



Servo & SQLGitHub

A JOURNEY OF TRANSFORMING FAILURES INTO A GOOD IDEA

Servo

- ▶ A new browser engine written from scratch in Rust by Mozilla
- ▶ Original Project: Write an Android frontend for servo
- ▶ Issues, Issues and issues
 - ▶ Zero knowledge about mobile development and Rust
 - ▶ Incomplete, outdated documentation
 - ▶ Almost no support from Mozilla
 - ▶ Possible directions pitched by mentor all led to dead-end
- ▶ Tried approaching in various ways, but to no avail
- ▶ Made installation and packaging guide and reported issues

SQLGitHub – Motivation (I)

- ▶ **Who is stealing all my easy bugs/issues?**
- ▶ There are very limited tools for managing GitHub organizations
- ▶ Open-source organizations are usually understaffed
- ▶ The Servo project on GitHub for example, contains 129 repositories managed by practically **1 person**.



SQLGitHub – Background

- ▶ **Organization:** Organizations are shared accounts where businesses and open-source projects can collaborate across many projects at once.
- ▶ **Repository:** A repository contains all of the project files, and stores each file's revision history.
- ▶ **Commit:** An individual change to a set of files.
- ▶ **Issue:** Suggested improvements, tasks or questions
- ▶ **Pull Request:** Proposed change to a repository.

Organization

Repository

Repository

Repository

- Commits
- Issues
- Pull Requests
- ...

- Commits
- Issues
- Pull Requests
- ...

- Commits
- Issues
- Pull Requests
- ...

SQLGitHub – Motivation (II)

- ▶ Common questions/problems an organization admin include:
 - ▶ Obtain certain metrics of the organization in machine-friendly format for post-processing (eg. KPI report)
 - ▶ Get the current list of projects hosted on GitHub
 - ▶ List of the most popular repositories in the organization
 - ▶ Get the list of issues closed (resolved) for the past 7 days
 - ▶ What are the critical issues that are still left open?
 - ▶ Who are the top contributors of the past month?
 - ▶ ... endless possible questions

Abstract

- ▶ SQLGitHub features a SQL-like syntax that allows you to:
Query information about an organization as a whole.
- ▶ You may also think of it as a better, enhanced frontend layer built on top of GitHub's RESTful API

```
SQLGitHub> select updated_at, title from servo.issues.closed.3 order by updated_at desc
[u'updated_at', u'title']
[datetime.datetime(2017, 10, 15, 15, 25, 5), u'OSX Travis fix and Gecko update']
[datetime.datetime(2017, 10, 15, 12, 22, 12), u'Change AttrValue::Url to AttrValue::ResolvedUrl']
[datetime.datetime(2017, 10, 15, 12, 22, 11), u'Add a default 'unstable' feature to CEF']
[datetime.datetime(2017, 10, 15, 12, 22, 9), u'Update stable Rust version to 1.20.0']
[datetime.datetime(2017, 10, 15, 11, 52, 13), u'Support Range<T>, RangeFrom<T>, RangeTo<T> and RangeFull']
[datetime.datetime(2017, 10, 15, 11, 19, 18), u'Update bindgen.']
[datetime.datetime(2017, 10, 15, 10, 57, 11), u'Upgrade to rustc 1.22.0-nightly (7778906be 2017-10-14)']
[datetime.datetime(2017, 10, 15, 9, 20, 51), u'style: Do not expose LocalMatchingContext.']
[datetime.datetime(2017, 10, 15, 8, 55, 24), u'Update domparsing spec links to not point at WHATWG']
[datetime.datetime(2017, 10, 14, 22, 15, 9), u'Introduce ClipChain']
[datetime.datetime(2017, 10, 14, 21, 34, 49), u'Update OSMesa.']
```

Introduction – Supported Schema

► **SELECT**

select_expr [, select_expr ...]

FROM {org_name | org_name.{repos | issues | pulls | commits}}

[**WHERE** where_condition]

[**GROUP BY** {col_name | expr}

[ASC | DESC], ...]

[**HAVING** where_condition]

[**ORDER BY** {col_name | expr | position}

[ASC | DESC], ...]

[**LIMIT** row_count]

Introduction – Use Case (I)

- ▶ Get name and description from all the repos in apple.
 - ▶ `select name, description from apple.repos`

```
SQLGitHub> select name, description from apple.repos
[u'name', u'description']
[u'cups', u'Official CUPS Sources']
[u'swift-lldb', u'This is the version of LLDB that supports the Swift programming language & REPL.']
[u'swift', u'The Swift Programming Language']
[u'swift-llbuild', u'A low-level build system, used by Xcode 9 and the Swift Package Manager']
[u'swift-package-manager', u'The Package Manager for the Swift Programming Language']
[u'swift-llvm', None]
[u'swift-clang', None]
```


Introduction – Use Case (II)

- ▶ Get last-updated time and title of the issues closed in the past week (7 days) in servo listed in descending order of last-updated time.
 - ▶ `select updated_at, title from servo.issues.closed.7 order by updated_at desc`

```
SQLGitHub> select updated_at, title from servo.issues.closed.7 order by updated_at desc
[u'updated_at', u'title']
[datetime.datetime(2017, 10, 16, 17, 19, 58), u'Remove the use of unstable Rust features.']
[datetime.datetime(2017, 10, 16, 16, 21, 38), u'fix windows build issue #18055']
[datetime.datetime(2017, 10, 16, 16, 21, 36), u'Install/build on Windows 10 not working']
[datetime.datetime(2017, 10, 16, 15, 12, 56), u'Fix tests']
[datetime.datetime(2017, 10, 16, 14, 49, 43), u'Make every function unsafe for now']
[datetime.datetime(2017, 10, 16, 14, 44, 28), u'style: Remove the ElementExt trait.']
[datetime.datetime(2017, 10, 16, 14, 27, 14), u'Add cast function for transform2d/3d.']
[datetime.datetime(2017, 10, 16, 14, 5, 44), u'Release build fix']
[datetime.datetime(2017, 10, 16, 14, 4, 17), u'Add Rust API for startup, shutdown, and canPlayType']
[datetime.datetime(2017, 10, 16, 14, 1, 17), u'Run JetStream benchmark']
```

Introduction – Use Case (III)

- ▶ Get top 10 most-starred repositories in servo.
 - ▶ `select concat(concat("(", stargazers_count, ") ", name), ": ", description) from servo.repos order by stargazers_count desc, name limit 10`

```
SQLGitHub> select concat(concat("(", stargazers_count, ") ", name), ": ", description) from servo.repos order by
[u'concat(concat("(", stargazers_count, ") ", name), ": ", description)']
[u'(10246) servo: The Servo Browser Engine']
[u'(801) webrender: A GPU-based renderer for the web']
[u'(529) html5ever: High-performance browser-grade HTML5 parser']
[u'(267) rust-url: URL parser for Rust']
[u'(266) cocoa-rs: Cocoa/Objective-C bindings for the Rust programming language']
[u'(200) gaol: Cross-platform application sandboxing for Rust']
[u'(158) ipc-channel: A multiprocess drop-in replacement for Rust channels']
[u'(113) rust-cssparser: Rust implementation of CSS Syntax Level 3']
[u'(111) homu: A bot that integrates with GitHub and your favorite continuous integration service']
[u'(106) rust-mozjs: Rust bindings to SpiderMonkey']
-
Total rows: 10
Total execution time: 9.585s
```

Introduction – Use Case (IV)

- ▶ Get top 10 contributors in servo for the past month (30 days) based on number of commits.
 - ▶ `select login, count(login) from servo.commits.30 group by login order by count(login) desc, login limit 10`

```
SQLGitHub> select login, count(login) from servo.commits.30 group by login order by count(login) desc, login limit 10
[u'login', u'count(login)']
[u'bors-servo', 374]
[u'emilio', 131]
[u'nox', 123]
[u'glennw', 73]
[None, 68]
[u'SimonSapin', 61]
[u'philn', 33]
[u'cpearce', 31]
[u'jdm', 26]
[u'bholley', 24]
-
Total rows: 10
Total execution time: 186.437s
```

Introduction – Technology Stack

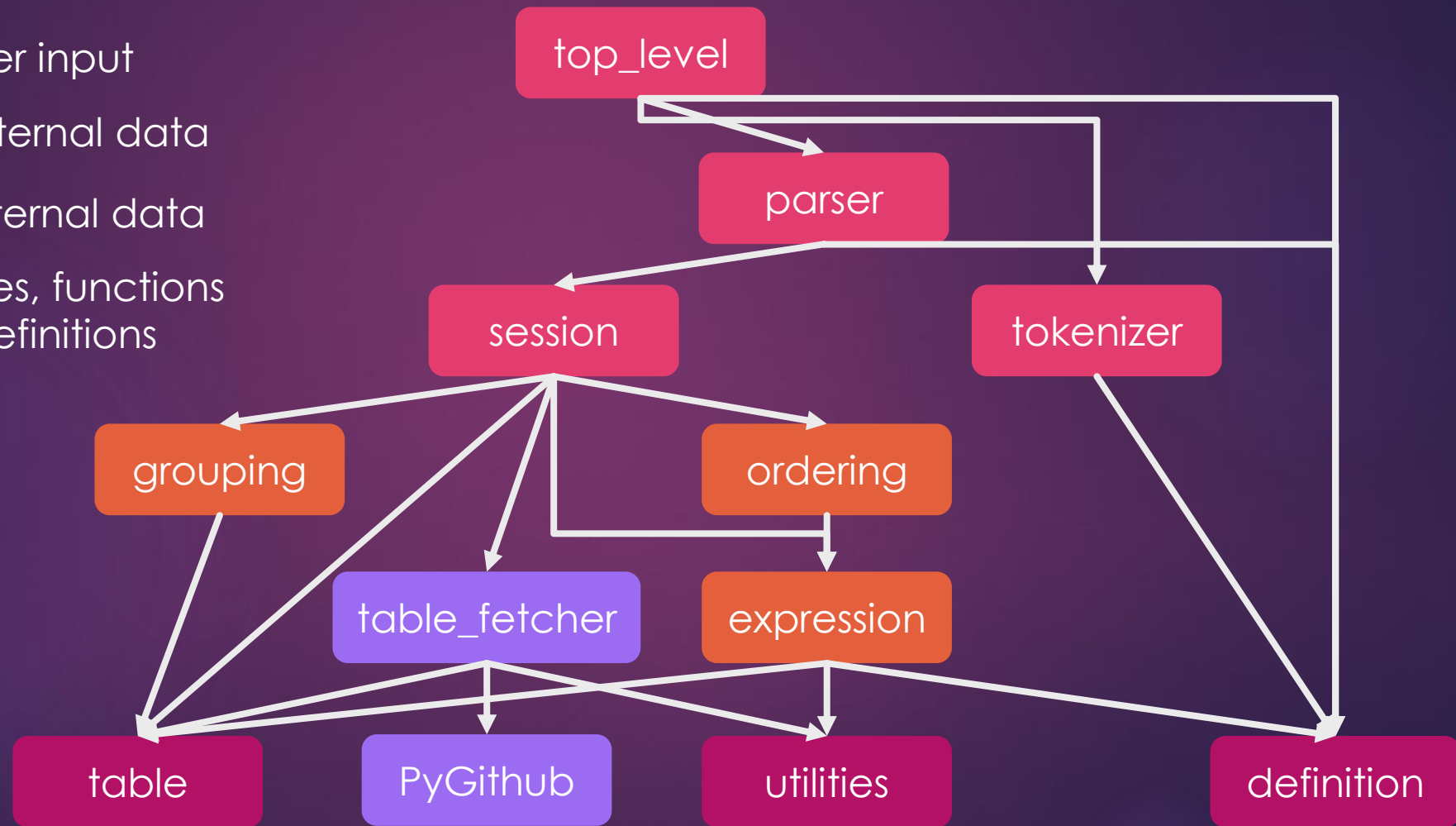
- ▶ **Python**
- ▶ **re** & **regex**, regular expression libraries
- ▶ **PyGithub** (patched), an unofficial client library for GitHub API
- ▶ **prompt_toolkit**, a library for building prompts
- ▶ **pygments**, a library for syntax highlighting

Introduction – (Simplified) Flow



Introduction – Architecture

- process user input
- process external data
- fetches external data
- base classes, functions and SQL definitions



Introduction – Challenges (I)

- ▶ Algorithm of parsing is almost identical to that of expression evaluation → waste of time
- ▶ **Lazy Parsing**: Only parse **clauses** (eg. select, from, where) and **comma-separated fields**
- ▶ Comma-separated fields, strings and escape characters
Evaluate this: `concat("[]\"Stars\"(: ", stargazers_count)`
- ▶ `concat("[]\"Stars\"(: ", stargazers_count)`
`concat("[]\"Stars\"(: ", stargazers_count)`
- ▶ `concat("[]\"Stars\"(: ", stargazers_count)`

Introduction – Challenges (II)

- ▶ Extracting relevant fields from expressions to fetch
- ▶ `select concat("[\"-> avg(stargazers_count)\": ", stargazers_count - avg(stargazers_count), "]" , name) from apple.repos where description like "%library%" order by id`
- ▶ Algorithm: for each expression,
 - ▶ Remove all literal strings. Use `r"\"(?:[^\\""]|\\.)*\\""` to match.
 - ▶ Find all possible tokens with `r"([a-zA-Z_]+)(?:[^\(a-zA-Z_]|$)"`.
 - ▶ For each token, check if it's a predefined token (ie. part of SQL).

Introduction – Challenges (III)

- ▶ Expression Evaluation is really complicated
 - ▶ Regular (eg. `concat`, `floor`) and Aggregate functions (eg. `max`, `min`)
 - ▶ Have to evaluate an entire table at once
 - ▶ Nested functions (eg. `sum(avg(field_a) + avg(field_b))`)
 - ▶ Use **recursive regex patterns** to extract tokens – `r"\((?:(>[^\(\)]+|(?R))*\)"`
 - ▶ Assign special precedence and insert extra logic in place
 - ▶ Operator Precedence
- ▶ **Modified 2-stack evaluation approach +**
- ▶ **Finite State Machine + One-token Lookahead**

Introduction – Challenges (IV)

- ▶ Python's built-in sort is not customizable:
`sorted(iterable, *, key=None, reverse=False)`
- ▶ order by requires sorting with multiple keys each with potentially different reverse:
`order by field_a desc, field_b asc, field_c, desc`
- ▶ Wrote **custom sort** that integrates better with the workflow

Future Directions

- ▶ Improve SQL, MySQL compatibility
- ▶ Extend to end users not just organizations
- ▶ Migrate to the new GraphQL backend (GitHub API v4)
- ▶ Integrate SQLGitHub directly on the server end (better efficiency and perhaps better security!)

Acknowledgements

- ▶ We would like to thank:
 - ▶ Shing Lyu, former software engineer at Mozilla Taiwan for the mentorship
 - ▶ Irvin Chen, Liaison of MozTW (Mozilla Taiwan Community) for coordinating the program
 - ▶ Prof. Cheng-Chung Lin for organizing the program