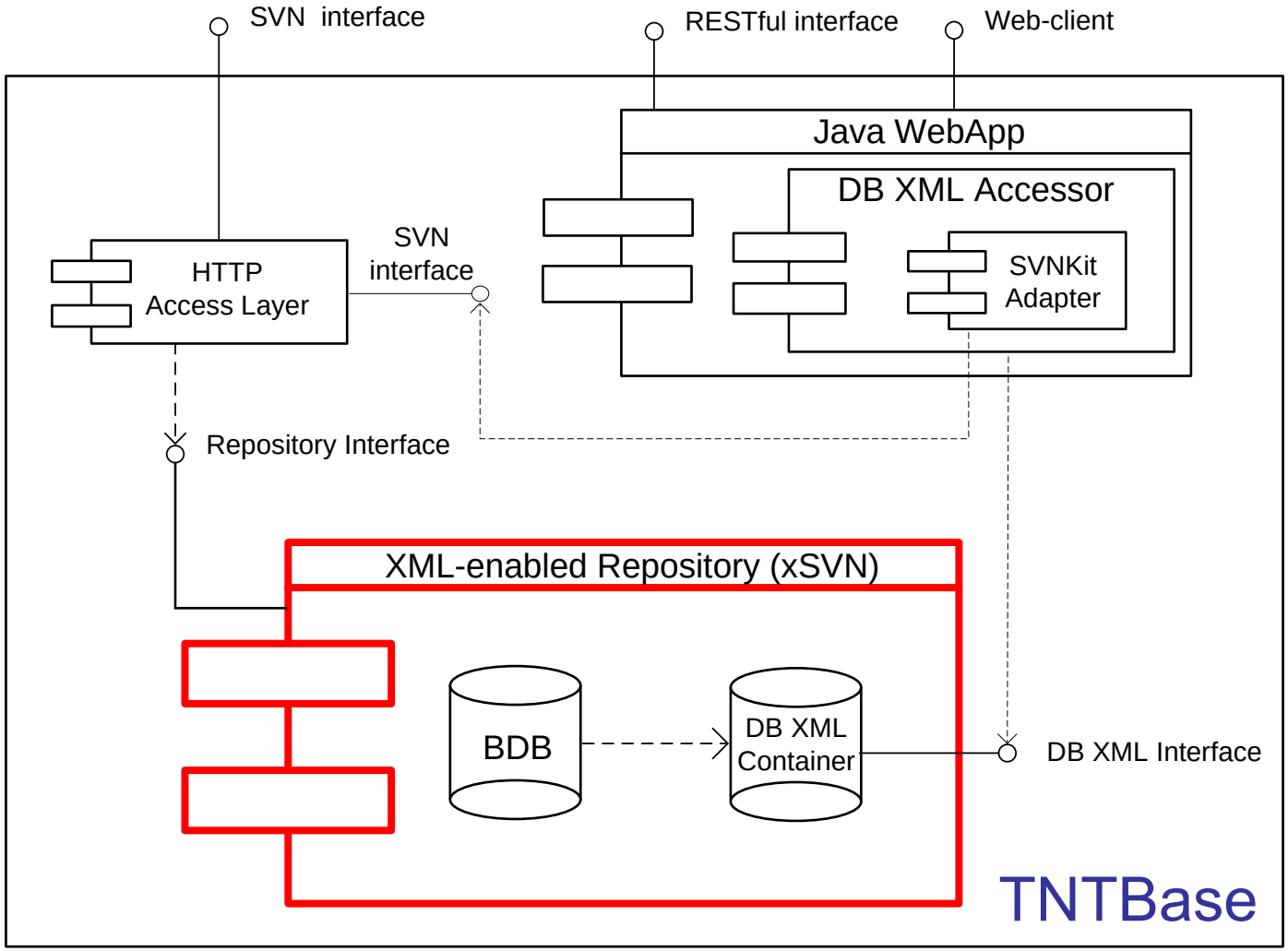


Web-Client

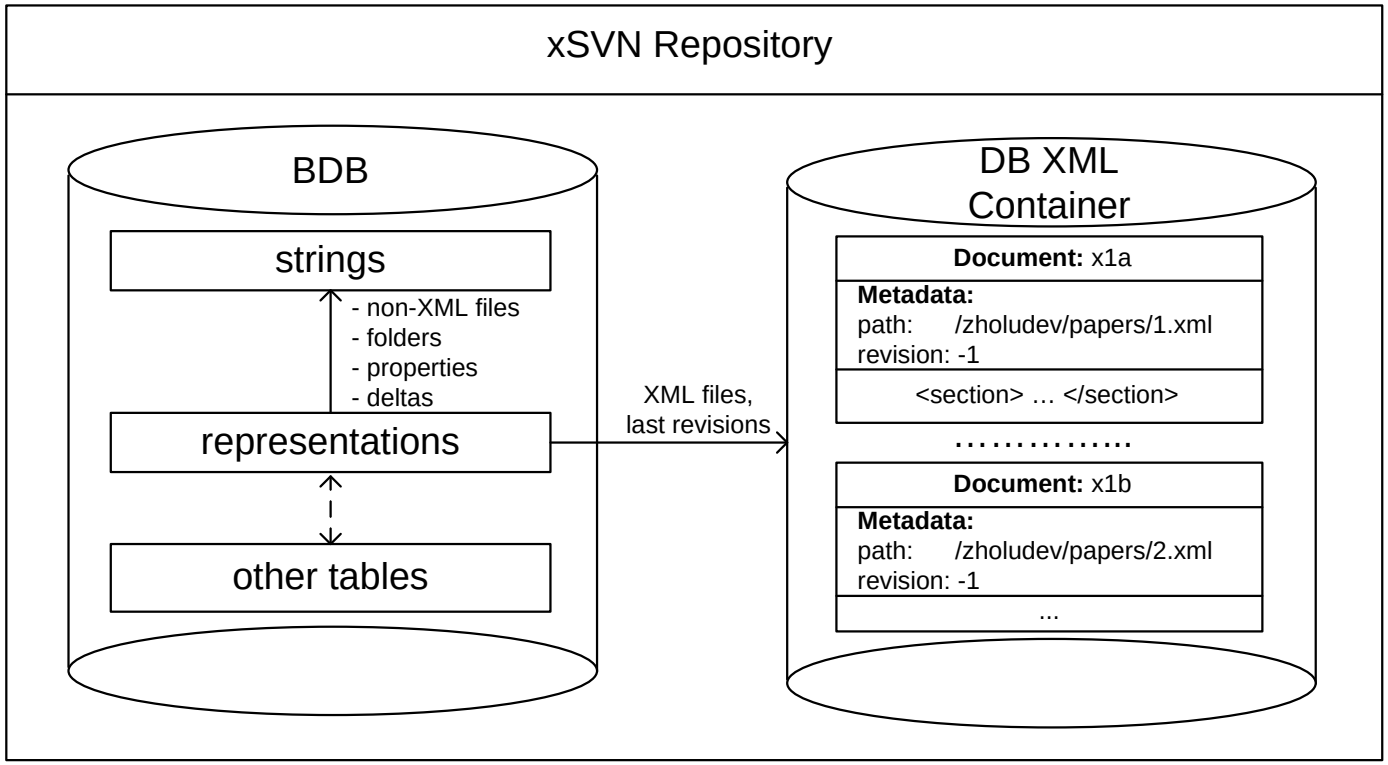
Demo



xSVN



xSVN – Subversion on Steroids



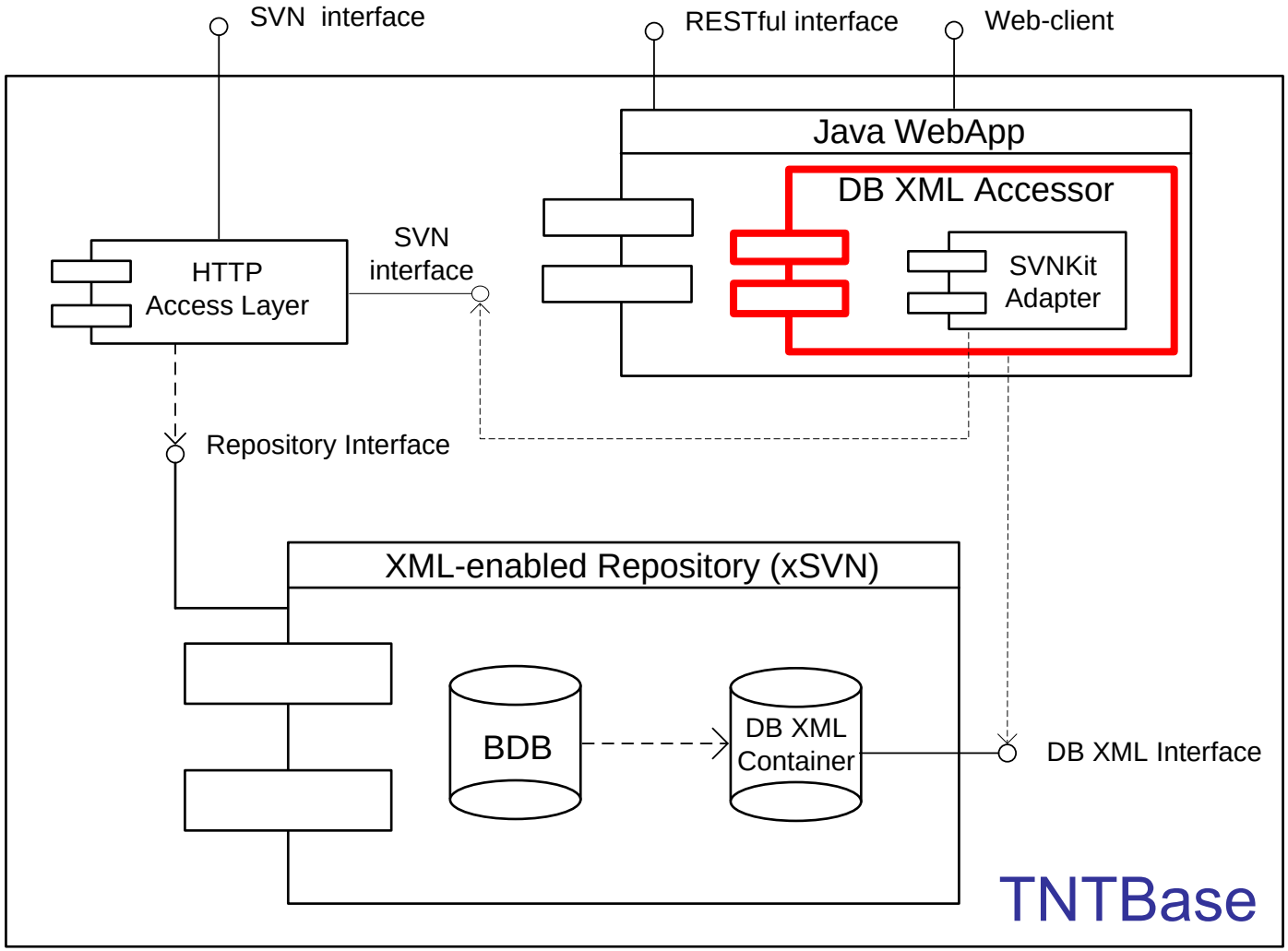
- Idea: store HEAD revisions of XML-files in Berkeley DB XML. Other information is held in Berkeley DB. Deltification mechanisms remain unchanged
- Any SVN client is able to communicate with xSVN

File System in xSVN

- Berkeley DB XML natively does not have a notion of a file system
- Introduced *File System Documents (FSDs)*
- Reflect only XML-content of folders in a FS
- Managing them via XQuery Update facilities
- Allow fast retrieval of folder content information

```
<entries xmlns="http://tntbase.mathweb.org/ns">
  <dir name="stylesheets"/>
  <dir name="references"/>
  <file name="paper_zholudev.xml"/>
  <file name="paper_kohlhase.xml"/>
  <vfile name="notations.vf" id="dbxml_54"/> <!-- will be explained later -->
</entries>
```

DB XML Accessor



Querying, modifying, caching...

- Querying arbitrary collections of XML-documents based on their *FS path*
- Modifying documents via XQuery Update with *preserving* xSVN *history*
- Querying *previous revisions* of XML-documents
- *Caching* query results
- Creating, managing and modifying *Virtual Files* (will talk later)

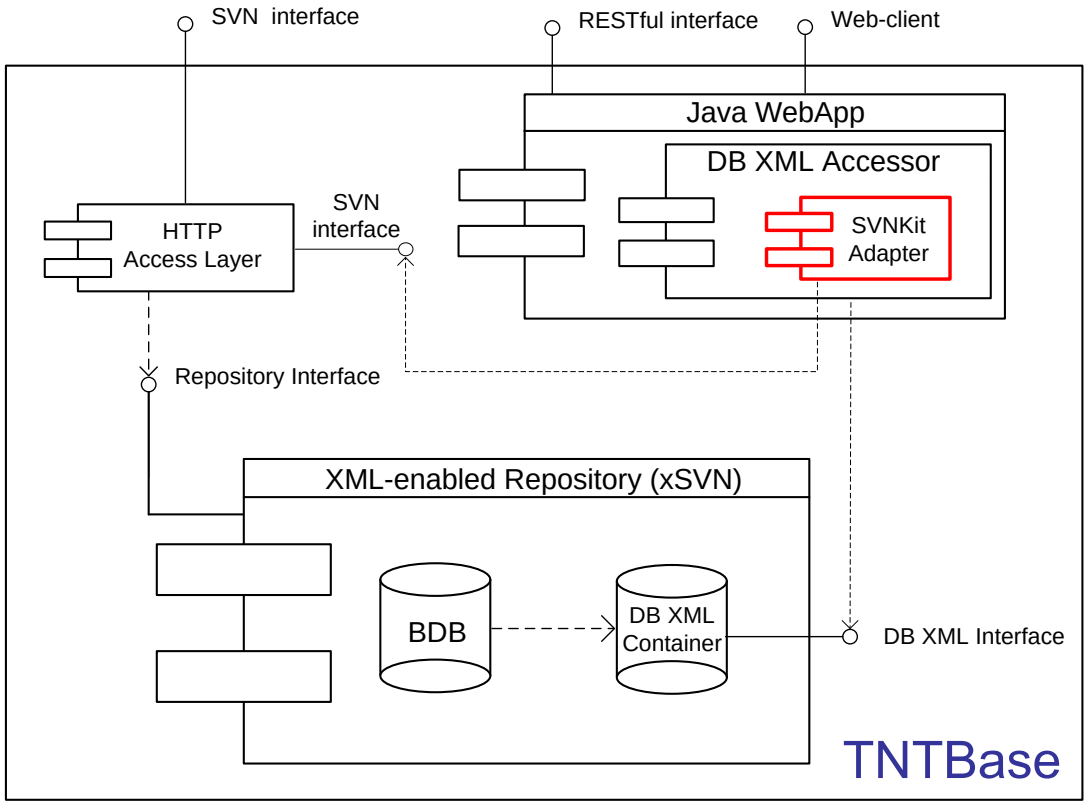
DB XML Accessor — Querying Document Collections

- Since we have a file system in an xSVN's container, we should adapt XQuery syntax
- Path information is stored in metadata of each document

- Accessing all files: `collection()`
- Accessing a single document: `doc(<document_path>)`
- Accessing an arbitrary collection of documents: `collection(<arbitrary_path>)`
- You can use wildcards and '//' in the latter case
- Example: `collection(/branches//*en*/ch01*.xml)//title`

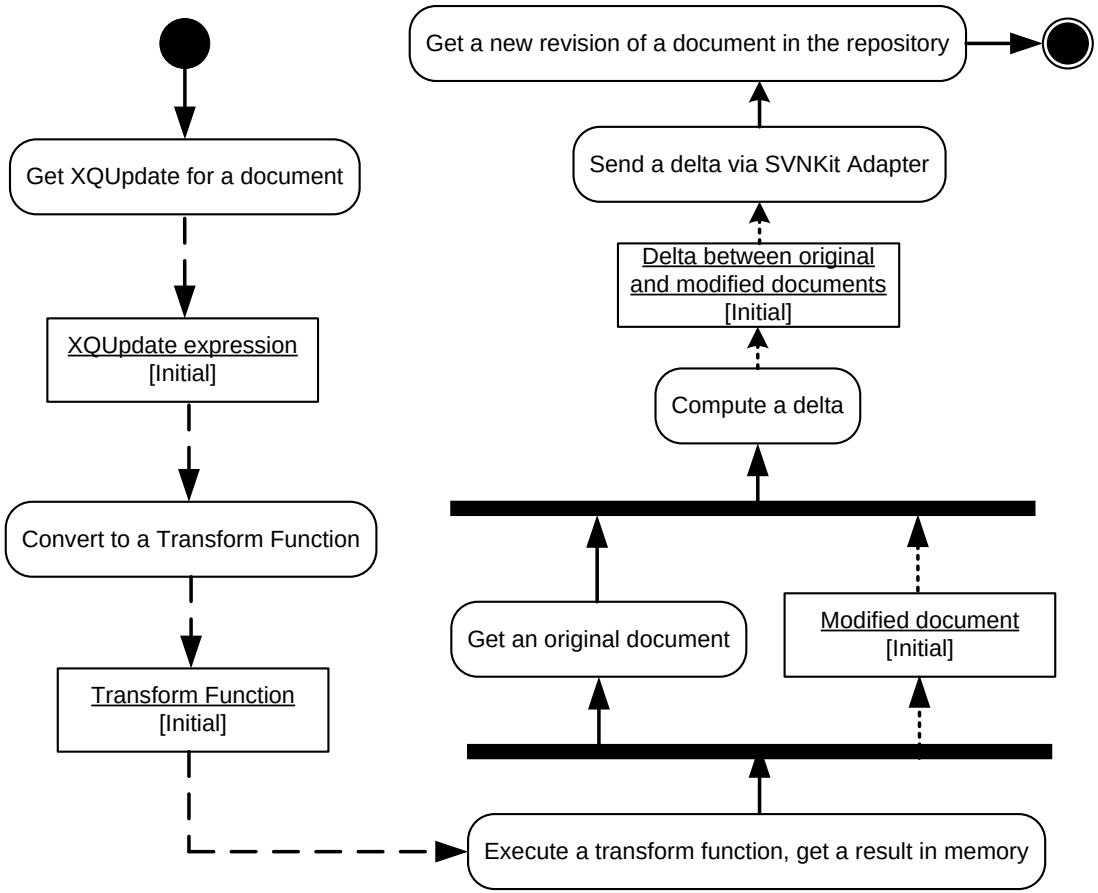
DB XML ACCESSOR — XQuery Update Facilities

- Want to preserve history when a document is modified via XQuery Update



- The SVNKITADAPTER module serves as an SVN client
- Substitute XQuery Update expressions by *Transform Functions*

DB XML Accessor — XQuery Update Facilities



- A new revision of a document has been committed
- All history information is preserved

DB XML Accessor — Querying Previous Revisions and Caching Query Results

What about querying previous revisions of documents?

- Cache previous revisions
- Query like a HEAD revision

When a query takes a while...

- Cache results

Virtual Files — Introduction

- A *Virtual File (VF)* is a TNTBase file system entity
- Contents of a VF are determined by an XQuery expression
- Help to aggregate certain information a single entity
- Like VIEWs in relational databases

- Example: we want to have titles of all chapters in SVN Book in English
- XQuery: `data(collection(//*en*/ch??-*.xml)//title)`

```
<?xml version="1.0" encoding="UTF-8"?>
<tnt:vfile name="/titles.vf" revision="-1" mime-type="text/xml"
  <tnt:results>
    <tnt:result>Subversion Quick-Start Guide</tnt:result>
    <tnt:result>Fundamental Concepts</tnt:result>
    <tnt:result>The Repository</tnt:result>
      .....
  </tnt:results>
</tnt:vfile>
```

Virtual Files — Creating and Retrieving

Creating:

- An XQuery expression and namespaces to execute it
- Revision number
- VF path in a repository

Retrieving:

- VF path

Virtual Files — Caching and Querying

Example: Find all titles that contain a "problem"

1. Cache results
2. Query: collection(/titles.vf)/title[contains(data(.), 'problem')]

Using the same syntax to query VFs

Virtual Files — Editing

- VFs can be *checked out* for editing
- After edited VF is *commited*, changes are propagated to the original documents
- Let's try it

```
<tnt:vfile name="titles2.vf" mode="edit" revision="41"
  <tnt:result file_path="/trunk/ch01-fundamental-concepts.xml"
              element_path="/appendix" element_name="title"
              element_type="element">
    <title>Subversion Quick-Start Guide</title>
  </tnt:result>
   ...
  <tnt:result file_path="/trunk/ch02-basic-usage.xml"
              element_path="/appendix/warning/para/chapter"
              element_name="title" element_type="element">
    <title>Fundamental Concepts</title>
  </tnt:result>
    .......
</tnt:vfile>
```

Future Work

- Stabilize the system further
- Improve efficiency
- Improve stress-tolerance
- Do some case studies
- Implement unified authentication
- Manage cached results wiser
- ... see tickets `http://trac.mathweb.org/tntbase`

OMDoc case study

- OMDoc — Open Mathematical Documents
- Encoding semantics of math
- *General Computer Science* lectures at Jacobs University
- More than 2000 documents
- Generation of presentational documents on the fly
- Definition lookup

Summary

- The TNTBase system – a versioned XML database system that can act as a storage solution for an XML-based deep web
- The implementation effort has reached a state, where the system has enough features to be used in experimental applications
- TNTBase can be used as a usual Version Control System. Mirroring facilities of xSVN inherited from SVN allow to use TNTBase safely without fear to lose important data

Bon appétit!

Questions???

Project page: `http://trac.mathweb.org/tntbase`

Why not XML-diff (yet)

- xSVN's deltification algorithms are inherited from normal SVN
- SVN is a very complex system
- The text-based diff-algorithms are efficient, fast and reliable, nicely fit in with SVN architecture
- Any advantage to changing the deltification *on the server*?
- XML-diff brings advantages *in the client*: smaller and less invasive deltas, more informative conflict resolution strategies
- The decision whether we want XML-diff on the server is deferred for the future