CS4221/CS5421

Tutorial 7: XML, DTD, and XPath

Mark Meng Huasong

School of Computing
National University of Singapore

Week 9, 2022 Spring

All the materials within presentation slides are protected by copyrights.

It is forbidden by NUS to upload these materials to the Internet.

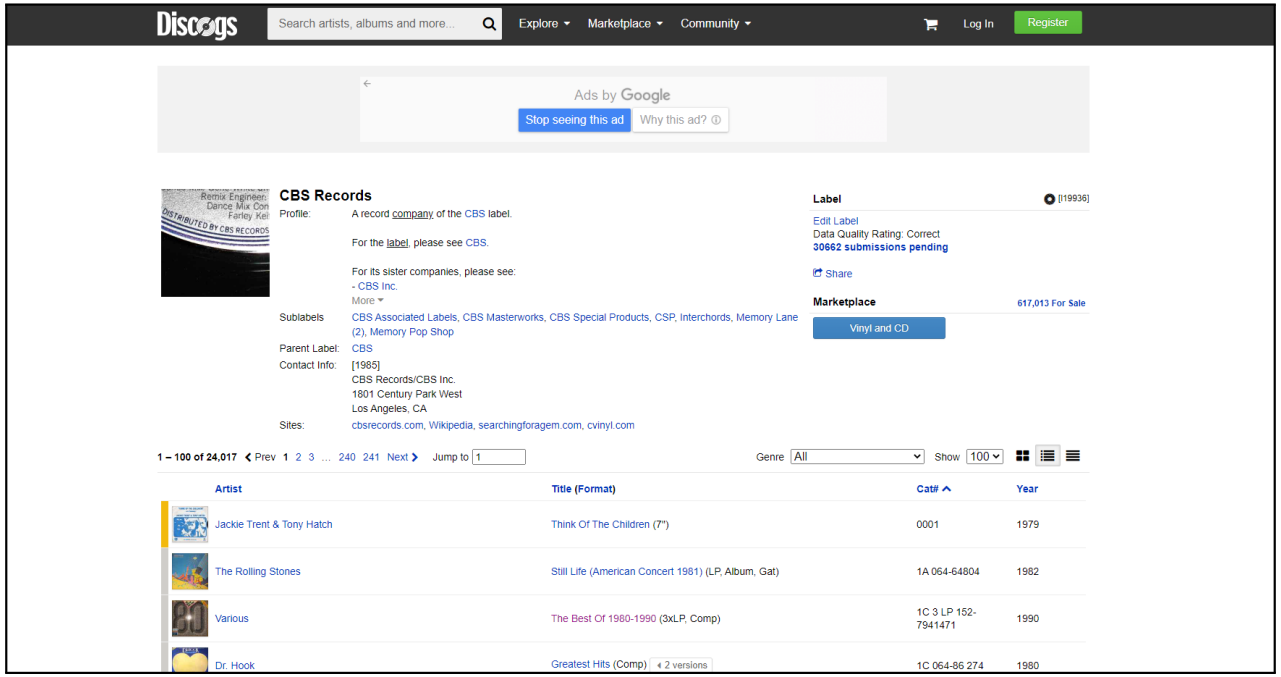
Announcement

Please pay attention to Zemmy's announcement on 14 March (6pm):

Apparently, the newest version of eXist-db cannot validate DTD properly. Please install eXist-db version 5.2.0 instead:
`https://github.com/eXist-db/exist/releases/tag/eXist-5.2.0`

Question 1 XML

Add two more albums from CBS records (check `www.discogs.com/label/19936-CBS-Records`, for instance). Update the XML document and make sure that it is well-formed using eXist-db.



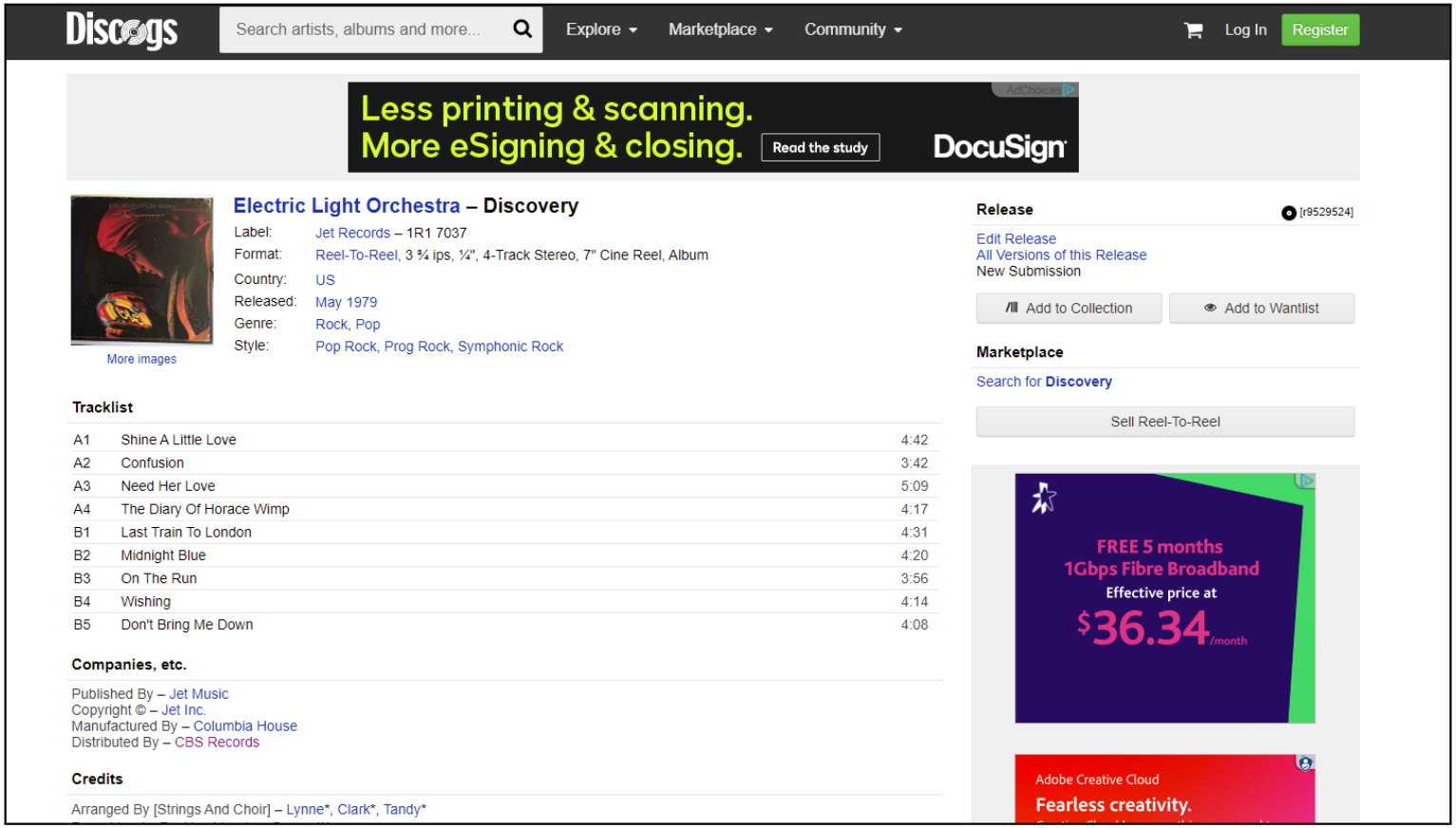
Question 1 Cont.

Solution: You can just feed the data into the XML code below:

```xml
<album>
  <title></title>
  <artists>
    <artist></artist>
  </artists>
  <songs>
    <song>
      <title></title>
      <duration></duration>
    </song>
  </songs>
  <genres>
    <genre></genre>
  </genres>
  <year></year>
</album>
```

Question 1 Cont

Take the album "Discovery" (`https://www.discogs.com/release/9529524-Electric-Light-Orchestra-Discovery`) as example.



Question 1 Cont.

We can just append the code below to the library.xml.

```
<album>
  <title>Discovery</title>
  <artists>
  <artist>
    <name>Electric Light Orchestra</name>
    <country>US</country></artist>
  </artists>
  <songs>
    <song><title>Shine A Little Love</title><duration>4:42</duration></song>
    <song><title>Confusion</title><duration>3:42</duration></song>
    <song><title>Need Her Love</title><duration>5:09</duration></song>
    <song><title>The Diary Of Horace Wimp</title><duration>4:17</duration></song>
    <song><title>Last Train To London</title><duration>4:31</duration></song>
    <song><title>Midnight Blue</title><duration>4:20</duration></song>
    <song><title>On The Run</title><duration>3:56</duration></song>
    <song><title>Wishing</title><duration>4:14</duration></song>
    <song><title>Don&apos;t Bring Me Down</title><duration>4:08</duration></song>
  </songs>
  <genres><genre>Pop</genre><genre>Rock</genre></genres>
  <year>1979</year>
</album>
```

Question 2 Document Type Definition (DTD)

Q2.1 Add an internal DTD for the XML document above. Make the DTD as tight as possible.

Solution: Add the DTD in the header of library.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library [
  <!ELEMENT library (album*)>
  <!ELEMENT album (title,artists,songs,genres,year)>
  <!ELEMENT artists (artist*)>
  <!ELEMENT genres (genre*)>
  <!ELEMENT songs (song+)>
  <!ELEMENT artist (name,country)>
  <!ELEMENT song (title,duration)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT country (#PCDATA)>
  <!ELEMENT duration (#PCDATA)>
  <!ELEMENT genre (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
]>
<library>
  <!-- Place your library data here -->
</library>
```

Question 2

DTD Syntax

Note that **"*" means zero or more**, **"+" means one or more**, **"?" means zero or one occurrences**, and **no sign after the text means exactly one**. Of these, exactly one is of course the most tight but not necessarily possible for the document we are working with.
#PCDATA is parsed character data, which we for this module essentially can think of as where we store raw values, i.e., values that do not consist of sub elements.

Q2.2 Validate the XML document with its DTD using eXist-db. Try to change the structure of the document so it is invalid and observe the error messages generated by eXist-db.

This is an open question based on the DTD defined. E.g., you may try to remove the **year** attribute from one **album**. You can also try with multiple **country** attributes in one **artist**.

Question 3

Translate the following queries into XPath.

Your answers to the questions should include an XPath query with complete path as well as one with a more concise path, where relevant. Your answers to the questions should include an XPath query with the full syntax and as well as one using the shorthand syntax.

Evaluate the queries using eXist-db. eXist-db only shows 10 elements by default. You should wrap your query in an outer element to easily see all elements, i.e. `<results>{ XPath query }</results>`. For brevity, the solutions are shown without this wrapping.

Question 3.1 (Simple Query Expressions)

Find the title of the songs.

Solution:
Let's try to find a answer with the explicit axes first (not the descendant::shorthand)

```
doc("library.xml")/child::library/child::album/child::songs/child::song/child::title
```

...or, equivalently with the descendant,

```
doc("library.xml")/descendant::song/child::title
```

By applying shorthand, we can query as follows:

```
doc("library.xml")/library/album/songs/song/title
```

```
doc("library.xml")//song/title
```

Question 3.1 Cont.

The query below does not work and gives the wrong result. (Why?)

```
doc("library.xml")/descendant::title
```

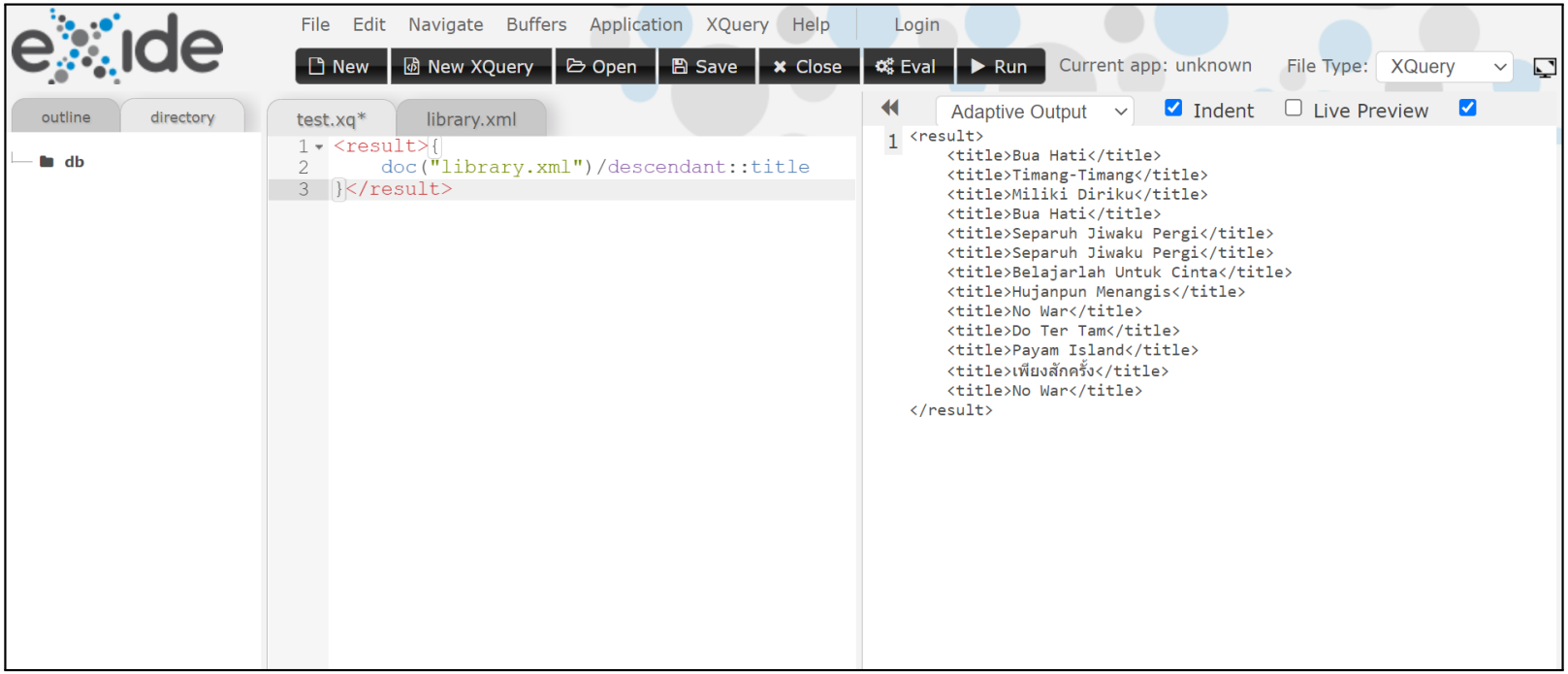


Figure: Output of the eXide (without adding extra albums from CBS)

Question 3.2 (Filter Expression)

Find the names of the Indonesian artists interpreting songs published between 1990 and 2000.

Solution:

(Query expression with the descendant)

```
doc("library.xml")/descendant::album[child::year>=1990 and child::year <=2000]
/child::artists/child::artist[child::country='Indonesia']/child::name
```

...or, equivalently,

```
doc("library.xml")//album[year>=1990 and year <=2000]/artists/artist[country='
     Indonesia']/name
```

To read more...

Documentation:

https://exist-db.org/exist/apps/doc/newrangeindex

Question 3.3 & 3.4 (Count Clause)

Find how many songs in the library were interpreted by Anang Ashanty.

Solution:

```
count(doc("library.xml")/child::library/child::album[child::artists/child::artist/
      child::name='Anang␣Ashanty']/child::songs/child::song)
```

...or, equivalently,

```
count(doc("library.xml")/library/album[artists/artist/name='Anang␣Ashanty']/songs/song
      )
<!-- the most concise form is shown below -->
count(doc("library.xml")//album[artists//name='Anang␣Ashanty']//song)
```

Caution:

```
count(doc("library.xml")//album[//name='Anang␣Ashanty']//song)
```

The expression above does not work because the condition inside [ ] became uncorrelated to node.

Question 3.3 & 3.4 Cont.

Find the title of the albums in the library with four songs or more.

Solution:

```
doc("library.xml")/child::library/child::album[count(child::songs/child::song)>=4]/
      child::title
```

...or, equivalently,

```
doc("library.xml")/library/album[count(songs/song)>=4]/title
```

Question 3.5 (Axes)

Propose an interesting query that uses different axes than "child" and "descendant". Write the query in English and in XPath.

Solution:
For example we may use `ancestor` to query all albums' titles that have include songs performed by the artist Anang Ashanty.

```
doc("library.xml")/child::library/child::album/child::artists[child::artist/
     child::name='Anang␣Ashanty']/ancestor::album/child::title
```

To read more...

Documentation: `https://exist-db.org/exist/apps/doc/xquery`
(§ Current Status of XQuery Support → Supported Optional Features)

For any further question, please feel free to email me:

huasong.meng@u.nus.edu

Copyright 2021 Mark H. Meng. All rights reserved.