100 points + 10 bonus points(8.75% of course grade)
Assigned: Sunday, April 9
Due: Monday, April 24, 11:55 pm EST

This homework should be done in parts as soon as relevant topics are covered in lectures and the corresponding parts are posted. If you wait until the last minute, you might be overwhelmed.

For some problems, you will need to use Gradiance. Access Gradiance via the "Gradiance" link on the course website. There is no need to turn in anything else for these problems; your scores will be tracked automatically.

For other problems, you will need to turn in the required files/codes electronically **on Sakai**. Please make sure that ***all*** required files have been uploaded in each resubmission if you submit multiple times.

Problem3 and extra credit problem should be completed on your local VM. Before you start working on these two problems, be sure to run the following command on your VM to sync and receive updates:

/opt/dbcourse/sync.sh

Problem 1 (20 points)
Complete the Gradiance homework titled "Homework 4.1 (Transactions)."

Problem 2 (24 points)
Complete the Gradiance homework titled "Homework 4.2 (XML)."

Problem 3 (56 points)
In /opt/dbcourse/assignments/hw4/ on your VM, you will find an XML file **congress.xml** containing information about the last (114th) US Congress. Logically, the file consists of two sections:

- Each person element under congress/people stores information about a legislator, including the roles he or she has served in the Congress. A role with type "rep" indicates a Representative (member of the House), while a role with type "sen" indicates a Senator (member of the Senate). A role is current if its current attribute equals 1.
- Each committee element under congress/committees stores information about a committee. It has a list of members, whose ids reference those of person elements in the first section; role specifies the role of the member in the committee (e.g., chair or ranking member). Oftentimes a committee can be divided into subcommittees. Each subcommittee element has its own list of members, which should be a subset of the committee members. A legislator can serve on multiple committees, and even multiple subcommittees under the same committee.
    - Note that there are some "dangling" person references from under congress/committees. Representative Mark Takai passed away in May 2016; Representative Ed Whitfield resigned in September 2016. But they still remained on some committee rosters.

Please refer to the document "XML Tips" on the course Web site for instructions on running saxonb-xquery, the Saxon XQuery processor. Write queries in XQuery to answer the following questions. Because Saxon does not use any indexes and does not have a sophisticated optimizer, query performance may be heavily influenced by the way you write your queries. If a particular query takes forever to run, consider reordering loops and evaluating selections (filters) as early as possible. Note that you can add comments to your queries by enclosing them in "(:" and ":)".

For each question below, say (a), write your XQuery in a file named 2a.xq, and generate the output file 2a.xml by running
saxonb-xquery -s /opt/dbcourse/assignments/hw4/congress.xml 2a.xq > 2a.xml

Turn in all your .xq and output .xml files.

(a) Find all legislators whose name ends with "Price". (For each of them, simply print the entire person element.) You can use **ends-with(str1, str2)** to test if str1 ends with str2.
(b) Find the legislator who serves as the role of "Chairman" on the Senate Select Committee on Intelligence (code name "**SLIN**"). Simply print the entire person element.
(c) List all current Senators born before 1942. Format each of them as an element of the form **<senator name="…"/>.** You can use **xs:date("1941-12-31")** < **xs:date("1942-01-01")** to test if the date 1941-12-31 precedes the date 1942-01-01.
(d) List the name, district, and party of each current Representative of NC. Format each of them as an element of the form **<representative name="…" district="…" party="…"/>** and sort them according to the district.
(e) List the names of current Senators who at some point also served as Representatives. Format each of them as an element of the form **<sen>…</sen>.**
(f) Find the number of current Senators for each party. For each party, format the output as an element of the form **<record count="…" party="…"/>.**
(g) List the names of legislators who are serving in ALL subcommittees in the committee. Format each of them as an element for the form **<person>…</person>.**

Extra Credit Problem X1 (10 points)

As part of the trendy "NoSQL" movement, CompSci 316 decides to port the beer drinker database to XML. For reference, here is the relational schema:

Drinker(*name*, *address*)
Bar(*name*, *address*)
Frequents(*drinker*, *bar*, *times_a_week*)
Likes(*drinker*, *beer*)
Serves(*bar*, *beer*, *price*)

We would like to represent all this data in one XML file. There are multiple ways to structure the data in a hierarchical manner. We would like to do the following: The document should list all bars first, and then all drinkers. Under a bar element, we also list all beers served at the bar. Under a drinker element, we also list all the beers that the drinker likes, and then all the bars that the drinker frequents. You should capture as many

constraints (e.g., keys and foreign keys) as possible. Design an XML Schema for this document, and submit both the XML Schema (.xsd) file and an XML (.xml) file that represents the data in the sample relational database instance (found in /opt/dbcourse/examples/db-beers/data/). You should check your XML file against your XML Schema with command xmllint (see "XML Tips" on the course Web site for instructions).