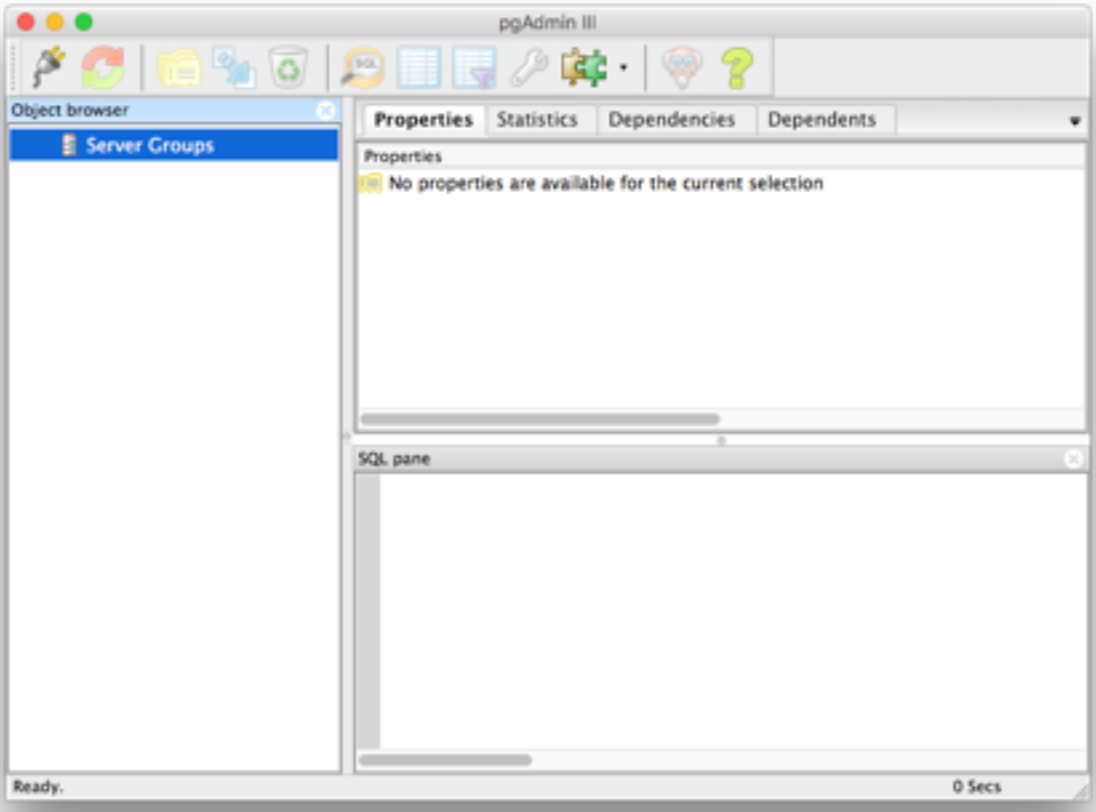
1. Download PostgreSQL from http://www.postgresql.org and install it on your computer.



Mutiat Alagbala

2. Short essay: Data vs. Information - Select a database in use today (real or imagined) and identify the elements of “data” stored therein and describe how the database organizes the “data” into “information”. Give contrasting examples of “data” and “information” that illustrate the meaninglessness of “data” without context and organization. Talk about the value the “information” provides once the component data is given context.

Consider and employee contact database for a large Business that manages several smaller businesses. Assume this database follows a relational data model. The data stored in such a database would be full of: names, emails, phone numbers (of many types i.e. cell, home office, etc.), office addresses, company branches or departments, position, genders, people who work under other people, branch or department descriptions, company site addresses, sub-branches and sub-departments, super-branches and super-departments and branch and department names. Individually any of these pieces of data are meaningless and unhelpful. For example, the purpose of a contact database would be to provide contact information for business employees so intracompany communication is more seamless. Therefore, a name without a number, address or email is very unhelpful in a contact database. Likewise a number that is not attached to the identity of some employee is similarly valueless.

The database turns this data into user contacts, company branches and departments and company sites all of which are connected. Users can sort data by any of these elements and the information they get when they access the database is collections of data organized such that humans read as contacts, business departments and branches and company sites. Each of which is an entity with specific attributes. Contact entities specify a persons name, email, phone numbers (i.e., cell, home office etc.), office address, which company branch or department they belong to, their position, gender, and who works under them. Business departments’ and branches’ entities will provide a name, company site addresses, branch or department descriptions, sub-branches and sub-departments, as well as super-branches and super-departments. Company site entities describe the branch to which they are connected, the site’s name and the address of that site.

Observing the span of the attributes of the entities it is obvious how a jumbled collection of names, addresses phone numbers, etc., would have no use compared to an organized database such as the one above. For example, if a CFO is looking for the manager of a certain branch of the company because that has generated lots of revenue and that person is deserving of a raise, a database which simply provides an organized list of names will not give him the information he seeks. More likely, he will need to go through contacts connected to said branch and find the contact with position “Senior Director.” Once he has found the information on the

senior director he can then obtain a name, phone number and email through which to instantiate his raise and notify him of the companies generosity. The name, phone number, position, and even the email are useless without the context they provide each other.

The data therefore gains its usefulness through the context that is provided by the database. This makes the database itself essential to the overall ebb and flow of the business. Without the organization of the data, what users would have instead of meaningful information is an alphabet soup of Strings, numbers, and symbols that are purposeless.

Mutiat Alagbala

3. Short Essay: Data Models - Briefly describe the hierarchical and network pre-relational data models. Explain their shortcomings in relation to the relational model. Considering this, what do you think of XML as a model for data storage?

The hierarchical data model is a database design where in each piece of data was nested under and connected to another piece of data with everything stemming from the database (root). In this model every datum can only have one datum under which it is nested (one parent). Whether that is the database itself or an item nested x amount of layers below the root, the rule still applies. It is also true that any datum, item or root can have multiple data or items nested under it (multiple children). So, if two pieces of data have identical children (i.e. two player having the same item in a game), redundancy reared its ugly head because a nested piece of data could not have more than one parent and therefore that data had to be repeated. Additionally any options not being referenced by a previous layer is either not represented or the previous layer is compromised with a false piece of data acting as a place holder. In the network model this design was improved by a small change that made a big difference. With a similar web-like design the network model could only have one root. However, the design allowed each piece of data to have multiple parents and multiple children.

Though these data models were great advancements from the days of file systems in their day, they still had a lot of issues. For example, most of the high-level query languages were not supported (Garcia-Molina, Ullman, and Widom 3). This meant that one could navigate from parent to parent through one piece of data. As simple as it sounds the process of actually navigating from datum to datum took large amounts of complicated code due to the models incompatibility with the higher level query languages. Additionally, neither of these data models solved the issue of children datum that do not have parents but need to be present in the database. Not only does this cause inconsistencies but it makes turning data into information more complicated then should be necessary.

XML seems like a bad idea as data storage because it has a hierarchical or structure therefore it is prone to inconsistencies, redundancies and overall inefficiency. XML as data storage is probably good for quick write-ups of databases, examples, demonstrations and other small scale uses. However, on a larger scale it seems like the obviously wrong choice. We know, record and memorize history so that we and the generations beyond us might not have to repeat mistakes. Isn't using something we know to be inefficient an example of such a mistake? Then again maybe the simplicity of XML makes it the better choice.

Works Cited

Garcia-Molina, Hector, Jeffrey D. Ullman, and Jennifer Widom. "Chapter 1." *Database Systems: The Complete Book*. 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2009. N. pag. Print.