XML

Dario COLAZZO

Plan

- Intro Web & XML
- Requête XML : XPath
- Programmation XML : DOM & SAX

Intro Web & XML

XML

XML (eXStensible Markup Language):

format de représentation de données semi-structurées

standardisé par le W3C (http://www.w3.org)

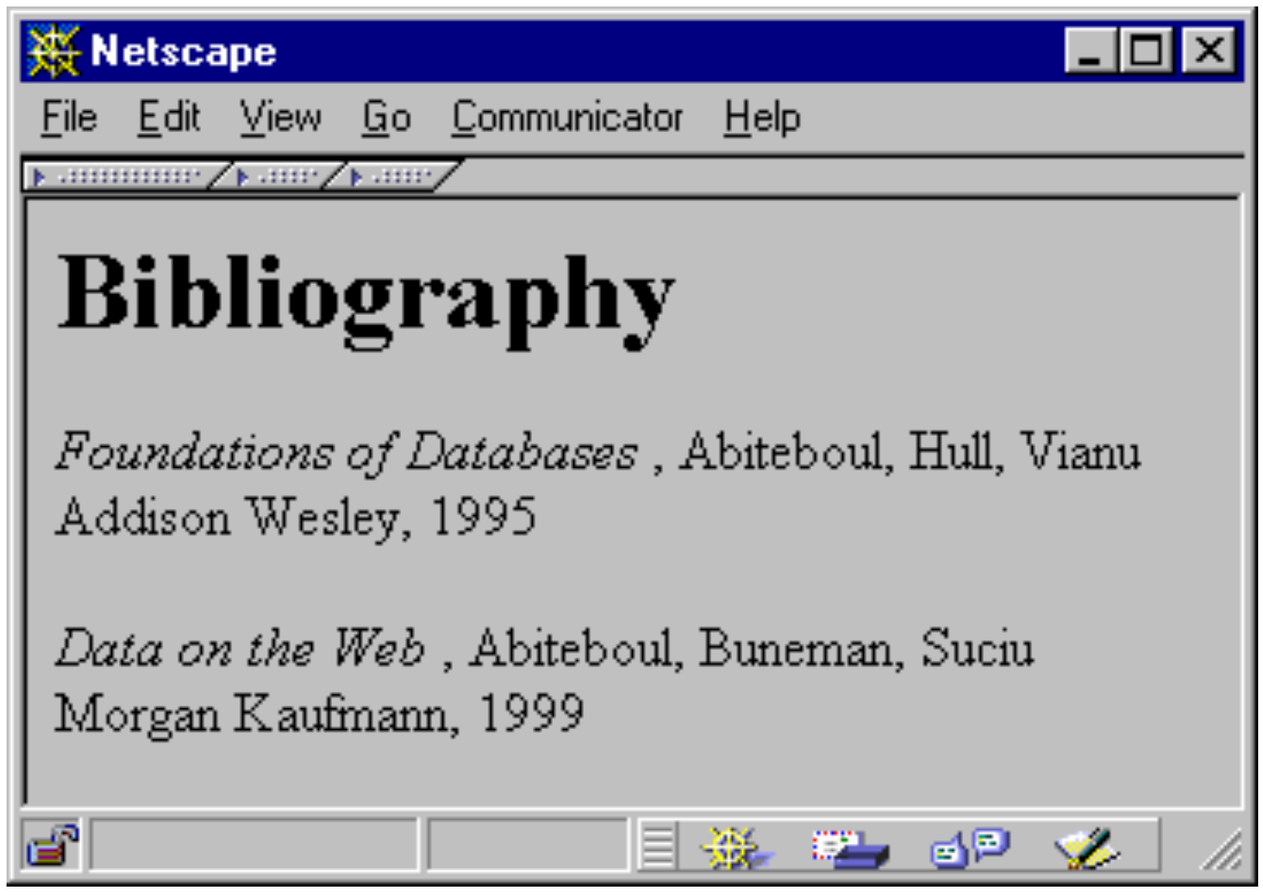
indépendant de la plateforme utilisé

Souvent présenté comme le successeur de HTML, en réalité

HTML est un langage pour déterminer l'affichage d'information via des pages Web

XML est un meta-langage, permettant de définir des nouveaux langages (dialectes XML), HTML en particulier.

Web : de HTML à XML



HTML décrit le format de présentation via browser

HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
        Abiteboul, Hull, Vianu
        <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
        Abiteboul, Buneman, Suciu
        <br> Morgan Kaufmann, 1999
```

XML

```
<bibliography>
        <book>
                <title> Foundations of Databases </title>
                <author> Abiteboul </author>
                <author> Hull </author>
                <author> Vianu </author>
                <publisher> Addison Wesley </publisher>
                <year> 1995 </year>
                <prix>10</prix>
         </book>
        <book>
                <title> Data on the Web </title>
                <author> Abiteboul </author>
                <author> Buneman </author>
                <author> Suciu </author>
                 <author> Manolescu</author>
                <publisher> Morgan Kaufmann </publisher>
                <year> 1999</year>
         </book>
</bibliography>
```

Avantages

Format standard de représentation et partage de données sur le Web

Il est basé sur une représentation textuelle et *balisée* (comme HTML)
==> facilement transférable sur le Web via HTTP

Plusieurs systèmes disponibles pour la création, analyse et gestion de documents XML

Largement utilisée pour l'exchange de données entre applications

Représentation de donnée dans plusieurs contextes 'hors du Web'

- MS World, Excel, etc.
- données scientifique
- fichiers de log

Elements

```
<bibliography>
      <book>   <title> Foundations… </title>
                  <author> Abiteboul </author>
                  <author> Hull </author>
                  <author> Vianu </author>
                  <publisher> Addison Wesley </publisher>
                        <year> 1995 </year>
      </book>
      …
</bibliography>
```

Attributs

```
<bibliography>
      <book   price=`120‘  label=‘fds’>
            <title> Foundations… </title>
            <author> Abiteboul </author>
            <author> Hull </author>
            <author> Vianu </author>
            <publisher> Addison Wesley </publisher>
            <year> 1995 </year>
       </book>
      <book   price=`110‘  label=‘dw’>
            <title> Data on the Web </title>
            <author> Abiteboul </author>
            <author> Buneman </author>
            <author> Suciu </author>
            <publisher> Morgan Kaufmann </publisher>
            <year> 1999</year>
            <cite   ref=‘fds’   ></cite>
       </book>
</bibliography>
```

Documents bien formés

Un document XML est bien formé s'il respecte des propriétés syntaxiques. En particulier :

Il contient les composantes nécessaires:

- un préambule, par ex. <?xml version="1.0" encoding="ISO-8859-1"?>
- un seul élément racine

Les tags sont balancés :

- tout tag ouvert doit avoir un tag fermé correspondant
- les tags sont correctement imbriqués: le dernier tag ouvert est le premier à être fermé

<TAG1><TAG2><TAG3>...</TAG3></TAG2></TAG1> OUI

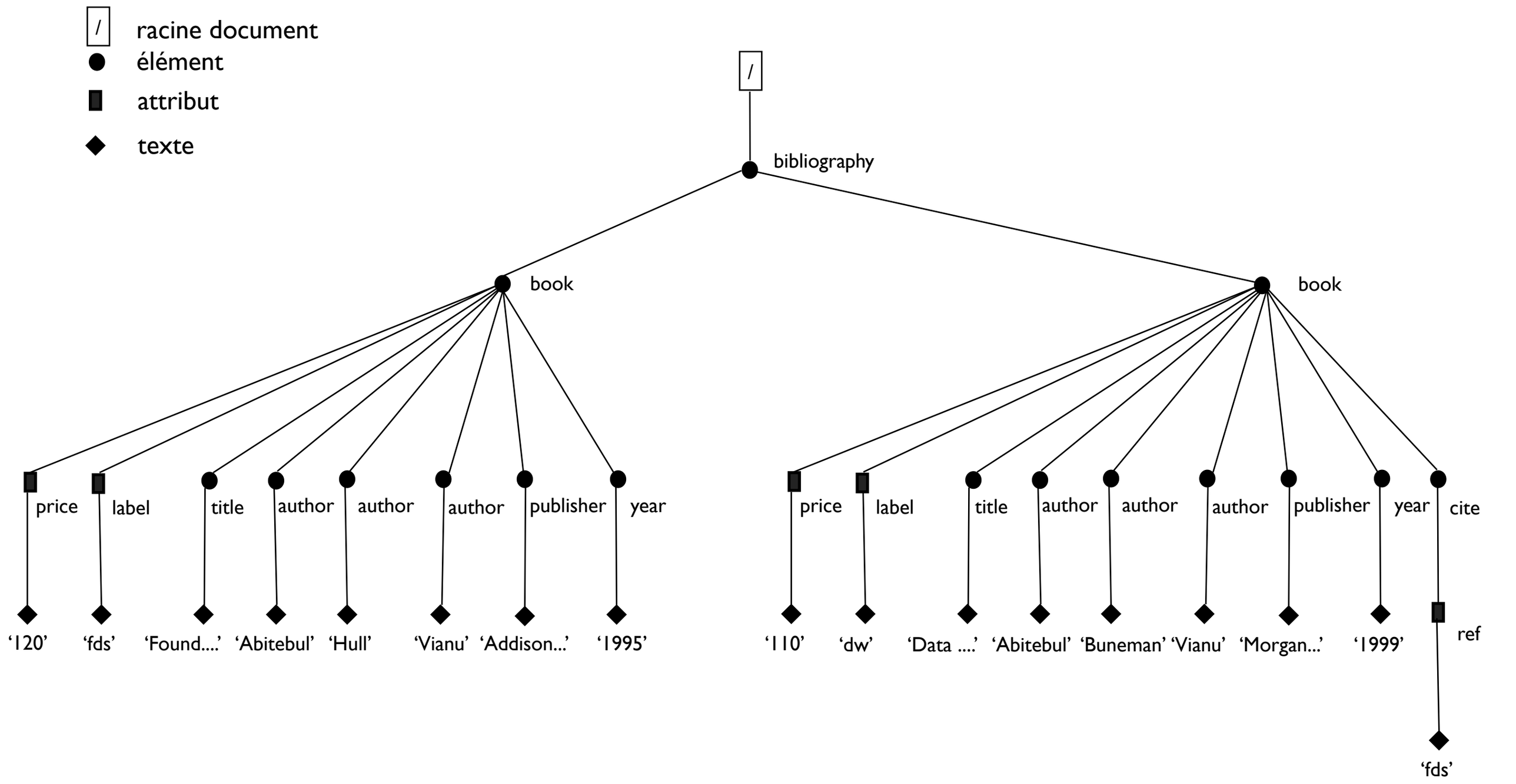
<TAG1><TAG2><TAG3>...</TAG2></TAG3></TAG1> NON

Document BF

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bibliography>
      <book   price=`120‘  label=‘fds’>
            <title> Foundations… </title>
            <author> Abiteboul </author>
            <author> Hull </author>
            <author> Vianu </author>
            <publisher> Addison Wesley </publisher>
            <year> 1995 </year>
       </book>
      <book   price=`110‘  label=‘dw’>
            <title> Data on the Web </title>
            <author> Abiteboul </author>
            <author> Buneman </author>
            <author> Suciu </author>
            <publisher> Morgan Kaufmann </publisher>
            <year> 1995 </year>
            <cite   ref=‘fds’  ></cite>
       </book>
</bibliography>
```

Sous forme d'arbre



DTD

Nous pouvons donner la spécification de notre langage de représentation de données bibliographique.

Via une DTD (Document Type Définition)

Un document D est *valide* par rapport à une DTD si la structure de D respecte la DTD

Plusieurs programmes existent pour vérifier la validation

DTD : ensemble de declarations spécifiant la structure d'un type d'élément ou d'attribut.

Type élément : tag + structure contenu (expression régulière)

Type attribut : tag + type valeur + contrainte

Type element

Type d'élément : tag + **structure** contenu (expression régulière)

<!ELEMENT tag structure>

La structure décrit le contenu via une expression régulière, par exemple :

A, B : un A suivi d'un B

A, B* : un A suivi de 0 ou plusieurs B

A, B?, C+ : un A, suivi de 0 ou 1 B, suivi d'au moins un C

A, (B | C)* : un A suivi de plusieurs B ou C dans n'importe quel ordre

Le type #PCDATA (chaine de caractères) peut être utilisé dans la structure :

(#PCDATA | A | B)* décrit un contenu mixte : du texte simple mélangé avec des éléments ayant tag A ou B, dans n'importe quel ordre

Aussi, la structure peut être soit EMPTY soit ANY

DTD pour ‘bibliography’

Déclaration des types élément

<!ELEMENT bibliography (book*)>

<!ELEMENT book (title, (author+ | editor+), publisher, year?, cite*)>

<!ELEMENT author (#PCDATA)>

<!ELEMENT editor (#PCDATA)>

<!ELEMENT publisher (#PCDATA)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT cite EMPTY>

Type attribut

Type d'attribut : tag + type contenu + contrainte

CDATA
ID
IDREF
{s1,....,sn}

#REQUIRED
#IMPLIED

.....

<!ELEMENT book (title, (author+ | editor+), publisher, year?, cite*)>

<!ATTLIST book **price** CDATA #IMPLIED>

<!ATTLIST book **label** ID #REQUIRED>

.....

<!ELEMENT cite EMPTY>

<!ATTLIST cite **ref** IDREF #IMPLIED>

DTD pour ‘bibliography’

<!ELEMENT bibliography (book*)>

<!ELEMENT book (title, (author+ | editor+), publisher, year?, cite*)>

<!ATTLIST book **price** CDATA #IMPLIED>

<!ATTLIST book **label** ID #REQUIRED>

<!ELEMENT author (#PCDATA)>

<!ELEMENT editor (#PCDATA)>

<!ELEMENT publisher (#PCDATA)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT cite EMPTY>

<!ATTLIST cite **ref** IDREF #IMPLIED>

Lier une DTD à un document

DTD externe

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE bibliography SYSTEM "biblio.dtd">

<bibliography>
        ........
<bibliography>
```

DTD interne

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE bibliography [

<!ELEMENT bibliography  (book* )>

<!ELEMENT book  (title,  (author+ | editor+ ), publisher, year?, cite* )>

.............

]>

<bibliography>.......
        ........
<bibliography>
```

XPath

XPath

Le langage permet de désigner un ou plusieurs nœuds dans un document XML, à l'aide d'expressions de chemin.

XPath est un sous langage de XSLT (et XQuery)

Exemples :

XSLT

```
<xsl:value-of select="/bibliography/book[1]/@price">
```

XQuery

```
for $x in $doc/bibliography//title/text()
return
    <titre>$x</titre>
```

Expressions XPath

Une expression XPath :

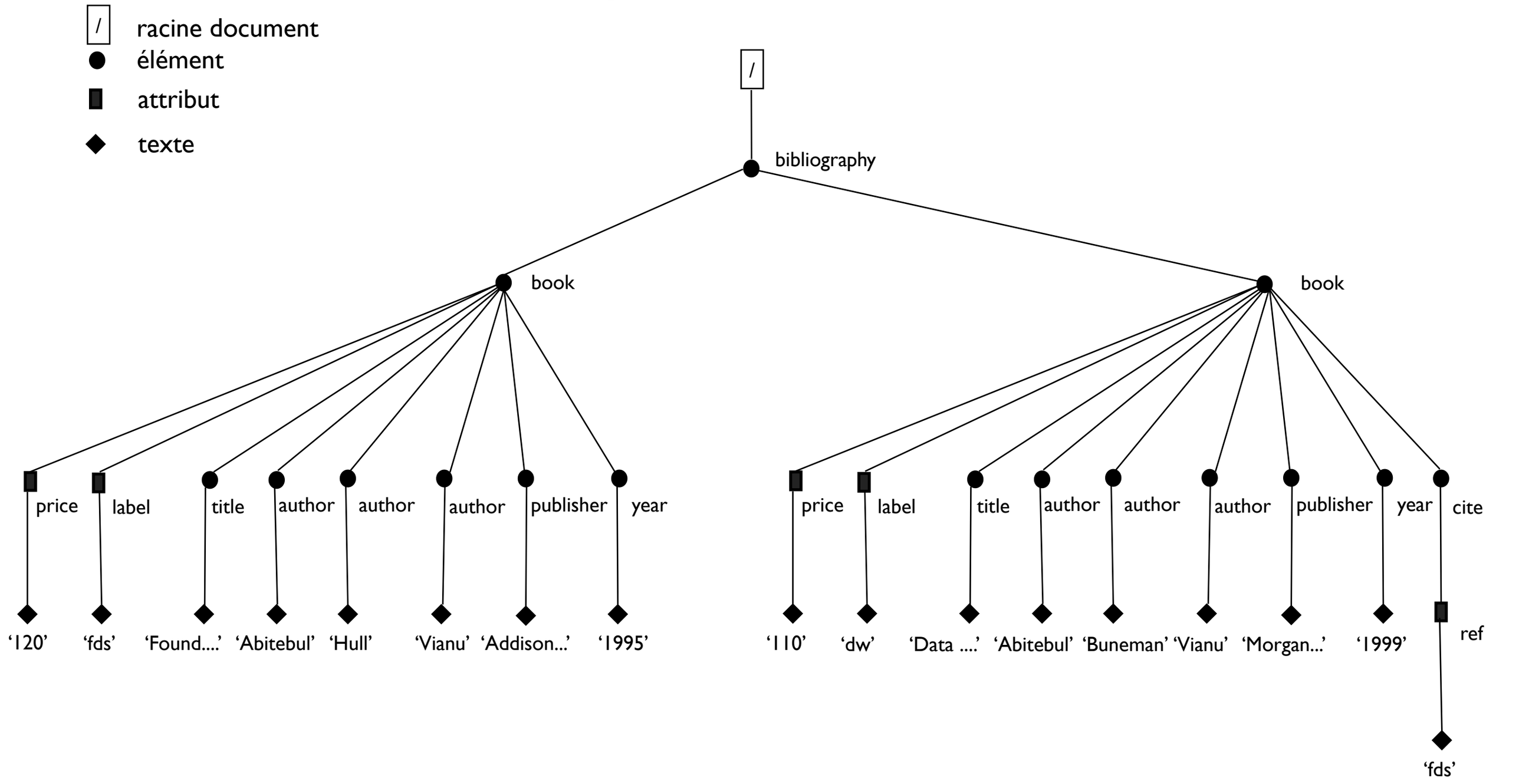
- s'évalue en fonction d'un nœud contexte
- désigne un ou plusieurs chemins dans l'arbre à partir du nœud contexte
- a pour résultat :
  - un ensemble de nœuds
  - ou une valeur, numérique, booléeenne ou alphanumérique

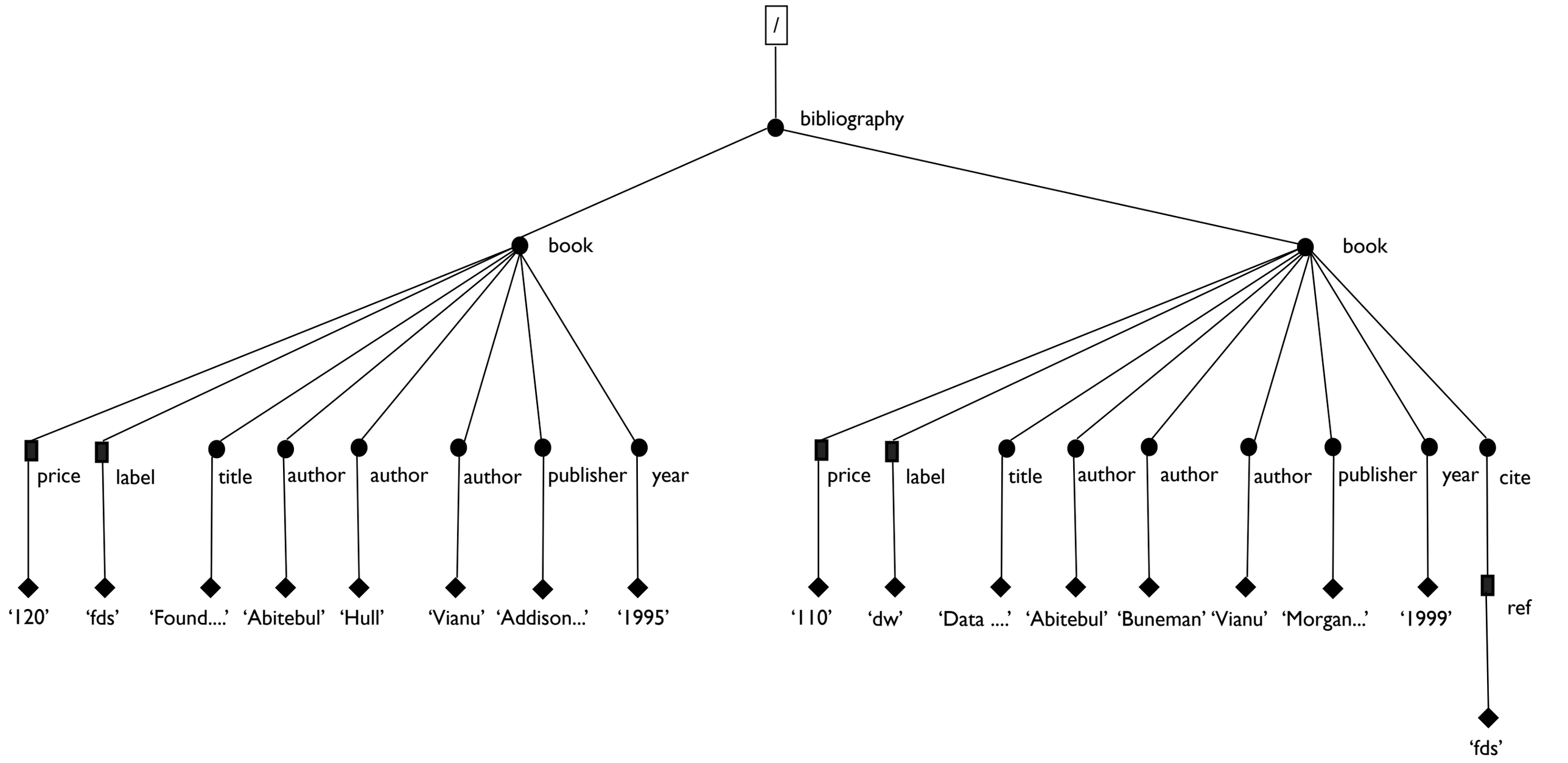
Exemple de reference

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bibliography>
      <book   price=`120‘  label=‘fds’>
            <title> Foundations… </title>
            <author> Abiteboul </author>
            <author> Hull </author>
            <author> Vianu </author>
            <publisher> Addison Wesley </publisher>
            <year> 1995 </year>
       </book>
      <book   price=`110‘  label=‘dw’>
            <title> Data on the Web </title>
            <author> Abiteboul </author>
            <author> Buneman </author>
            <author> Suciu </author>
            <publisher> Morgan Kaufmann </publisher>
            <year> 1995 </year>
            <cite   ref=‘fds’  ></cite>
       </book>
<bibliography>
```

Exemple de reference




/
bibliography
book
book
price
label
title
author
author
author
publisher
year
'120'
'fds'
'Found....'
'Abitebul'
'Hull'
'Vianu'
'Addison...'
'1995'
price
label
title
author
author
author
publisher
year
cite
'110'
'dw'
'Data ....'
'Abitebul'
'Buneman'
'Vianu'
'Morgan...'
'1999'
ref
'fds'

Requêtes XPath

Une requête XPath est une suite d'étapes :

[/]étape /étape /.../étape

Deux types de requêtes:

chemin absolu /bibliography/book

chemin relatif book/title

Une étape XPath

Une étape : trois composants

axe::filtre[prédicat1][prédicat2]

l'axe : sens de parcours des nœuds

le filtre : type des nœuds qui seront retenus

le(s) prédicat(s) : propriétés que doivent satisfaire les nœuds retenus par axe::filtre

On peut faire une union de chemins : //title | book/@price

Evaluation d'une requête XPath

etape1 /etape2 /..... /etapeN

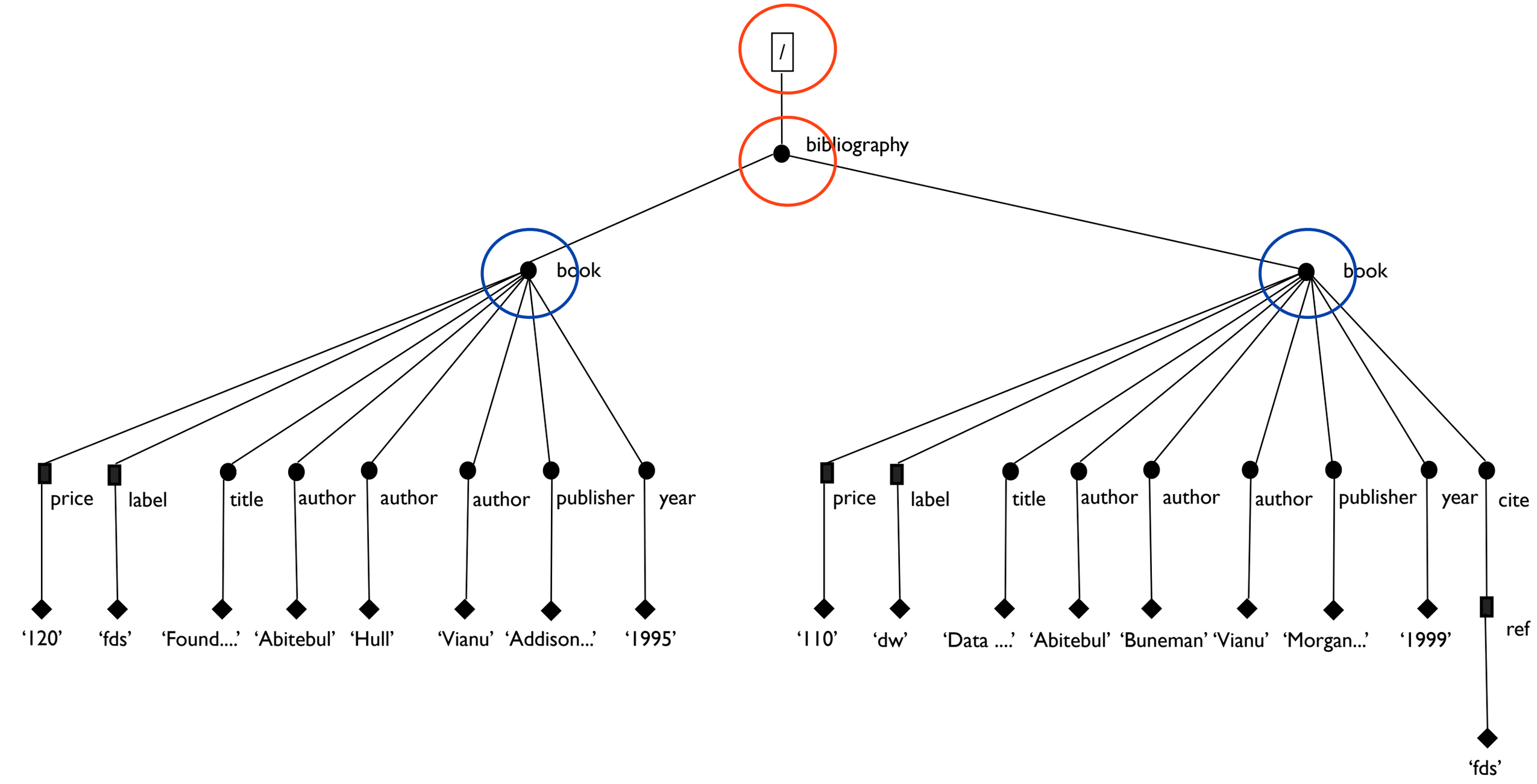
A partir du nœud contexte, on évalue l'étape 1 ; on obtient un ensemble de nœuds ;

on prend alors, un par un, les nœuds de cet ensemble, et on les considère chacun à leur tour comme nœud contexte pour l'évaluation de l'étape 2 ;

à chaque étape, on prend successivement comme nœud contexte chacun des nœuds faisant partie du résultat de l'étape précédente.

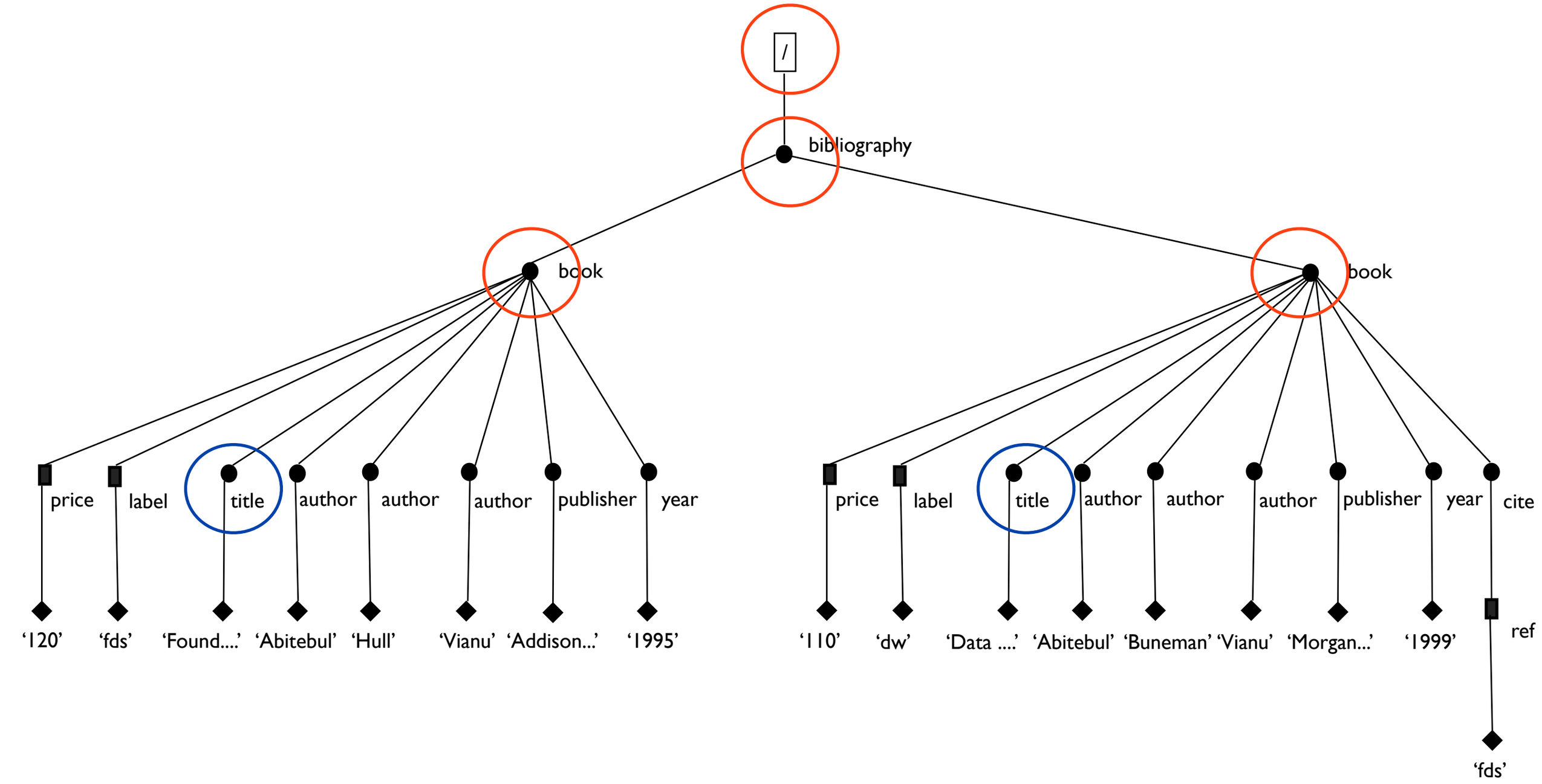
/child::bibliography/child::boook

Syntaxe abrégée : /bibliography/book



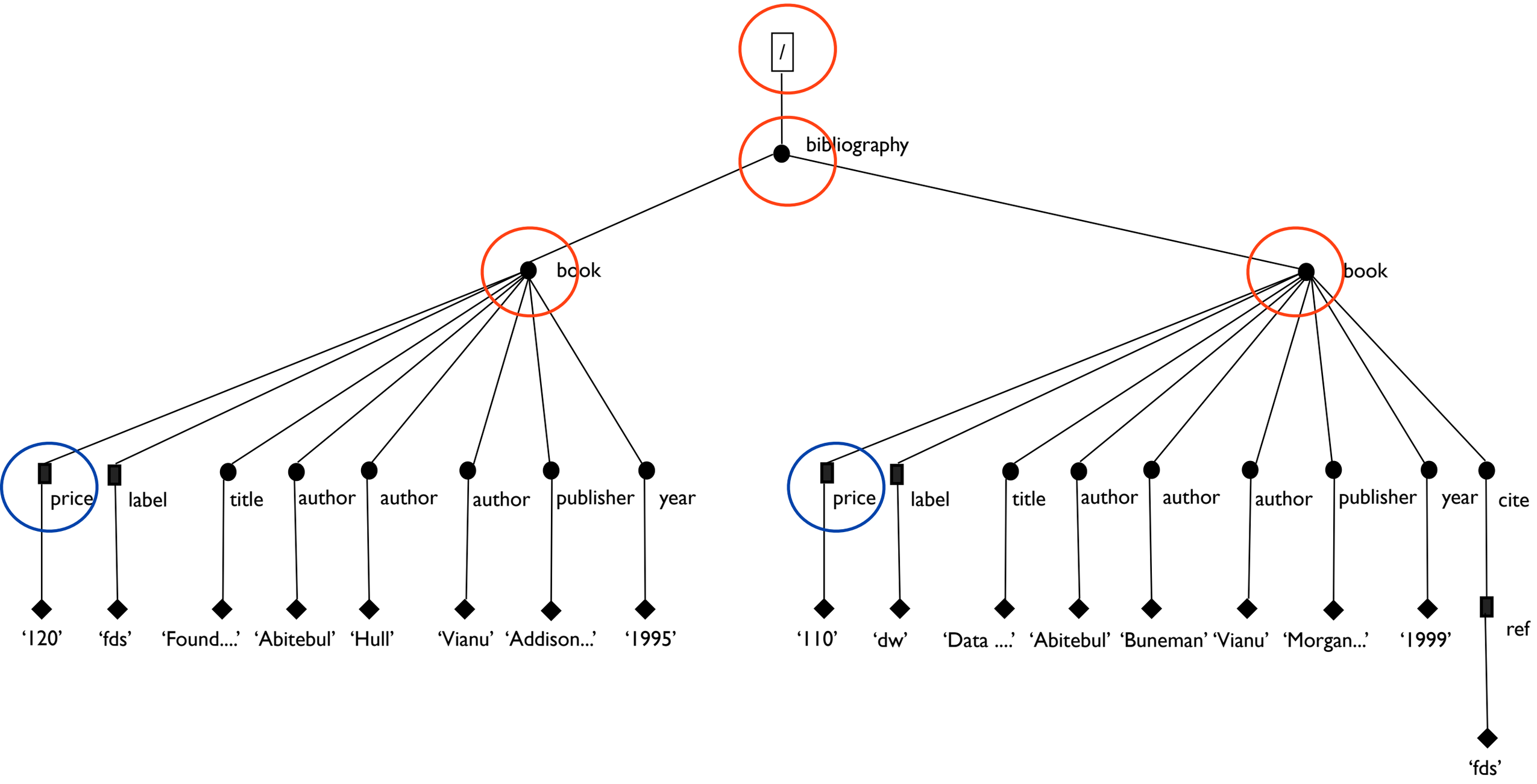
/child::bibliography/child::book/child::title

Syntaxe abrégée : /bibliography/book/title



/child::bibliography/child::book/attribute::price

Syntaxe abrégée : /bibliography/book/title/@price



Contexte d'evaluation

etape1 /etape2 /..... /etapeN

Une étape s'évalue en tenant compte d'un contexte constitué de un nœud contexte, position initiale de l'étape ;

ce nœud fait lui-même partie d'un ensemble obtenu par évaluation de l'étape précédente ;

- on connaît la taille de cet ensemble (fonction *last()*)

- on connaît la position du nœud contexte dans cet ensemble (fonction *position()*)

Les axes XPath

axe::filtre[predicat1]...[predicatN]

Un axe XPath sélectionne un ensemble de nœuds :
en effectuant une navigation à partir du nœud contexte, selon la direction de l'axe ;

Par exemple, l'axe child effectue une navigation vers le bas à partir du nœud contexte, et sélectionne les fils de celui-ci.

Les axes XPath permettent de naviguer verticalement et horizontalement dans l'arbre XML.

Les filtres XPath

axe::**filtre**

Un filtre raffine l'ensemble de nœuds retournés par l'axe

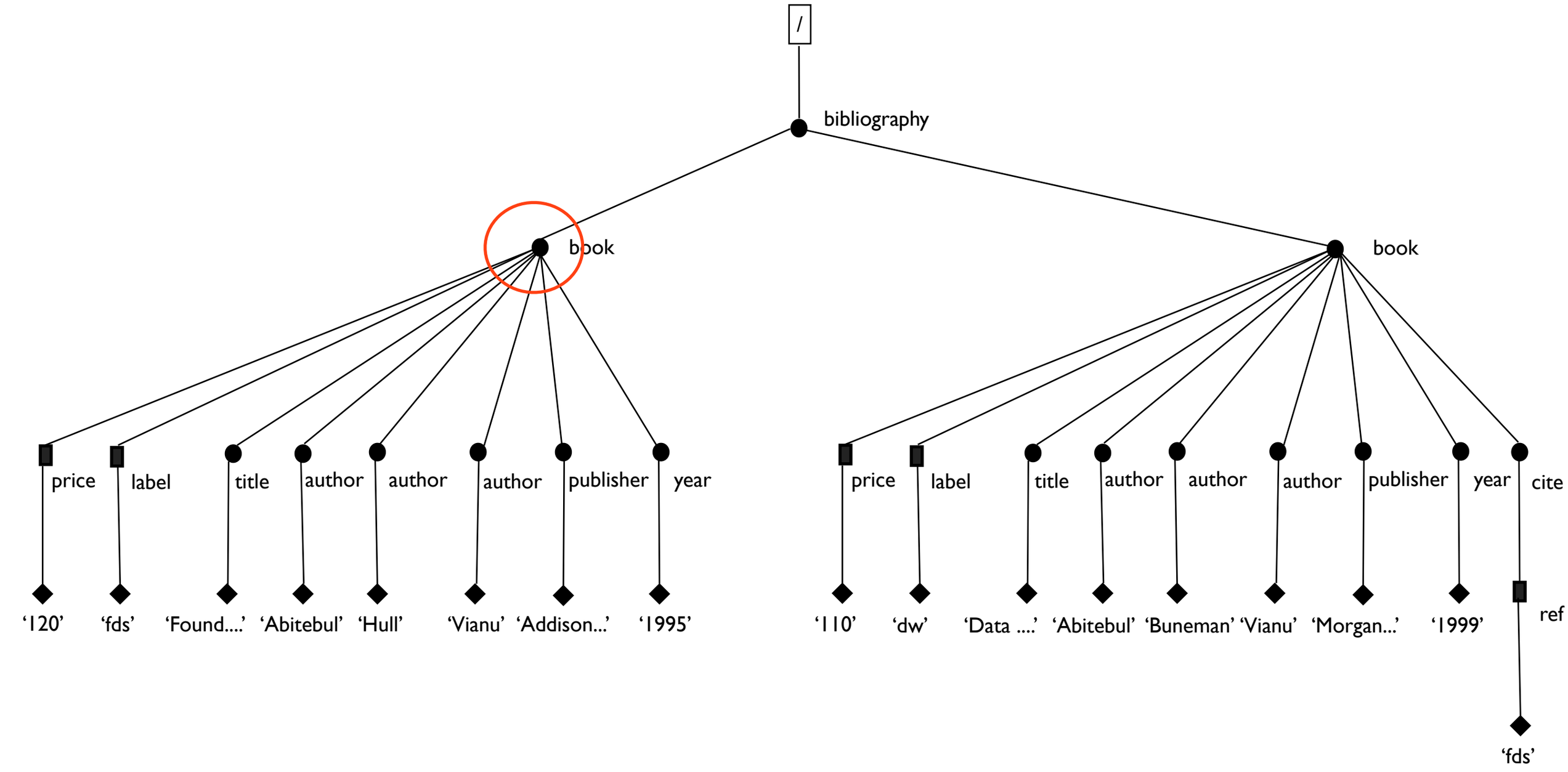
child::book les fils de nœud contexte ayant tag book

child::node() tout les fils du nœud contexte

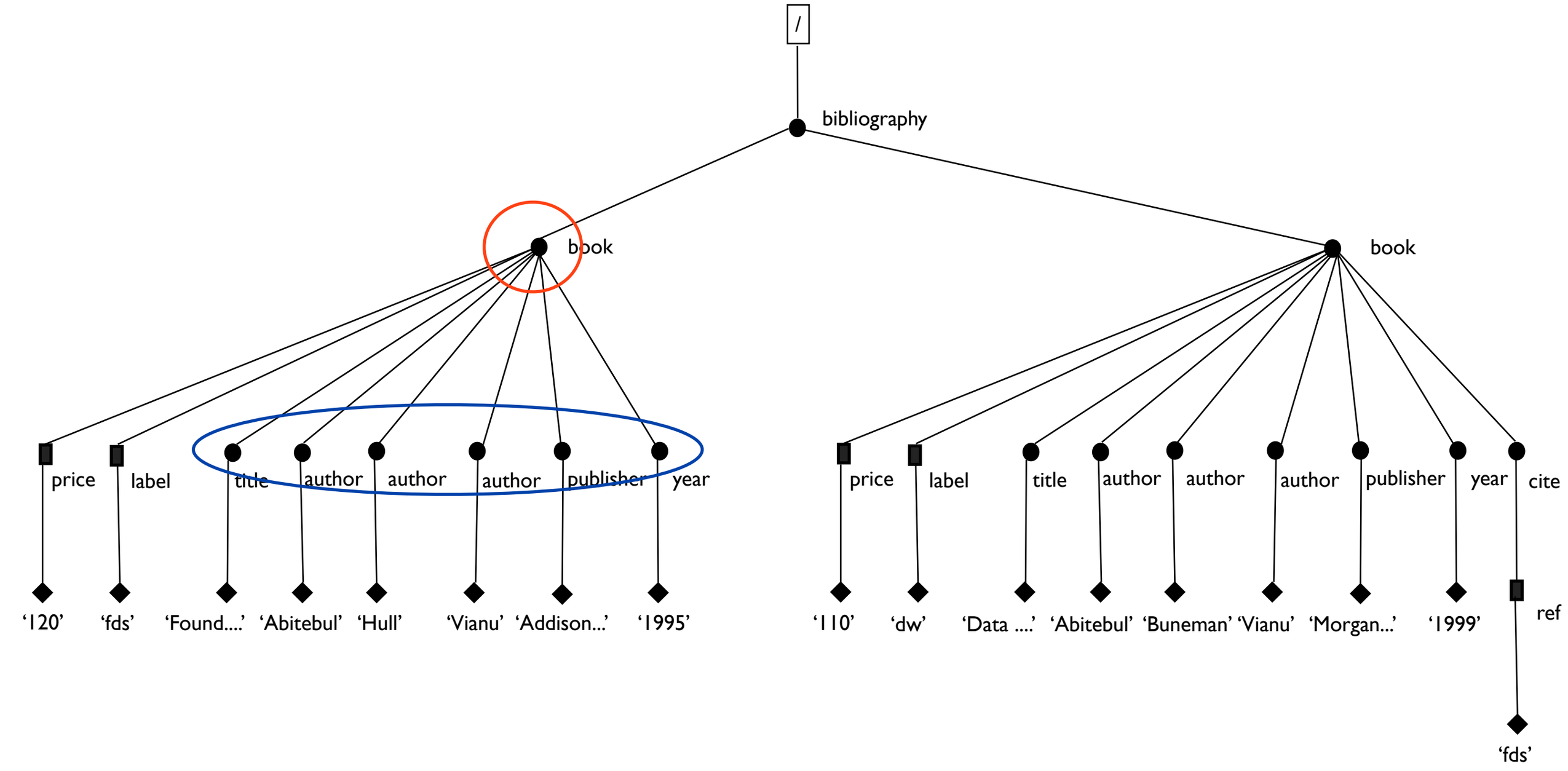
child::* tout les fils élément du nœud contexte

child::text() tout les fils texte du nœud contexte

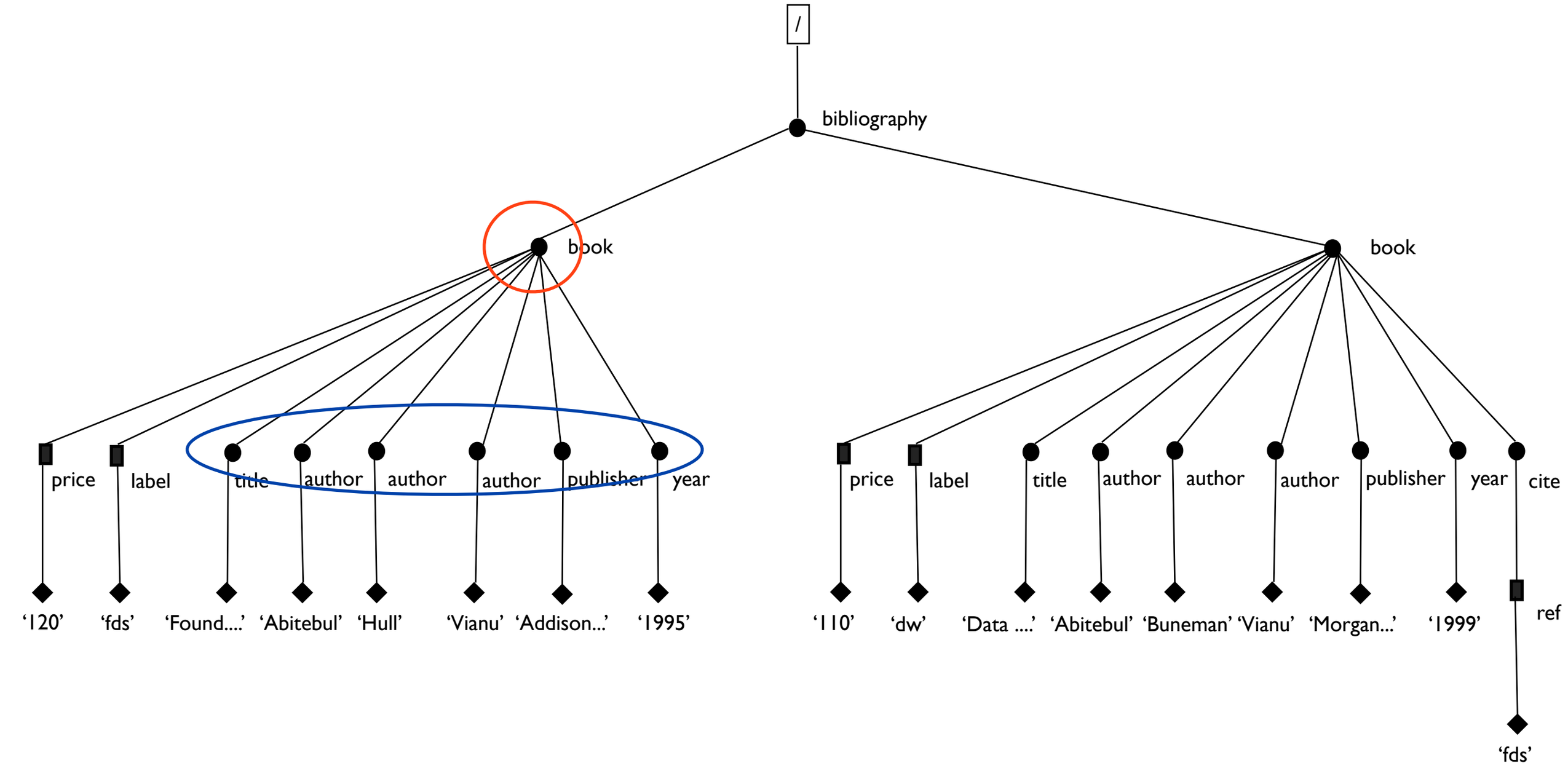
Nœud contexte



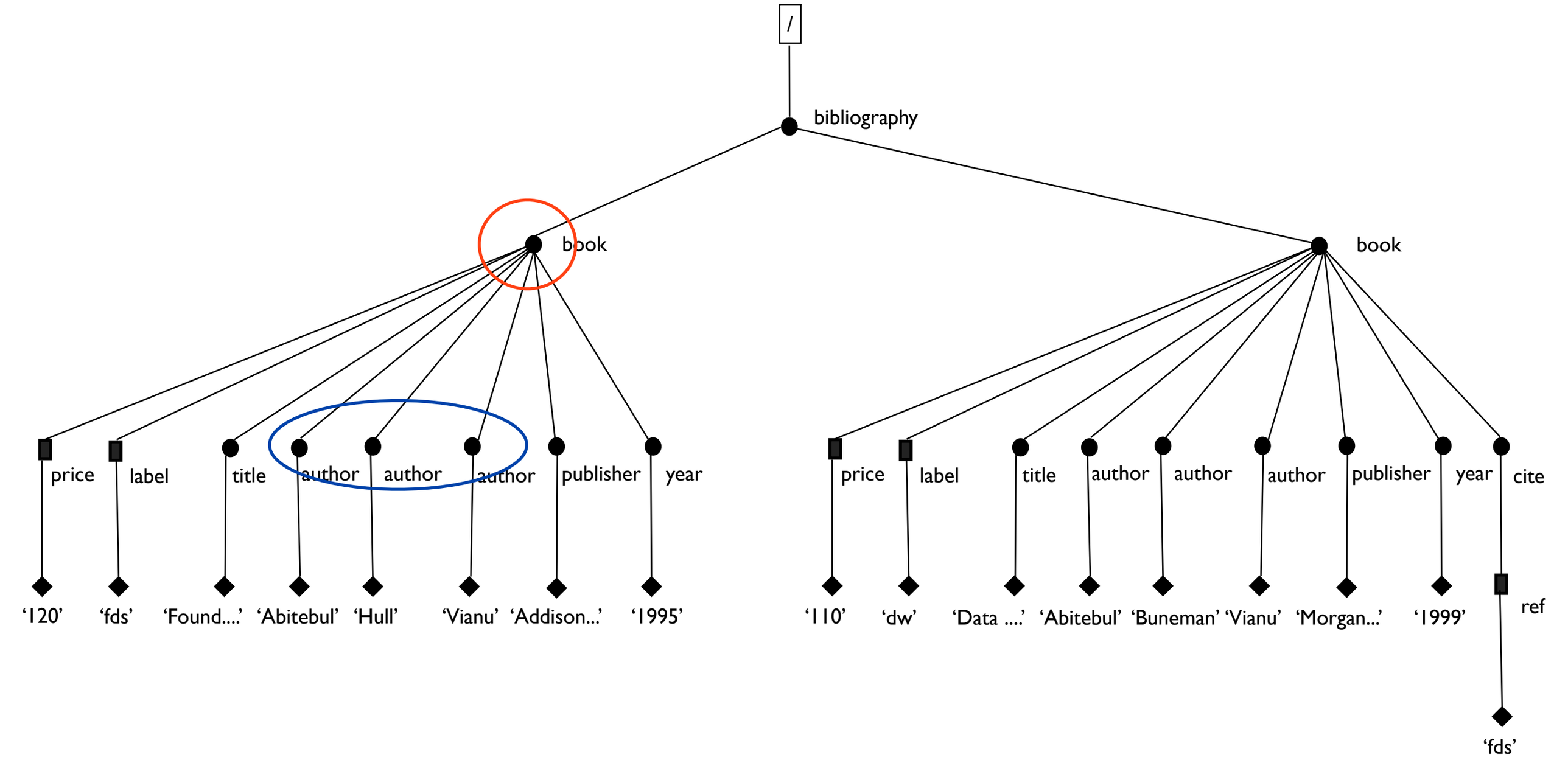
L’axe child


/
bibliography
book
price
label
title
author
author
author
publisher
year
‘120’
‘fds’
‘Found....’
‘Abitebul’
‘Hull’
‘Vianu’
‘Addison...’
‘1995’
book
price
label
title
author
author
author
publisher
year
cite
‘110’
‘dw’
‘Data ....’
‘Abitebul’
‘Buneman’
‘Vianu’
‘Morgan...’
‘1999’
ref
‘fds’

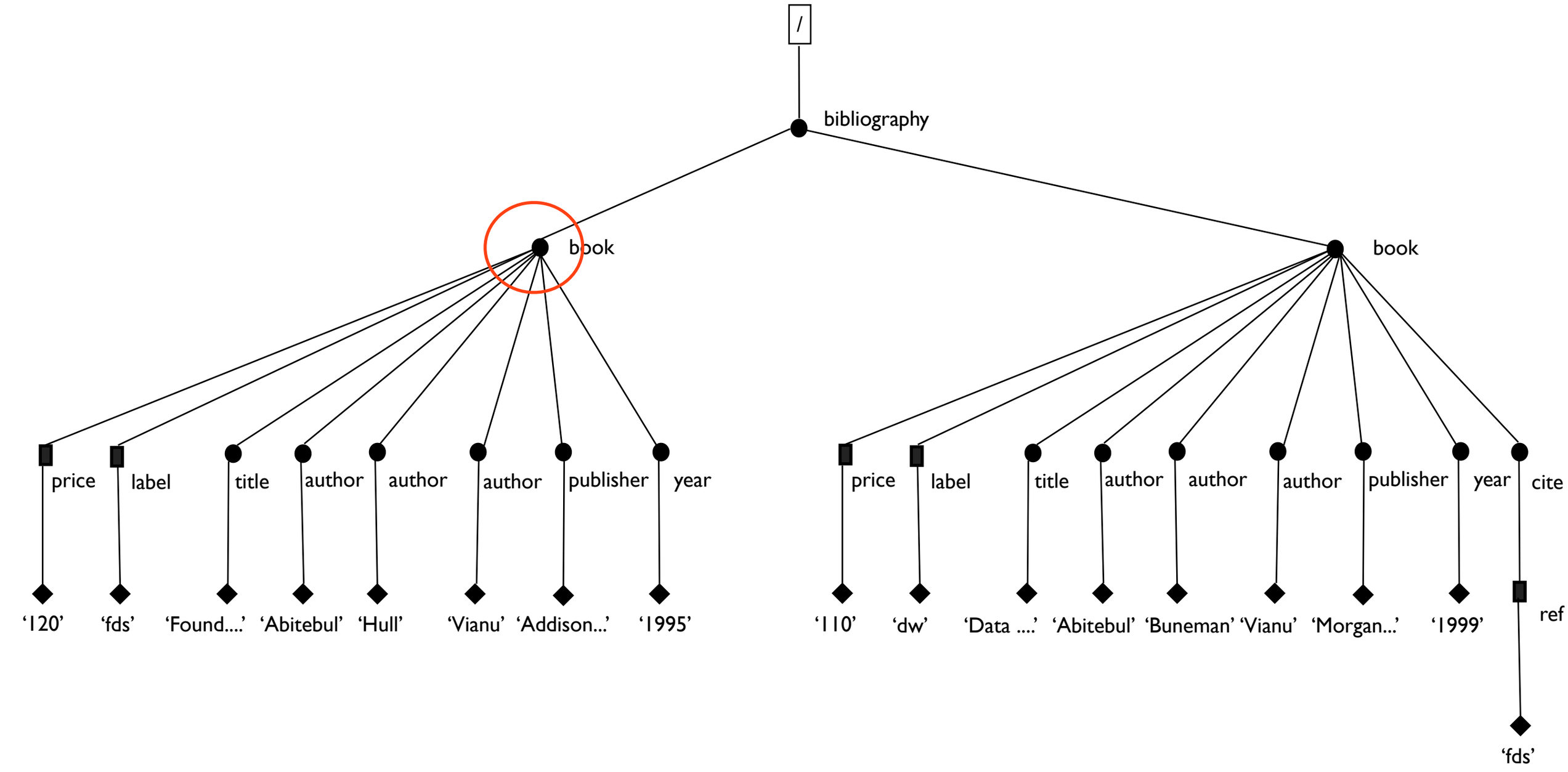
L'étape child::node()



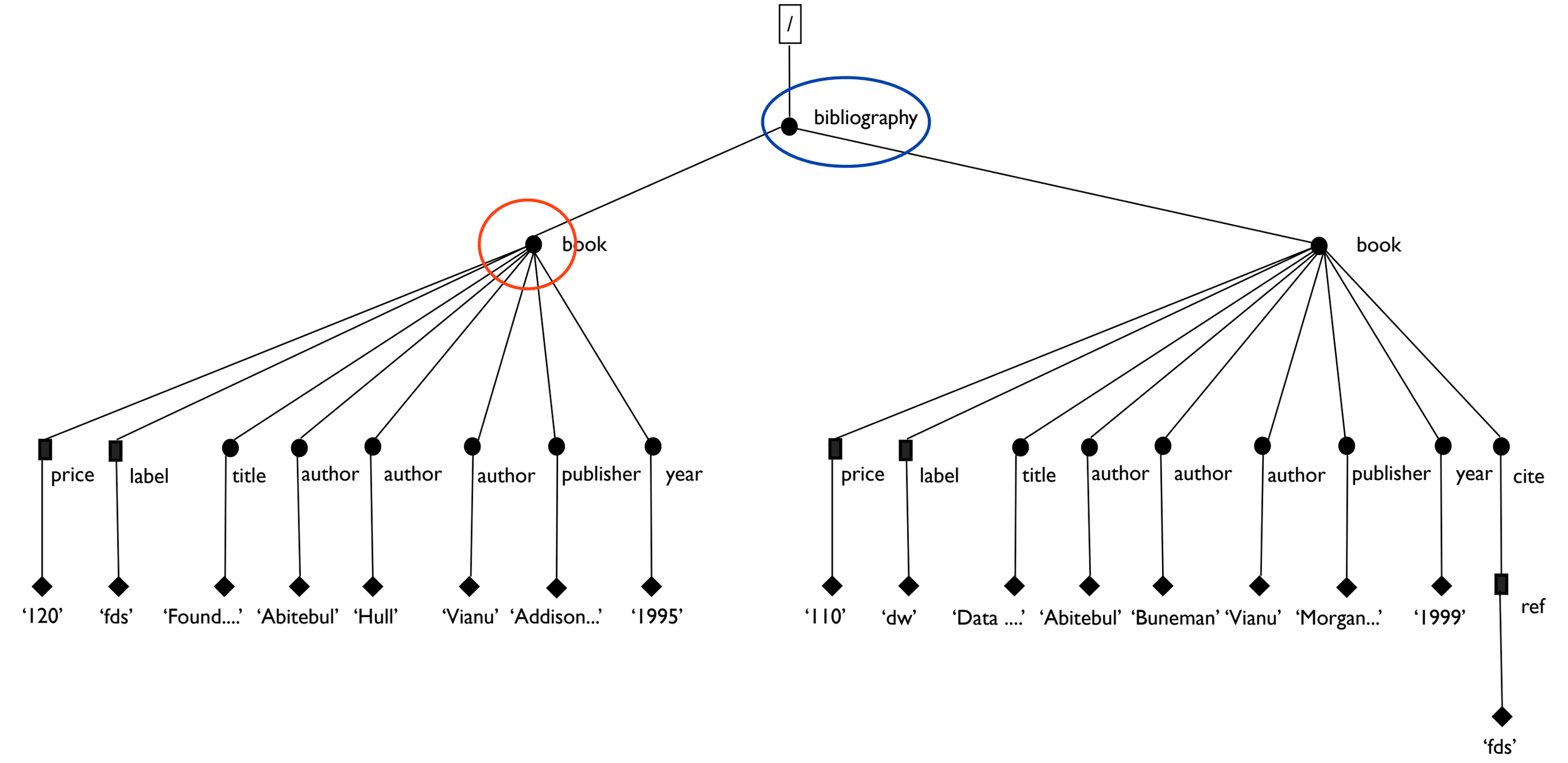
L'étape child::author



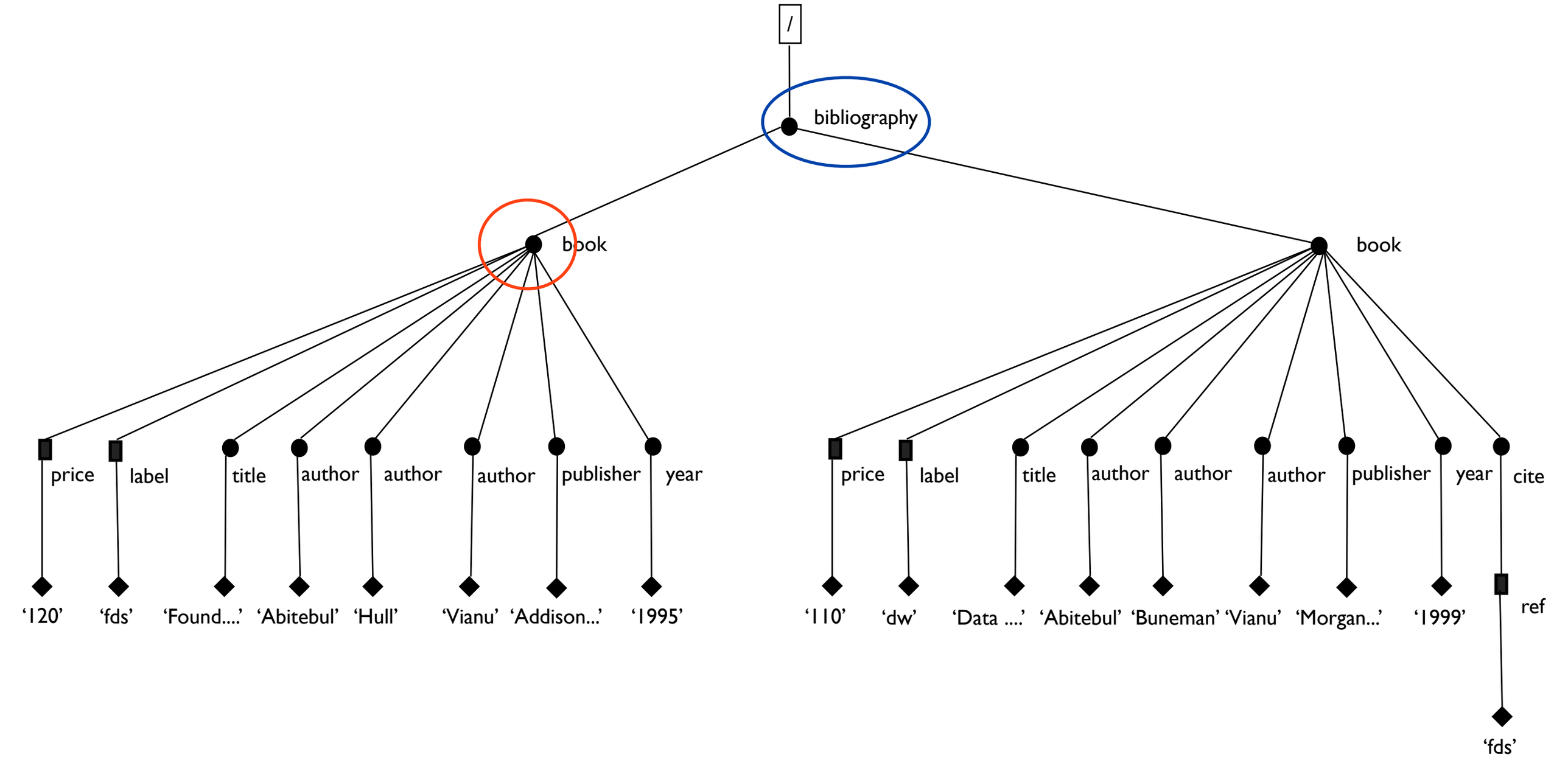
L'étape child::text()



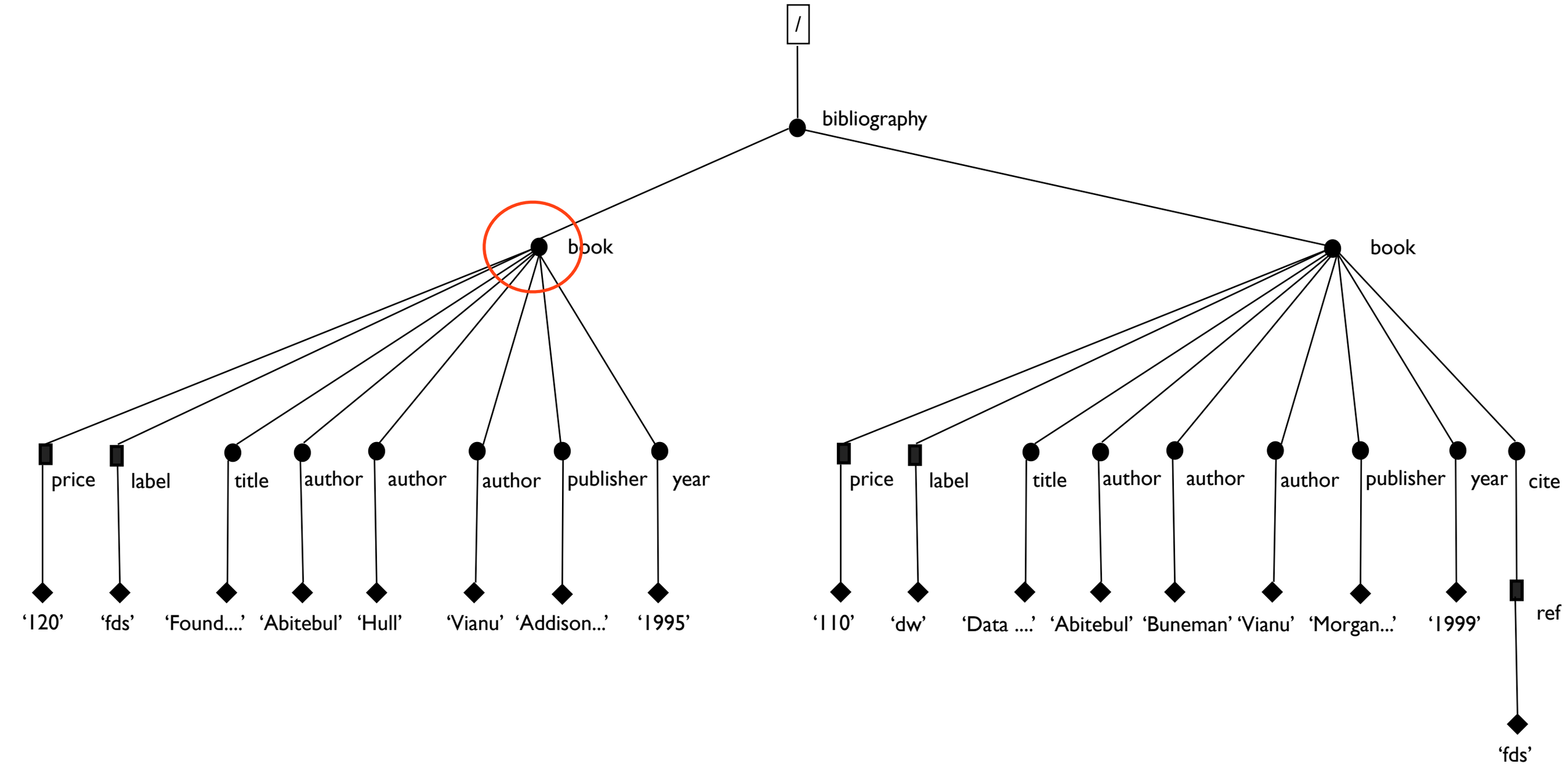
L’axe parent


/
bibliography
book
price
label
title
author
author
author
publisher
year
‘120’
‘fds’
‘Found....’
‘Abitebul’
‘Hull’
‘Vianu’
‘Addison...’
‘1995’
book
price
label
title
author
author
author
publisher
year
cite
‘110’
‘dw’
‘Data ....’
‘Abitebul’
‘Buneman’
‘Vianu’
‘Morgan...’
‘1999’
ref
‘fds’

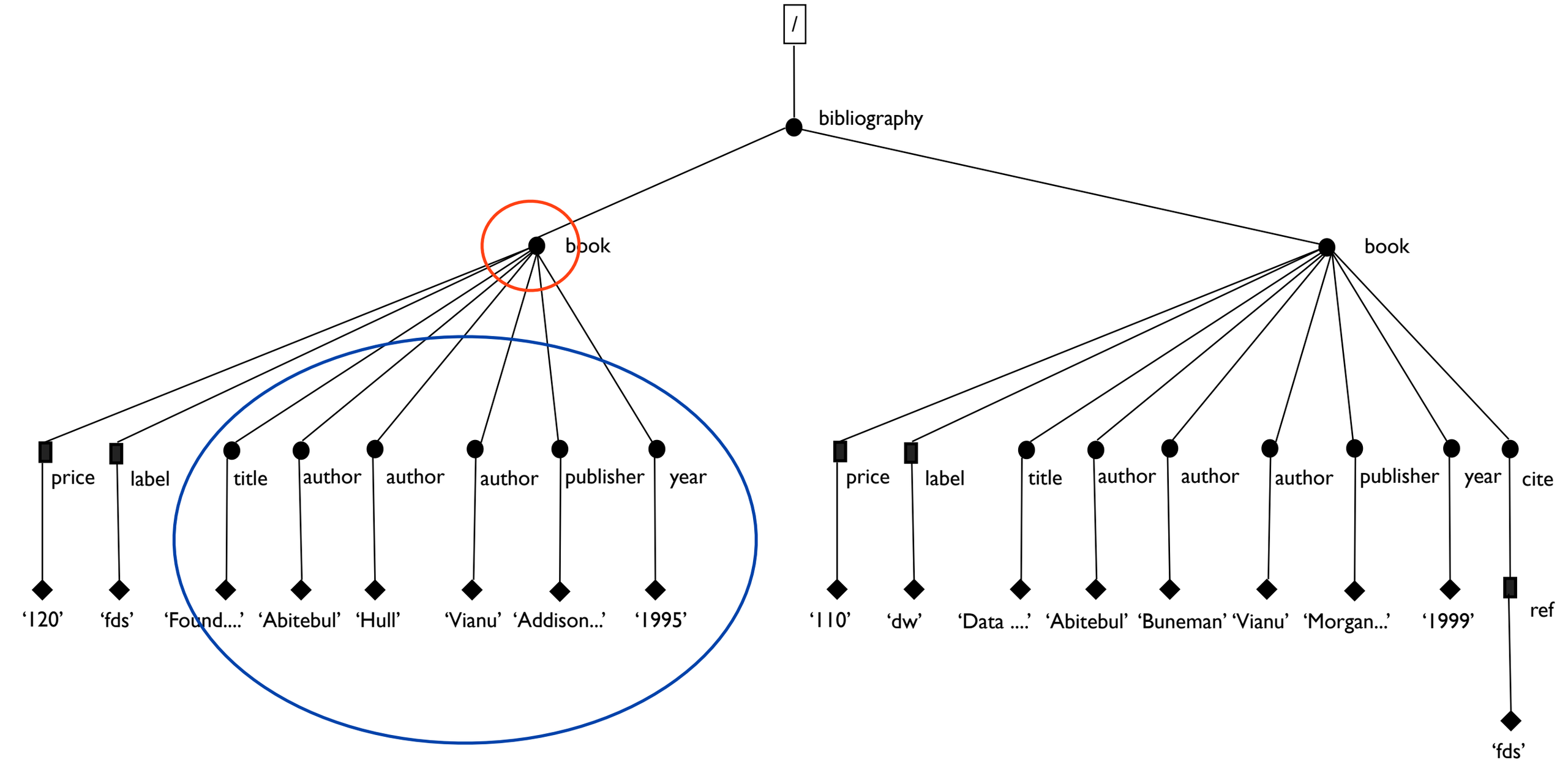
L'étape parent::bibliography



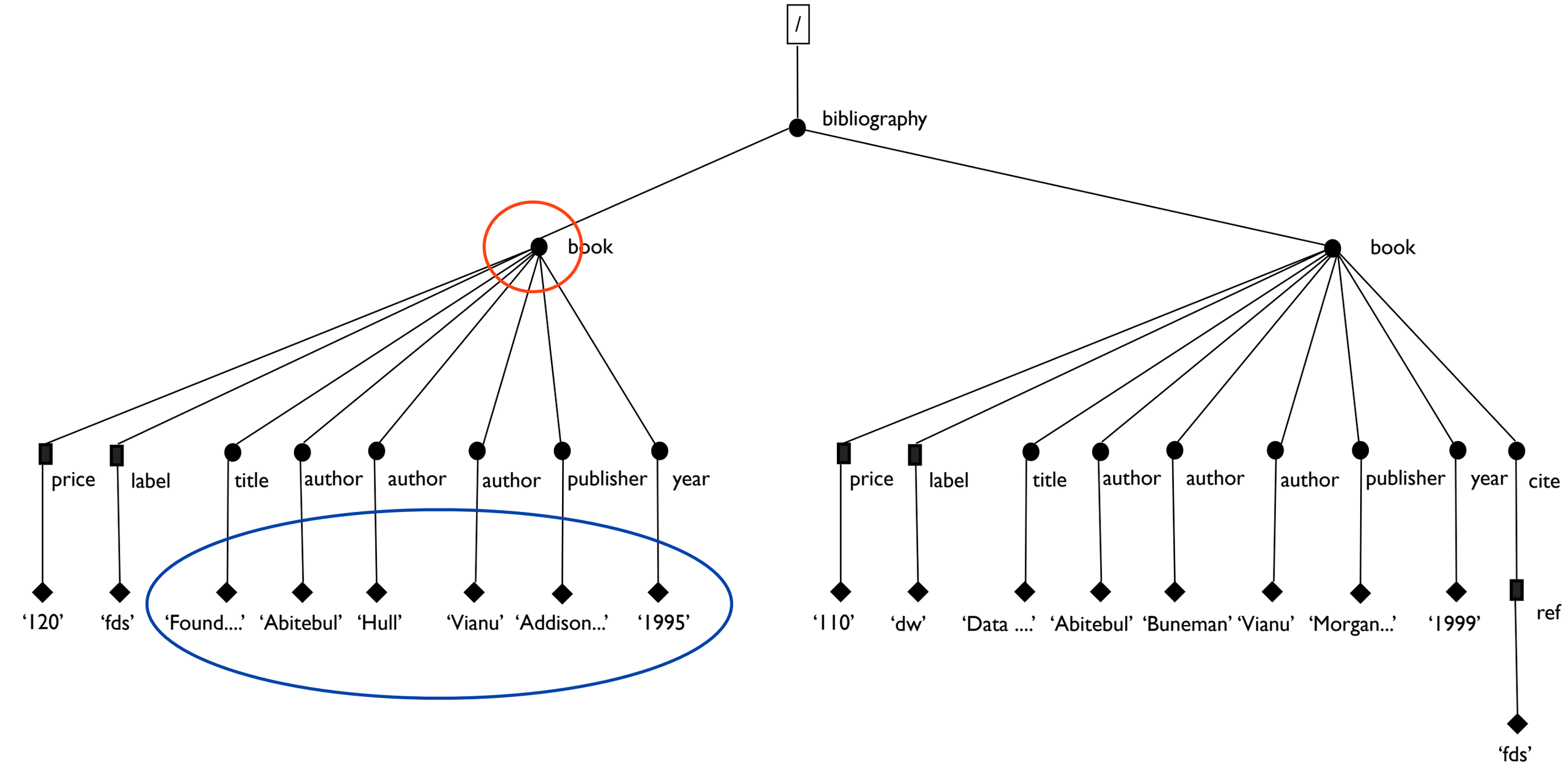
L'axe parent::year



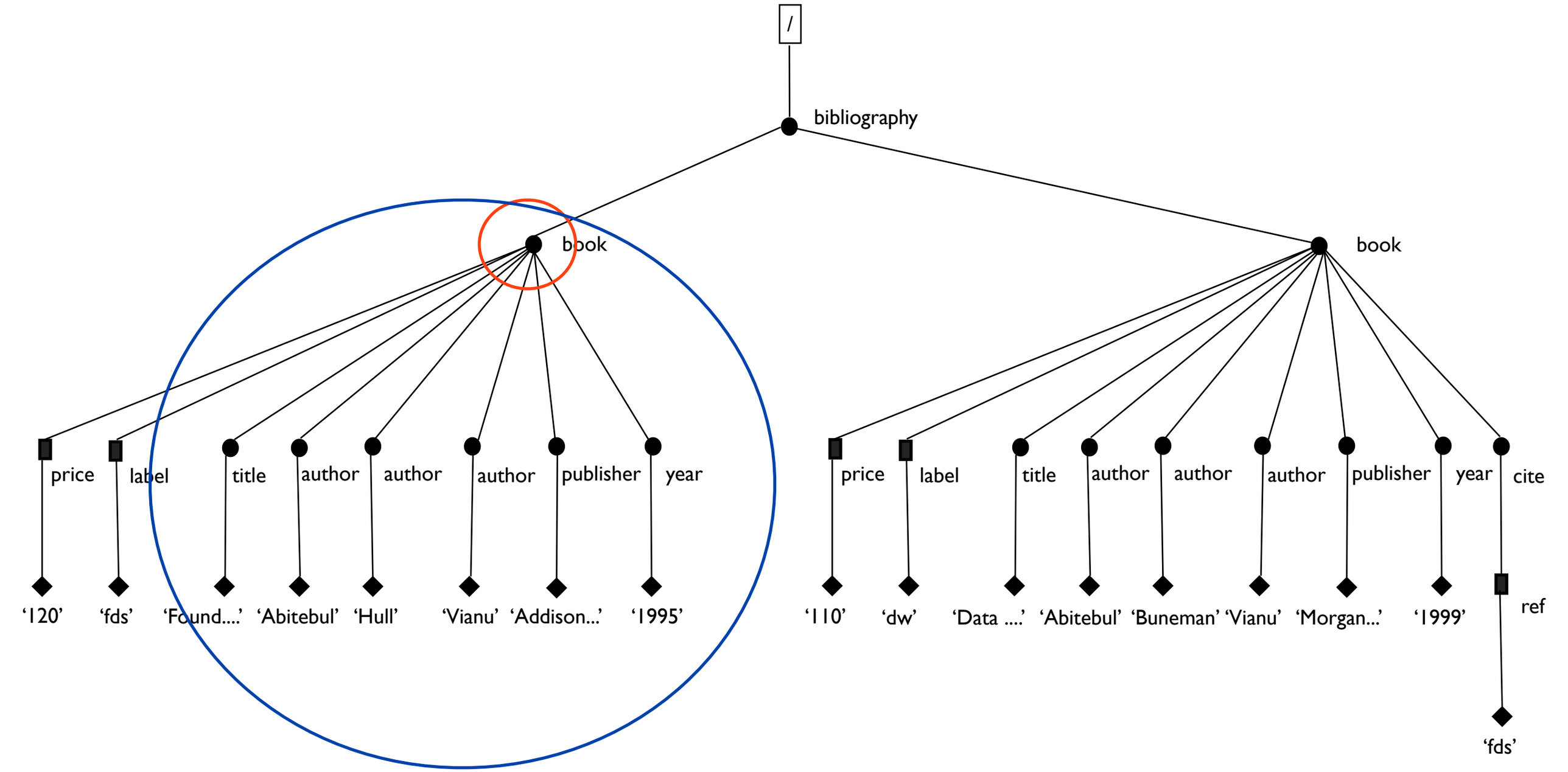
L’axe descendant



L'étape descendant::text()



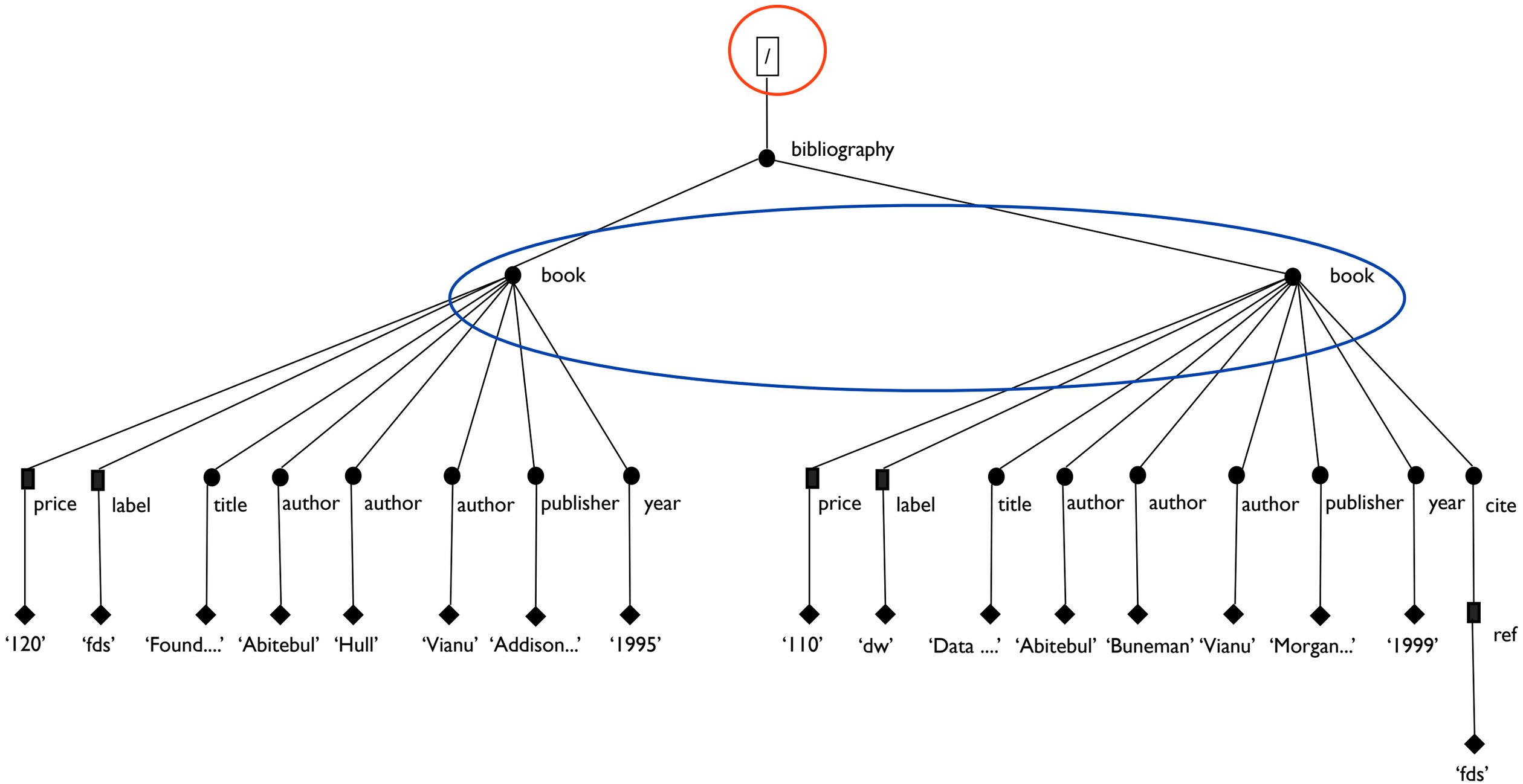
L’axe descendant-or-self



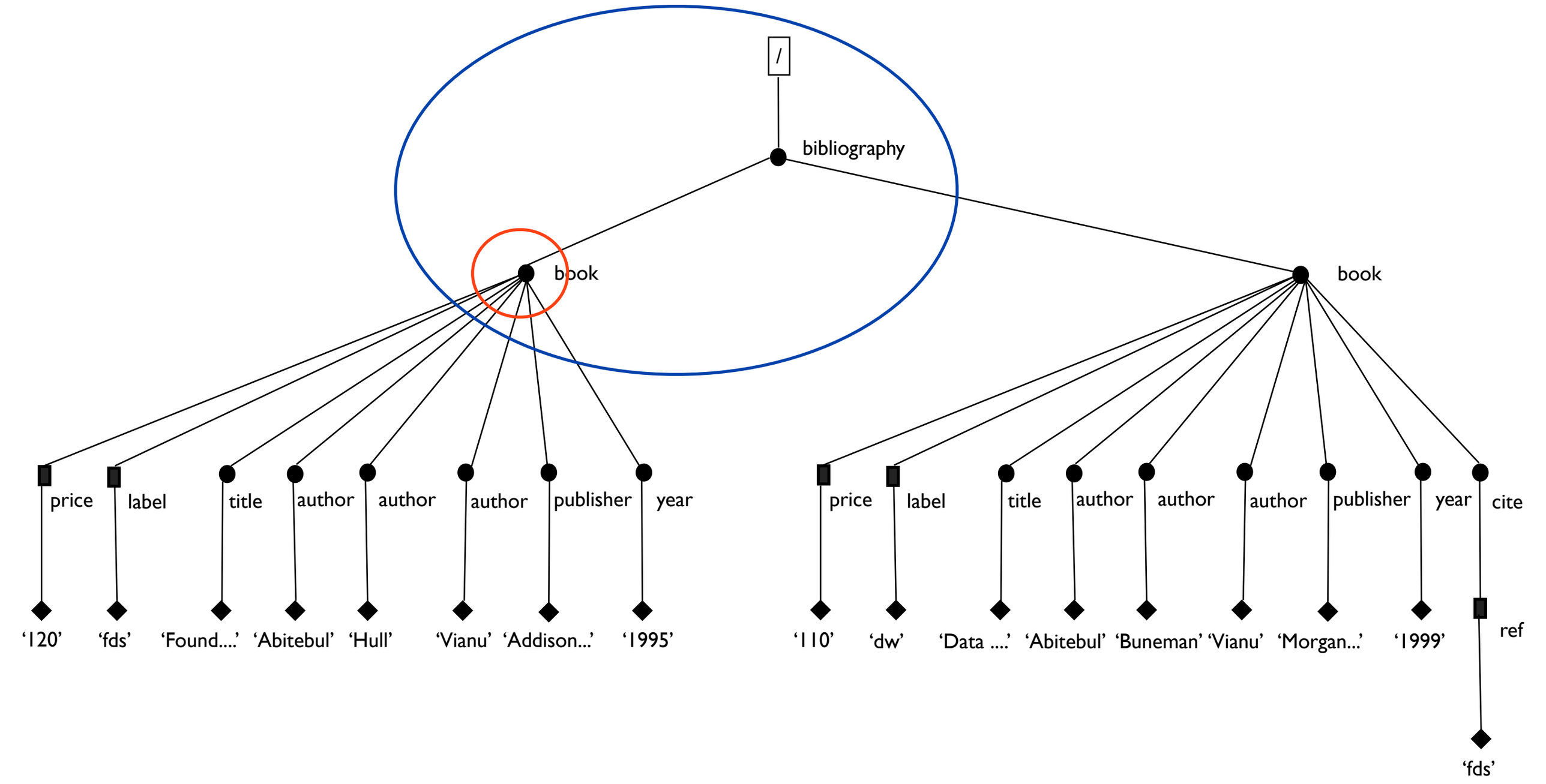
Notation //A

Abréviation de /descendant-or-self::node()/child::A

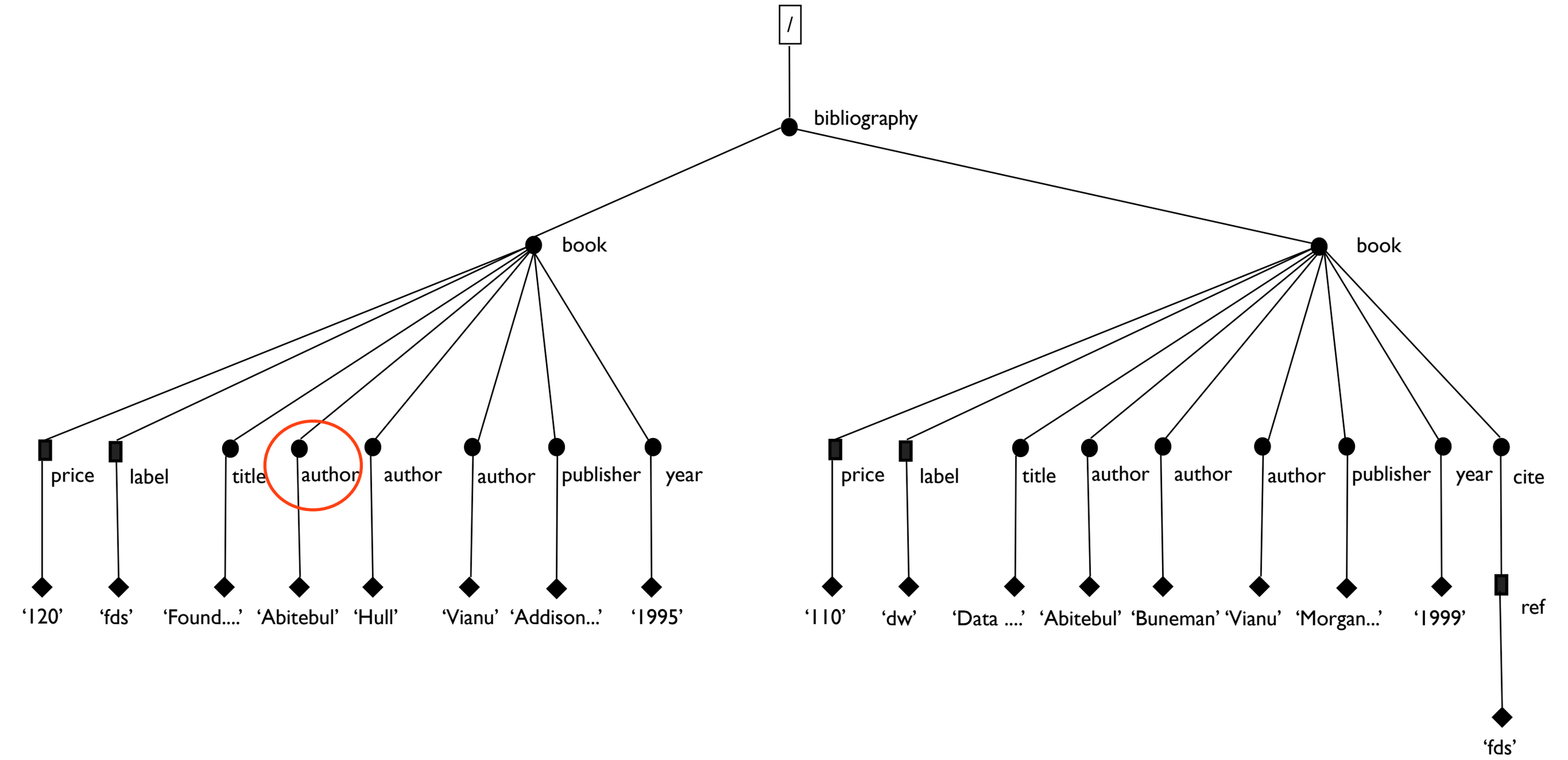
//book


/
bibliography
book
book
price
label
title
author
author
author
publisher
year
price
label
title
author
author
author
publisher
year
cite
'120'
'fds'
'Found....'
'Abitebul'
'Hull'
'Vianu'
'Addison...'
'1995'
'110'
'dw'
'Data ....'
'Abitebul'
'Buneman'
'Vianu'
'Morgan...'
'1999'
ref
'fds'

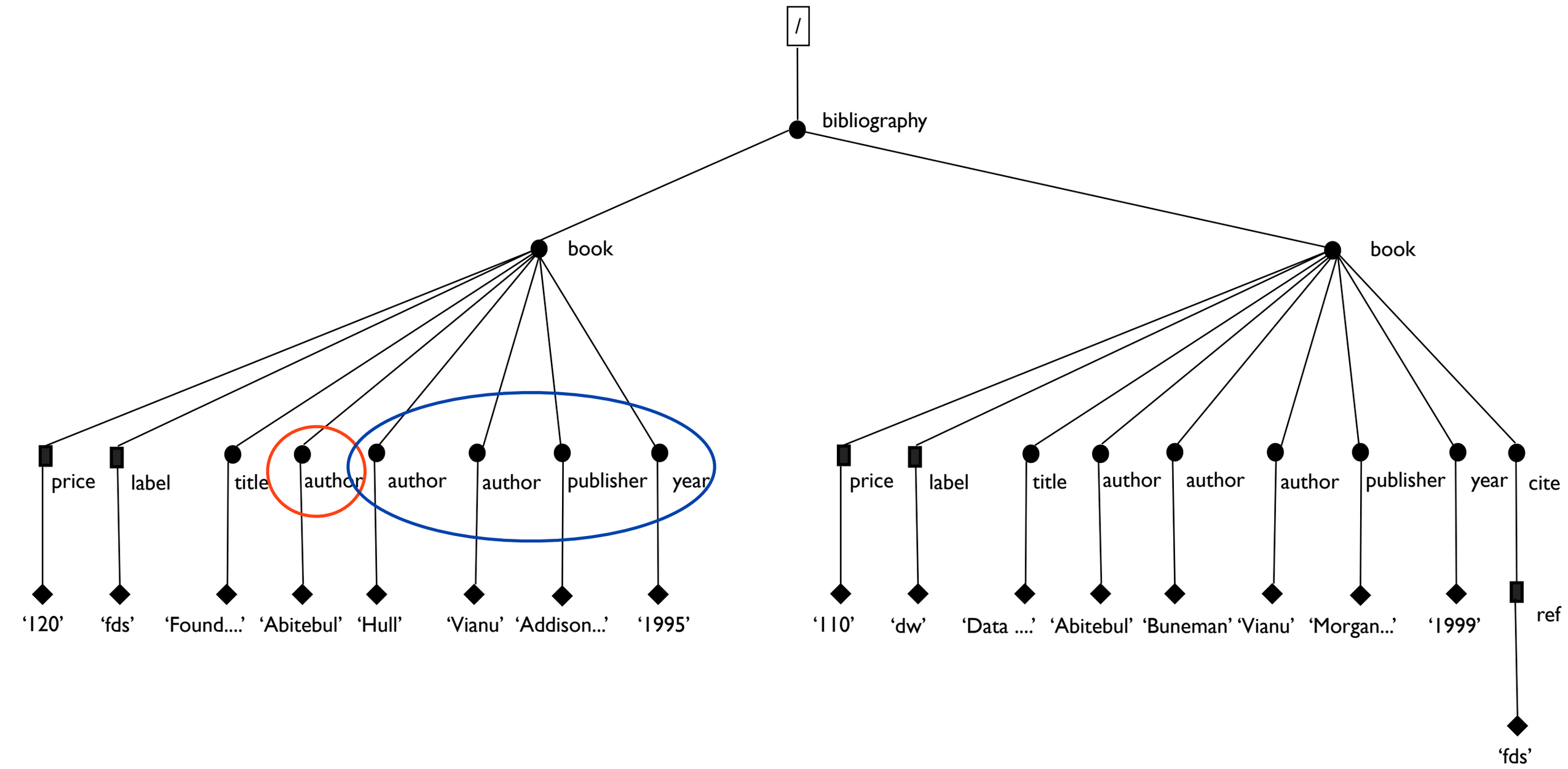
L'axe ancestor-or-self



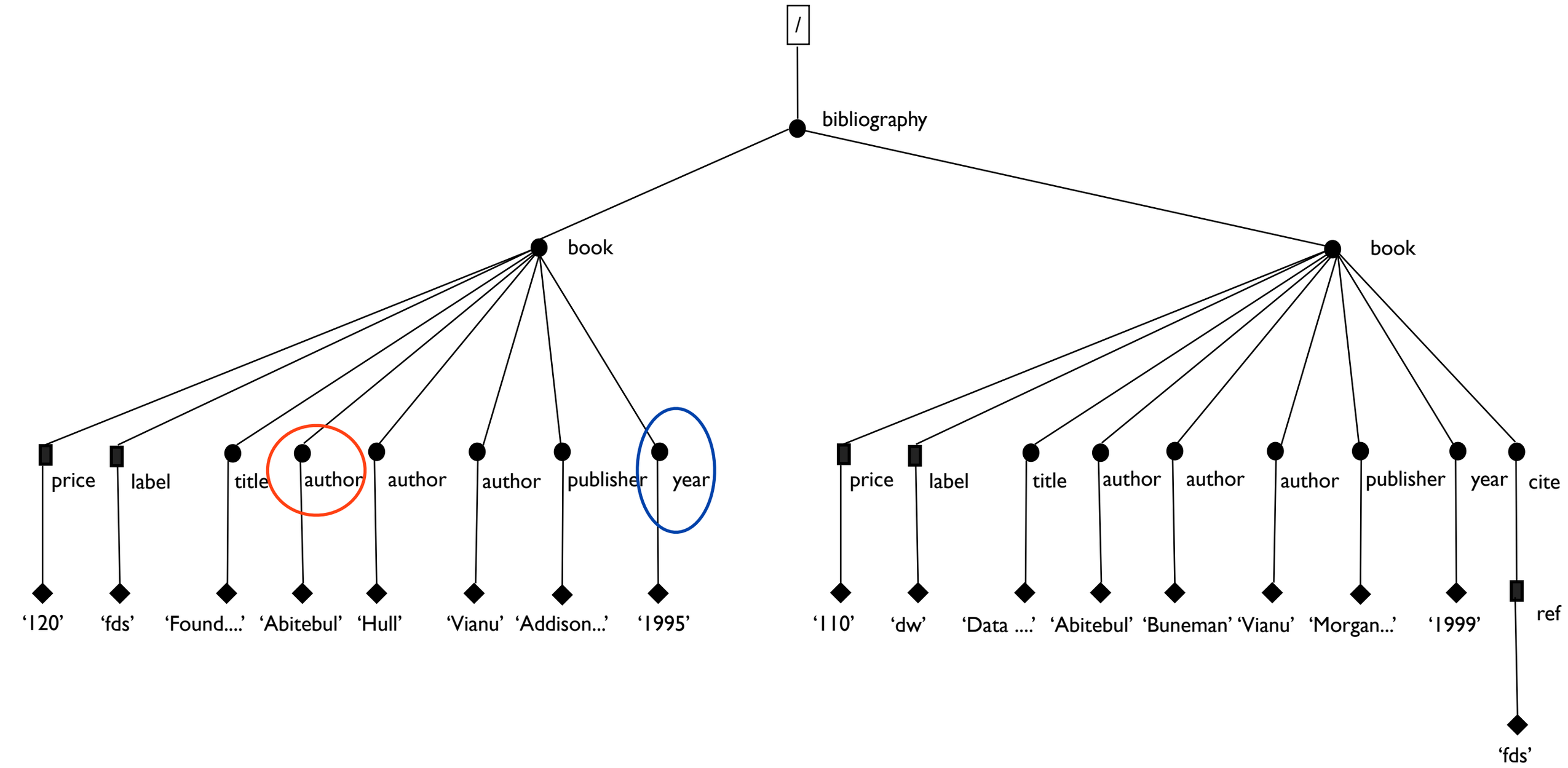
On change de nœud contexte



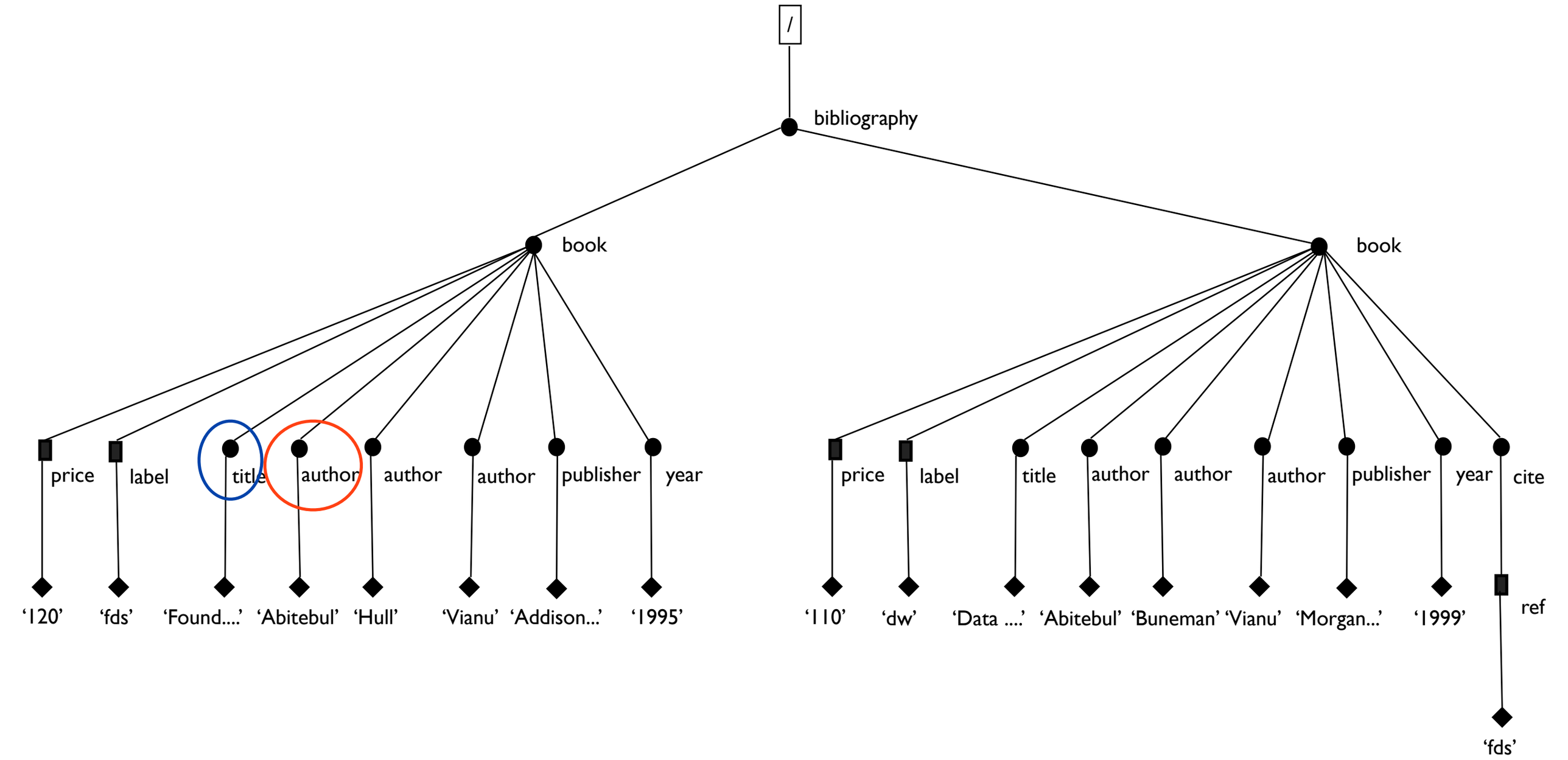
L’axe following-sibling



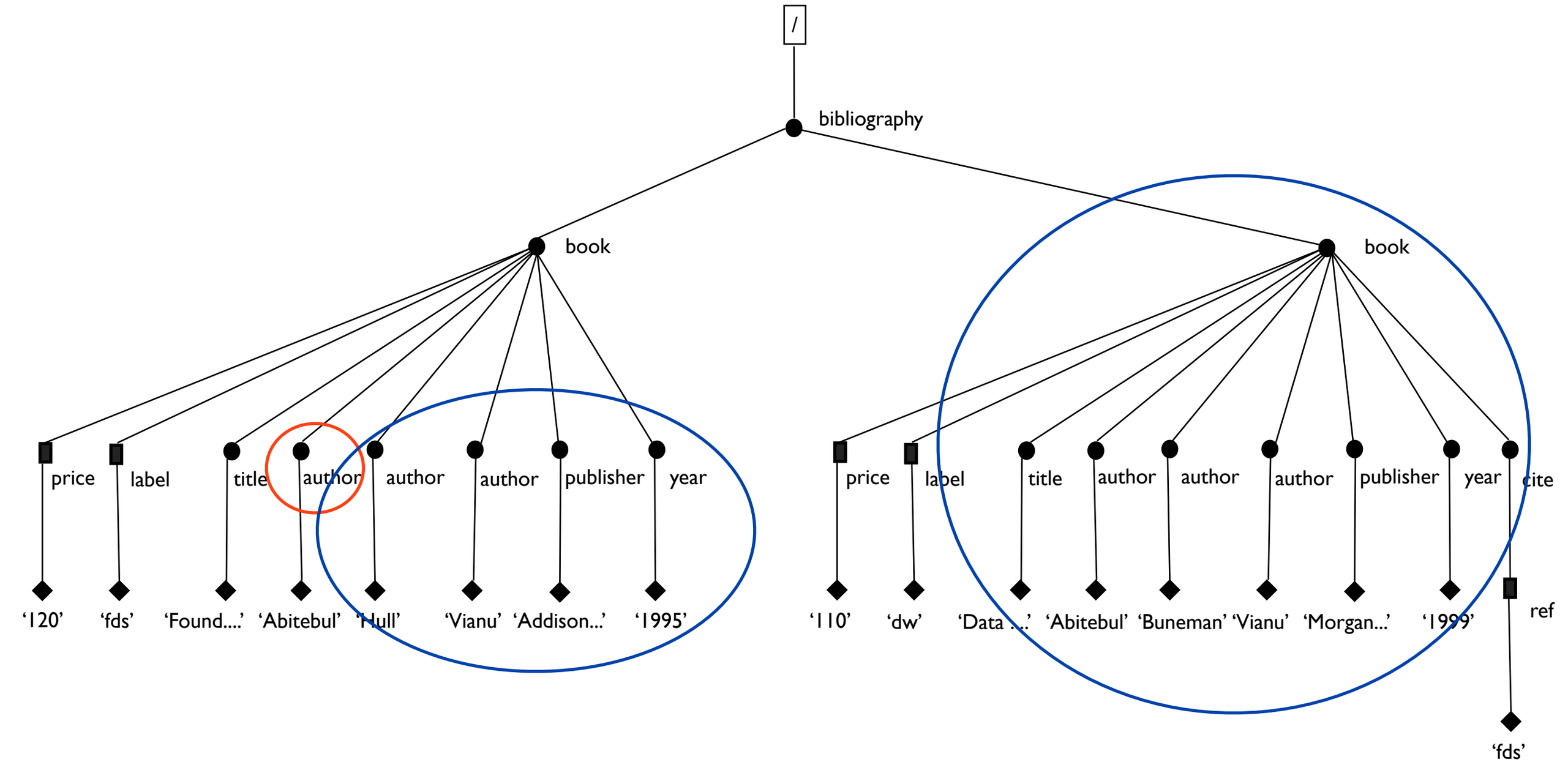
L’étape following-siblings::year



L’axe preceding-sibling

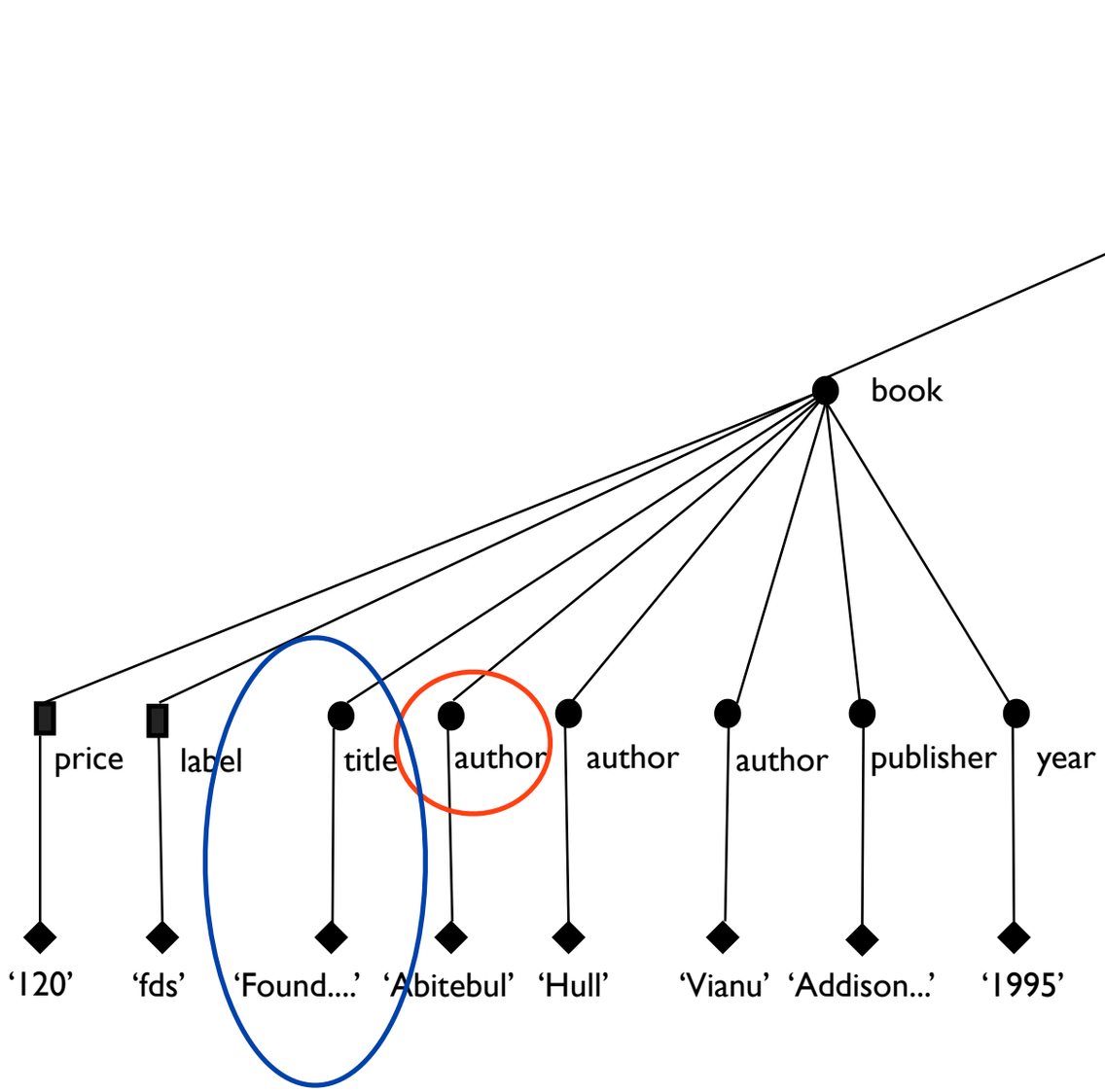


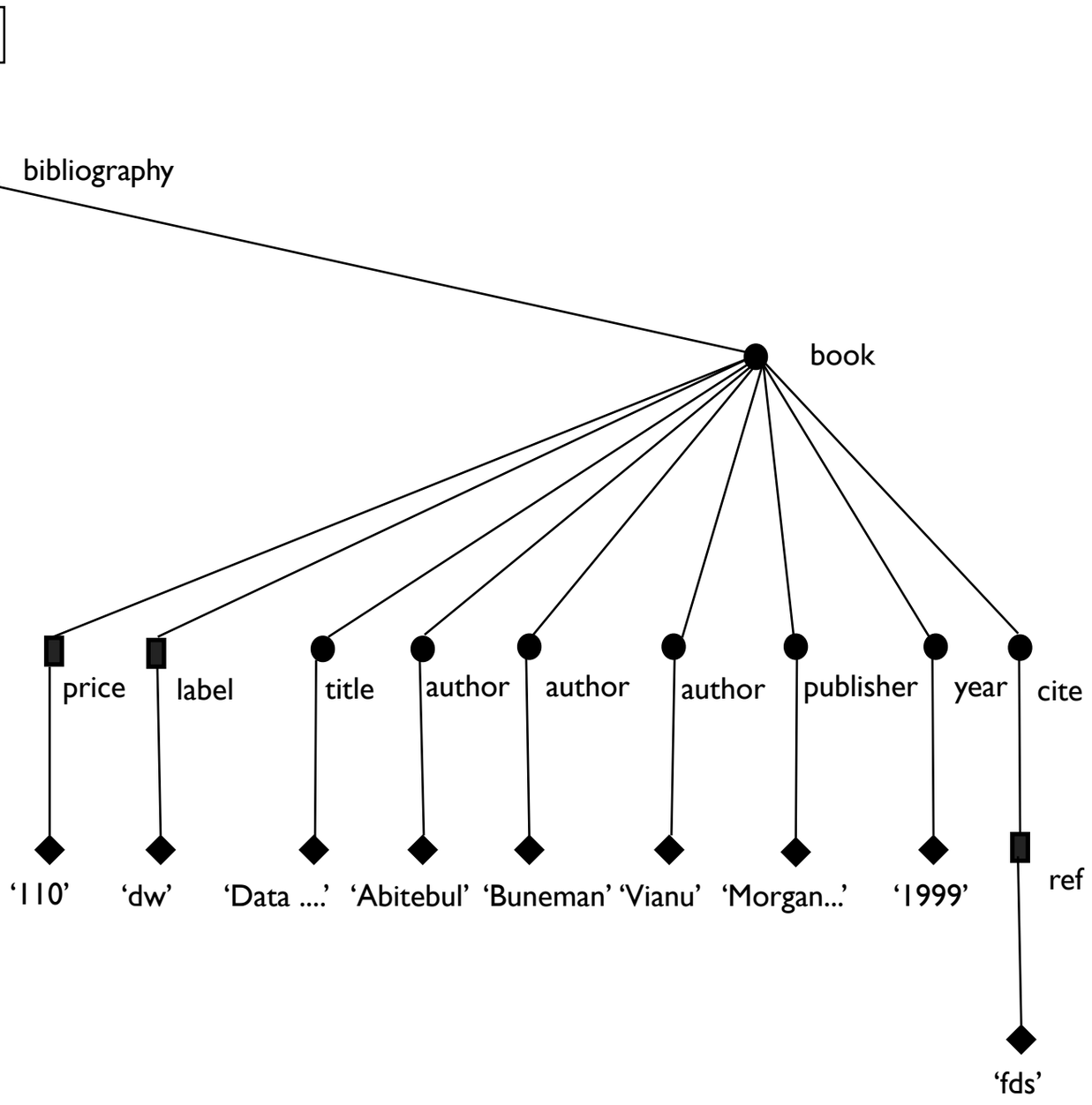
L’axe following



L’axe preceding

$$[\texttt{preceding:}\ \textit{NodeTest}]_{\text{Axis}} \equiv [\texttt{ancestor-or-self::node()/preceding-sibling::node()/descendant-or-self::}\textit{NodeTest}]_{\text{Expr}}$$





Formal semantics - https://www.w3.org/TR/xquery-semantics/

$$[\texttt{following-sibling::}\ \textit{NodeTest}]_{\text{Axis}} == [$$

```
let $e := .
where fn:not($e/self::attribute())
return $e/parent::node()/child:: NodeTest [.>>$e]
```

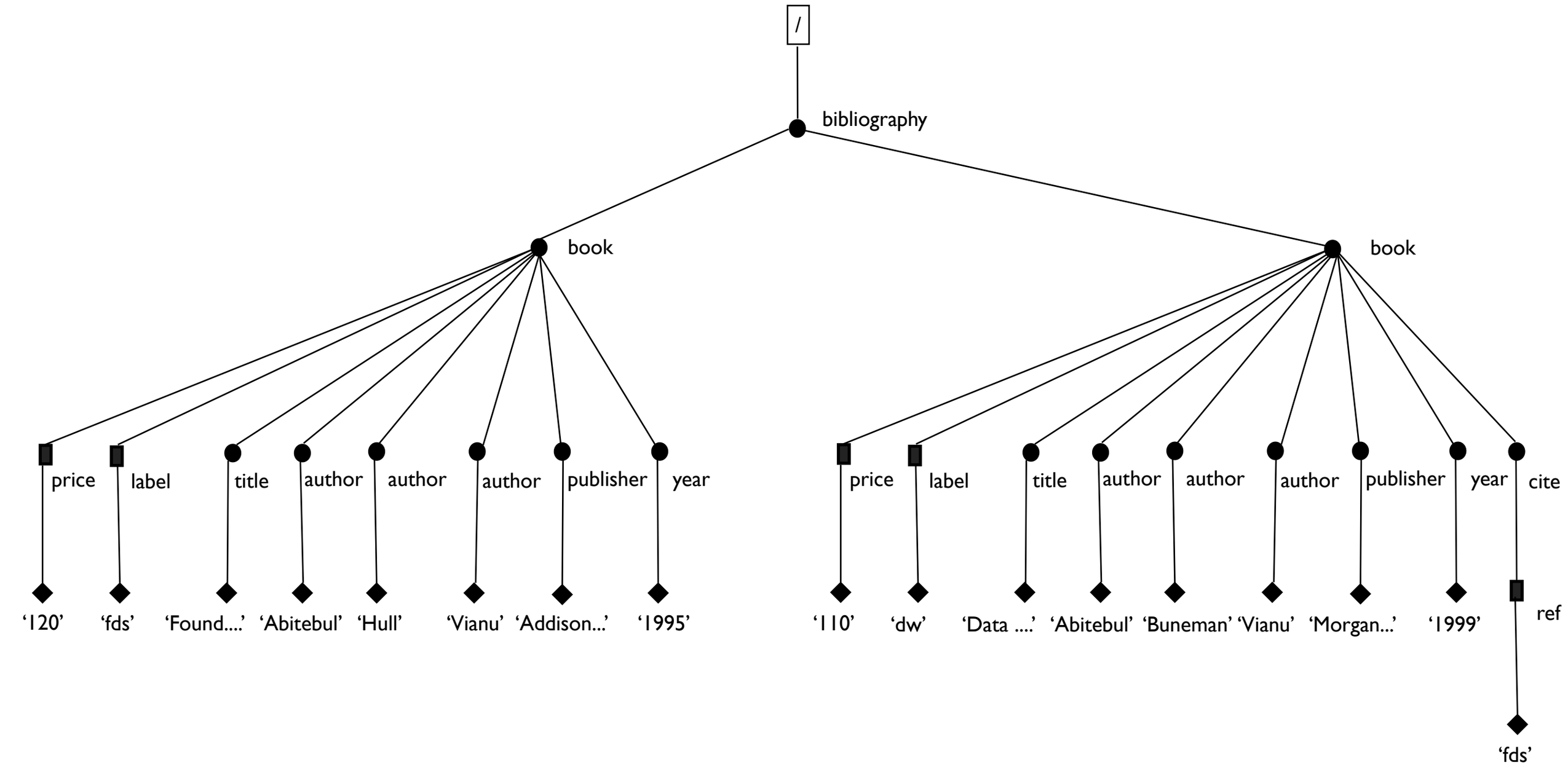
$$]_{\text{Expr}}$$

$$[\texttt{following::}\ \textit{NodeTest}]_{\text{Axis}} == [$$

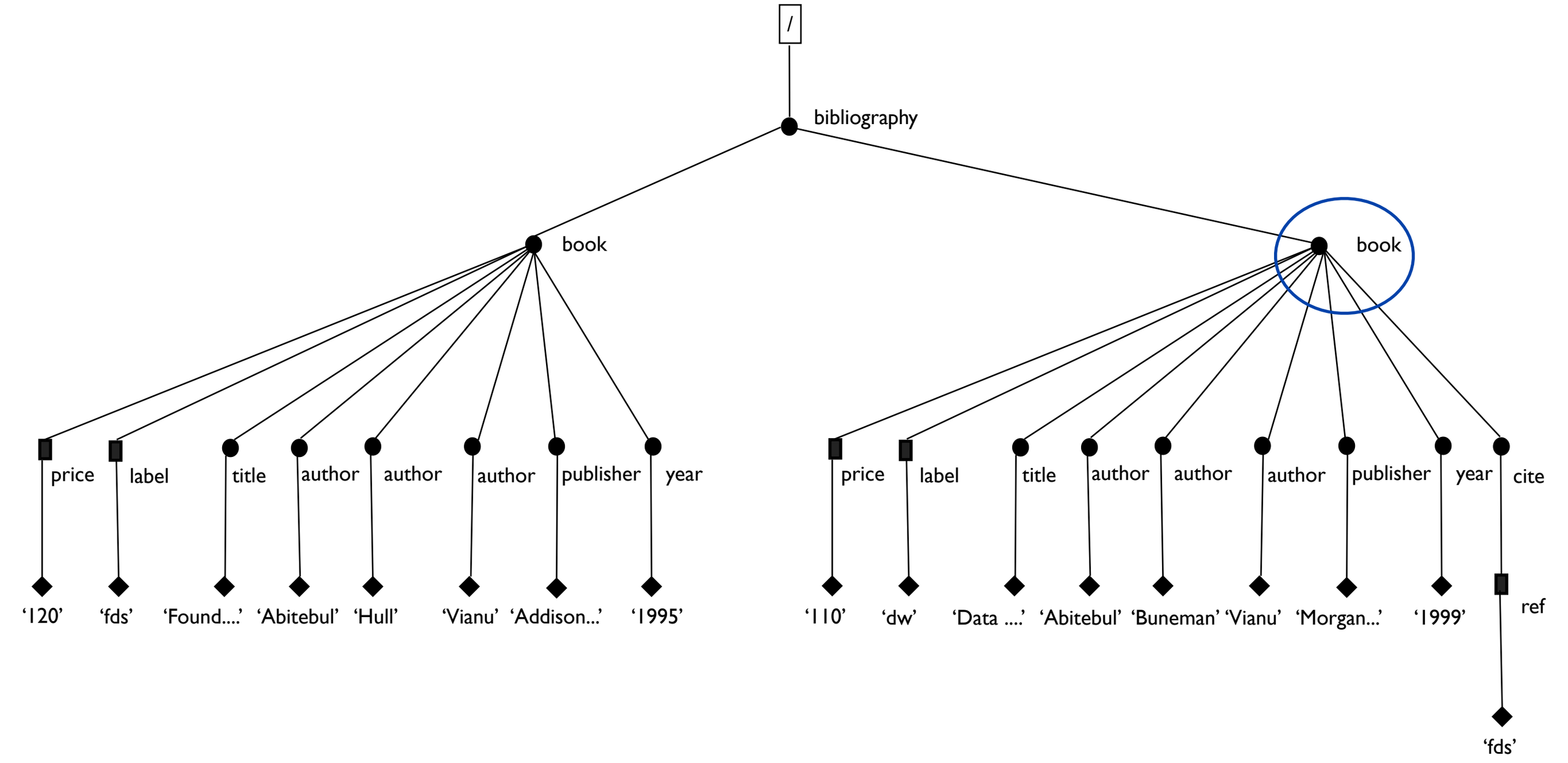
```
let $e := .
return
  fn:root() / descendant:: NodeTest [. >> $e]
  except
  $e / descendant::node()
```

$$]_{\text{Expr}}$$

/descendant::node()/child::cite/parent::node() ?



/descendant::node()/child::cite/parent::node()



Predicats

axe::filtre[**predicat1**]...[**predicatN**]

Prédicat : expression booléenne constituée d'un ou plusieurs tests, composés avec les connecteurs logiques habituels and et or

Types de test :

- toute expression XPath, dont le résultat est convertie en booléen*

- une comparaison ou un appel de fonction.

* il faut connaître les règles de conversion

Proximity position

2.4 Predicates

An axis is either a forward axis or a reverse axis. An axis that only ever contains the context node or nodes that are after the context node in document order is a forward axis. An axis that only ever contains the context node or nodes that are before the context node in document order is a reverse axis. Thus, the ancestor, ancestor-or-self, preceding, and preceding-sibling axes are reverse axes; all other axes are forward axes. Since the self axis always contains at most one node, it makes no difference whether it is a forward or reverse axis. The **proximity position** of a member of a node-set with respect to an axis is defined to be the position of the node in the node-set ordered in document order if the axis is a forward axis and ordered in reverse document order if the axis is a reverse axis. The first position is 1.

A predicate filters a node-set with respect to an axis to produce a new node-set. For each node in the node-set to be filtered, the PredicateExpr is evaluated with that node as the context node, with the number of nodes in the node-set as the context size, and with the proximity position of the node in the node-set with respect to the axis as the context position; if PredicateExpr evaluates to true for that node, the node is included in the new node-set; otherwise, it is not included.

A PredicateExpr is evaluated by evaluating the Expr and converting the result to a boolean. If the result is a number, the result will be converted to true if the number is equal to the context position and will be converted to false otherwise; if the result is not a number, then the result will be converted as if by a call to the **boolean** function. Thus a location path `para[3]` is equivalent to `para[position()=3]`.

Exemples

/A/B[@att1] : les nœuds /A/B qui ont un attribut @att1

/A/B[attribute::att1='a1'] : les nœuds /A/B qui ont un attribut @att1 valant 'a1'

/A/B/descendant::text()[position()=1]
Le premier nœud de type texte descendant d'un /A/B.

S'abrège : /A/B/descendant::text()[1]

Conversion dans /A/B[@att1]

On s'intéresse aux nœuds de type B fils de l'élément racine A.

Parmi ces nœuds on ne prend que ceux pour lesquels le prédicat [@att1] s'évalue à true

Cette expression s'évalue avec pour nœud contexte un élément B

[@att1] vaut true ssi @att1 renvoie un ensemble de nœuds non vide

Types dans XPath

On peut effectuer des comparaisons, des opérations.

Cela implique un typage et des conversions de type.

Types XPath :

les numériques
les chaînes de caractères
les booléens (true et false)
les ensembles de nœuds

Conversions

Deux conversions sont toujours possibles :

Vers une chaîne de caractères : utile pour la production de texte en XSLT
Ex. (xsl:value-of)

Vers un booléen : utile pour les tests effectués dans XSLT
Ex. (xsl:if, xsl:when)

Conversion booléennes automatiques

Numériques : 0 ou NaN sont false, tout le reste est true

Chaînes : une chaîne vide est false, tout le reste est true

Ensembles de nœuds : un ensemble vide est false, tout le reste est true

Conversion booléennes automatiques

Numériques : 0 ou NaN sont false, tout le reste est true

Chaînes : une chaîne vide est false, tout le reste est true

Ensembles de nœuds : un ensemble vide est false, tout le reste est true

Fonctions

concat(chaîne1, chaîne2, ...) : pour concaténer des chaînes

contains(chaîne1, chaîne2) : teste si chaîne1 contient chaîne2

count (expression) : renvoie le nombre de nœuds désignés par expression

name() : renvoie le nom du nœud contexte

not(expression) : permet d'exprimer la négation

Exemples au tableau