CSCI 585 midterm exam: 6/9/22
Duration: 90+30 minutes

Please read the following items carefully, before starting the test:

• the exam is **open** books/notes/devices - feel free to look up whatever you want, from wherever! But don't look around 'too much', you'll lose time!

• from Q1 through Q10 (worth 5 points each), you can **choose any 7**, for a total of 35; if you want you can do **8, 9, or even, all 10** - we will ADD up ALL your scores (even partial scores) from **all** the questions you answer, then **CAP it at 35** - sick, right?!

• there are no 'trick' questions, or ones with long calculations or formulae or needing lengthy explanations, and there's certainly nothing to memorize [remember - it's all OPEN :)] BUT - this doesn't at all mean that the questions are trivial!

• please do answer carefully: in other words, **answer only WHAT IS ASKED**, otherwise you won't get points (eg. if a question is -ABOUT- 2PL, don't DESCRIBE/DEFINE 2PL!) - it's the quality (of your answer) that counts, not quantity (verbosity)... You can't demand points later, for providing non answers

• **please do **NOT** cheat**- this means NOT communicating with anyone via any device/medium/channel - you will get a 0, and be reported to SJACS, if you are found to have cheated; ANY attempt to get help from others in any form is a VIOLATION, as per , sections 11.11
https://policy.usc.edu/scampus-part-b/
through 11.14 [read it, if you are not familiar with it]

• when the time is up (90 minutes), stop your work, then spend the rest of time (30 minutes) on submission [students with DSP accommodations - your exam duration will be as per DSP determination] - **submitting pastthe deadline comes with a penalty**, because it is not fair to others if you go over when they don't

• note that you need to **submit each answer separately** (not all of them as a single PDF) - it's a Crowdmark req., it enables questions to be graded in parallel

Good luck! Hope you enjoy answering the questions, hope you find them to be easy, fun, thought-provoking.

Q1 [1*5 = 5 points].

Consider a typical web page, to be eqvt to a 'table row'. Such a table will have hundreds of billions of rows, given there are hundreds of billions of pages [this is just an observation - it's not a bad thing].

a. What would be the natural PK for such a table?

A. The URL.

b. What would be non-PK?

A. Page contents.

c. What would be FK?

A. Links!

d. How could referential integrity violation happen?

A. When pages' links point to a non-existent page (leading to '404 Not Found'!).

e. What might be a better PK?

A. A hashed (eg. using bit.ly) version of the URL, or a bookmark with manually shortened titles, etc.

Q2 [2+2+1 = 5 points].

a. Explain 'self join', using an example that we did not discuss in class.

A. Countries and neighbors (or states and neighbors, or counties and neighbors...), friends, book references...

b. What would be an example of a table that would have MULTIPLE (ie more than one) types of self-join? Note - this is NOT asking about recursive self joins!

A. People (employees) in a company - there can be FK columns for 'manager', 'spouse', 'neighbor', 'work friend', 'project partner', 'workout partner', etc.

c. Recursive self join is equivalent to what data structure?

A. Tree.

Q3 [1*5 = 5 points].

Let's play a game called,"I load data into a site (ie. into a DB connected to a site), you query it".

Eg. I (job seekers) load resumes into LinkedIn, you (employers) query it; I (stock market) load stock prices into stock sites, you (investors) query it.

Provide FIVE more such examples - look at your apps, sites you frequent, USC (!), etc. for tips/hints :) Express each in the same 'I..., you...' format like the above two samples.

A:

Ralph's product manager loads grocery item data into their DB, shoppers query them

Lyft/Uber riders load data into Lyft/Uber, drivers query it

eBay auctionable items get loaded onto eBay.com, bidders query them

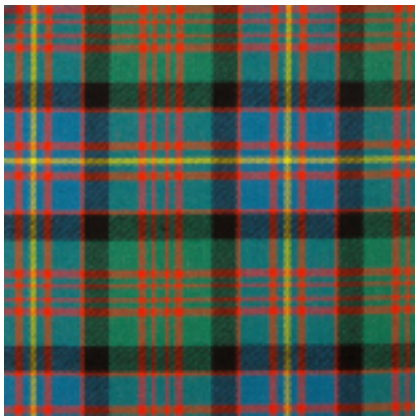
USC Registrar and depts load course catalog into USC SOC, students query them

Twitter users load data (tweets) onto twitter.com, APIs/other users query them

... MANY more such examples are allowable answers

Q4 [1+1+1+1+1 = 5 points].

Below is a Scottish tartan pattern (called 'Cameron of Erracht', if you're curious!):



a. How does it 'relate' to data?

A. The pattern resembles a table, where each row/column intersection (cell) is a 'relation'.

b. You would get the same overall tartan pattern (roughly speaking) whether you weave the horizontal ("weft") threads over the vertical ("warp") threads, or vice versa. How does this equivalence (that the order doesn't matter) compare, when we restrict rows and columns in a table?In other words, what happens if we swap the order?
A. Whether we PROJECT some columns first, then SELECT from

them, or SELECT rows first and then PROJECT the columns we want, we would get the same result.

c. When order doesn't matter in a binary operation, what is that property called?

A. Commutative Property (or 'Commutation').

d. What is an example type where order does matter? A. Matrix

(for mult), vector (for cross product), number division...

e. Even there are two different restriction operators in relational algebra, there is only one in SQL, that does both. Why?

A. Because one is enough! The 'SELECT' keyword accepts column names (to PROJECT), and the WHERE condition (to SELECT).

Q5 [2+2+1 = 5 points].

a. What is a real-life ('RL') example of 2PL? Explain briefly (ie don't just name it).

A. For a project where resources are shared by multiple people (eg scissors, printers, flash drives, etc), oftentimes good wisdom dictates that we first gather all of what we want, THEN start the project, and proceed without getting stopped mid-way.

b. What is a(n) RL example of 2PC?

A. A 'coordinator' - eg. a purchasing coordinator, or a mob boss (!), or a dept. manager, etc.

c. In 2PC, the coordinator simultaneously (ie in parallel) polls the worker nodes for yes/no (ready to commit, or not). In a situation where node transaction failures are common, what might be the

advantage of polling them one by one, instead?

A. Once any node in the list replies with a 'no', we don't need to poll the rest - we can skip to Phase 2, and broadcast an 'ABORT' message to all the nodes.

Q6 [2+2+1= 5 points].

What is a design activity that is similar in principle, to table normalization? What factor do we consider, in doing this (ie. what is the basis)?

A. Class design (eg. in C++, Java, etc.). We use 'separation of concerns, 'or tight cohesion', as the guide - related members and methods would be housed in a single class.

How about EER - what other design activity resembles this? Again, what is the basis?

A. Class hierarchy (ie superclass/subclass) design. The basis is this: common members and methods would be moved 'up' to a common superclass, while each subclass will retain items (members, methods) that are specific to it.

Too much 'clean design' might not always be a good thing, with tables. How so?

Because it would lead to table fragmentations, hence, slow query execution that involves joining the numerous tables.

Q7 [1+1+3*1 = 5 points].

a. In our course context, what is 'connectivity'?

A. Connectivity is about connecting a 'backend' (DB or a data

generation 'service' in general) to a web server.
b. Why are there, SO MANY 'connectivity' technologies?

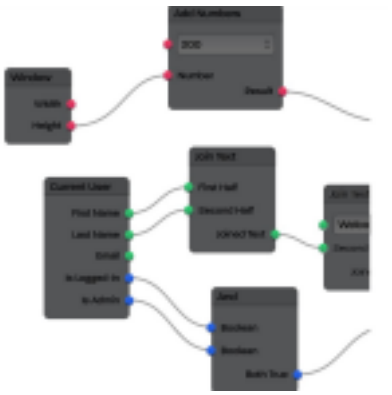
A. Because data is CENTRAL to so many operations that are carried out by lay people, businesses, companies, gov't organizations, educational institutions... Connecting data services to a web server, and from there to clients, is the most effective way to provide access to data.

c. Briefly explain THREE connectivity solutions, in a sentence or two for each: be sure to pick an 'older' one, a later (middle years, eg. late 90s) one, and a current one.

A. OBDC or JDBC..., CFML or cg-bin etc, microservices/Flask/Node... etc.

Q8 [1+1+1+2 = 5 points].

a. What is the following (called)?



A. Dataflow!

b. What is its alternative?

A. A script, where the ops are traditional function calls. c.

Why is this (what's shown) preferable to the alternative?

A. Because the graph can be selectively updated, saving vast amounts of computation (by not recomputing what can be reused).

d. EXPLAIN how it works (ie. what makes this better), using a small diagrammed example of your own.

A. A graph 'manager' tracks what nodes have completed running, what needs to run next, and runs them accordingly. That way, a change made in one node would lead the manager to mark affected/downstream nodes as 'dirty', and run just those (recursively). Your own diagram can be related to simple math operations, or image processing, or traffic analysis (ie TensorFlow), or a traditional data pipeline, etc, etc.

Q9 [1*5 = 5 points].

Compare SQL with C++, in terms of the following five programming language aspects, using a sentence or two for each: datatypes, operators, variables, function, class.

A.

Datatypes: C++ has int, short, byte, char, double, float, string... SQL has number, date, boolean, string (varchar2)

Operators: C++ has the usual ones (+ etc) plus &&, += etc; SQL has just + - * / ^ for operators.

Vars: C++ has them based on types (eg int i;); in SQL, columns can be considered vars, as can tables (because they can be named views)

Function: Both languages allow user-created ones; both come with built-in ones too (via 'libraries')

Class: in C++, we have the 'class' keyword; in SQL, the CREATE TABLE command serves as 'class definition'

Q10 [1+2*2 = 5 points].

a. What is Docker, **in your own words** (not a Googled result)?

A. Docker is a VM-lite of sorts - a few processes (a process tree) that serve as a contained OS ('containers'), to run applications ('images')

b. What are two advantages of using Docker (ie. why is it popular)? Again, explain each advantage in a sentence or two, in your own words.

A. Applications can be made to be self-contained and deployed as images, which are guaranteed to run on any host platform - there is no dependency on host services (eg the host does not need to have requisite libraries installed). Also, container instances can be trivially scaled, ie. multiple ones can be run in parallel.