CSCI-585 Summer 2023
Midterm 1 Rubrics

Q1.

- ➔ An entity declaration for storing data, is loosely like a class definition, in a coding language - in what sense?
- ➔ But the definition isn't quite analogous - what is missing?
- ➔ What would be the advantage in making entities, entirely analogous to classes?

Ans.

- Entities have data fields and classes also have data fields (1 pt)
- Classes have methods/operations associated with them. However, Data Entities do not. (1 pt)
- The advantage would be that we'd be able to define behaviors/operations on the stored data as well. This means we could call methods directly on data, eg. price.placeOnSale() (1 pt)

Q2.

- ➔ SQL resembles a traditional coding language such as C/C++, JS, Python. How?
- ➔ SQL is not a 'full-blown' language though. Why not?
- ➔ Given that so many types of modern data (eg. your Spotify playlist) isn't stored in tables, why does SQL continue to be relevant?

Ans.

- Syntactic and syntaxes define instructions (1 pt) There are commands, built-in functions, operators, expressions, etc.
- Does not support logical branching (1 pt) Also, no variables, looping, class definition, etc.
- The syntax is known by lots of developers, and is highly expressive/capable [comparable queries become more complex to express, in traditional coding languages] (1 pt)

Q3.

- ➔ In pseudocode form, how would you express SELECT ... FROM ... WHERE..., when it comes to analyzing data in a table?
- ➔ How would you express via pseudocode, the Cartesian product operation on two tables?
- ➔ What would be pseudocode for a classic JOIN operation that combines data from two tables?

Ans. We can accept any pseudocode, however loose, as long as it captures the logic behind the operations.

- Here is an example:

```
function selectFromWhere(table, columns, condition):
    result = empty table
    for each row in table:
        if condition(row):
            selectedRow = createRowWithSelectedColumns(row, columns)
            result.addRow(selectedRow)
    return result
```

(Grading: 0.5 given to the table traverse part, 0.5 given to the correctness of the rest)

- **Here is an example:**

```
function cartesianProduct(table1, table2):
    cartesianResult = empty table
    for each row1 in table1:
        for each row2 in table2:
            cartesianRow = concatenate(row1, row2)
            cartesianResult.addRow(cartesianRow)
    return cartesianResult
```

(Grading: 0.5 given to the row combination part, 0.5 given to the correctness of the rest)

- Here is an example:

```
function joinTables(table1, table2, joinColumn):
    result = empty table
    for each row1 in table1:
        for each row2 in table2:
            if row1[joinColumn] == row2[joinColumn]:
                joinedRow = concatenate(row1, row2)
                result.addRow(joinedRow)
    return result
```

(Grading: 0.5 given to the data combination part, 0.5 given to the correctness of the rest)

Q4.

➔ What do we gain, by carrying out normalization? The answer is NOT about describing 1NF, 2NF etc!!

➔ Normalization involves steps: 0NF -> 1NF, etc. What is the importance of (need for) these explicit steps (ie. why not skip them)?

➔ How does data normalization relate to classic software development [what similarities exist]?

Ans.

- **Any two (0.5 point each) of the following:**
  - Reduce redundancy
  - Reduce anomalies
  - Improve data consistency across the dataset
  - Improve storage efficiency
  - Improve query efficiency
  - Improve flexibility and scalability
- The systematic steps ensure that we don't accidentally overlook a partial or a transitive dependency.
- **Any one (1.0 point) of the following (it's about 'separation of concerns', ie abstraction/modularization, and loose coupling+tight binding; in other words, it's about minimizing duplication/redundancy):**
  - Both are systematic approaches
  - Both utilize modular design
  - Both utilize abstraction and encapsulation
  - Both require maintainability and extensibility
  - Both require data integrity and consistency
  - Both for performance optimization
  - Or any other explanation that make sense for both

Q5.

➔ What exactly is the problem, when we do distributed data processing with a coordinator?

➔ What alternate mechanism [other than the use of a coordinator] would you propose, for fixing the problem?

➔ What might be the problem with your alternative proposal?

Ans.

- **Any one (1.0 point) of the following:**
  - Single point of failure
  - Communication overhead
  - Coordination bottleneck
- The answer depends on the answer to the previous question. A sample answer: decentralized coordination or distributed consensus protocols. **(1.0 point for any correct solution) [eg. each node would wait to hear directly from all other nodes - this leads to excessive communication]**
- The answer depends on the answer to the previous question. A sample answer: complexity, excessive transmission/bandwidth utilization. **(1.0 point for any correct problem pointed out)**

Q6.

Windows (OS) is pretty widely deployed in the world.
Imagine a small company that has been MS Windows since the mid-1980s. It continues to use 80s era DBs such as Access and Paradox, to store valuable company data.

- ➔ What is the danger of doing so [continuing to use Access etc]?
- ➔ How would you help the company 'rescue' the data that resides in Access etc [so that Access and friends can be retired from continuing use by the company]? Be specific - describe your approach.

Ans.

- The answer doesn't need to be exactly the same, but should explain the principles in a meaningful and correct way. **(1.0 point)**: Here is a list of correct answers:
  - Security Vulnerabilities
  - Limited Scalability and Performance
  - Data Incompatibility and Integration Challenges
  - Etc.
  - The product might become discontinued (no more bug fixes, features, support); people who know how to use it would be hard to find
- The answer doesn't need to be exactly the same, but should give a correct answer based on the challenge the student pointed out above. **(1.0 point) The solution is to use ADO.NET to retrieve entire tables (with SELECT * FROM <table>), and save them as XML files on disk (text-based, easy to parse and transform to JSON or csv, ingest into modern DBs).**

Q7.

- ➔ What is data?

Ans.

- **Data refers to CHARACTERISTICS we select/define/specify, for an entity.** Answers such as 'raw facts', 'information' etc are useless (they don't explain anything) and are not acceptable.

Q8.

- ➔ What is data modeling?

Ans.

- From the slides: Iterative and progressive process of creating a specific data model for a determined problem domain (e.g., an application)
- **It is the process of selecting the appropriate STRUCTURE for the given data (eg. hierarchy/tree, network/graph, tables etc).**

Q9.

➔ What three core principles did you learn, related to database modeling/design, ie. when it comes to storing and analyzing data? Explain each in a few lines.

Ans.

- The answer doesn't need to be exactly the same, but should explain the principles in a meaningful and correct way. **(1.0 point each, 0.5 point for the principle, 0.5 for correct explanation)** Here is a list of potential answers:
  - Data Integrity
  - Normalization
  - Scalability
  - Consistency
  - Avoiding needless redundancy
  - Correct Abstraction/model (eg E-R)
  - Each "row" (data object, ie collection of columns and values) needs to be made unique, eg via a primary key or rowkey
  - Etc.

Q10.

In the course of DB design, we moved away from file systems, preferring to store data in tables, etc.

➔ But, if you were to make a case to use a text-based file system for a modern app you create (where you would store data as plaintext, ie ASCII), what three distinct advantages would you list? Explain each, in a line or two.

Ans.

- The answer doesn't need to be exactly the same, but should explain the advantages in a meaningful and correct way. **(1.0 point each, 0.5 point for the advantage, 0.5 for correct explanation)** Here is a list of potential answers:
  - Easier Version Control
  - Customized query formats
  - Easy Reusability/Portability
  - **Might be faster/easier to parse**
  - **Can store it on disk compactly, eg using a custom compression scheme**
  - **Easily editable to make small changes**

Q11.

➔ What mechanism exists in SQL, to create your own functions (commands)?

➔ Often we analyze two columns of data together, eg. 'weight' and 'blood pressure' of patients. What two functions would you consider creating, to do such analysis? Name them, and explain what they would do, and return.

Ans.

- User Defined Functions **(1.0 point)**
- **correlation(), regression(); correlation() would measure, on a -1 to 1 scale, the correlation between the columns, while regression() would fit an equation (eg a line, with a specific slope and intercept)**
  [the specific correct answer is above - vague descriptions are NOT acceptable!]

Q12.

Pick two apps you use on your smartphone - for each, describe in pseudocode-SQL, what data-oriented queries you generate as you go about using your app. For each, describe:

- ➔ what action you are carrying out (ie what you are 'making the app do')
- ➔ what SQL might result from your action [this does not need to be syntactically-valid, or complete, SQL code]

Ans.

- The answer depends on what students choose. The keywords don't need to be the same as SQL, but the queries should be correct **(0.5 point)** and unambiguous **(0.5 point)**. **In other words, the actions (eg. looking for someone on LinkedIn), and the corresponding pseudo-SQL, need to make sense (plausible, correct).**

Q13.

- ➔ Summarize in a few lines, the contents of the 'Data modeling' and 'E-E-R diagrams' lectures.

Ans.

- **Data modeling**: Data modeling is an iterative and progressive process of creating a specific data model, which is a simple representation of complex real-world data structures, for a determined problem domain. (Chapter: Data modeling) The answer should explain **abstraction of real world facts** in a meaningful way. **(1.0 point)**
- **EER**: The answer should explain the difference between EER and the original ER diagram in a meaningful way, especially about **specialization and constraints (NEED to mention disjoint vs overlapping constraints, and partial vs total completeness)**. **(1.0 point) [**

Q14.

- ➔ The homeworks in this course are specifically designed to provide hands-on practice with handling data. With that in mind, summarize the intent ("point") of HW1, HW2.

Ans.

- HW1 is about designing an ER diagram, which is the designing phase of a database. The answer should point out "**designing (relational) database**" or the equivalents. **(1.0 point) [it is about succinctly capturing the structure of a database in a single**

diagram that is comprised of entities/tables [collections of related columns] and the relationships (1:1, 1:M, M:N) between them].

- HW2 is about using queries to pull data from a database. The answer should point out **“using database/ query data from database”** or the equivalents. **(1.0 point) [it is about using SQL to query the data held in tables].**

Q15.

➔ The IMO specifies that ships transmit AIS data that contain several pieces ("columns"), eg. ship's identity. What two other pieces of data would you add to the existing list of data requirements?

Ans.

- Departure terminal
- SOS data
- Ship’s dimensions
- Carrying capacity [cargo, humans]
- …