


# NOSQL ВНУТРИ SQL

приземленные вопросы  
практического применения



Профессиональная конференция  
разработчиков высоконагруженных  
систем

 Дмитрий Долгов, Mindojo

 [github.com/erthalion](https://github.com/erthalion)

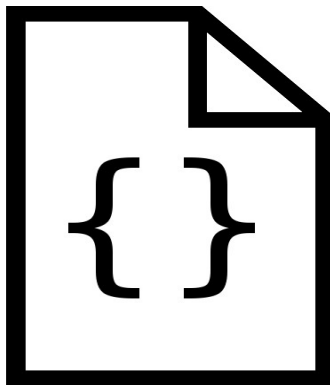
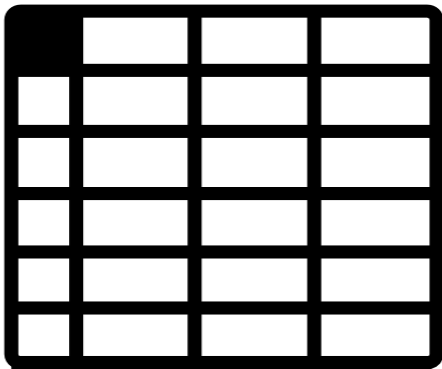
 @erthalion

 9erthalion6 at gmail dot com

# Данные

# Данные


# Данные



Данные нужно хранить в  
соответствующем формате:

Данные нужно хранить в  
соответствующем формате:

→ Отдельные хранилища,  
единый интерфейс


## Данные нужно хранить в соответствующем формате:

- Отдельные хранилища, единый интерфейс
- Единое хранилище, разные форматы





# Отдельные хранилища




## Отдельные хранилища

→ Конкретный формат обрабатывается  
наилучшим образом 

## Отдельные хранилища

- Конкретный формат обрабатывается наилучшим образом 
- Производительность, дублирование 

## Отдельные хранилища



- Конкретный формат обрабатывается наилучшим образом 
- Производительность, дублирование 
- Вопросы интеграции компонентов 

# Единое хранилище




## Единое хранилище

→ Не требует интеграции 

## Единое хранилище

- Не требует интеграции 
- Производительность, дублирование 

## Единое хранилище

- Не требует интеграции 
- Производительность, дублирование 
- Поддержка со стороны БД 





# КТО?

- Postgresql (hstore/json/jsonb)
- MySQL (json)
- Oracle
- MSSql
- db2

Легкий способ начать ~~бегать по~~  
~~утрам~~ использовать документы в  
реляционной базе

```
-- PG since 9.4
select jsonb_build_object(
    'id', 1,
    'data', 'aaa'
);

-- MySQL since 5.7
select json_object(
    'id', 1,
    'data', 'aaa'
);
```

```
-- PG since 9.4
select jsonb_agg(query) from (
    select id, data
    from jsonb_table
) query;
-- MySQL since 8
select json_objectagg('key', val)
as 'key_val' from t1;
```

-- PG

```
copy table_name(jsonb_column_name)
from 'data.json';
```

-- MySQL

```
load data infile 'data.json'
into table table_name (json_column_name);
```

- Загрузка дампа из внешних источников
- Некорректные данные с валидной структурой - json5
- Битые данные - ручное исправление, линтеры

# ПРОИЗВОДИТЕЛЬНОСТЬ



# Факторы

## Факторы

→ Структура данных на диске

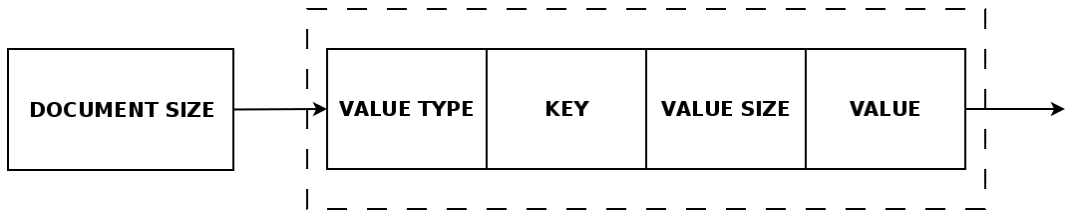
## Факторы

- Структура данных на диске
- Сериализация данных

## Факторы

- Структура данных на диске
- Сериализация данных
- Поддержка индексов

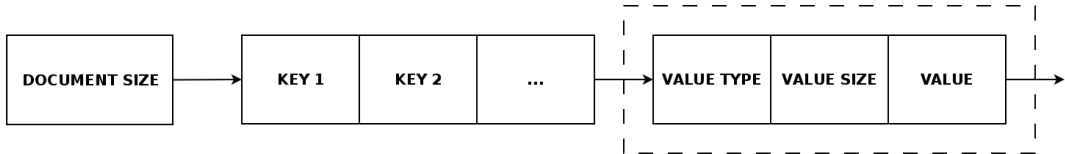
# Bson



```
bson.dumps({"a": 3, "b": u"xyz"})
```

```
\x17\x00\x00\x00\x10a\x00\x03\x00\x00\x00\x02b\x00\x04\x00\x00\x00xyz\x00\x00
```

# Jsonb



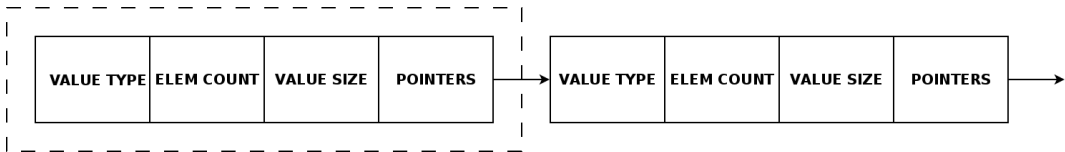
```
select pg_relation_filepath(oid),
relpages from pg_class
where relname = 'table_name';
```

```
pg_relation_filepath | relpages
-----+-----
base/40960/325477   |          0
(1 row)
```

```
\x10\x03\x00\x00\x00ab\x00\x00\x00\x00\x00\x00\x80\x03\x00xyz\x00\x00\x00\x00
```



# MySQL json



## Сериализация данных

- MongoDB - дерево Document -> Elements
- Postgresql - JsonbValue со списком элементов
- MySQL - ленивая структура с указателями

## Индексы

- MongoDB - индексы для полей
- Postgresql - общий индекс, индексы для полей
- MySQL - виртуальные колонки для индексирования

# ТЕСТИРОВАНИЕ



YCSB 0.8,  $10^6$

16GB memory, 4 core 2.3GHz

Postgresql 9.5.4

MongoDB 3.2.9

MySQL 5.7.9

AWS EC2 m4.xlarge

16GB memory, 4 core 2.3GHz

## Воспроизводимость

erthalion/YCSB

erthalion/ansible-ycsb

## Конфигурация

shared\_buffers

effective\_cache\_size

innodb\_buffer\_pool\_size

transaction\_sync, method



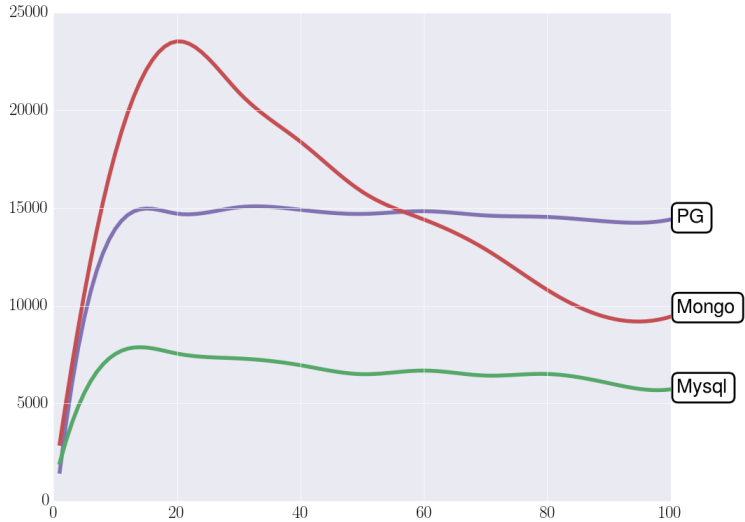
# Простая выборка по ключу с jsonb\_path\_ops индексом

”Маленький документ”

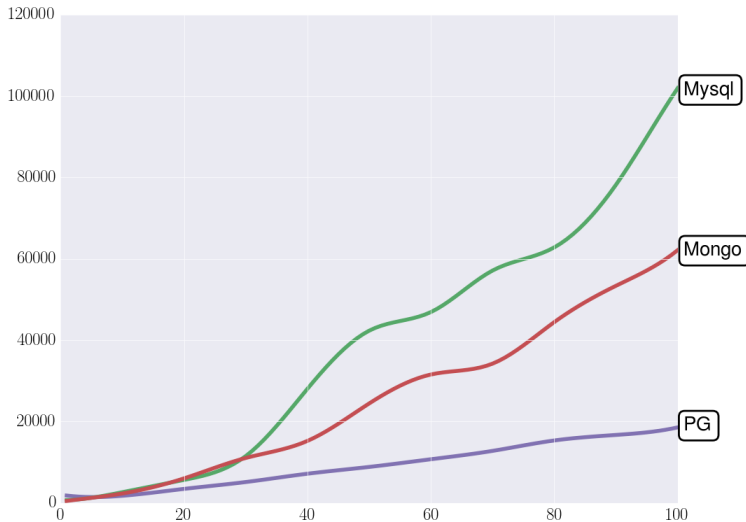
10 полей

без вложенности

## Throughput (ops/sec)



## Latency 99% ( $\mu s$ )



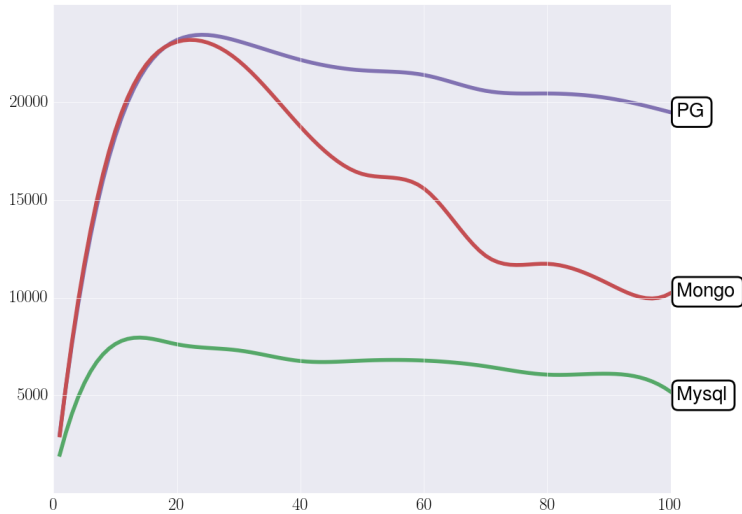
# Простая выборка по ключу с Btree индексом

”Маленький документ”

10 полей

без вложенности

## Throughput (ops/sec)



## Latency 99% ( $\mu s$ )

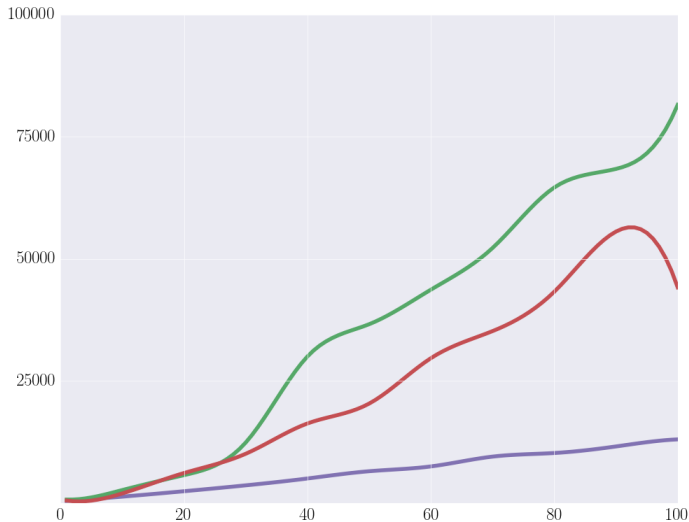
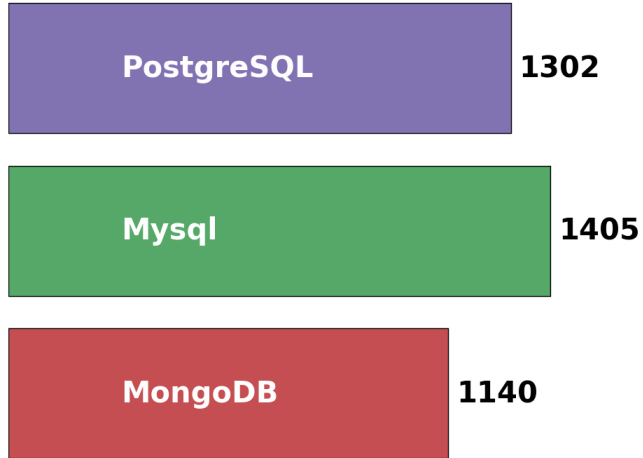
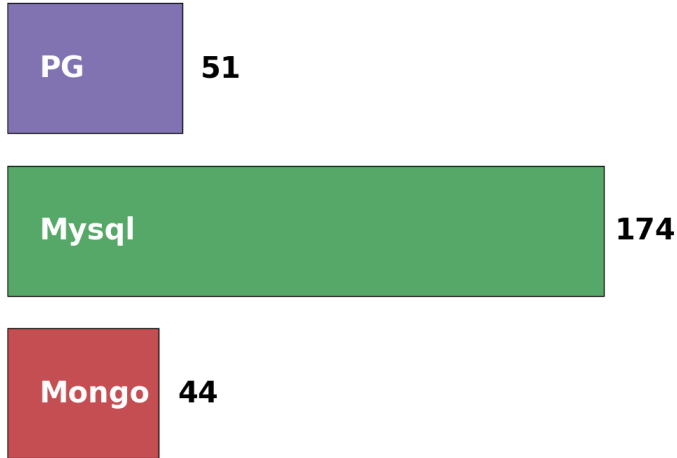


Table size (mb)



Index size (mb)





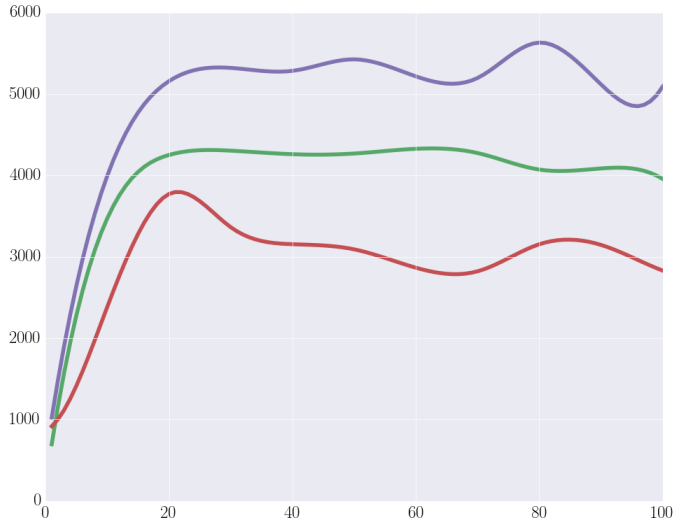
Выборка 50%, обновление 50%

"Маленький документ"

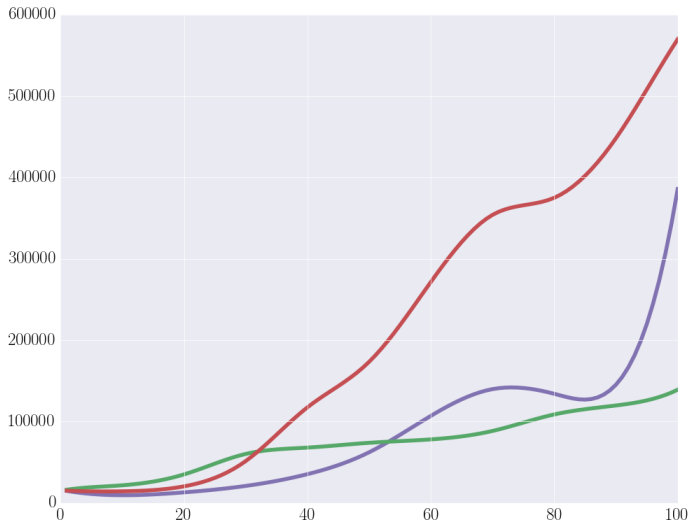
10 полей

без вложенности

## Throughput (ops/sec)



## Latency 99% ( $\mu s$ )



## Выборка 50%, обновление 50%

”Большой документ”

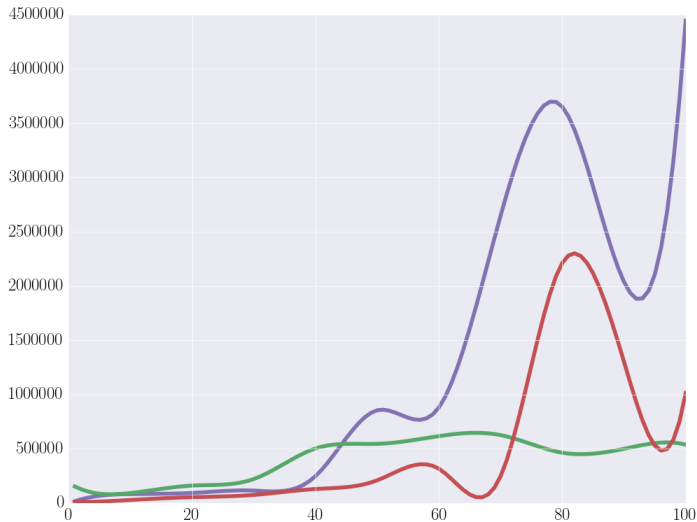
100 полей удвоенной длины

без вложенности

## Throughput (ops/sec)



## Latency 99% ( $\mu s$ )



# Вопросы?