The Kador – Semantic management of document collections.

Philippe Rigaux

March 4, 2011

What is the point?

Apply WebDam results to a specific application area: **document management**.

Why?

Documents are everywhere, and currently poorly managed (consider your hierarchical file system: are you happy with it?)

The goal

1. Model document management functionalities with a WebLog-like language;
2. Try to build a system based on these principles, using other components (e.g., data management systems);
3. check that the result does bring something new (the guess is: yes);

If everything is successful, then create a startup and make a lot of money.

Starting point

Users interact with documents (and document collections) in many:

1. You create, edit, manipulate documents on your desktop;
2. You search, collect, browse documents on the Web;
3. You share and exchange documents with your friends and colleagues.
4. And now, you must deal with many other contexts: your SmartPhone, your iPad, ...

Fact: we all spend a lot of time in document manipulation.

Question: is there a way to make our life easier by **declaring** these manipulations, with a generic approach?

And the bet i: yes, something like WebDamLog could be the missing layer between **document apps** (users), **documents repositories** and **document services**.

What do we expect from the modeling language?

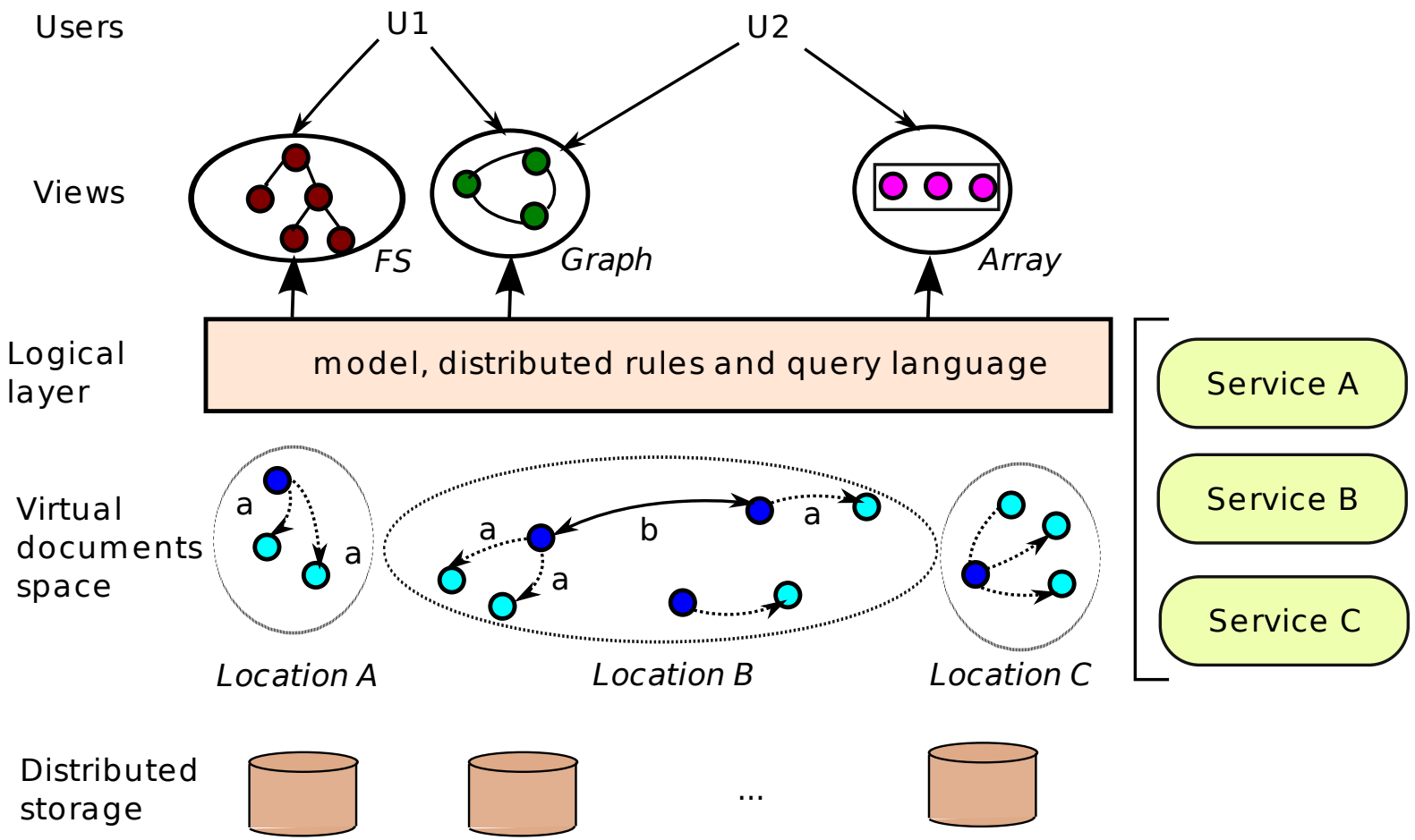
Do not add any complexity! users must be able to directly use their familiar environment – e.g.

- in a folder, editable with standard Desktop softwares,
- and as a resource in a Web application,
- and, after any appropriate transformation, on you iPad.

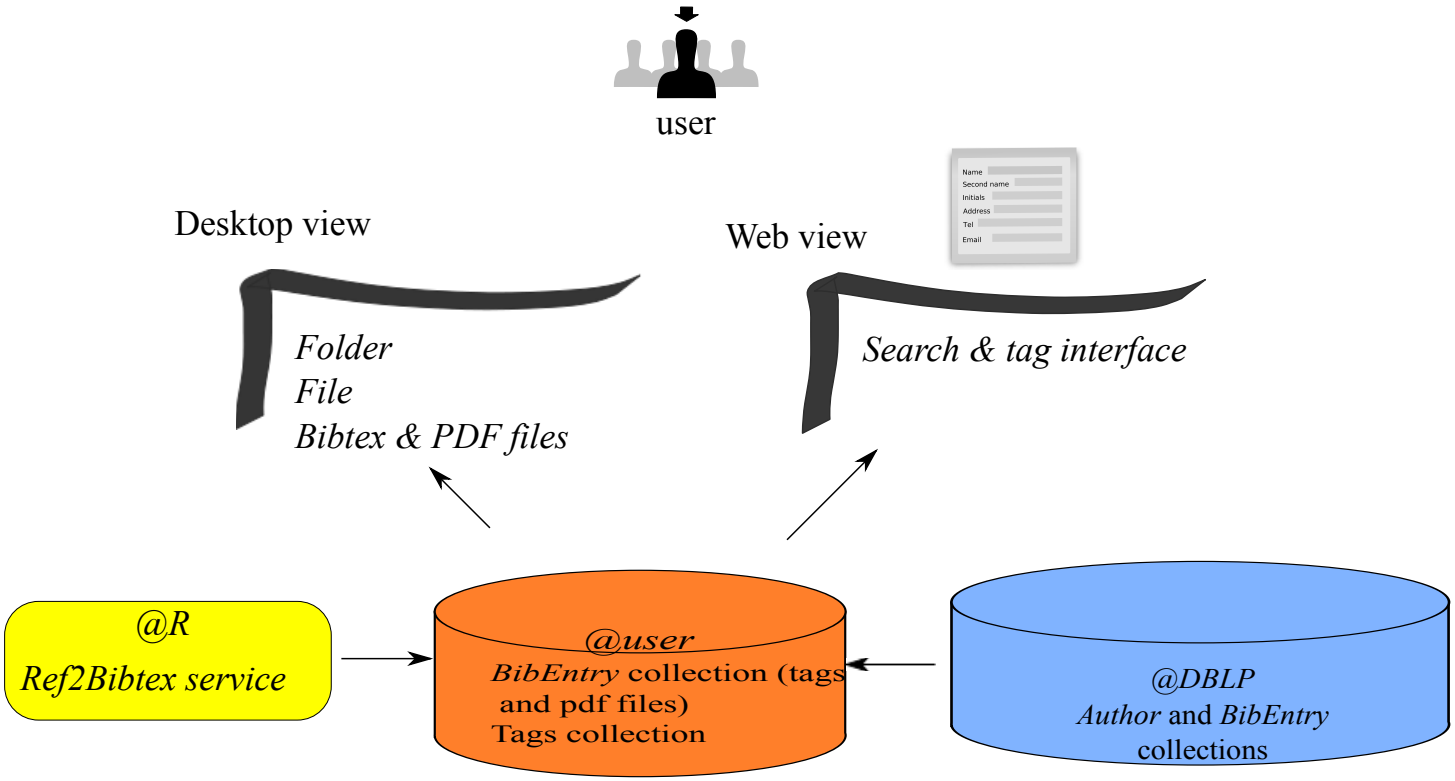
Uniformly support content management functionalities.

- **Create** and **edit** documents; associate **transformations** and **derivations**.
- **Share** with other users; manage **conflicts** and reconciliation; manage access rights.
- Annotate, classify.
- and, of course, **search**, including **by content**.

The vision


Users
U1
U2
Views
FS
Graph
Array
Logical layer
model, distributed rules and query language
Service A
Service B
Service C
Virtual documents space
a
a
a
a
b
a
Location A
Location B
Location C
Distributed storage
...

Instance of the vision: The easyBib application



Outline

In the rest of the talk:
The Model. Basically WebdamLog, tailored to content management.

Illustration with your the schema and rules of easyBib: shows how to make LaTeX bibiography management **trivial**.

And a **demo** !

Note of caution: work in progress, comments welcome.

Types and schemas

Documents are typed in a **complex values** model, with **references**.

```
Book {title : string,
      authors : [Author],
      publisher: string,
      year: int
      }
```

- The schema of an intentional collection is a pair $\{\_id : \mathbf{I}, value : \tau\}$, where τ is a document type.
- the schema of an intentional collection is simply a document type τ.

Instances

Instance of a document:

```
Book@DBLP{_id : &d1
                {authors : [&a1, &a2, &a3, &a4, &a5],
                 title : "Web Data Management",
                 publisher : “Cambridge University Press”,
                 year : "2001",
                 }
}
```

Ids can be provided or automatically assigned by the system.

Distribution

Instances can be distributed in several ***locations***.

Example: add tags to a book reference; attach a PDF content to this document.

```
TaggedBook@u{_id : &d1
                book: &b,
             tags : tags,
             content : pdfFile
             } :- Book@DBLP(b),
                        Tag@u(t), Tagged@u(b.id, tags, pdfFile)
```

Replication is a trivial variant.

Derivation (functions) – production of new contents

Restucturation: produce a derived content (a Bibtex entry) for each DBLP ref.

Bibtex@u {bibtex: b,ref: &i } :-
Book@DBLP <&i, p>, Fbibtex@R{p, b}

where Fbibtex transforms a `Bibentry` instance to a Bibtex string.

Extraction: call an external service (location S) that extracts terms from a PDF document.

Index@u {article: &i,token: t } :-
Book@DBLP <&i, {pdf: p}>, Pdf2Term@S{p, t}

Pdf2Term is the service. It captures all pairs (p, t) such that t is a term in p.

Understanding easyBib: the Desktop View

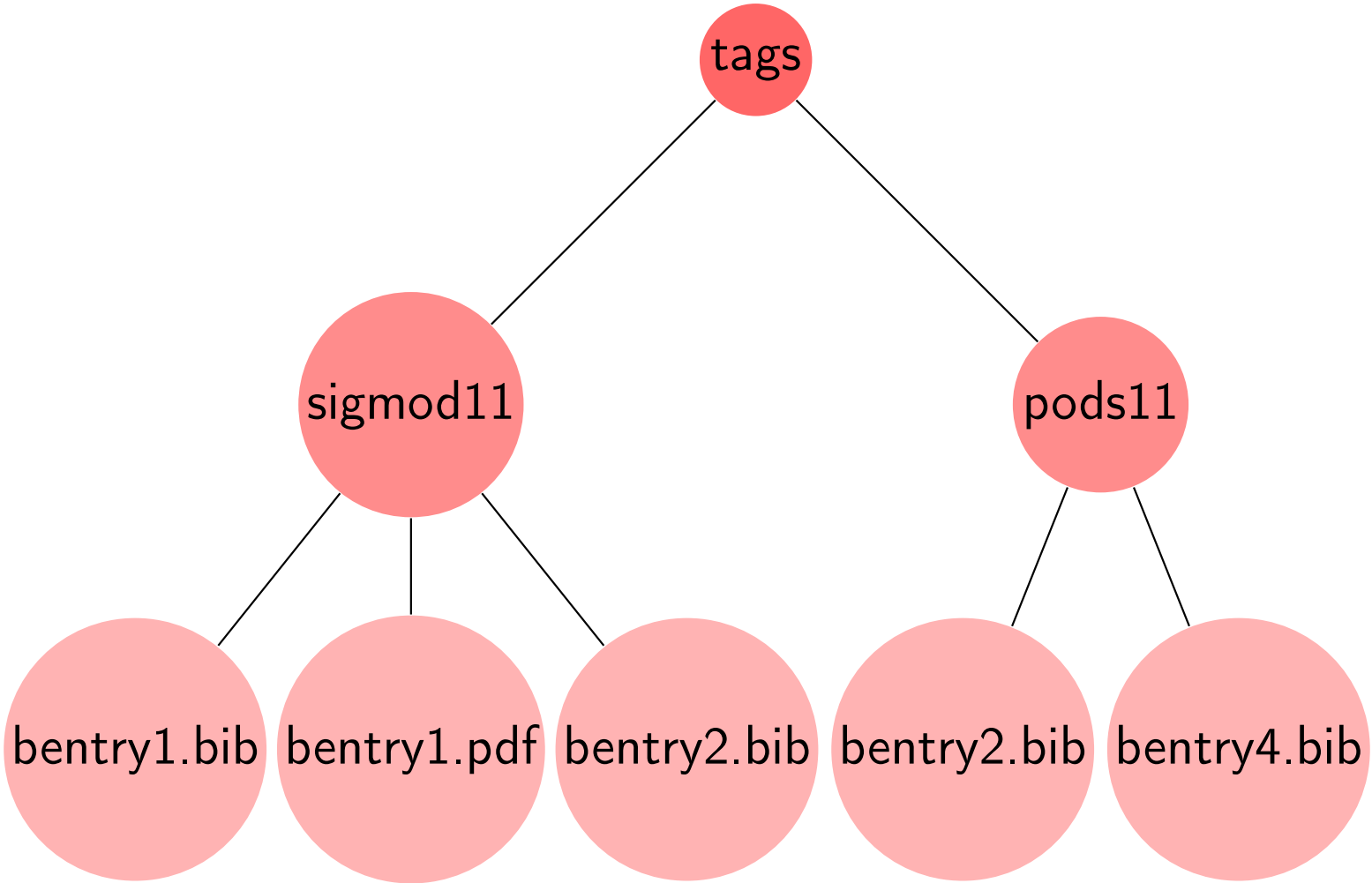
Allows to create, **by query**, a virtual file system (VFS) that presents the documents.

Relies on two predefined predicates:

- `Folder (name: string, parent: Folder)`
- `File (name:string, extension: string, content: blob, folder: Folder)`

Example: create a virtual `tags` VFS, at location G, with one folder for each tag t, and in each tag/folder, the list of bibentries "tagged" with t.

The *tags virtual file system*



Populating the view (1: folders)

First, create the root with:

Folder@G (name: “tags”, parent: `null`) :-

Next, for each tag, we create a second directory level.

Folder@G (name: t.label,parent: i1) :- Folder@G (<i1, “tags”, `null`>),
Tag@u(t)

Populating the view (2: files)

Populate the "tag" directories with PDF files.

File@G (name:b.title, extension: "pdf", content: b.pdf, parent: i1) :-
Folder@G (<i1, t, _>), TaggedBook@u (b), t in b.tags,
b.pdf not null

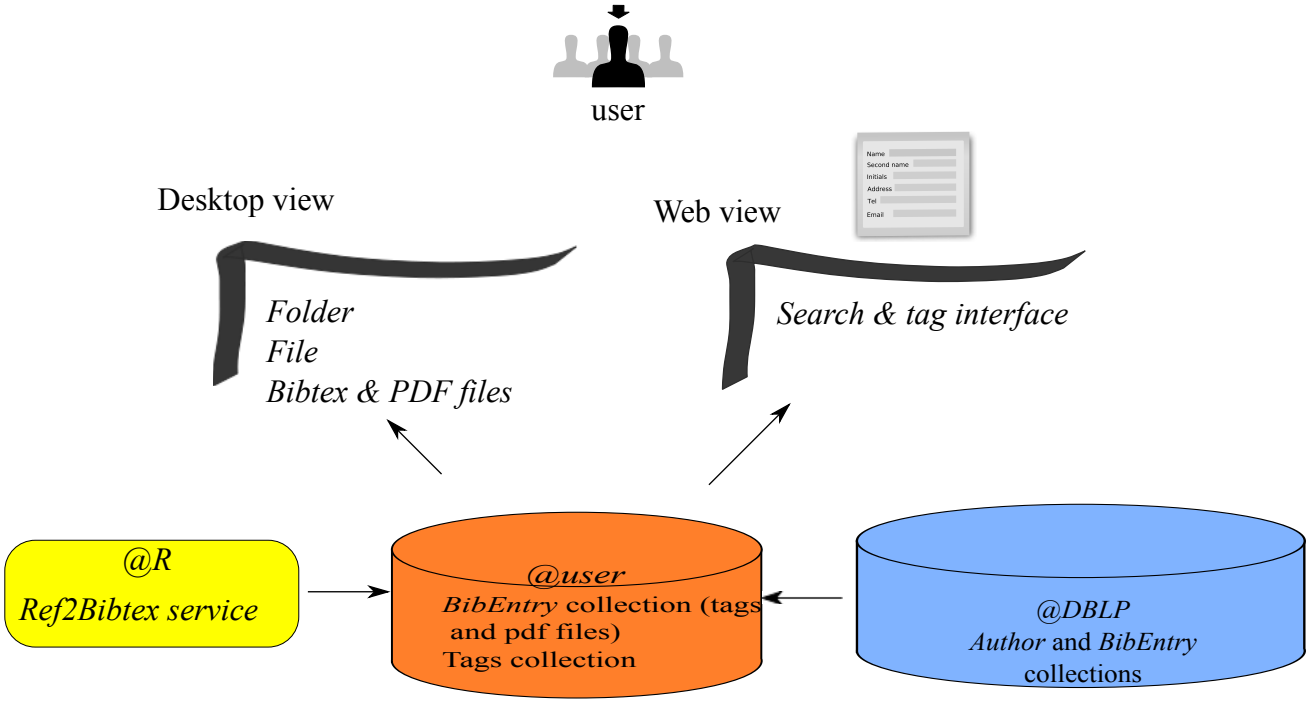
Finally, create Bibtex entries by calling the derivation service:

File@G (name:b.title, extension: "bib", content: s, parent: i1) :-
Folder@G (<i1, t, _>), TaggedBook@u (b),
t in b.tags, Ref2Bibtex@R(b, s)

Of course, a "file" may appear in as many Virtual File Systems as we want.

The demo

Serge, if you read this: I will organize a personal session for you.



To summarize

Main points of interests (IMO)

- **Store your documents in a virtual space.**
  -> no need to worry anymore about monolithic organization.
- **Describe schema, behavior and manipulation** of your documents with a consistent language.
  -> makes it easy to express and understand what the system aims at.
- **Create views to manage documents organizations** that fit a specific user context.
  -> any user action is reflected in all the views.

Work in progress: language implementation, and query evaluation over very large document datasets.

The reverse point of view is also an interesting research topic: how to extend file systems capabilities with DB-like modeling, search, and indexing features.