A Data Model for Digital Libraries

Nicolas Spyratos[1] Carlo Meghini[2] Jitao Yang[1]

[1]Université Paris Sud - LRI, Paris
[2]Consiglio Nazionale delle Ricerche - ISTI, Pisa

MPG eScience Seminar 2009 on Repository Systems
Garching, June 25-26, 2009

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Motivations

"A recent self-assessment amongst service centres in the large European CLARIN project revealed that about 80% of the institutes are restructuring their repositories"

- To facilitate the development of software systems that manage curated collections of digital objects (DLs).
- To create a yardstick, against which to "measure" DLs.
- To facilitate the establishment of DL as a research area in computer science (by highlighting the mathematical structure underlying a DL).

In a way that is:

- *As simple as possible, but not simpler.*
- Compliant with the Web, the largest and most accessed DL ever.

Goal

We need a level of abstraction over the overwhelming amount of details involved in the management of a DL, *i.e.*, a *data model.*

Operations provided by the model:

- *describe* an object of interest according to the vocabulary of the community;
- *discover* objects of interest based on content and/or description;
- *view* the content of a discovered object;
- *identify* an object of interest, in the sense of assigning to it an identity;
- *re-use* objects in a different context.

We want to define structures for carrying out these operations and give algorithms for their implementation.

A small digression

Persistence *vs* preservation.

The data model we are looking for is for persisting the digital objects and the related knowledge, *i.e.*for short-term access.

For long-term access, we need to preserve the information.

A different data model, trying to cope with the consequences of the passage of time.

Digital Objects

A DL includes a set of digital objects.

A DL is very different from a database. A database contains *representations* of facts and of the involved objects.

Digital objects are *not* representations.

Intuitively, we think of a digital object as a piece of information in digital form such as a PDF document, a JPEG image, a URI and so on.

A digital object can be processed by a computer, for instance it can be stored in memory and displayed on a screen.

O : a (non-empty, countable) collection of digital objects.

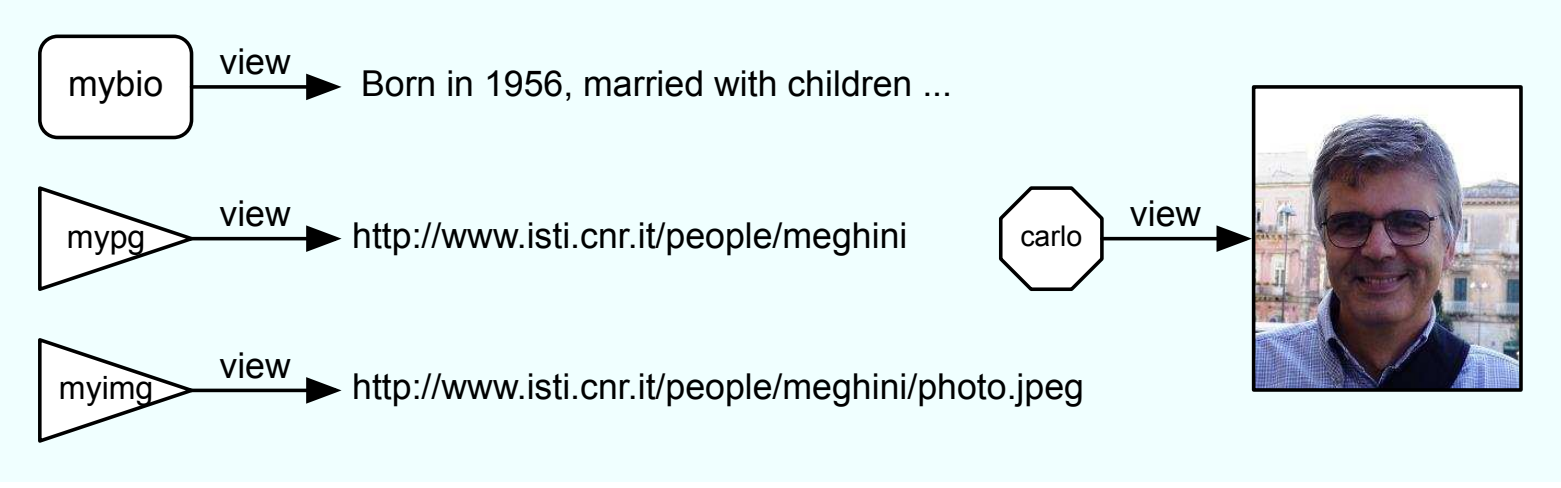
A digital object has 4 basic features: view, content, versions and descriptions.

View

We assume that each digital object can be *viewed* using an appropriate mechanism.

view(o) : the view of o

view is a total function having the set O as domain. The range of view is outside the scope of our model.



Some technical preliminaries

We need to define two fundamental notions: schema and database.

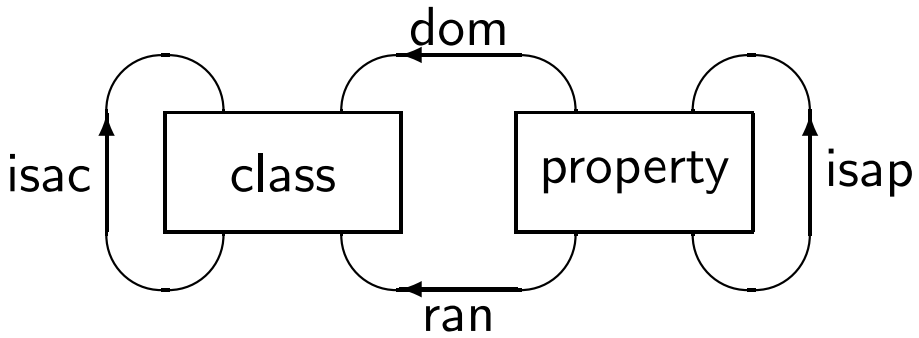
In a traditional database, there is one schema.

In a DL there are many schemas, each addressing a specific aspect of digital objects.

How to stay simple and meet this requirement?

We try to do schema and database generation with a single operation on graphs.

Schemas



Formally, a schema is a function σ that associates:

- each node n of the MSG with a finite subset $\sigma(n)$ of digital objects, $\sigma(n) \subseteq \mathsf{O}$, and
- each arrow $f : X \to Y$ of the MSG with a set-valued function $\sigma(f)$:

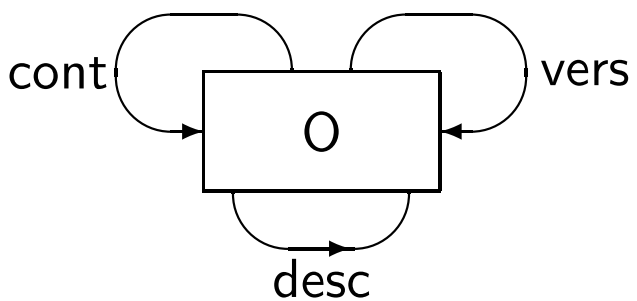
$$\sigma(f) : \sigma(X) \to \mathcal{P}(\sigma(Y)).$$

Schemas

Practically, a schema consists of:

- a set of classes
- a class taxonomy (is-a classes)
- a set of properties
- a property taxonomy (is-a property)
- the domain(s) of each property
- the range(s) of each property

The digital repository schema:



Database

A *database over* a schema s is a function d that associates:

- each class c in s with a finite subset $d(c)$ of digital objects, $d(c) \subseteq \mathsf{O}$, the *extension* of c in d
- each property p in s with a set-valued function $d(p)$ over the digital objects, $d(p) : \mathsf{O} \rightarrow \mathcal{P}(\mathsf{O})$.

A database d over a schema s is a *model* of s if it satisfies the following conditons:

1. if c is a sub-class of c' in s, then $d(c) \subseteq d(c')$;
2. if p is a sub-property of p' in s, then $d(p) \subseteq d(p')$;
3. if c is a domain of property p in s, then $\mathit{def}(d(p)) \subseteq d(c)$;
4. if c is a range of property p in s, then $\mathit{range}(d(p)) \subseteq d(c)$.

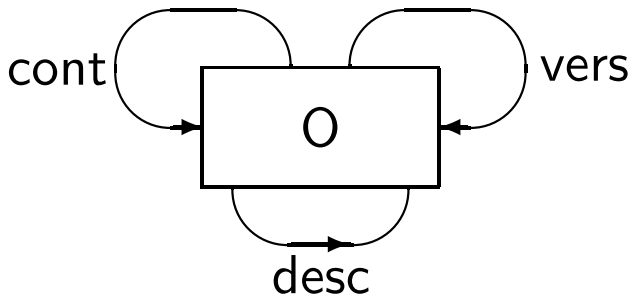
Digital libraries

A digital library consists of three parts:

- the DL *repository*, which holds the content of the DL, as a set of structured objects, their versions and their descriptions;
- the DL *catalog*, which holds the descriptions of the DL content, and
- the DL *knowledge base*, which holds the knowledge enriching objects and their descriptions.

The DL Repository

The DL repository is a database over the repository schema:



Content

We define *content* over O to be a set-valued function cont on O :

$$\text{cont} : \text{O} \rightarrow \mathcal{P}(\text{O})$$

such that for each object $o \in \mathit{def}(\text{cont})$, cont($o$) is a *finite, possibly empty* set of objects.

cont(o) : the *content* of o

def(cont) : the *identifiers*

document: a rendering of some content on a specific device

- we do not exclude the case in which $o \in \text{cont}(o)$
- content is dynamic (in time and space).

Special objects

Given a content function:

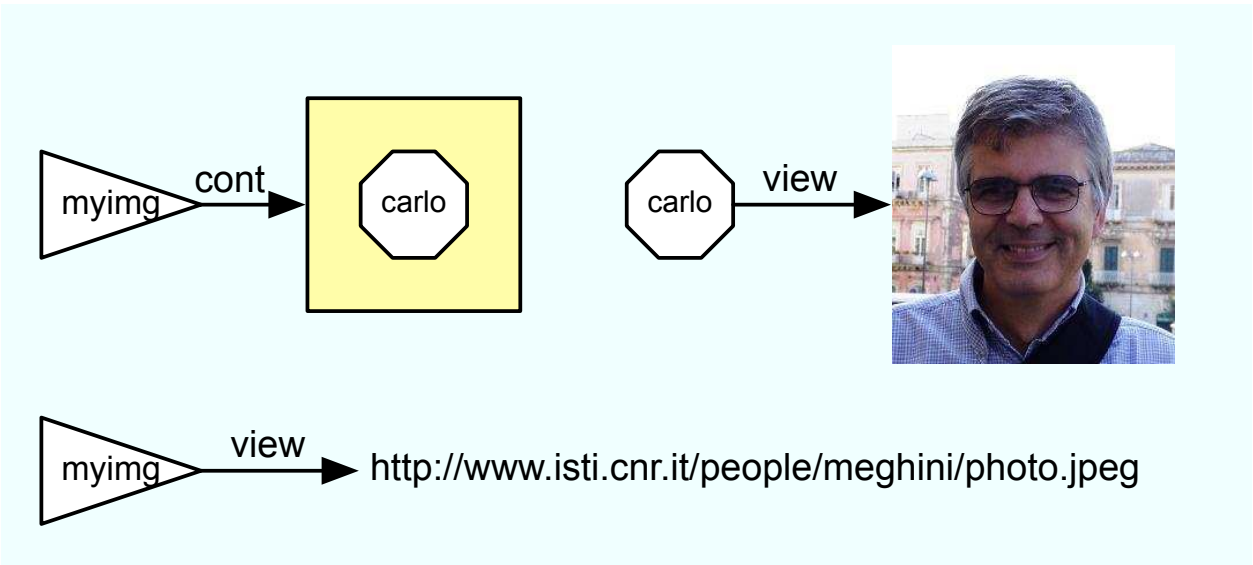
- the inactive objects are those not used currently, but available. They may enter the content function either as identifiers or as elements of content at any later point in time.
- the initial objects: identifiers of *collections.*
  - A special category: objects with empty content
- the terminal objects: "pure" content objects, contributing to the content by their view.

An image identified by a URI

myimg: a digital object (a URI)
view(myimg)=`http://www.isti.cnr.it/people/meghini/photo.jpeg`

carlo: a digital object (an image)
view(carlo)=*a photograph*

cont(myimg)={carlo}



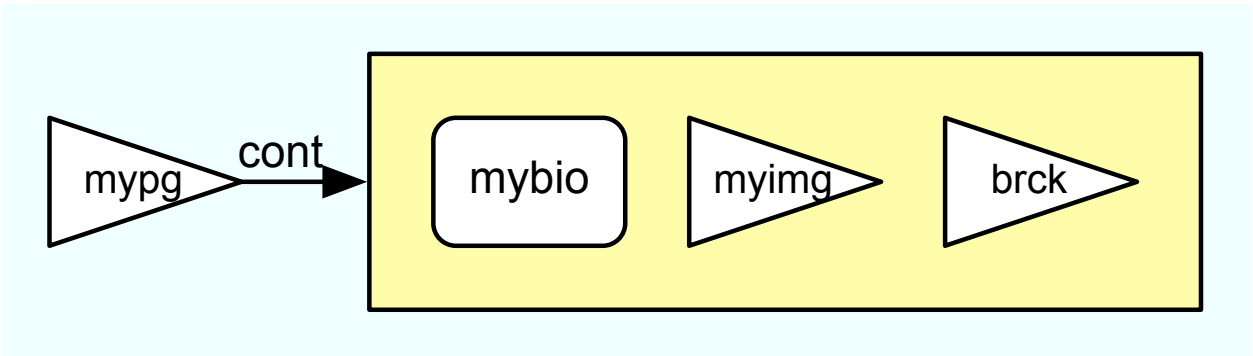
An Web page

mypg: a digital object (a URI)
view(mypg)=`http://www.isti.cnr.it/people/meghini/index.html`

mybio: a digital object (a text)
view(mybio)="Born 1956, married with children, ..."

brck: a digital object (a URI)
view(brck)=`http://www.bricksfactory.org`

cont(mypg)={mybio,myimg,brck}



Versions

The user is working on a text, of which he wants to maintain versions:

- folder o
  - file o_1
    - text t_1 (view(t_1) : the initial text)
  - file o_2
    - text t_2 (view(t_2) : the modified text)

We view o as the identifier of our text and o_1 and o_2 as two versions of it.

Which version represents o at any point in time? any of the two, depending on context.

The versions of o are alternatives for o, not necessarily its evolution in time.

The *versions* over O :

$$\text{vers} : \mathsf{O} \rightarrow \mathcal{P}(\mathsf{O})$$

such that for each object $o \in \mathit{def}(\text{vers})$, vers($o$) is a finite, possibly empty set of objects not containing o.

vers(o) : the *versions* of o.

Relationship with the Web architecture

The web architecture is based on three fundamental notions: *resource*, *representation* and *identifier.*

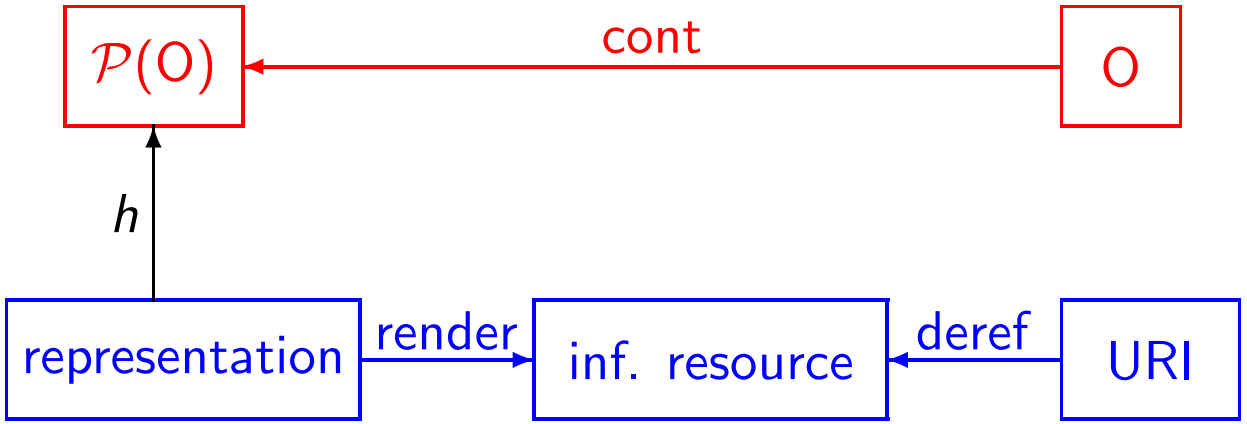
- A resource "can be anything that has identity".
  - An *information resource* is a resource all of whose "essential characteristics can be conveyed in a message".
- A representation is "data that encodes information about resource state".
- An identifier is "an object that can act as a reference to something that has identity". The Web uses a single global identification system: the Uniform Resource Identifiers (URI).

A resource is obtained by *de-referencing* its URI, which for HTTP URIs implies *rendering* one of its representations.

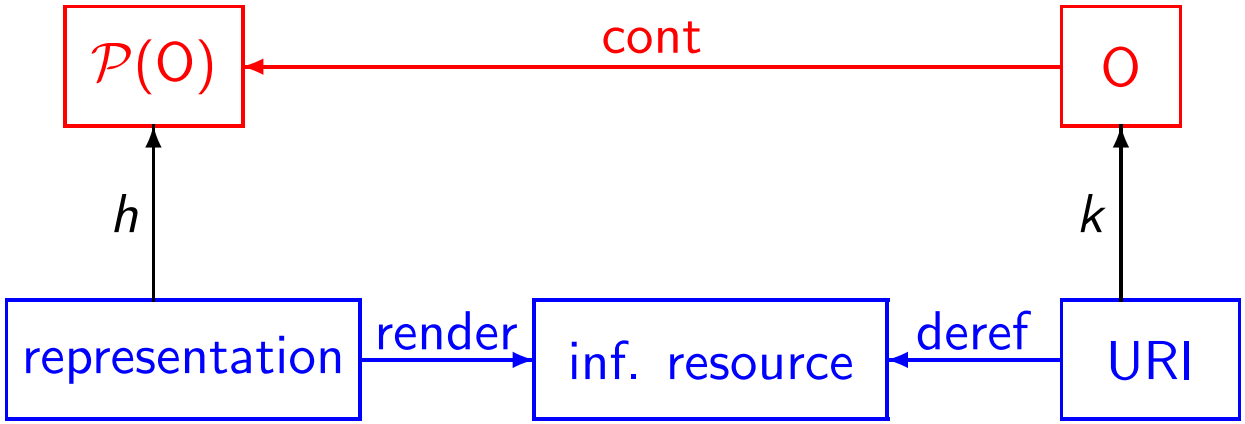

$\mathcal{P}(O)$
cont
O


$\mathcal{P}(O)$
cont
O

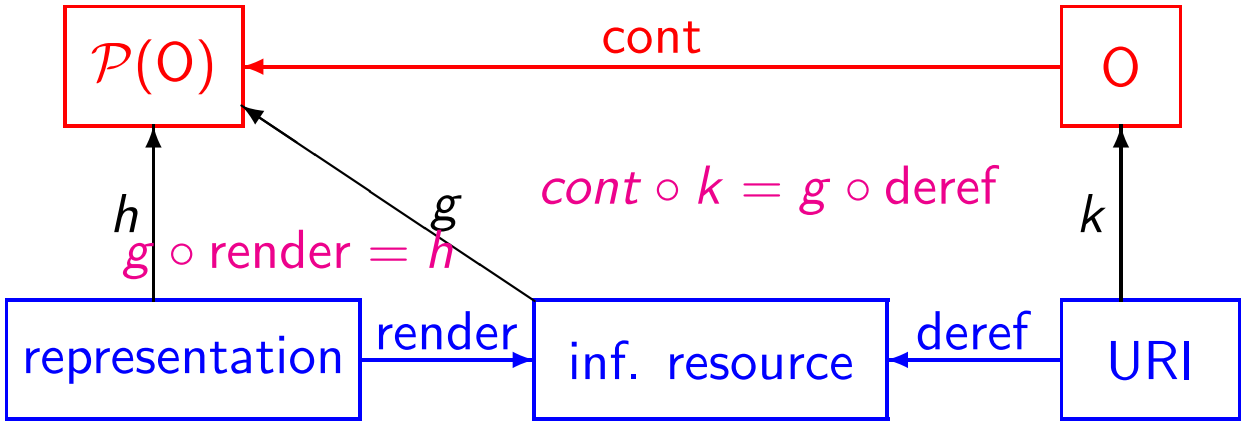

representation
render
inf. resource
deref
URI



- h associates each representation to the set of objects it contains



- h associates each representation to the set of objects it contains
- k associates each URI to an identifier, 1:1



- h associates each representation to the set of objects it contains
- k associates each URI to an identifier, 1:1

Given h and k, there is a unique g which satisfies the constraints.

Descriptions support the interpretation, the discovery, and the management of content.

Descriptions are statements about the DL objects and related entities.

A *description schema s* is a schema whose properties have all the same domain, called the *description class.*

Any database over a description schema is called a *description* database.

In a description database, any instance of the description class is called a *description identifier.*

A description identifier along with the property values defined on it, form a *description.*

description over O :

$$\text{desc} : O \rightarrow \mathcal{P}(O)$$

such that for each object $o \in \mathit{def}(\text{desc})$, desc($o$), is a finite, possibly empty set of description identifiers

desc(o) : the *descriptions* of o.

The Catalog

The Catalog of a DL is a database containing the descriptions of the objects in the Repository of the DL.

Two description schemas are *compatible* if they do not have the same description class.

Given $n \geq 1$ pairwise compatible description schemata $s_1, \ldots, s_n$, the *schema catalog* s over $s_1, \ldots, s_n$ is the pointwise union of the description schemata.

Given a DL repository r and schema catalog s with description classes $c_1, \ldots, c_n$, a catalog cat for r is a database over s, such that:

1. all description identifiers occurring in the repository also occur in the catalog, *i.e.* $range(r(\text{desc})) \subseteq \bigcup_{i=1}^{n} cat(c_i)$;
2. the same description identifier cannot be in two different description schemas, *i.e.* $cat(c_i) \cap cat(c_j) = \emptyset$ for all $1 \leq i, j \leq n$ and $\neq j$.

For instance, the description of an object representing the painting Monna Lisa in the DL repository might be identified by o and consist of the following (property: value) pairs:

- creator: Leonardo da Vinci
- type: oil painting
- creation: event123

where event123 is an object representing the event that brought Monna Lisa into existence.

The DL knowledge base

Roughly speaking, the DL knowledge base is a database consisting of knowledge about the values in the descriptions, or any other information accessible through those values.

For instance, event123 occurring in the previous description of Monna Lisa could be described in the DL KB as an instance of a class Event, with the following property values:

- actor: Leonardo da Vinci
- time period: 1503 - 1506
- place: 7003163

where 7003163 identifies Florence in the Getty Thesaurus of Geographic Names.

If "oil painting" is a class in the KB schema and a sub-class of "painting", this will further enrich the description of Monna Lisa.

The KB schema can be thought of as the union of the schemas used for enriching the KB (like in the previous example).

These schemas can range from very simple, such as classification schemas, subject heading systems, or taxonomies, to very articulated, such as for instance the CIDOC CRM.

Conclusions and future work

We have the main elements of a DL model, compliant with the web architecture.

Next steps:

- query language
- data manipulation language
- implementation

Thank you!

Any question?