Индексы. Ускоряемся!

Андрей Коняев
ka@arenadata.io

ARENADATA

Странная задачка

1) Таблица на 1 млрд записей

2) 15 колонок. Строковые и числовые типы. Часть коррелируют, часть шум.

3) Выдача всех значений по набору ключей. Десятки строк в выхлопе.
   ```
   select * from t where x = val;
   ```

4) Менее 0.5 секунды на 17 потоков.

Машина для теста

1) Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz (16 vCPU)
2) 128 GB RAM
3) RAID-1 собранный на HDD.

Тестовая таблица

```
CREATE TABLE default.akonyaev_test
(
    `ind1` Int32,
    `ind2` Int32,
    `randString` String,
    `korrString` String,
    `korrInt` Int32,
    `randInt` Int32,
    `tsWithDisp` Int64
)
ENGINE = MergeTree()
ORDER BY (ind1, ind2)
SETTINGS index_granularity = 8192
```

Наполним данными

```
INSERT INTO akonyaev_test_ SELECT
    rand() % 20000 AS ind1,
    (ind1 * 1000000) + (rand() % 1000000) AS ind2,
    toString(generateUUIDv4()) AS randString,
    dictGetString('t', 'x', toUInt64(ind1 % 214) + (rand() % 10)) AS korrString,
    ind1 + (rand() % 10) AS korrInt,
    rand() AS randInt,
    number + (rand() % 1000)
FROM numbers(0, 1000000000)
```

Тестовая таблица

```
SELECT *
FROM akonyaev_test
ORDER BY tsWithDisp ASC
LIMIT 5
```

ind1	ind2	randString	korrString	korrInt	randInt	tsWithDisp
18004	825048820	6539d576-3656-4e56-b5bc-f1554ce46983	MAGIC	18008	-1999049292	16
14020	1135672132	eaca7abc-7499-41cb-aef9-9cdb5433f3b6	yourself!	14020	634574020	24
10017	1427075425	fb34c32c-92e0-4ba7-ab51-36d3d66a1e4a	different	10024	-1532957279	30
17012	-167632172	20572494-595d-4e63-98fb-4fd8e48c3f7e	however	17014	63237012	30
2001	2001702001	420e2b3f-d09f-4d78-ba1c-5df881683881	Muggle-ridden	2002	-118265295	37

Тестовая таблица

```
SELECT
    name,
    type,
    (data_compressed_bytes / 1024) / 1024 AS compressed_data_mb,
    (data_uncompressed_bytes / 1024) / 1024 AS uncompressed_data_mb,
    (compressed_data_mb / uncompressed_data_mb) * 100 AS p
FROM system.columns
WHERE table = 'akonyaev_test'
```

name	type	compressed_data_mb	uncompressed_data_mb	p
ind1	Int32	17.52880859375	3814.697265625	0.45950720000000006
ind2	Int32	38.93845462799072	3814.697265625	1.020748225
randString	String	35249.55929374695	35285.94970703125	99.89686996216216
korrString	String	28.33785915374756	6292.211601257324	0.4503640524117946
korrInt	Int32	17.532913208007812	3814.697265625	0.45961480000000005
randInt	Int32	3830.718102455139	3814.697265625	100.41997662499999
tsWithDisp	Int64	4948.574763298035	7629.39453125	64.86195913750001

Готовим первый тест

1) clickhouse-client --query "select ind1,ind2 from akonyaev_test order by randInt limit 10000" > ./ind

2) cat ./ind | awk {'print "http://localhost:8123/?query=select%20*%20from%20akonyaev_test_fb_3%20where%20ind1="$1"%20and%20ind2="$2";"'} > ./first_test

3) siege -c 5 -b --time=1m -f ./first_test

Первый тест

```
Lifting the server siege...
Transactions:                  11738 hits
Availability:                 100.00 %
Elapsed time:                  59.04 secs
Data transferred:             600.34 MB
Response time:                  0.02 secs
Transaction rate:             198.81 trans/sec
Throughput:                    10.17 MB/sec
Concurrency:                    4.78
Successful transactions:       11738
Failed transactions:               0
Longest transaction:           12.64
Shortest transaction:           0.00
```

Первый тест

```
Lifting the server siege...
Transactions:                  11738 hits
Availability:                 100.00 %
Elapsed time:                  59.04 secs
Data transferred:             600.34 MB
Response time:                  0.02 secs
Transaction rate:             198.81 trans/sec
Throughput:                    10.17 MB/sec
Concurrency:                    4.78
Successful transactions:       11738
Failed transactions:               0
Longest transaction:           12.64
Shortest transaction:           0.00
```

ARENADATA

Странная задачка. Level №2

1) Таблица на 1 млрд записей

2) 15 колонок. Строковые и числовые типы. Часть коррелируют, часть шум.

3) Выдача всех значений по произвольному набору фильтров. Десятки строк в выхлопе.
   `select * from t where x = val;`

4) Менее 0.5 секунды на 17 потоков.

Тест

```
cat ./s |  awk {'print
"http://localhost:8123/?query=select%20*%20from%20ak
onyaev_test%20where%20randString=%27"$1"%27;"'} >
./queries

siege -c 5 -b --time=1m -f ./queries
```

Провал

```
Lifting the server siege...
Transactions:                     85 hits
Availability:                 100.00 %
Elapsed time:                 299.51 secs
Data transferred:               0.01 MB
Response time:                 17.15 secs
Transaction rate:               0.28 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    4.87
Successful transactions:          85
Failed transactions:               0
Longest transaction:           19.80
Shortest transaction:          15.88
```

Провал

Уныние - грех!
Берём индексы.

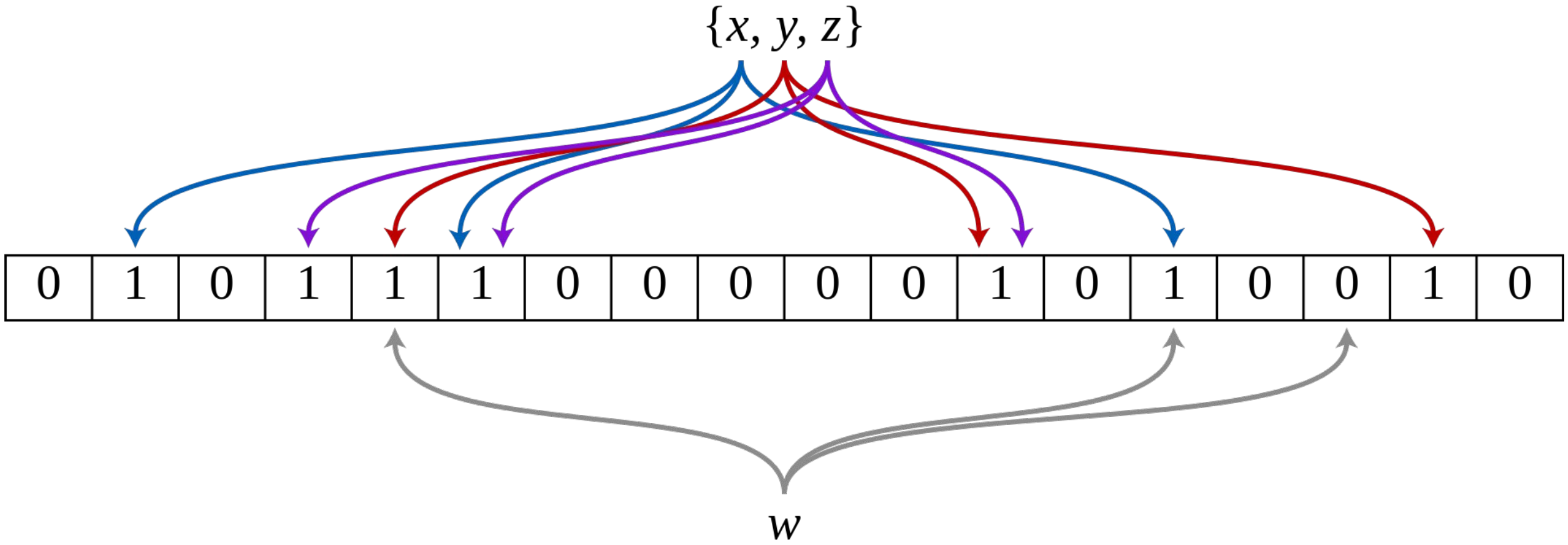
- minmax - для цифровых столбцов с локальностью внутри куска.
- set(max_rows) - для всех типов. Выгоден при небольшом количестве уникальных значений и корреляции набора и первичного ключа.
- bloom_filter([false_positive]) - фильтр Блума для колонки. Поддерживаются почти все типы. Пропускает блоки для функций: equeals,notEquals,in,notIn. То что нужно для нашего randString.

Фильтр Блума.

Фильтр Блума

- структура данных может точно ответить что элемента в множестве нет.
- имеет возможность false-positive ошибки, но не false-negative.
- чем больше элементов пропустили через фильтр - тем выше вероятность false-positive ошибки
- фильтр Блума для множества - это битовый массив. Чем он длиннее - тем вероятность ошибки ниже.

Фильтр Блума



Оптимизируем поиск по randString

- SET allow_experimental_data_skipping_indices = 1 (если делать тесты в кластере то выставляйте через users.xml)
- alter table akonyaev_test add INDEX bloom_index randString TYPE bloom_filter(false_positive) GRANULARITY gran_val

Для каждой вариации создаётся отдельная таблица, т.к. индекс не пересчитывается после ALTER. optimize с FINAL поможет только если есть что мёржить.

Варианты

Granularity \ False-positive p.	0.005	0.025	0.125
8192	2.74 secs	2.35 secs	2.35 secs
1000	2.04 secs	1.70 secs	1.70 secs
50	1.64 secs	1.34 secs	1.34 secs
5	1.35 secs	1.19 secs	1.19 secs
1	1.31 secs	1.45 secs	1.45 secs

- Это для пяти потоков в siege.
- Время ответа обратно пропорционально зависит от количества потоков долбящихся в ClickHouse.

ARENADATA

Сравним transaction rate.

Granularity \ False-positive p.	0.005	0.025	0.125
8192	x 6.7	x 7.8	x 6.0
1000	x 9.0	x 10.8	x 6.5
50	x 11.2	x 13.8	x 7.6
5	x 13.6	x 15.5	x 7.1
1	x 14.1	x 12.7	x 5.3

- Сравнение с transaction rate без индекса.

ARENADATA

И чем мы заплатим за это?
Размер индекса.

Granularity \ False-positive p.	0.005	0.025	0.125
8192	1,5G	958M	599M
1000	1,5G	958M	599M
50	1,8G	1,2G	770M
5	1,5G	959M	600M
1	1,5G	961M	602M

- Это индекс на 35ГБ шумовых данных.
- Размер линейно зависит от заданной вероятности.

Но фильтр Блума не панацея.

Колонкам в которых небольшое количество уникальных значений и эти значения имеют корреляцию с первичным ключём фильтр Блума может дать приемущество, но не так как set и minmax.

Но фильтр Блума не панацея.

Возьмём две колонки - korrInt и randInt.

```
SELECT
    korrInt,
    randInt
FROM akonyaev_test
ORDER BY randString ASC
LIMIT 1000000
```

На основе этой выборки сформируем http-запросы для siege.

ARENADATA

minmax

```
SELECT * FROM akonyaev_test where korrInt=val1 and randInt=val2
```

Индекс	без индекса	bloom на randInt	minmax на korrInt	minmax на korrInt + bloom на randInt
Response time	4.35 secs	0.75 secs	0.02 secs	0.02 secs
Transaction rate	1.15 trans/sec	6.62 trans/sec	238.11 trans/sec	260.18 trans/sec

Есть нюанс

```
SELECT * FROM akonyaev_test where korrInt=val1 and randInt=val2
```

Индекс	Без индекса	bloom_filter - korrInt minmax - korrInt	minmax - korrInt bloom_filter - korrInt
Response time	4.35 secs	0.39 secs	0.02 secs
Transaction rate	1.15 trans/sec	12.64 trans/sec	238.11 trans/sec

Используется первый индекс для колонки подходящий для условия в секции WHERE.

Так как Вы можете создать несколько индексов на колонку - **БУДЬТЕ БДИТЕЛЬНЫ!**

ARENADATA

И в конце...

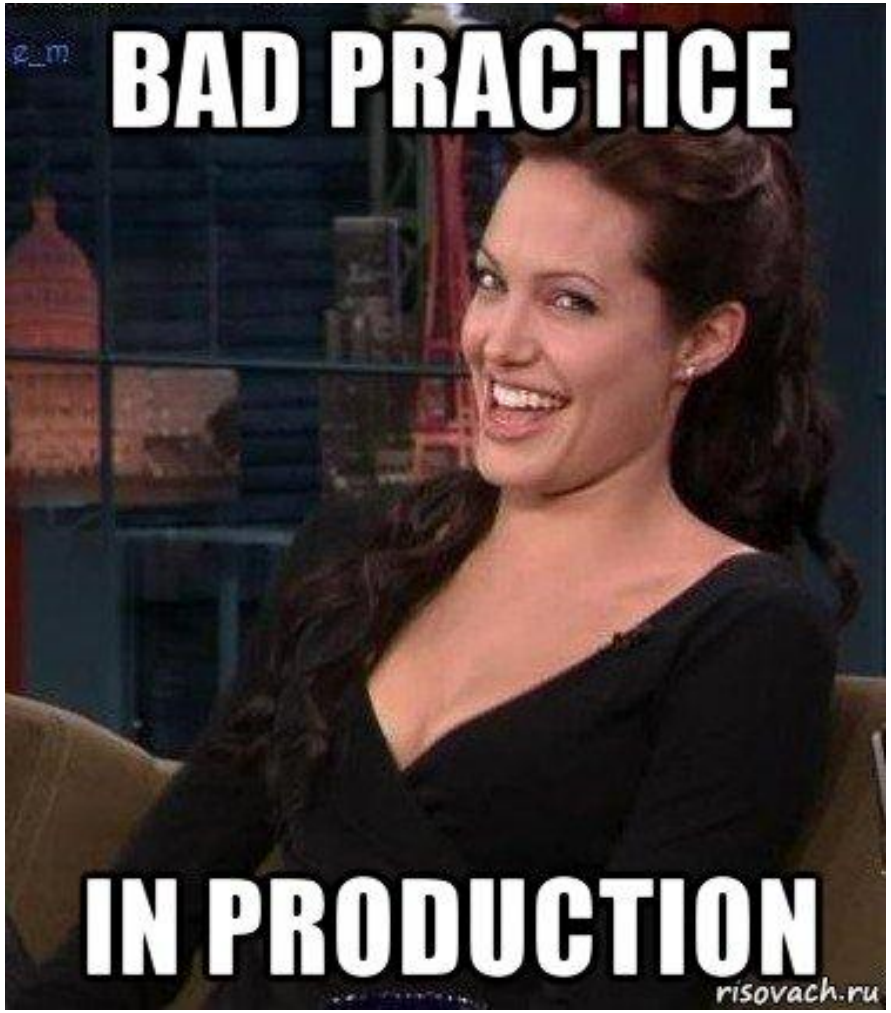
ClickHouse не задумывался для таких нагрузок. Это вообще не его задачи.

Для этого есть key-value и многое другое.

Но если очень нужно - то он постарается!

Если знать распределение данных и отношения столбцов - то можно выработать рекомендации по индексам.

А потом это автоматизировать...



ARENADATA

Q&A

Андрей Коняев
ka@arenadata.io

ARENADATA