Caleb Rogers

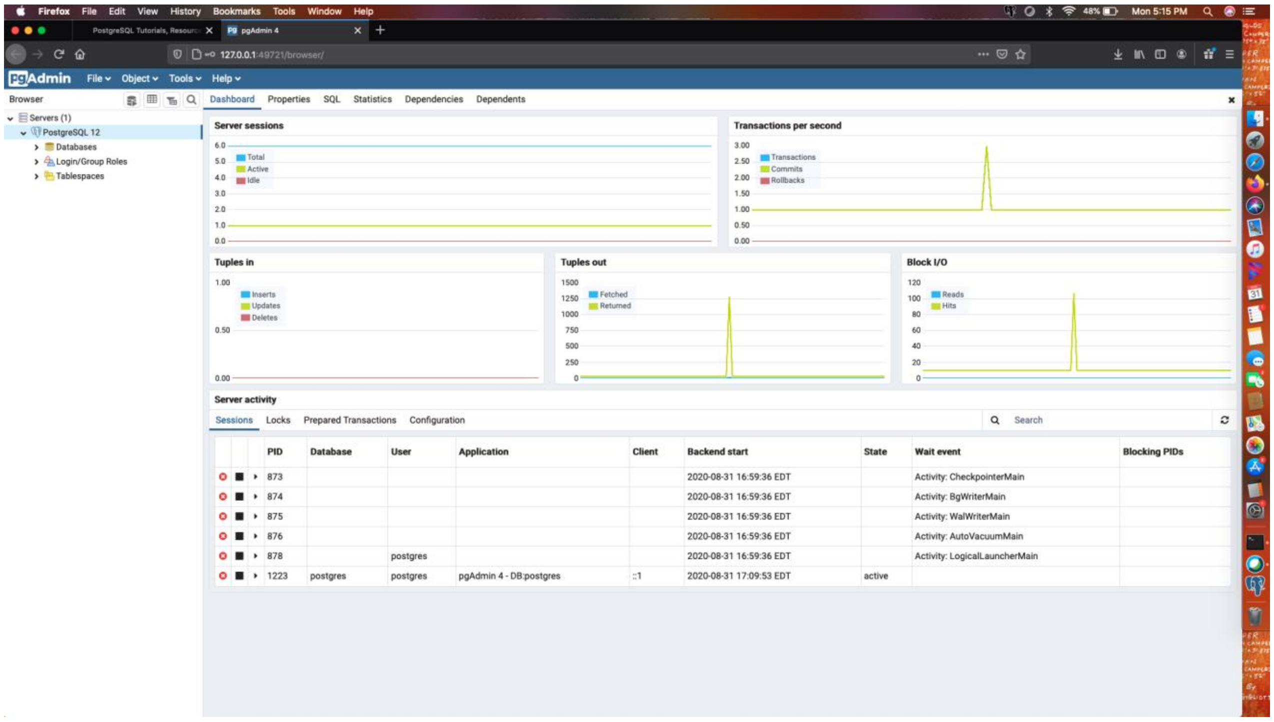
Professor Alan Labouseur

August 31st, 2020

Database Systems

# Lab 1

## pgAdmin 4:



| PID | Database | User | Application | Client | Backend start | State | Wait event | Blocking PIDs |
|---|---|---|---|---|---|---|---|---|
| 873 | | | | | 2020-08-31 16:59:36 EDT | | Activity: CheckpointerMain | |
| 874 | | | | | 2020-08-31 16:59:36 EDT | | Activity: BgWriterMain | |
| 875 | | | | | 2020-08-31 16:59:36 EDT | | Activity: WalWriterMain | |
| 876 | | | | | 2020-08-31 16:59:36 EDT | | Activity: AutoVacuumMain | |
| 878 | | postgres | | | 2020-08-31 16:59:36 EDT | | Activity: LogicalLauncherMain | |
| 1223 | postgres | postgres | pgAdmin 4 - DB:postgres | ::1 | 2020-08-31 17:09:53 EDT | active | | |

**Short Essay 1:** *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.*

Most organizations require databases to organize their data into information that allows their operations to work. This is because the world is made up of data, but without understanding the data and having context to the data, said data is useless. When an organization can add context to necessary data, they then have useful information that can be used to carry out their business. Information is very valuable. Storing information and being able to understand it to carry out business creates a demand for databases. An example of an organization that uses a database is Marist College's Client Technologies.

When a student, professor, or employee at Marist needs technical support, they can contact ResNet to receive help. As an employee of Client Technologies, when ResNet receives a call or email related to a technical issue, we will search our database with a name or CWID to find the person calling and make a new ticket related to the specific incident. ResNet's database is the Marist Mainframe, and it contains a profile for everyone associated with Marist. Having this database is crucial for understanding a client's history and status; and what we as a department can do to help.

When currently in contact with a client, it is very important to collect as much data possible. The first data that needs to be obtained is either a name or CWID that allows us to search the client in our database. Once the client is found in the database, we have context that our client is associated to Marist, has a profile to make a ticket on, and then can continue to help. Poor data like an incorrect name or CWID only offers useless information, as we cannot help anyone who doesn't have a confirmed profile. Once the client is found in the database, we have enough information to continue. ResNet then must collect data on what the issue is, and the severity of the issue. The data provided on the issue is used to determine how to solve the issue, and this solution needs context to the data given. If the IT employee knows the solution, the context is given from the knowledge of the employee. Otherwise the employee will need to research a solution, or ask an employee in a superior position. If context is found to support the data of the client, then useful information on the issue can be relayed back to the client, otherwise further action will be required to find the necessary context.

Once understanding the client and the issue, data on the severity must be found. Knowing whether the client needs immediate action or an appointment is necessary on determining how ResNet will help. ResNet must communicate with the client on whether they will come to the office, if an employee needs to go on-site, or if the problem can be solved remotely. Using data received on the severity of the issue and where the solution will take place is necessary for ResNet to look at our schedule, find context on who and when an employee can help, and then we have the information necessary to schedule an appointment and attempt solving the issue.

There is a lot of data that must be obtained from clients in order to apply knowledge and context to result useful information. The absence of some of data can make finding a solution very difficult, as context cannot be applied if the data isn't there; and if the data is there but it is poor data, then the context applied will either be difficult to obtain, or often incorrect. The result of poor data combined with possibly bad context will often lead to a solution that will not work. Databases, like the Marist Mainframe, are necessary for storing all this important data with the context provided through the filter of an employee. These databases are crucial for extracting information and this information, is the key that helps the world go round.

**Short Essay 2:** *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?*

Database Management Systems (DBMS) originally were extensive and required large computers to handle their size. Before Ted Codd introduced the relational model, DBMS's large size required a visualization of the data. This demand resulted with data models like the tree-based hierarchical model and the graph-based network model. These pre-relational models helped display a structure of information for the databases, but could only support low-level query languages. Codd met the need for a more efficient model with the relational database system. This model has paved the way for how programmers interact with databases to this day.

The relational DBMS wouldn't exist without the pre-relational hierarchical and network data models. Appearing in the 1960's, DBMS's were in need to track information for large organizations like banking systems, airline reservation systems, and corporate record keeping. The hierarchal data model displayed semi structured data in a tree-orientated model. This model operated at a physical level. This falls short of the relational model, as it could only be programmed at a low-level. Another data model, the network model, was graph-oriented. Similar to the hierarchical model, the network model also ran at a physical level and could not be programmed at a high level like relational DBMS's. Though both pre-relational models can only be programmed at a low-level, they do offer a flexibility in the ability to make changes in the database that is not as simple in relational models. Pre-relational models do offer more flexibility, though relational models are more efficient and still offer reasonable versatility. This efficiency and the ability to program at a high level is the reason relational models are still preferred.

While pre-relational models were based on trees and graphs, the relational model was alternatively based on tables and arrays. Relational DBMS's allowed complex data structures to handle rapid responses to queries, without the programmer having to be concerned with the storage structure. This meant queries could be expressed in high-level languages, resulting in the commonly used Structured Query Language (SQL). Moving to high-level languages greatly increased efficiency and set the path to how programmers use databases today. Though SQL offers a reliable language for relational models, XML is a language that uses hierarchically nested tagged elements, in ways similar to older pre-relational models. The advantage to XML is that it better displays data transfer and is better readable from programmers. While XML has the better elements of a more physical based language, it also shares the inefficiency from pre-relational models. XML is a good option if the programmer wanted a more versatile option that better helps data track, but a more efficient language is often needed to maintain the large databases of today.