HW1.md

Homework 1 - Databases and Conceptual Data Models

1. What are the 4 different types of data models?

1. High-level (conceptual) - *ER, UML, OO*
    - This is the end user's conception of their data in a human-understandable way.
    - This is a tool for understanding the data enough to derive an implementation model from it
2. Low-level (physical) - *XML, JSON*
    - This is how data is actually stored on the disk (the code inside of a Database Management System, or **DBMS**)
3. Implementation (representational) combines conceptual and physical data models - *Relational*
    - Understood by database developers
4. NoSQL - *column, key-value, document stores*
    - This is the 'newest' data model
    - Very popular right now due to explosion of big data

2. Considering the 3 categories of metadata, give specifics examples of each for an electronic medical records system. Assume the actual diagnosis, test results and medical images are secured in the database, but other aspects, such as appointments, if an insurance claim was filed, etc. are considered metadata. Be creative – there are very few wrong answers to this.

1. Structural Metadata: (*A way to define how objects are put together - the structure of things*)
    - One example of Structural Metadata here is the collection of tables of people in the system. There may be an "Doctors" table, which contains dozens of columns such as name, DOB, gender, department, specialization, medical doctorate degree, and more. Additionally, there may be a "Patients" table, which contains slightly less information than the "Doctors" table, but still contains things like name, DOB, insurance provider, medical history, etc. Another example of Structural Metadata would be the relationships between Doctors and Patients. For example, there may be a "doctorId" column in the "Patients" table which matches with a "doctorId" in the "Doctrs" table, thereby creating a relatinship between Patients and Doctors. Structural Metadata defines the literal structure for these tables and relationships.
2. Administrative Metadata: (*Relates to the technical source of objects*)
    - In our example, some Administrative Metadata includes the date at which the database was created as well as the date at which each patient and doctor was entered into the database. The collection of database permissions (who is the owner, who can edit the database, who can simply view the database, etc.) is another example of Administrative Metadata.
3. Descriptive Metadata: (*Everything else - information describing the objects*)
    - In our example, some Descriptive Metadata would include the maximum and minimum character limits for columns such as "name" in the "Doctors" and "Patients" table, as well as possible constraints on whether or not that column needs to be specified at all when entering a new entry into the database, or if it can be left blank, or *NULL*.

3. Why is program-data independence needed?

Program-data independence means that the information describing the structure of the database is loosely coupled with the actual data that the database is holding. This means that the data will remain intact and accessible regardless of changes to the database structure. This is important, because we may (and certainly will) need to change the structure of a database even after it has been put into production. This means that at the time of our modifications to the database, there will already be existing data in the database. Without program-data independence, this data could be at risk when we make our modifications to the database structure. However, when we **do** have program-data independence, we can trust that we've built our system in such a way that we can say "I'm going to go and change up the database, and I can trust that the data will be fine, because the data is **independent** of the database structure. That way, once I'm done with my changes, everything will work just fine, and we won't lose any precious data!"

4. List 2 advantages to Database Management Systems and 1 disadvantage

```
- Advantages:
    1. The engineers tasked with creating software do not need to "reinvent the wheel" in creating a database for
their application(s).
    2. The task of decoupling the database with the other parts of the application is very easily accomplished.
DBMS's are inherently capable of only database tasks, and so it is easy to make sure that the job of maintaining the
database is isolated from the rest of the application(s)
- Disadvantages:
    1. When creating a database using a DBMS, the database administrators must work with the rules of that DBMS
specifically. If engineers need only a small subset of the functionality from that DBMS, or they need some
functionality not natively supported by the DBMS, than the database administrators are out of luck. They must follow
the rules of the DBMS.
```

5. Using the ERDPlus tool, draw an EER diagram given the following specifications for a Ship Database:

```
- A Ship has a unique name, owner, year built and age (computed).
- A Captain has a unique ID, name, birthdate, home country and date hired.  A captain serves as the captain of one
ship but may not currently have a ship, and a ship must have one and only one captain.
- A Ship_type has a unique type name, and also a tonnage and a hull.  A ship is assigned one Ship_type. Many ships
can be of the same Ship_type but there may be no ships of that type.
- A Mechanic has a unique ID, name, birthdate and certification number.  A Mechanic works on one or more ships and a
Ship has one or more mechanics working on it on a specified Date.
- A State/Country has a unique name and a continent.
- A Body_of_Water has a unique name, maximum_depth and multiple continents.
- A HomePort has a name and a capacity.  The HomePort name is only unique within a particular state/country but not
across all state/countries. A HomePort is located in one state/country and on one Body_of_water.  Of course, a
state/country and Body_of_water can have one or more HomePorts.  A HomePort can house many ships, but a ship has
only one HomePorts.
- The Ship_Movement has a timestamp, longitude and latitude. The timestamp is composed of a date and time. The
timestamp is unique for a particular ship, not across all ships. A ship can have zero to many Ship_movements, but a
Ship_movement is associated with one and only one ship.
```

6. The EER and UML model are not available in ERD Plus so you can use any drawing tool you would like to redraw the Owner and Mechanic entities as a specialization of a “Person” entity for the following. Include the Ship entity and any relationships with the new subclasses.

- EER model
- UML model