Project: Stage 2
Deliverables

Outline

This document describes the deliverables for the **Stage 2** of the ArferDB.

It is a companion document to the **Stage 2** description document that provided instructions for building ArferDB's query processor. This document outlines XML generation and describes the final deliverable.

XML Generation

In the previously released document we described the steps for processing XPLite queries. The result of an XPLite query over an XML repository is a collection of XML nodes from the XML documents stored in the repository.

The output of XPLite queries shown to the users is the XML fragments representing each individual XML node. Upon completion of the query processing task, ArferDB shall generate an XML representation of each node in the answer set.

When doing so, the following guidelines shall be followed.

Empty Query Results. If an XPLite query returns no results, ArferDB shall simply output (print):

```
no XML nodes selected
```

Element Nodes. A result of an XPLite query can be either a set of XML element nodes, or a set of XML attribute nodes. We can formally prove that a mixed set can never be a result of an XPLite query. When the result of a query is a collection of XML element nodes, the query output is constructed out of the XML fragments representing each node and XML file names as follows:

- The result set is ordered first by the id of the XML document to which an XML node belongs, and then by the node Id within the document[1]
- For each XML document, first its name is printed on a separate line. One line is skipped after that. This is followed by the XML representing each node. There should be an empty line between every two consecutive XML fragments. There should be two empty lines after the last XML fragment is printed.

Example. Consider the following two XML documents:

[1]Generally speaking, ordering XML documents is not needed, but this will ensure that the actual output of each query is deterministic and easy to evaluate.

```
test1.xml:

<?xml version="1.0">
<root>
   <a id = "1">
      <b>Hello</b>
      <b>World!</b>
   </a>
   <a id = "2">
      <c>42</c>
   </a>
</root>
```

```
test2.xml

<?xml version="1.0">
<plan>
   <a>Wake up!</a>
   <b>
     <a time ="morning" id="0">breakfast</a>
     <a> drive to work</a>
   </b>
   <a time = "afternoon">
      <task>meeting</task>
   </a>
  </plan>
```

Let an XPlite query be `/child::a[child::*]`. There are three XML nodes that match this query: two `<a>` element nodes from `test1.xml` abd the last `<a>` node from `test2.xml`. The output of this query shall look as follows:

```
test1.xml

<a id ="1">
  <b>Hello</b>
  <b>World</b>
</a>

<a id = "2">
   <c>42</a>
</a>


test2.xml

<a time="afternoon">
  <task>meeting</task>
<a>
```

Note. You can tweak the output a bit to make it stand out a bit more (using spacers to separate XML nodes from different files and from within the same file, and so on, but *it is sufficient to generate the output above*.

XML for a single XML node. The XML fragment for a single XML node is a string containing the fragment of the XML document in question rooted at the given XML node. The string shall include all attributes for the XML node, and for all its descendant nodes.

Formatting. XML fragments generated by you must be readable. You are not expected to produce pretty print of outstanding quality, but you are not allowed to simply place the entire fragment into a single line of text. Line breaks and tabs/spaces can be used for some straightforward splitting of the fragment into individual lines, and for indentation.

Attribute nodes. The other type of nodes that can be present in the result set are attribute nodes. To display an attribute node in the output of an XPLite query, you output the opening tag of its parent XML element *with only the attribute corresponding to the attribute node* present.

Example. Consider the XPLite query `/descendant::*/attribute::attribute()` over a repository consisting of documents `test1.xml` and `test2.xml`. The result of this query is a nodeset containing all five attributes occurring in the two documents. ArferDB shall print the following output:

```
test1.xml

<a id="1">

<a id="2">


test2.xml

<a time="morning">

<a id="0">

<a time="afternoon">
```

ArferDB Executable

Each team must produce an command-line ArferDB client that allows the user to type ArferDB commands and XPLite queries. The design of the client program and the specific syntax of the command language are left to individual teams (in recognition that some teams have already started work on the client program). However, each team's ArferDB client shall implement the following list of commands.

XML repository creation. This command shall take a name of the XML repository to create. Upon receiving this command, ArferDB shall create an empty repository with a given name. See **Step 1** specifications for the specific description of how a repository creation command must work.

XML repository deletion. This command taks a name of an existing XML repository and removes the repository with the provided name from disk.

List of XML repositories. As the result of executing this command, ArferDB shall print the list of all XML repositories currently available.

Insert an XML document into a repository. This command takes two parameters: an XML repository name and a name/location of an XML document to be inserted into the specified repository. As the result of this command, the specified XML document is inserted into the repository. See Stage 1 documentation for the exact specification for this command (and the list of possible error conditions).

XPLite query. This command shall consist of the name of an XML repository and an XPLite path expression. The result of this command is the output discussed in the **XML Generation** section of this document.

Note: Each XPLite query is applied to exactly one repository. To enforce this, ArferDB either needs to implement a use command, or it ineeds to provide XPLite query command syntax that includes both the path expression and the repository name. The former can look as follows:

```
ArferDB> use Test

repository Test is now active
ArferDB> /descendant::c

test1.xml

<c>42</c>

ArferDB>
```

The latter command syntax can look as follows:

```
ArferDB> Test:/descendant::c

test1.xml

<c>42</c>
```

Exit command. Self-explanatory. The effect of the command is the end of the work of ArferDB client.

These are the **required** commands. Your implementation can include additional testing/management commands that can be used in your demo to illustrate the work of ArferDB.

Scalability

The version of ArferDB you submit and demonstrate to the instructor must be able to operate a `tinyFS` disk of reasonably large size: at least up to serveral megabytes. The demo will include some stress tests of your implementation.

You can achieve this either via hardcoding a reasonably large number of disk blocks into the tinyFS call to create a disk, or via implementing some commands that allow the user to set parameters of the current ArferDB storages.

Additionally, the buffer size shall be set to a relatively large number. One way to work it out is to set the buffer size to be 10% of the number of blocks on the `tinyFS` disk. This will create a reasonable size buffer, but will not obviate our work by making the buffer too big, and thus capable of storing app data ArferDB operates with in main memory.

Software Demonstration

The evaluation of your ArferDB submission will proceed in two steps:

1. Step 1: ArferDB demo. Each team will demo its Stage 2 implementation by running the ArferDB client and showing how it processes XPLite queries.
2. Step 2: Individual evaluation. After the demo time, the instructor will evaluate each submission. During the demo time, we will install your version of ArferDB on instructor's account, and verify that it works as intended. After the demo is over (possibly on a different day), the instructor will spend some time evaluating your submission in detail and assigning it a numeric score.

Testing

A test suite of XPLite queries (and a test suite of XML documents to accompany them) will be released for you. The suite will contain tests for each XPLite feature. Due to the nature of the XPLite syntax, some features (e.g., axes, nodetests) are more commonly used than others (e.g., a specific built-in predicate). This will be reflected in the test suite.

A separate suite of stress tests (mostly consisting of some larger XML documents, and a number of queries that may take a significant time to complete) will be released as well. This suite is concentrated on scaleability and the ability to ArferDB to work with large data collections/XML documents.

Timeline

ArferDB code deliverable is due Friday, December 6, 11:59pm.

ArferDB demos can be scheduled for Wednesday, December 4, Thursday, December 5, Wednesday, December 11, Thursday, December 12 and Friday, December 13. **My strong preference is to have all demos complete during the final week of classes.** I want ArferDB to be complete before you head into your finals week.

The reason for a break in demo schedule is my December 6 – December 10 out-of-state trip.

Deliverables

Your **December 6** set of deliverables shall consist of:

- **Code.** The entire codebase of ArferDB.
- **Team breakdown.** A text document **team.txt** containing the list of team members and the breakdown of contributions of each team member to the codebase.
- **Documentation.** A `README` file detailing (a) project compilation and running instructions and (b) the syntax of the ArferDB client commands.

Submission process

Submit using `handin`:

```
$  handin dekhtyar arferdb <files>
```

Additionally, put the entirety of your project on your team's github course repository.

Good Luck!