Nicholas Comito
Database Management Lab 1
Professor Labouseur

1)

This past summer, I interned at a financial advisory office where the database used to keep record of both clients and advisors was called ClientConnect. The purpose of this database was to organize data so advisors could pull information regarding a client's personal details and financial records. The data stored to the database included name, address, phone number, primary advisor, account number, and total assets. This data was converted into information for advising purposes. For example, any client with "6MX" to begin their account was classified under managed accounts and was assigned one of two primary advisors. Any client with "HMM" to begin their account was deemed a brokerage account. The database was also used to allocate the assets each client had. The assets included cash, mutual funds, bonds, and individual stocks. In essence, ClientConnect was able to generate a portfolio for the advisor to analyze.

Without context added to data, there is no way for anyone to understand what the data represents. For example, a professor tells you an assignment is due at 12:00. Is that 12:00 am or 12:00 p.m.? Is it the Eastern Time zone or pacific? There are many uncertainties involving a simple example but without context those uncertainties cannot be answered. While interning and filling out a client's asset allocation, I was required to indicate whether I was recording number of shares or currency. If it was currency, I need to indicate the type since there were some clients who lived in London. 400 shares of Apple stock is far from the same thing as USD $400 of Apple Stock.

Data plus context equals information and information equals the power to make decisions. Decisions cannot be made unless an individual is able to analyze the situation. Refereeing back to ClientConnect, no investment decision could be made unless the advisor knew the amounts and types of assets the client was working with or if the client was a brokerage client or a managed client due to restrictions regarding the type of accounts. Without information it is impossible to make any rational or educated decision

2)

The hierarchal data model is a tree like design that uses only the one-to-many relationship for its data. The model needs to start with a root node with fields stemming off of it. Data in this model is stored in records with each child record only having one parent. This model was developed in early stages of database design and was used in the information management systems by IBM. The Hierarchal is superior than the Flat Files System developed before it in the fact that it is physically independent and the programs can be used on any system that supports it. However this early model has limitations one of which is that it does not allow loops to be present in the program. The network model is similar to the hierarchal model but allows for loops to be present in the program and child records can have multiple parent records. The model is very flexible since it allows owner (parent) files to be linked to multiple member (child) files. This flexibility gives the network model a many to many relationship regarding its data.

Both models however have limitations to the superior relational data models created by Edgar Codd. In both the hierarchal and network models, the structure of the program needs to be known before any code was written. According to the textbook, these early models could not support high-level query languages as well. Queries in the

relational model are able to handle high-level languages, which leads to higher efficiency for programmers. Relational databases can always be adjusted without re-structuring the model, where as hierarchal and network data models were difficult to expand upon. The relational model also solves duplication issues that can arise in the both the netweork and hierarchal model.

Based on the functions of both XML storage and relational databases, I personally would feel more comfortable using the relational database to store my data. XML files are adequate for small file sizes, but if working with big data, the file may not be able to hold as much data as needed forcing a reduction in file size. According to "Comparing XML and relational storage: A best practices guide" by IBM in 2005, relational databases would be used over XML models "when the highest possible performance is required". However IBM also suggests using XML when schema is volatile and the data is hierarchal in nature. The relationship of data in an XML file is in a hierarchy. Given this information, I would consider using XML files for holding small data involving, for example, the employees within a business, but would lean towards relational database models for larger and more complex data.