SQLGitHub — Managing GitHub Organization Made Easier

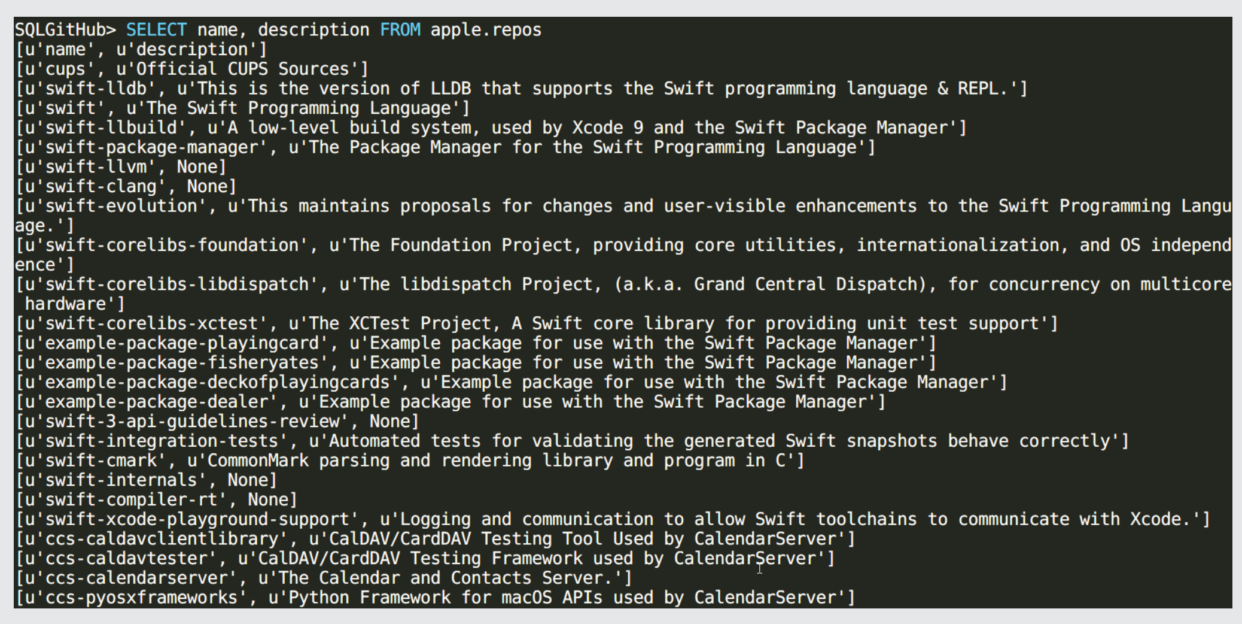
Authors: Jasmine Chen (0113110 陳柏翰) <jasmine.chen.cs@gmail.com>, Jeremy Wang (0312546 王子元) <jeremywangyuan@gmail.com>
Mentor & Adviser: Shing Lyu <shing.lyu@gmail.com>, Cheng-Chung Lin <cclin@cs.nctu.edu.tw>

Abstract

Managing open-source communities is a common problem for tech companies. Investing resources in open-source communities usually provides little-to-no benefit. Therefore, the open-source team within a company is usually under-staffed or even nonexistent.
The largest software development and open-source collaboration platform, GitHub, offers a comprehensible API which allows developers to fetch information and manage projects programmatically. This API however, fails to provide an entry point to retrieve aggregate data of an organization which made it tedious to use for organization administrators.
This defect prompted us to "repackage" this API by designing a SQL-like syntax to query aggregate information of an organization. This makes the work of organization administrators much easier. We name it "SQLGitHub".

Description

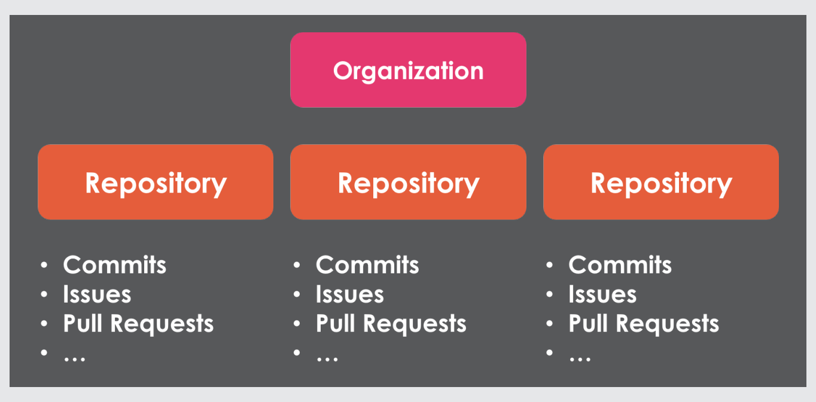
SQLGitHub features a SQL-like syntax that allows you to query aggregate information about an organization. You may also think of it as a better, enhanced frontend layer built on top of GitHub's RESTful API.


```
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]
[u'swift-evolution', u'This maintains proposals for changes and user-visible enhancements to the Swift Programming Langu
age.']
[u'swift-corelibs-foundation', u'The Foundation Project, providing core utilities, internationalization, and OS independ
ence']
[u'swift-corelibs-libdispatch', u'The libdispatch Project, (a.k.a. Grand Central Dispatch), for concurrency on multicore
 hardware']
[u'swift-corelibs-xctest', u'The XCTest Project, A Swift core library for providing unit test support']
[u'example-package-playingcard', u'Example package for use with the Swift Package Manager']
[u'example-package-fisheryates', u'Example package for use with the Swift Package Manager']
[u'example-package-deckofplayingcards', u'Example package for use with the Swift Package Manager']
[u'example-package-dealer', u'Example package for use with the Swift Package Manager']
[u'swift-3-api-guidelines-review', None]
[u'swift-integration-tests', u'Automated tests for validating the generated Swift snapshots behave correctly']
[u'swift-cmark', u'CommonMark parsing and rendering library and program in C']
[u'swift-internals', None]
[u'swift-compiler-rt', None]
[u'swift-xcode-playground-support', u'Logging and communication to allow Swift toolchains to communicate with Xcode.']
[u'ccs-caldavclientlibrary', u'CalDAV/CardDAV Testing Tool Used by CalendarServer']
[u'ccs-caldavtester', u'CalDAV/CardDAV Testing Framework used by CalendarServer']
[u'ccs-calendarserver', u'The Calendar and Contacts Server.']
[u'ccs-pyosxframeworks', u'Python Framework for macOS APIs used by CalendarServer']
```

Background Information

The structure of GitHub is as follows:

- Organization: Organizations are shared accounts where 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.



Motivation

Managing a GitHub repository (or project) is no easy task, let alone managing a typical GitHub organization with dozens, hundreds, or even thousands of repositories. Common tasks for an organization administrator usually involves obtaining certain metrics of the organization in human or machine-friendly format for post-processing. The specific tasks may include:

- Get the current list of projects hosted on GitHub
- List of the current repositories in the organization ordered by popularity
- 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

For each of the listed tasks, it would take roughly the following steps:

1. Utilize libraries for HTTP requests and JSON manipulation
2. Obtain API key from GitHub
3. Lookup API reference to find the needed API URIs
4. Authenticate with obtained API key
5. List all repositories within the organization
6. For each repository, list all needed targets (e.g. Issues, commits …etc.)
7. Navigate through the complex structure of targets to extract wanted information
8. Aggregate extracted data
9. Filter by hardcoded conditions
10. Output information in friendly format

The above steps would take roughly **a week (2400 minutes)** for an ordinary developer without experience with GitHub APIs.

Using our tool SQLGitHub, the process would be simplified to something like this:

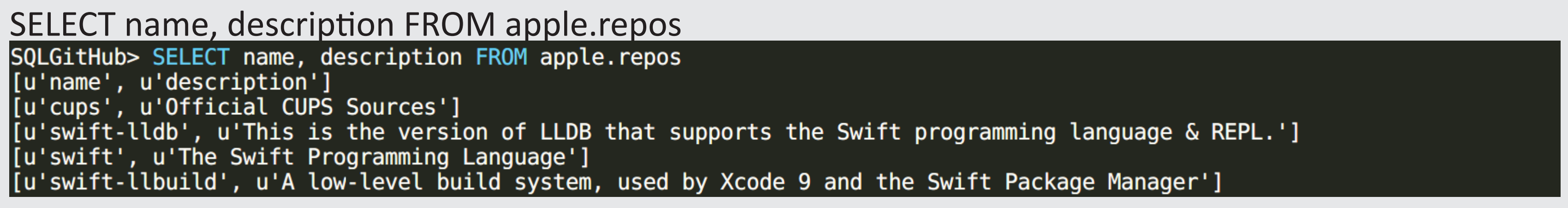
1. Obtain API key from GitHub
2. Lookup required information from the reference
3. Launch SQLGitHub and enter SQL query

The above steps would take **less than 20 minutes** for someone with knowledge of basic SQL (about 120x speedup).

Example Use Cases

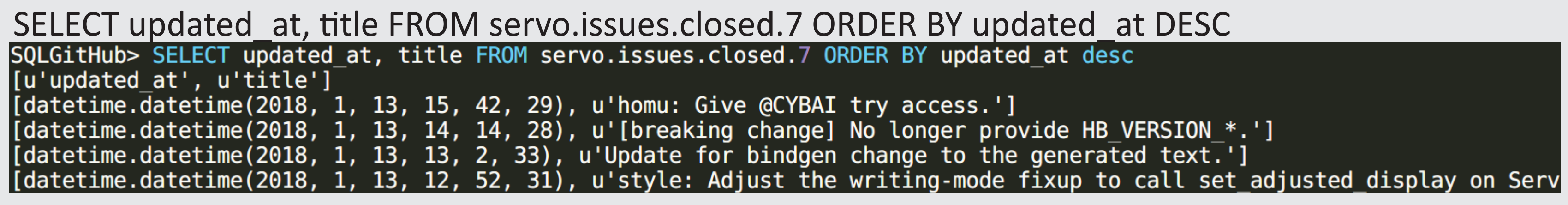
1. **Get name and description from all the repos in apple.**

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']
```

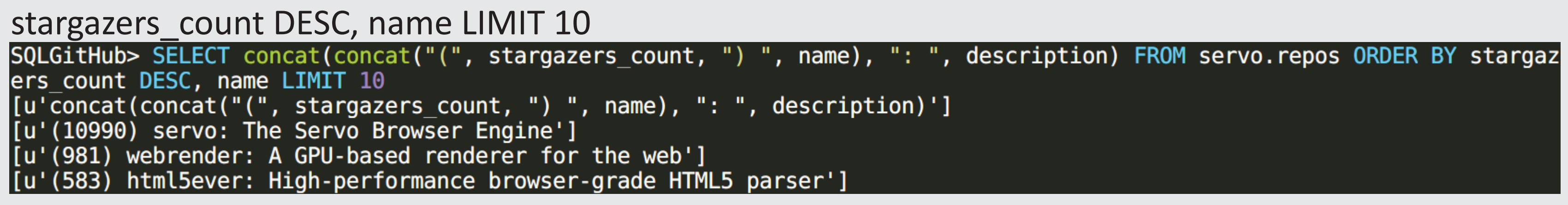
2. **Get last-updated time and title of the issues closed in the past week (7 days) in servo listed in descending order of last-updated time.**

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(2018, 1, 13, 15, 42, 29), u'homu: Give @CYBAI try access.']
[datetime.datetime(2018, 1, 13, 14, 14, 28), u'[breaking change] No longer provide HB_VERSION_*.']
[datetime.datetime(2018, 1, 13, 13, 2, 33), u'Update for bindgen change to the generated text.']
[datetime.datetime(2018, 1, 13, 12, 52, 31), u'style: Adjust the writing-mode fixup to call set_adjusted_display on Serv
```

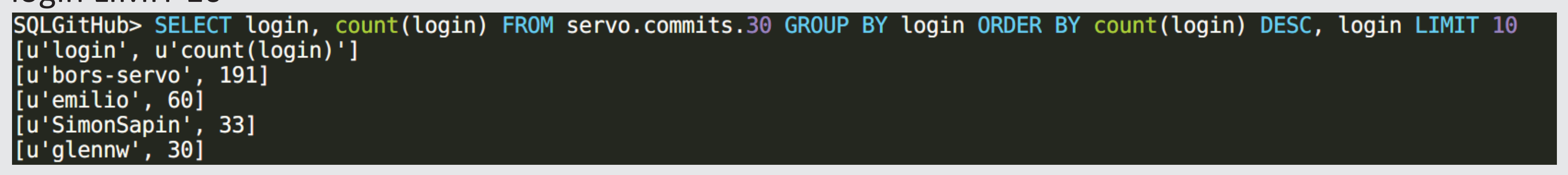
3. **Get top 10 most-starred repositories in servo.**

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 stargaz
ers_count DESC, name LIMIT 10
[u'concat(concat("(", stargazers_count, ") ", name), ": ", description)']
[u'(10990) servo: The Servo Browser Engine']
[u'(981) webrender: A GPU-based renderer for the web']
[u'(583) html5ever: High-performance browser-grade HTML5 parser']
```

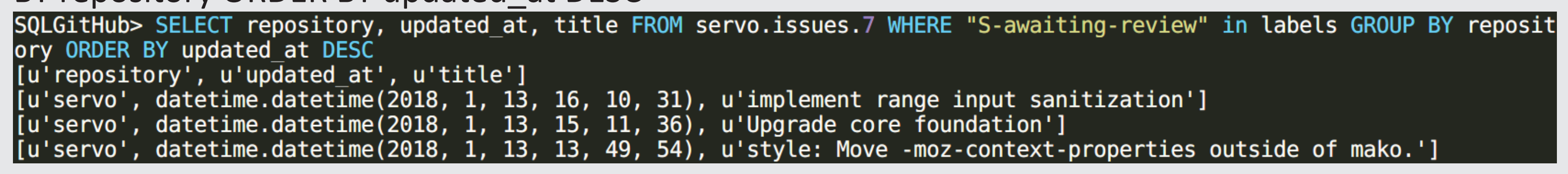
4. **Get top 10 contributors in servo for the past month (30 days) based on number of commits.**

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', 191]
[u'emilio', 60]
[u'SimonSapin', 33]
[u'glennw', 30]
```

5. **Get the issues that are still waiting for review in servo for the past week (7 days).**

SELECT repository, updated_at, title FROM servo.issues.7 WHERE "S-awaiting-review" in labels GROUP BY repository ORDER BY updated_at DESC


```
SQLGitHub> SELECT repository, updated_at, title FROM servo.issues.7 WHERE "S-awaiting-review" in labels GROUP BY reposit
ory ORDER BY updated_at DESC
[u'repository', u'updated_at', u'title']
[u'servo', datetime.datetime(2018, 1, 13, 16, 10, 31), u'implement range input sanitization']
[u'servo', datetime.datetime(2018, 1, 13, 15, 11, 36), u'Upgrade core foundation']
[u'servo', datetime.datetime(2018, 1, 13, 13, 49, 54), u'style: Move -moz-context-properties outside of mako.']
```

Specification

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} [ASC | DESC], ...]
    [LIMIT row_count]
```

SUPPORTED FUNCTIONS (SELECTED)

String Functions:
"concat", "concat_ws", "find_in_set", "insert", "instr", "length", "locate", "lcase", "lower", "left", "mid", "repeat", "right", "replace", "strcmp", "substr", "substring", "ucase", "upper"
Numeric Functions:
"avg", "count", "max", "min", "sum", "abs", "ceil", "ceiling", "exp", "floor", "greatest", "least", "ln", "log", "pow", "power", "sign", "sqrt"
Date & Advanced Functions:
"curdate", "current_date", "current_time", "current_timestamp", "curtime", "localtime", "localtimestamp", "now", "bin"

SUPPORTED OUTPUT FORMAT

Python, HTML, CSV

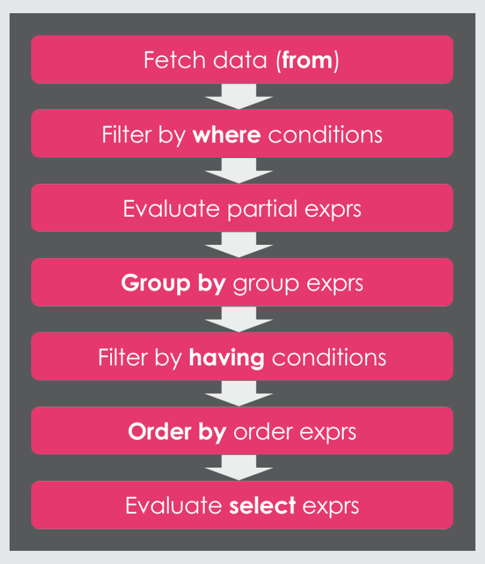
Method

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

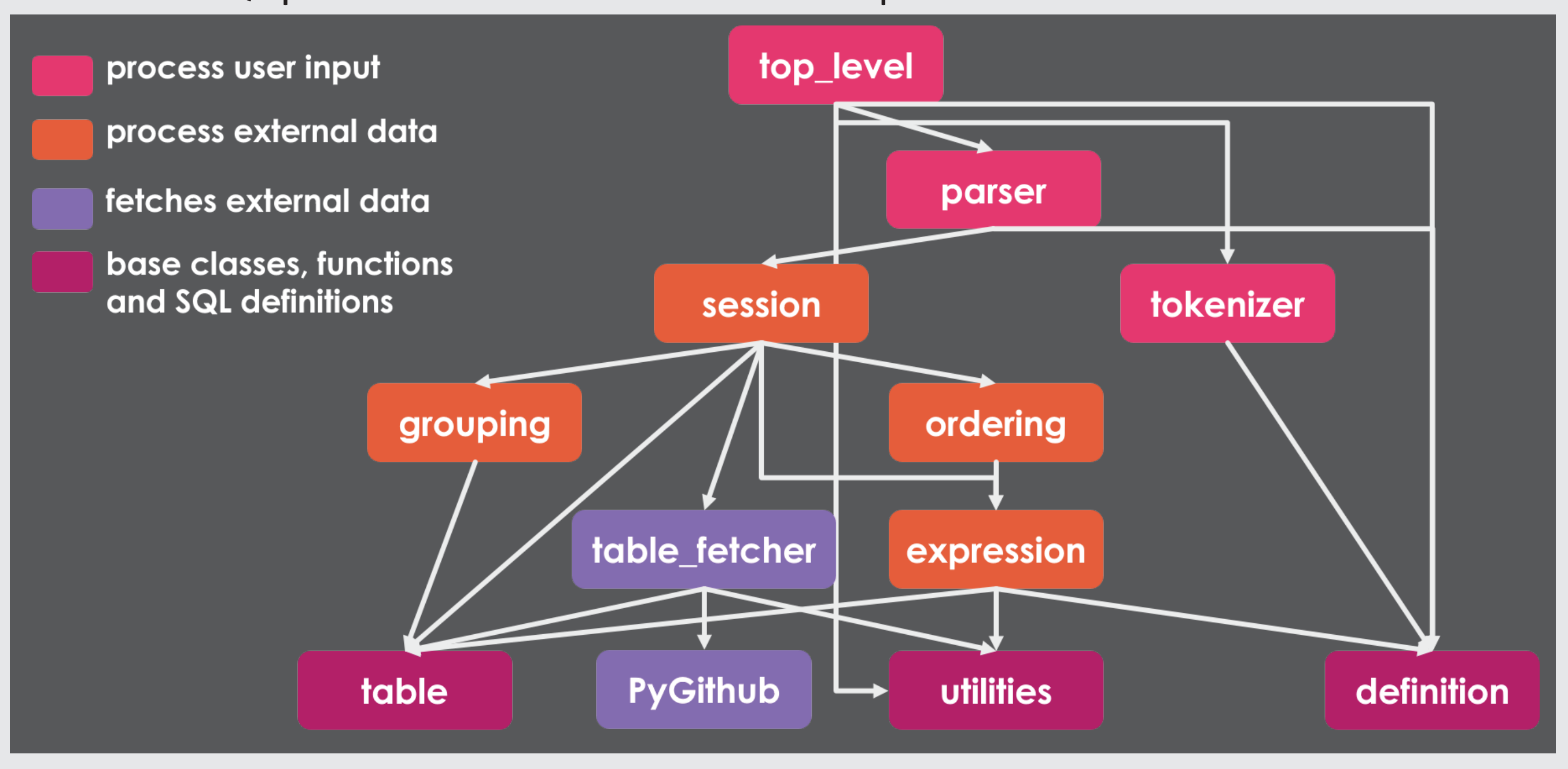
FLOW

1. Fetch data with required fields from GitHub API
2. Evaluate where conditions and filter fetched data
3. Evaluate group exprs and other "field" exprs
4. Generate table groups by values of group exprs
5. Evaluate having conditions and filter tables
6. Sort within and between tables
7. Evaluate select exprs



ARCHITECTURE

A custom SQL parser written from scratch & a patched GitHub API client



CHALLENGES

- No local indexed store

This means we need to perform complex extraction and manipulation on both input and fetched data

- Parsing SQL is tricky and tedious

Comma-separated fields, strings and escape characters make things especially tedious

- Evaluating expressions is incredibly hard
  - There are regular functions which look at values within the same row and aggregate functions which need to look at values within the same column
  - Other considerations include nested functions and operator precedence

Results

PROS

- Useful for GitHub organization owners: Shortened the time needed for maintenance tasks by 120x
- An easier-to-use, better and more versatile API frontend (Compared to RESTful and GraphQL)
- High customizability thanks to good compatibility with SQL (MySQL)
- Modularized, can be reused/integrated as a library
- Better efficiency and security if integrated on servers

CONS

- Slow (information is retrieved over the internet + RESTful API)

Migrate to GitHub API V4 (GraphQL backend) would help, but this would sacrifice functionalities as the new API is far from backwards-compatible to GitHub API V3.

Future Directions

- Improve it to production quality by refactoring code base and writing tests
- Promote to more GitHub organization owners
- Extend to end users not just organizations
- Implement an experimental GraphQL backend (GitHub API v4)
- Implement this concept directly on an API 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