

XML and Relational Databases

Exercise 1: XML to Relational Mapping

- 1.1.** Apply the following XML to Relation mappings on the specified data:
- a.) Apply schema-based Shredding to the Flight Example. Use the data *flights_data_no_nesting.xml* and schema: *flights_schema_no_nesting.xsd*. This is the same schema used in all our previous exercises.
 - b.) Apply schema-based Shredding to the Flight-Example. Use the data *flights_data_nesting.xml* and the schema *flights_schema_nesting.xml*. This schema we created especially for this exercise.
 - c.) Translate the *flights_data_nesting.xml* to a relational model using the Binary/Edge approach.
- 1.2.** In each of the above mappings, transform the following XQuery into SQL:
- a) "Determine" all flight destination of the passenger named "Santa Claus"
 - b)* "Determine all flights itineraries from 'NPL' to 'SPL' with one, two, three or *infinite number of stops*"

Exercise 2: Node IDs

- 2.1.** Add NodeIDs to the *flights_data_no_nesting.xml* document, using one of the following types of IDS. **Assign IDs which maintain document order!**
- a) integer IDs
 - b) double IDs
 - c) Dewey order
- 2.2.** Write the new Nodes Ids for the document resulted after the execution of the following XUpdate (**only new nodes get new IDS, the other IDs remain unchanged. Document IDs overall should maintain document order.**)

```
let $doc := doc('flights_data_no_nesting.xml')
do update insert <Reservation>
  <date>2006-12-26</date>
  <flightRef>LX183</flightRef>
  <passRef>000111</passRef>
</Reservation>
before $doc/Reservation[1]
```

- 2.3** Fill in the following table and estimate the properties of the different ID types:

	Size	Doc Order (next)	Ancestor/Descendant	Sibling/Before/After node (navigation)	Insert	Computation intensive
Integer Ids						
Float Ids						
Dewey						