НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Добавление в ClickHouse поддержки constraints

Курсовая работа

Глеб Новиков, 3 курс ВШЭ БПМИ

Алексей Миловидов, руководитель группы разработки ClickHouse

Факультет компьютерных наук
01.03.02 «Прикладная математика и информатика»

6 июня 2019 г.

Ограничения (constraints) — набор обязательных правил для данных, находящихся в таблице базы данных. Например:

- `DEFAULT some_default_value`
- `NOT NULL`
- `UNIQUE [(column_1[, column_2])]`
- `CHECK expression`
- `...`

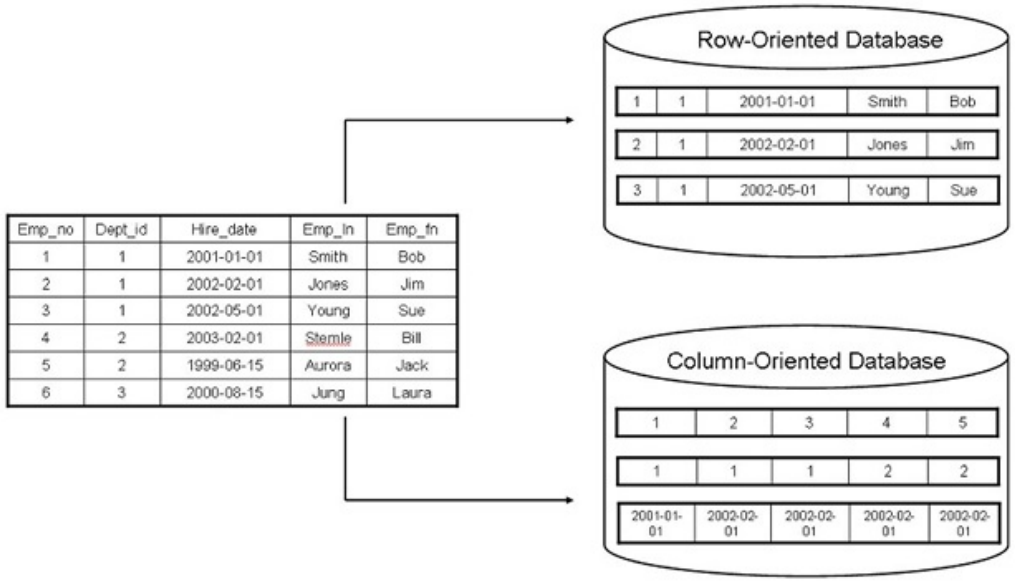
Таблица продуктов с примером объявления разных типов ограничений:

```
CREATE TABLE products (
    id integer PRIMARY KEY,
    shop_id integer REFERENCES shops,
    product text NOT NULL,
    price numeric CHECK (price > 0)
);
```

Колоночные базы данных

Что это и зачем?

- Данные хранятся не по строкам, а по колонкам:

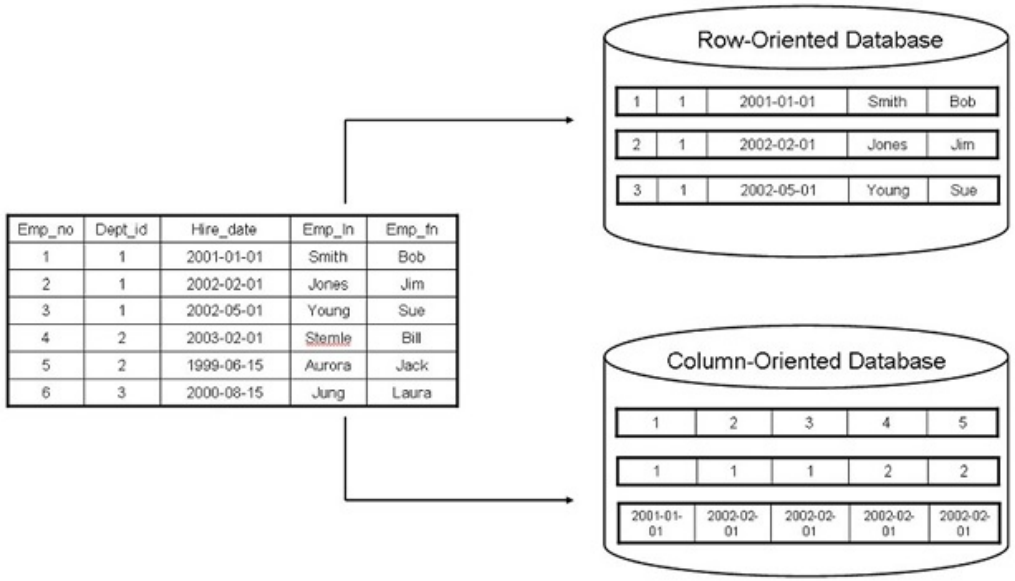


Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura

Колоночные базы данных

Что это и зачем?

- Данные хранятся не по строкам, а по колонкам:



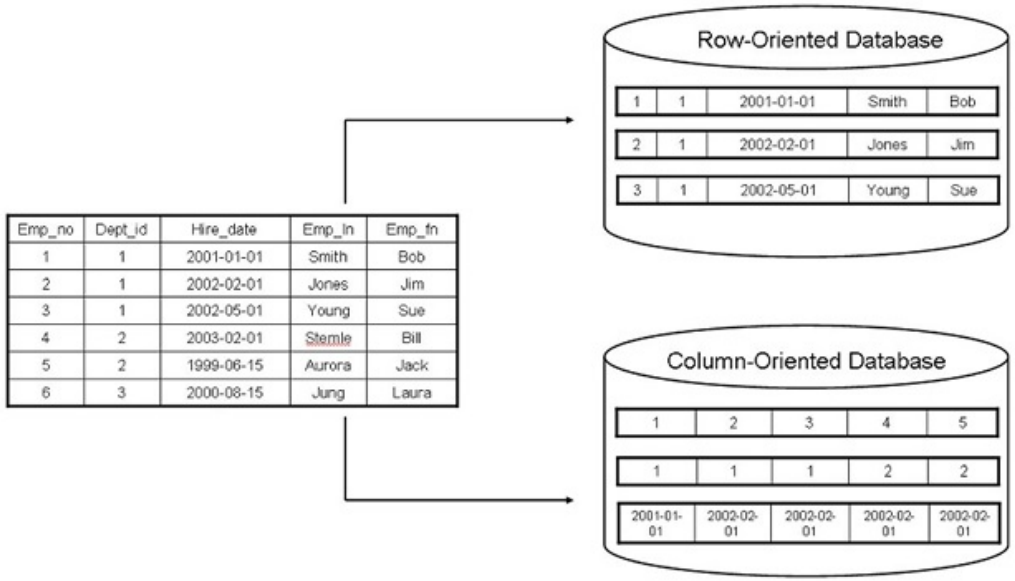
Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura

- Используются для хранения сырых неаггрегированных данных (например логов)

Колоночные базы данных

Что это и зачем?

- Данные хранятся не по строкам, а по колонкам:



Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura

- Используются для хранения сырых неаггрегированных данных (например логов)
- В основном используются для аггрегационных online запросов (возможно, приближённых)

Колоночные базы данных

Производительность ClickHouse

Full results:

Query	ClickHouse (19.1.6)		Vertica (7.1.1)		MonetDB		MySQL (5.5.32, MyISAM)	
SELECT count() FROM hits	x13.30 (0.133 s.)	x1.60 (0.016 s.)	x4.49 (0.045 s.)	x3.35 (0.033 s.)	x2.89 (0.029 s.)	(0.004 s.)	(0.010 s.)	(0.010 s.)
SELECT count() FROM hits WHERE AdvEngine	(0.086 s.)	(0.012 s.)	x1.45 (0.125 s.)	x3.70 (0.044 s.)	x31.40 (2.700 s.)	x11.64 (0.140 s.)	x2562.67 (220.390 s.)	x19526.67 (234.320 s.)
SELECT sum(AdvEngineID), count(), avg(Resol	x1.00 (0.254 s.)	(0.041 s.)	(0.254 s.)	x4.08 (1.167 s.)	x4.34 (1.100 s.)	x3.57 (0.147 s.)	x869.37 (220.450 s.)	x4836.83 (198.310 s.)
SELECT sum(UserID) FROM hits	x1.79 (0.478 s.)	(0.047 s.)	(0.297 s.)	x1.29 (0.061 s.)	—	—	x777.40 (207.600 s.)	x4055.11 (190.590 s.)
SELECT uniq(UserID) FROM hits	(0.655 s.)	(0.104 s.)	x1.38 (0.901 s.)	x8.47 (0.851 s.)	x11.45 (7.500 s.)	x53.85 (5.600 s.)	x421.31 (275.960 s.)	x2411.92 (250.840 s.)
SELECT uniq(SearchPhrase) FROM hits	(0.581 s.)	(0.228 s.)	x2.21 (1.284 s.)	x4.35 (0.991 s.)	x26.68 (15.500 s.)	x55.70 (12.700 s.)	x502.87 (292.170 s.)	x1114.65 (254.140 s.)
SELECT min(EventDate), max(EventDate) FROI	(0.105 s.)	x1.24 (0.048 s.)	x1.43 (0.150 s.)	x1.76 (0.068 s.)	x9.52 (1.000 s.)	(0.038 s.)	x719.14 (75.510 s.)	x1973.50 (76.110 s.)
SELECT AdvEngineID, count() FROM hits WHE	(0.076 s.)	(0.012 s.)	x1.96 (0.149 s.)	x6.92 (0.083 s.)	x9.05 (0.688 s.)	x2.19 (0.026 s.)	x2683.42 (203.940 s.)	x15345.00 (184.140 s.)
SELECT RegionID, uniq(UserID) AS u FROM hit	(0.892 s.)	(0.509 s.)	x2.07 (1.848 s.)	x3.61 (1.837 s.)	x218.61 (195.000 s.)	x420.43 (214.000 s.)	x322.06 (287.280 s.)	x496.11 (252.520 s.)
SELECT RegionID, sum(AdvEngineID), count() /	(1.077 s.)	(0.542 s.)	x3.97 (4.270 s.)	x7.96 (4.314 s.)	x235.84 (254.000 s.)	x492.62 (267.000 s.)	x278.03 (299.440 s.)	x520.33 (282.020 s.)
SELECT MobilePhoneModel, uniq(UserID) AS u	x1.48 (0.529 s.)	(0.171 s.)	(0.358 s.)	x1.56 (0.267 s.)	x63.92 (22.900 s.)	x116.96 (20.000 s.)	x610.44 (218.710 s.)	x1155.03 (197.510 s.)
SELECT MobilePhone, MobilePhoneModel, uniq	x1.29 (0.541 s.)	(0.188 s.)	(0.421 s.)	x2.03 (0.381 s.)	x48.49 (20.400 s.)	x110.64 (20.800 s.)	x523.15 (220.100 s.)	x1048.78 (197.170 s.)
SELECT SearchPhrase, count() AS c FROM hits	(0.975 s.)	(0.659 s.)	x5.43 (5.294 s.)	x6.49 (4.260 s.)	x99.49 (97.000 s.)	x174.51 (115.000 s.)	x953.28 (929.450 s.)	x1319.79 (869.740 s.)
SELECT SearchPhrase, uniq(UserID) AS u FRO	(1.479 s.)	(0.812 s.)	x5.98 (8.849 s.)	x10.77 (8.745 s.)	x75.05 (111.000 s.)	x103.45 (84.000 s.)	—	—
SELECT SearchEngineID, SearchPhrase, count	(1.052 s.)	(0.734 s.)	x4.13 (4.470 s.)	x5.73 (4.203 s.)	x606.28 (656.000 s.)	—	x1105.75 (1196.420 s.)	—
SELECT UserID, count() FROM hits GROUP BY	(0.952 s.)	(0.728 s.)	x1.62 (1.542 s.)	x2.07 (1.506 s.)	x7.14 (6.800 s.)	x9.48 (6.900 s.)	—	—
SELECT UserID, SearchPhrase, count() FROM	(2.212 s.)	(1.756 s.)	x3.72 (8.230 s.)	x4.63 (8.130 s.)	x119.35 (264.000 s.)	x133.26 (234.000 s.)	—	—
SELECT UserID, SearchPhrase, count() FROM	(1.230 s.)	(1.053 s.)	x4.12 (5.105 s.)	x5.01 (5.178 s.)	x180.49 (222.000 s.)	x270.09 (279.000 s.)	—	—
SELECT UserID, toMinute(EventTime) AS m, Se	(4.613 s.)	(4.030 s.)	x4.57 (21.097 s.)	x5.24 (21.114 s.)	x254.28 (1173.000 s.)	—	—	—
SELECT UserID FROM hits WHERE UserID = 1	x2.00 (0.477 s.)	x7.40 (0.074 s.)	(0.299 s.)	x16.35 (0.164 s.)	x16.32 (3.900 s.)	(0.001 s.)	x1873.50 (447.720 s.)	x19909.00 (199.090 s.)
SELECT count() FROM hits WHERE URL LIKE '	(1.648 s.)	(0.582 s.)	x2.23 (3.682 s.)	x6.89 (3.665 s.)	x16.63 (27.400 s.)	x3.20 (1.700 s.)	x353.60 (582.730 s.)	x369.79 (196.730 s.)
SELECT SearchPhrase, any(URL), count() AS c	(2.002 s.)	(0.645 s.)	x2.00 (3.996 s.)	x6.01 (3.875 s.)	x2.00 (4.000 s.)	x4.03 (2.600 s.)	x290.98 (582.540 s.)	x319.21 (205.890 s.)
SELECT SearchPhrase, any(URL), any(Title), cc	(3.676 s.)	(1.479 s.)	x1.38 (5.086 s.)	x3.38 (5.000 s.)	—	—	x154.72 (568.750 s.)	x146.82 (217.150 s.)

Бенчмарки ClickHouse и других колоночных БД

https://clickhouse.yandex/benchmark.html

Ограничения в колоночных БД

Почти

не нужны:

- `UNIQUE`
- `FOREIGN KEY`

Ограничения в колоночных БД

Почти

не нужны:

- `UNIQUE`
- `FOREIGN KEY`

Нужны:

- `PRIMARY KEY`
- `DEFAULT`
- `NOT NULL`
- `CHECK`

Задача

Формулировка требований

Реализовать поддержку `NOT NULL` *и* `CHECK` *ограничений (на самом деле достатточно только* `CHECK`*, так как существуют функции* `isNull` *и* `isNotNull`*).*

Вместе с объявлением нужны все сопутствующие возможности: отдельное добавление и удаление ограничений, а так же исключение при обработке некорректных данных в процессе вставки.

Задача

Запрос объявления ограничений

```
CREATE TABLE products (
    id              UInt32,
    shop_id         UInt32,
    title           String,
    price           Decimal32(6),
    CONSTRAINT non_empty_title CHECK NOT empty(title),
    CONSTRAINT price_above_zero CHECK price > 0
) ENGINE = MergeTree ORDER BY (id);
```

Конвеер выполнения запроса

```
...
  . executeQuery
    . setting up query context and settings
    . parseQuery
      . query parsing to AST
    . InterpreterFactory::get
      . choose interpreter based on AST type and meta
    . interpreter.execute()
      . storages and meta changes
      . pre-processing of data streams
    . processing of data streams
    . query logging etc
```

Конвеер выполнения запроса

```
...
  . executeQuery
    . setting up query context and settings
    * > parseQuery
      * > query parsing to AST
    * > InterpreterFactory::get
      * > choose interpreter based on AST type and meta
    * > interpreter.execute()
      * > storages and meta changes
      * > pre-processing of data streams
    * > processing of data streams
    . query logging etc
```

Архитектура

Затронутые библиотеки:

```
> src
  . DataStreams
  . Interpreters
  . Parsers
  . Storages (for data and meta data)
  . [Core, Client, IO, ... ]
```

Реализация

CREATE запрос, парсеры

1. Parsers

Реализация

CREATE запрос, парсеры

1. Parsers
   1.1 `ASTConstraintDeclaration` — хранение мета-данных ограничений из запроса

Реализация

CREATE запрос, парсеры

1. Parsers
   1.1 `ASTConstraintDeclaration` — хранение мета-данных ограничений из запроса
   1.2 Правки `ASTColumns` для хранения ограничений и базовой работы с ними

Реализация

CREATE запрос, парсеры

1. Parsers
   - 1.1 `ASTConstraintDeclaration` — хранение мета-данных ограничений из запроса
   - 1.2 Правки `ASTColumns` для хранения ограничений и базовой работы с ними
   - 1.3 `ParserConstraintDeclaration` — парсер объявления ограничения (строки)

Реализация

CREATE запрос, парсеры

1. Parsers
   - 1.1 `ASTConstraintDeclaration` — хранение мета-данных ограничений из запроса
   - 1.2 Правки `ASTColumns` для хранения ограничений и базовой работы с ними
   - 1.3 `ParserConstraintDeclaration` — парсер объявления ограничения (строки)
   - 1.4 Правки `ParserTablePropertyDeclaration` — добавление парсинга ограничения по ключу `CONSTRAINT`

2. Storages

2. Storages
    2.1 `ConstraintsDescription`

2. Storages
   - 2.1 `ConstraintsDescription`
   - 2.2 Правки `IStorage` — добавление работы с `ConstraintsDescription`

2. Storages
   - 2.1 `ConstraintsDescription`
   - 2.2 Правки `IStorage` — добавление работы с `ConstraintsDescription`
   - 2.3 Добавление поддержки в `MergeTree*` структуры

Реализация

CREATE запрос, хранение и интепретация

2. Storages
   - 2.1 `ConstraintsDescription`
   - 2.2 Правки `IStorage` — добавление работы с `ConstraintsDescription`
   - 2.3 Добавление поддержки в `MergeTree*` структуры
3. Interpreters

Реализация

CREATE запрос, хранение и интепретация

2. Storages
   - 2.1 `ConstraintsDescription`
   - 2.2 Правки `IStorage` — добавление работы с `ConstraintsDescription`
   - 2.3 Добавление поддержки в `MergeTree*` структуры
3. Interpreters
   - 3.1 Добавление работы с ограничениями в `InterpreterCreateQuery`

Реализация

INSERT запрос, потоки данных и интерпретация

1. DataStreams

Реализация

INSERT запрос, потоки данных и интерпретация

1. DataStreams
    1.1 `CheckConstraintsBlockOutputStream` — проверка ограничений на блоке данных в момент его записи

Реализация

INSERT запрос, потоки данных и интерпретация

1. DataStreams
   - 1.1 `CheckConstraintsBlockOutputStream` — проверка ограничений на блоке данных в момент его записи
   - \+ `Columns/ColumnsCommon`: добавлена функция `memoryIsByte`, переписана `memoryIsZero`

Реализация

INSERT запрос, потоки данных и интерпретация

1. DataStreams
   - 1.1 `CheckConstraintsBlockOutputStream` — проверка ограничений на блоке данных в момент его записи
   - \+ `Columns/ColumnsCommon`: добавлена функция `memoryIsByte`, переписана `memoryIsZero`
2. Interpreters

Реализация

INSERT запрос, потоки данных и интерпретация

1. DataStreams
   - 1.1 `CheckConstraintsBlockOutputStream` — проверка ограничений на блоке данных в момент его записи
   - \+ `Columns/ColumnsCommon`: добавлена функция `memoryIsByte`, переписана `memoryIsZero`
2. Interpreters
   - 2.1 Обёртка в `CheckConstraintsBlockOutputStream` потока блоков данных

Реализация

ALTER запросы

1. Parsers

Реализация

ALTER запросы

1. Parsers
   1.1 Правки `ASTAlterCommand` — добавление нового типа команд, хранение AST ограничений

Реализация

ALTER запросы

1. Parsers
   - 1.1 Правки `ASTAlterCommand` — добавление нового типа команд, хранение AST ограничений
   - 1.2 Правки в `ParserAlterCommand` — парсинг нового типа команд

Реализация

ALTER запросы

1. Parsers
   1.1 Правки `ASTAlterCommand` — добавление нового типа команд, хранение AST ограничений
   1.2 Правки в `ParserAlterCommand` — парсинг нового типа команд
2. Storages

Реализация

ALTER запросы

1. Parsers
   - 1.1 Правки `ASTAlterCommand` — добавление нового типа команд, хранение AST ограничений
   - 1.2 Правки в `ParserAlterCommand` — парсинг нового типа команд
2. Storages
   - 2.1 Правки `AlterCommand` — реализация добавления и удаления ограничений

Реализация

ALTER запросы

1. Parsers
   - 1.1 Правки `ASTAlterCommand` — добавление нового типа команд, хранение AST ограничений
   - 1.2 Правки в `ParserAlterCommand` — парсинг нового типа команд
2. Storages
   - 2.1 Правки `AlterCommand` — реализация добавления и удаления ограничений
   - 2.2 Прокидвание `ConstraintsDescription` во всех `alter` и `alterTable` методах разных `Storages`

Примеры

Объявление и удаление constraints

```
CREATE TABLE products (
    id              UInt32,
    shop_id         UInt32,
    title           String,
    price           Decimal32(6),
    CONSTRAINT non_empty_title CHECK NOT empty(title),
    CONSTRAINT price_above_zero CHECK price > 0
) ENGINE = MergeTree ORDER BY (id);
```

Примеры

Объявление и удаление constraints

```
ALTER TABLE products
    DROP CONSTRAINT non_empty_title;

ALTER TABLE products
    DROP CONSTRAINT price_above_zero;

ALTER TABLE products
    ADD CONSTRAINT non_empty_title
            CHECK NOT empty(title);

ALTER TABLE products
    ADD CONSTRAINT price_above_zero
            CHECK price > 0;
```

Примеры

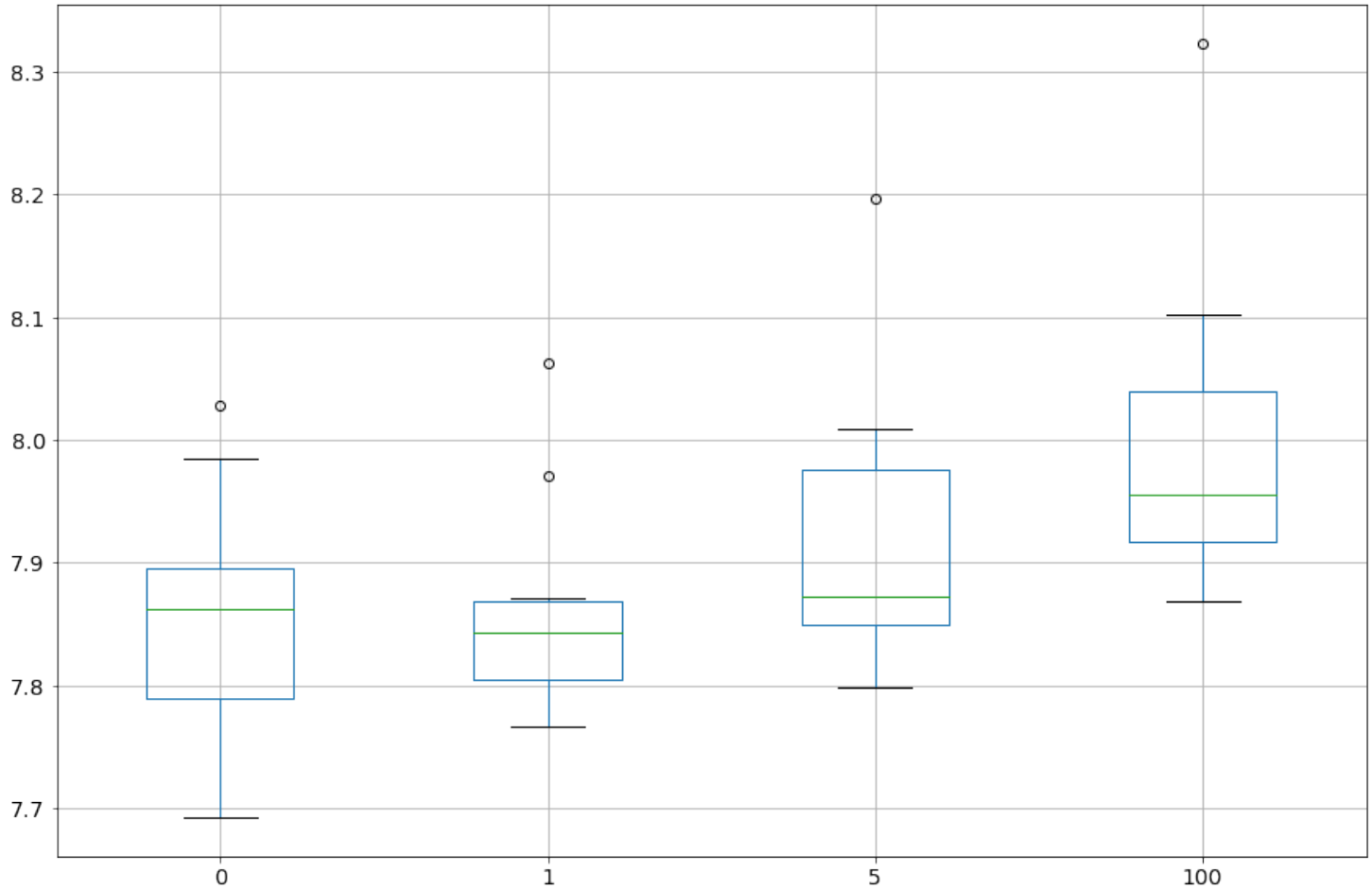
Исключения при нарушении constraints

```
clickhouse :) INSERT INTO products VALUES (1, 2, '', 1.1);

Code: 49. DB::Exception: Constraint non_empty_title
is not satisfied, constraint expression: NOT empty(title).

clickhouse :) INSERT INTO products VALUES (1, 2, 'a', 0);

Code: 49. DB::Exception: Constraint price_above_zero
is not satisfied, constraint expression: price > 0.
```

Бенчмарк

Разное количество ограничений на одной таблице

1 млн. строк, ~130 колонок, 854 МБ.



Время вставки датасета в сек.

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных

Код доступен в PR # 5273 по ссылке

https://github.com/yandex/ClickHouse/pull/5273/

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse

Код доступен в PR # 5273 по ссылке

https://github.com/yandex/ClickHouse/pull/5273/

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений
   - 4.2 Добавление и удаление ограничений

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений
   - 4.2 Добавление и удаление ограничений
   - 4.3 Проверка ограничений на вставляемые данные

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений
   - 4.2 Добавление и удаление ограничений
   - 4.3 Проверка ограничений на вставляемые данные
5. Написаны тесты запросов и исключений на вставку

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений
   - 4.2 Добавление и удаление ограничений
   - 4.3 Проверка ограничений на вставляемые данные
5. Написаны тесты запросов и исключений на вставку
6. Проведён простой бенчмарк, потеря около 2%

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что было сделано?

1. Изучены существующие constraints в различных базах данных
2. Изучены особенности колоночных БД, в частности ClickHouse
3. Изучена необходимая часть архитектуры ClickHouse
4. Реализовано:
   - 4.1 Объявление ограничений
   - 4.2 Добавление и удаление ограничений
   - 4.3 Проверка ограничений на вставляемые данные
5. Написаны тесты запросов и исключений на вставку
6. Проведён простой бенчмарк, потеря около 2%
7. Расширена документация, добавлены описания constraints

Код доступен в PR # 5273 по ссылке

`https://github.com/yandex/ClickHouse/pull/5273/`

Заключение

Что дальше?

1. Подробное тестирование на всех движках таблиц

Заключение

Что дальше?

1. Подробное тестирование на всех движках таблиц
2. Более подробные исключения: строка в данных, значения в строке

Заключение

Что дальше?

1. Подробное тестирование на всех движках таблиц
2. Более подробные исключения: строка в данных, значения в строке
3. Команда на проверку ограничений по существующим данным в таблице

Заключение

Что дальше?

1. Подробное тестирование на всех движках таблиц
2. Более подробные исключения: строка в данных, значения в строке
3. Команда на проверку ограничений по существующим данным в таблице
4. Оптимизации запросов с использованием информации из ограничений

Заключение

Что дальше?

1. Подробное тестирование на всех движках таблиц
2. Более подробные исключения: строка в данных, значения в строке
3. Команда на проверку ограничений по существующим данным в таблице
4. Оптимизации запросов с использованием информации из ограничений
5. ...

Спасибо за внимание!

Добавление в ClickHouse поддержки constraints

Глеб Новиков, 3 курс ВШЭ БПМИ

Алексей Миловидов, руководитель группы разработки ClickHouse

Факультет компьютерных наук
01.03.02 «Прикладная математика и информатика»

6 июня 2019 г.