CS386D
4/25/20
V1.0

Homework 10

Assigned: 4/24/20
Due: 5/1/20 on canvas, 11:59PM

Objectives: Another three part homework
Part 1: Completion of AWS basics
Part 2: Deeper per the use of EXPLAIN. Use 3-way joins. Still simple enough intuition and qualitative assessment apply, but optimizer really must make calculations.
Part 3: Starting Datalog. Real exercise with the languages syntax. (Semantics and execution, next week.)

Part 1: Configuring and using Amazon EC2 and RDS instances, as a client-server, [2 tiers of a 3 tier architecture]

You are, once again, being asked to run your loader from homework 5a and insert the data in a Postrgres database. This time, your loader should run on an AWS, EC2 instance instead of your own machine. The Postgres server should run, as it did in homework 8, as a managed database instance in RDS.

Detailed instructions are forthcoming and will be located in the supplement folder.

Load the ~1.25Gbytes of data into a database and time how long it took. Normalize it as a measure of bandwidth and complete the following table. The data for the first two rows comes from the previous homework. This homework concerns only the last row.

Client location	Server location	Load time	Bytes/second loaded
Local-machine	Local machine		
Local machine	AWS		
AWS	AWS		

Discussion:

In a post I expressed surprise at the slow rate of the data load from a client outside AWS to an RDS instance inside RDS. As my direct experience with this, (in a commercial setting, and by luck), was only with all the computational resources located in AWS, I suggested a homework to replicate this configuration would be coming. At the moment I wrote that, I was teasing. The pedagogical issue in the previous assignment was for you to learn, by practice, that large database transfers over a network can be very cumbersome, and a further assignment, (this one), would simply produce an obvious result. Later I realized that there are two additional issues that make this third part of the assignment valuable.

First, is, with the advent of the Internet, *the standard* architectural organization of most new software development is known as a three-tier architecture. In the original definition of such that architecture the three tiers were, a browser (for I/O with the user), a web application server (where most all the custom application code runs), and a database server (to implement a transactive, persistent data store). See Ramakrishnan and Gehrke. text starting on page 239. Upon completion of this iteration of the loader

homework you will have learned how to "spin-up", connect and use AWS server instances for two of those three tiers.

Second, despite cost, other advantages, and the meteoric growth of cloud providers, there are still many corporations that for regulatory and cultural reasons insist that portions of their IT infrastructure remain on-premise and behind their firewalls. The IT industries solution to that is called *hybrid-clouds.* In a hybrid cloud, IT resources located at a cloud providers data centers are connected to the IT resources that remain at the corporation's locations. A next major version of the supporting software was released by each of the major players in the months before the Covid pandemic hit. That latest version was heralded as the version that would enable wholesale transition to hybrid clouds. (FYI, Kubernetes is seen as the critical functionality). If, hybrid clouds have finally arrived, then its time to take the next *big thing* seriously, and that is *multiclouds*. The driving value proposition is that if hybrid-clouds with Kubernetes can dynamically configure corporate workloads across a hybrid cloud, then individual workload components could be run in different clouds, and the choice of cloud could be driven by special functionality offered by that cloud or factors of cost and availability. It is clear from trade articles, (and seriously highlighted in one that I just recently read), the execution and management of large-scale data transfers among clouds is one of the critical features to be addressed by upcoming multicloud software infrastructure. So, one last iteration of an exercise in data loading at scale has been assigned.

Part 2: Continue part 2 from homework 9

Per instructions from homework 9, develop the query plans for the following queries.

-- these three-way joins will have to be expressed as two, two-way joins ANDed together.
10. A.ht = B.ht = C.ht
11. A.ten = B.ten = C.ten
12. If 10 and 11 produce different plans, then try these queries using the other columns. See if you can identify a threshold where the plan changes. Report the pair of queries that bracket the transition, else simply report "10 and 11 produce the same plan"

13. A.pk = B.ht = C.ht -- something else to try
14. A.pk = B.hund = C.hund
15. Same as 12, as applied to 13 and 14

16. A'.ht = B'.ht = C'.ht -- rinse and repeat
17. repeat 16 for the ten column
18. Same as 12, as applied to 16 and 17.

In the first set of queries there was a test to determine if the optimizer chose the join order for 2 way joins. Just to be certain, check that for 3-way joins.

19. A.ht = B.ht = Cot
20. C.ot = B.ht = A.ht
21. A.pk = B'.ht = C'ot
22. C'.ot = B'.ht = A.pk

Part 3: Datalog Syntax

Identical to what was created for the 16 Introduction to Graph Database lecture, as lecture notes 17 Datalog1, there is an accompanying folder with the Zoom Recordings and a file titled links.

The link to Video Notes reaches a writable Google spreadsheet, where I hope you may contribute specific comments.

The link to lecture web site reaches a page akin to an online course as proper online courses are usually organized and presented. This includes review questions for individual sections of the lecture. *Do review probem 1 – 8.*