PROJECT REPORT

Design and Transformation
of a Tour Operator's XML document

Realised by:

Hien X. PHAM

01 February, 2022

Data Pipeline 1

Design

Working Environment

Working environment for this project includes:

- Program: Liquid Studio 2021
- XML Validator: Microsoft .Net (XSD 1.0 aware)
- Debug Engine: Saxon HE 9.8
- Online services: jsonformatter.org (beautify), jsonlint.com (json validator)
- Project's Github Repository: https://github.com/hienpham15/Data_Pipeline_1.

Schema Model

Firstly, the schema is designed with the recommended best practices in mind [1]. To summarize, for this schema, the important points are:

- Namespaces are declared at the root node and use prefixes.
- Entities references are limited as it poses a serious security risk [2].
- Attributes are only used for metadata.
- Elements are more preferable since it is expandable and allow simple or complex types.

Secondly, with the assumption of the tour operator working in France, the schema is defined as a series of ***Operations***. Each ***Operation*** has a unique ID and 9 different sections including:

- **Type**: each ***Operation*** has 3 type:
  - Inbound: from foreign countries to France.
  - Outbound: from France to foreign countries.
  - Domestic: domestic tours inside France.
- **Price**: total cost of a tour and its respective currency.
- **CurrencyExchangeService**: indication whether or not the tour provides a currency exchange service.
- **Insurance**: insurance information of the tour.
- **Period**: duration of the tour.
- **Booking**: booking information, including the channel of which the tour is booked.
- **TourCategory**: information about which type of activities the tour is oriented toward (Culture, Wellness, ...)
- **Itineraries**: detail information (accommodation, transportation, visiting locations, meals) on each day of the tour.

Discussion

In the context of this project, I understand the need to fix all possible aspects of a tour operator into a single schema. However, in reality, this is not a good practice (will be discussed below) and should be avoided.

The reasons for the above argument are:

- Adding sections and sub-sections turns the schema into a heavily nested tree. This adds unnecessary complexity and make it more difficult for exploiting/transforming the data. As John Maeda has said: "Simplicity equals sanity" [3], it's the more reason we should make the schema as simple and user-friendly as possible.
- Additionally, the time complexity of searching an unbalanced binary tree is already $O(n^2)$. Hence, processing a query or transformation of an unbalanced n-ary tree can take an undesirable amount of time (upper bound of time complexity can be exponential).

Following OpenTravel schema practices [4], it is better to break each section, sub-section into separated schemas. This allows a more comprehensible structure while easier to query and accessing data in much shorter time. However, in many cases, over-use of this practice can cause the entire schema database bigger and bigger, hence, increasing the space complexity.

In conclusion, there is no superior approach to one problem, as an engineer, one's job is to find a balance between space and time complexity.

Transformation

Scenario 1

At the end of the year 2022, the tourist agency want to have a statistics number on each type of the tour operation (Inbound, Outbound, Domestic).

Suggested transformation:

- Search for all operations which have a begin date between 01-Jan-2022 and 31-Dec-2022.
- Count the type of each valid operation.

Scenario 2

The tourist agency would like to have a list of available activities for each category of tour. For example: the list of cultural activities with respective date and location.

Suggested transformation:

- Recursively move through each **Operation**.
- At each operation, get the type of the tour and recursively move through each available date.
- Get all available activity of each date and its respective location.

Scenario 3

The tourist agency has plans for future expansion with more detail about a tour operation. They have decided to convert all attribute of a node into its child element for the expandability.

Suggested transformation:

- Apply identity transformation.
- At each node, its attribute is turned into node's child element.

Scenario 4

A partner of the tourist agency sent a data request, they wanted to know the schedule of each available tour operation. They have specifically asked for the return results to be in a HTML table for displaying in their website.

Suggested transformation:

- Form a table with 4 columns (tour type, date, time and activity).
- At each **Operation**, iteratively traverse the **Itineraries** node to obtain required information.

Scenario 5

An online service sent a request for general information of tour operation based on its ID number. The information requested includes: type, duration, price of the tour, does the tour have insurance ? does it support currency exchange service?. Additionally, with the constraints of the data system of the service, the results must be in json format.

Suggested transformation:

- Generate an **xsl:key** to look for Operation which matches the request ID.
- Use different template for different node to transform the XML information into JSON format.

References

[1] Yves Savourel, Jirka Kosek and Richard Ishida. Best practices for XML Internationalization, 2008. URL https://www.w3.org/TR/xml-i18n-bp/. 2

[2] OWSP, www-community. XML External Entity (XXE) processing, . URL https://github.com/OWASP/www-community/blob/master/pages/vulnerabilities/XML_External_Entity_(XXE)_Processing.md. 2

[3] John Maeda. *The Laws of Simplicity*. The MIT Press, July 2006. 3

[4] OenTravel (provided by Liquid Technologies. Documentation for OpenTravel, . URL https://schemas.liquid-technologies.com/OpenTravel/2008B/?page=policy2.html. 3