Advanced Databases Summative

Bradley Mackey

for 15th March 2019

1 Tree Structure

See the attached document for the tree diagram of `books.xml`. This is available on the **final page** of this report.

2 Querying

2.1 XPath

2.1.1 Find all Mary Poppins books which were published before WWII.

1. *Query*:

```
//book[@publicationDate<"1939"][title[contains(text(),"Mary Poppins")]]
```

2. *Interpretation*: This query first uses the Restricted Kleene Closure to find every occurrence of `<book>` in the XML structure. This ensures that we find all books at all levels of the database (if the database were to be grown in the future). We then use a tag predicate of `[@publicationDate<"1939"]` to filter out all books that were NOT released before 1939. For all the books returned by this, we then check if they have a `title` property. If they do, we use a nested predicate to check the title of the book. If the book title contains the string `"Mary Poppins"` (not including these quotation marks), the full book node will be returned.

3. *Limitations*: this query will only find Mary Poppins books if her name appears directly in the title. If such a Mary Poppins book existed that did not feature her name in the title, it would not be returned by our query. It is not sufficient just to check the author or illustrator as they may have been involved in creating other books (not Mary Poppins). Additionally, WWII started in September of 1939 and—as we have no way to distinguish the month of publication—it may be the case that a book published in 1939 was actually published before WWII. We have no way of telling this, we just ignore all books published in 1939 to be sure.

4. *Output*:

```
<book publicationDate="1934">
   <title> Mary Poppins </title>
   <author> P. L. Travers </author>
```

```
   <illustrator> Mary Shepard </illustrator>
</book>
<book publicationDate="1935">
   <title> Mary Poppins Comes Back </title>
   <author> P. L. Travers </author>
   <illustrator> Mary Shepard </illustrator>
</book>
```

2.1.2 Find all movies that have academy nominations.

1. *Query*:

```
//movie[normalize-space(academyNominations/text()) > "0"]
```

2. *Interpretation*: This query first uses the Restricted Kleene Closure to find every occurrence of `<movie>` in the XML structure. This ensures that we find all movies at all levels of the database (if the database were to be grown in the future). We then use a tag predicate of `academyNominations`, filtering out all movies that do not include an `<academyNominations>` node. We access the text stored at this node using the XPath `text()` function. As the XML files provided have whitespace surrounding the text in each node, we use the `normalize-space` function to remove this whitespace, as to not retrieve inaccurate results. We then ensure that the number of nominations is greater than 0, as it could be possible a movie could have an `<academyNominations>` node with 0 nominations. Movie nodes filtered out by this predicate are not included in our results.

3. *Limitations*: this query relies on the fact that `academyNominations` are stored as an integer inside the `<academyNominations>` node. If the nominations were instead listed out (as `<nominator>` nodes for example), the query would not be able to identify these. Otherwise, if all movies follow this style of listing nominations, there are no further limitations.

4. *Output*:

```
<movie language="En" type="film" year="1964">
   <title> Mary Poppins </title>
   <actress> Julie Andrews </actress>
   <actor> Dick Van Dyke </actor>
   <actor> David Tomlison </actor>
   <actress> Glynis Johns </actress>
   <producer> Walt Disney </producer>
   <director> Robert Stevenson </director>
   <academyNominations> 13 </academyNominations>
</movie>
<movie language="En" type="film" year="2013">
   <title> Saving Mr. Banks </title>
   <actress> Emma Thompson </actress>
   <actor> Tom Hanks </actor>
   <academyNominations> 1 </academyNominations>
</movie>
```

2.2 XQuery

2.2.1 How many books have there been published about Mary Poppins?

1. *Query*:

```
let $b := doc("books.xml")//book[title[contains(text(),"Mary Poppins")]]
return <result>{count($b)}</result>
```

2. *Interpretation*: we start by using the `let` keyword to bind to all results of a query at once, allowing us to perform aggregate operations on the results of the query. We open the `"books.xml"` file such that we can query it. Using a similar XPath style query as question 1, we query the document for all `book` nodes using the Restricted Kleene Closure. This will allow us to find all possible `<book>` nodes at any point in the database, even if the database were to grow in size in the future. We then use a predicate to assert that each returned `book` has a `title` property. Then, we use a nested predicate to evaluate the text of the title, using the `text()` function. We assert that the title must contain the string `"Mary Poppins"`, such that all books about Mary Poppins are returned. Then, in the `return` block, we perform an aggregate `count` of all the results, returning this in a `<result>` tag.

3. *Limitations*: Similar to question 1, this will only count books where `"Mary Poppins"` appears in the title directly. If there is a book that is actually about Mary Poppins but does not include her name in the title, it will not be included in the book count.

4. *Output*:

```
<result>8</result>
```

5. *XPath*: As XPath and XQuery share the same functions library, we are able to construct an XPath query to accomplish the same functionality. We use the same path component with exactly the same predicates as the XQuery query. We then wrap this within the `count` function that simply counts the number of returned results.

```
count(//book[title[contains(text(),"Mary Poppins")]])
```

2.2.2 List the books that have been published before the second Mary Poppins movie appeared.

1. *Query*:

```
let $yrs := (
    for $mov in doc("media.xml")//movie
    where $mov[title[contains(text(),"Mary Poppins")]]
    order by number($mov/@year)
    return $mov
)
let $yr2 := number($yrs[2]/@year)
for $b in doc ("books.xml")//book
where $b[number(@publicationDate) < $yr2]
return $b
```

2. *Interpretation*: we firstly need to find what year the second Mary Poppins movie appeared. Therefore, we perform a nested query to return a list of Mary Poppins movies, ordered by the year that they were released. This allows us to directly index into the list using `$yrs[2]/@year` to get the year that the second movie was released. This ordering and indexing is a nice interface to easily change this value in the future if the database required it—we simply just change the number of the index to get books released before that specific movie. We then fetch all books from the books database, using a `where` predicate to only return books that were released before this second movie was released by comparing the `@publicationDate` to the value of `$yr2`. We then return each result that matches this predicate as the successful query cases.

3. *Limitations*: this technique suffers from the same limitations as the previous queries, as movies without `"Mary Poppins"` in the title will not be considered Mary Poppins movies, as these are what is matched against. Additionally, in order for us to find the second released movie, we sort through all movies with Mary Poppins in the title first. If the database contained many movies with `"Mary Poppins"` in the title, this sorting could take a non-trivial amount of time and impact database performance (as sorting is at least an $\mathcal{O}(n \log n)$ operation).

4. *Output*:

```
<book publicationDate="1934">
  <title>Mary Poppins</title>
  <author>P. L. Travers</author>
  <illustrator>Mary Shepard</illustrator>
</book>
<book publicationDate="1935">
  <title>Mary Poppins Comes Back</title>
  <author>P. L. Travers</author>
  <illustrator>Mary Shepard</illustrator>
</book>
<book publicationDate="1943">
  <title>Mary Poppins Opens the Door</title>
  <author>P. L. Travers</author>
  <illustrator>Mary Shepard</illustrator>
</book>
<book publicationDate="1952">
  <title>Mary Poppins in the Park</title>
  <author>P. L. Travers</author>
  <illustrator>Mary Shepard</illustrator>
</book>
<book publicationDate="1962">
  <title>Mary Poppins From A to Z</title>
  <author>P. L. Travers</author>
  <illustrator>Mary Shepard</illustrator>
</book>
```

5. *XPath*:

2.2.3 How much later has the most recent Mary Poppins movie appeared, compared to the second book about Mary Poppins?

1. *Query*:

   ```
   query
   ```

2. *Interpretation*:

3. *Limitations*:

4. *Output*:

5. *XPath*: