# CSE312 Open Source Report

## Technology: PostgreSQL Database


## Group: Lawzeem Latif, Mckayla Brefka, and Victoria Faltisco


For phase 2 we used PostgreSQL as our database. PostgreSQL is a free and open-source object-related database. This database utilizes and extends SQL to safely store data. PostgreSQL was not only designed to be similar to UNIX, it was designed to be portable. It is able to be run on many platforms including Mac OS X, Solaris and Windows making it perfect for our application. The mirror GitHub repository of Postgre is found here: https://github.com/postgres/postgres The GitHub for PostgreSQL is found here:https://github.com/postgres

The source code is available under the PostgreSQL license, which is also open source, meaning that it is free to use, modify, and distribute in any means necessary. The license can be found here: https://www.postgresql.org/about/licence/

Hand in hand with our Django models, we utilized the PostgreSQL features to tailor the database to our application. This resource is customizable, it allows custom functions to be added in various programming languages including C/C++, Java, and in our case Python. PostgreSQL offers us as developers user-defined types, table inheritance, foreign key references, and multi-table inheritance for one-to-one field references. Within our models we used CharField, ImageField, ManytoManyField, and ForeignKey. These inputs allowed us to customize the data types for our database, and how to best store our data without the underlying complications.

Once the database fields are established, the database works by using queries. PostgreSQL parses, analyzes, plans, and then executes. With each call our application makes to the database, PostgreSQL parses, analyzes, plans, and then executes.

Beginning with parsing, the query goes through a set of steps. First the parser converts the query string into parse tree nodes which then are converted to tokens. The query type is then derived using these tokens, and it is finally loaded and generated into a raw query tree. The documentation for the query tree is found here: https://www.postgresql.org/docs/9.6/querytree.html.

Once the query tree is generated, transformations are applied on the different clauses of the query. Posgres then makes a list of the different columns that will be modified within this query.  Documentation for transformations: https://www.postgresql.org/docs/9.5/infoschema-transforms.html

Moving forward, a plan tree is generated containing which execution model is needed. The tree includes ‘plan nodes’ which each hold information about a task including cost. These nodes can be referred to as tasks. Postgres takes the associated cost of each task to estimate execution time. This is part of the output of this step to move onto the

execution. Documentation for the Plan/Optimization step: https://www.postgresql.org/docs/9.5/planner-optimizer.html

Lastly, Postgres begins the execution recursively moving through the plan tree until the results are complete.