Archive Tutorial

Simon Farrow   Holger Meuss   Andreas Wicenec

Central point of information:

Archive tutorial (CVS: in `Archive/doc/`)

Today:

- ➜ Unique Identifiers for documents
- ➜ `XMLEntityStruct`: communicating documents
- ➜ Using the Archive Interface
- ➜ Example

Note: official version has tag `ARCHIVE_20030616`

Unique Identifiers (UIDs)

have the form of a URI:

`uid://`*global*/*local*

e.g.: `uid://X0123456789abcdef/X01234567`

- ➜ Global part (16 chars) and local part (8 chars) are hexadecimal numbers
- ➜ Users can reserve global parts to locally create unique UID
- ➜ More general identifiers possible in the future: `ngas://`..., `file://`...
- ➜ UIDs must be fetched: `IDENTIFIER_ARCHIVE.getIdNamespace()`
- ➜ UID syntax checking provided
- ➜ No version information included

XMLEntityStruct

- ➜ used in most communication with Archive
- ➜ struct XmlEntityStruct

```
struct XmlEntityStruct
  {
    string xmlString;
    string entityId;
    string entityTypeName;
    string schemaVersion;
  };
```

Archive Interface

➜ Interface for storing, deleting and querying XML documents
➜ Document is identified by UID and version number

➜ update
➜ un/delete
➜ status
➜ retrieve
➜ query

`void` **update** `(in XmlEntityStruct entity):`

- ➜ Combines store and update functionality
- ➜ If UID already exists: new version is generated
- ➜ Else: document with version number 0 is generated

`void` **delete** `(in URI identifier, in long version, in boolean deep):`

- ➜ deletes (logically) a given version of a document
- ➜ can always be restored
- ➜ Specify `version` -1 for deleting the latest version
- ➜ `deep`: delete recursively other documents being referenced by deleted document. *Might be removed...*

`void` **undelete** `(in URI identifier, in long version, in boolean deep):`

- ➜ restores a deleted version
- ➜ parameters as in `delete`

`StatusStructSeq` **status** `(in URI identifier, in boolean deleted):`

- ➜ returns status information for *all* versions of document
- ➜ `deleted` specifies whether information about deleted documents shall be included

A `StatusStruct` contains the following fields:

- ➜ `boolean readLock`
- ➜ `boolean updateLock`
- ➜ `string permissions`
- ➜ `boolean deleted`
- ➜ `string entityTypeName`
- ➜ `string schemaVersion`
- ➜ *Fix shall include version number :-)*

`XmlEntityStruct` **retrieve** `(in URI identifier, in long version):`

- ➜ retrieves a given version of document
- ➜ version -1 stands for the newest undeleted version

`Cursor` **query** `(in string query, in string schema):`

- ➜ **order of parameters in IDL file wrong!**
- ➜ XPath query against document
- ➜ Returns `Cursor` object containing document fragments

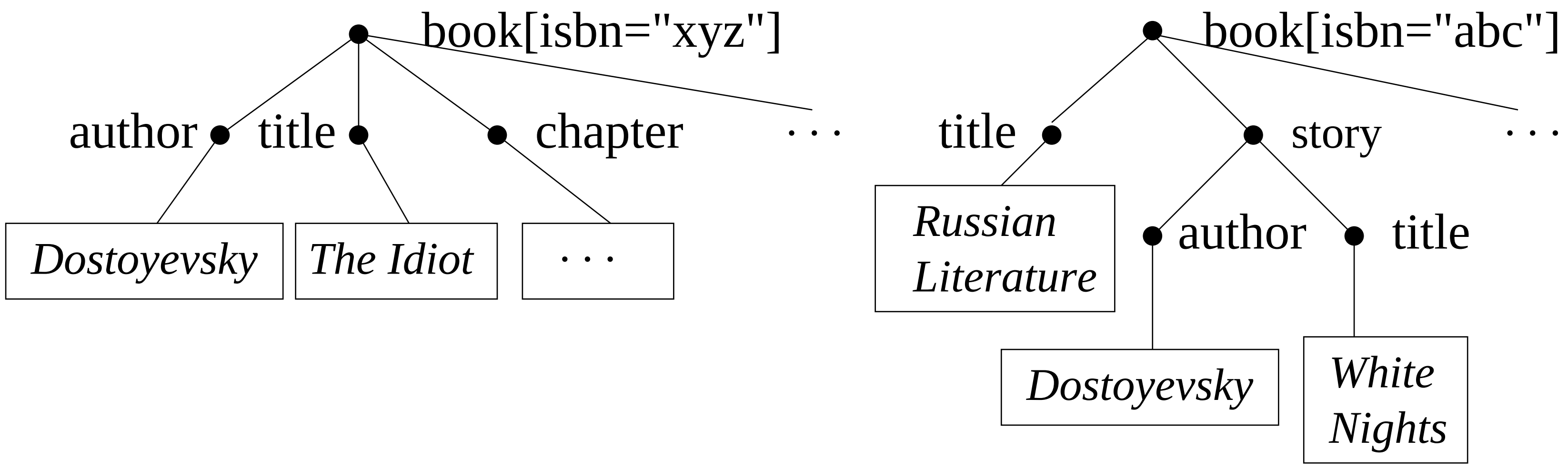
Words of warning:

- ➜ Millions of documents in database: restrictions on queries
- ➜ Namespaces not yet supported in `query`

Excursion: XPath

W3C standard for “pointing” at parts XML documents, similar to paths in file systems:

```
/book/title
/book//title
/book[./author="Dostoyevsky"]/title
/book@isbn
```



Example document:

```
<book isbn="xyz">
  <author> Dostoyevsky </author>
  <title> The Idiot </title>
  <chapter> ... </chapter>
  ...
</book>
```

Syntactic Constructs:

- / child step
- // descendant step
- . self step
- .. parent step
- @ attribute step
- Simple functions
- Unabbreviated syntax: richer language (not sure whether all will be supported)

CURSOR

- ➜ Simple instrument to deal with big result sets
- ➜ works like `Iterator` in Java:
- ➜ Method `hasNext()` to check whether more results exist
- ➜ Method `next()` to get next result of type `QueryResult`:
  - ➜ `identifier`: UID of full document (`URI`)
  - ➜ `version`: version (`long`)
  - ➜ `xml`: XML string, i.e. document fragment (`string`)
- ➜ Method `count()`: number of results in Cursor. *Might be removed...*
- ➜ Results are living in archive (`Offshoot` Interface)

Permissions and Locks

- ➜ Infrastructure for permissions and locks prepared
- ➜ Concrete realization not yet sure

User groups:

- ➜ All
- ➜ Principal Investigator
- ➜ Authorized

Permissions

- ➜ Open (All)
- ➜ Restricted (Only Authorized and Principal Investigator)
- ➜ Protected (Only Authorized)
- ➜ Hidden (Only Authorized)

Locks can be requested by clients:

- ➜ Read
- ➜ Update

Example Client

`ARCHIVE/src/alma/archive/client/OperationalClient.java`

- ➜ get Archive and Identifier reference:

```
Operational archive = ...
Identifier ident = ...
```

- ➜ get an ID:

```
String id = ident.getIdNamespace();
```

- ➜ create document structure

```
XmlEntityStruct struct = new XmlEntityStruct();
struct.entityId = id;
struct.xmlString = "<example>example</example>";
...
```

- ➜ store document structure

```
arch.update(struct);
```

- ➜ retrieve document

```
XmlEntityStruct struct = arch.retrieve(id,-1);
```