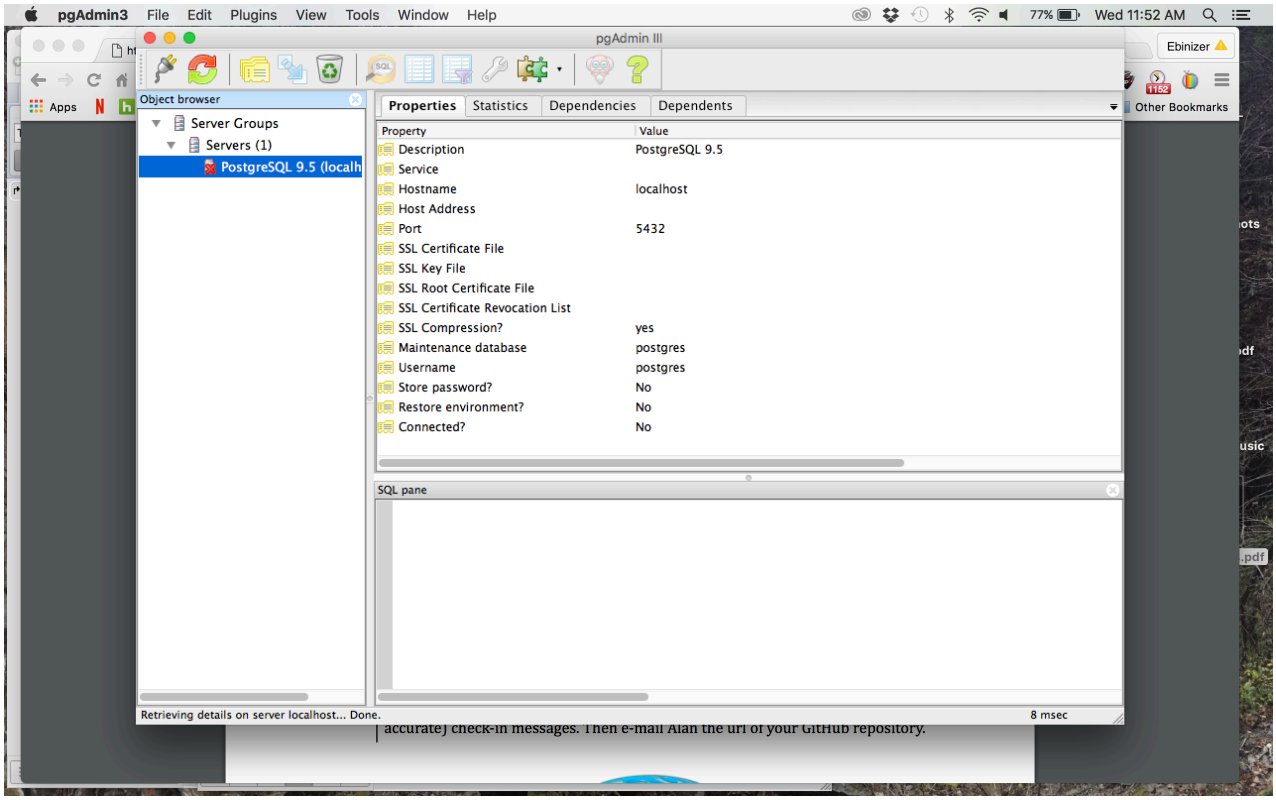

2: A database is more than just storage of data, its real job is removing the ambiguity from the data by giving it structure, which turns the data into information. I see a database like a microwave. Before I put a cup of Velveeta macaroni and cheese into the microwave I can't eat it, so it holds no value to me. The ingredients haven't been infused with one another and so they don't look very pleasing at all. But after the cup has been in the microwave everything makes sense. The food smells, looks, and tastes great and in turn I'm very happy. The database that Google Maps uses does the same thing as the microwave. Google Maps pulls information from a "database of places" when a user wants to know the location of something or puts in an address. The data that goes into the database would be things like street address, state, city, country, longitude and latitude, type of entity, entity name, etc. The output is information because the database will organize the data and make relations between fields, which removes the ambiguity. If a user types in "restaurant" than Google Maps will return the locations, distances, and names of multiple restaurants in the vicinity. The database will use street addresses, states, and cities to produce locations, longitude and latitude coordinates will be used to come up with distances, and entity type and name will print the names of the restaurants. There are certainly more fields that are used in the database, but without a database Google Maps couldn't function because there would be no context for this data.

3: The hierarchical database model was thought of after people gained some common sense and realized the file system was inefficient. The hierarchical model or Information Management System is made up of records that are

connected to one another. A record is a collection of fields, each of which hold a data value. This model took a big step in the right direction because it was the first time that the program was separated from the files of the program. But the biggest problem with this model is that the same data value can show up more than once. This is because of the one-to-many relationships that this model uses, which means that a child can only have one parent. This model works for file management systems on a local computer because there is not that much data and a child really only needs one parent. But when there is a lot of data being stored, than the replication of fields becomes too inefficient and wasteful of space. This redundancy creates data inconsistency, which is the worst problem that a database could have. The hierarchical model was seen as being to constricting so it eventually got morphed into the network model.

The network model might look similar to the hierarchical model at first glance but it doesn't just support one-to-many relationships, it has the capability to handle many-to-many relationships (as well as one-to-one), so a child can now have multiple parents and there no need for data replication. This not only makes this model more flexible, it allows for closed loops to be used, and it makes the retrieval of records much easier. This model fixed the problems that the hierarchical model had, but it created another problem for itself – complexity. Changing or accessing a record becomes much more difficult because the whole structure of database could change. The structure of the system has to be well known to able to work with this data, which is why this model is not very user-friendly. There are clearly a number of reasons why these two models had a much shorter expiration date than the relational model, but the main reason, I believe, is because they lacked scalability.

Of the little I know about XML it does seem to lack scalability as well. It can work on a very small scale because it is portable and the code describes the structure, but it doesn't have many of the features databases have today: security, data integrity, performance requirements, and storage among other things. The portability of XML makes it attractive, but at the moment it can't compete with real databases.