Lab 3.2

Semi-Structured XML Processing

By: Christen Ford (2741896)

Date: 3/13/2019

Software Setup:

The various software utilized for this lab are detailed in this section. In order to reproduce these results, one must ensure that the following software is installed.

Software for the parser includes:

* Python3 (the latest version is preferred)
  + Parsel – Python3 module for parsing DOM objects.

Software for storing the data generated by the parser:

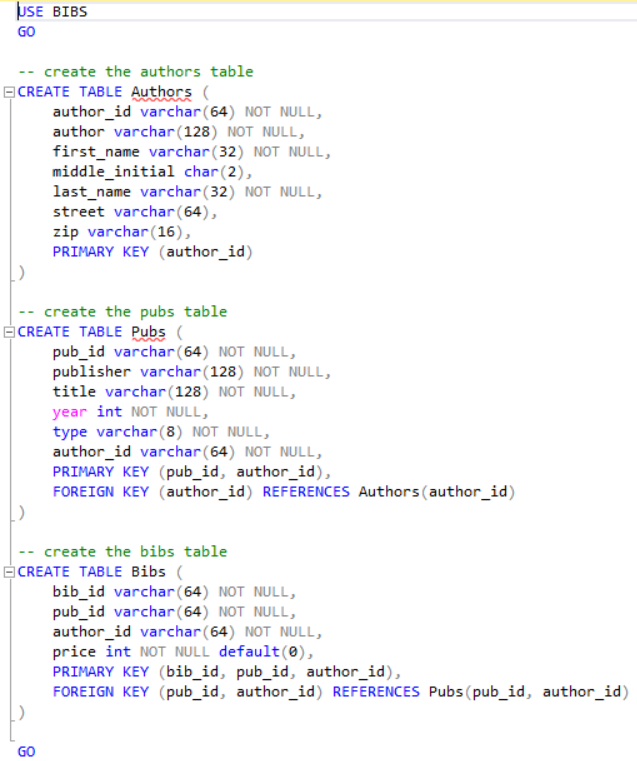
* Microsoft SQL Server 2017

The parser is platform independent. While it has been written to conform with the XML spec for the given input file for the lab. It should (untested) work with other XML files that follow the same XML schema as that defined in the input file. I chose to use XPath to navigate the document rather than a DOM parser.

I initially had written the parser to use a customized .NET XML Parser, but that ended up being cumbersome. The parser ended up needing to be highly coupled with the input document which was not what I wanted. I can do the parsing with XPath and Parsel in a much cleaner fashion that is much easier to modify in the future.

SQL Server Source Code and Output:

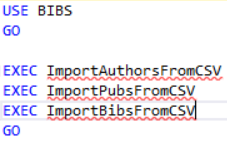
This is the SQL source code for the file “SQL/Create Database.sql”:



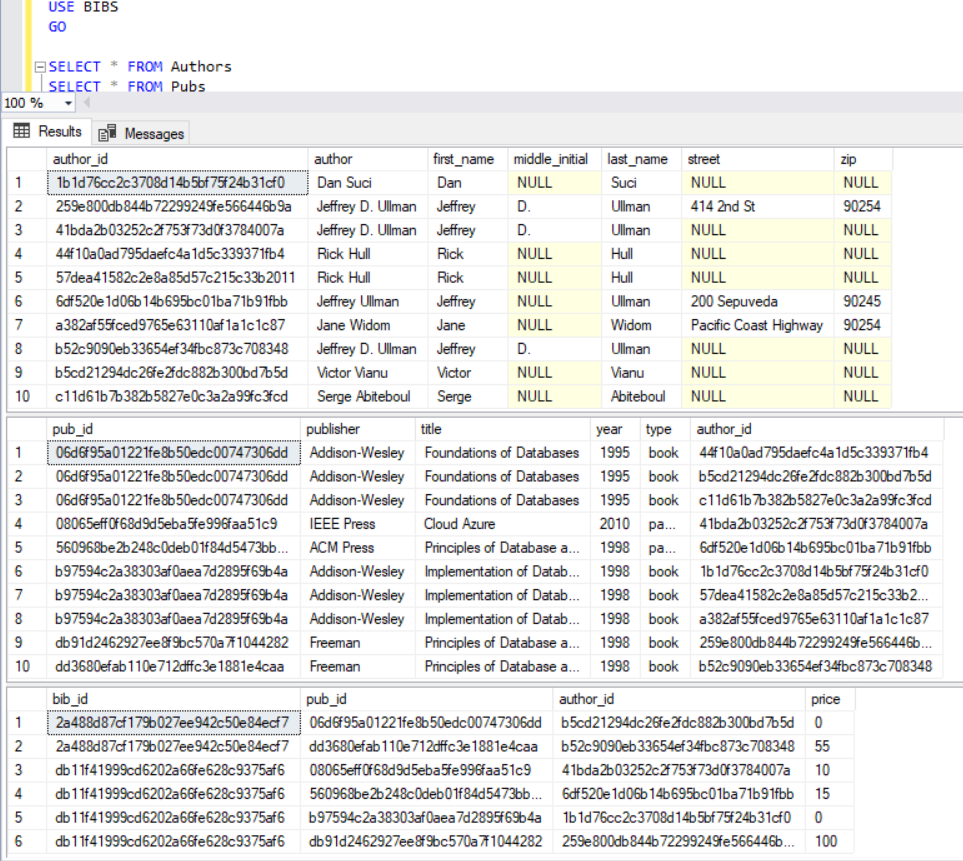
This is the SQL source code the file “SQL/Create Import Procedures.sql”:



This is the SQL source code for “SQL/Import Data From CSV.sql”:



Output Produced by the “SQL/Import Data From CSV.sql” file:



Source Code for the Parser (parser.py):

#! /usr/bin/python3

import csv

import hashlib

import random

import os

import parsel

import random

import string

# define variables needed for writing the CSVs

\_csv\_root = './CSV/'

\_bibs\_fname = 'bibs.csv'

\_pubs\_fname = 'pubs.csv'

\_authors\_fname = 'authors.csv'

# used during the id generation phase of bib csv creation

\_alphabet = list(string.ascii\_letters + string.digits)

# shuffle the letters

for \_ in range(7):

random.shuffle(\_alphabet)

def generate\_id():

'''

Generates a 'unique' identifier for a bibliography. These identifiers

are used to associate papers/books with a bibliography.

These are not supposed to be cryptographically 'secure', only that

they have a good chance to be unique.

'''

# try to get an md5 generator

m = hashlib.md5()

# can fail on platforms that are FIPS compliant, default to sha1

# if that is the case

if not m:

m = haslib.sha1()

# generate a 256 character length string

hash\_value = ''.join([random.choice(\_alphabet) for \_ in range(256)])

# digest the string

m.update(bytes(hash\_value.encode('utf-8')))

# return the digest as a string

return m.hexdigest()

def get\_name\_str(first, middle, last):

if len(middle) > 0:

return "{0} {1} {2}".format(first, middle, last)

else:

return "{0} {1}".format(first, last)

def write\_csv(bibs):

'''

Writes extracted XML data to the following CSV files:

bibs.csv: CSV containing all bibliography information for

each bibliography.

pubs.csv: CSV containing all publications listed by each

bibliography.

authors.csv: CSV containing the authors listed on each

publication.

Params:

bibs: A list containing bibliography entries.

A bibliography is a list containing a list of dictionaries where each

dictionary is a bibliography entry that is indexed by the following

keys:

publisher, title, year, authors, type, price

Note: The author key is a dictionary as well indexed with the following

keys:

first-name, middle-initial, last-name, address

In some cases, the address entry will be None (the author does not list

an address); in other cases, the address will be a two entry tuple

containing first the street, then the zip code.

'''

try:

# create the csv dir if it does not exist

if not os.path.exists(\_csv\_root):

os.mkdir(\_csv\_root)

# manually open each csv file

bibs\_file = open(\_csv\_root + \_bibs\_fname, 'w')

pubs\_file = open(\_csv\_root + \_pubs\_fname, 'w')

authors\_file = open(\_csv\_root + \_authors\_fname, 'w')

# create the csv objects

bibs\_writer = csv.writer(bibs\_file, quoting=csv.QUOTE\_MINIMAL)

pubs\_writer = csv.writer(pubs\_file, quoting=csv.QUOTE\_MINIMAL)

authors\_writer = csv.writer(authors\_file, quoting=csv.QUOTE\_MINIMAL)

# write the headers to each file

bibs\_writer.writerow(['bib\_id', 'pub\_id', 'author\_id', 'price'])

pubs\_writer.writerow(['pub\_id', 'publisher', 'title', 'year', 'type', 'author\_id'])

authors\_writer.writerow(['author\_id', 'author', 'first-name', 'middle-initial', 'last-name', 'street', 'zip'])

# loop through the bibliographies

for bib in bibs:

bibs\_id = generate\_id()

# loop through the publications

for pub in bib:

pub\_id = generate\_id()

pubs\_base\_record = []

pubs\_base\_record.append(pub\_id)

pubs\_base\_record.append(pub['publisher'])

pubs\_base\_record.append(pub['title'])

pubs\_base\_record.append(str(pub['year']))

pubs\_base\_record.append(pub['type'])

author\_id = None

for author in pub['authors']:

# generate an author id

author\_id = generate\_id()

author\_record = []

pubs\_record = pubs\_base\_record.copy()

auth\_name = get\_name\_str(author['first-name'], author['middle-initial'], author['last-name'])

author\_record.append(auth\_name)

author\_record.append(author['first-name'])

author\_record.append(author['middle-initial'])

author\_record.append(author['last-name'])

if author['address']:

author\_record.append(author['address'][0])

author\_record.append(str(author['address'][1]))

else:

author\_record.append('')

author\_record.append('')

authors\_writer.writerow([author\_id, \*author\_record])

pubs\_writer.writerow([\*pubs\_record, author\_id])

bibs\_writer.writerow([bibs\_id, pub\_id, author\_id, pub['price']])

# remember to close them so they write out properly

bibs\_file.close()

pubs\_file.close()

authors\_file.close()

except IOError:

# print an error message

print('An error occured while attempting to write the CSV files! Please try again.')

# delete the files if they exist, since they are likely corrupt

if os.path.exists(\_csv\_root + \_bibs\_fname):

os.remove(\_csv\_root + \_bibs\_fname)

if os.path.exists(\_csv\_root + \_pubs\_fname):

os.remove(\_csv\_root + \_pubs\_fname)

if os.path.exists(\_csv\_root + \_authors\_fname):

os.remove(\_csv\_root + \_authors\_fname)

def get\_name\_tuple(name\_parts):

'''

Gets an authors name give a tuple of name parts.

'''

first = name\_parts[0]

middle = name\_parts[1] if len(name\_parts) == 3 else ""

last = name\_parts[2 if len(name\_parts) == 3 else 1]

return (first, middle, last)

def extract\_author\_info(authors\_sel):

'''

Extracts author information given an author selector.

'''

authors = []

for author\_sel in authors\_sel:

author = dict()

# if there are no more elements, then are at a value node

if int(float(author\_sel.xpath("count(.//\*)").get())) == 0:

name = get\_name\_tuple(author\_sel.xpath("./text()").get().split(' '))

author['first-name'] = name[0]

author['middle-initial'] = name[1]

author['last-name'] = name[2]

# otherwise the author contains additional elements

else:

# possible combinations of names include <name> and

# <first-name>,<last-name>

# check for the first possibility

if int(float(author\_sel.xpath("count(.//name)").get())) > 0:

name = get\_name\_tuple(author\_sel.xpath(".//name/text()").get().split(' '))

author['first-name'] = name[0]

author['middle-initial'] = name[1]

author['last-name'] = name[2]

# otherwise it is the second one

else:

author['first-name'] = author\_sel.xpath('.//first-name/text()').get()

author['middle-initial'] = ''

author['last-name'] = author\_sel.xpath('.//last-name/text()').get()

# pull out the authors address if it exists

if int(float(author\_sel.xpath("count(.//address)").get())) > 0:

addr\_strt = author\_sel.xpath(".//street/text()").get()

addr\_zip = author\_sel.xpath(".//zip/text()").get()

author['address'] = (addr\_strt, addr\_zip)

else:

author['address'] = None

authors.append(author)

return authors

def extract\_pub\_info(pub\_sel):

'''

Extracts publication (book/paper) information given a publication

selector.

'''

pub\_info = dict()

pub\_info['publisher'] = pub\_sel.xpath('.//publisher/text()').get()

pub\_info['title'] = pub\_sel.xpath('.//title/text()').get()

pub\_info['year'] = pub\_sel.xpath('.//year/text()').get()

pub\_info['authors'] = extract\_author\_info(pub\_sel.xpath('.//author'))

return pub\_info

def main():

doc = ""

with open("bibs.xml", 'r') as xml:

for line in xml:

doc = doc + line.strip()

root\_selector = parsel.Selector(text=doc)

# holds publication information

bibs = []

# select all of the bib elements

for bib\_root in root\_selector.xpath('//bib'):

# prime a variable to hold the publication info

bib = []

#select all of the book elements

for book in bib\_root.xpath('.//book'):

pub\_info = extract\_pub\_info(book)

pub\_info['type'] = "book"

try:

pub\_info['price'] = book.attrib['price']

except KeyError:

pub\_info['price'] = 0

bib.append(pub\_info)

# select all of the paper elements

for paper in bib\_root.xpath('.//paper'):

pub\_info = extract\_pub\_info(paper)

pub\_info['type'] = "paper"

try:

pub\_info['price'] = paper.attrib['price']

except:

pub\_info['price'] = 0

bib.append(pub\_info)

bibs.append(bib)

# write the data to a few csv files

write\_csv(bibs)

def test():

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

#test()

Input File for the Parser (bibs.xml):

<?xml version="1.0" encoding="UTF-8"?>

<bibs>

<bib>

<book>

<publisher>Addison-Wesley</publisher>

<author>Serge Abiteboul</author>

<author>

<first-name>Rick</first-name>

<last-name>Hull</last-name>

</author>

<author>Victor Vianu</author>

<title>Foundations of Databases</title>

<year>1995</year>

</book>

<book price="55">

<publisher>Freeman</publisher>

<author>Jeffrey D. Ullman</author>

<title>Principles of Database and Knowledge Base Systems</title>

<year>1998</year>

</book>

</bib>

<bib>

<book>

<publisher>Addison-Wesley</publisher>

<author>Rick Hull</author>

<author>

<first-name>Jane</first-name>

<last-name>Widom</last-name>

<address>

<street>Pacific Coast Highway</street>

<zip>90254</zip>

</address>

</author>

<author>Dan Suci</author>

<title>Implementation of Databases</title>

<year>1998</year>

</book>

<book price="100">

<publisher>Freeman</publisher>

<author>

<name>Jeffrey D. Ullman</name>

<address>

<street>414 2nd St</street>

<zip>90254</zip>

</address>

</author>

<title>Principles of Database and Knowledge Base Systems</title>

<year>1998</year>

</book>

<paper price="15">

<publisher>ACM Press</publisher>

<author>

<name>Jeffrey Ullman</name>

<address>

<street>200 Sepuveda</street>

<zip>90245</zip>

</address>

</author>

<title>Principles of Database and Knowledge Base Systems</title>

<year>1998</year>

</paper>

<paper price="10">

<publisher>IEEE Press</publisher>

<author>Jeffrey D. Ullman</author>

<title>Cloud Azure</title>

<year>2010</year>

</paper>

</bib>

</bibs>

Output Created by the Parser:

The contents of ‘CSV/authors.csv’:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| author\_id | author | first-name | middle-initial | last-name | street | zip |
| c11d61b7b382b5827e0c3a2a99fc3fcd | Serge Abiteboul | Serge |  | Abiteboul |  |  |
| 44f10a0ad795daefc4a1d5c339371fb4 | Rick Hull | Rick |  | Hull |  |  |
| b5cd21294dc26fe2fdc882b300bd7b5d | Victor Vianu | Victor |  | Vianu |  |  |
| b52c9090eb33654ef34fbc873c708348 | Jeffrey D. Ullman | Jeffrey | D. | Ullman |  |  |
| 57dea41582c2e8a85d57c215c33b2011 | Rick Hull | Rick |  | Hull |  |  |
| a382af55fced9765e63110af1a1c1c87 | Jane Widom | Jane |  | Widom | Pacific Coast Highway | 90254 |
| 1b1d76cc2c3708d14b5bf75f24b31cf0 | Dan Suci | Dan |  | Suci |  |  |
| 259e800db844b72299249fe566446b9a | Jeffrey D. Ullman | Jeffrey | D. | Ullman | 414 2nd St | 90254 |
| 6df520e1d06b14b695bc01ba71b91fbb | Jeffrey Ullman | Jeffrey |  | Ullman | 200 Sepuveda | 90245 |
| 41bda2b03252c2f753f73d0f3784007a | Jeffrey D. Ullman | Jeffrey | D. | Ullman |  |  |

The contents of ‘CSV/pubs.csv’:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| pub\_id | publisher | title | year | type | author\_id |
| 06d6f95a01221fe8b50edc00747306dd | Addison-Wesley | Foundations of Databases | 1995 | book | c11d61b7b382b5827e0c3a2a99fc3fcd |
| 06d6f95a01221fe8b50edc00747306dd | Addison-Wesley | Foundations of Databases | 1995 | book | 44f10a0ad795daefc4a1d5c339371fb4 |
| 06d6f95a01221fe8b50edc00747306dd | Addison-Wesley | Foundations of Databases | 1995 | book | b5cd21294dc26fe2fdc882b300bd7b5d |
| dd3680efab110e712dffc3e1881e4caa | Freeman | Principles of Database and Knowledge Base Systems | 1998 | book | b52c9090eb33654ef34fbc873c708348 |
| b97594c2a38303af0aea7d2895f69b4a | Addison-Wesley | Implementation of Databases | 1998 | book | 57dea41582c2e8a85d57c215c33b2011 |
| b97594c2a38303af0aea7d2895f69b4a | Addison-Wesley | Implementation of Databases | 1998 | book | a382af55fced9765e63110af1a1c1c87 |
| b97594c2a38303af0aea7d2895f69b4a | Addison-Wesley | Implementation of Databases | 1998 | book | 1b1d76cc2c3708d14b5bf75f24b31cf0 |
| db91d2462927ee8f9bc570a7f1044282 | Freeman | Principles of Database and Knowledge Base Systems | 1998 | book | 259e800db844b72299249fe566446b9a |
| 560968be2b248c0deb01f84d5473bb61 | ACM Press | Principles of Database and Knowledge Base Systems | 1998 | paper | 6df520e1d06b14b695bc01ba71b91fbb |
| 08065eff0f68d9d5eba5fe996faa51c9 | IEEE Press | Cloud Azure | 2010 | paper | 41bda2b03252c2f753f73d0f3784007a |

The contents of ‘CSV/bibs.csv’:

|  |  |  |  |
| --- | --- | --- | --- |
| bib\_id | pub\_id | author\_id | price |
| 2a488d87cf179b027ee942c50e84ecf7 | 06d6f95a01221fe8b50edc00747306dd | b5cd21294dc26fe2fdc882b300bd7b5d | 0 |
| 2a488d87cf179b027ee942c50e84ecf7 | dd3680efab110e712dffc3e1881e4caa | b52c9090eb33654ef34fbc873c708348 | 55 |
| db11f41999cd6202a66fe628c9375af6 | b97594c2a38303af0aea7d2895f69b4a | 1b1d76cc2c3708d14b5bf75f24b31cf0 | 0 |
| db11f41999cd6202a66fe628c9375af6 | db91d2462927ee8f9bc570a7f1044282 | 259e800db844b72299249fe566446b9a | 100 |
| db11f41999cd6202a66fe628c9375af6 | 560968be2b248c0deb01f84d5473bb61 | 6df520e1d06b14b695bc01ba71b91fbb | 15 |
| db11f41999cd6202a66fe628c9375af6 | 08065eff0f68d9d5eba5fe996faa51c9 | 41bda2b03252c2f753f73d0f3784007a | 10 |