March 26, 2019

Ilaria Battiston

Google Summer of Code 2019

Project proposal
PostgreSQL - Develop Performance Farm Database and Website

1 Basic details

University: Università degli Studi di Milano-Bicocca

Location & time zone: Milan, Italy, CEST

Date of birth: 16/01/1998

Email: ilaria.battiston@gmail.com

LinkedIn: ilaria-battiston

GitHub: IlariaB

Postgres Slack: Ila

Curriculum Vitae: IlariaB/Curriculum-Vitae

2 Abstract

PostgreSQL is a famous database management system, widely used by servers and with an open source license. This means there is a large community dedicated to developing and bug-fixing, intensely working through mailing lists.

The Performance Farm is an useful way to test Postgres' functionalities while changes are being made, to analyse its efficacy on different operating systems. To further support the community effort, this project has to be extended building a database and a website on top of it, to make results easier to browse and display.

My goal is making a web application using Python to interface server and client. The code will be scalable, portable and light-weight, while providing users a functional interface to interact with the performance data.

The website will rely on a database, with optimised structure and queries to guarantee speed and an efficient use of the resources. The Django framework will be deployed so that the browser can send requests of search, review and storage.

The application will also take care of parsing, users handling and securing connections, implementing RESTful API and respecting web standards.

All changes will be subject to testing and bug fixes, to have a complete and coherent project with a clear documentation so that the final product is easy to set up and maintain.

3 About me

My name is Ilaria Battiston, and I'm a 21-year-old female student coming from Milan, Italy. I'm currently attending my third year at Università degli Studi di Milano-Bicocca and I plan to get my Bachelor's degree in July.

My current goal is to become a data scientist. I'm particularly interested in Health Big Data, Bioinformatics and Data Management: I'm applying for masters in those fields, and the first choice is Data Engineering and Analytics at Technical University of Munich.

My skills are mostly self-taught by reading books or doing online courses, and in the last year I decided to turn my theoretical knowledge into something more practical and work on projects such as the GSoC, to keep on learning and improving my skills.

I started using PostgreSQL in November 2018, during my internship in a local research team, making analytics on health data. During these months I've mostly working on query optimization and physical access structures. I'm also taking Advanced Databases classes at my university.

In the past months I had the opportunity to get in touch with the PostgreSQL community, especially at FOSDEM 2019, where I encountered some really kind associates and I participated to most events of the database tracks.

Next May I'm going to attend PGConf in Leipzig and PGDay in Bologna: I'm really looking forward to both of them since this community gave me so much already and I can't wait to participate to the events.

Other than that, I manage unixMiB, an association of volunteers at my university whose aim is to spread Open Source knowledge. We organize events and offer open solutions for the issues of our peers. We collaborate with organizations such as FSFE, Mozilla and LibreOffice: our current main goal is hosting KDE Akademy 2019 in Milan.

I'm also working sponsored by the Wikimedia Foundation as community manager of WikiToLearn, a no-profit project.

4 Project approach

4.1 Expectations

I chose Develop Performance Farm Database and Website since it immediately caught my attention. I have experience with databases since I extensively use them while making analytics, but I also like web developing: seeing there was a way to blend those fields, storing and displaying performance data, made me interested and wanting to look deeper into it.

Other than that, it can be an amazing way to refresh my current skills and integrate them with the new abilities I will acquire working on databases and websites.

The GSoC wouldn't only be a great opportunity to improve and grow as a person, but also a way to connect with the community, build new relationships and keep on participating in the future. I will engage in being a positive addition and bring some useful contributions. I am motivated, hard-working and able to take constructive criticism while maintaining an optimistic attitude.

The project requires Python and SQL programming skills, and I have both. I've never used Django, but I started to read the documentation and I have a good understanding of web development paradigms. I previously used PHP and JavaScript to implement web clients for data visualization and inspection.

After the GSoC period I expect to have a clear understanding of the Django framework and related practical skills such as building web applications in Python. I also wish to learn more about PostgreSQL, how the performance data is composed and how it can be stored and fetched.

Since I'm going to graduate in July, I plan to have all my academic work done by the first week of June, so I can fully focus on GSoC. I work part time, yet this only takes me a hour every day on average. I will take a week of vacation in August, but other than that I will be free most of the time. I can also work while I'm on holiday if needed.

4.2 Work breakdown

I already cloned `pgperffarm` from its upstream source, and I started looking at the existing code. I will open issues on GitHub if necessary, while adapting a bit the environment to my computer; I also took a look at the given example, the Build Farm, and understood its functioning.

I normally use a Mac, but I set up a Linux-based (Debian 9) virtual machine to have a fresh installation of all components and to separate project files from personal ones. I am already testing the code of Performance Farm Project on both the operating systems, taking note of differences and bugs which come up.

So far I think working from macOS and connecting to Debian with `ssh` and `rsync` is the best option, but if any issues arise I can easily transfer everything to a physical partition with Linux installed or try AWS cloud computing.

After setting up the environment, coding can begin. Aside from bug fixing, refactoring and documentation writing, which will be done progressively, I can see there are two major tasks: interacting with the database and building a web site on top of it.

I will do my best to keep the code (especially server-side) simple to understand, minimal and scalable. It would be useful to have a functioning application on Linux, macOS and Windows: I will begin focussing on the first, and if possible then extend the beta version of the project.

To construct the schema, I can sketch ER models and carefully analyse the structure to grant normalisation and referential integrity. I will always keep in mind the purpose and requirements of the project, since the server-side code extensively interacts with the DBMS.

I already saw what the JSON file looks like, and the database will be created according to the related fields. I'm going to run some tests to see which type of indexing and table structure works faster: the data could be split into several relations, or kept in a bigger one. I can come up with some probable frequently-executed queries, optimize them and integrate them in the GUI.

Handling users and roles is something to still be carefully planned, yet I would like to have a simple structure. I've seen a couple bugs of Django, especially older versions, and having a too complex system might compromise the portability.

All Python code will be organised in packages, each of them containing specific parts of the project. There will be a web interface, relying on server-side technologies. I want to make the configuration documentation as clear as possible, covering every potential issue.

The Django infrastructure will establish a connection with the database, fetch results, parse them and send them to the client; it could also reverse the process with JSON files uploaded by the client, if necessary.

Security is an important aspect as well: all sensitive data, if present, is going to be encrypted or omitted. There are some Django extensions, such as `django-pgcrypto`, which integrate this functionality. Aside from protecting single fields, the security of the system will be implemented on the whole server-side. I looked at the documentations and there are some useful modules like CSRF protection. The site will be deployed behind HTTPS.

The Performance Farm website will allow users to upload, search and review test results. This requires having the possibility to recognise an user (I thought about using the animal credentials, but this has to be discussed), so there must be a login and logout option.

The interface can also offer a sample of recent collected data ordered by timestamp, and the possibility to select other parameters for sorting. All potentially sensitive information on displayed data will be removed, to allow privacy-preserved analytics.

Searches will be possible according to operating system, date and other relevant fields: results will be parsed, eventually serialized and displayed in the web page.

Every user can have a personal profile, consisting in a summary of the account details along with all the submitted data. If the same user has more devices, they can be either appear altogether or in separate tabs.

Another idea would be to implement the REST framework via `django-rest`, which is useful to build APIs although it's only compatible with Django 1.11 and later. Otherwise, this might work using native Python via HTTP client libraries to contact the API, for instance `requests`.

A goal which could be accomplished is porting the project from Python2 to Python3: Python2 is going to be no longer maintained past 2020. Dependencies should be checked, but other than that it's just code refactoring and optimization such as fixing the `import` and `print` statements.

5 Schedule

5.1 Community bonding

Community bonding will be a great opportunity to get to know mentors and start experimenting with the code. During this period, I aim to start developing a professional relationship with my team and refining the hypothetical schedule, turning it into a more practical one with the new knowledge acquired.

I commit to always be reachable and stay in touch with mentors and eventual teammates for both community bonding and coding phases, providing any additional personal information needed such as telephone number or emergency contact numbers. I will use the communication best practices stated in the GSoC and Postgres websites.

It will surely be easier to have an overall idea of the project after a full immersion, with some orientation from my mentor. I plan to strengthen my skills of Python and PostgreSQL with self coding, especially using Django.

The latter is the most important competence I need to work on, reading manuals, practising and seeing examples. I'm going to extensively look through the existing code and data which will be stored in the database, breaking the parts down to understand their purpose.

At the same time, refreshing my knowledge on JavaScript and the AJAX programming paradigm might be useful for the frontend development. I will rely on Django as much as possible, yet if the website requires a high level of interactivity with the user I can implement some front-end code with JS as well.

After having a deeper insight into the project and all the involved tools, I can check whether expectations and goals are realistic, eventually adjusting them.

5.2 Coding phases

I organized the schedule so that I have some spare days at the end of each phase, to debug code or repair eventual issues that may come up. After each step I'll write detailed reports and documentation which will be merged and published in the end.

- May 27 - coding officially begins
  1) At most 1 week to have a full understanding of the existing resources, a complete environment setup, a dedicated GitHub repository and access to the test data
  2) 1 week to parse JSON and plain files, having a deep comprehension of their structure
  3) 1-2 weeks to build a database and optimise its architecture
  4) Some days to fix bugs and test the data insertion from raw files
- June 28 - end of Phase 1
  1) 2 weeks to completely implement the server-side infrastructure, configuring users, connection to the database and fetching results
  2) 1 week to work implementing queries from the user interface and parsing results to have a clear and understandable output
  3) 1 week to build the browser application and refactor the client-side code
- July 26 - end of Phase 2
  1) 1 week to (optionally) implement REST and work on data serialization, or else keep on working on the client code
  2) 1-2 weeks for final code refactoring, debugging and optimization
  3) 1 week to finish writing documentation and release the project
- August 26 - end of Phase 3

After GSoC, I would love to keep in touch. I want to attend some other Postgres conferences, I will be available to fix bugs users might find in the future and implement more features which might be needed.