A picture containing text, queen, clipart

Description automatically generated

**RIGA TECHNICAL UNIVERSITY  
Faculty of Computer Science and Information Technology**

**Institute of Applied Computer Systems**

**DSP797 – Data Models in Database Systems**

**Practical Work 3: Document databases and hierarchical data model**

Variant 01.xml

**Student Name:** Balasuriyage Aritha Dewnith

KUMARASINGHE **Student Card No:** 203AEB014

2023./2024.

Table of Contents

[Introduction 3](#_Toc124717276)

[1.Xquery 3](#_Toc124717277)

[2.MongoDB 5](#_Toc124717278)

[Conclusion 8](#_Toc124717279)

# 

# Introduction

The goal of this practical work is to get familiar with implementation and querying of NoSQL databases. The first task of the assignment consists of writing either XPaths or XQuerys in order to retrieve the required information from the given XML document. The second part requires the conversion of the aforementioned XML file to a JSON file and then to upload said file into a MongoDB, and then to run of MongoDB queries in order to retrieve data.

# 1.Xquery

In order to complete this task that requires the querying of a XML document within a NoSQL database software which in this case will be eXist. The queries for each of the task have been implemented as functions within the same document.

Code:

**declare variable $file:= doc("01.xml");**

**declare function local:getnameatindex($p as xs:decimal?)**

**as xs:string?**

**{**

**let $name := $file/companies/company[$p]/name/text()**

**return $name**

**};**

**declare function local:getallatindex($p as xs:decimal?)**

**as xs:string?**

**{**

**let $data := data($file/companies//company[$p])**

**return $data**

**};**

**declare function local:getssn($p as xs:string?)**

**as xs:anyAtomicType\***

**{**

**let $data := doc("01.xml")/companies/company[name=$p]//employee/ssn**

**for $ssn in $data**

**return $ssn**

**};**

**declare function local:getemployee($id as xs:string?,$cnt as xs:string?)**

**as xs:anyAtomicType\***

**{**

**let $emps := doc("01.xml")/companies/company[@id=$id]//employee**

**for $emp in $emps**

**where $emp/@country/string()=$cnt**

**return $emp**

**};**

**<company\_name>{data(local:getnameatindex(21))}</company\_name>,**

**<company\_data>{local:getallatindex(12)}</company\_data>,**

**<employee\_ssn>{local:getssn("Vipe")}</employee\_ssn>,**

**<employee\_details>{local:getemployee("13","Germany")}</employee\_details>**

Result:

Graphical user interface, text, application, email

Description automatically generated

# 

# 2.MongoDB

The second task was implemented solely using python. The implementation required the use of various python libraries. Each of the subtasks such as the conversion of the XML to JSON and the inserting of said file into MongoDB are implemented within python script.

Code:

import json

import pymongo

from pymongo import MongoClient

import xmltodict

import urllib3

import traceback

def xml\_to\_json():

#The code below is meant to read the xml file directly from the web link but due to reasons beyond me it returns a invalid token error

#url = 'https://estudijas.rtu.lv/pluginfile.php/3466435/question/questiontext/3871149/1/5415911/01.xml'

#http = urllib3.PoolManager()

#response = http.request('GET', url)

#try:

# data\_dict = xmltodict.parse(response.data)

#except:

# print("Failed to parse xml from response (%s)" % traceback.format\_exc())

with open("01.xml") as xml\_file:

data\_dict = xmltodict.parse(xml\_file.read())

xml\_file.close()

json\_data = json.dumps(data\_dict,indent=2)

with open("data1.json", "w") as json\_file:

json\_file.write(json\_data)

json\_file.close()

return 0

def open\_json(file\_name):

with open(file\_name) as file:

file\_data=json.load(file)

return file\_data

def connect\_mongodb():

uri = "mongodb+srv://Aritha\_K:12345@cluster0.bjpmnx8.mongodb.net/test?ssl=true&ssl\_cert\_reqs=CERT\_NONE"

try:

client = MongoClient(uri,

connectTimeoutMS=30000,

socketTimeoutMS=None)

print("Connection successful")

except:

print("Unsuccessful")

return client

def insert\_mongodb\_json():

client = connect\_mongodb()

db=client["03"]

collection=db["03"]

collection.drop()

collection.insert\_one(open\_json('data.json'))

#print(collection.find\_one())

return collection

if \_\_name\_\_ == "\_\_main\_\_":

xml\_to\_json()

collection=insert\_mongodb\_json()

query\_1=collection.find\_one({}, { "companies.company": { "$slice": [20, 1] } })

print('Result of query one:')

print(query\_1["companies"]["company"][0]["name"])

print('Result of query two:')

query\_2=collection.find\_one({}, { "companies.company": { "$slice": [11, 1] } })

print(query\_2)

print("Result of 3:")

pipeline = [

{

"$unwind": "$companies.company"

},

{

"$unwind": "$companies.company.employees.employee"

},

{

"$project": {

"employee\_name": "$companies.company.employees.employee.name",

"company\_name": "$companies.company.name"

}

}

]

query\_3 = collection.aggregate(pipeline)

for r in query\_3:

if r["company\_name"]== "Vipe":

print(r["employee\_name"])

Result:

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Conclusion

While doing this assignment I was introduced to NoSQL databases specifically XML and JSON. When it comes to XML I learned how to run XQuerys and Xpaths in order to retrieve information from XML files using a software that is designed for semi structured XML files called eXist. XQuery is interesting since it allows the implementation of functions with the same script file which is something that I haven’t experienced when using SQL. XML certainly follows a format that is very different to a conventional database and the same is true for JSON but those documents are very similar to python dictionaries. Which is somewhat easier to sort through using python list/dictionary indexing but quite channelling to query using MongoDB. Given this difficulty a combination of MongoDB queries to retrieve data and then cutting out (peeling away the unnecessary dictionary and list layers) the clutter using python was necessary.

Overall, I can see the utility of NoSQL within document databases, and I was very interested in exploring its functionality even further. As far as document structuring, I cannot conceive of a better way to organise text based files than JSON, and given that you can use python indexing.