Carlie Maxwell
Lab #1
1/23/16

A database that is used everywhere are library databases. These databases store both data and information about the books that the specific library contains. The elements of data that are stored are both simply numbers and names. The database takes these meaningless pieces of information and turns them into something useful by providing surrounding context. In order for people to easily navigate the data in the database, it has to first label the numbers as the ID number of the book. This helps the user to realize that they are numbers with meaning and that purpose is to correlate with specific books contained in that library. Additionally, the numbers data element is contextualized into information by writing that they are the number of copies that the library has of that book. Lastly, the numbers listed can represent the checkout and due date of that book. The useless numbers listed in the database can be changed into information quite easily by providing simple labels such as ID number, copies in the library, and checkout/return date.

The meaningless names that are just written down in the database can be changed to have meaning by showing that they are the authors of the book that you're searching for. In order to create data into information you have to remove all ambiguity by being precise.

Without the context given, such as "Book ID number," the user would have no idea what the number listed meant. Without this context and organization, the data is simply meaningless. Once the information is provided, the user can then interpret it as an ID number and be able to search for the book at any time in the database or ask the librarian where it may be located in the library by providing the number.

The hierarchical model looks at the data as if it were arranged in a hierarchy. This means that it is sorted the same as for example, what looks like a family tree. At the top there is a root node at level 0, and off of that, below it, are branches on what is considered to be the next level, 1. The branches occur when that element goes “inside” the one above it. For example, a root node would be the World of a Game, under it would be the players, because the players are in the game, and under that would be the items that are associated to that player, because each player has an item. The problem with this model is that it only represents what is currently happening; it has to currently apply to the situation in order to be connected. Because of this, the model doesn’t represent elements that can be of use in the future. Additionally, it repeats data, which is inefficient. Each player could have an item B, causing you to write that twice, which you don’t want because it is a waste of space.

The network model is similar to the hierarchical model, in the sense that they are similarly arranged in the tree format. The difference between the two is that the network model connects branches together, allowing a parent node to share a child node, rather than writing it twice. This allows you to not be redundant in writing the same element twice in order to save space. The problem with this model is that it is difficult for the user to use, along with the hierarchical model. The user’s need to understand how the data is stored in order to get information out of the database, which not everyone knows how to do. This caused these models to fall flat.

The relational model has 0 repetition because each element are written in separate tables, such as all the “items” in the Item table, so you only list them once and then pull them as you need them. All elements are listed in a table even if they are not in use, which solves the problem of leaving information out. Additionally, it creates many-to-many and one-to-many relationships. The relational database it a lot simpler for the user to navigate because they don’t have to have an understanding of

how the graphs are coded, rather they only have to know the table name of what you're looking for.

Having an XML model for heavy data storage would be a bad idea, as it is essentially the opposite of the highly successful relational database. XML is a document-oriented database, which has no structure and is considered to be "schemaless." Although these types of databases are really flexible, this can work as a disadvantage. When the database has little to no structure, it makes it very difficult for the user to get data out of it, which ultimately decreases the value of the database. This creates the database to consist of entirely dangerous data.