

# **Enhancing XML Support in PostgreSQL**

## **CS297 Report**

**Khang Nguyen**

Khang Nguyen ([khangng@yahoo.com](mailto:khangng@yahoo.com))

San Jose State University

Advisor: Dr. Chris Pollett

Spring 2006

## **Abstract**

PostgreSQL is a database management system derived from the POSTGRES system developed at UC Berkeley. PostgreSQL is one of most popular open source databases available today. Currently, there are tools available to import XML data into PostgreSQL tables [IT02]. However, PostgreSQL does not support XML data natively. The ultimate goal of this project, which will be fully implemented in the next semester (CS298), is to extend PostgreSQL to natively store XML data and to implement some recent algorithms to allow for efficient XPath based retrieval of stored XML data [SL02]. In this paper, I will describe the work and the lessons learned from this semester (CS297), which served as the preparations for the CS298 project.

## Introduction

The main goal of the project is to extend PostgreSQL in the directions of natively storing XML data in tree-like structures, supporting XPath-based queries. The implementation portion of this CS297 semester can be divided into the three main deliverables. As the ground work for CS298, each deliverable has its own defined purposes and goals.

The goal of **Deliverable 1** was to import Dr. Pollett's digital document records, which are in an XML format, into a PostgreSQL database. The Deliverable 1 program scanned through the XML input file and removed invalid XML characters, then loaded up the appropriate JDBC driver for PostgreSQL database. The program cleared out data in the database by truncating the parent and child tables. The program maintained the referential relationships between the parent and child tables via reference keys. The program parsed the XML input file and imported the digital document records into the tables.

The goal of **Deliverable 2** was to develop a web-based application to search and display the digital document records imported into the PostgreSQL database by Deliverable 1. The Deliverable 2 program ran on web browsers and prompted an input box for users to enter their search criteria. It

supported the capabilities of searching for partial characters of a word, a complete word, or a phrase (a group of words). If the program found the matching records, it displayed the records in a clean and well defined format, which is easy to view. If the program found no matching records, it would display a message to let the users know so.

Last but not least, the goal of **Deliverable 3** was to implement a number of GiST (Generalized Search Tree) indexing functions. Deliverable 3 implemented a number of functions of Tetrahedron indexing schema such as *tetrahedron\_eq()*, *tetrahedron\_ne()*, *tetrahedron\_ge()*, *tetrahedron\_le()*, *tetrahedron\_area()*, *tetrahedron\_volume()*, and etc.

## **I. Deliverable 1**

### **1. Goal:**

The main goal of this deliverable here was to install and to get familiar with how the PostgreSQL database system works and behaves [KD06]. Another goal was to install and get familiar with:

- The most current JDK (Java Development Kit) 1.5,
- The JDBC driver for the PostgreSQL database connectivity
- The XPath functions in Xalan.jar.
-

## 2. Implementation:

The source code is composed of the following three Java files:

- *IConstantValues.java*
- *UploadRecs.java*
- *XmlFileProcessor.java*

The *IConstantValues.java* file contains constant values commonly used in XML file processing and database connectivity. The *UploadRecs.java* file acts as the main entry point of the program. It is optional to pass in the name of the XML file to process as the argument[0]. If no arguments are given, the program will look for and process the default XML file name.

First of all, the program scans through the input file and removes the invalid XML characters and writes the new contents to a new and temporary file.

Secondly, the program loads up the JDBC driver for PostgreSQL database.

Thirdly, it truncates the parent table, “*Digital\_Doc*” and the corresponding child table, “*Digital\_Allowed\_User*” before repopulating records into these

tables. Next, it instantiates an instance of the *xmlFileProcessor* class and execute the following functions in order. “*readWellFormedXmlFile*”

function will return the handle to the parsed XML document. Then,

“*processingXMLDoc*” function imports each digital document record into the parent table “*Digital\_Doc*” and imports the document record’s allowed

users data into the child table “*Digital\_Allowed\_User*”. The one-to-many relationship from the parent to the child table is maintained by the referential *DocID* keys. At the end, it displays the total number of records inserted into the parent table.

### 3. Diagram:

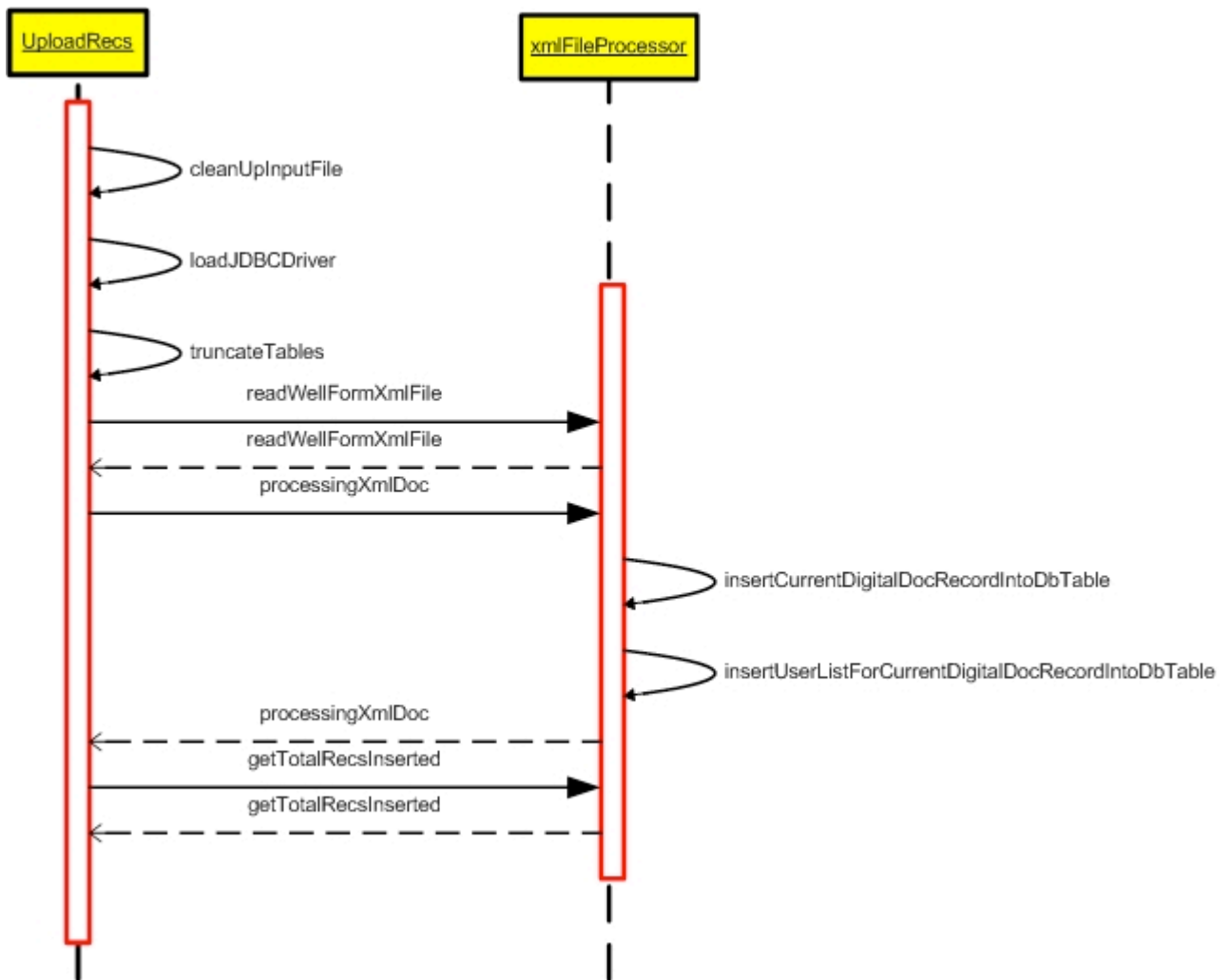
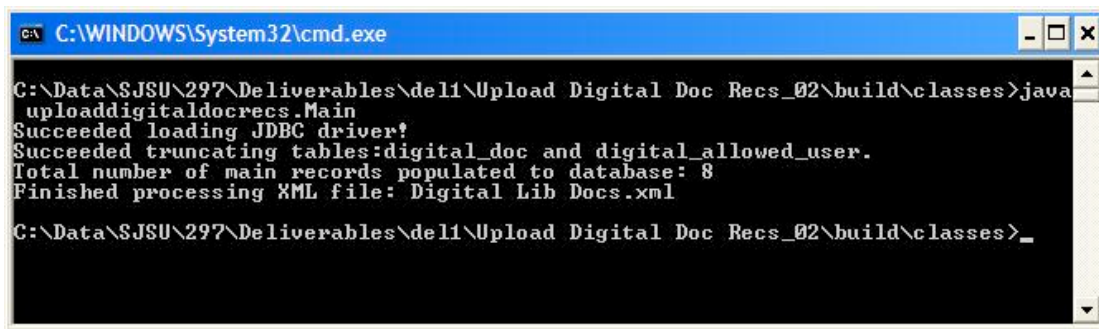


Figure 1: XML sequence diagram of Deliverable 1

## 4. Results:



```
C:\WINDOWS\System32\cmd.exe
C:\Data\SJSU\297\Deliverables\del1\Upload Digital Doc Recs_02\build\classes>java
uploaddigitaldocrecs.Main
Succeeded loading JDBC driver!
Succeeded truncating tables:digital_doc and digital_allowed_user.
Total number of main records populated to database: 8
Finished processing XML file: Digital Lib Docs.xml
C:\Data\SJSU\297\Deliverables\del1\Upload Digital Doc Recs_02\build\classes>_
```

**Figure 2:** The output on the given input file, *Digital Lib Docs.xml*.

## II. Deliverable 2

### 1. Goal:

The main goal of this deliverable was to develop a web-based application to search records in Dr. Pollett's digital document library, stored in the PostgreSQL database of deliverable 1. Another goal was to install and get familiar with

- Apache HTTP Web Server
- PHP Scripting Language
- Configurations between Apache HTTP Web Server, PHP module and PostgreSQL database.

## 2. Implementation:

The source code is contained in the following PHP script file:

- *search\_digital\_docs.php*

This PHP script files is organized into the following four distinguished sections:

- HTML section
- Search class
- Result class
- Db\_Connection class

From a start page on the internet browser, users enter some search criteria preferably relating to author last name, author first name, document title, citation, or description and click the *Search* button. The web server executes the script in the *search\_digital\_docs.php* file, in the following fashion.

First, the script displays the typical HTML head section including the document title.

Secondly, the script enters the HTML body section. Within the HTML body section, it instantiates a *search* object. It then executes the function *generate\_search\_section()* of this *search* object to generate an HTML



search form, which will collect users' search criteria through the input text box. Along with the input text box, the *Submit* button is for submitting the search request to the web server, and the *Reset* button is for resetting the contents of the input text box to the default value, which is blank in this case.

After the *search* object, it creates a *result* object and executes the function *generate\_result\_section()* of the *result* object. This function makes a connection to the database, assembles the SQL query, submits the query to the database, and receives the result set object returned from the database, and displays the results to the browser if there are any matching records in the result set object.

At the end of the script, it carries out some programming maintenance work by setting both the *search* object and *result* object to null values.

### 3. Diagram:

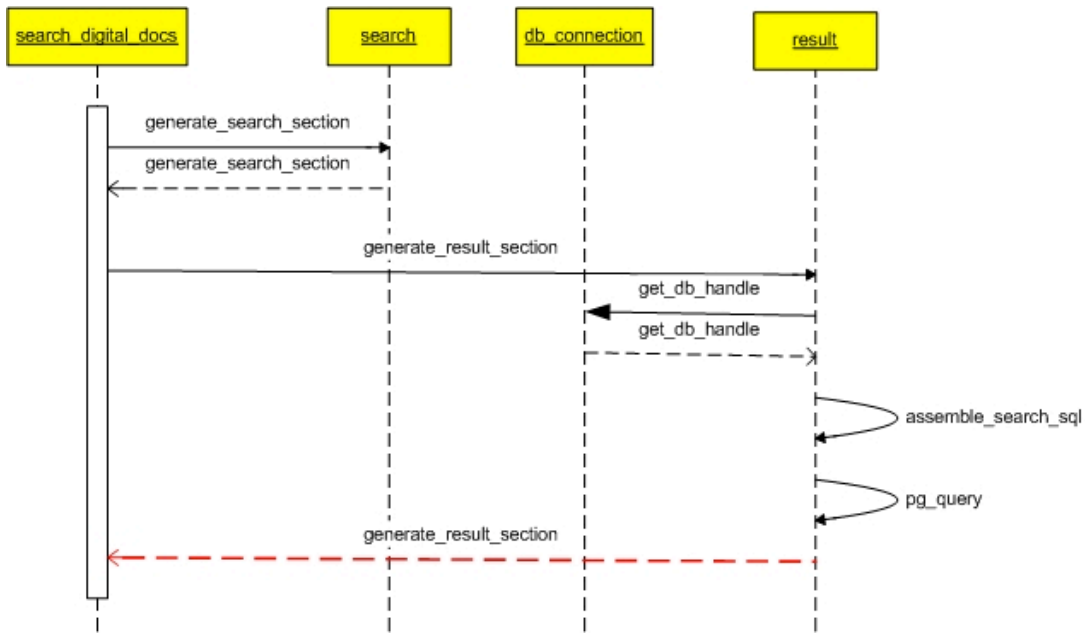


Figure 3: XML sequence diagram of Deliverable 2

### 4. Results:

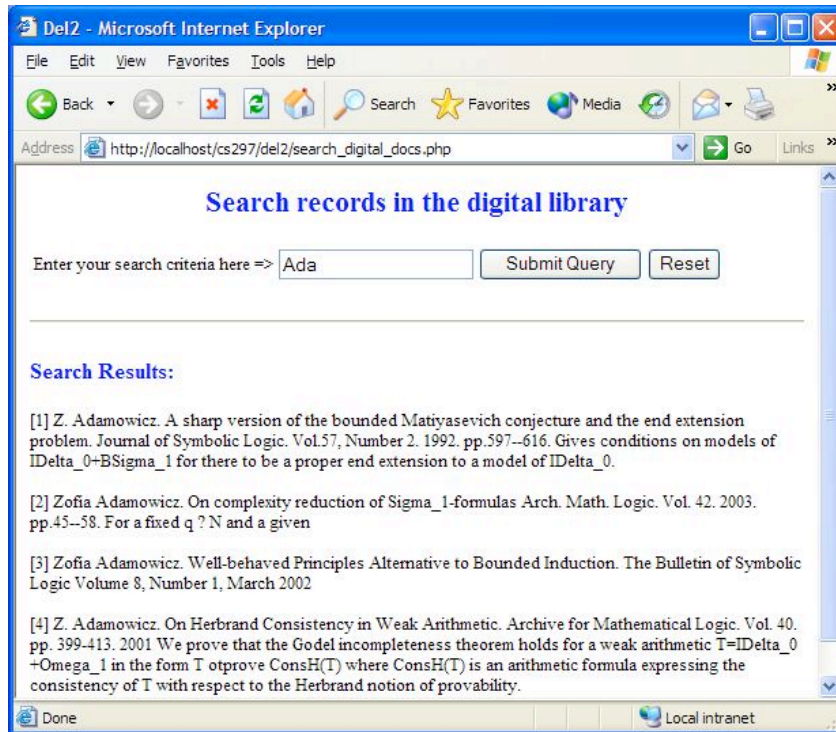
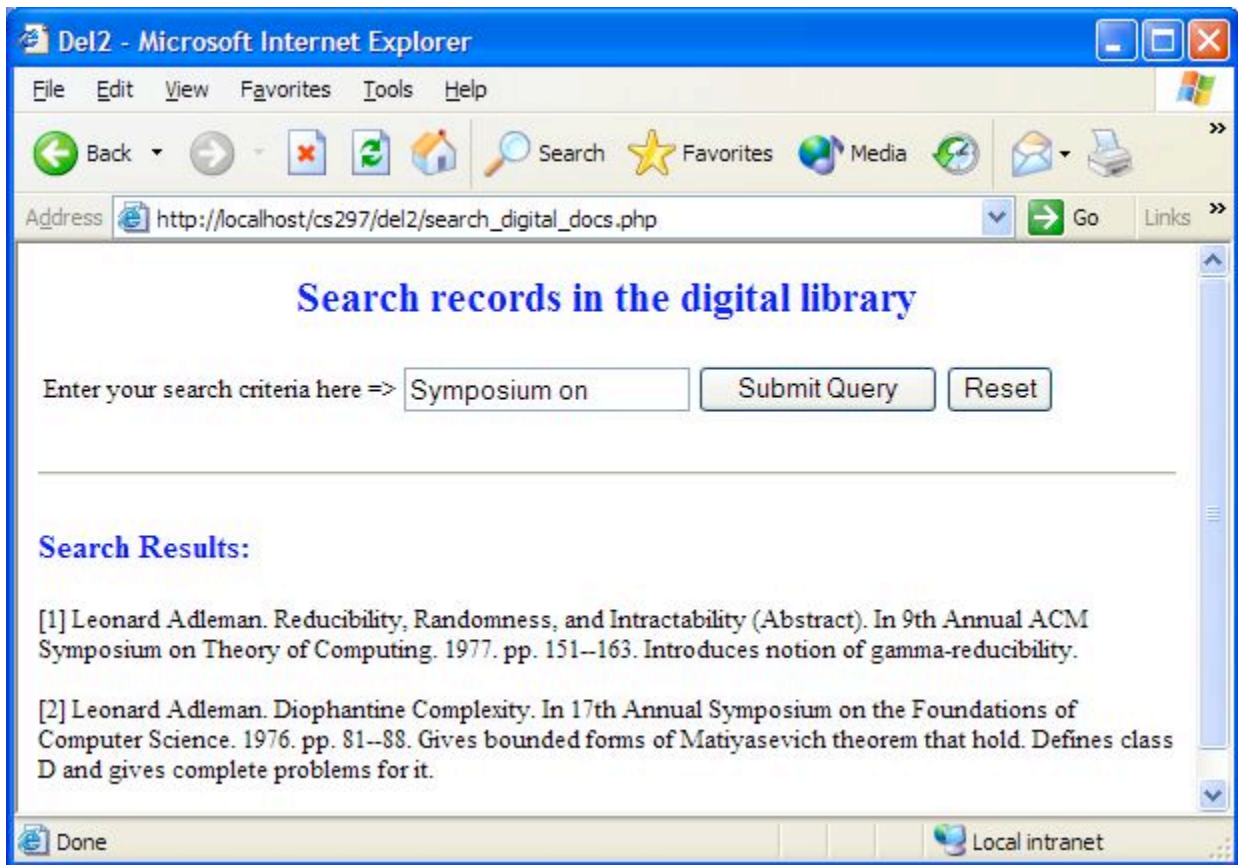


Figure 4: The output of a partial word search.



**Figure 5:** The output of a phrase of words search.

### III. Deliverable 3

#### 1. Goal:

The main goal of this deliverable was to get familiar with the GiST

(Generalized Search Tree) indexing schemes by

- Installing the *Cube* contributed module.
- Implementing a number of functions of Tetrahedron indexing schema.

## 2. Implementations:

These are the functions to implement for Tetrahedron indexing schema:

- FUNCTION *tetrahedron\_in(cstring)*
- FUNCTION *tetrahedron\_out(tetrahedron)*
- TYPE *Tetrahedron*
  - A user defined type storing the characteristics of tetrahedrons.
- FUNCTION *tetrahedron(text)*
  - Convert a text string into a tetrahedron object.
- FUNCTION *tetrahedron\_eq(tetrahedron, tetrahedron)*
  - Return true if the first tetrahedron is equal to the second tetrahedron.
- FUNCTION *tetrahedron\_ne(tetrahedron, tetrahedron)*
  - Return true if the first tetrahedron is not equal to the second tetrahedron.
- FUNCTION *tetrahedron\_ge(tetrahedron, tetrahedron)*
  - Return true if the first tetrahedron is greater than or equal to the second tetrahedron.
- FUNCTION *tetrahedron\_le(tetrahedron, tetrahedron)*
  - Return true if the first tetrahedron is less than or equal to the second tetrahedron.

- FUNCTION *tetrahedron\_area(tetrahedron)*
  - Compute the surface area of tetrahedron
  - $\text{Area} = \sqrt{3} * L^2$
- FUNCTION *tetrahedron\_volume(tetrahedron)*
  - Compute the volume of tetrahedron
  - $\text{Volume} = 1/12 * \sqrt{2} * L^3$

## **Future Work**

In the next semester of CS298, we will extend beyond the work done in this semester. We will extend PostgreSQL functionalities to natively store XML data by using the GiST (Generalized Search Tree) indexing schema. Unlike the traditional storing method of relational databases, we will store XML data in the tree-structured form. So, PostgreSQL can support XML data natively and efficiently. We will also implement some recent tree-structured algorithms [PDG05] to add new capabilities to PostgreSQL to support XPath based retrieval of XML data. With the new capabilities added, we should be able to retrieve XML data from

PostgreSQL database with more efficiency and flexibility via powerful XPath based queries.

## References

- [IT02] Igor Tatarinov, Stasis D. Viglas. Storing and Querying Ordered XML Using a Relational Database System. ACM SIGMOD. 2002.
- [SL02] Shiyong Lu, Yezhou Sun, Mustafa Atay, Farshad Fotouhi. A New Inlining Algorithm for Mapping XML DTDs to Relational Schemas. ANSI. 2002.  
<http://wwwedit.cs.wayne.edu:8080/~shiyong/papers/xsdm02.pdf>
- [PDG05] PostgreSQL Development Group. PostgreSQL 8.1.0 Documentation. 2005.
- [KD06] Korry Douglas, Susan Douglas. PostgreSQL. Sams Publishing. 2006.
- [EC01] Elizabeth Castro. XML For The World Wide Web. PitchPit Press. 2001.