DSTI Data ScienceTech Institute

Applied MSc in Data Analytics

Applied MSc in Data Engineering & Artificial Intelligence

DATA PIPELINE PROJECT REPORT -S24

PROJECT : Modeling and Exploitation of an XML-Based Management System for Supporting Disabled and Elderly Care

Instructor: ***Catherine Faron***

Group members

Yulia Chernova

Wafa Bouakez

Susmitha Ann Alex

Claudine Uwitije

1. INTRODUCTION

As society ages and the prevalence of disabilities rises, the demand for compassionate, coordinated care for elderly and disabled individuals has become increasingly critical. A well-organized system to manage information about these individuals and their service needs is essential. This project aims to develop a comprehensive XML database model for a collaborative platform that addresses the social and medical care requirements of these populations. Many individuals in these groups struggle to maintain their independence, which is why our project seeks to create an integrated system that connects assistance seekers with essential services such as medical care, home care, and various activities and service providers. This system will enhance communication and coordination among services.

To achieve this, we designed an XML Schema that effectively captures a variety of information related to assistance seekers, service offerings, participant engagement, and scheduling logistics. Additionally, we implemented a range of scenarios on the XML database to visualize and manipulate the data using XSL stylesheets, improving accessibility and usability. By exploring various data formats, including XML and JSON, we aim to ensure that our platform is adaptable and responsive to the evolving needs of the community it serves.

This report will outline the modeling process, detail the resulting XML database, and describe the XSL stylesheets developed to provide critical insights and promote collaboration among all stakeholders involved in caring for elderly and disabled individuals.

2. WORK-LOAD DISTRIBUTION

Team Member	Tasks
Claudine Uwitije	**Scenario 1:** Assistance Seeker Statistics **Scenario 2:** Assistance Seekers sorted by age with Conditional Comments **Scenario 8:** Dynamic transformation of a part of Health Management System Data into JSON Format
Susmitha Ann Alex	**Scenario 3**:Assistance Seekers Gender & SeekerType Statistics Visualisation Report **Scenario 4**: Filtering the providers details using location
Wafa Bouakez	database/schema creation **Scenario 5:**Assistance Seeker Table Transformation **Scenario 6** : Relance Report Table Transformation
Yulia Chernova	**Scenario 7:** Transforming Assistance Seeker Data to Another XML Format.

3. WORKING ENVIRONMENT

We used a local development environment with tools like Python and libraries such as lxml for XML parsing and transformation.

4. HEALTH DATABASE AND SCHEMA

To create the database, the work started with designing the schema, the goal was to develop a management system where you can manage assistance seekers, health providers, multiple services, and activities, as well as a node for user participation whether in services or activities.

The schema uses different types to ensure a modular and clear way of structuring the data, such as infoType, userType, providerType, seekerType, aloneType, and others. The use of <key> and <keyref> elements was essential but quite challenging. These were used to link activities and services (serviceID and activityID) with health providers (providerID), and then connect everything with assistance seekers, who can have multiple scheduled services and activities. The relationships between all these nodes can be visualized in the user participation node.
Additional attributes were used to add useful contextual information such as occurrence, unit of age, type of handicap (acute or chronic), etc.

- Challenges:

The biggest challenge was using the key and keyref elements correctly, initially, the XPath expressions were incorrect and the placement of keyref was sometimes inaccurate, so it had to be corrected several times, especially since there were multiple levels of linked contexts. The testing of the database while constructing it allowed us to identify and fix these path errors. ChatGPT was used to address some path issues, and correctly place the references.

5. SCENARIOS - XSL STYLESHEET

Scenario 1: Assistance Seeker Statistics

This XSL stylesheet presents demographic information, such as average, minimum, and maximum ages, along with gender distribution percentages across different categories of assistance seekers. It employs **xsl: variable** to temporarily store values for calculations. The **xsl: for-each** construct iterates through the XML nodes to dynamically retrieve and process this data. Furthermore, **xsl: sort** organizes the assistance seekers by age in descending order, ensuring that the oldest and youngest individuals are prioritized in the display.

Scenario 2: Sorted HTML Report for Assistance Seekers with Conditional Comments

This XSL stylesheet transforms XML data into an HTML table specifically for assistance seekers aged 50 and above, categorized as elderly. The data is sorted by age in descending order, and personalized comments are included based on predefined conditions. The key steps are as follows:

- **Parameter definition:** Two parameters (age threshold and seeker type) dynamically filter seekers by age (>50) and type (elderly).
- **Dynamic title:** The report title is generated dynamically based on the filtering criteria.
- **Iterating and Filtering:** An **xsl: for-each** loop iterates through the seekers, sorting them by age and filtering to display only relevant data.
- **Conditional Comments:** An **xsl:choose** structure creates personalized comments for seekers who are both handicapped and over 50 as they may need a special care

Scenario 3: Assistance Seekers Gender & Seeker Type Statistics Visualisation Report

Implemented a Python solution to visualize the gender & Seeker Type distribution among assistance seekers within the collaborative platform. Utilizing Python's data manipulation libraries, processed the XML

data representing the assistance seekers and their respective genders. The results were rendered in an HTML format (T3_gender_pie_chart.html) using Python's XML and visualization tools, generating a clear graphical representation of the gender distribution. This approach allowed for an interactive and easily interpretable visualization directly in a web browser, enhancing the platform's ability to analyze demographic trends among its users. This visualization supports better decision-making by providing insights into the gender-related dynamics within the platform's target population.

Scenario 4: Filtering the providers details using location

In this scenario, health providers' records or data were dynamically filtered by city from an XML file using an HTML and JavaScript-based interface. The XML data was first normalized by breaking down address fields into individual components like street, city, state, and zip code. We then loaded this normalized data dynamically in the HTML dropdown, allowing users to search or filter cities. JavaScript was used to fetch and parse the XML, extract city information, and populate the dropdown options.

Challenges faced: 1. XMLHttpRequest at file from origin has been blocked by CORS policy(resolved).
2. Couldn't dynamically populate the values in the search dropdown(Still not resolved).

Solution: Run a local web server - Instead of opening the HTML file directly in the browser, serve it via a local web server. For this, ran "python -m http.server 8000" command in the terminal and navigated to http://localhost:8000/T4_filter-location.html (Took the help of ChatGPT to overcome this challenge).

Scenario 5: Assistance Seeker Table Transformation:

This transformation converts XML data into an HTML table displaying the assistance seeker's name, service ID, participation year (extracted from a participation date), and a decision column which compares the participation year to a threshold (2023) to determine if the data should be archived or kept open, acting as a trigger for the system to generate notifications or other actions.

Challenges:

-Using correct XPath expressions to target the exact nodes and data.

-Employing variables for a cleaner and understandable XSLT allowing dynamic paths and easy reuse of filtered data.

-Using last() to retrieve the last service, as an assistance seeker may have more than one associated service.

Scenario 6: Relance Report Table Transformation:

This transformation generates a report with frequently accessed data for health providers such as the provider's name, email, and whether they work alone or are associated with a company. It also includes a computational column that calculates the days between the scheduled date and the current date using days-from-duration(), date subtraction, and current-date().There is also a decision column to determine whether the associated assistance seeker needs rescheduling or if no action is required based on a specific threshold.

Challenges:

-Complex XPath expressions were needed to point to the exact service for the current provider ID.

-Initial attempts with count and date subtraction using functions were incorrect, so the transformation was simplified by removing those functions and using a simpler approach.

Common Challenges for Both Transformations(5 and 6) :

-Both transformations relied on date manipulations and functions not available in XSLT version 1.0 which is the default in Notepad++. Therefore, I had to use XSLT version 2.0 by downloading the Saxon JAR file and using a plugin in Notepad++ (NppExec) to run the transformations.

Steps to Run XSLT 2.0 Transformations in Notepad++:

Download Saxon JAR File: Download the Saxon HE JAR file (saxon-he-10.x.jar or later version) from Saxonica's official website.

Install NppExec Plugin: In Notepad++, install the NppExec plugin via Plugins > Plugins Admin > Search for NppExec > Install it.

Run the Transformation: Use NppExec to run the transformation with the following script:

cd "C:\path\to\saxon\folder"

java -jar saxon-he-10.x.jar -s:input.xml -xsl:transform.xsl -o:output.html

Replace C:\path\to\saxon\folder with the actual path to your Saxon JAR file.

Replace input.xml with the path to your XML file.

Replace transform.xsl with the path to your XSL file.

Replace output.html with the desired output file path.

Scenario 7: Transforming Assistance Seeker Data to Another XML Format.

In the 7 transformation, we aimed to convert the XML data related to Assistance Seekers and their associated Health Providers into a simplified XML format. The output of this transformation focuses on essential details (Assistance Seeker's name, age, associated Health Providers and their offered services). We looped through each Assistance Seeker in the XML data using the XPath for-each function, which allowed us to extract the Name, Age, and associated Health Providers for each seeker. The XSLT transformation was structured using templates to dynamically create elements. This includes creating the root element <AssistanceData>, and for each seeker, a nested <Seeker> element. Inside, we created <Provider> elements for each health provider associated with the seeker. For each provider, we also extracted the Service Type by checking the ProviderID against the available services. We used the <xsl:choose> and <xsl:when> statements to match the service type.

Scenario 8: Dynamic transformation of a part of Health Management System Data into JSON Format

This transformation aims to extract data in JSON format related to health providers and services. It also dynamically handles different provider types (e.g., service companies and individual providers). The main challenge of managing multiple health provider types was addressed using `xsl:when` within `xsl:choose`. Key steps included retrieving relevant data and dynamically looping through health providers and services to extract the necessary information.

6. CONCLUSION

In conclusion, the project successfully modeled and explored an XML database to support a collaborative platform for social and medical care tailored to disabled and elderly individuals. This platform brings together a wide range of services such as personal assistance, medical care, transportation, and support for caregivers. The developed XML schema represents the key entities and relationships involved in this ecosystem, including people requiring assistance, service providers, and the scheduling of activities. Additionally, by creating valid XML databases and designing XSL stylesheets for various data visualization scenarios, we demonstrated the adaptability of the data model to handle real-life queries and outputs in both XML and JSON formats.

The project further highlights the benefits of using XML for managing structured data in a collaborative healthcare environment, such as flexibility and interoperability, especially for complex service-based interactions. However, challenges arose when defining a unified model that could account for the diversity of services and dynamic participation of individuals and companies. The chosen modeling solutions addressed these challenges, balancing the need for simplicity with the capacity to handle a wide variety of scenarios. Through the detailed analysis of one particularly complex scenario, we demonstrated the robustness of our approach, showcasing how effective schema design and transformation tools can support real-time data exploitation and visualization in a sensitive and collaborative healthcare environment.

REFERENCE

1. Kahate, Atul. *XML and Related Technologies*. Pearson India, 2024. *Learning Oreilly*, https://learning.oreilly.com/library/view/xml-and-related/9781282652262/.
2. Iwashokun, Ade-Ibijola. Parsing of Research Documents into XML Using Formal Grammars, South Africa, 2024, https://www.researchgate.net/publication/377892002_Parsing_of_Research_Documents_into_XML_Using_Formal_Grammars#fullTextFileContent.