Выпускная квалификационная работа

Аналитика исходного кода ClickHouse с помощью ClickHouse

Ширин Никита, гр. 155
Научный руководитель: Руководитель группы разработки ClickHouse в Яндексе,
Миловидов Алексей Николаевич

Предметная область

- Извлечение различных статистик из Git-репозитория
- Использование аналитической СУБД для хранения данных о репозитории

Актуальность задачи

- Извлеченные из данных репозитория статистики полезны для аналитиков, исследующих различные аспекты работы над проектом, и разработчикам
- Существующие решения не обладают достаточной гибкостью и удобством использования
- Желающим извлечь нетривиальную статистику приходится писать специальные программы

Цель и задачи дипломной работы

Цель: получить инструмент, позволяющий извлекать нетривиальные статистики посредством SQL-запросов

Задачи:

- Обзор существующих решений - GitHub API и GitStats
- Разработать архитектуру базы данных под управлением СУБД ClickHouse
- Выбрать архитектуру и реализовать программу, наполняющую базу данных
- Выбрать архитектуру и реализовать web-приложение, обращающееся к базе данных
- Провести эксперименты подсчета нетривиальных статистик на примере репозитория ClickHouse

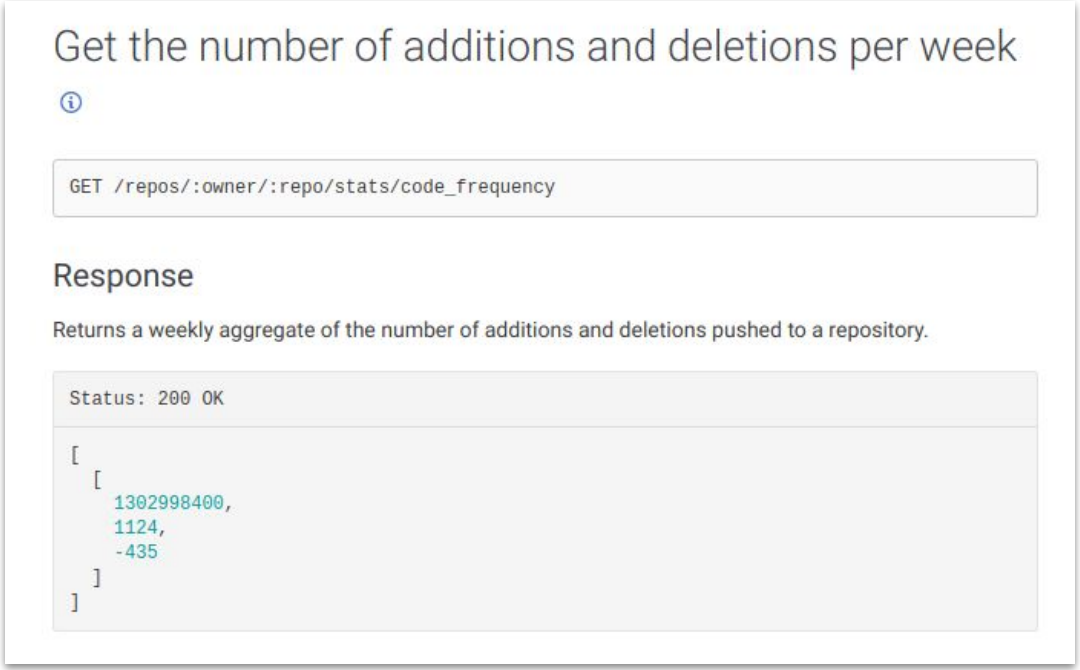
Формальная постановка

Разработка приложения, получающего на вход git-репозиторий, сохраняющего данные о репозитории в базу данных под управлением СУБД ClickHouse и предоставляющего пользователю интерфейс, позволяющий задавать SQL-запросы к базе данных для подсчета нетривиальных статистик.

Обзор существующих решений

GitHub API

- REST API
- Кэширование ответов
- Небольшое число поддерживаемых запросов



Get the number of additions and deletions per week

```
GET /repos/:owner/:repo/stats/code_frequency
```

Response

Returns a weekly aggregate of the number of additions and deletions pushed to a repository.

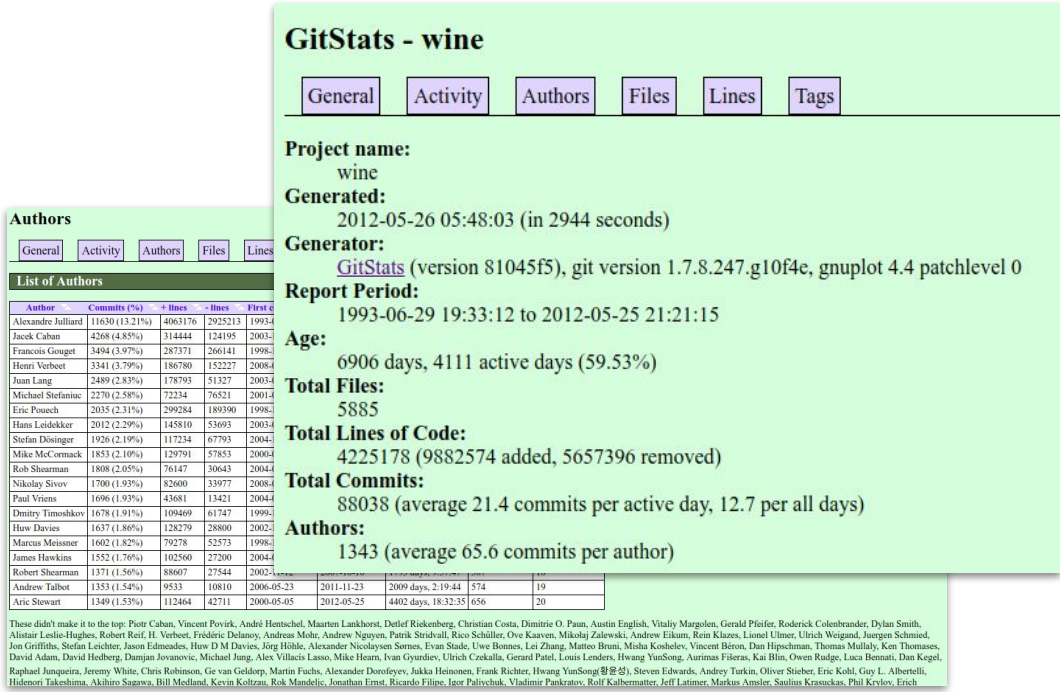
```
Status: 200 OK
```

```
[
  [
    1302998400,
    1124,
    -435
  ]
]
```

Обзор существующих решений

GitStats

- Приложение на python
- HTML-отчет
- Для расширения функционала требуется модификация исходного кода



GitStats - wine

General | Activity | Authors | Files | Lines | Tags

Project name: wine
Generated: 2012-05-26 05:48:03 (in 2944 seconds)
Generator: GitStats (version 81045f5), git version 1.7.8.247.g10f4e, gnuplot 4.4 patchlevel 0
Report Period: 1993-06-29 19:33:12 to 2012-05-25 21:21:15
Age: 6906 days, 4111 active days (59.53%)
Total Files: 5885
Total Lines of Code: 4225178 (9882574 added, 5657396 removed)
Total Commits: 88038 (average 21.4 commits per active day, 12.7 per all days)
Authors: 1343 (average 65.6 commits per author)

Authors

General | Activity | Authors | Files | Lines

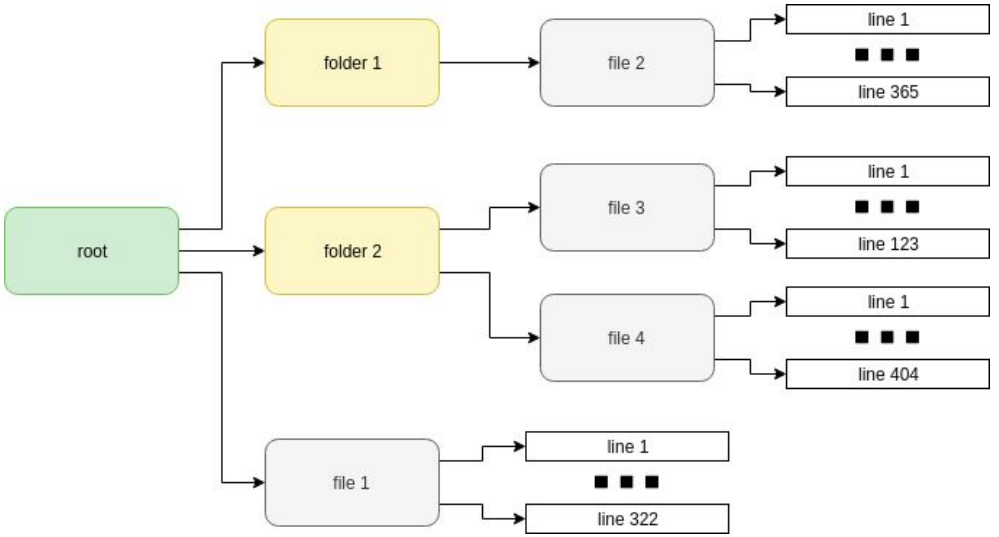
List of Authors

Author	Commits (%)	+ lines	- lines	First c			
Alexandre Julliard	11630 (13.21%)	4063176	2925213	1993-0			
Jacek Caban	4268 (4.85%)	314444	124195	2003-1			
Francois Gouget	3494 (3.97%)	287371	266141	1998-1			
Henri Verbeet	3341 (3.79%)	186780	152227	2008-0			
Juan Lang	2489 (2.83%)	178793	51327	2003-0			
Michael Stefaniuc	2270 (2.58%)	72234	76521	2001-0			
Eric Pouech	2035 (2.31%)	299284	189390	1998-1			
Hans Leidekker	2012 (2.29%)	145810	53693	2003-0			
Stefan Dösinger	1926 (2.19%)	117234	67793	2004-0			
Mike McCormack	1853 (2.10%)	129791	57853	2000-0			
Rob Shearman	1808 (2.05%)	76147	30643	2004-0			
Nikolay Sivov	1700 (1.93%)	82600	33977	2008-0			
Paul Vriens	1696 (1.93%)	43681	13421	2004-0			
Dmitry Timoshkov	1678 (1.91%)	109469	61747	1999-1			
Huw Davies	1637 (1.86%)	128279	28800	2002-0			
Marcus Meissner	1602 (1.82%)	79278	52573	1998-0			
James Hawkins	1552 (1.76%)	102560	27200	2004-0			
Robert Shearman	1371 (1.56%)	88607	27544	2002-1			
Andrew Talbot	1353 (1.54%)	9533	10810	2006-05-23	2011-11-23	2009 days, 2:19:44	574	19
Aric Stewart	1349 (1.53%)	112464	42711	2000-05-05	2012-05-25	4402 days, 18:32:35	656	20

These didn't make it to the top: Piotr Caban, Vincent Povirk, André Hentschel, Maarten Lankhorst, Detlef Riekenberg, Christian Costa, Dimitrie O. Paun, Austin English, Vitaliy Margolen, Gerald Pfeifer, Roderick Colenbrander, Dylan Smith, Alistair Leslie-Hughes, Robert Reif, H. Verbeet, Frédéric Delanoy, Andreas Mohr, Andrew Nguyen, Patrik Stridvall, Rico Schüller, Ove Kaaven, Mikołaj Zalewski, Andrew Eikum, Rein Klazes, Lionel Ulmer, Ulrich Weigand, Juergen Schmied, Jon Griffiths, Stefan Leichter, Jason Edmeades, Huw D M Davies, Jörg Höhle, Alexander Nicolaysen Sørnes, Evan Stade, Uwe Bonnes, Lei Zhang, Matteo Bruni, Misha Koshelev, Vincent Béron, Dan Hipschman, Thomas Mullaly, Ken Thomases, David Adam, David Hedberg, Damjan Jovanovic, Michael Jung, Alex Villacís Lasso, Mike Hearn, Ivan Gyurdiev, Ulrich Czekalla, Gerard Patel, Louis Lenders, Hwang YunSong, Aurimas Fišeras, Kai Blin, Owen Rudge, Luca Bennati, Dan Kegel, Raphael Junqueira, Jeremy White, Chris Robinson, Ge van Geldorp, Martin Fuchs, Alexander Dorofeyev, Jukka Heinonen, Frank Richter, Hwang YunSong(황윤성), Steven Edwards, Andrey Turkin, Oliver Stieber, Eric Kohl, Guy L. Albertelli, Hidenori Takeshima, Akihiro Sagawa, Bill Medland, Kevin Koltzau, Rok Mandeljc, Jonathan Ernst, Ricardo Filipe, Igor Paliychuk, Vladimir Pankratov, Rolf Kalbermatter, Jeff Latimer, Markus Amsler, Saulius Krasuckas, Phil Krylov, Erich

Архитектура приложения
База данных
Первый вариант

- Независимое заполнение для различных коммитов
- Множество операций с жестким диском
- Оптимизация для хранения



Структура таблицы: commit_hash (FixedString(20)), commit_date (Date), commit_time (DateTime), commit_message (String), author_name (String), author_email (String), file_name (String), file_path (String), file_extension (String), line_num (UInt32), line (String)

Архитектура приложения
База данных
Второй вариант

- Имитация репозитория python-структурами
- Хранение “blame”
- Две таблицы.
- Оптимизация для скорости запросов

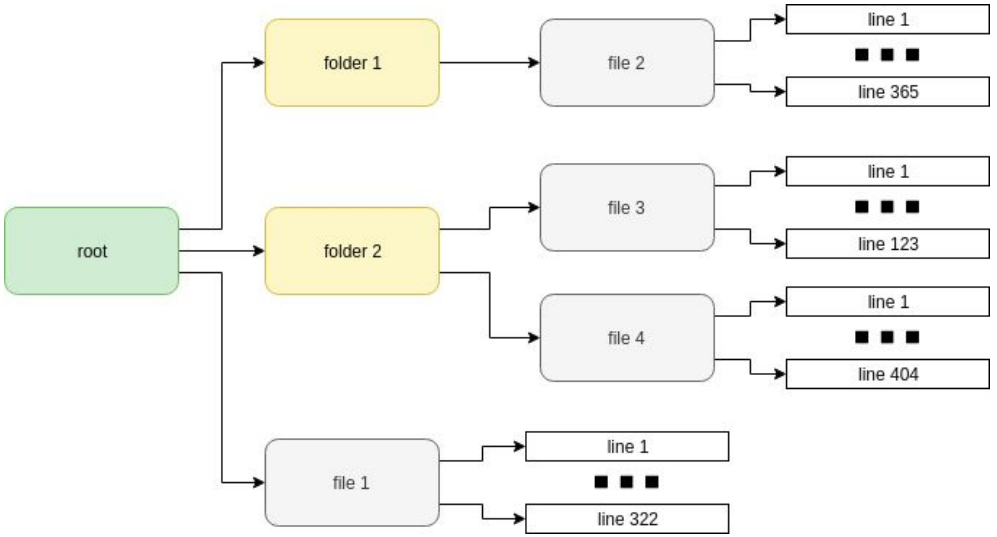


Таблица lines: commit_hash (FixedString(40)), file_name (String), file_path (String), file_id (UUID), line_num (UInt32), line (String), last_change (FixedString(40))

Таблица commits: commit_hash (FixedString(40)), commit_time (DateTime), commit_message (String), author_name (String), author_email (String)

Архитектура приложения
База данных
Результаты для репозитория ClickHouse

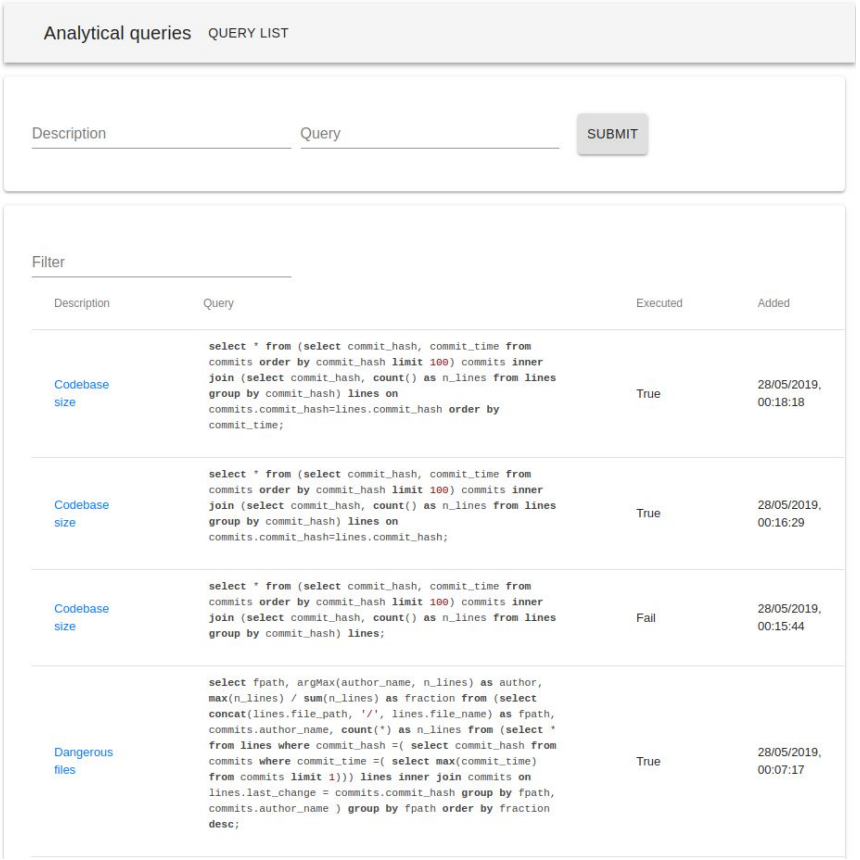
Вариант	Первый вариант	Второй вариант
Размер таблицы	600 Гб	200 Гб
Время заполнения	10 стуок	6 суток
Время выполнения запросов	1000 секунд	Сильно быстрее

Характеристики компьютера: Intel Core i7-7700, 16 Гб RAM и накопитель типа HDD

Код написан на Python3 с использованием библиотек git, pydriller, clickhouse-driver

Web-приложение

- Серверная часть написана на языке Python3 с использованием Flask
- PostgreSQL для хранения истории SQL-запросов
- Отдельная программа выполняет невыполненные запросы
- Графический интерфейс написан на TypeScript с использованием React



Эксперименты

Analytical queries QUERY LIST

Commits by int weekday

```
select weekday, count() as n_commits from (select  cast(formatDateTime(commit_time, '%w') as Int8) as weekday from commits) group by weekday order by weekday;
```

Elapsed 1 secs.

X Y GET GRAPH

weekday (Int8)	n_commits (UInt64)
0	1753
1	4163
2	3851
3	4363
4	4215
5	4307
6	1648

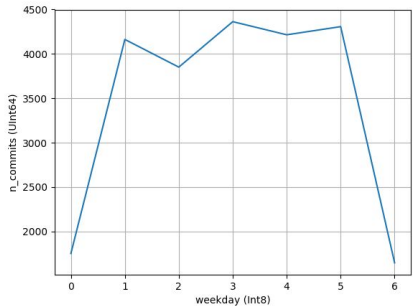
Analytical queries QUERY LIST

Commits by int weekday

select weekday, count() as n_commits from (select cast(formatDateTime(commit_time, '%w') as Int8) as weekday from commits) group by weekday order by weekday;

Elapsed 1 secs.

X weekday Y n_commits GET GRAPH



Эксперименты

Analytical queries QUERY LIST

Codebase size

```
select * from (select commit_hash, commit_time from commits order by commit_hash limit 100) commits inner join (select commit_hash, count() as n_lines from lines group by commit_hash) lines on commits.commit_hash=lines.commit_hash order by commit_time;
```

Elapsed 46 secs.

X Y GET GRAPH

commit_hash (FixedString(40))	commit_time (DateTime)	lines.commit_hash (FixedString(40))	n_lines (UInt64)
00f23ee493b547c2f62922491b404ed61aceab4c	2012-03-05 08:29:16	00f23ee493b547c2f62922491b404ed61aceab4c	34337
002d6a617850a2e1ca3c154df94403d52b83f66e	2013-01-07 12:27:39	002d6a617850a2e1ca3c154df94403d52b83f66e	59949
00e0307de17c74f3ead7cc948b43e70dd18f5925	2013-03-14 17:04:50	00e0307de17c74f3ead7cc948b43e70dd18f5925	64864
00f2cceac0218e3b08a60b17182e2d6f0fa7a050	2013-07-12 21:47:19	00f2cceac0218e3b08a60b17182e2d6f0fa7a050	70912
00b10d30df41b5b1f79ad65e607c30ba9e712293	2014-03-11 11:32:47	00b10d30df41b5b1f79ad65e607c30ba9e712293	93943
00d9c285719981a7fcd1dc5ee7d820cb59674771	2014-03-13 21:44:00	00d9c285719981a7fcd1dc5ee7d820cb59674771	94071
003c7b30cdf118e4b079134273c4762929bb7274	2014-03-14 19:42:30	003c7b30cdf118e4b079134273c4762929bb7274	95670
00d5cfbe45acdd80ae90f9ef687059b644dac206	2014-04-03 19:57:36	00d5cfbe45acdd80ae90f9ef687059b644dac206	95997
00074ffaa5d2282e824efec2fa0882dade66f717	2014-07-03 15:22:12	00074ffaa5d2282e824efec2fa0882dade66f717	110977

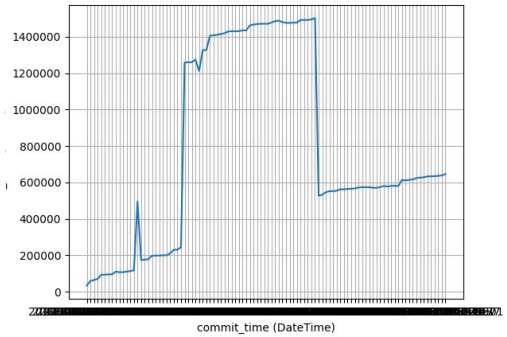
Analytical queries QUERY LIST

Codebase size

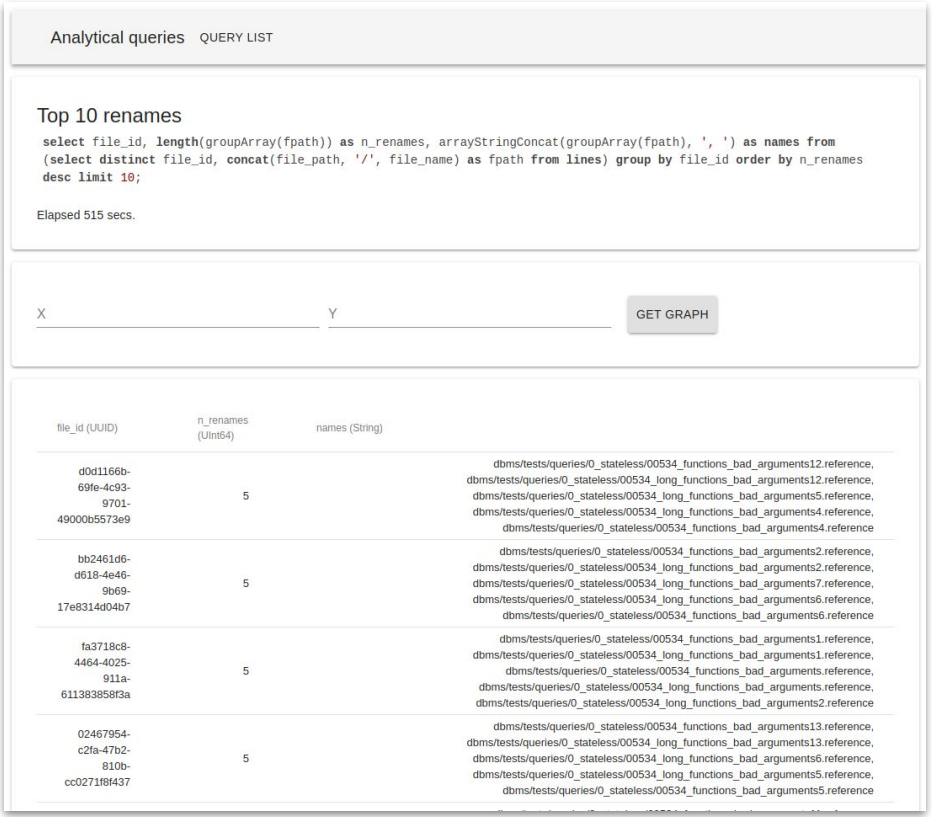
```
select * from (select commit_hash, commit_time from commits order by commit_hash limit 100) commits inner join (select commit_hash, count() as n_lines from lines group by commit_hash) lines on commits.commit_hash=lines.commit_hash order by commit_time;
```

Elapsed 46 secs.

X time Y n_lines GET GRAPH



Эксперименты



Analytical queries QUERY LIST

Top 10 renames

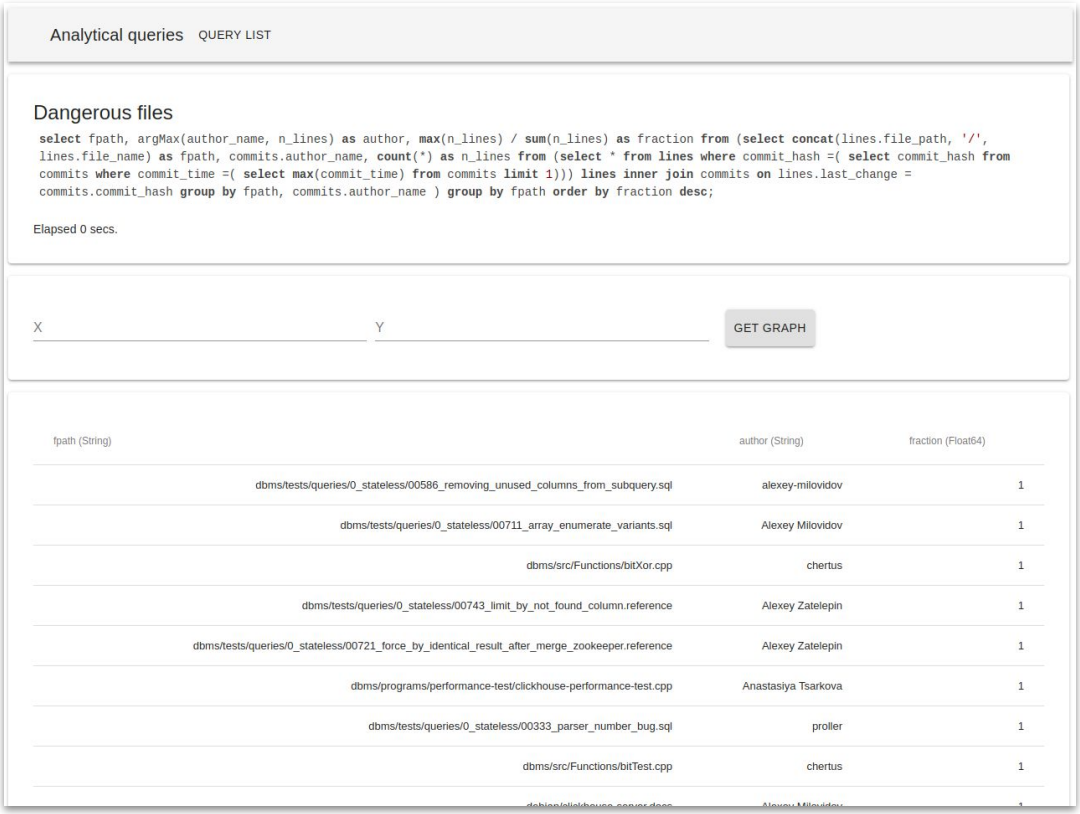
```
select file_id, length(groupArray(fpath)) as n_renames, arrayStringConcat(groupArray(fpath), ', ') as names from
(select distinct file_id, concat(file_path, '/', file_name) as fpath from lines) group by file_id order by n_renames
desc limit 10;
```

Elapsed 515 secs.

X Y GET GRAPH

file_id (UUID)	n_renames (UInt64)	names (String)
d0d1166b-69fe-4c93-9701-49000b5573e9	5	dbms/tests/queries/0_stateless/00534_functions_bad_arguments12.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments12.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments5.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments4.reference, dbms/tests/queries/0_stateless/00534_functions_bad_arguments4.reference
bb2461d6-d618-4e46-9b69-17e8314d04b7	5	dbms/tests/queries/0_stateless/00534_functions_bad_arguments2.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments2.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments7.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments6.reference, dbms/tests/queries/0_stateless/00534_functions_bad_arguments6.reference
fa3718c8-4464-4025-911a-611383858f3a	5	dbms/tests/queries/0_stateless/00534_functions_bad_arguments1.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments1.reference, dbms/tests/queries/0_stateless/00534_functions_bad_arguments.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments2.reference
02467954-c2fa-47b2-810b-cc0271f8f437	5	dbms/tests/queries/0_stateless/00534_functions_bad_arguments13.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments13.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments6.reference, dbms/tests/queries/0_stateless/00534_long_functions_bad_arguments5.reference, dbms/tests/queries/0_stateless/00534_functions_bad_arguments5.reference

Эксперименты



Analytical queries QUERY LIST

Dangerous files

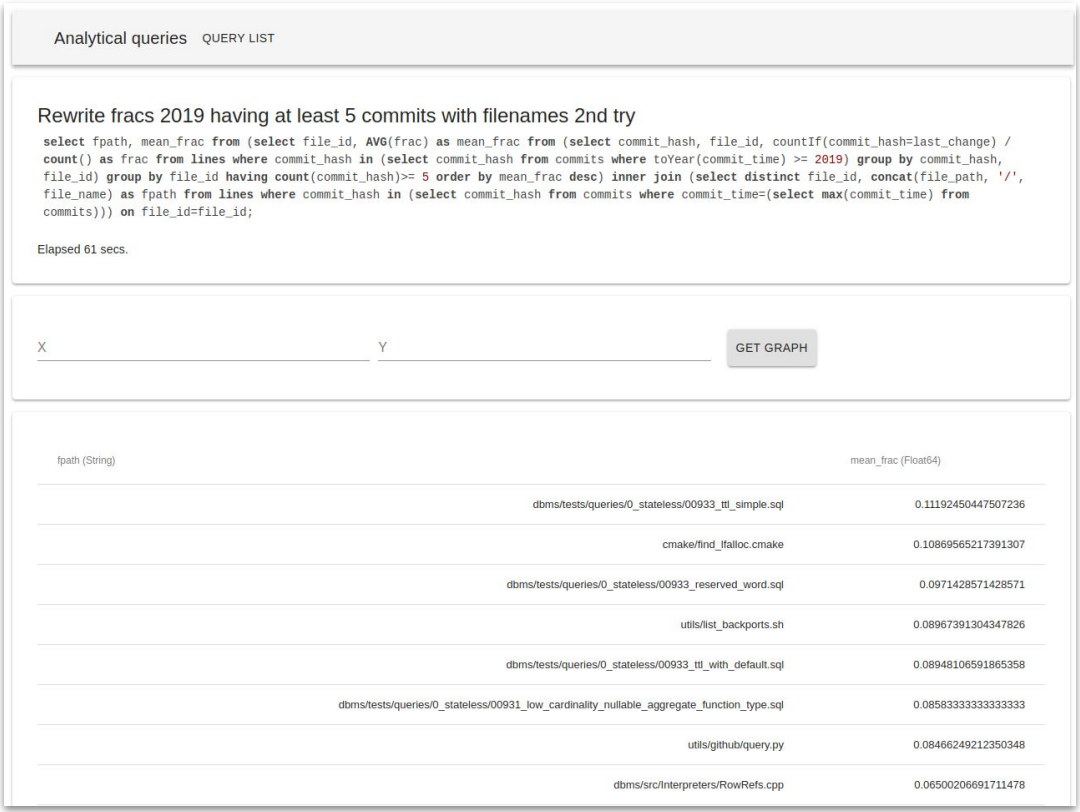
```
select fpath, argMax(author_name, n_lines) as author, max(n_lines) / sum(n_lines) as fraction from (select concat(lines.file_path, '/',
lines.file_name) as fpath, commits.author_name, count(*) as n_lines from (select * from lines where commit_hash =( select commit_hash from
commits where commit_time =( select max(commit_time) from commits limit 1))) lines inner join commits on lines.last_change =
commits.commit_hash group by fpath, commits.author_name ) group by fpath order by fraction desc;
```

Elapsed 0 secs.

X Y GET GRAPH

fpath (String)	author (String)	fraction (Float64)
dbms/tests/queries/0_stateless/00586_removing_unused_columns_from_subquery.sql	alexey-milovidov	1
dbms/tests/queries/0_stateless/00711_array_enumerate_variants.sql	Alexey Milovidov	1
dbms/src/Functions/bitXor.cpp	chertus	1
dbms/tests/queries/0_stateless/00743_limit_by_not_found_column.reference	Alexey Zatelepin	1
dbms/tests/queries/0_stateless/00721_force_by_identical_result_after_merge_zookeeper.reference	Alexey Zatelepin	1
dbms/programs/performance-test/clickhouse-performance-test.cpp	Anastasiya Tsarkova	1
dbms/tests/queries/0_stateless/00333_parser_number_bug.sql	proller	1
dbms/src/Functions/bitTest.cpp	chertus	1
debian/clickhouse-server.docs	Alexey Milovidov	1

Эксперименты



Analytical queries QUERY LIST

Rewrite fracs 2019 having at least 5 commits with filenames 2nd try

```
select fpath, mean_frac from (select file_id, AVG(frac) as mean_frac from (select commit_hash, file_id, countIf(commit_hash=last_change) /
count() as frac from lines where commit_hash in (select commit_hash from commits where toYear(commit_time) >= 2019) group by commit_hash,
file_id) group by file_id having count(commit_hash)>= 5 order by mean_frac desc) inner join (select distinct file_id, concat(file_path, '/',
file_name) as fpath from lines where commit_hash in (select commit_hash from commits where commit_time=(select max(commit_time) from
commits))) on file_id=file_id;
```

Elapsed 61 secs.

X Y GET GRAPH

fpath (String)	mean_frac (Float64)
dbms/tests/queries/0_stateless/00933_ttl_simple.sql	0.11192450447507236
cmake/find_lfalloc.cmake	0.10869565217391307
dbms/tests/queries/0_stateless/00933_reserved_word.sql	0.097142857142857
utils/list_backports.sh	0.08967391304347826
dbms/tests/queries/0_stateless/00933_ttl_with_default.sql	0.08948106591865358
dbms/tests/queries/0_stateless/00931_low_cardinality_nullable_aggregate_function_type.sql	0.08583333333333333
utils/github/query.py	0.08466249212350348
dbms/src/Interpreters/RowRefs.cpp	0.06500206691711478

Результаты работы

- Проведен обзор существующих решений - GitHub API и GitStats
- Выбранная архитектура базы данных поддерживает функционал GitHub API и GitStats, а также предоставляет возможность считать пользовательские статистики посредством SQL-запросов
- Программа, наполняющая базу данных написана на языке Python
- Web-приложение написано на языке TypeScript с React, серверная часть - на языке Python с Flask
- В экспериментах продемонстрированы нетривиальные запросы, реализуемые с помощью приложения, на примере репозитория ClickHouse

Спасибо за внимание