XDW: XML Data Warehousing

Georgio Yammine, Rami Naffah

Abstract— This report focuses on implementing a tool in Java that allows one or multiple users to store a huge number of versions while saving space. This can be achieved by using one of several approaches of XML versioning, which are explained in this paper. Moreover, this tool allows temporal querying, meaning that the user can search for a specific element in a certain version (past or present); two approaches were implemented and are explained in what follows: Filter-Based Querying and Keyword-Based Querying.

———————— ◆ ————————

CONTENTS

INTRODUCTION AND BACKGROUND

In recent years, cloud-based technology has revolutionized the world, allowing companies to easily store and retrieve data which they find important and valuable. Many global companies use data warehousing to organize data that streams from corporate branches around the world. A Data Warehouse can use techniques such as version control to maximize efficiency in terms of storage. In other words, it can reduce the amount of space used.
A Data Warehouse can help a corporation avoid various challenges. It delivers enhanced business intelligence, saves time, enhances data quality and consistency, as well as improves the decision-making process by maintaining a cohesive database of current and historical data.

Previous work has been done regarding diff and patch of tree structured xml documents. We exploited the well-known techniques for finding the edit distance between tree structures, to allow the comparison and differencing of semi-structured XML documents. Hence, we developed an application that compares two XML documents, provides the symmetric and reversible diff file containing the changes that can transform a file into the other, and that can apply these changes on a file to get the other.
Some of the several applications of XML comparison and diff are version control, change management and data warehousing. Therefore, in this study, we propose to exploit techniques for version control and data warehousing. Also, we will be managing finding, scoring and browsing changes between different versions of a document, as well as temporal queries and index maintenance.

Tools Used

To implement our project we used Java 8, JavaFX and FXML with JFoenix and TilesFX Libraries.

Application Configuration

Our application is file-based and saves all the data locally. Once the user creates a project, it will be saved in a directory of his choice. The project directory will consist of the following:

1- An “output” directory: user can get any version from the application. The latter will be saved under this directory.
2- A “src” directory. The latter contains two other directories: “available files” in which is saved the latest version, and “diffs” in which are saved the backward deltas which we can use to get previous versions when needed.
3- A project file named after the created project name. Its extension is “.xdw”.

XML Versioning

XML versioning is a technique used to store different versions of the XML data generated at different times. [1]
The objective of this part is to explain the different approaches used in XML versioning, while stating the advantages and disadvantages of each, as well as to justify the approach we used in our project.
The basic approach is to store all the different versions of an XML document. While this

approach provides easy access to all the versions at any time, it consumes a huge amount of space and allows data redundancy.
Another approach is to store the original (oldest) version with subsequent deltas between the versions added. That would allow the user to patch any document version using the deltas. However, there is a trade-off: reduction in storage usage implies additional processing required to compose deltas and to patch the original version with them to get the required version.
Last but not least, we decided to store the latest version with backward deltas because we considered that the user will usually need the newer versions in his projects more than the older, and therefore newer versions will require less number of patching (less amount of processing time) than older versions while also keeping the latest version available.

Temporal Querying

A user may want to access a particular version of a document and search for a specific XML element, or to search for specific changes between two versions of a document (what changed between version n and version n-1 for instance). In this section, we will explain how querying was implemented and applied in our application.

There are two approaches regarding this matter:

1- Filter-Based Querying
2- Keyword-Based Querying

Filter-Based Querying:

In this approach, the user is provided with the ability to filter the search by selecting values in the provided Combo boxes. Initially, the user is provided with one combo box, and depending on his choice, new combo boxes may appear.

We used XPath as our querying language. In our implementation, whenever the user selects a value in a combo box, the latter is added to the expression. After he is done, the user presses the "Query" button, and therefore the application returns the query result provided by the XPath expression. Note that in our application, the filter-based search can only be used whenever the "Changes Only" check box is selected. That is because our diff file format is known and limited, unlike other XML documents which nodes can be deeply nested and therefore, the filter-based approach would be limited. However, the "All" option is selected by default for non "Changes only" and by pressing the Query button, the application will provide the user with all the nodes contained in the selected version of an XML document.

Keyword-Based Querying:

As mentioned above, the filter-based approach has a major limitation. The keyword-based querying approach exceeds this limitation. A toggle button is provided to the user. When it is selected, the combo boxes disappear, and a search bar appears instead. The user has to write an XPath expression in the search bar to be able to search for specific elements in a version of an XML document. To make it user friendly, we implemented an Auto Completion Pop Up feature that would offer the user a list of suggestions/predictions. This will help him write a correct

XPath expression, as well as lets him know the available elements in the version rather than guessing what to search for.

The Auto Completion Pop Up works as follows:

1) Initially, when the user writes "/", a list appears showing all of the elements present in the document preceded by "//". For instance, "//book".
2) After choosing one of the choices, if the next character pressed is a "/", the Pop Up is updated with a list showing all the children of the previously chosen element. For example, "//book/page".
3) Moreover, if a "[" is added, the list will show us the possible positions of the last chosen element. However, if an "@" follows the "[" character, the Pop Up will be updated, listing all the attribute names of the last element entered.
4) Last but not least, if the user uses the Backspace key to erase some of the characters he wrote, the program will go back to the last entered character out of ("//", '/', '[' and '@') and update the list accordingly.

Application Features

We tried to make our app user friendly and appealing to the eye.

Once the app is launched the user is greeted with a welcome screen which has 2 tabs:

- Open Tab: includes a button to open a project from file and a table showcasing Recent and Pinned Projects for quick opening.
- New Tab: ask for Project Directory, Project Name, and Project Author and create a new project.

While a project is being loaded the app will display a Splash Screen Showing in animation a Welcome message and the name of the application with the project's name and author.

The main application is divided into 5 tabs.

- Home Tab: is made of 15 Tiles each having a certain functionality. Some of which are:
    - Space Saved Tile which shows the space saved by using this versioning technique over the default storing all the versions method. The result is computed based on the following formula: S = $1 - \frac{Project\ Directory\ Size}{Sum\ of\ all\ versions\ Size}$.
    - Changes Tile: It showcases the number of Updates, Insertions and Deletions in the last added version. If pressed, it will take the user to the Query Tab and display those changes in detail.
    - Calendar Tile: Shows the days a new version was added.

    and more…
- Version Tab: Shows the versions added in a table with the following information: Version number, author, date added, size, diff size, similarity, status, get button and query button.
    - The Get button will return that version in the output folder.
    - The Query button will select take to the query tab to query that version.

- User tab: in which the user selects an existing user or creates a new one. The tab also includes a Contribution Graph which shows the number of contributions made by that user in the past year, where each column represents a week and each rectangle represents a day which color concentration depends on the number of contributions made on that day.
- About Tab: which includes a description of our application.

Also, the app can be used as the default program for .xwd files and therefore the projects can be launched without having to open the app first.

The Application User Interface can be seen in Appendix A.

Conclusion

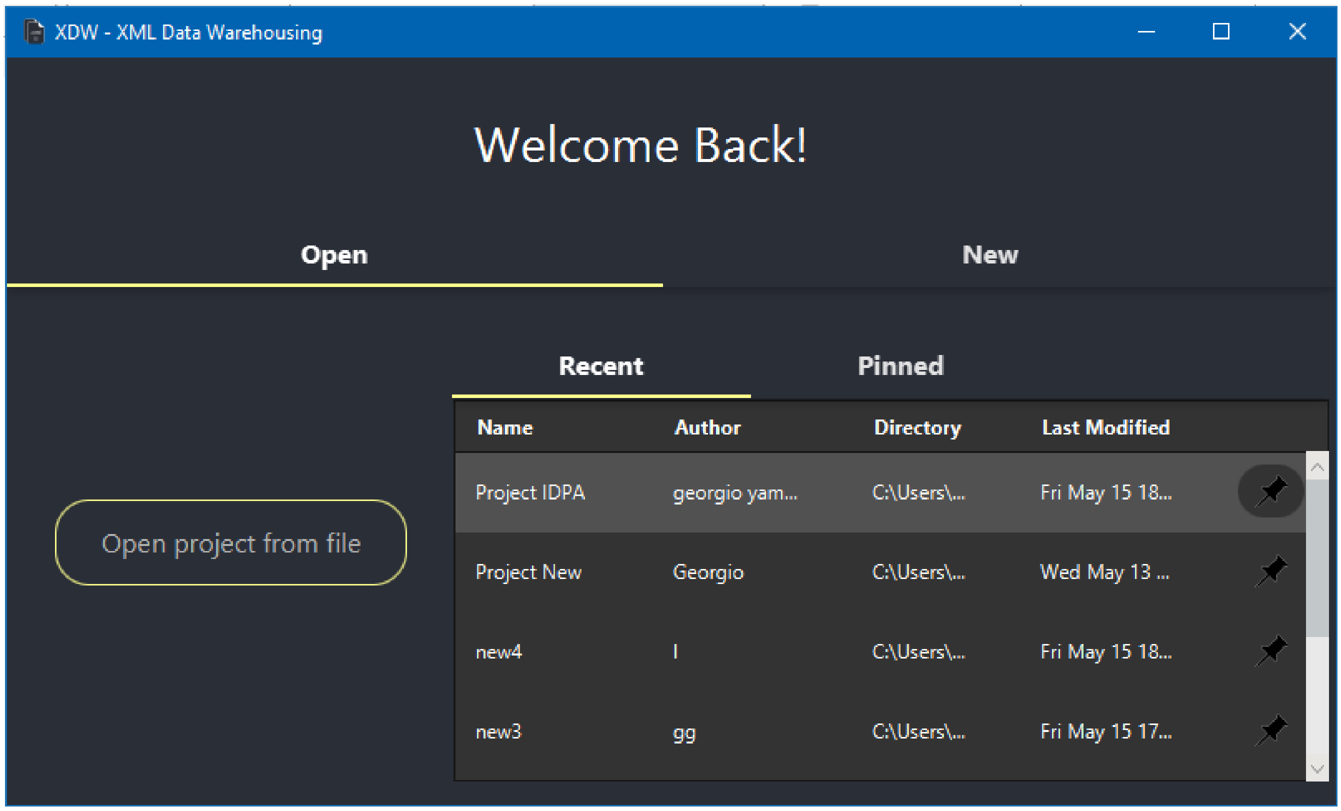
In recent years, the proliferation of XML data sources on the Web has introduced the need of tools that help in information retrieval, data comparison, manipulation and data warehousing. In this paper, we propose a tool for XML Data Warehousing, Version Control and Temporal Querying, where huge number of files or versions need to be stored. The tool we developed uses XML Diff and Patch to get diff files and therefore, save space. The application stores the last version of the document along with the backward deltas in order to query the past.

This tool can be further expanded by adding the following features: 1) Monitoring Changes, 2) Archiving and 3) Mirroring.

References

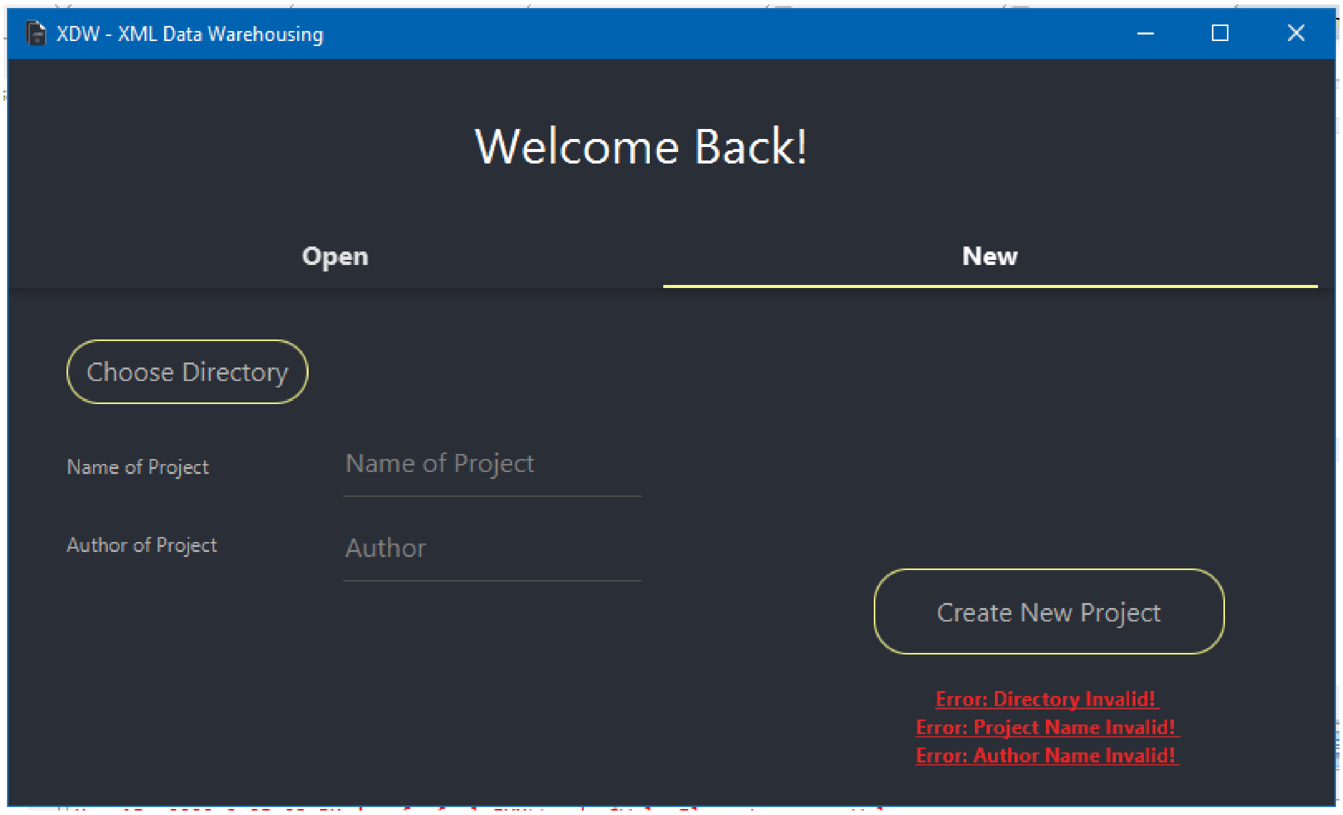
[1] Marian A.; Abiteboul S. and Mignet L., Change-Centric Management of Versions in an XML Warehouse. Proceedings of the International Conference on Very Large Data Bases (VLDB), pp. 581-590, 2001

Appendix A: Application Interface


XDW - XML Data Warehousing

Welcome Back!

Name	Author	Directory	Last Modified
Project IDPA	georgio yam...	C:\Users\...	Fri May 15 18...
Project New	Georgio	C:\Users\...	Wed May 13 ...
new4	l	C:\Users\...	Fri May 15 18...
new3	gg	C:\Users\...	Fri May 15 17...

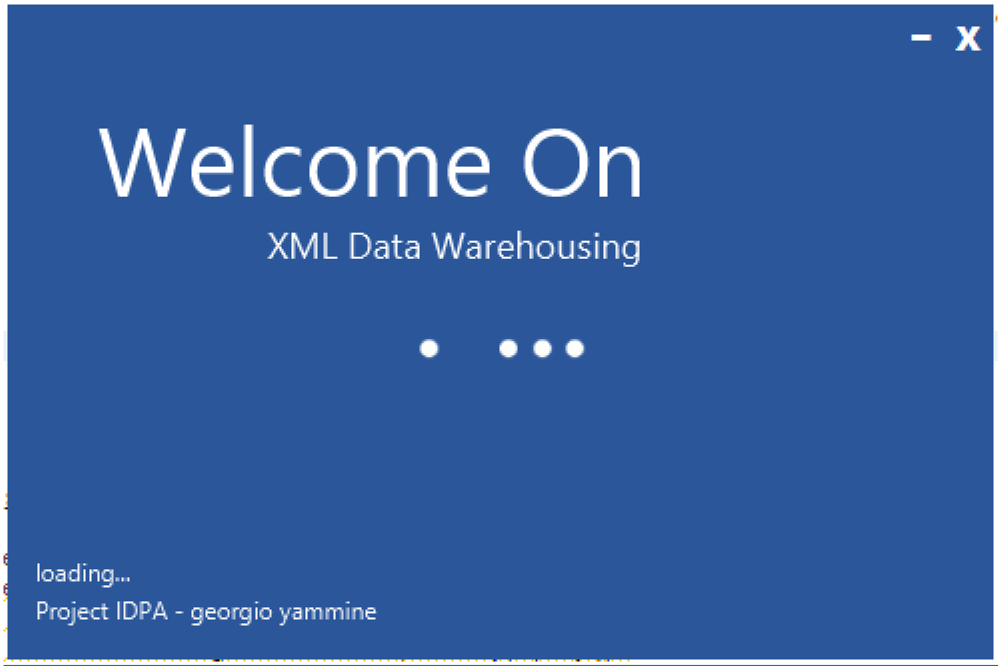

XDW - XML Data Warehousing

Welcome Back!

Name of Project: Name of Project

Author of Project: Author

Error: Directory Invalid!
Error: Project Name Invalid!
Error: Author Name Invalid!

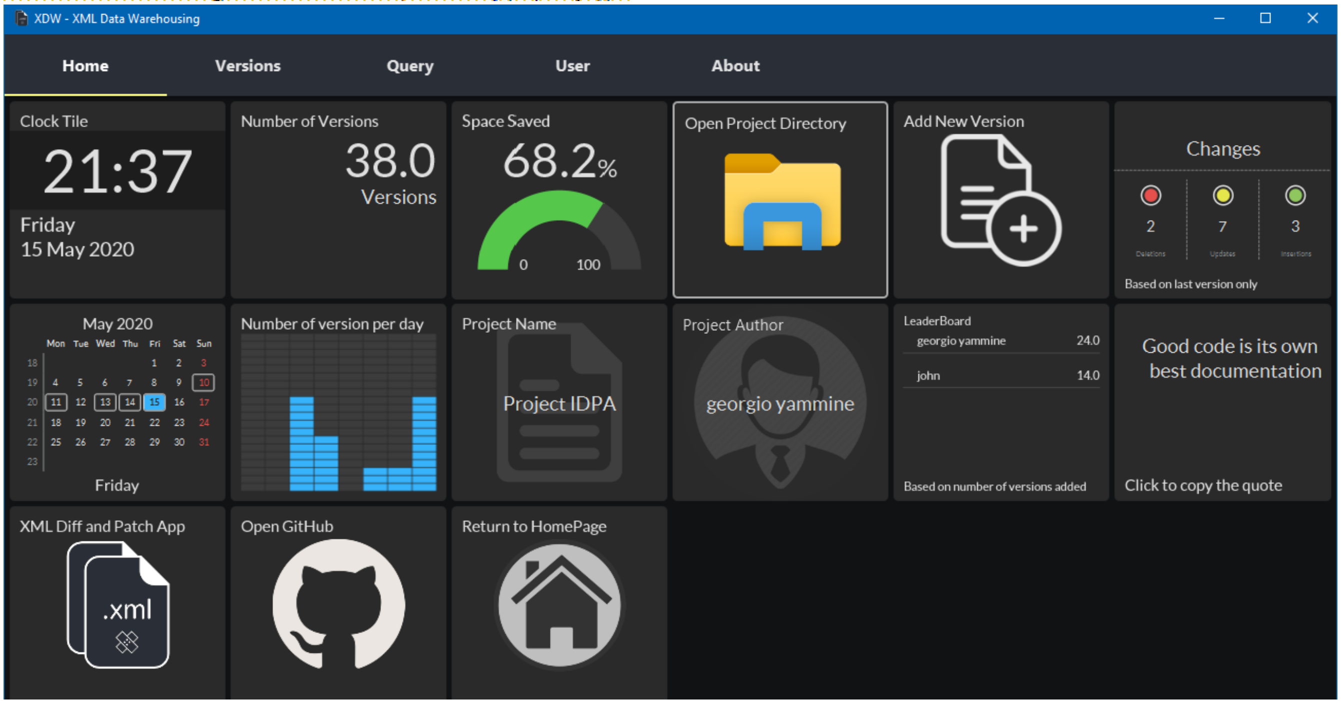


Welcome On

XML Data Warehousing

loading...

Project IDPA - georgio yammine



21:37

Friday
15 May 2020

38.0
Versions

68.2%

2 7 3

Based on last version only

Friday

Project IDPA

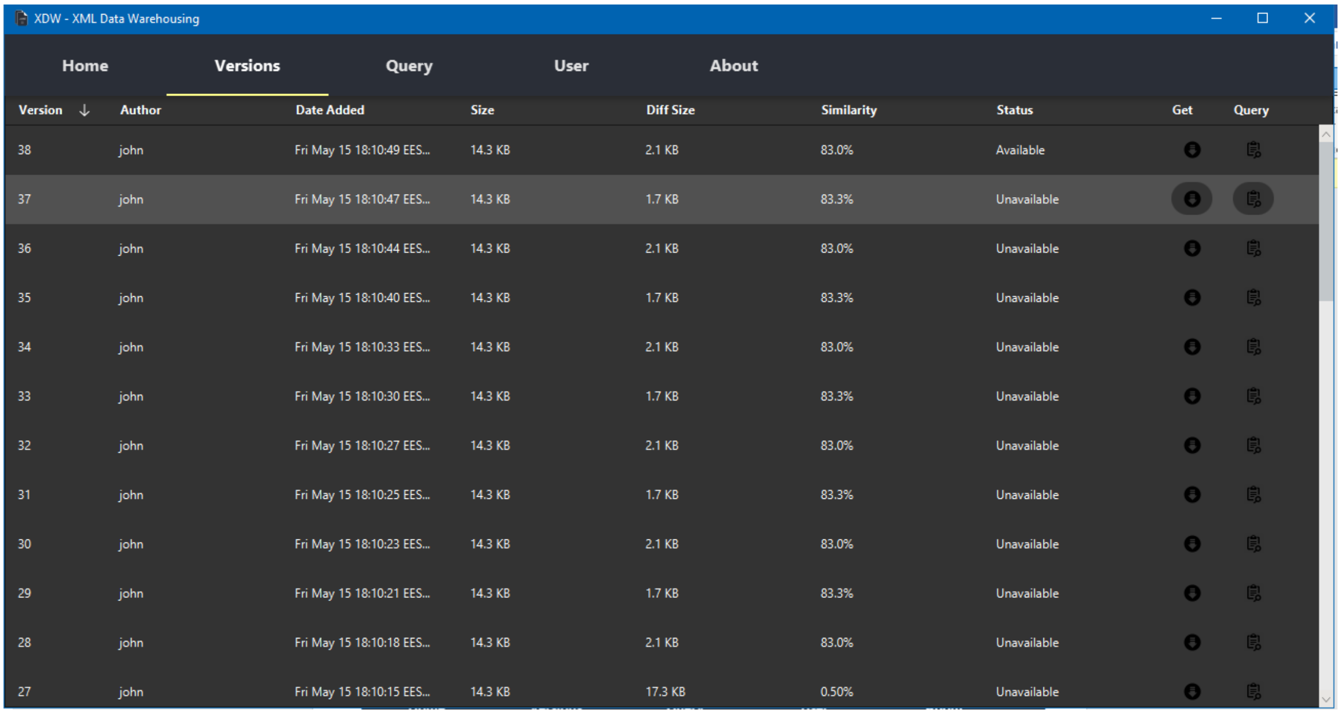
georgio yammine

georgio yammine	24.0
john	14.0

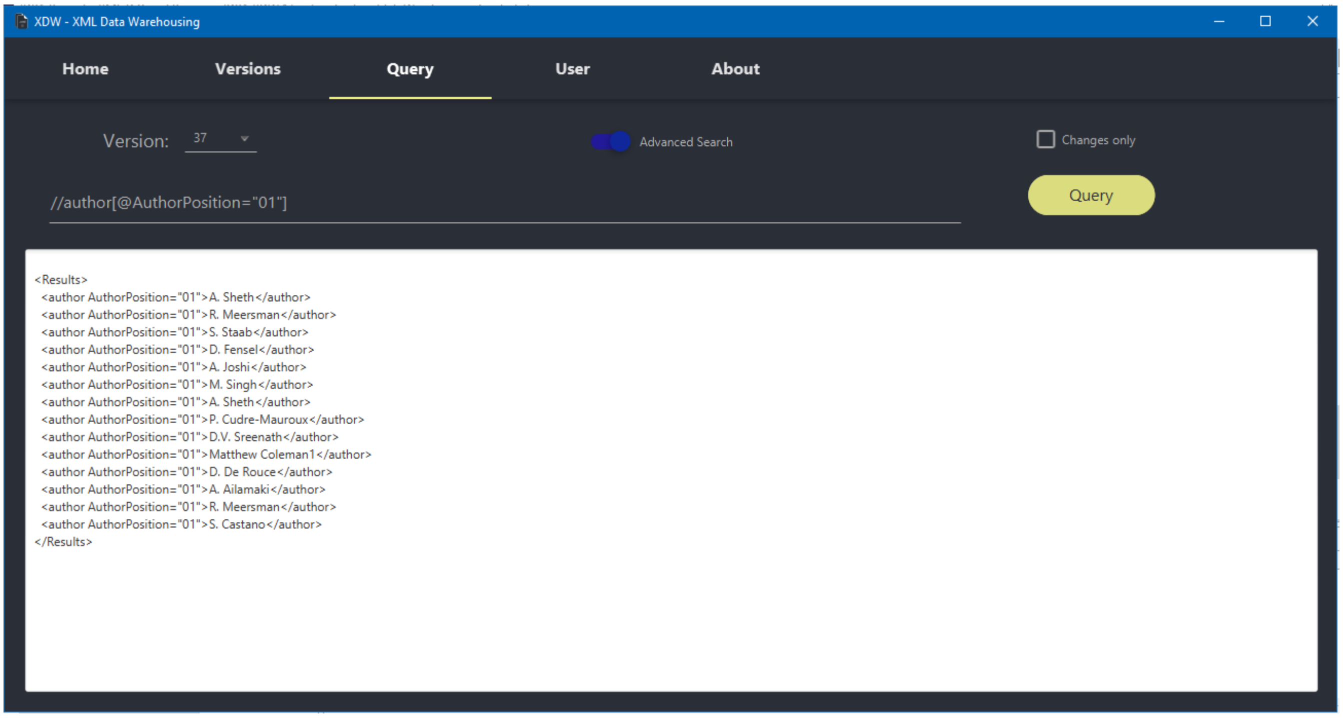
Based on number of versions added

Good code is its own best documentation

Click to copy the quote



Version	Author	Date Added	Size	Diff Size	Similarity	Status	Get	Query
38	john	Fri May 15 18:10:49 EES...	14.3 KB	2.1 KB	83.0%	Available		
37	john	Fri May 15 18:10:47 EES...	14.3 KB	1.7 KB	83.3%	Unavailable		
36	john	Fri May 15 18:10:44 EES...	14.3 KB	2.1 KB	83.0%	Unavailable		
35	john	Fri May 15 18:10:40 EES...	14.3 KB	1.7 KB	83.3%	Unavailable		
34	john	Fri May 15 18:10:33 EES...	14.3 KB	2.1 KB	83.0%	Unavailable		
33	john	Fri May 15 18:10:30 EES...	14.3 KB	1.7 KB	83.3%	Unavailable		
32	john	Fri May 15 18:10:27 EES...	14.3 KB	2.1 KB	83.0%	Unavailable		
31	john	Fri May 15 18:10:25 EES...	14.3 KB	1.7 KB	83.3%	Unavailable		
30	john	Fri May 15 18:10:23 EES...	14.3 KB	2.1 KB	83.0%	Unavailable		
29	john	Fri May 15 18:10:21 EES...	14.3 KB	1.7 KB	83.3%	Unavailable		
28	john	Fri May 15 18:10:18 EES...	14.3 KB	2.1 KB	83.0%	Unavailable		
27	john	Fri May 15 18:10:15 EES...	14.3 KB	17.3 KB	0.50%	Unavailable		



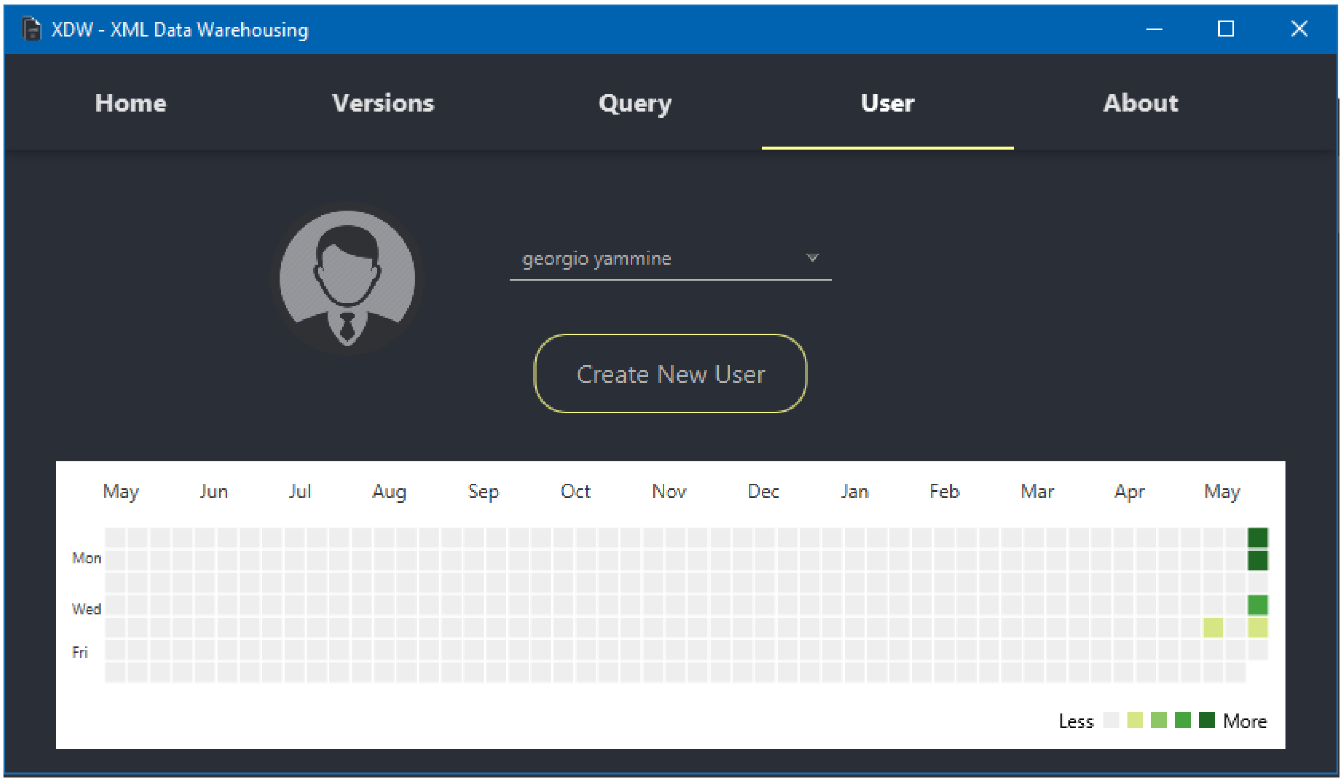
Version: 37

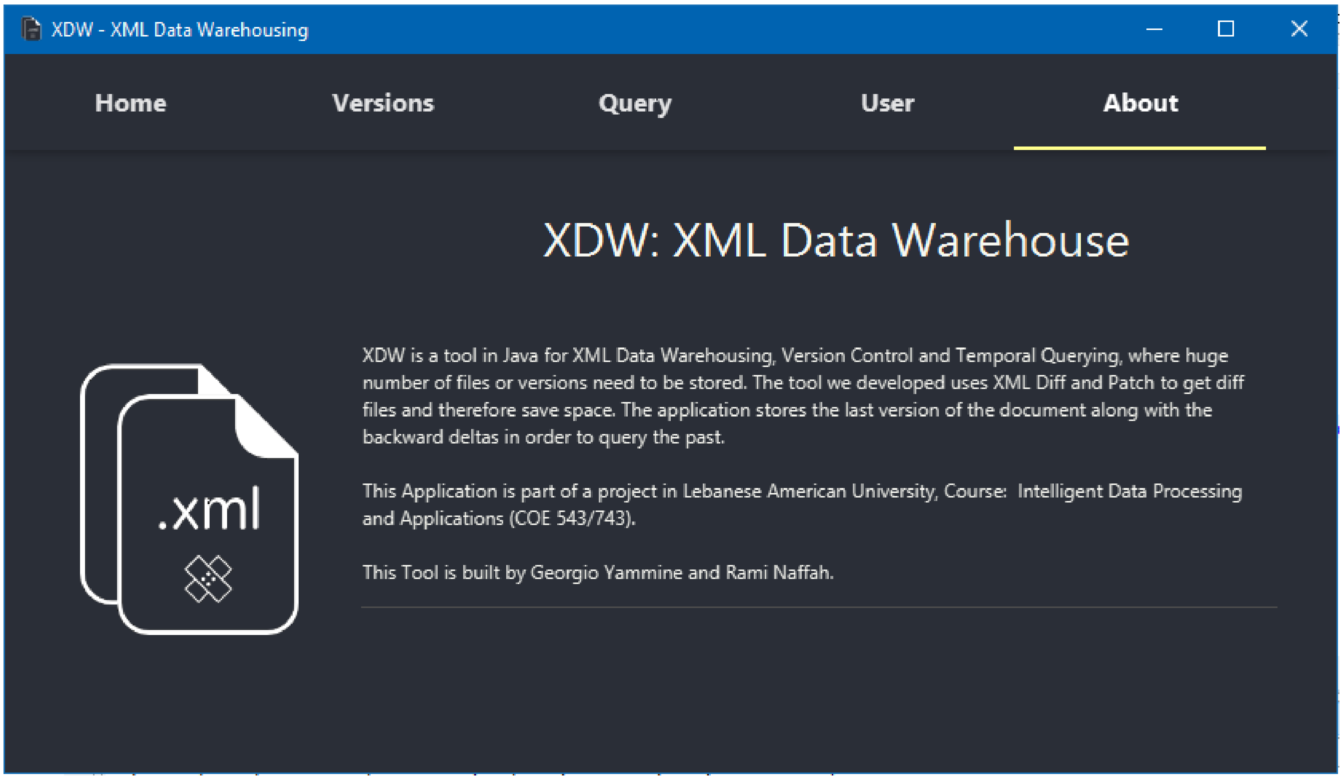
Advanced Search

Changes only

//author[@AuthorPosition="01"]

```
<Results>
  <author AuthorPosition="01">A. Sheth</author>
  <author AuthorPosition="01">R. Meersman</author>
  <author AuthorPosition="01">S. Staab</author>
  <author AuthorPosition="01">D. Fensel</author>
  <author AuthorPosition="01">A. Joshi</author>
  <author AuthorPosition="01">M. Singh</author>
  <author AuthorPosition="01">A. Sheth</author>
  <author AuthorPosition="01">P. Cudre-Mauroux</author>
  <author AuthorPosition="01">D.V. Sreenath</author>
  <author AuthorPosition="01">Matthew Coleman1</author>
  <author AuthorPosition="01">D. De Rouce</author>
  <author AuthorPosition="01">A. Ailamaki</author>
  <author AuthorPosition="01">R. Meersman</author>
  <author AuthorPosition="01">S. Castano</author>
</Results>
```


XDW - XML Data Warehousing
Home
Versions
Query
User
About
georgio yammine
Create New User
May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
Mon
Wed
Fri
Less
More

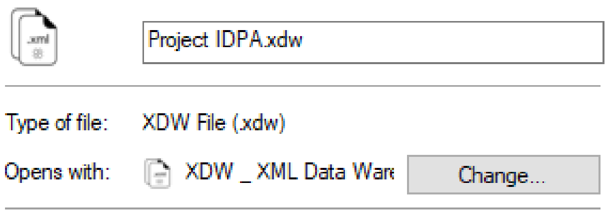


XDW: XML Data Warehouse

XDW is a tool in Java for XML Data Warehousing, Version Control and Temporal Querying, where huge number of files or versions need to be stored. The tool we developed uses XML Diff and Patch to get diff files and therefore save space. The application stores the last version of the document along with the backward deltas in order to query the past.

This Application is part of a project in Lebanese American University, Course: Intelligent Data Processing and Applications (COE 543/743).

This Tool is built by Georgio Yammine and Rami Naffah.



	Project IDPA.xdw
Type of file:	XDW File (.xdw)
Opens with:	XDW _ XML Data Ware Change...