Умные алгоритмы обработки строк в ClickHouse

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Яндекс

Данила Кутенин

HighLoad++ Siberia 2019

Профессиональная конференция для разработчиков высоконагруженных систем

Строки

- Любая последовательность байт в алфавите Σ. “*abacaba*”
- UTF-8-закодированные строки. “привет”, “嘿”
- Учёт регистра. “NeVeR gOnNa GiVe YoU uP”

Задачи

- Поиск подстроки, по регулярному выражению
- Поиск многих подстрок, многих регулярных выражений
- Приближённый поиск
- Обработка UTF-8 строк
- Форматирование строк

Строки в ClickHouse

- **ColumnString. Chars + Offsets**

Chars = “string\0anotherone\0andanotherone\0”
Offsets = [7, 18, 32]

- Безопасное чтение 16-байтового регистра

Строки в ClickHouse

- **ColumnFixedString. Chars + FixedConstant N.**

Chars = “yandexgoogleamazon”

N = 6

- Безопасное чтение 16-байтового регистра
- Есть **ColumnConst** с типом String
- Ещё **LowCardinality** (но это тема для отдельного разговора)

Поиск

```
Samples: 651K of event 'cycles:pp', Event count (approx.): 687435178877
Overhead  Shared Object                 Symbol
  32.77%  clickhouse                    [.] LZ4_decompress_fast
  17.70%  clickhouse                    [.] DB::deserializeBinarySSE2<4>
  16.04%  clickhouse                    [.] DB::VolnitskyBase<DB::VolnitskyImpl<true, true> >::search
   7.33%  [kernel]                      [k] copy_user_enhanced_fast_string
   2.86%  clickhouse                    [.] DB::deserializeBinarySSE2<1>
   2.79%  clickhouse                    [.] CityHash_v1_0_2::CityHash128WithSeed
   2.05%  clickhouse                    [.] memcpy
   0.94%  [kernel]                      [k] put_page
   0.89%  clickhouse                    [.] tcmalloc::CentralFreeList::RemoveRange
   0.82%  [kernel]                      [k] find_get_entry
   0.76%  clickhouse                    [.] DB::deserializeBinarySSE2<3>
   0.67%  clickhouse                    [.] DB::deserializeBinarySSE2<2>
   0.61%  clickhouse                    [.] MemoryTracker::alloc
   0.55%  libc-2.19.so                  [.] memset
   0.54%  clickhouse                    [.] tcmalloc::CentralFreeList::InsertRange
   0.49%  clickhouse                    [.] CurrentMemoryTracker::free
```

Поиск. Определения.

haystack (англ. сено) — строка, в которой мы ищем.

needle (англ. иголка) — строка (или регулярное выражение), по которой мы ищем.

Поиск. Алгоритмы

Один needle, один haystack

1. Knuth-Morris-Pratt
2. Boyer-Moore
3. Boyer-Moore-Horspool
4. BNDM (Backward Nondeterministic Dawg)
5. Two-way (memmem)
6. Rabin-Karp
7. Поиск по индексу
8. ...

Много needle, один haystack

1. Aho-Corasick
2. Поиск по индексу
3. Rabin-Karp(?)

Что использует ClickHouse?

Ничего из вышеперечисленного
Но мы честно всё попробовали

Поиск. Алгоритмы

Volnitsky algorithm

Source:

http://volnitsky.com/project/str_search/index.html

```
SELECT count()
FROM hits_100m_single
WHERE URL LIKE '%yandex%'
SETTINGS max_threads = 1

┌─count()─┐
│ 18401331 │
└─────────┘

1 rows in set. Elapsed: 5.337 sec. Processed 100.00 million ro
ws, 9.38 GB (18.74 million rows/s., 1.76 GB/s.)
```

Поиск. Алгоритмы

haystack — **abacabaaca** и needle — **aaca**. Хэш-таблица {aa → 0, ac → 1, ca → 2}, Step = 3.

```
0 1 2 3 4 5 6 7 8 9
a b a c a b a a c a
    ^ - курсор здесь

0 1 2 3 4 5 6 7 8 9
a b a c a b a a c a
  a a c a

0 1 2 3 4 5 6 7 8 9
a b a c a b a a c a
        ^ - курсор здесь

0 1 2 3 4 5 6 7 8 9
a b a c a b a a c a
              ^ - курсор здесь
```

```
0 1 2 3 4 5 6 7 8 9
a b a c a b a a c a
          a a c a

Нашли совпадение
```

Реализация в ClickHouse:

dbms/src/Common/Volnitsky.h

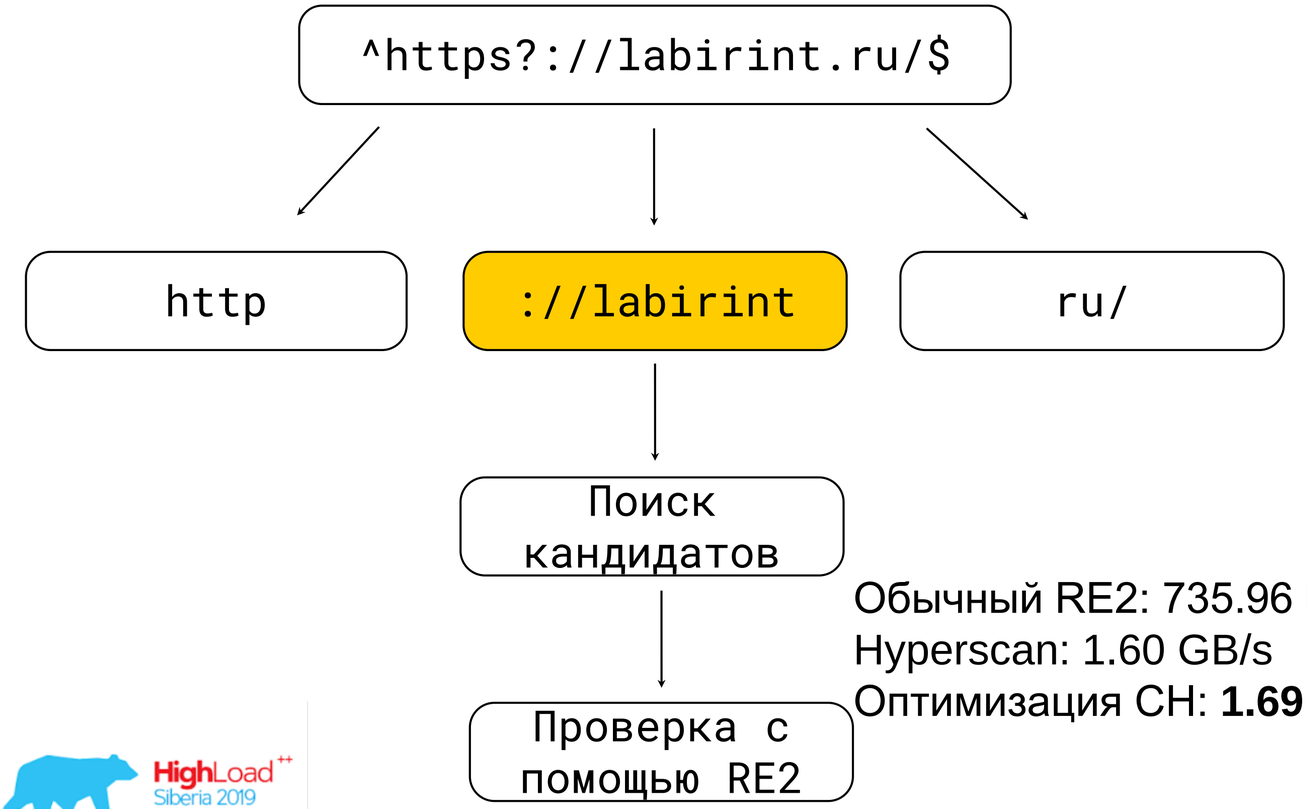
```
class VolnitskyBase;
```

Поиск. Алгоритмы. Mb/s. Больше лучше

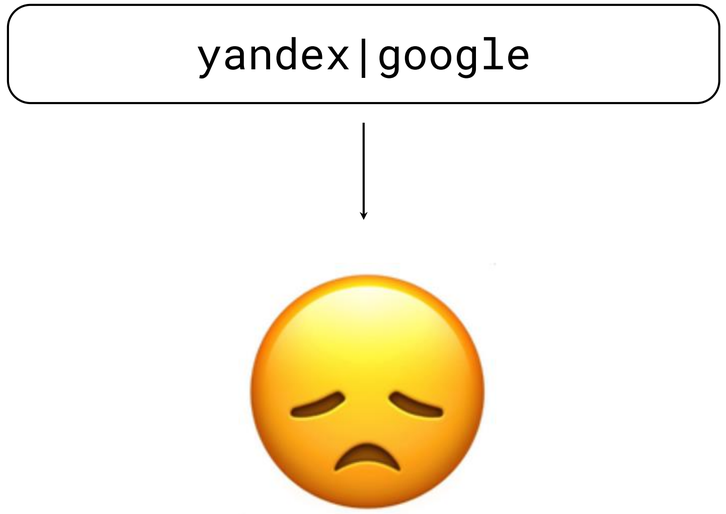
	memmem (Two-way)	Boyer Moore	Boyer Moore Horspool	Knuth Morris Pratt	Volnitsky
SELECT sum(position(URL, 'google')) FROM hits_100m_single settings max_threads=1	655	865	920	600	1660
SELECT count() FROM hits_100m_single WHERE URL LIKE '%metrika%' settings max_threads=1	710	940	930	653	1800
SELECT sum(position(URL, 'yandex')) FROM hits_100m_single settings max_threads=1	637	780	790	580	1560
SELECT SearchPhrase, any(URL), any(Title), count() AS c, uniq(UserID) FROM hits_100m_single WHERE (Title LIKE '%Яндекс%') GROUP BY SearchPhrase ORDER BY c DESC LIMIT 10 settings max_threads=1	578	600	687	540	880

Поиск. Где используется алгоритм

- **position(hstk, ndl), -UTF8, -CaseInsensitive, -CaseInsensitiveUTF8**
- **LIKE ‘%somestring%’**
- **Оптимизация поиска регулярных выражений, в т.ч. LIKE**



Обычный RE2: 735.96 MB/s
Hyperscan: 1.60 GB/s
Оптимизация CH: **1.69 GB/s**



Но мы придумали решение и тут!

Поиск. Алгоритмы

MultiVolnitsky algorithm

Реализация в ClickHouse:

dbms/src/Common/Volnitsky.h

class MultiVolnitskyBase;

```
SELECT sum(multiSearchAny(URL, ['yandex', 'google', 'yahoo']))
FROM hits_100m_single
SETTINGS max_threads = 1

┌─sum(multiSearchAny(URL, ['yandex', 'google', 'yahoo']))─┐
│                                                 18441259 │
└──────────────────────────────────────────────────────────┘

1 rows in set. Elapsed: 7.747 sec. Processed 100.00 million ro
ws, 9.38 GB (12.91 million rows/s., 1.21 GB/s.)
```

Поиск. Алгоритмы

Пусть будет строка haystack – **`he loves his herbal shelter`** и needles равны **`[she, his, her]`.**

Хэш-таблица {sh → {0, 0}, he → {{1, 0}, {0, 2}}, hi → {0, 1}, is → {1, 1}, er → {1, 2}}. Step = 2.

```
he_loves_his_herbal_shelter
 ^ курсор тут
```

```
he_loves_his_herbal_shelter
   ^ курсор тут
```

```
he_loves_his_herbal_shelter
     ^ курсор тут
```

```
he_loves_his_herbal_shelter
      ^ курсор тут
```

```
he_loves_his_herbal_shelter
         ^ курсор тут
```

```
he_loves_his_herbal_shelter
          ^ курсор тут
```

```
he_loves_his_herbal_shelter
        she
```

```
he_loves_his_herbal_shelter
         her
```

```
he_loves_his_herbal_shelter
               ^ курсор тут
```

```
he_loves_his_herbal_shelter
                ^ курсор
тут
```

Поиск. Алгоритмы

Пусть будет строка haystack – **`he loves his herbal shelter`** и needles равны **`[she, his, her].`**

Хэш-таблица {sh → {0, 0}, he → {{1, 0}, {0, 2}}, hi → {0, 1}, is → {1, 1}, er → {1, 2}}. Step = 2.

```
he_loves_his_herbal_shelter
                    ^ курсор тут
```

```
he_loves_his_herbal_shelter
                      ^ курсор тут
```

```
he_loves_his_herbal_shelter
                     she
```

```
he_loves_his_herbal_shelter
                     her
```

```
he_loves_his_herbal_shelter
                                ^ курсор тут
```

```
he_loves_his_herbal_shelter
                                  ^ курсор тут
```

```
he_loves_his_herbal_shelter
                                   her
```

Сравнение алгоритмов для поиска

	Multi Volnitsky	Volnitsky n раз	Aho Corasick	RE2	Hyperscan
`'yandex', 'google'`	1.49 GB/s	1.18 GB/s	851.32 MB/s	329.39 MB/s	788.21 MB/s
`'yandex', 'google', 'yahoo', 'pikabu'`	1.27 GB/s	762.13 MB/s	780.65 MB/s	303.58 MB/s	748.03 MB/s
`'yandex', 'google', 'http'`	1.69 GB/s	1.00 GB/s	1.19 GB/s	773.28 MB/s	626.18 MB/s
`'Honda', 'Хонд', 'HONDA'`	900.07 MB/s	741.97 MB/s	730.20 MB/s	168.26 MB/s	814.30 MB/s
`'yandex', 'google', 'facebook', 'wikipedia', 'reddit'`	1.22 GB/s	677.39 MB/s	813.57 MB/s	267.00 MB/s	757.16 MB/s
`'news.ngs.ru', 'she.ngs.ru', 'afisha.ngs.ru', 'business.ngs.ru', '//ngs.ru/', '//m.ngs.ru/'`	1.35 GB/s	681.91 MB/s	890.13 MB/s	220.24 MB/s	850.83 MB/s

Сравнение алгоритмов для поиска

	Multi Volnitsky	Volnitsky n раз	Aho Corasick	RE2	Hyperscan
'newFlat=YES', 'newbuilding', 'siteId=', 'novostrojka', 'nb.phone.show', 'nb.show', 'pik/'	906.08 MB/s	584.09 MB/s	737.94 MB/s	252.77 MB/s	724.39 MB/s
'kvartiry', 'nedvizhimost', 'kommercheskaya_nedvizhimost', 'garazhi_i_mashinomesta', 'doma_dachi_kottedzhi', 'zemelnye_uchastki', 'komnaty', 'nedvizhimost_za_rubezhom'	1.05 GB/s	644.00 MB/s	757.43 MB/s	250.75 MB/s	775.68 MB/s
'ут', 'утк', 'утко', 'утконос', 'enrjyjc', 'utkonos', 'enrjyjc', 'www', 'http', 'enrfyjc', 'гелщтщы'	500.44 MB/s	254.45 MB/s	453.21 MB/s	134.45 MB/s	532.78 MB/s
'бэбиблок', 'бэбиблог', 'бебиблок', 'бебиблог', 'blog', 'беби блог', 'бэби блог', 'бб', ',t,b,kju', 'бейбиблог', 'бейбиблог.ру', ',\',b,kju', 'бэйбиблог', 'бейби блог', 'бэби блок', 'бэбибл', 'бебиб', 'бебибло'	106.74 MB/s	74.21 MB/s	394.98 MB/s	139.00 MB/s	511.22 MB/s

Сравнение алгоритмов для поиска

	Multi Volnitsky	Volnitsky n раз	Aho Corasick	RE2	Hyperscan
'fitnes-kluby', 'sportivnoe-oborudovanie-atributika', 'krytye-sportivnye-ploshchadki', 'pejntbol-strajk-i-hard-bol', 'strelkovye-kluby-tiry', 'sportivnye-organizatsii', 'basseyny-plavatelnye', 'otkrytye-sportivnye-ploshchadki-bazy', 'gornolyzhnye-sklony', 'pryzhki-s-parashyutom', 'sportivnye-sektsii', 'yakht-kluby', 'joga-centry-i-instruktory', 'bukmekerskie-kontory', 'skalolazanie-voskhozhdenie-v-gory', 'fekhtovalnye-kluby', 'drugoj-sport-i-fitnes', 'boevye-iskusstva'	889.12 MB/s	198.91 MB/s	640.33 MB/s	195.09 MB/s	642.42 MB/s

Сравнение алгоритмов для поиска

	Multi Volnitsky	Volnitsky n раз	Aho Corasick	RE2	Hyperscan
'chelyabinsk.74.ru', 'doctor.74.ru', 'transport.74.ru', 'm.74.ru', '//74.ru/', 'chel.74.ru', 'afisha.74.ru', 'diplom.74.ru', 'chelfin.ru', '//chel.ru', 'chelyabinsk.ru', 'cheldoctor.ru', '//mychel.ru', 'cheldiplom.ru', '74.ru/video', 'market', 'poll', 'mail', 'conference', 'consult', 'contest', 'tags', 'feedback', 'pages', 'text'	418.14 MB/s	230.13 MB/s	413.58 MB/s	183.31 MB/s	639.61 MB/s
'p17266p66989p97b7', 'p17266p66988p285b', 'p17266p66986pa4e8', 'p15926p65809pbab6', 'p15926p65810p2672', 'p15926p65811p9afa', 'p15926p65812p97e3', 'p15926p65813pd214', 'p15926p65813pd214', 'p15926p65815p350b', 'p15926p65816pe52c', 'p15926p65814p0cc9', 'p15926p65817p4cea', 'p15926p65818p9b20', 'p15926p65860p4435', 'p15926p65861p1f1e', 'p15926p65862p6f5b', 'p15926p65864pef1c', 'p15926p64433p9fca', 'p15926p64435p9eb7', 'p15926p64436p5e2c', 'p14762p59496p5de8', 'pfc4a'	1.20 GB/s	330.57 MB/s	990.87 MB/s	524.60 MB/s	1.10 GB/s

Сравнение алгоритмов для поиска

	Multi Volnitsky	Volnitsky n раз	Aho Corasick	RE2	Hyperscan
'вуман','вумен ру женский журнал','вумен форум', 'вуманжурнал ру','воман','www.woman.ru','вуман ру', 'woman ru','женский журнал','форум вумен','devty', 'вумен.ру','вумен ру форум','вуманжурнал','женский форум', 'журнал вумен','цщьфт','женский журнал вумен','women', 'woman форум','devty he','вуман.ру','женский форум вумен', 'women.ru женский сайт','форум вумен ру','вум','сайт вумен', 'воменс.ру','devfy','вомен','woman.ru журнал','woman.ru форум', 'вуменру','вуман форум','цщьфтюкг','devfy he', 'wom','вумен форум новое', 'вумен форум новые','вумэн','форум вуман	42.19 MB/s	31.10 MB/s	380.23 MB/s	140.07 MB/s	441.12 MB/s

Сравнение алгоритмов для поиска

- До 10-15 needle MultiVolnitsky обыгрывает всех (97% запросов)
- Деградация при большом количестве похожих needle
- Ускорение от большой минимальной длины needle

Новые фичи ClickHouse (19.5)

- **multiSearchAny(h, [n_1, …, n_k])** — хоть кто-то из needle
- **multiSearchFirstPosition(h, [n_1, …, n_k])** — самая левая позиция
- **multiSearchFirstIndex(h, [n_1, …, n_k])** — самый левый индекс
- **multiSearchAllPositions(h, [n_1, …, n_k])** — все первые позиции
- **Суффиксы -UTF8, -CaseInsensitive, -CaseInsensitiveUTF8**

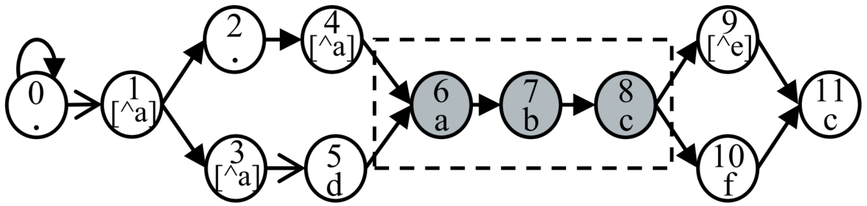
Все функции за основу используют MultiVolnitsky

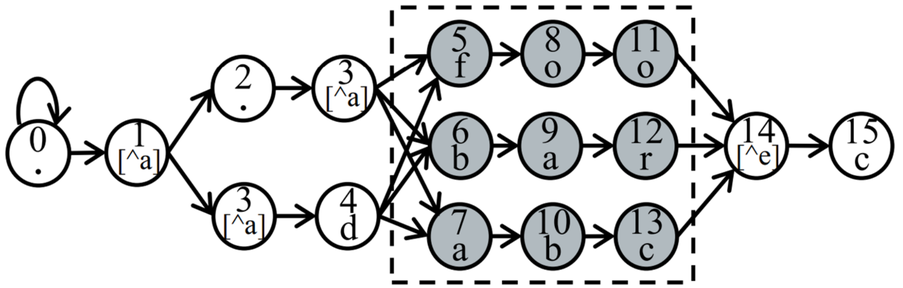
Сравнение алгоритмов для поиска множества регулярных выражений

Добавили поддержку Hyperscan

Статья на USENIX 2019 года. 5 версия. 12 лет разработки
Длина строки меньше 2^32. Недокументированное количество памяти

Зато быстрый





Сравнение алгоритмов для поиска

	RE2	Hyperscan
'/t[0-9]+-', '/questions/7{9}[0-9]+'	565.51 MB/s	743.88 MB/s
'ножниц.*вырубн', 'ножниц.*рычажн', 'ножниц.*гильотин'	155.72 MB/s	505.31 MB/s
'f[ae]b[ei]rl', 'ф[иаэе]б[еэи][рпл]', 'афиукд', 'a[ft],th', '^ф[аиеэ]?б?[еэи]?$', 'берлик', 'fab', 'фа[беьв]+е?[рлко]'	183.32 MB/s	422.65 MB/s
'/questions/q*', '/q[0-9]*/', '/questions/[0-9]*'	524.58 MB/s	936.12 MB/s
'//ngs.ru/$', '//m.ngs.ru/$', '//news.ngs.ru/$', '//m.news.ngs.ru/$', '//ngs.ru/\\?', '//m.ngs.ru/\\?', '//news.ngs.ru/\\?', '//m.news.ngs.ru/\\?'	543.41 MB/s	619.82 MB/s
'[ми][аеэпви][нм][асзи][иус]*', '[mn][aeauo][nm]s[yyi]*', 'ru', 'v[ft']v[cp][be]', 'www', 'ьфьын', 'маиси', 'mam', 'amsy', 'маммси', 'амси', 'vfvc'	141.17 MB/s	463.87 MB/s

Новые фичи ClickHouse (19.5)

- **multiMatchAny(h, [n_1, …, n_k])** — хоть кто-то подходит
- **multiMatchAnyIndex(h, [n_1, …, n_k])** — любой индекс из match

В функциях используется честный hyperscan.

Приближённый поиск

- **Хотим искать похожие строки между собой. (яндекс ~ индекс)**

Левенштейн — квадратично тяжело и “сильно” легче быть не может.

- **Опечатки в запросах по Левенштейну**

```
SELECT sum(multiSearch(URL, ['yandex', 'google', 'yahoo']))
FROM hits_100m_single
SETTINGS max_threads = 1

Received exception from server (version 19.9.1):
Code: 46. DB::Exception: Received from localhost:9000, ::1. DB::Exception: Unk
nown function multiSearch. Maybe you meant: ['multiSearchAny'].

0 rows in set. Elapsed: 0.002 sec.
```

Приближённый поиск

Есть расстояние Хэмминга. Есть алгоритм O(|Σ|n log n)

Использует Быстрое Преобразование Фурье — плохо

|Σ| = 30 уже много

Приближённый поиск

- **n-граммное расстояние**
- **Сделали ngramDistance**
- **0 — похожи, 1 — нет**
- **4-граммное расстояние**

abcda → {abcd, bcda}; Size = 2

bcdab → {bcda, cdab}; Size = 2

|{abcd, cdab}| / (2 + 2) = 0.5

```
SELECT DISTINCT SearchPhrase
FROM hits_100m_single
ORDER BY ngramDistance(SearchPhrase, 'clickhouse') ASC
LIMIT 20
```

```
┌─SearchPhrase─┐
│ tickhouse    │
│ clockhouse   │
│ house        │
│ clickhomecyprus │
│ 1click       │
│ uhouse       │
│ teakhouse.ru │
│ teakhouse.com │
│ madhouse     │
│ icehouse     │
│ doghouse     │
│ funhouse     │
│ dollhouse    │
│ houses.ru    │
│ bighouses    │
│ uhouse.ru    │
│ tic house    │
│ dog house    │
│ luk house    │
│ house m.d    │
└──────────────┘
```

Приближённый поиск

- Реализация ngramDistance

dbms/src/Functions/FunctionsStringSimilarity.cpp

```
struct NgramDistanceImpl;
```

- Больше 2^15 — не похожи друг на друга
- Наивно — 150 MB/s, SSE — 250 MB/s

Приближённый поиск

- **ngramDistance(haystack, needle)**
- **-UTF8, -CaseInsensitive, -CaseInsensitiveUTF8**

CaseInsensitiveUTF8 хак — только для русских и английских букв. Зануляем 5-й бит всех и нулевой бит нулевого байта, если байтов больше одного.

- **ngramSearch(haystack, needle) (19.8)** — приближенный поиск подстрок. Эксперимент.

Приближённый поиск

- **multiFuzzyMatchAny(haystack, distance, [n_1, ..., n_k])**
- **multiFuzzyMatchAnyIndex(haystack, distance, [n_1, ..., n_k])**

Поиск по Левенштейну по регулярному выражению

distance+1 конечных автоматов (слои)

Переходы между слоями — “штраф”

UTF-8-кодировка

Кодовые точки	Первый байт	Второй байт	Третий байт	Четвёртый байт
U+0000 . . . U+007F	00. . . 7F			
U+0080 . . . U+07FF	C2. . . DF	80. . . BF		
U+0800 . . . U+0FFF	E0	**A0**. . . BF	80. . . BF	
U+1000 . . . U+CFFF	E1. . . EC	80. . . BF	80. . . BF	
U+D000 . . . U+D7FF	ED	80. . . **9F**	80. . . BF	
U+E000 . . . U+FFFF	EE. . . EF	80. . . BF	80. . . BF	
U+10000 . . . U+3FFFF	F0	**90**. . . BF	80. . . BF	80. . . BF
U+40000 . . . U+FFFFF	F1. . . F3	80. . . BF	80. . . BF	80. . . BF
U+100000 . . . U+10FFFF	F4	80. . . **8F**	80. . . BF	80. . . BF

UTF-8-кодировка. Вычисление длины

- 0xBF = -65
- 0x80 = -128
- 0xC2 = -62
- 0x7F = 127
- Первые байты в
  [0xC2, 0x7F]
- Не первые байты в
  [0x80, 0xBF]

Кодовые точки	Первый байт	Второй байт	Третий байт	Четвёртый байт
U+0000 ... U+007F	00... 7F			
U+0080 ... U+07FF	C2... DF	80... BF		
U+0800 ... U+0FFF	E0	**A0**... BF	80... BF	
U+1000 ... U+CFFF	E1... EC	80... BF	80... BF	
U+D000 ... U+D7FF	ED	80... **9F**	80... BF	
U+E000 ... U+FFFF	EE... EF	80... BF	80... BF	
U+10000 ... U+3FFFF	F0	**90**... BF	80... BF	80... BF
U+40000 ... U+FFFFF	F1... F3	80... BF	80... BF	80... BF
U+100000 ... U+10FFFF	F4	80... **8F**	80... BF	80... BF

UTF-8-кодировка. Вычисление длины

Вход:

```
0x00 0xE0 0x80 0xF4 0x80 0xBF 0xBF 0x7A 0xEE 0x82 0x8A 0xE1 0x80 0x80 0x0E 0x0A

0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF 0xBF
```

↓ _mm_cmpgt_epi8 (pcmpgtb)

```
0xFF 0xFF 0x00 0xFF 0x00 0x00 0x00 0xFF 0xFF 0x00 0x00 0xFF 0x00 0x00 0xFF 0xFF
```

↓ _mm_movemask_epi8 (pmovmskb)

```
0b1101000110010011
```

↓ __builtin_popcount (popcnt)

8

UTF-8-кодировка. Проверка на валидность

- **isValidUTF8(string) (19.7)** — проверяет, является ли строка корректно UTF-8-закодированной.

  Наивно 900 MB/s. Нам мало.

- **Использовали алгоритм “диапазонов”.**
  Source: https://github.com/cyb70289/utf8/

  Соптимизировали до **1.22 GB/s**

- **toValidUTF8(string) (19.8)** — заменяем некорректные символы UTF-8 на символ `\uFFFD`.
  No rocket science.

Последние приятности

- **format(pattern, strings...) (19.8)** — Много SSE. Pattern — упрощённый python format.

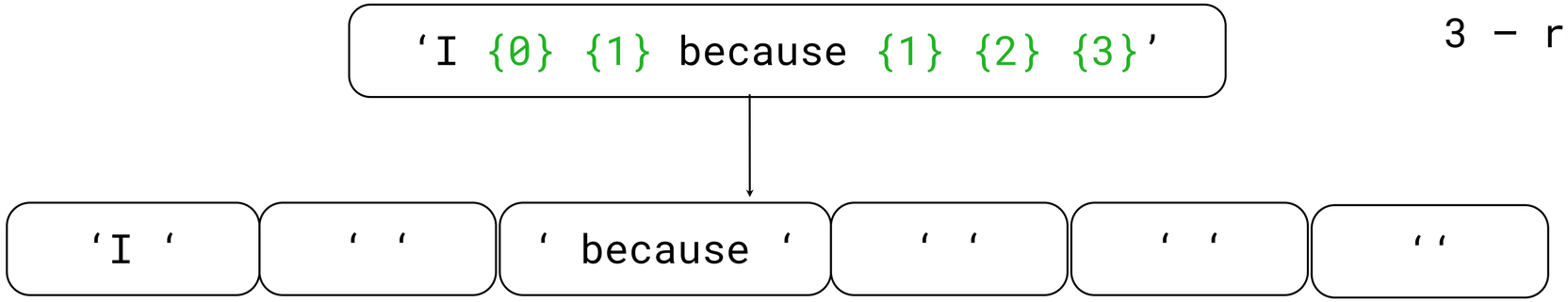
```
SELECT format('{1} {0} {1}', 'World', 'Hello')

┌─format('{1} {0} {1}', 'World', 'Hello')─┐
│ Hello World Hello                       │
└─────────────────────────────────────────┘

SELECT format('{} {}', 'Hello', 'World')

┌─format('{} {}', 'Hello', 'World')─┐
│ Hello World                       │
└───────────────────────────────────┘
```

- **concat(strings...)** — конкатенация строк, соптимизировали до 60%

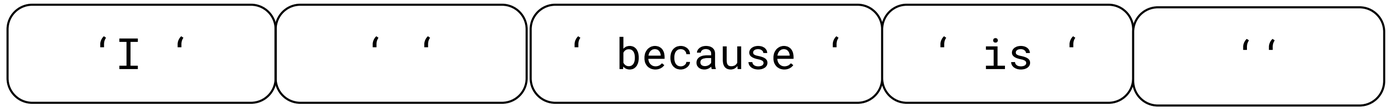
Последние приятности

- **format(pattern, strings...)**

```
0 – feeling
1 – subject
2 – verb
3 – reason
```



- Если, например, verb, константный, переклеим строки.



- SSE по 16 байт поклеим нужные строки между блоками.


```
‘I like ClickHouse because ClickHouse is fast’
‘I don’t like Vertica because Vertica is slow’
‘I like HighLoad because HighLoad is a good conference’
```

Яндекс

Спасибо

Данила Кутенин

Разработчик в группе разработки нового рантайма

danlark@yandex-team.ru

@Danlark

Митап уже завтра!

https://events.yandex.ru/events/ClickHouse/26-June-2019/