CS561

Web Data Management

Spring 2013

Assignment 1

Professor: Vassilis Christophides

Deadline: **25/03/2013**

DTD Description

In Figure 2 you can see a Document Type Definition (auction.dtd), which describes a basic auctions webpage. The main entities come in two groups: **person**, **open auction**, **closed auction**, **item**, and **category** on the one side and entities akin to **annotation** on the other side. The relationships between the entities in the first group are expressed through references, as depicted with arrows in Figure 1 (implemented by attributes of the type ID/IDREF). The relationships between the entities of the second group, which take after natural language text and are document-centric element structures, are embedded into the sub-trees to which they semantically belong. The entities we just mentioned carry the following semantics:

- **Items** are the objects that are on sale in an auction or that already have been sold. Each *item* carries a unique identifier and bears properties like payment (credit card, money order, . . .), a reference to the seller, a description etc., all encoded as elements. Each item is assigned a world region represented by the item's parent element.
- **Open auctions** are auctions in progress. Their properties are the privacy status, the bid history (i.e., increases or decreases over time) with references to the bidders and the seller, the current bid, the time interval within which bids are accepted, the status of the transaction and a reference to the item being sold, among others.
- **Closed auctions** are auctions that are finished. Their properties are the seller (a reference to a person), the buyer (a reference to a person), a reference to the respective item, the price, the number of items sold, the date when the transaction was closed, the type of transaction, and the annotations that were made before, during and after the bidding process.
- **Persons** are characterized by name, email address, phone number, mail address, homepage URL, credit card number, profile of their

interests, and the (possibly empty) set of open auctions they are interested in and get notifications about.

- **Categories** feature a name and a description; they are used to implement a classification scheme of *items*. A *category-graph* links categories into a network.

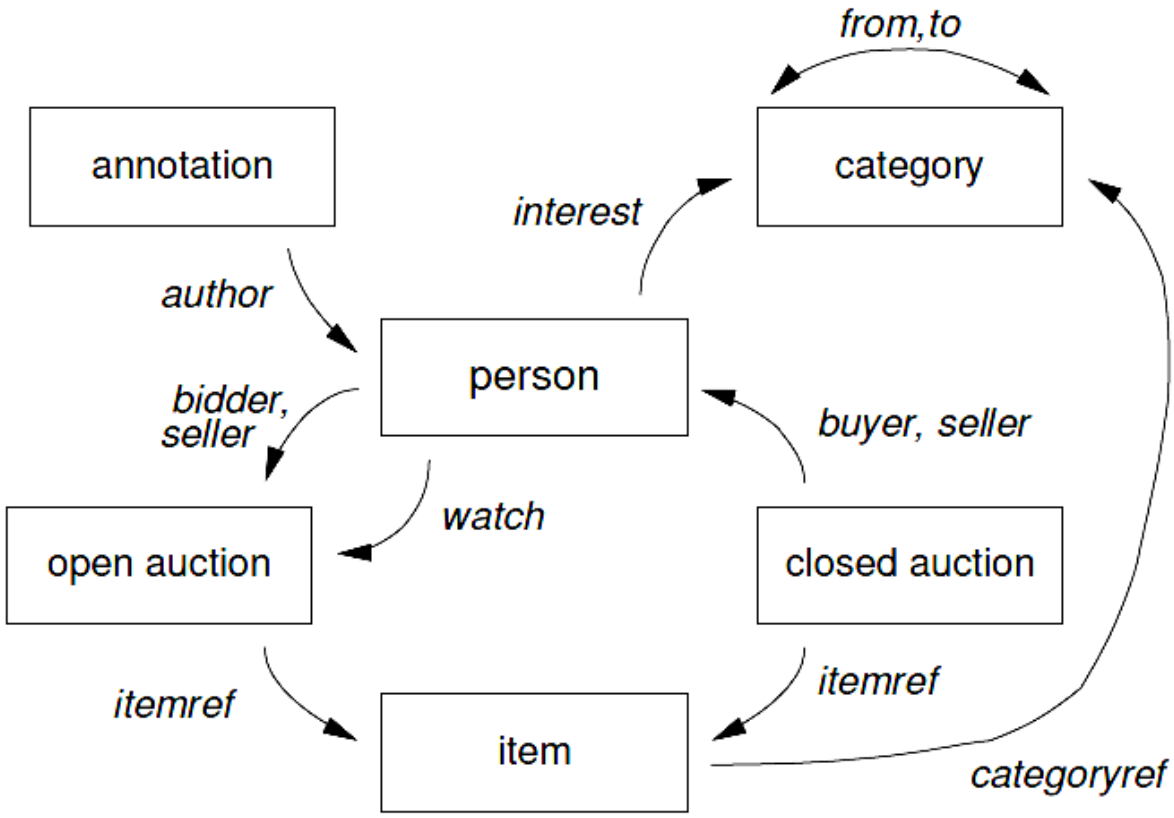

Figure 1: Basic entities in auction.dtd

We emphasize that these entities constitute the relatively structured, i.e., data oriented part of the document. Their sub-element structure is fairly regular on a per entity basis but there are predictable exceptions such as that not every person has a homepage; in a relational DBMS, these exceptions would typically be taken care of by NULL values. Another characteristic of these entities is that, apart from occasional list types such as bidding histories, the order of the input is not particularly relevant. On the other hand, the sub-elements of the document-centric part of the database, namely those of annotation and similar elements, do not accentuate the above aspects. Here the length of strings and the internal structure of sub-elements varies greatly. The markup now comprises itemized lists, keywords, and even visual formatting instructions and character data, doing its best to imitate the characteristics of natural language texts. This warrants that the database covers the full range of XML instance incarnations, from marked-up data structures to traditional prose.

Using the described DTD and an XML document generator, we created various XML documents, which you can download from the following location:

http://www.csd.uoc.gr/~hy561/assignments/assign1/data.tar.gz

Software setup

In the context of this assignment, you are asked to download the following two XPath/XQuery processors and load the given collection of XML documents into these processors.

eXist-db Native XML Database:

http://exist-db.org/exist/apps/homepage/index.html

SAXON XSLT and XQuery Processor:

http://saxon.sourceforge.net/

The latest versions of the above processors are also available in the course's webpage. It is suggested that you download them from this webpage for consistency purposes.

You can find a quick start guide of eXist-db in

http://exist-db.org/exist/apps/doc/quickstart.xml

Also, you can find a getting started guide for Saxon in

http://www.saxonica.com/documentation/about/gettingstarted/gettingstartedjava.xml

In order to use the processors, you should have installed a late version of Java.

Querying the XML collection

The main goal of this assignment is to express the following queries in XPath and XQuery format, and run them using the software above.

- **Query 1**
  "Get the name of the item with item ID "item20748", which is placed in Asia"
- **Query 2**
  "Get the initial bids of all the open auctions"

- **Query 3**
  "Get the reserves of all the open auctions in which a specific bidder (person148427) made a bid before another specific bidder (person10487)"
- **Query 4**
  "Get the number of items that cost more than 40"
- **Query 5**
  "Get the quantities of the items appearing in open auctions"
- **Query 6**
  "Get the number of elements that have at least one 'text' sub-element"
- **Query 7**
  "Get the names of all the items that include the word 'gold' in their description"
- **Query 8**
  "Get the keywords that are emphasized in the annotations of the closed auctions"
- **Query 9**
  "Get the email addresses of the sellers for all the auctions that have at least one keyword emphasized in their annotation"
- **Query 10**
  "Get the number of female sellers appearing in open auctions"

Files to submit

The files you should submit to the course's email address (hy561@csd.uoc.gr) are the following:

1. A text file containing the above queries expressed in XPath and XQuery.
2. 10 XML files, each one containing the results for the corresponding query.
3. A report in which you will include the running time for each query and each processor, along with your comments on these findings. In the report, you also have to include your system setup (processing power, RAM, OS etc.).

Notes

- Because of the big size of the XML collection, you can use and modify the given test.xml file to test your queries, before running them on the whole collection.
- Also because of the size of the collection, both processors will need a lot of main memory.
  It is suggested that you increase the maximum heap size of eXist (http://tinyurl.com/brcpx5n) and Saxon (command line option *–Xmx* http://tinyurl.com/c87t3wc).
- You can load all the files in eXist by creating a collection (File->Create collection) and storing the files inside this collection (File->Store files/directories). Then you can run queries on the collection by selecting your newly created collection in the context menu (after following the "Tools->Find" path.
  Example:
  XPath:

```
/site/regions/africa
```

  XQuery:

```
for $doc in collection('/db/collection')
let $regions := $doc/site/regions
return $regions/africa
```

- To use all the given files in Saxon, you can use the file gen-collection.xml. Your queries should be stored in a query file. The command is:
  java -Xmx4500m -cp saxon9he.jar net.sf.saxon.Query query.file
  Example of query files:
  XPath:

```
doc("./data/gen-collection.xml")/site/regions/africa
```

  XQuery:

```
for $doc in collection('./data/gen-collection.xml')
let $regions := $doc/site/regions
return $regions/africa
```

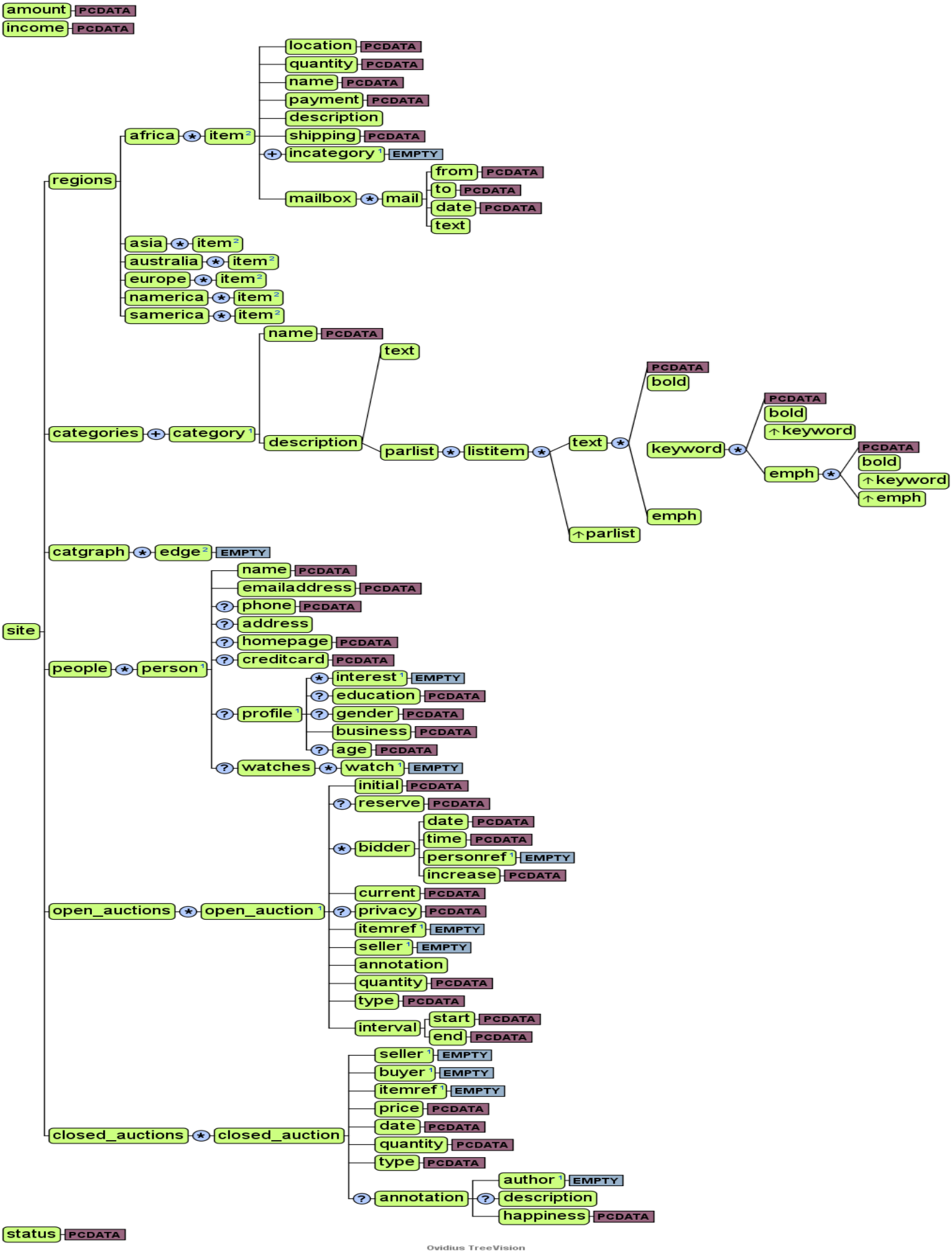

Figure 2: auction.dtd