Tools for Structuring and Searching Text

Frank Wm. Tompa

WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE

cs.uwaterloo.ca

OVERVIEW

XML structure

Navigational access

- XPath — accessing components
- XQuery — querying documents

Transformation

- XSLT — transforming documents
- CSS — formatting documents for the Web

Search engines

- sgrep — structured search
- Wumpus — scaling up to larger collections

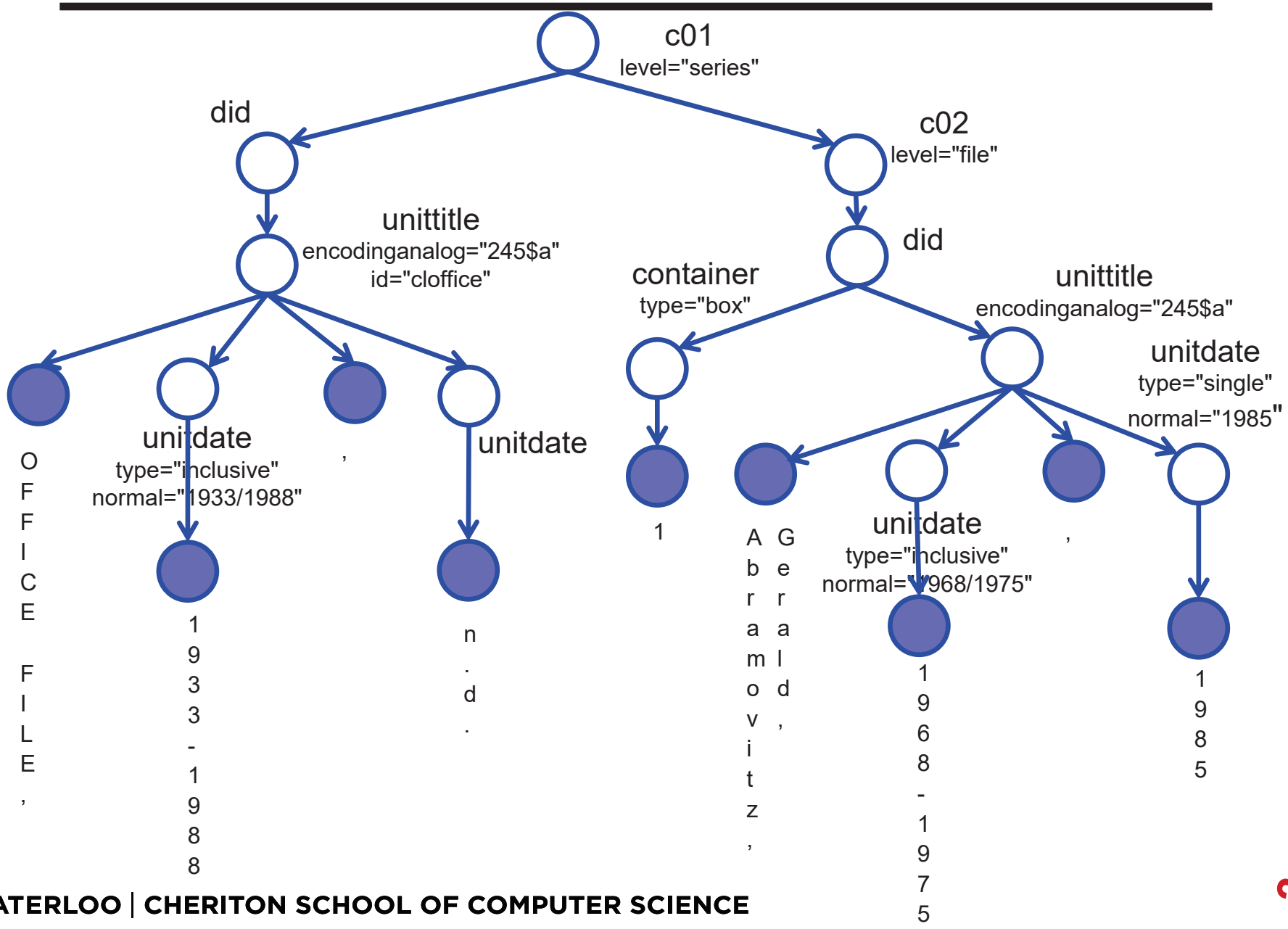
XML STRUCTURE *

```
OFFICE FILE, 1933-1988, n.d.

Box 1          Abramovitz, Gerald, 1968-1975, 1985
```

```
<c01 level="series">
   <did>
      <unittitle encodinganalog="245$a" id="cloffice">OFFICE FILE,
         <unitdate type="inclusive" normal="1933/1988">1933-1988</unitdate>,
         <unitdate>n.d.</unitdate>
      </unittitle>
   </did>
   <c02 level="file">
      <did>
         <container type="box">1</container>
         <unittitle encodinganalog="245$a">Abramovitz, Gerald,
            <unitdate type="inclusive" normal="1968/1975">1968-1975</unitdate>,
            <unitdate type="single" normal="1985">1985</unitdate>
         </unittitle>
      </did>
   </c02>
</c01>
```

* Example taken from *EAD Best Practices at the Library of Congress: Archival Finding Aid <shttp://www.loc.gov/rr/ead/lcp/lcp_dsc.html>*

XML TREE


c01
level="series"
did
c02
level="file"
unittitle
encodinganalog="245$a"
id="cloffice"
did
container
type="box"
unittitle
encodinganalog="245$a"
unitdate
type="single"
normal="1985"
unitdate
type="inclusive"
normal="1933/1988"
unitdate
1
unitdate
type="inclusive"
normal="1968/1975"
O F F I C E F I L E ,
1 9 3 3 - 1 9 8 8
,
n . d .
A b r a m o v i t z ,
G e r a l d ,
,
1 9 6 8 - 1 9 7 5
1 9 8 5

XML TREE

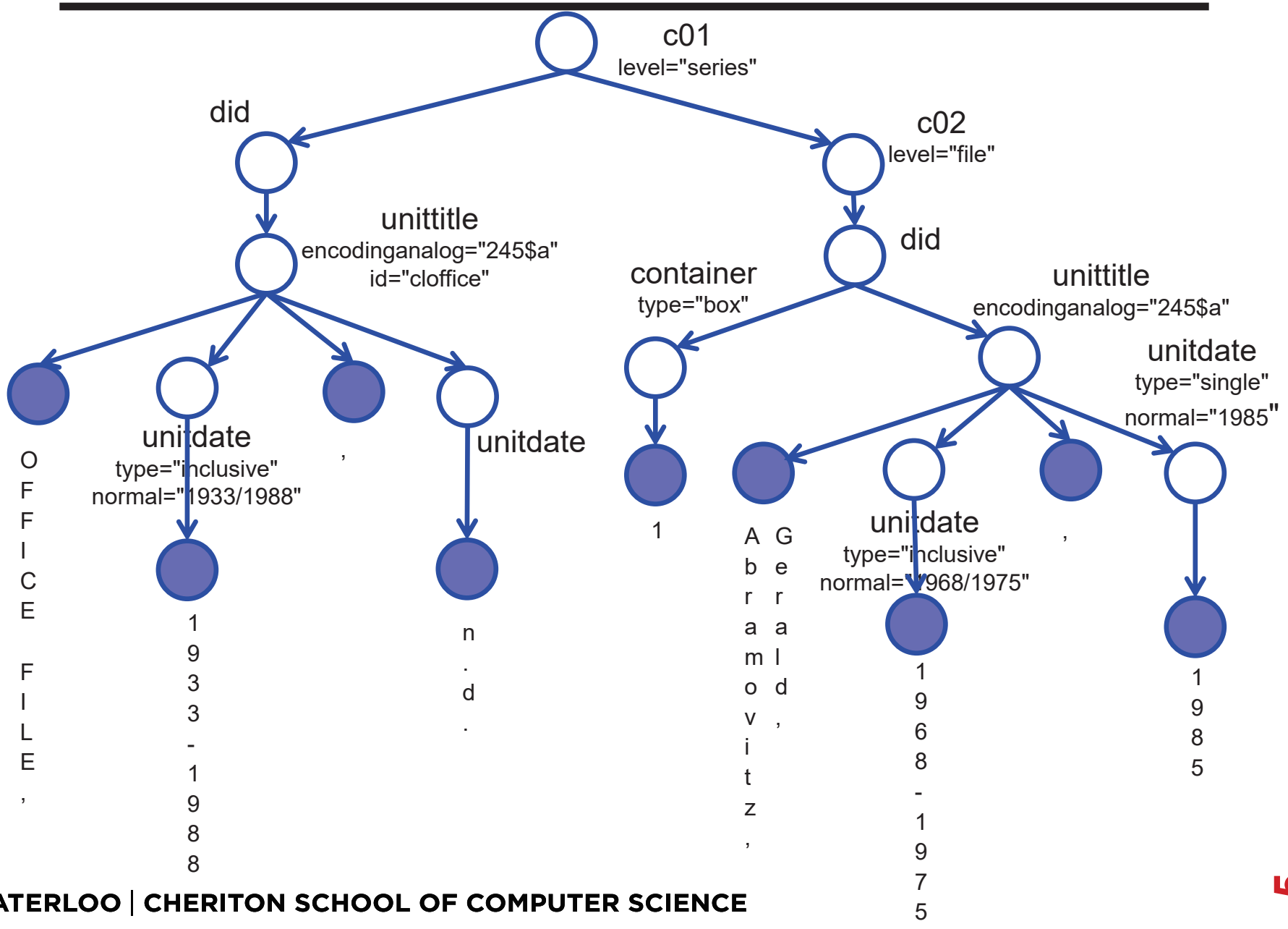
Nodes

- Root (omitted in my drawings, for simplicity)
- Element
- Attribute (included with parent in my drawings)
- Text
- Processing Instruction
- Comment
- Namespace

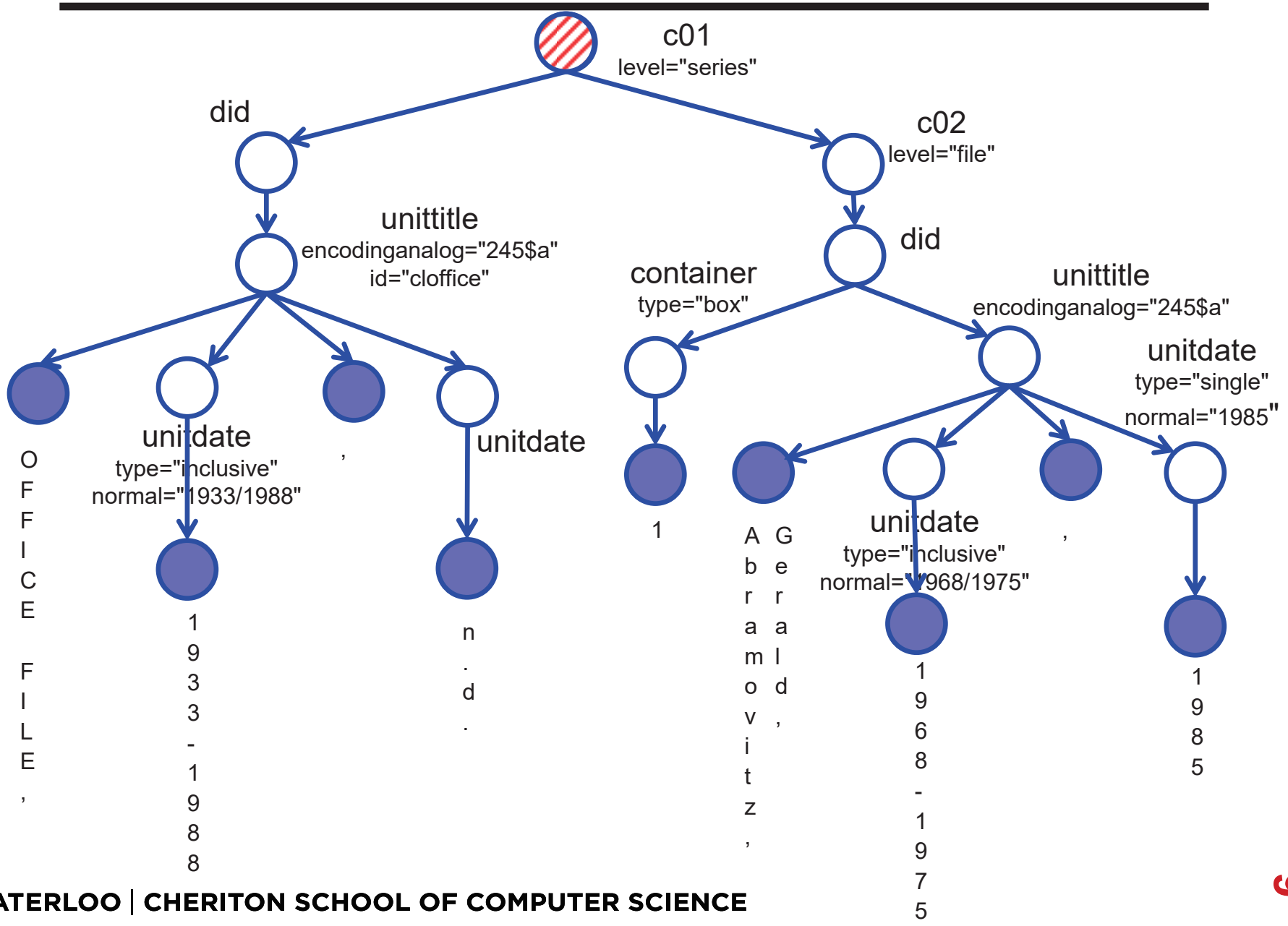
Edges

- connect parent to child
- in *document order*; therefore nodes (other than attribute and namespace nodes) have position wrt parent

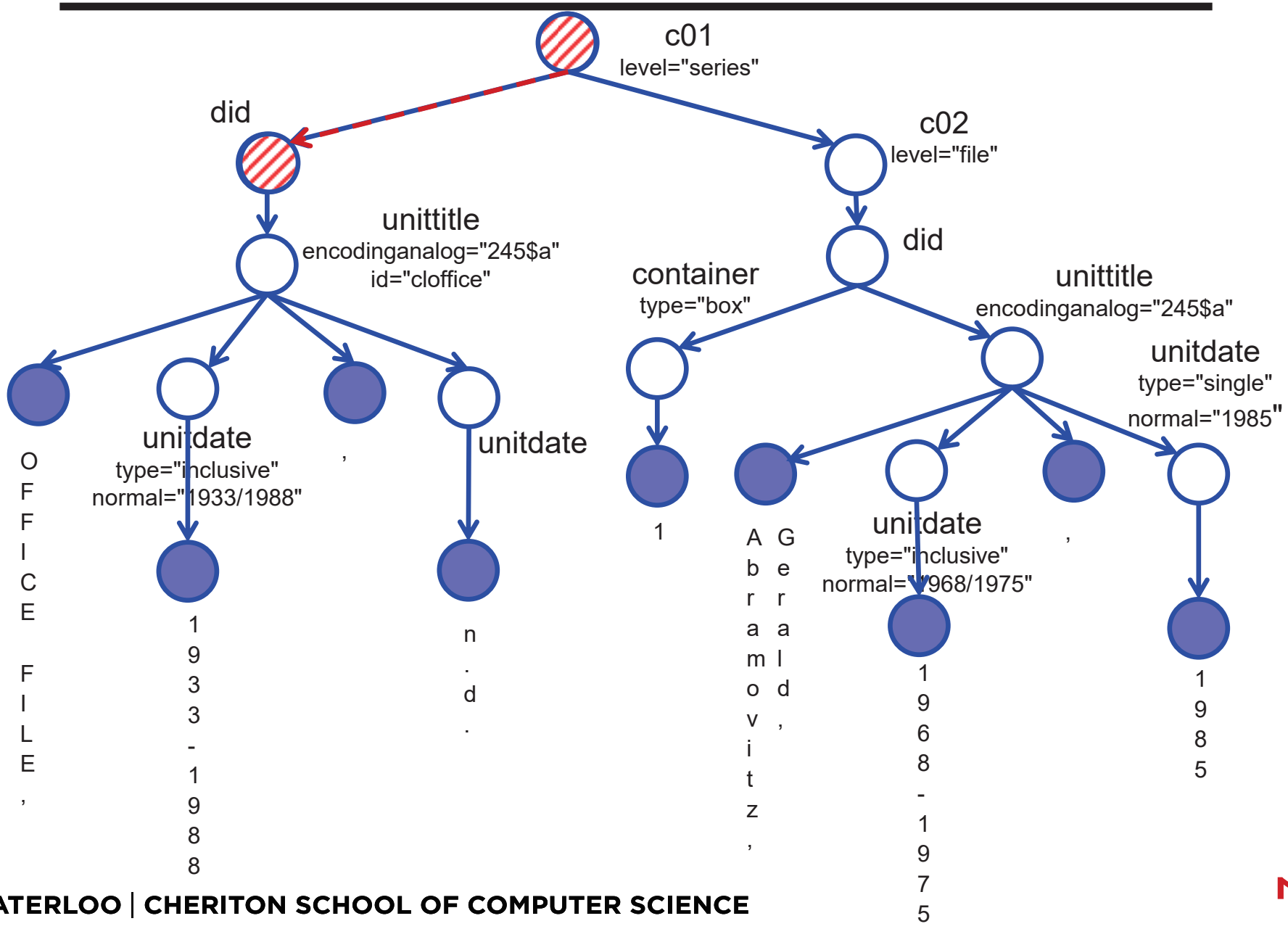
XPATH


c01
level="series"
did
c02
level="file"
unittitle
encodinganalog="245$a"
id="cloffice"
did
container
type="box"
unittitle
encodinganalog="245$a"
unitdate
type="single"
normal="1985"
unitdate
type="inclusive"
normal="1933/1988"
unitdate
1
unitdate
type="inclusive"
normal="1968/1975"
O F F I C E F I L E ,
1 9 3 3 - 1 9 8 8
,
n . d .
A b r a m o v i t z ,
G e r a l d ,
,
1 9 6 8 - 1 9 7 5
1 9 8 5

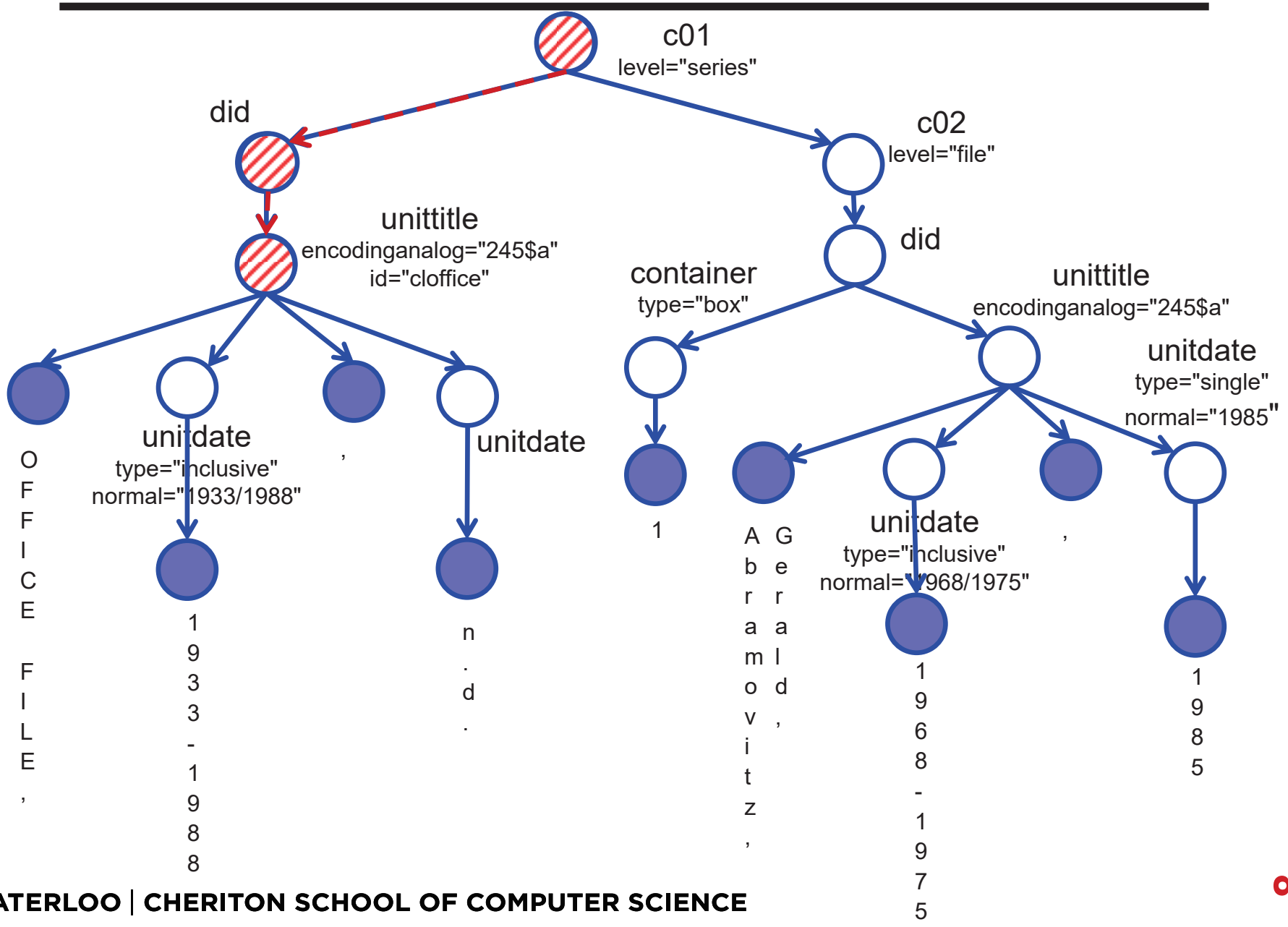
XPATH /c01


c01
level="series"
did
c02
level="file"
unittitle
encodinganalog="245$a"
id="cloffice"
did
container
type="box"
unittitle
encodinganalog="245$a"
unitdate
type="single"
normal="1985"
unitdate
type="inclusive"
normal="1933/1988"
unitdate
1
unitdate
type="inclusive"
normal="1968/1975"
O F F I C E F I L E ,
1 9 3 3 - 1 9 8 8
,
n . d .
A b r a m o v i t z ,
G e r a l d ,
,
1 9 6 8 - 1 9 7 5
1 9 8 5

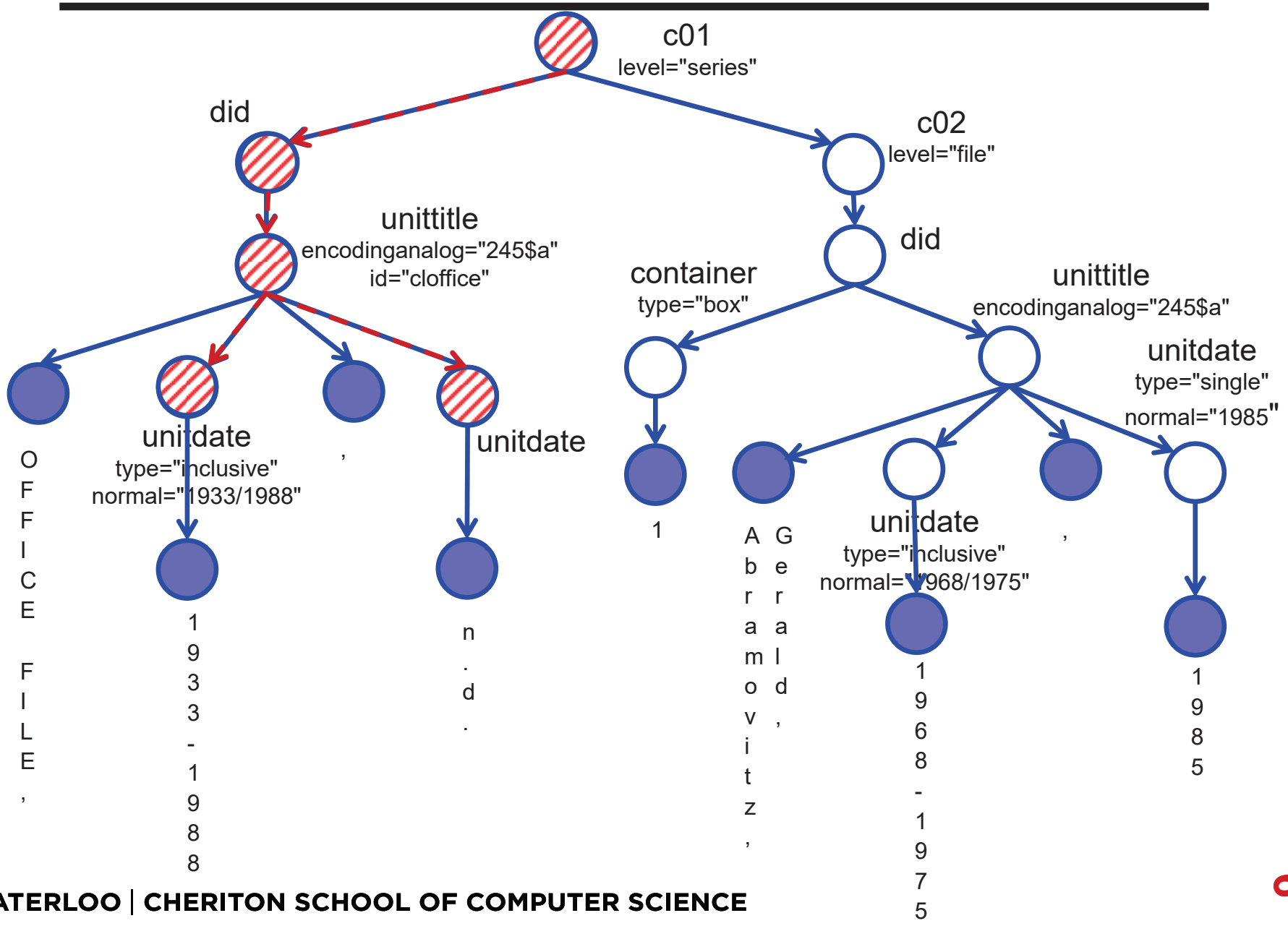
XPATH /c01/did



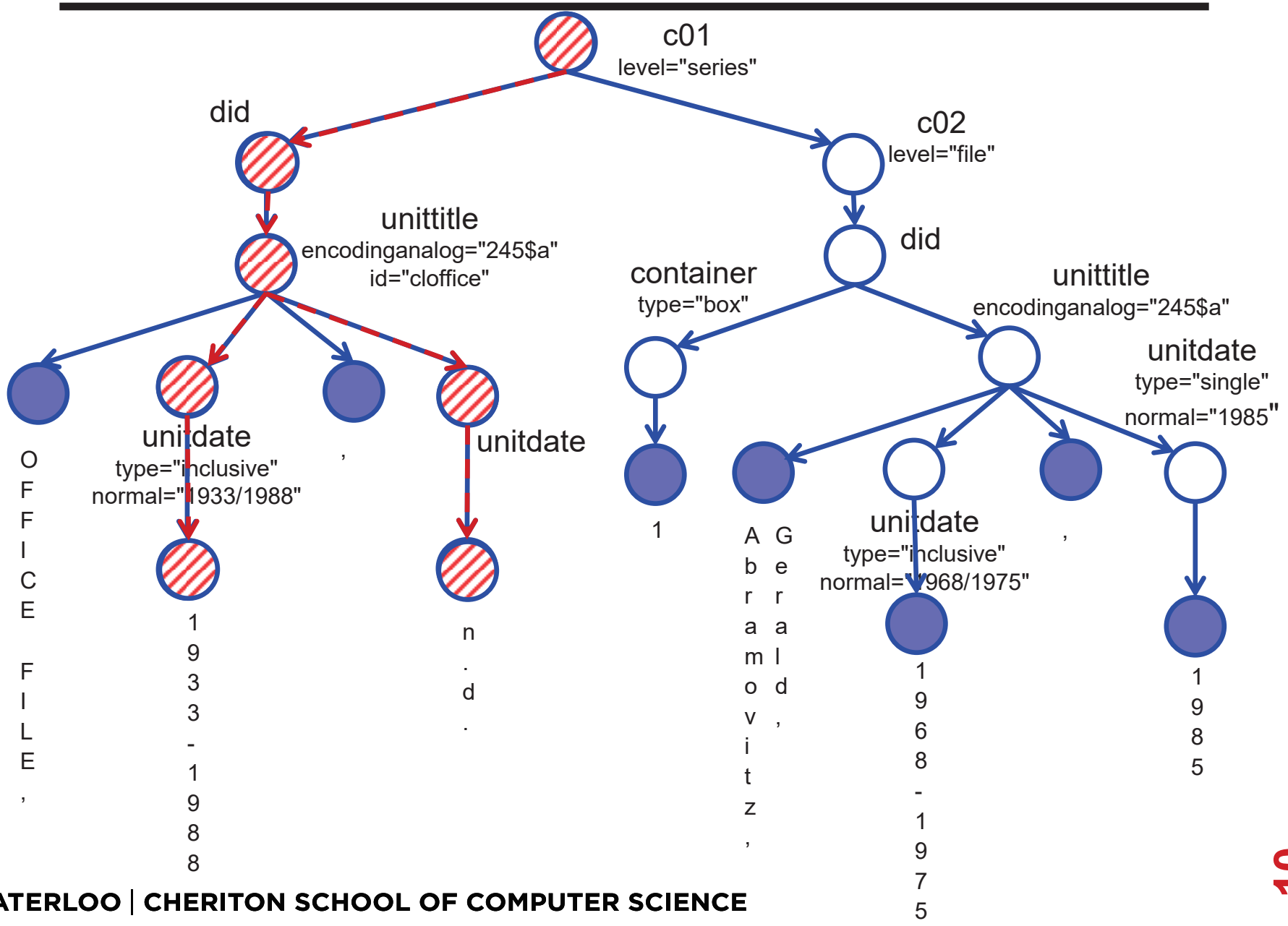
XPATH /c01/did/unittitle



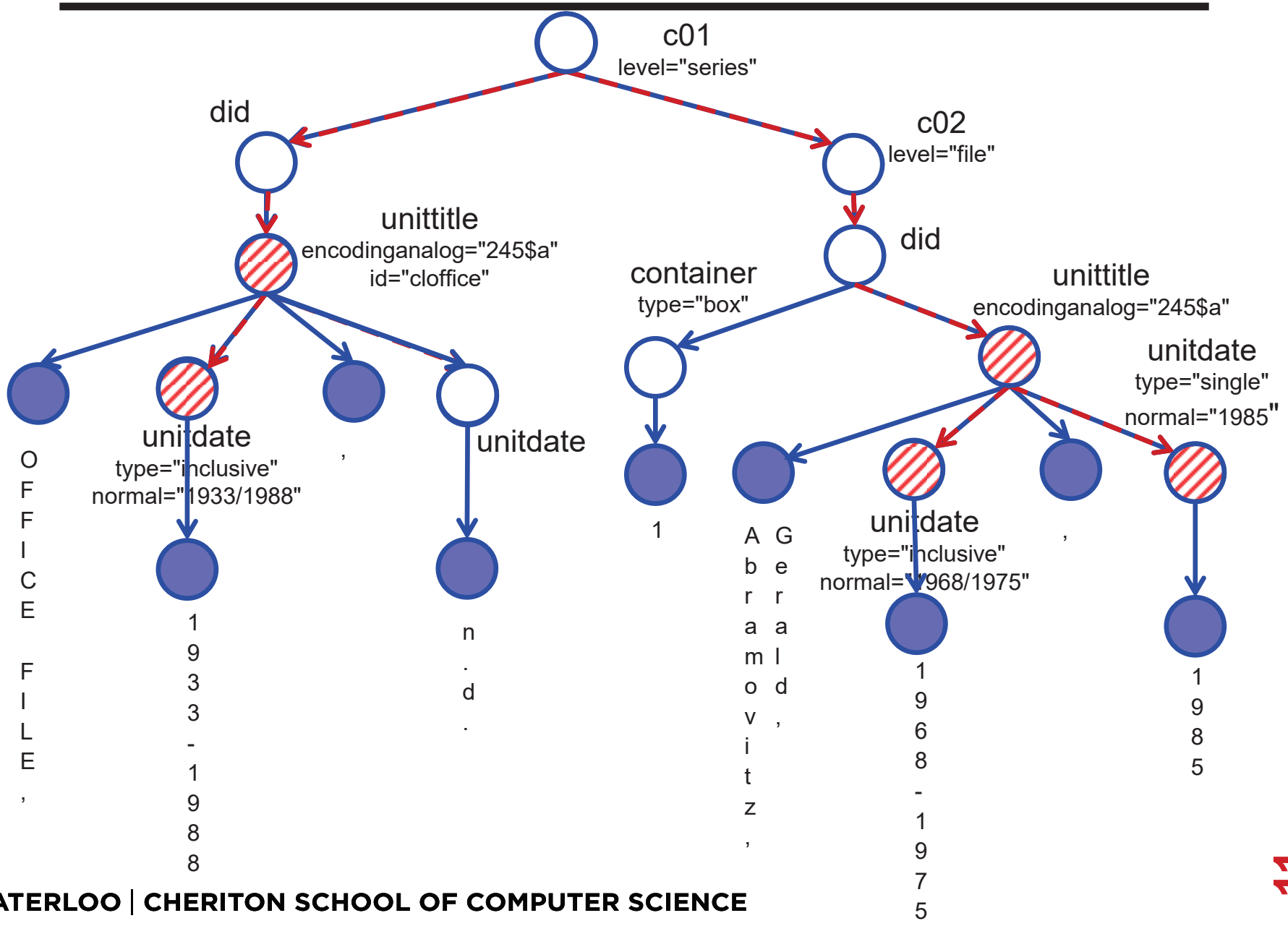
XPATH /c01/did/unittitle/unitdate



XPATH /c01/did/unittitle/unitdate/text()



XPATH //unittitle/unitdate[@type]



XPATH

single step: *axis::node-test[predicate]*

- e.g., `child::unitdate[attribute::type and parent::unittitle]`
- axes:
  - *forward axes*: child, descendent, self, descendent-or-self, following-sibling, following, attribute, namespace
  - *reverse axes*: parent, ancestor, ancestor-or-self, preceding-sibling, preceding

path: "/" step | step | path "/" step

- e.g., `/descendant-or-self::unittitle/child::unitdate[attribute::type]`
- abbreviated notation (`/`, `//`, `.`, `..`, `@`)
- e.g., `//unittitle/unitdate[@type]`
- *N.B.* Paths can always be rewritten to avoid using reverse axes.

wild cards:

- `node( )`, `*`, and `@*`

XQUERY

To query a collection of documents

- like a database
- uses XPath to specify document components
- e.g., `doc(MyArchive.xml)//unitdate[@type]`

More expressive power through FLWOR expressions

- `for`, `let`, `where`, `order by`, `return`
- e.g.,

```
<archiveDates>
{for $d in doc(MyArchive.xml)//unitdate[@type]
 return <date>{$d/text()}</date>}
</archiveDates>
```

Now also includes

- update functionality
- full-text search capability

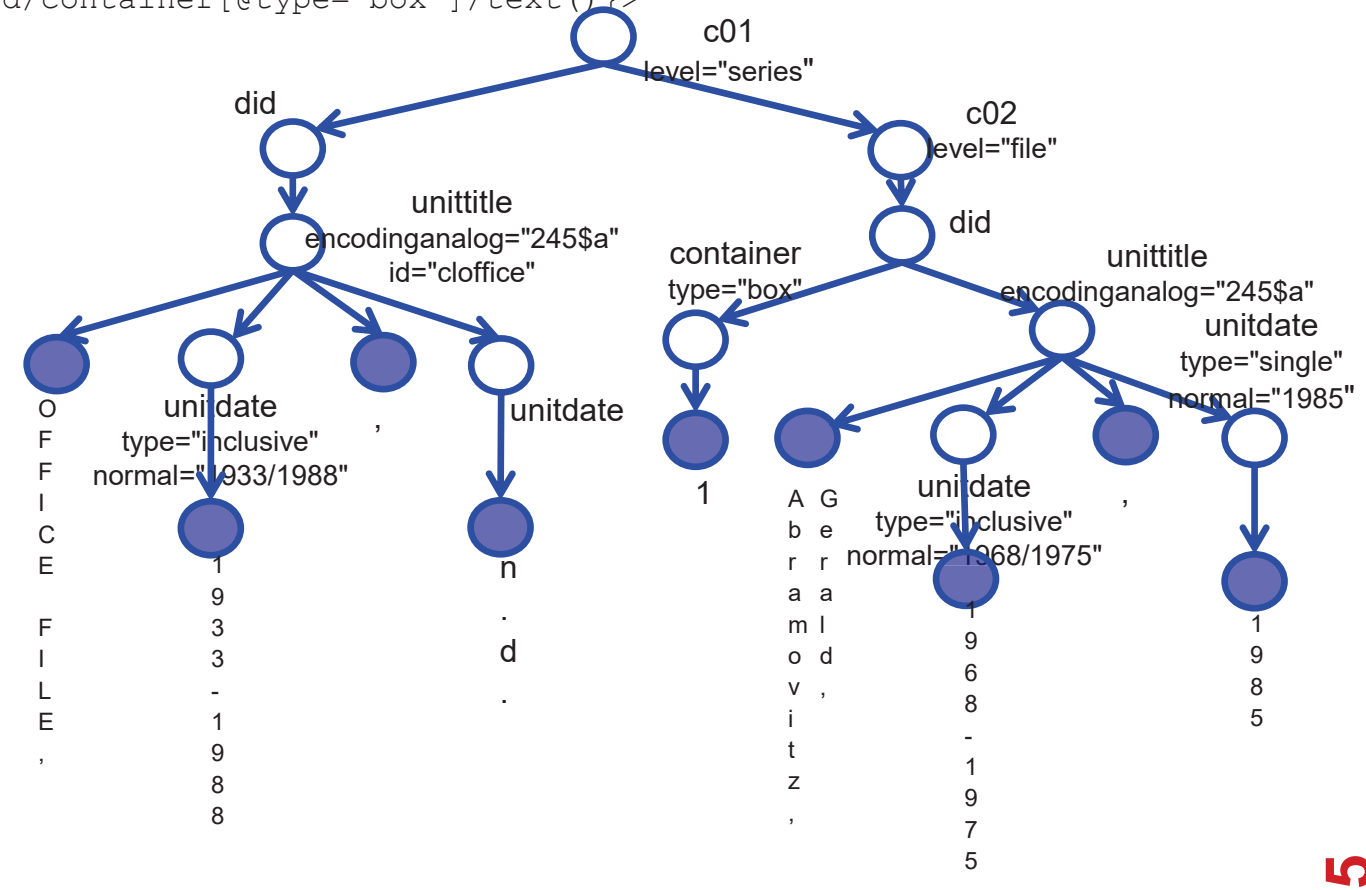
XQUERY EXAMPLE

Which items are stored in boxes and cover more than one date or date range?

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

XQUERY EXAMPLE

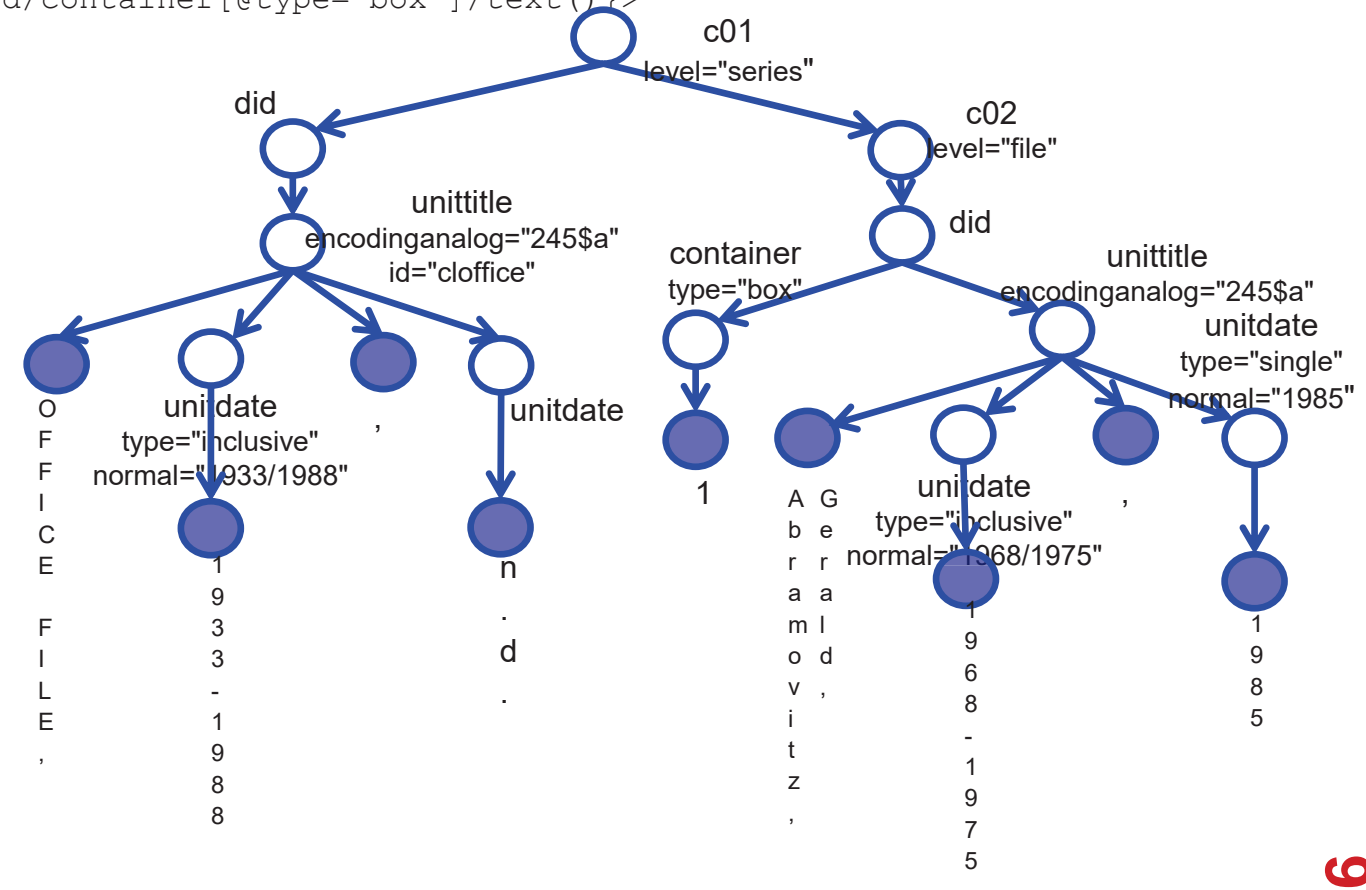
```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```



XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

<multiDatedBoxes>

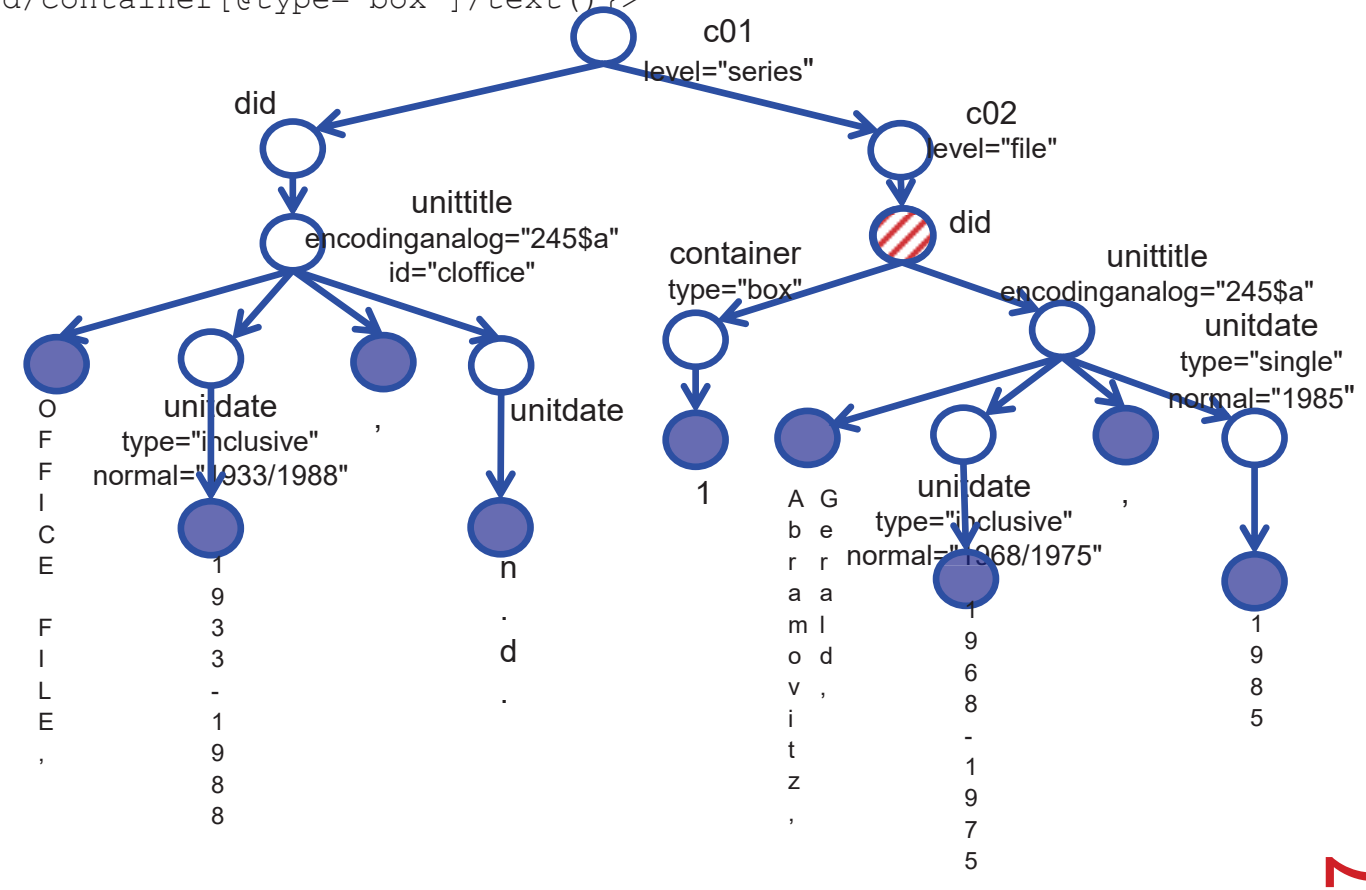


</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

<multiDatedBoxes>

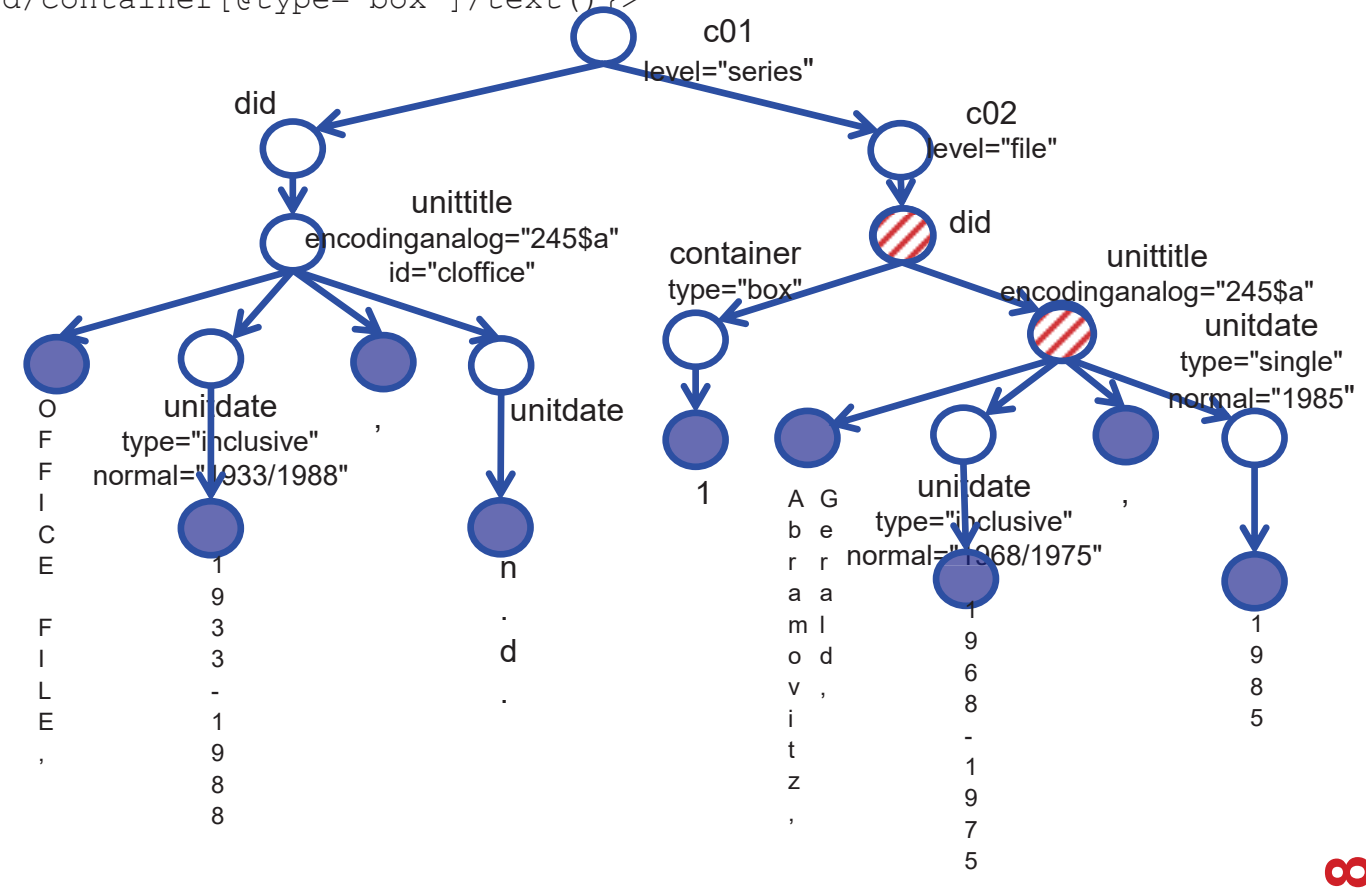


</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

<multiDatedBoxes>

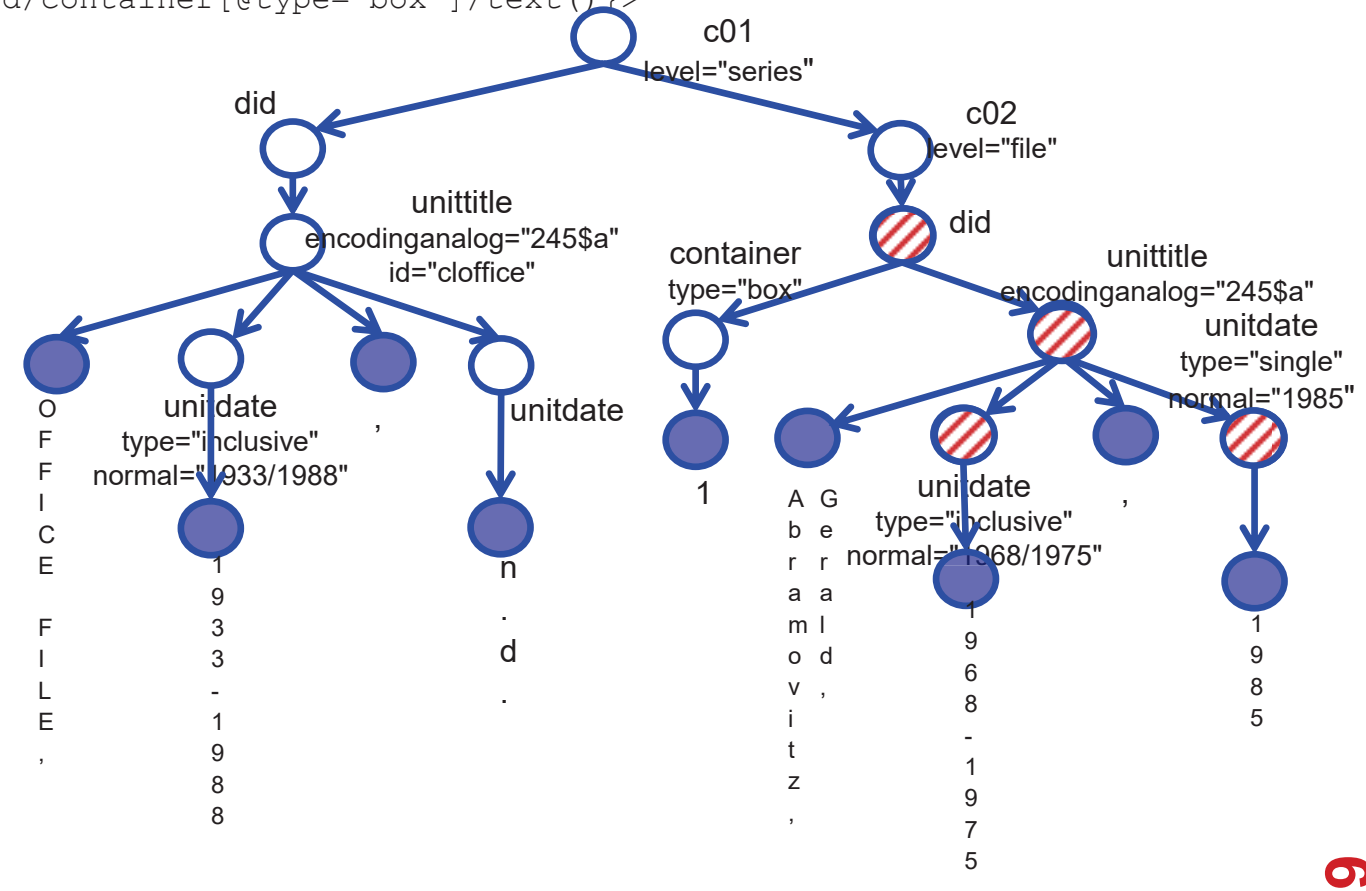


</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

<multiDatedBoxes>

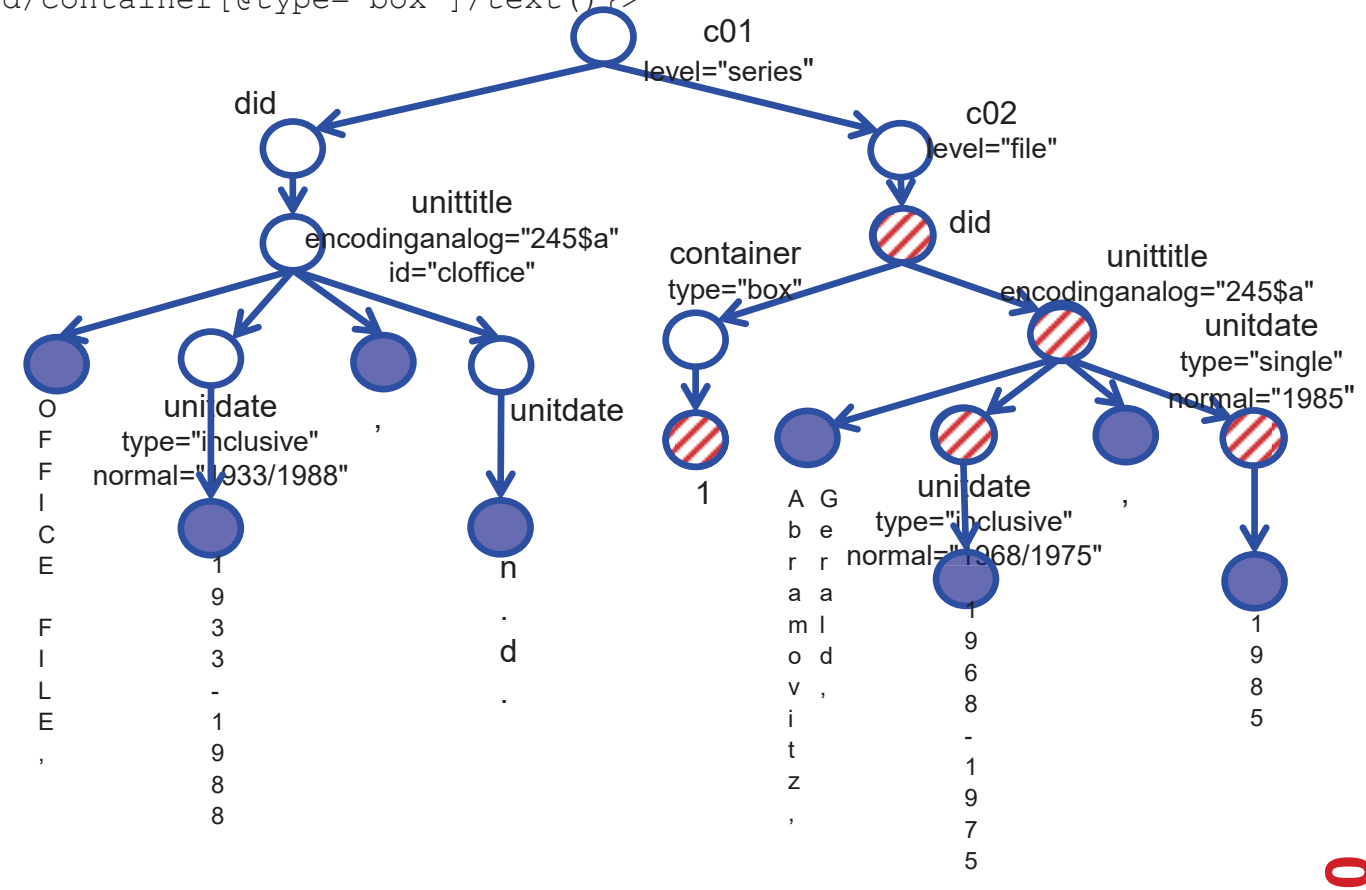


</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

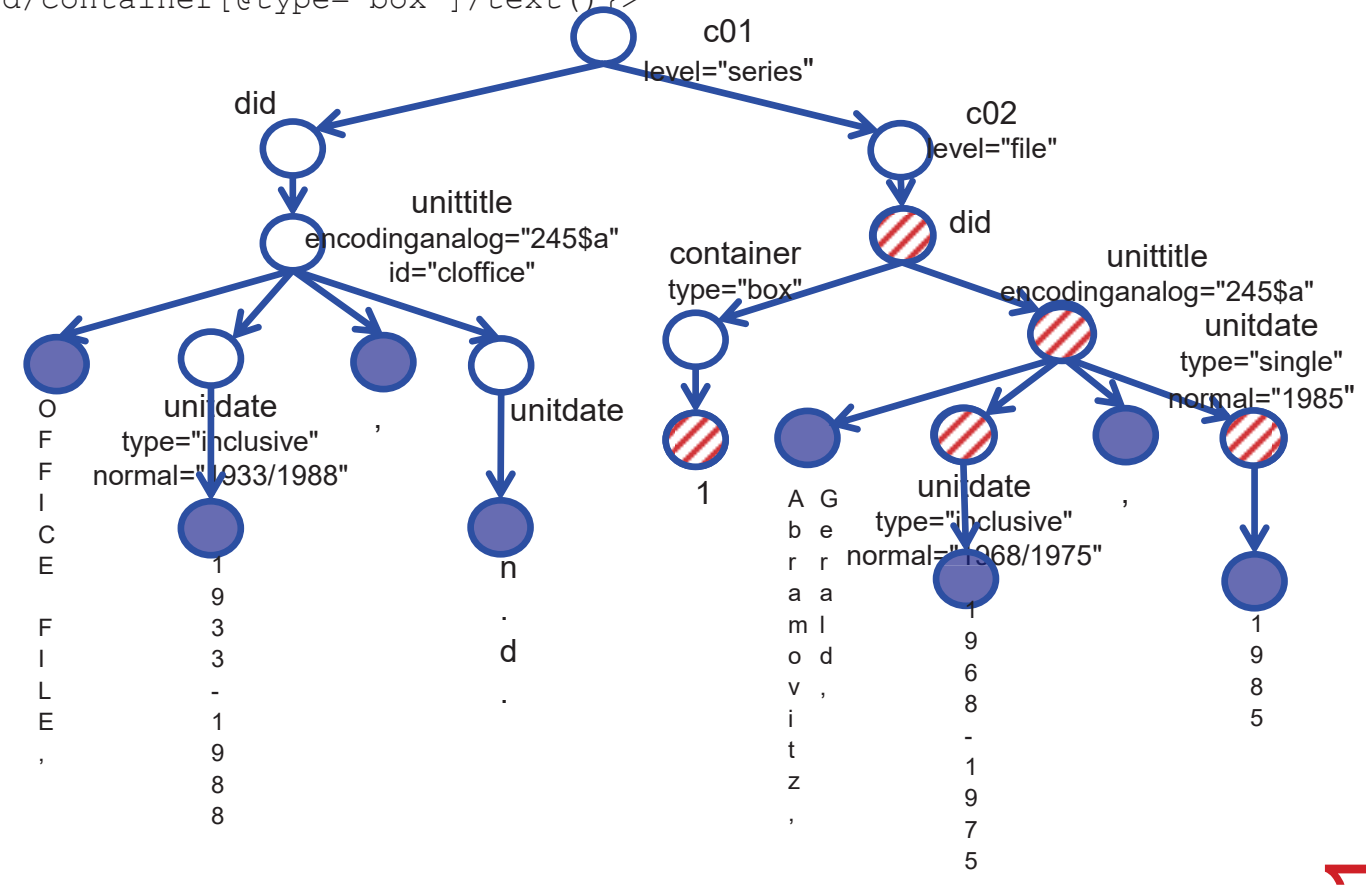
<multiDatedBoxes>



</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```



<multiDatedBoxes>

<box num=1>

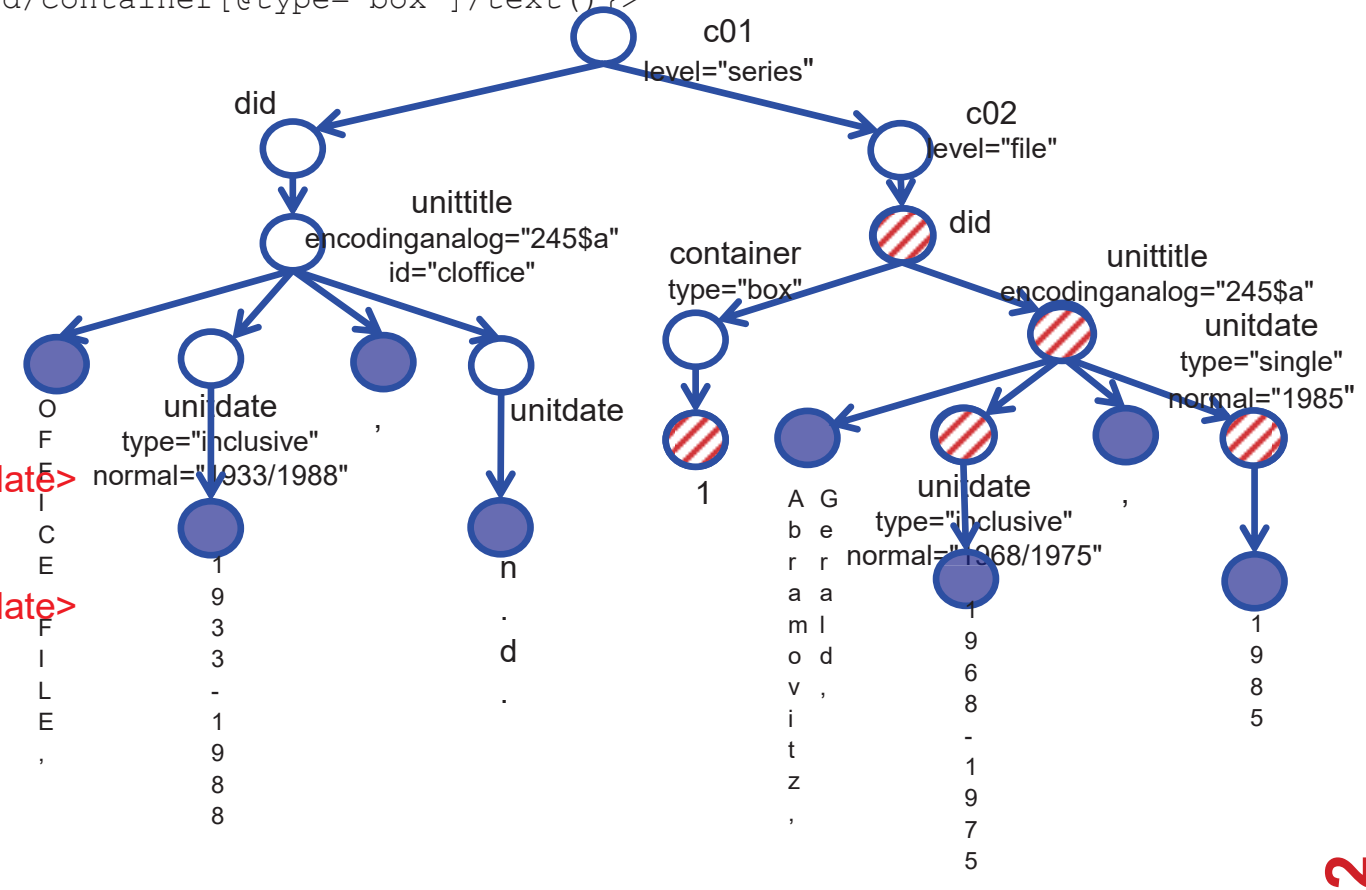
</box>

</multiDatedBoxes>

XQUERY EXAMPLE

```
<multiDatedBoxes>
{for $did in doc(MyArchive.xml)//did[container[@type="box"]],
     $ut in $did/unittitle
 let $ud = $ut/unitdate[@normal]
 where count($ud) > 1
 order by $did/container[@type="box"]/text()
 return <box num={$did/container[@type="box"]/text()}>
        {$ut}
        </box>
}
</multiDatedBoxes>
```

```
<multiDatedBoxes>
  <box num=1>
    <unittitle ….>
      Abramovitz, ….
      <unitdate ….> … </unitdate>
      ,
      <unitdate ….> … </unitdate>
    </unittitle>
  </box>
</multiDatedBoxes>
```



XSLT

Functional language, converting source tree into result tree

- *push-pull* model
- based on matching source tree patterns using XPath

Program is a *stylesheet*

- set of *templates*
  - what to match
  - what actions to take when match is made
- precedence rules to decide template activation
- *modes* to force rule application in phases

Default templates:

```
<xsl:template match ="*|/" mode="#all">
    <xsl:apply-templates/>
</xsl:template>

<xsl:template match="text()|@*" mode="#all">
    <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="processing-instruction()|comment()" mode="#all"/>
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <archive>
   <xsl:apply-templates select="@level" />
   </archive>
</xsl:template>

<xsl:template match="@level">
   <xsl:element name="{@level}"/>
      <xsl:apply-templates />
   </xsl:element>
</xsl:template>
```

```
<xsl:template match="unittitle|container">
   <xsl:copy>
      <xsl:copy-of select="@*" />
      <xsl:apply-templates />
   </xsl:copy>
</xsl:template>
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <archive>
   <xsl:apply-templates select="@level" />
   </archive>
</xsl:template>

<xsl:template match="@level">
   <xsl:element name="{@level}"/>
      <xsl:apply-templates />
   </xsl:element>
</xsl:template>
```

```
<xsl:template match="unittitle|container">
   <xsl:copy>
      <xsl:copy-of select="@*" />
      <xsl:apply-templates />
   </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
   <unitdate>
      <xsl:value-of select="@normal" />
   <unitdate>
</xsl:template>
```

XSLT EXAMPLE

<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <archive>
   <xsl:apply-templates select="@level" />
   </archive>
</xsl:template>

<xsl:template match="@level">
   <xsl:element name="{@level}"/>
      <xsl:apply-templates />
   </xsl:element>
</xsl:template>
```

```
<xsl:template match="unittitle|container">
   <xsl:copy>
      <xsl:copy-of select="@*" />
      <xsl:apply-templates />
   </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
   <unitdate>
      <xsl:value-of select="@normal" />
   <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
   <unitdate>
      <xsl:apply-templates />
   <unitdate>
</xsl:template>
```

XSLT EXAMPLE

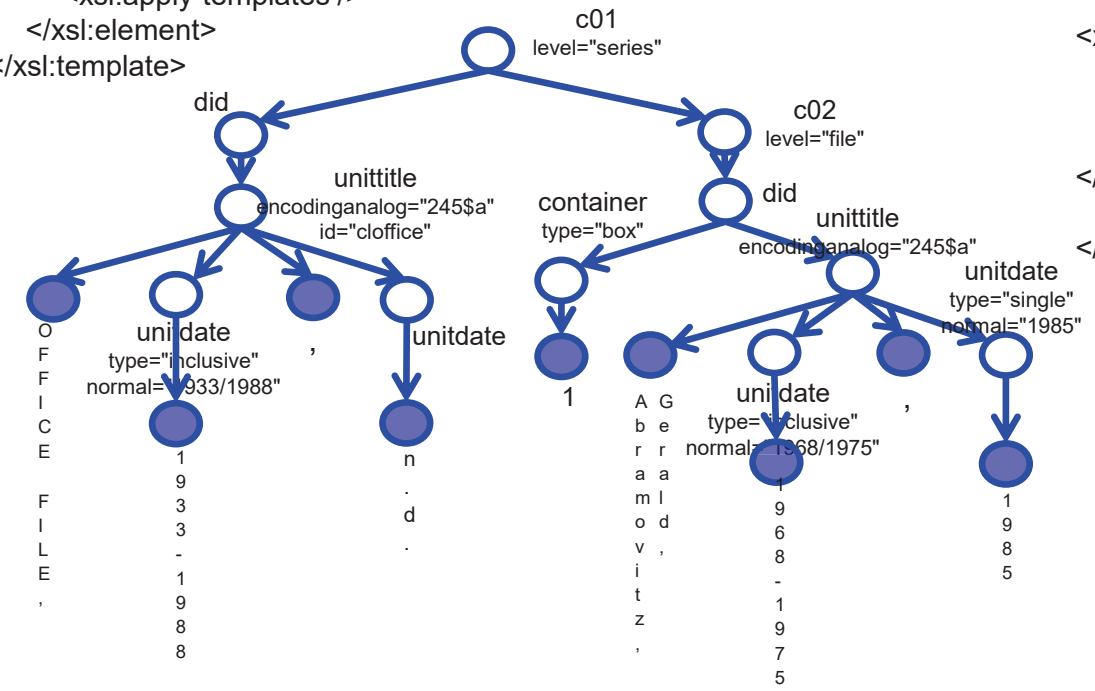
<c01 level="series">, <c02 level="file"> → <series>, <file>
<unitdate … normal="1933/1988">1933-1988 → <unitdate>1933/1988
<did> *deleted*

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <archive>
   <xsl:apply-templates select="@level" />
   </archive>
</xsl:template>

<xsl:template match="@level">
   <xsl:element name="{@level}"/>
      <xsl:apply-templates />
   </xsl:element>
</xsl:template>
```

```
<xsl:template match="unittitle|container">
   <xsl:copy>
      <xsl:copy-of select="@*" />
      <xsl:apply-templates />
   </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
   <unitdate>
      <xsl:value-of select="@normal" />
   <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
   <unitdate>
      <xsl:apply-templates />
   <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

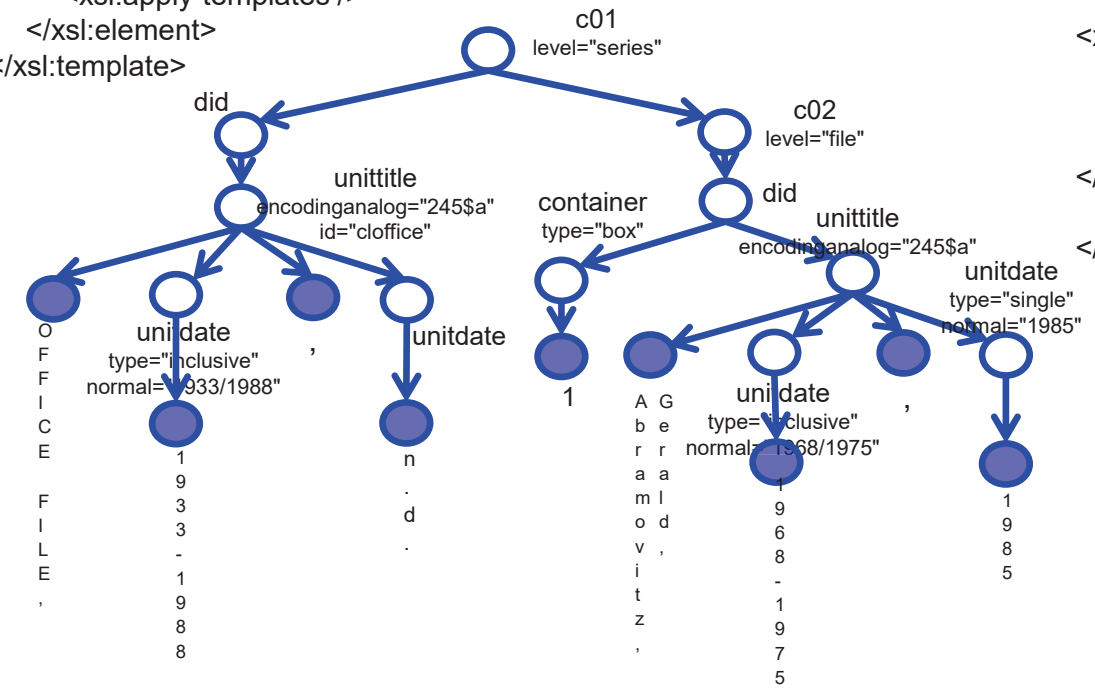
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

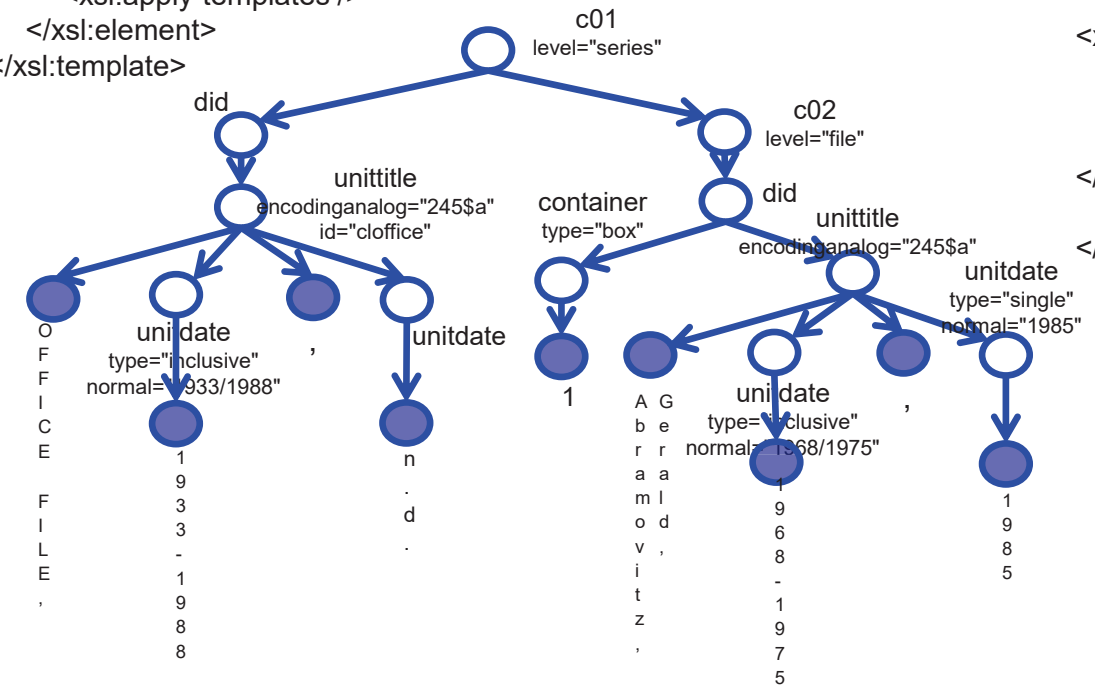
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

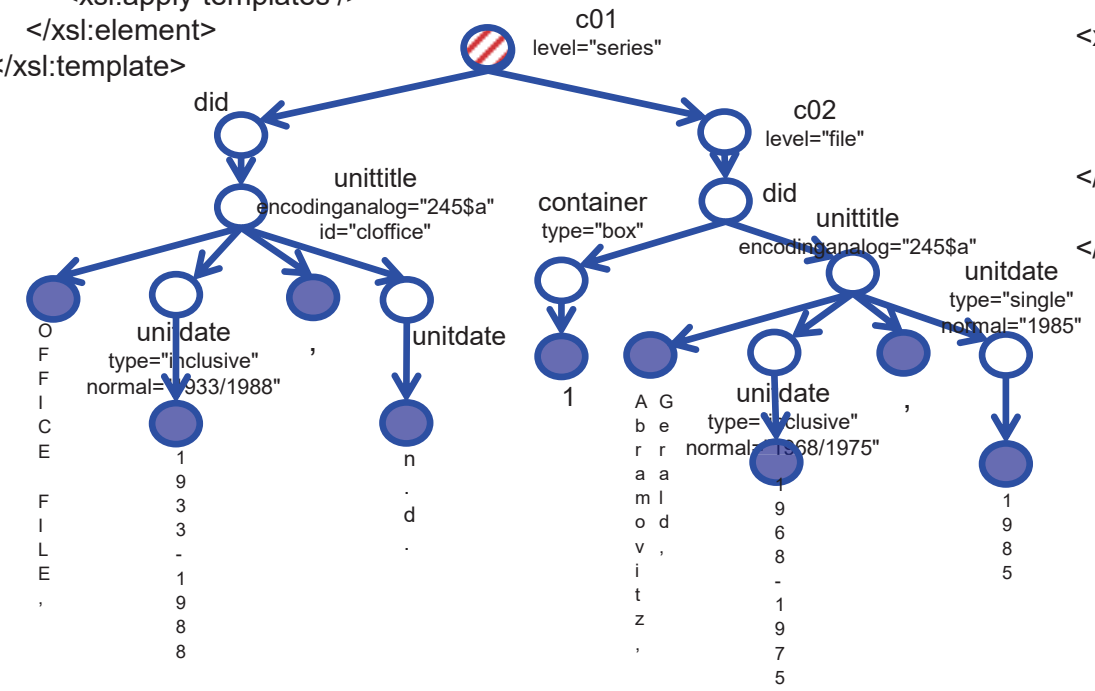



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

archive

XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

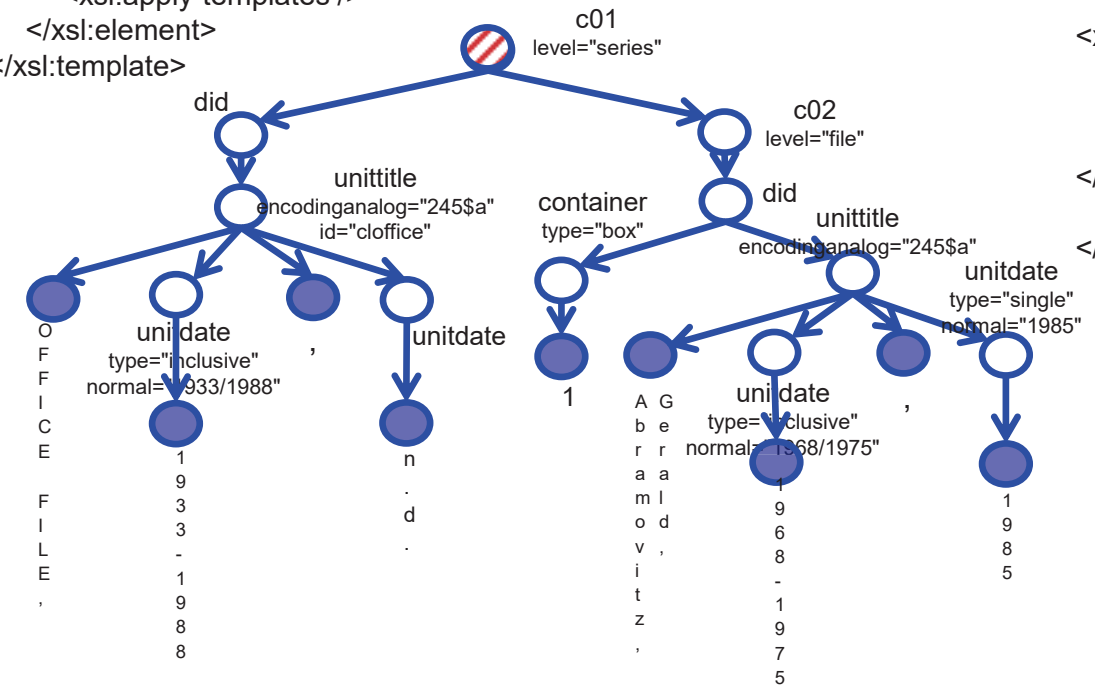



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

archive

XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

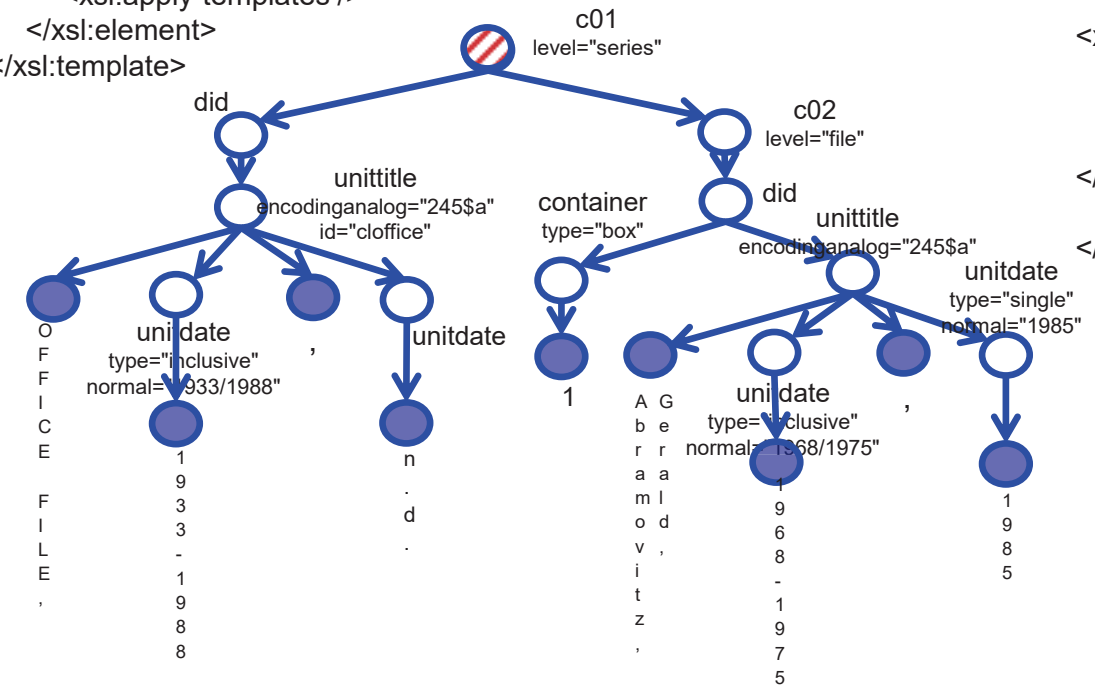



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

archive

XSLT EXAMPLE

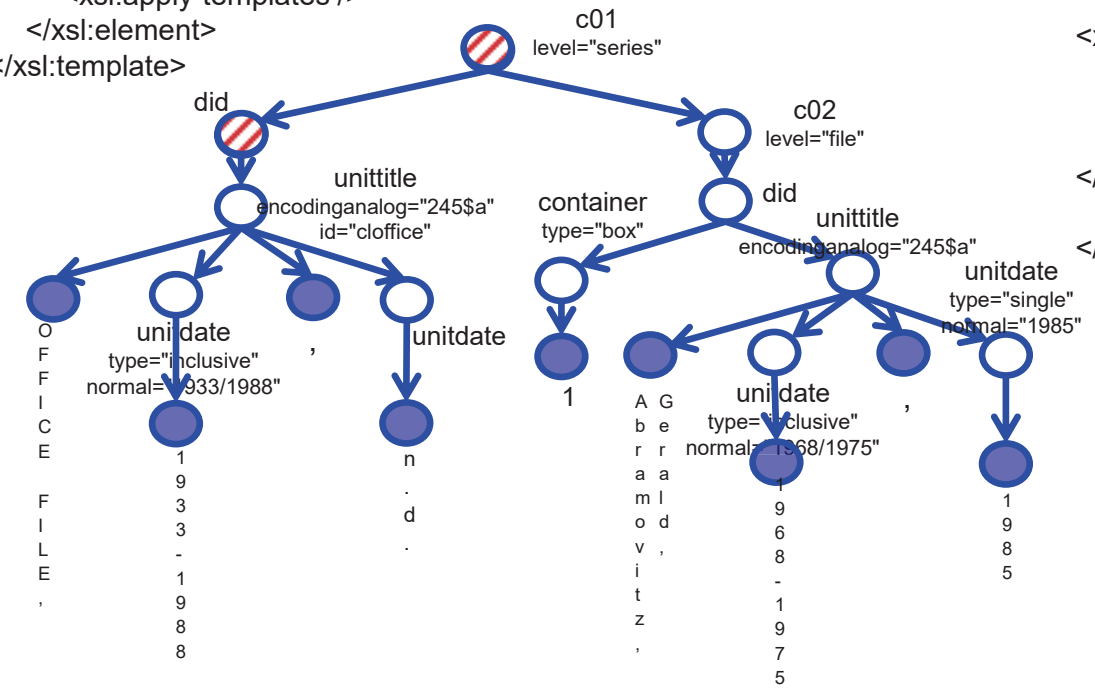
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

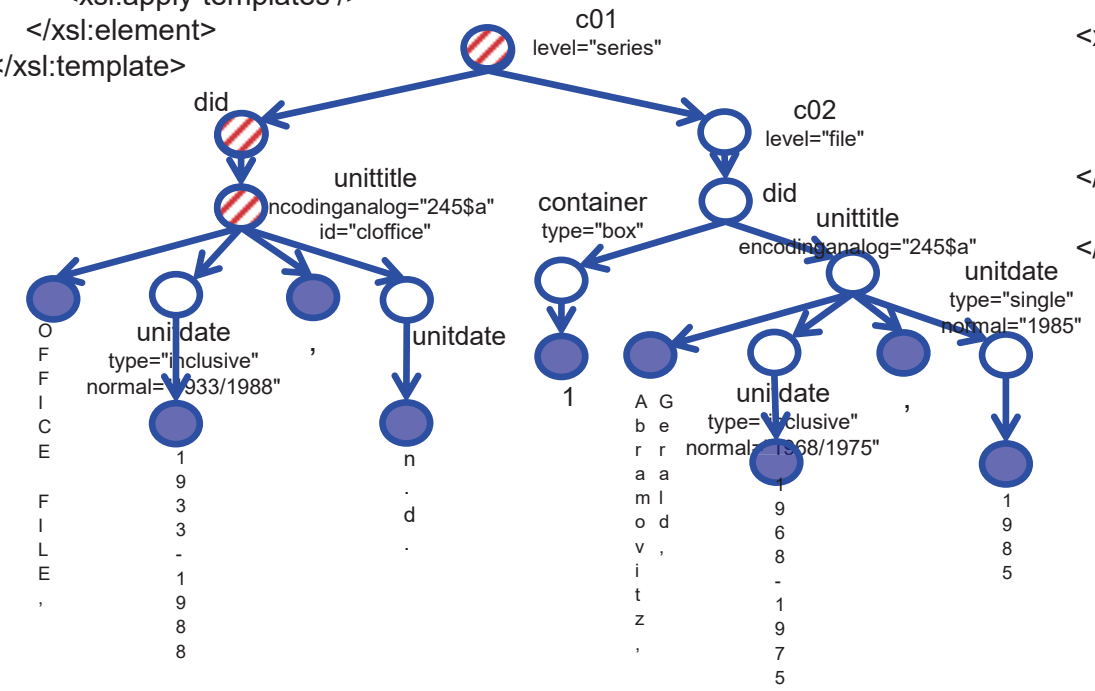
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

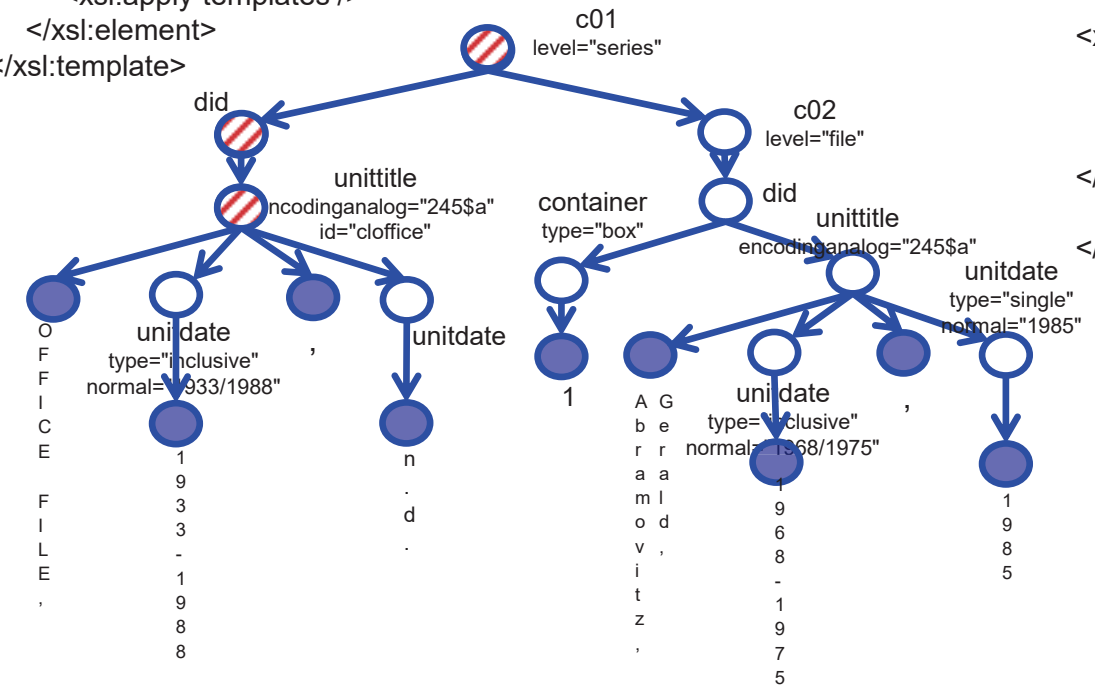
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

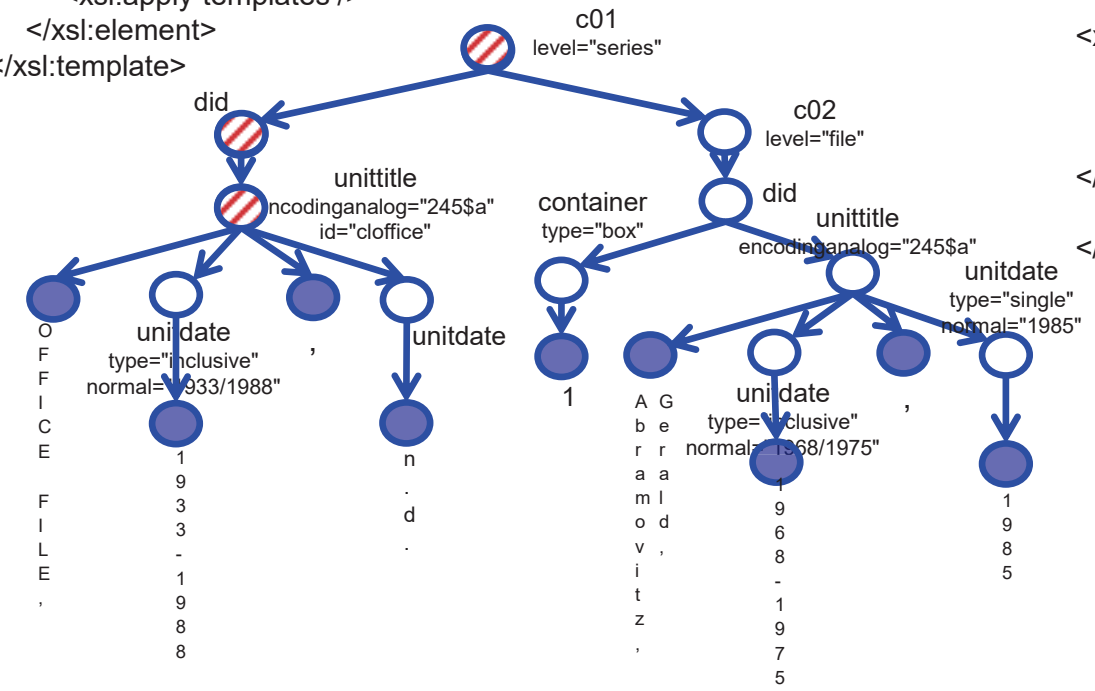
```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```



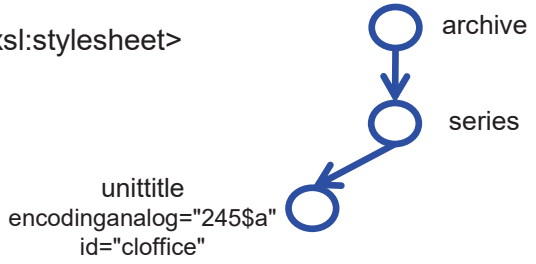

```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

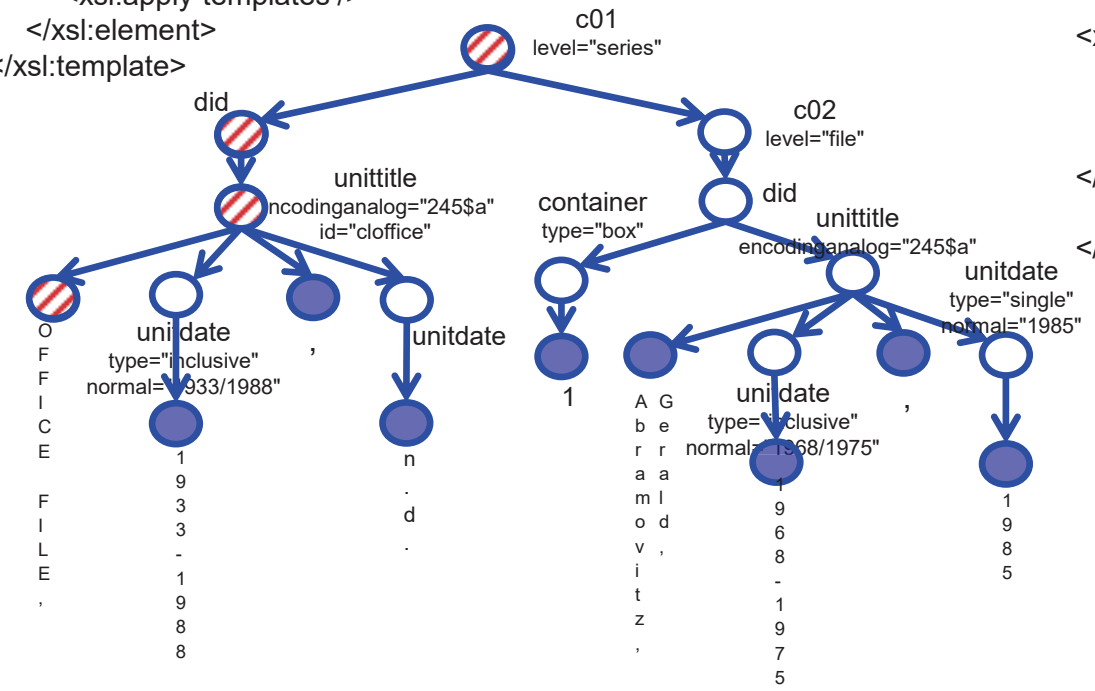



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

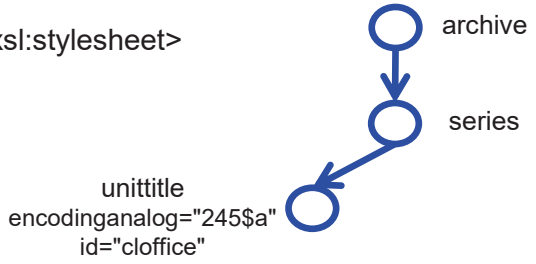


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

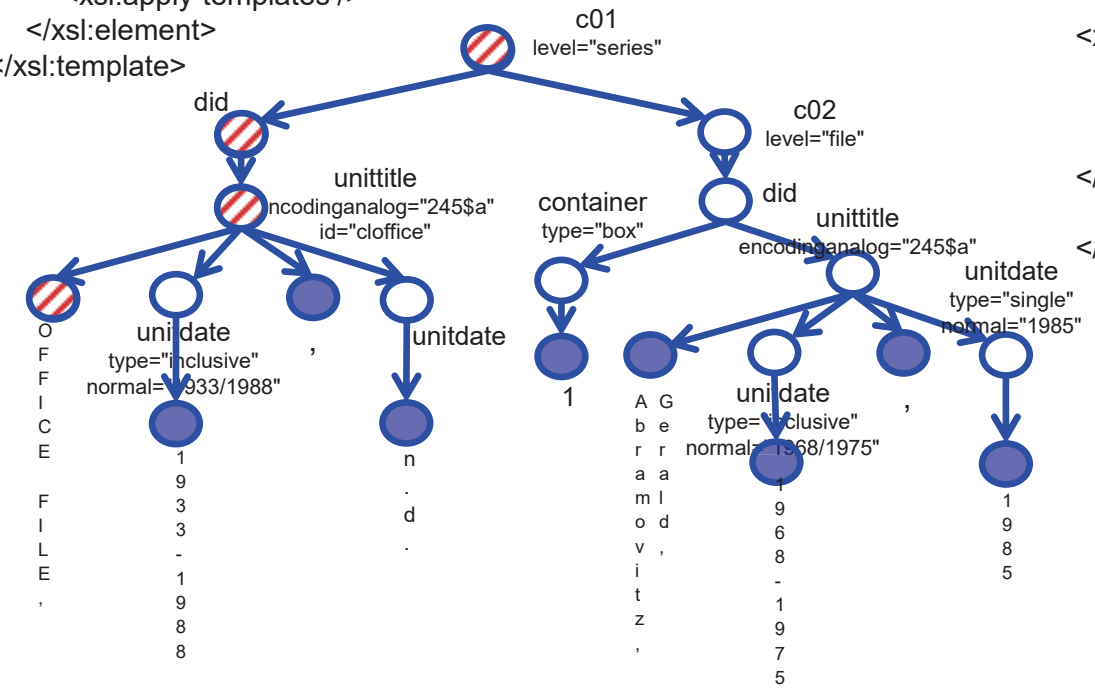



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

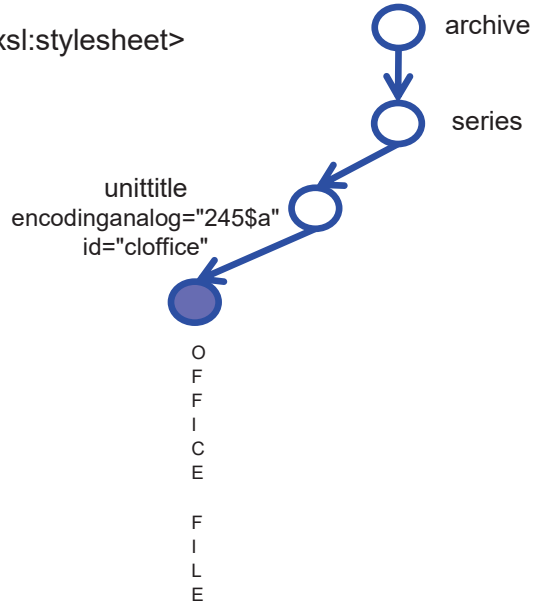


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

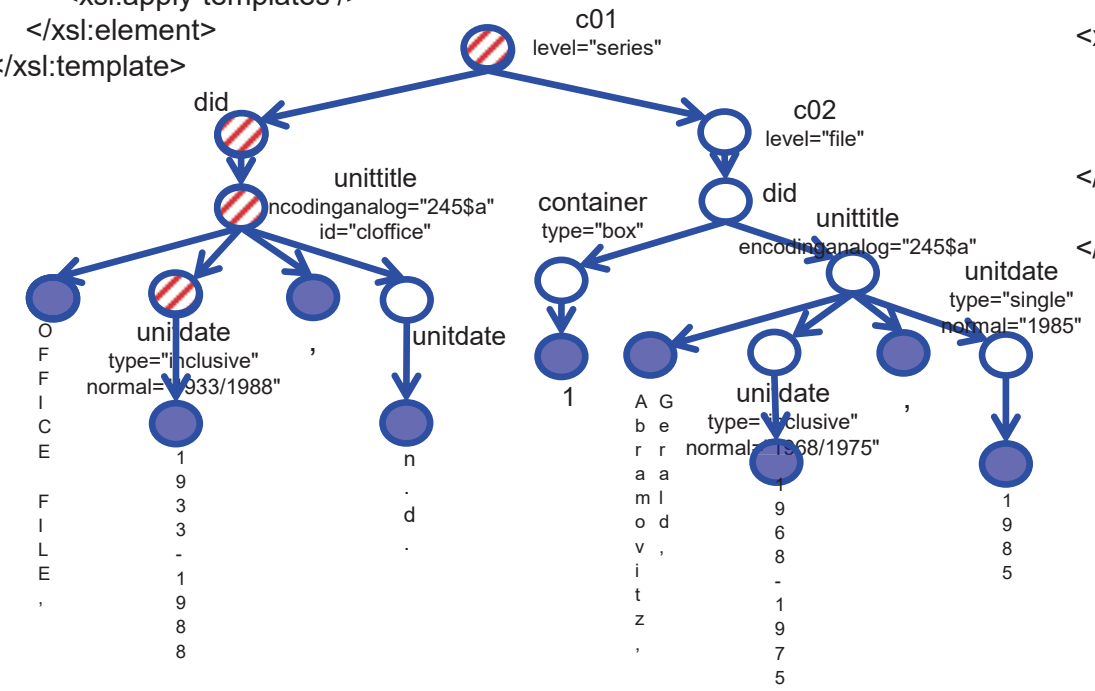



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

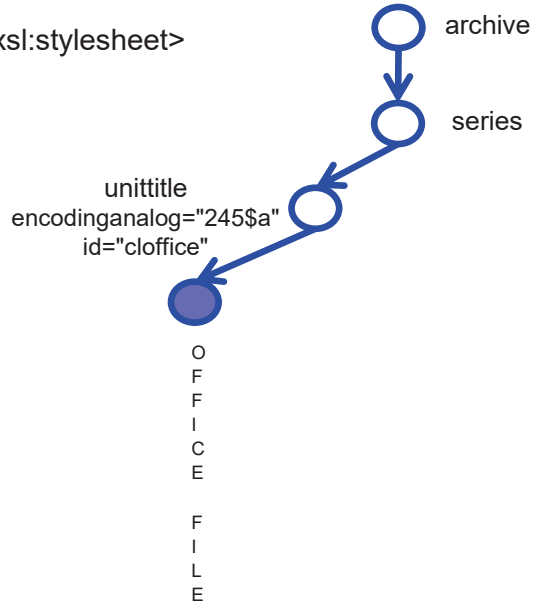


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

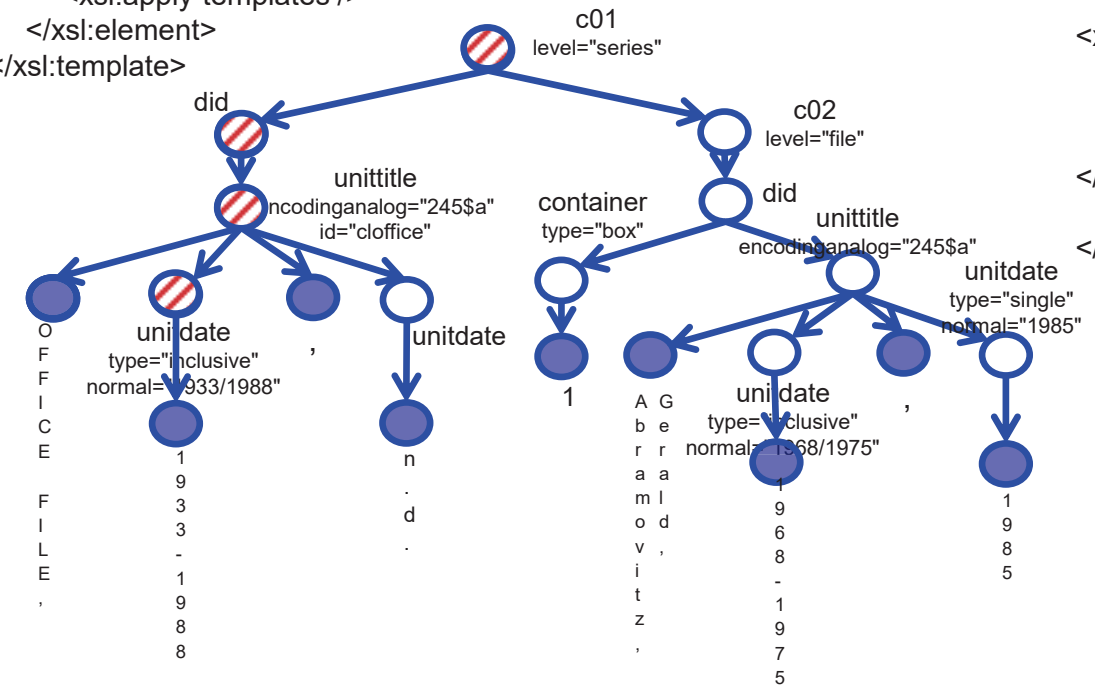



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

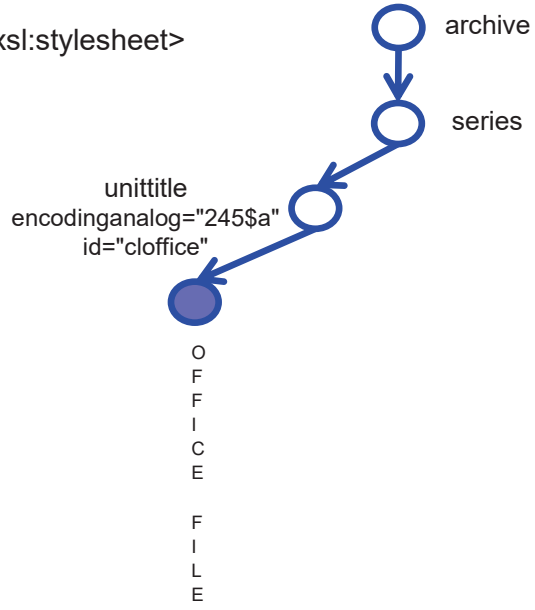


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

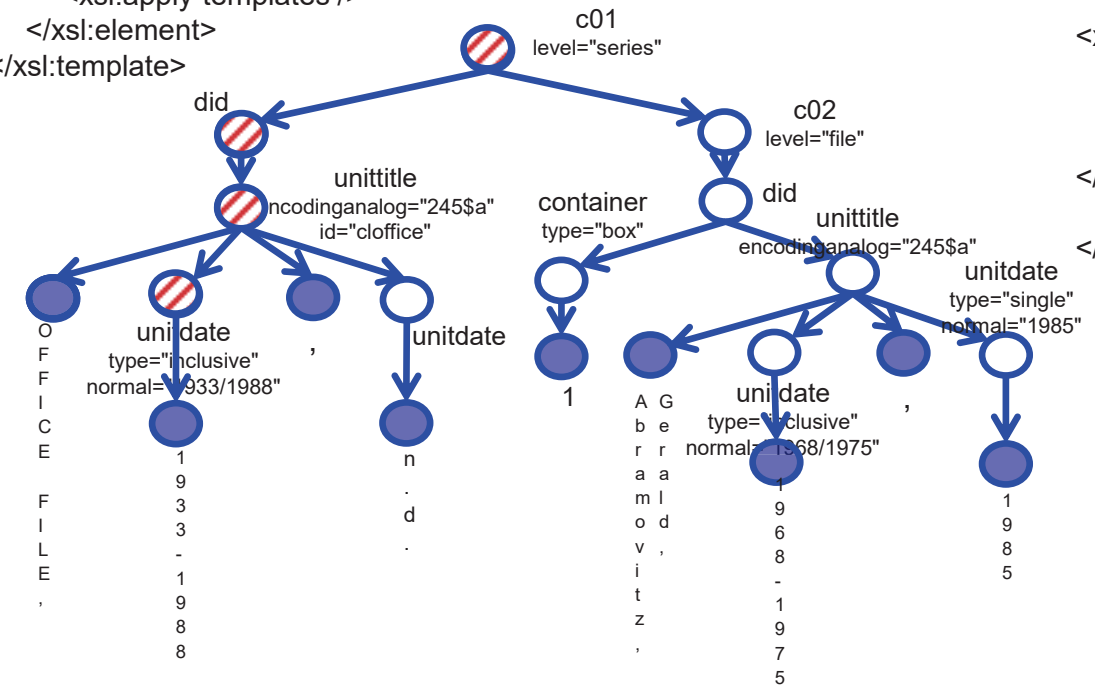



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

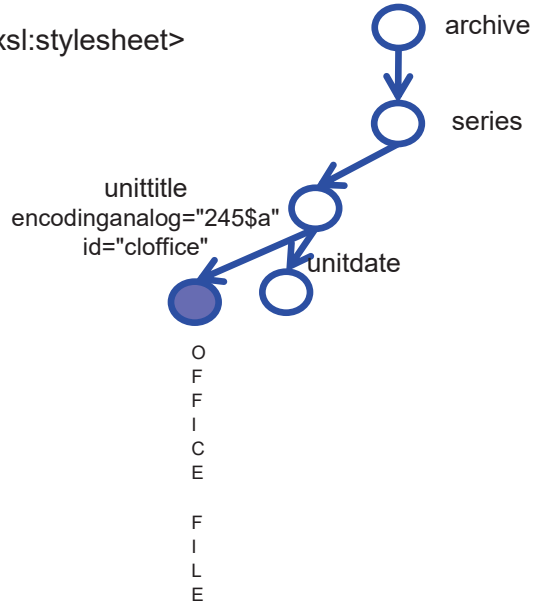


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

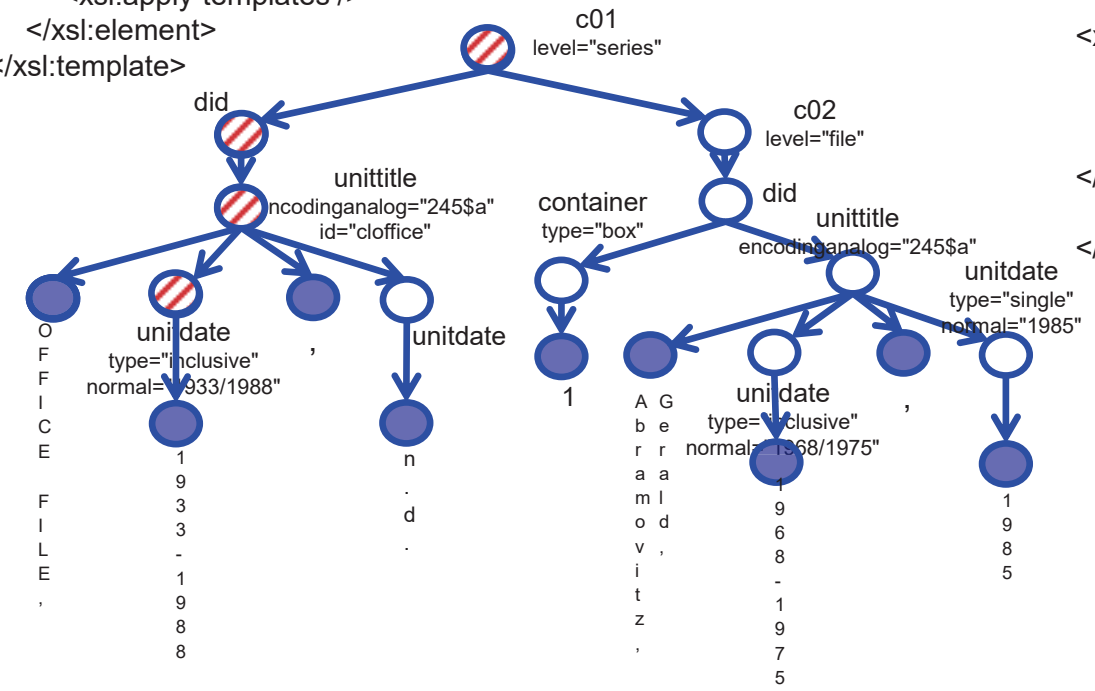



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

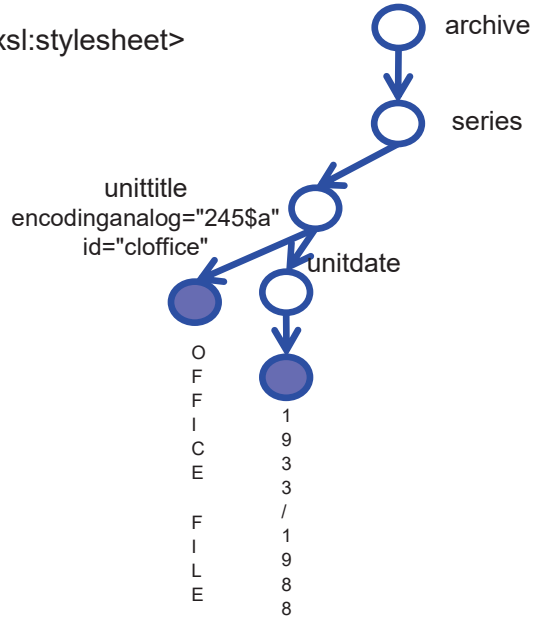


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

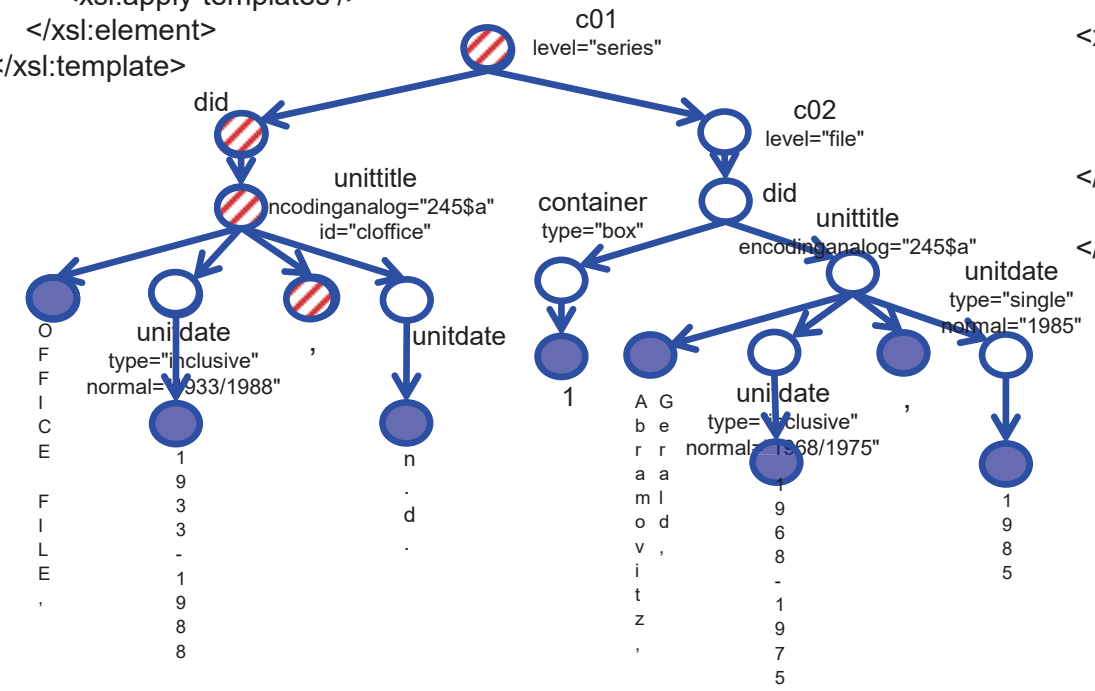



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

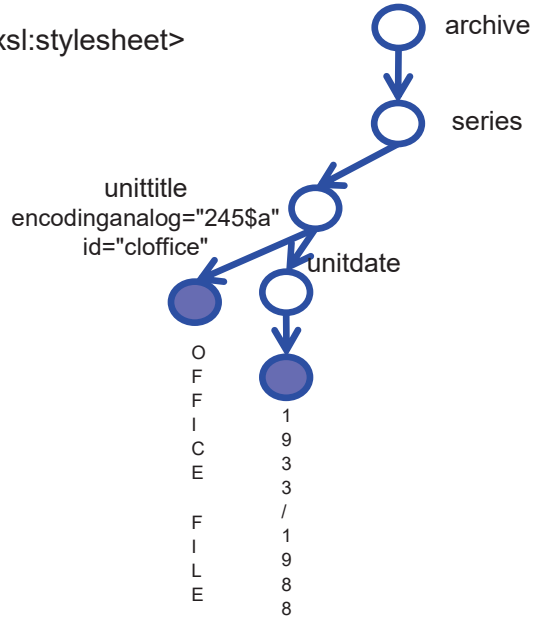


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

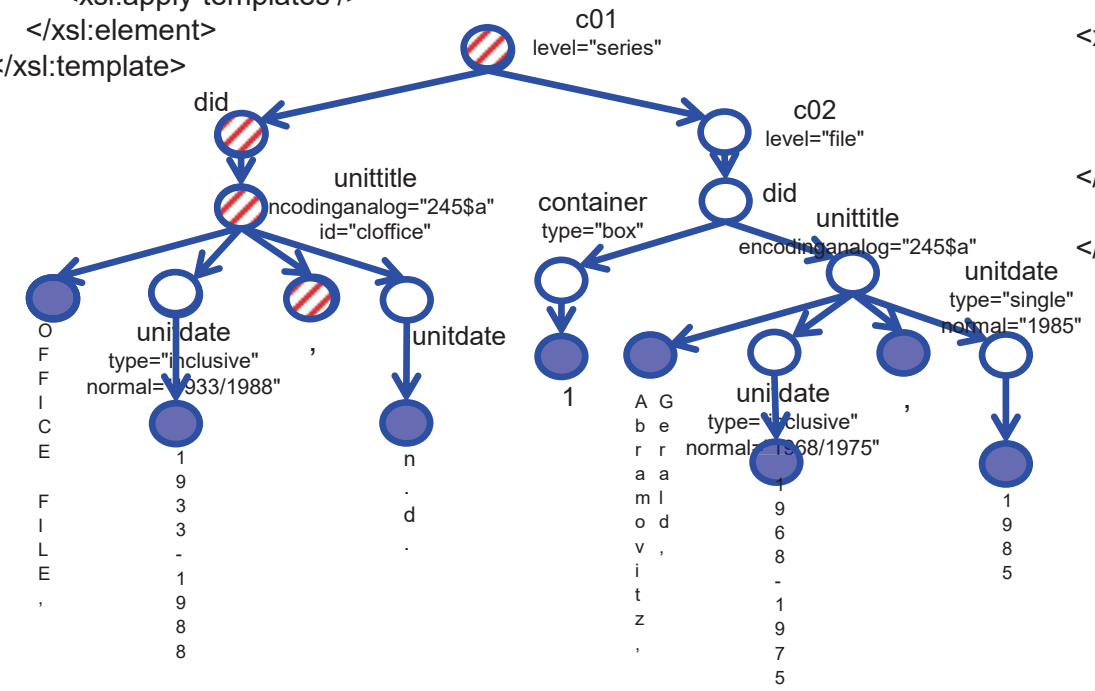



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

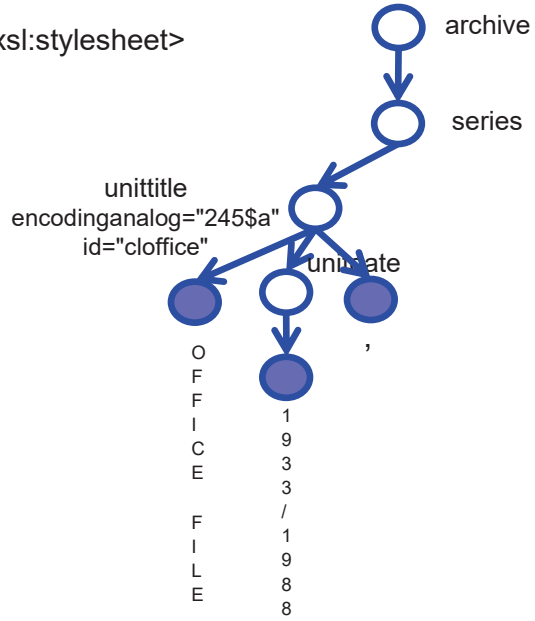


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

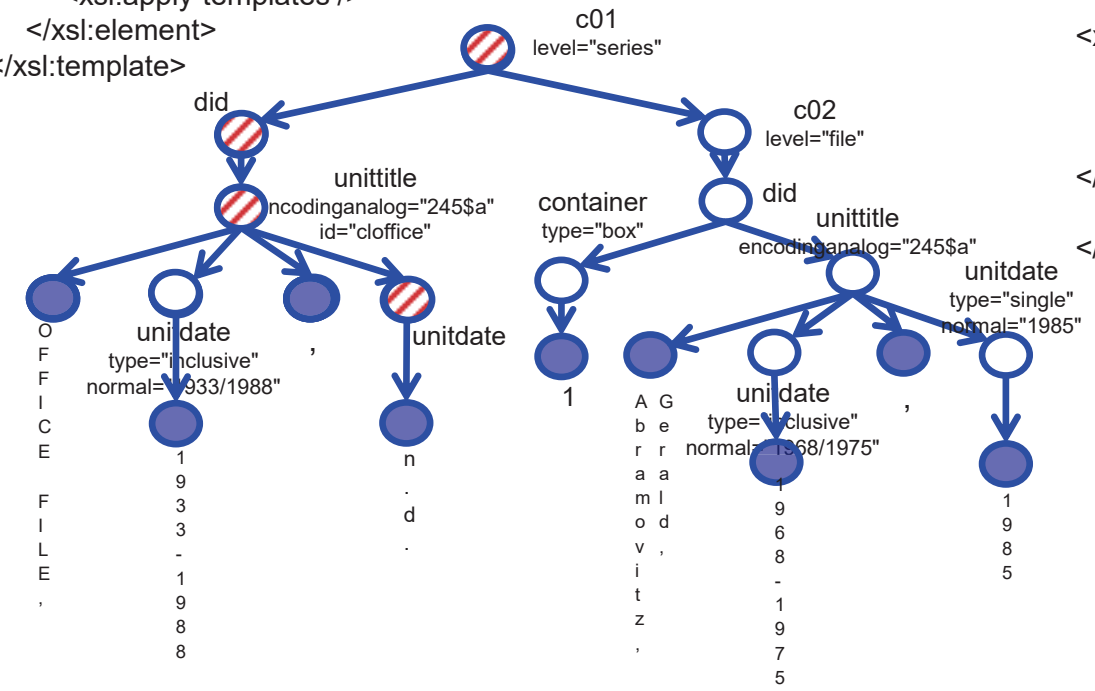



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

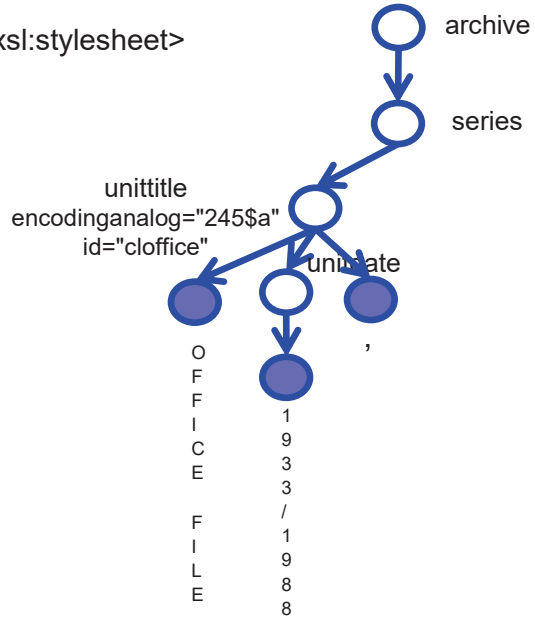


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

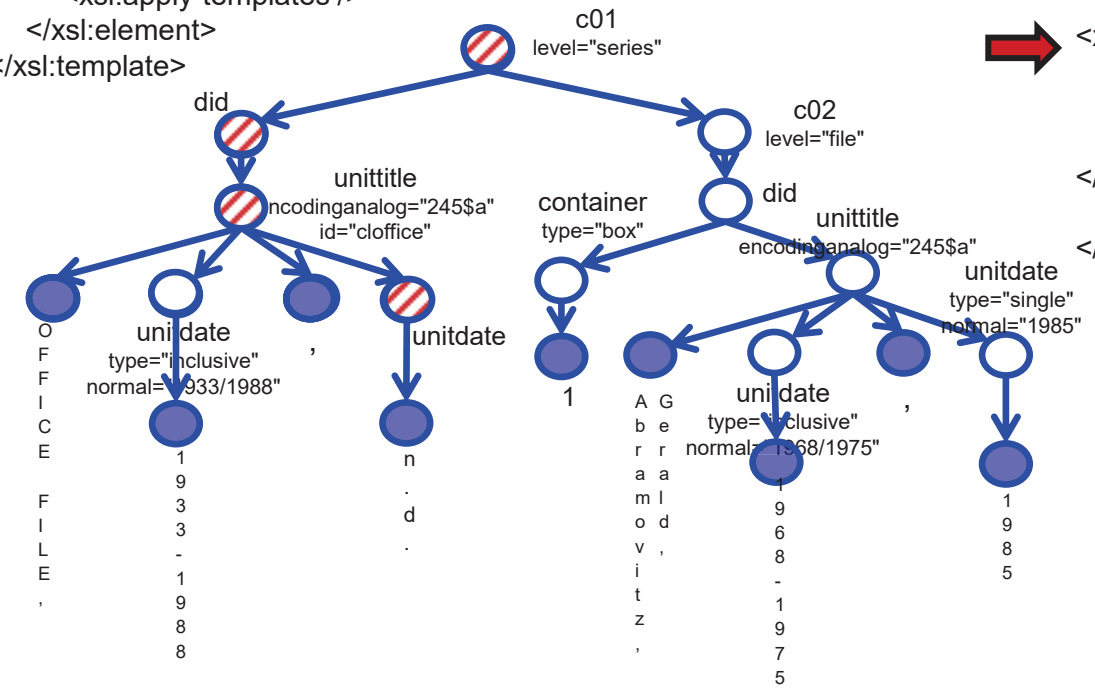



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

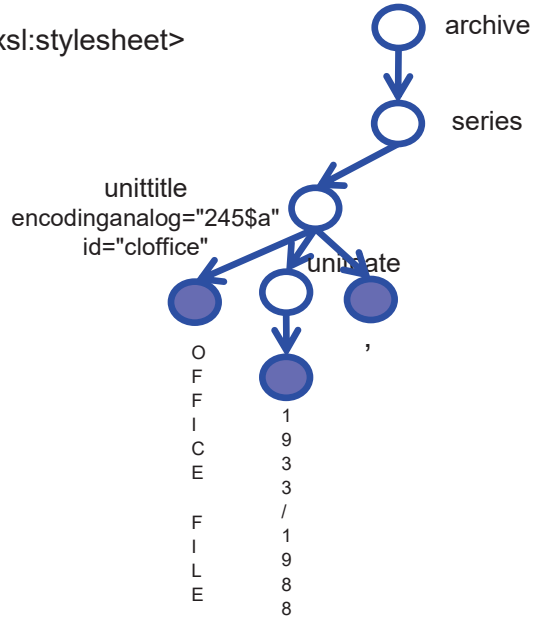


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

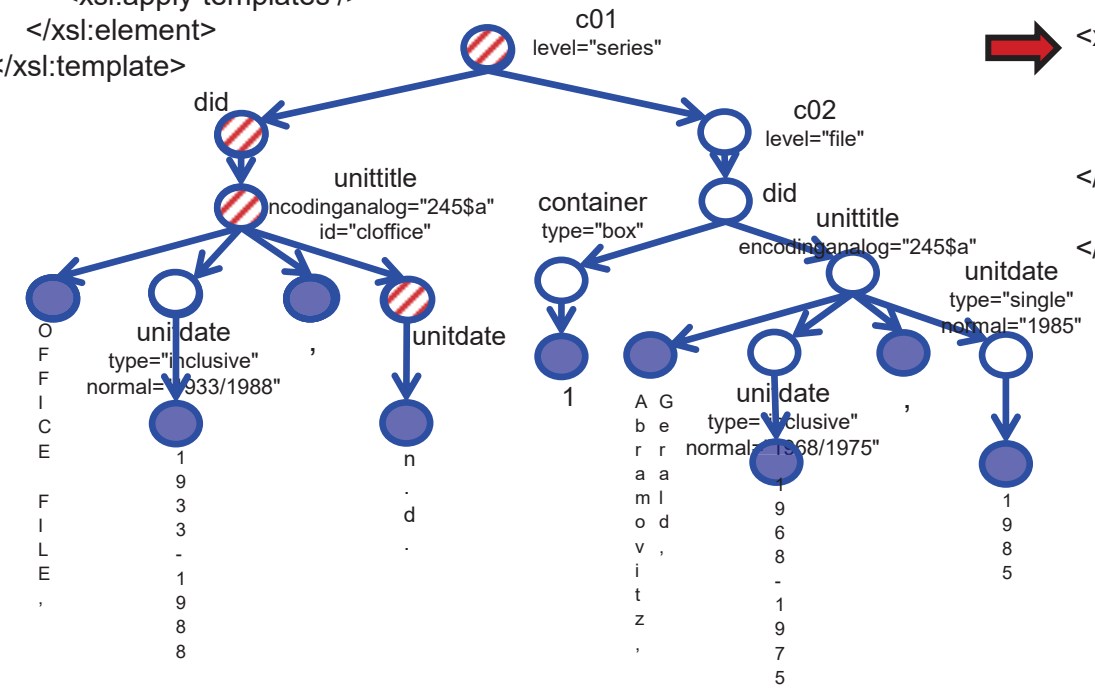



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

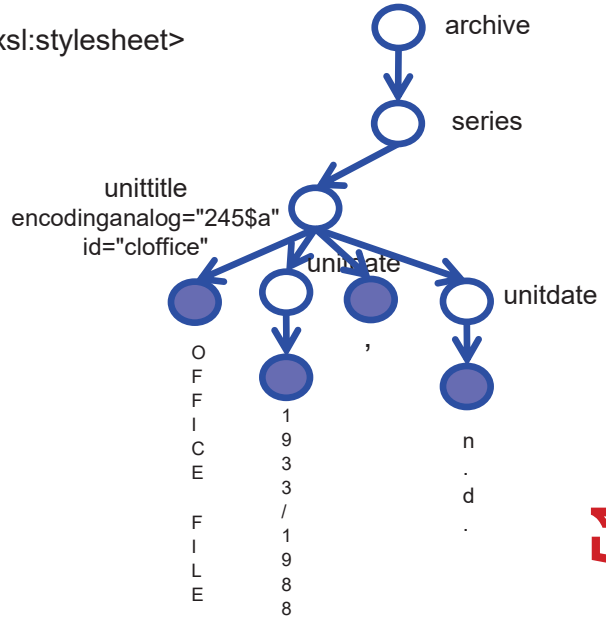


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

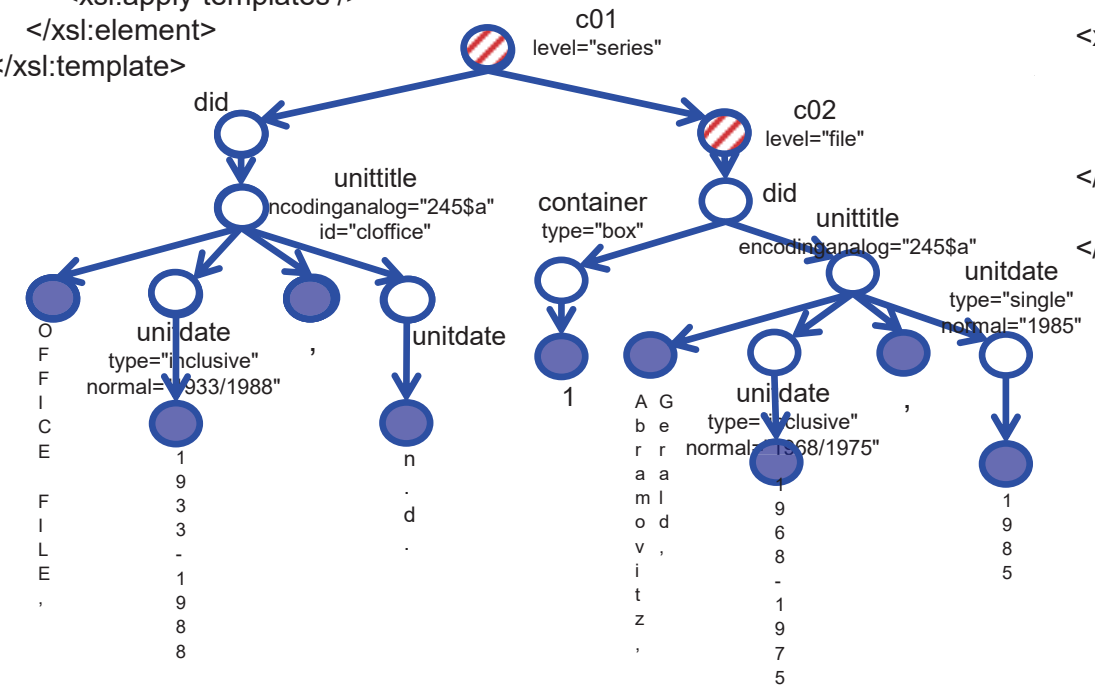



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

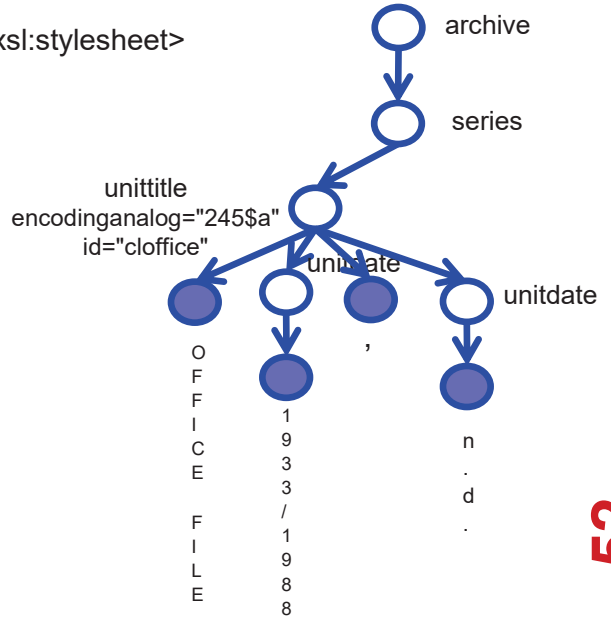


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

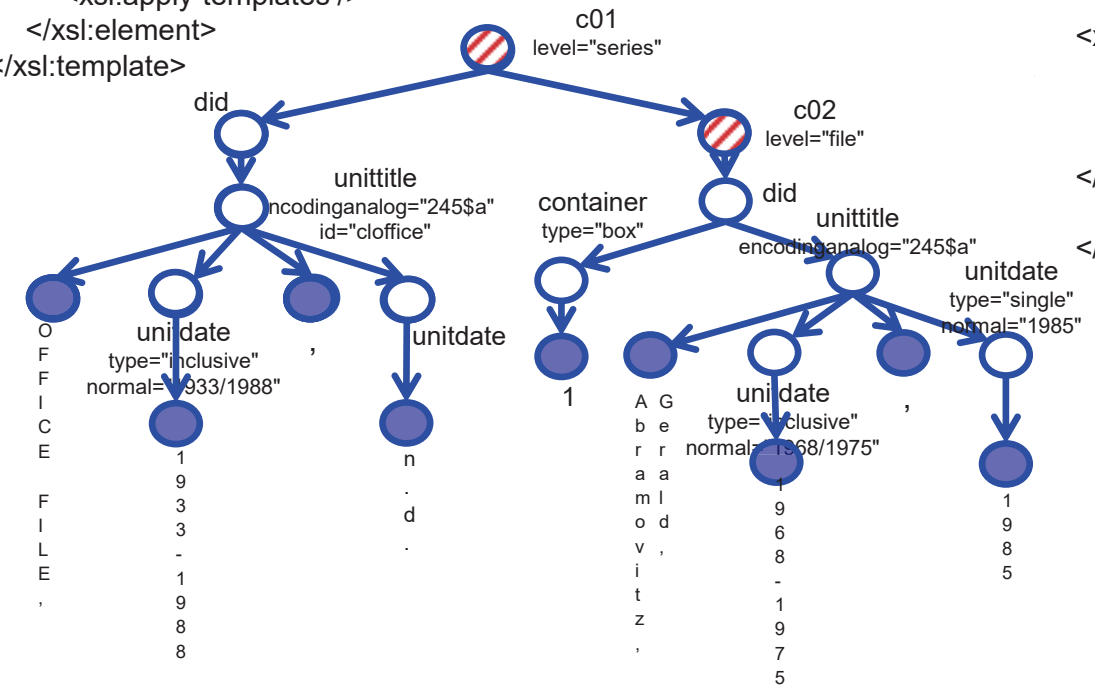



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

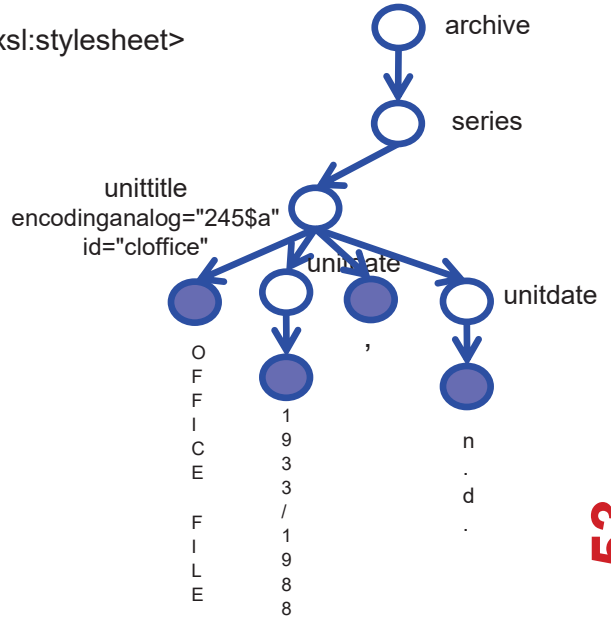


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```

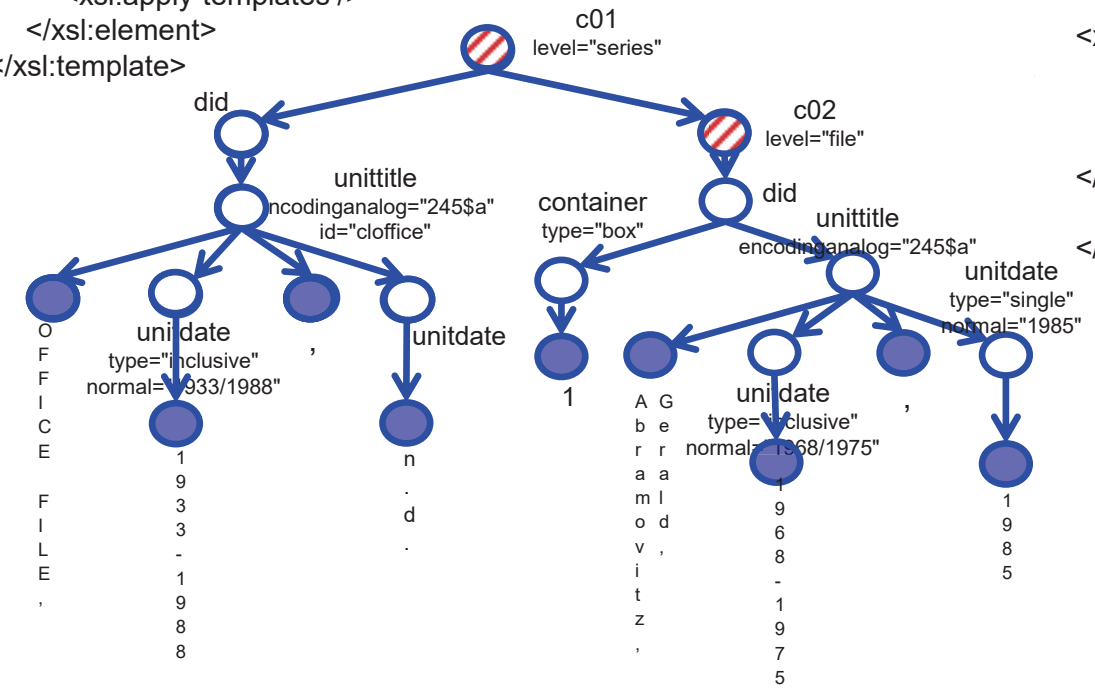



```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```

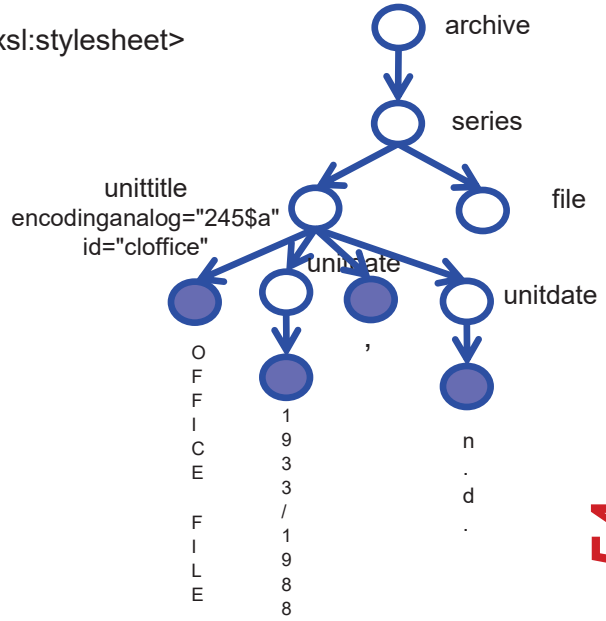


XSLT EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <archive>
    <xsl:apply-templates select="@level" />
    </archive>
</xsl:template>

<xsl:template match="@level">
    <xsl:element name="{@level}"/>
        <xsl:apply-templates />
    </xsl:element>
</xsl:template>
```




```
<xsl:template match="unittitle|container">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:apply-templates />
    </xsl:copy>
</xsl:template>

<xsl:template match="unitdate[@normal]">
    <unitdate>
        <xsl:value-of select="@normal" />
    <unitdate>
</xsl:template>

<xsl:template match="unitdate[not(@normal)]">
    <unitdate>
        <xsl:apply-templates />
    <unitdate>
</xsl:template>

</xsl:stylesheet>
```



XQUERY AND XSLT PROCESSOR

http://saxon.sourceforge.net/

"Saxon-HE (home edition) is an open source product available under the Mozilla Public License. It provides implementations of XSLT 2.0, XQuery 1.0, and XPath 2.0 at the basic level of conformance defined by W3C. It is available for both Java and .NET.

"Saxon-PE (professional edition) is a commercial product available at modest prices from Saxonica Limited. It adds a number of features to Saxon-HE, including support for Saxon extensions and extensibility mechanisms, support for new features defined in XQuery 3.0 including higher-order functions, and easier configuration of features such as localization for different languages, and support for external object models such as JDOM, XOM, and DOM4J.

"Saxon-EE (enterprise edition) is the fully-featured commercial product, essentially a renaming of the previous Saxon-SA to reflect the fact that it now offers much more than just schema-awareness. As well as a fully conformant XSD 1.0 and XSD 1.1 schema processor, and support for schema-aware XSLT and XQuery processing, it offers many other features including streaming in XSLT and XQuery, support for XQuery updates, and advanced query optimizer, compilation of XQuery and XSLT code to Java bytecode, and much more. For full details, see the Saxonica web site."

CSS — CASCADING STYLE SHEETS

Light transformations to format XML for the Web

- convert *descriptive* markup to *presentation* markup
- attach formatting rules to XML elements

Components

- selector: which element is affected
- declaration: what formatting to apply
  - property (e.g., font, background, border, positioning)
  - value to use for that property

Attach stylesheet to document

- <?xml-stylesheet type="text/css" href="myStylesheet.css" ?>

Processor included with modern browsers

CSS EXAMPLE

```
c01 {
        display:block;
         font-family: helvetica, verdana, sans-serif;
         font-size: 20px; padding: 20px; }

c02 {
        display:block; margin-top: 10px; }

did {
        display:block; }

container[type="box"]:before {
        content: "Box "; }

container + unittitle {
        margin-left: 80px; }
```

CSS EXAMPLE

```
c01 {
        display:block;
         font-family: helvetica, verdana, sans-serif;
         font-size: 20px; padding: 20px; }

c02 {
        display:block; margin-top: 10px; }

did {
        display:block; }

container[type="box"]:before {
        content: "Box "; }

container + unittitle {
        margin-left: 80px; }
```

OFFICE FILE, 1933-1988, n.d.

Box 1 Abramovitz, Gerald, 1968-1975, 1985

SGREP — STRUCTURED GREP

http://www.cs.helsinki.fi/u/jjaakkol/sgrep.html

Lightweight search tool

Based on text regions

- patterns define extents of regions
- XML component types (e.g, elements, attributes) pre-defined
- ancestor-descendant and parent-child patterns available
- uses M4 for applying pattern definitions

Define base patterns in a control file

- easy reuse

Query result

- count of matching regions
- contents of all matching regions
- list of pairs of offsets for all matching regions

SGREP EXAMPLE

Contents of EADdefs

```
define(ELEMENTS, ( stag($1) .. etag($1) ))
define(DATE, ( ELEMENTS("unitdate") ))
define(CHUNKS, ( ELEMENTS("c01") or ELEMENTS("c02") ))
define(FILETAG, ( (stag("*") containing first_bytes(1,CHUNKS))
                                    containing attvalue("file") ))
define(FILE, ( inner(CHUNKS containing FILETAG) ))
define(BOXTAG, ( stag("container") containing attvalue("box") ))
define(BOX, ( FILE containing BOXTAG ))
```

Query

```
sgrep  -g xml  -p m4  -f EADdefs                                    \
       -e '(BOX containing "Abramovitz")'  MyArchive.xml
sgrep  -g xml  -p m4  -f EADdefs  -c                                \
       -e '(DATE in BOX)'  MyArchive.xml
```

WUMPUS

http://www.wumpus-search.org/

High-performance search engine

- uses an index for fast search over large files
- includes several ranking algorithms for approximate matches
- single- or multi-user support
- file system indexing service
    - tracks changes in the file system
    - updates the index accordingly

GCL query language

- user-defined regions
- containment, proximity, counting operators, etc.

Limited XPath support

WUMPUS EXAMPLE

```
MACRO:CHUNK = (("<c01 " + "<c02 ") .. "</did>"))
MACRO:BOXTAG = ("<container type=\"box\">")
MACRO:FILE = ($(CHUNK) > "level=\"file\""))
MACRO:BOX = ($(FILE) > $(BOXTAG))
MACRO:DATE = ("<unitdate " .. "</unitdate>")
@gcl [get] $(BOX) > "Abramovitz"
302834 302867 <c02 level="file"> <did> <container type=
@xpath doc("/u/jdoe/MyArchive.xml")/c01/c02//unitdate[2]
302857 30864
@get 302857 302864
<unitdate type="single" normal="1985">1985</unitdate>
```

SUMMARY

Many tools to work with XML data

XML transducers specified by W3C

- CSS: rendering
- XSLT: transformation
- XQuery: querying
- DOM: parsing
- Canonical XML: canonicalization
- XSL: rendering
- XInclude: merging
- XProc: pipelining a sequence of operations

Many more transducers from other sources

Search engines

- sgrep
- Wumpus
- Lucene

READ MORE

Airi Salminen and Frank Tompa, *Communicating With XML*, Springer, 2011.

Joe Fawcett, Liam R. E. Quinn, and Danny Ayers, *Beginning XML, 5th Edition*, Wiley (Wrox), 2012.