ZSON, или о прозрачном сжатии JSONB

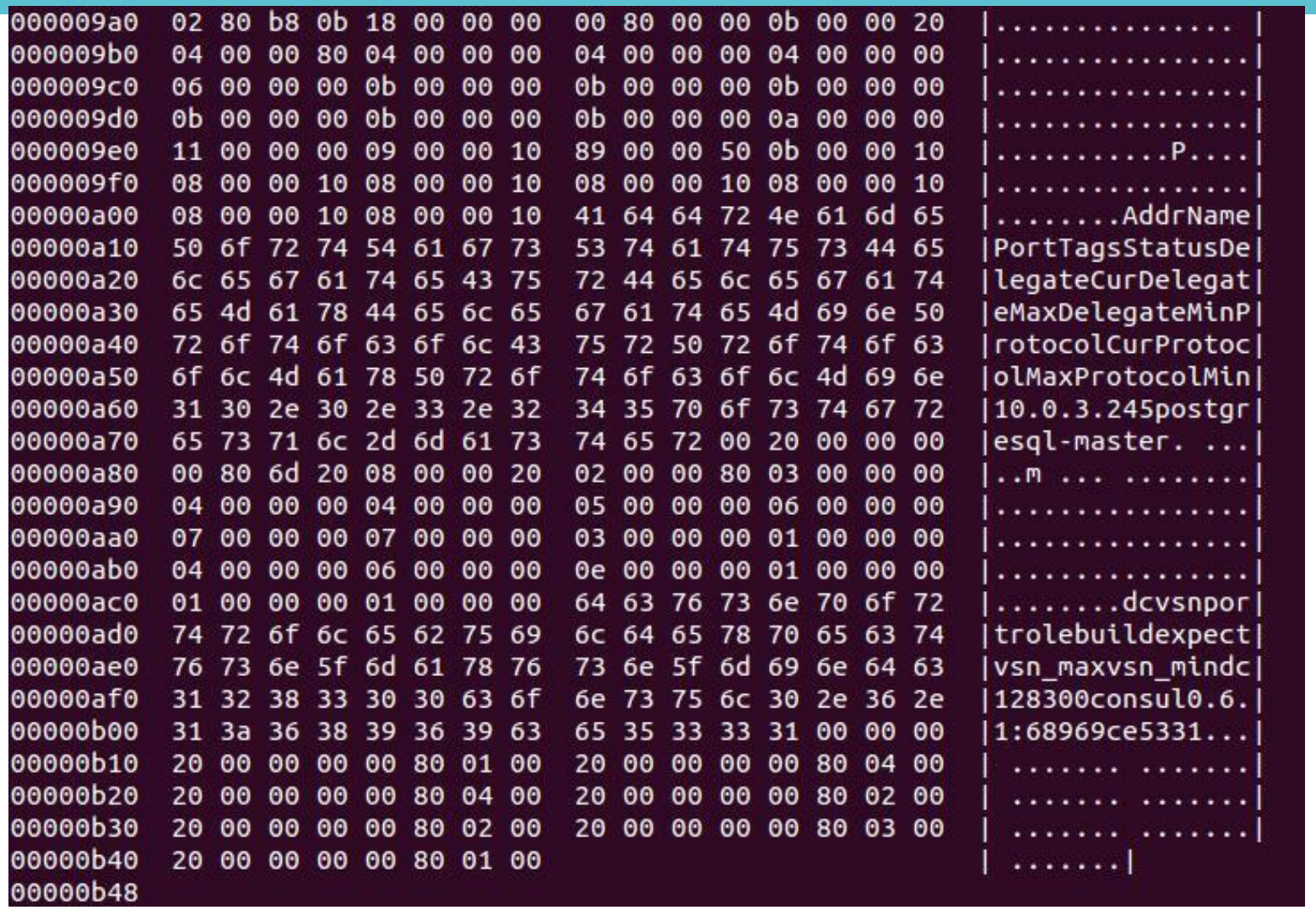
Александр Алексеев
twitter.com/afikson
mail@eax.me

postgrespro.ru

Два слова о себе

- http://eax.me/
- http://devzen.ru/

Как выглядит JSONB



```
000009a0  02 80 b8 0b 18 00 00 00  00 80 00 00 0b 00 00 20  |............... |
000009b0  04 00 00 80 04 00 00 00  04 00 00 00 04 00 00 00  |................|
000009c0  06 00 00 00 0b 00 00 00  0b 00 00 00 0b 00 00 00  |................|
000009d0  0b 00 00 00 0b 00 00 00  0b 00 00 00 0a 00 00 00  |................|
000009e0  11 00 00 00 09 00 00 10  89 00 00 50 0b 00 00 10  |...........P....|
000009f0  08 00 00 10 08 00 00 10  08 00 00 10 08 00 00 10  |................|
00000a00  08 00 00 10 08 00 00 10  41 64 64 72 4e 61 6d 65  |........AddrName|
00000a10  50 6f 72 74 54 61 67 73  53 74 61 74 75 73 44 65  |PortTagsStatusDe|
00000a20  6c 65 67 61 74 65 43 75  72 44 65 6c 65 67 61 74  |legateCurDelegat|
00000a30  65 4d 61 78 44 65 6c 65  67 61 74 65 4d 69 6e 50  |eMaxDelegateMinP|
00000a40  72 6f 74 6f 63 6f 6c 43  75 72 50 72 6f 74 6f 63  |rotocolCurProtoc|
00000a50  6f 6c 4d 61 78 50 72 6f  74 6f 63 6f 6c 4d 69 6e  |olMaxProtocolMin|
00000a60  31 30 2e 30 2e 33 2e 32  34 35 70 6f 73 74 67 72  |10.0.3.245postgr|
00000a70  65 73 71 6c 2d 6d 61 73  74 65 72 00 20 00 00 00  |esql-master. ...|
00000a80  00 80 6d 20 08 00 00 20  02 00 00 80 03 00 00 00  |..m ... ........|
00000a90  04 00 00 00 04 00 00 00  05 00 00 00 06 00 00 00  |................|
00000aa0  07 00 00 00 07 00 00 00  03 00 00 00 01 00 00 00  |................|
00000ab0  04 00 00 00 06 00 00 00  0e 00 00 00 01 00 00 00  |................|
00000ac0  01 00 00 00 01 00 00 00  64 63 76 73 6e 70 6f 72  |........dcvsnpor|
00000ad0  74 72 6f 6c 65 62 75 69  6c 64 65 78 70 65 63 74  |trolebuildexpect|
00000ae0  76 73 6e 5f 6d 61 78 76  73 6e 5f 6d 69 6e 64 63  |vsn_maxvsn_mindc|
00000af0  31 32 38 33 30 30 63 6f  6e 73 75 6c 30 2e 36 2e  |128300consul0.6.|
00000b00  31 3a 36 38 39 36 39 63  65 35 33 33 31 00 00 00  |1:68969ce5331...|
00000b10  20 00 00 00 00 80 01 00  20 00 00 00 00 80 04 00  | ....... .......|
00000b20  20 00 00 00 00 80 04 00  20 00 00 00 00 80 02 00  | ....... .......|
00000b30  20 00 00 00 00 80 02 00  20 00 00 00 00 80 03 00  | ....... .......|
00000b40  20 00 00 00 00 80 01 00                           | .......|
00000b48
```

Суть проблемы

- JSONB избыточен
- Место на диске
- Память
- Как следствие - IO и TPS

В чем заключается идея

“А давайте, зная кое-что о формате, сожмем его лучше PostgreSQL, сэкономим место, память, выиграем по I/O”

Народная мудрость

Чтобы сделать что-то правильно, нужно сделать это хотя бы три раза.

Попытки 1, 2, 3

- ZLIB, LZ4, LZMA, ...
  - Не сильно лучше PGLZ
- Свой словарный метод сжатия
  - Жмет хуже ZLIB и медленнее LZ4
- ZSON
  - Успех!

Суть идеи

Сжатие в два прохода:

- Шаг 1 - замена общих строк на коды
- Шаг 2 - обычное сжатие PGLZ (*)

Должно стать как минимум не намного хуже того, что есть

Установка

Собираем и устанавливаем:

```
cd /path/to/zson/source/code
make
sudo make install
```

Прогоняем тесты:

```
make installcheck
```

Подключаемся к PostgreSQL:

```
psql my_database
```

Включаем расширение:

```
create extension zson;
```

Обучение

Процедура:

```
zson_learn(
    tables_and_columns text[][],
    max_examples int default 10000,
    min_length int default 2,
    max_length int default 128,
    min_count int default 2
)
```

Пример использования:

```
select zson_learn('{{"tbl1", "col1"}, {"tbl2", "col2"}}')
```

zson_extract_strings

```
CREATE FUNCTION zson_extract_strings(x jsonb)
    RETURNS text[] AS $$
DECLARE
    jtype text;
    jitem jsonb;
BEGIN
    jtype := jsonb_typeof(x);
    IF jtype = 'object' THEN
        RETURN array(select unnest(z) from (
                select array(select jsonb_object_keys(x)) as z
            union all (
                select zson_extract_strings(x -> k) as z from (
                    select jsonb_object_keys(x) as k
                ) as kk
            )
        ) as zz);
    ELSIF jtype = 'array' THEN
       RETURN ARRAY(select unnest(zson_extract_strings(t)) from
            (select jsonb_array_elements(x) as t) as tt);
    ELSIF jtype = 'string' THEN
        RETURN array[ x #>> array[] :: text[] ];
    ELSE -- 'number', 'boolean', 'bool'
        RETURN array[] :: text[];
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Как хранится словарь

```
CREATE TABLE zson_dict (
  dict_id SERIAL NOT NULL,
  word_id INTEGER NOT NULL,
  word text NOT NULL,
  PRIMARY KEY(dict_id, word_id)
);
```

Использование

Проверяем словарь:

```
select * from zson_dict;
```

Теперь используем zson:

```
create table zson_example(x zson);
insert into zson_example values ('{"aaa": 123}');
select x -> 'aaa' from zson_example;
```

Как это работает

```
CREATE FUNCTION zson_in(cstring)
    RETURNS zson
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT IMMUTABLE;

CREATE FUNCTION zson_out(zson)
    RETURNS cstring
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT IMMUTABLE;

CREATE TYPE zson (
    INTERNALLENGTH = -1,
    INPUT = zson_in,
    OUTPUT = zson_out,
    STORAGE = extended -- try to compress
);

CREATE FUNCTION jsonb_to_zson(jsonb)
    RETURNS zson
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT IMMUTABLE;

CREATE FUNCTION zson_to_jsonb(zson)
    RETURNS jsonb
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT IMMUTABLE;

CREATE CAST (jsonb AS zson) WITH FUNCTION jsonb_to_zson(jsonb) AS ASSIGNMENT;
CREATE CAST (zson AS jsonb) WITH FUNCTION zson_to_jsonb(zson) AS IMPLICIT;
```

Словарь в памяти

```c
#define DICT_MAX_WORDS (1 << 16)

typedef struct {
    uint16 code;
    bool check_next; // next word starts with the same nbytes bytes
    size_t nbytes; // number of bytes (not letters) except \0
    char* word;
} Word;

typedef struct {
    int32 dict_id;
    uint32 nwords;
    Word words[DICT_MAX_WORDS]; // sorted by .word, word -> code
    uint16 code_to_word[DICT_MAX_WORDS]; // code -> word index
} Dict;

typedef struct DictListItem {
    Dict* pdict;
    union {
        time_t last_clean_sec; // for first list item
        time_t last_used_sec; // for rest list items
    };
    struct DictListItem* next;
} DictListItem;
```

Схема кодирования

```
// VARHDRSZ
// zson_version [uint8]
// dict_version [uint32]
// decoded_size [uint32]
// hint [uint8 x PGLZ_HINT_SIZE]
// {
//skip_bytes [uint8]
//... skip_bytes bytes ...
//string_code [uint16], 0 = no_string
// } *
```

Также предусмотрено

- Переобучение (важно - кэш!)
- Удаление старых словарей

См README.md

Бенчмарки

- Получали до **+ 11.8% TPS** [1]
  - Если данные не влезают в память, документы большие
- На тех же данных **-5% TPS** если все влезло в память
- Был случай когда размер данных ***увеличился*** **на 19.4%**
- После `#define PGLZ_HINT_SIZE 0` **уменьшился на 10.4%**

Вывод: использовать нужно с умом, zson за вас сам все не сделает.

[1] см docs/benchmark.md

Ссылки по теме

- https://github.com/postgrespro/zson
- https://habr.ru/p/312006/

Postgres Professional

http://postgrespro.ru/

+7 495 150 06 91

info@postgrespro.ru

postgrespro.ru