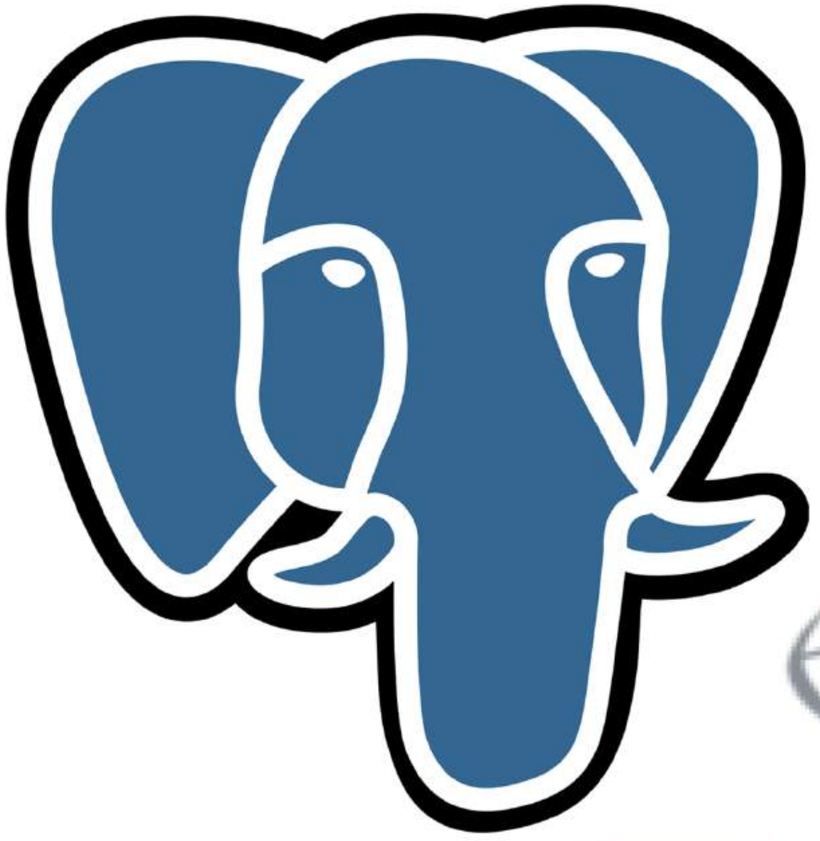
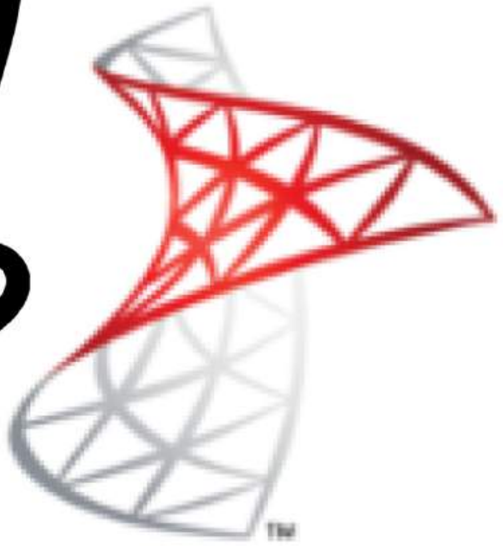
«Дух машины хранит
секреты древних.
Пренебрегая ритуалом,
ты отрекаешься от веры»

PG-укротитель



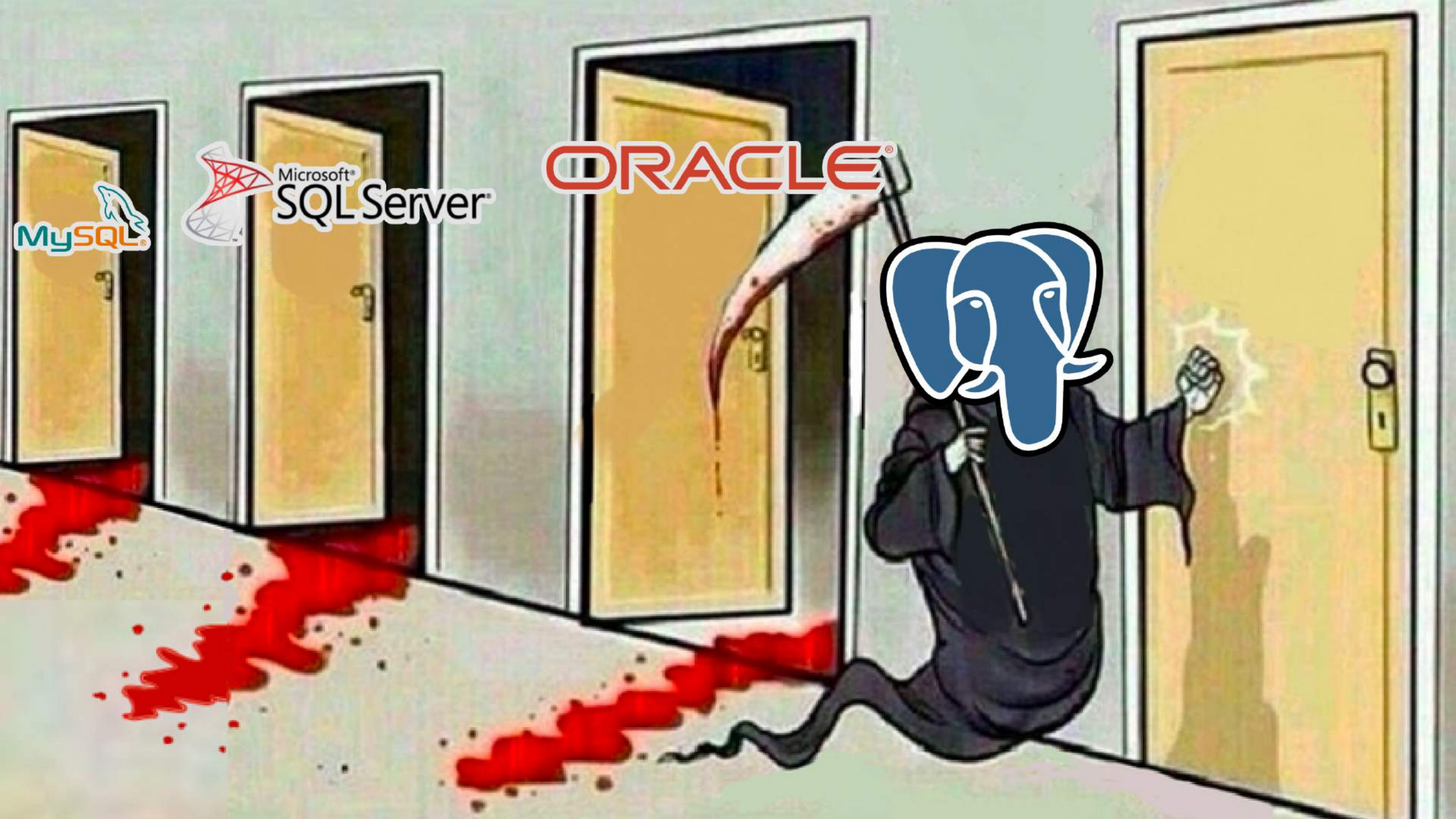


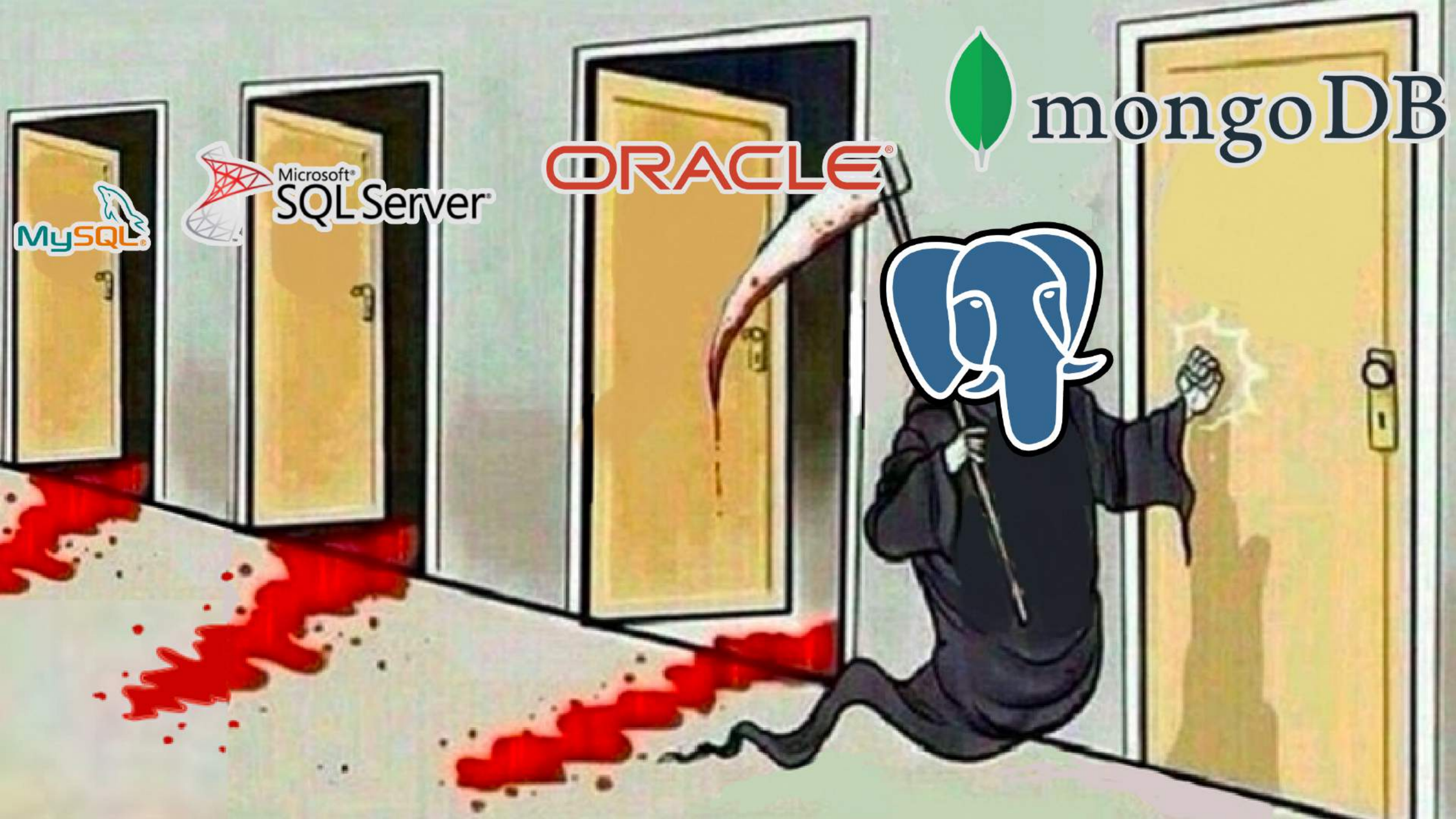
Microsoft®
SQL Server®

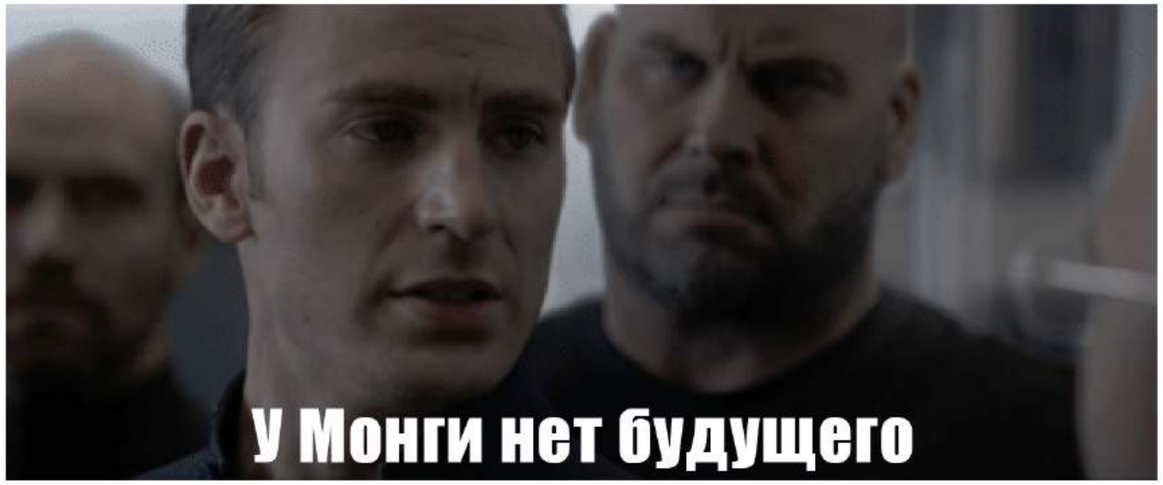
ORACLE®

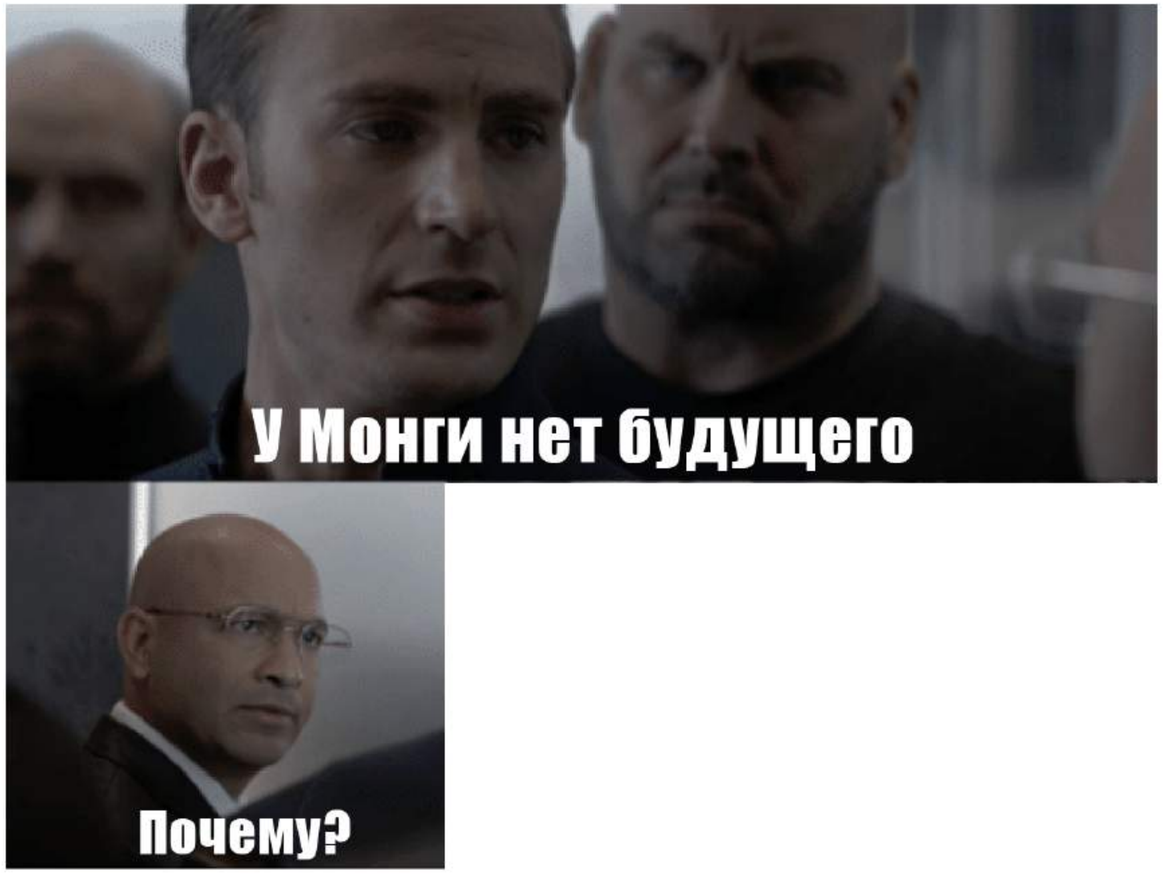
MySQL®

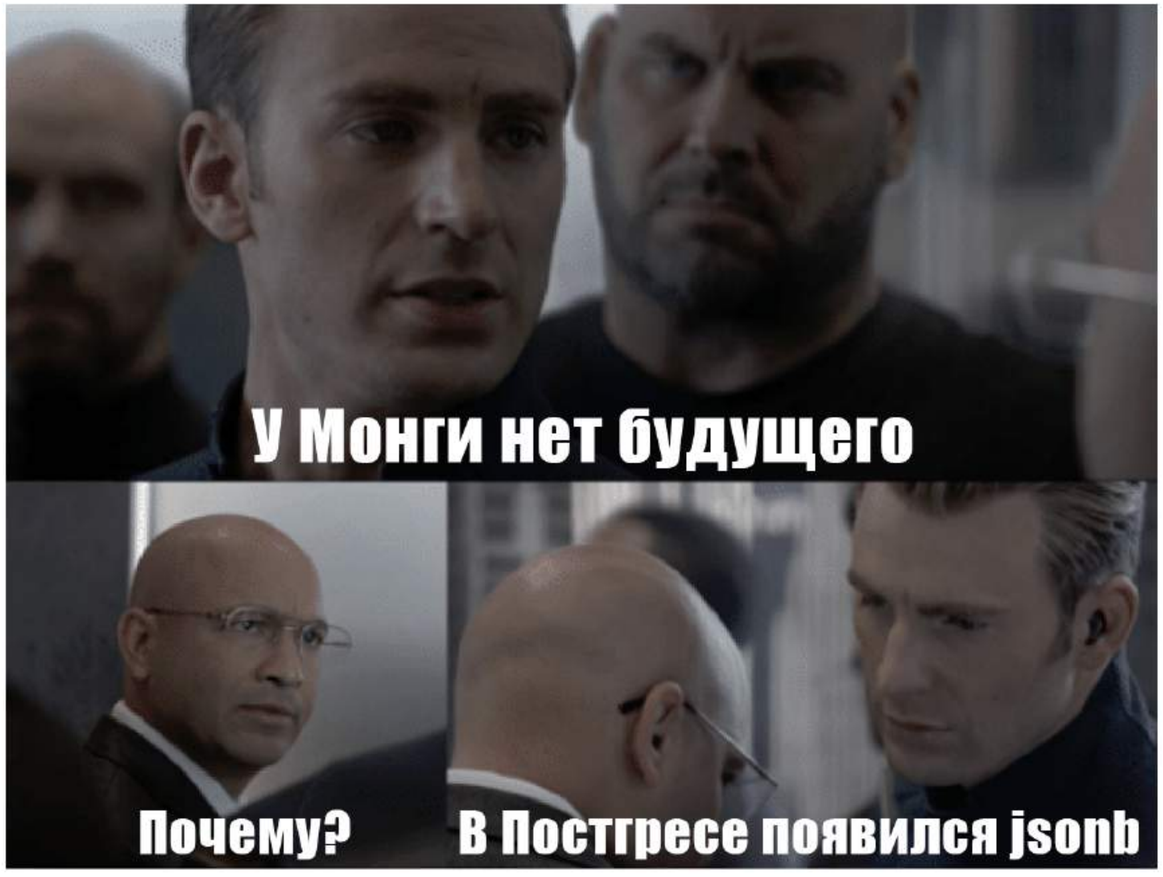

MySQL
Microsoft
SQL Server
ORACLE


MySQL
Microsoft SQL Server
ORACLE


MySQL
Microsoft SQL Server
ORACLE
mongoDB


У Монги нет будущего


У Монги нет будущего
Почему?


У Монги нет будущего
Почему?
В Постгресе появился jsonb


У Монги нет будущего
Почему?
В Постгресе появился jsonb

blogonlymail 8 months ago

Нужно руки оторвать тем, кто придумал в реляционную базу Json пихать. Открыли ящик пандоры для всяких лузер-разработчиков.

 2  **Reply**

blogonlymail 8 months ago

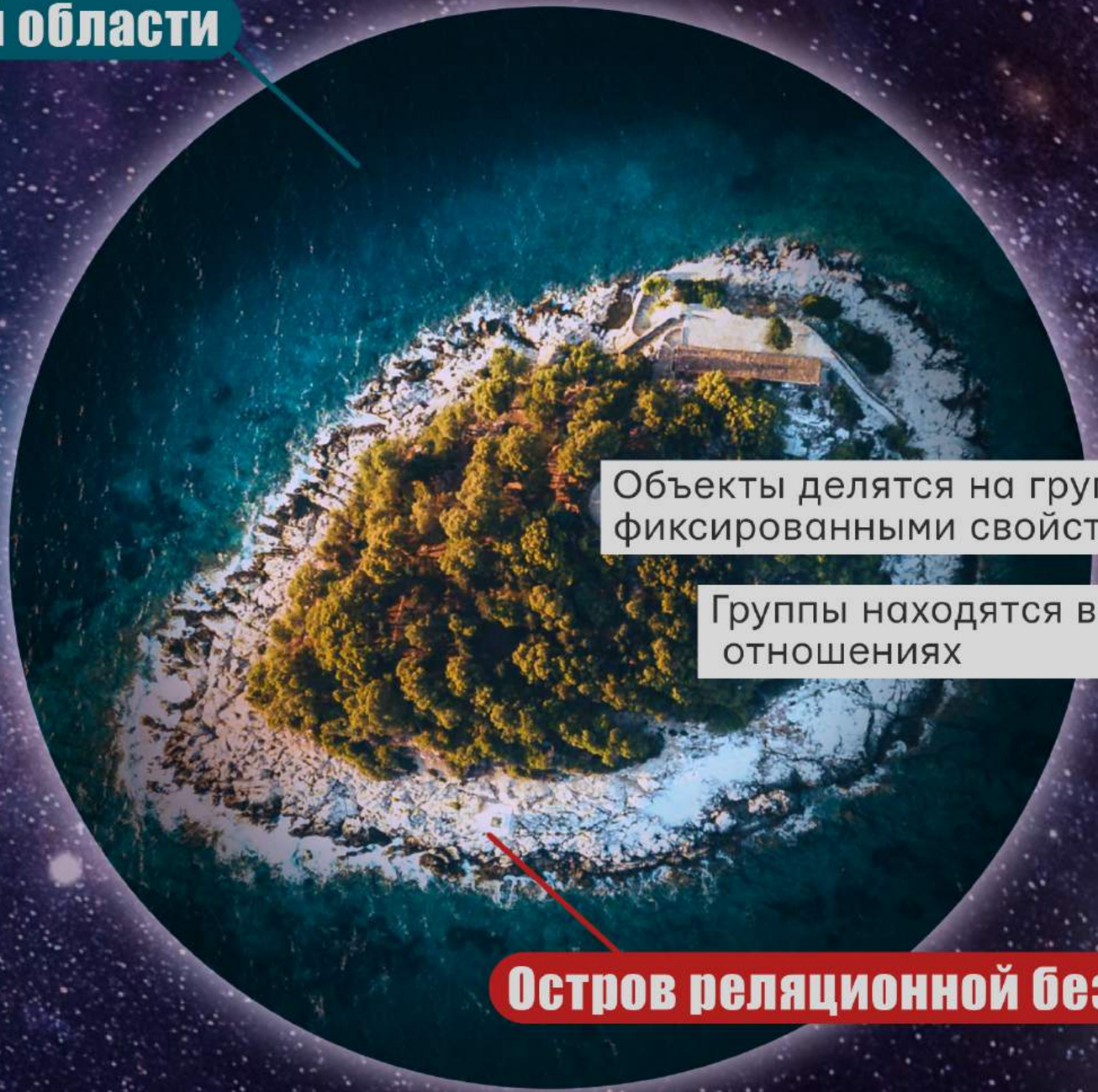
Нужно руки оторвать тем, кто придумал в реляционную базу Json пихать.
Открыли ящик пандоры для всяких лузер-разработчиков.

2 Reply


Океан предметной области


Океан предметной области
Остров реляционной безмятежности

Океан предметной области

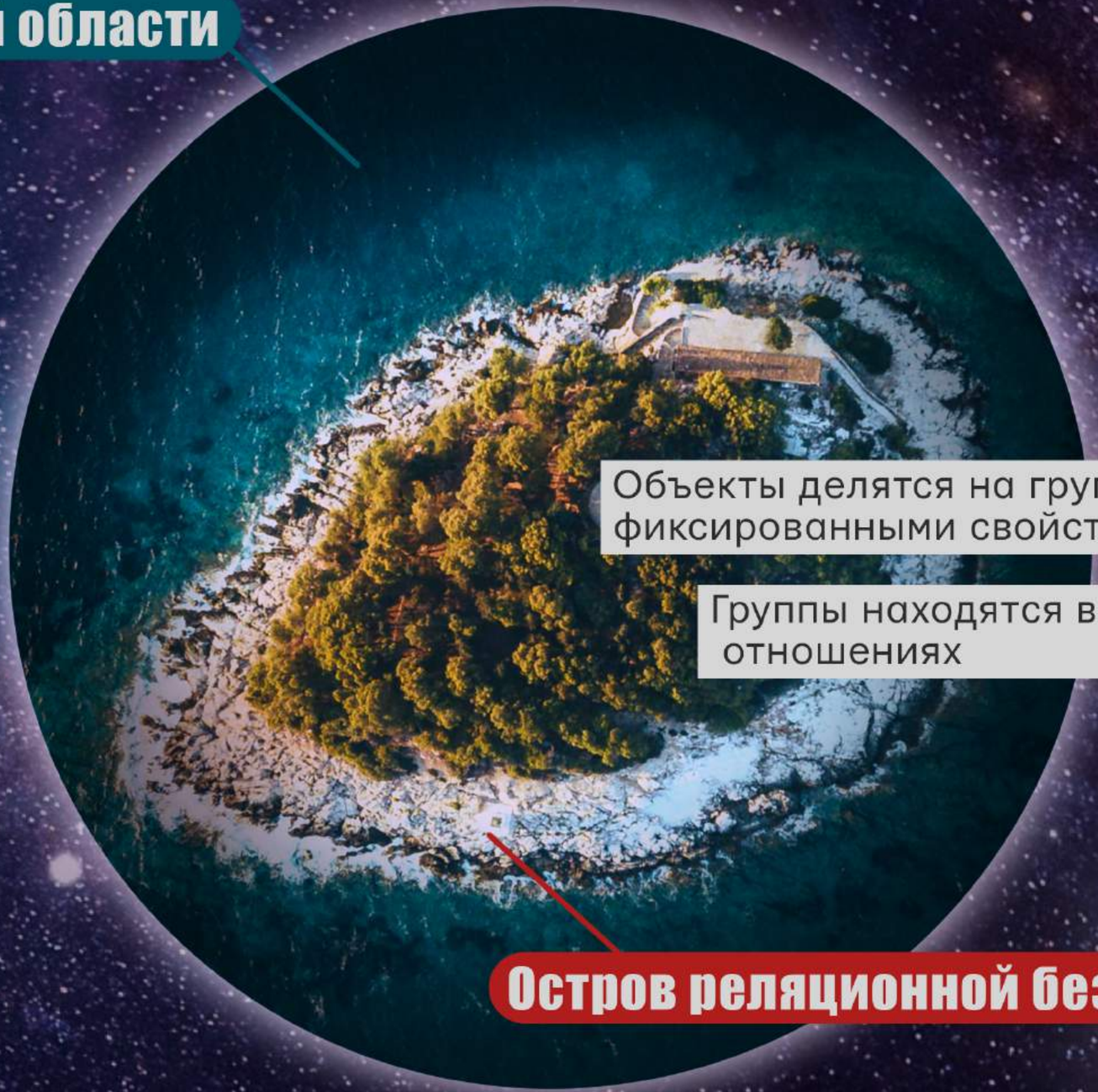


Объекты делятся на группы с фиксированными свойствами

Группы находятся в чётких отношениях

Остров реляционной безмятежности

Океан предметной области



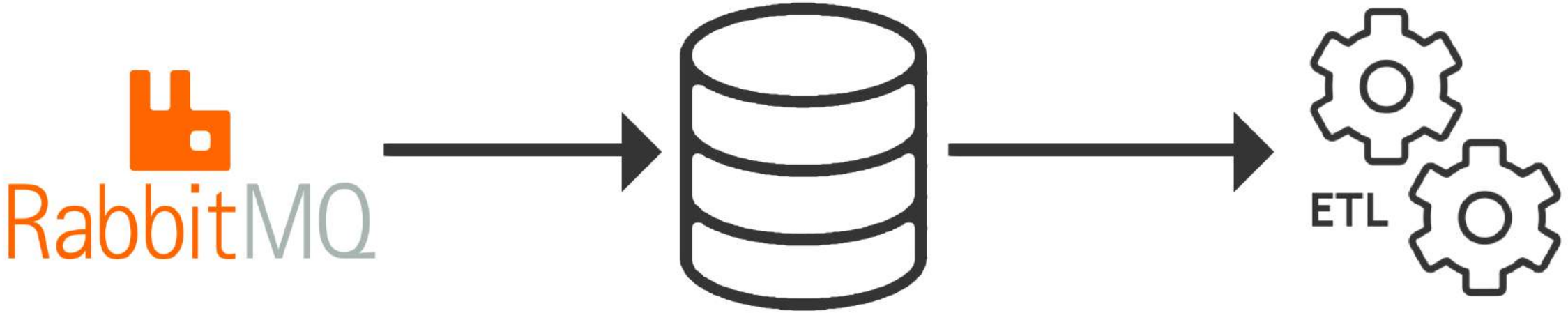
Объекты делятся на группы с фиксированными свойствами

Группы находятся в чётких отношениях

Остров реляционной безмятежности


Океан предметной области
Остров реляционной безмятежности


Океан предметной области
Пик произвольных событий
Остров реляционной безмятежности




```xml
<changeSet id="8" author="sazonovfm">
    <createTable tableName="events">
        <column name="event_id" type="varchar(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="payload" type="varchar"/>
    </createTable>
</changeSet>
```

```sql
create table events
(
    event_id varchar(36) not null
        primary key,
    payload  varchar
);
```


Океан предметной области
Пик произвольных событий
Остров реляционной безмятежности


Океан предметной области
Шельф составных групп
Пик произвольных событий
Остров реляционной безмятежности


Договоры
Договоры найма
Договоры займа

Table inheritance

Table inheritance

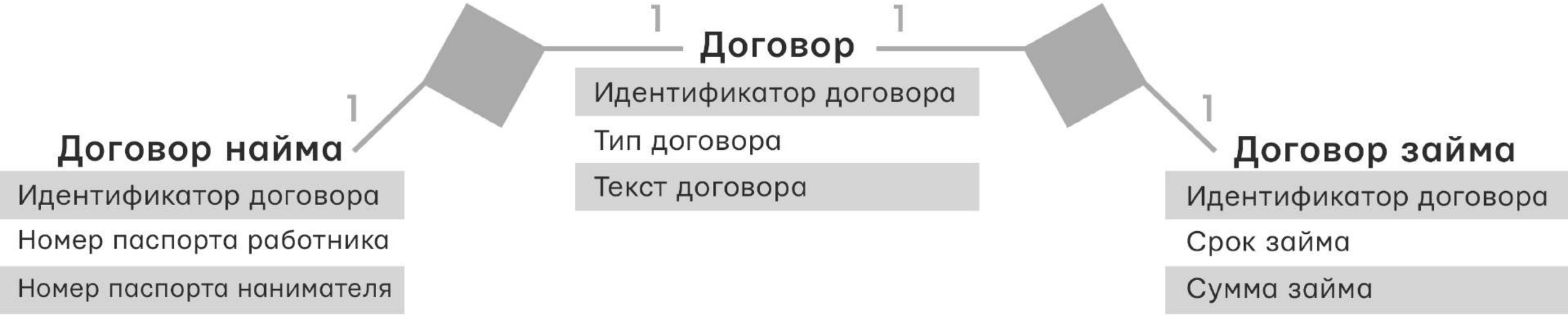
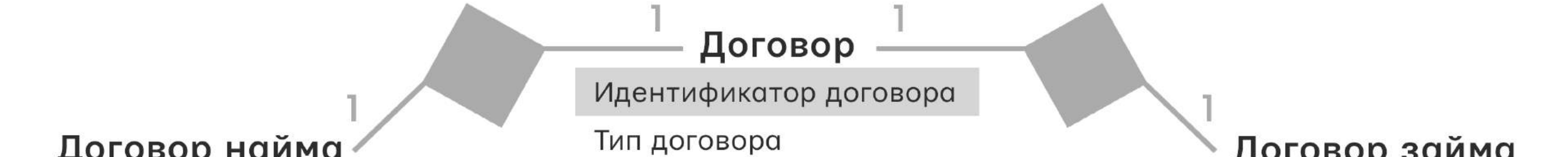


Table inheritance



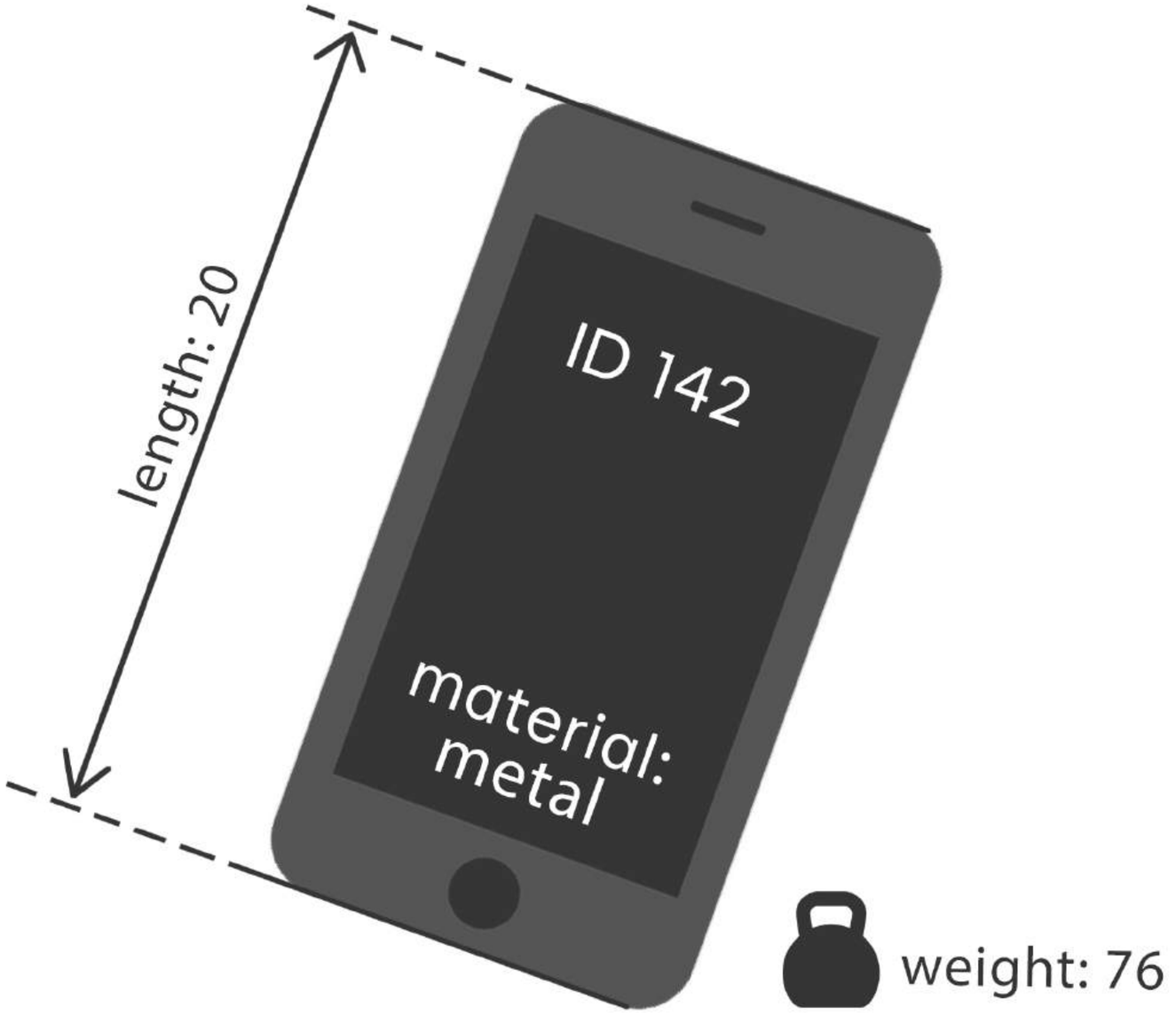

```
{
  "agreement_id": 12345,
  "agreement_type": "HIRE",
  "content": "hello",
  "employer_passport": 12345678,
  "employee_passport": 23456789
}
```

```
{
  "agreement_id": 1234,
  "agreement_type": "LOAN",
  "content": "hello",
  "amount": 789,
  "duration": 60
}
```


Океан предметной области
Шельф составных групп
Пик произвольных событий
Остров реляционной безмятежности

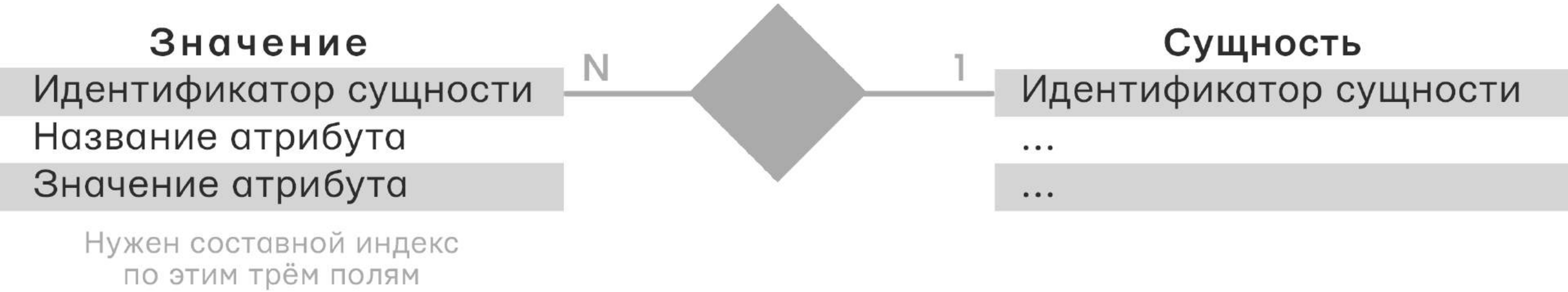

Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности

Каждый товар уникален



EAV (Entity Attribute Value)

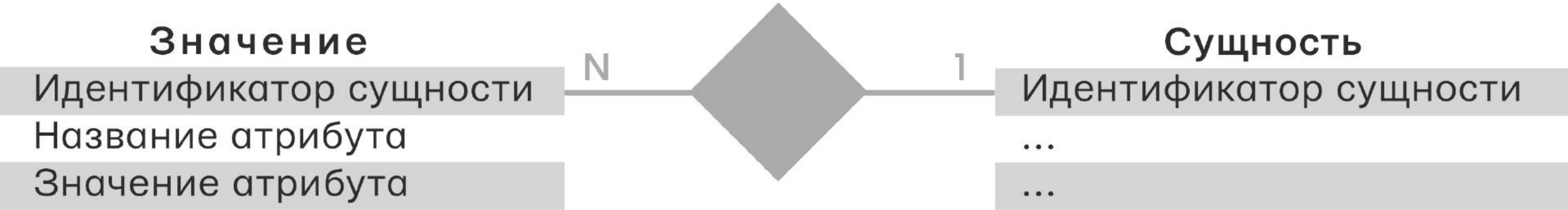
EAV (Entity Attribute Value)



Нужен составной индекс
по этим трём полям

DBA одобряет

EAV (Entity Attribute Value)



Нужен составной индекс
по этим трём полям

```
{
  "id": 1234,
  "length": 20,
  "weight": 76,
  "material": "metal"
}
```

DBA одобряет

EAV (Entity Attribute Value)

ID сущности	Название	Значение
123	weight	70
123	height	30
123	material	metal
123	age	21
123	city	Agraba
127	weight	70
127	height	30

Будет преобразован в JSON массив

≠

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

EAV (Entity Attribute Value)

ID сущности	weight	height	material	age	city
123	70	30	metal	21	Agraba
127	70	30	null	null	null

JSON friendly =

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

EAV (Entity Attribute Value)

БД

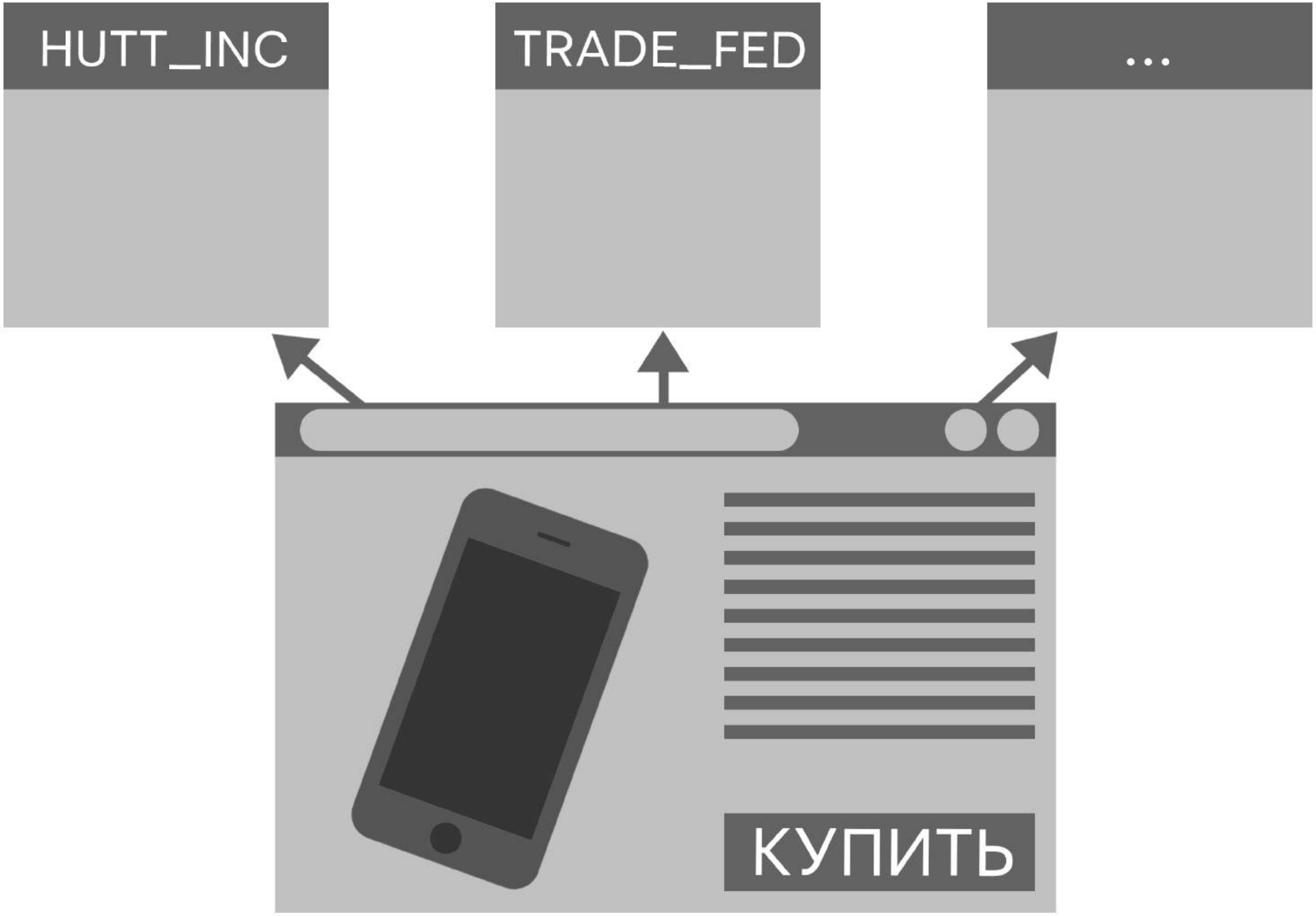
```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

FrontEnd

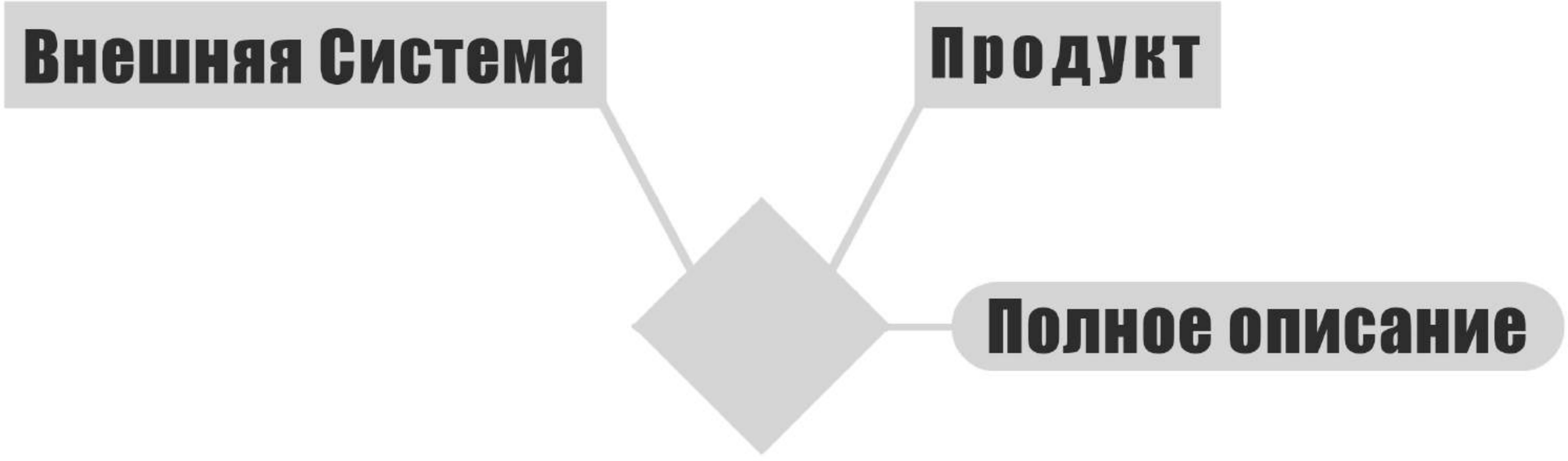
```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```


Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности


Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов


HUTT_INC
TRADE_FED
...
КУПИТЬ

Внешние коды продуктов



EAV

sys_id	product_id	ext_id
HUTT_INK	123	BEST_PHONE
TRADE_FED	123	ULTRA_PHONE

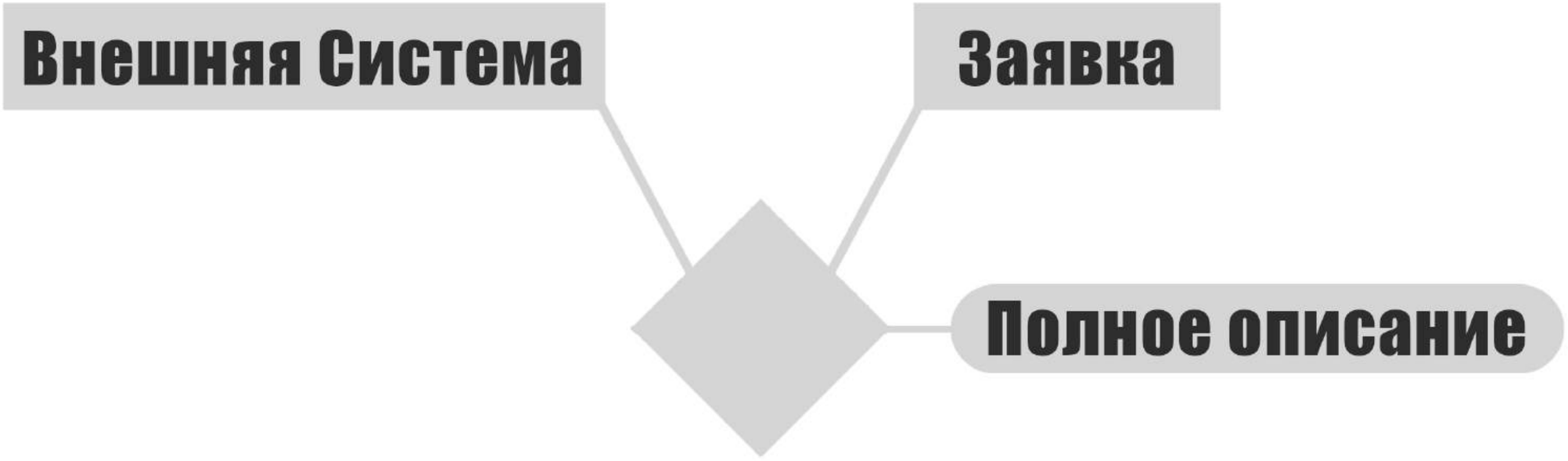
JSON

product_id	ext_codes
123	{ "HUTT_INC" : "BEST_PHONE", "TRADE_FED" : "ULTRA_PHONE" }


Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов


Океан предметной области
Риф полных описаний
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов

Полные описания заявок



EAV

app_id	attr_name	attr_val
143	smurf	km_232_f12

JSON

app_id	ext_data
143	CENSORED

Полные описания продуктов

EAV

app_id	attr_name	attr_val
143	smurf	km_232_f12

JSON

app_id	ext_data
143	CENSORED

Как хранятся большие объекты

PERSON

Name	About
Илья	Весёлый
Фёдор	Умный
Александр	Творческий
Оксана	Лучшая
Олег	Согласно «Повести временных лет», Олег был родичем (соплеменником) Рюрика. В.Н.Татищев со ссылкой на Иоакимовскую летопись считает его шурином — братом жены Рюрика, которую называет Ефандой[7]. Точное происхождение Олега в «Повести временных лет» не указывается. Предания, связанные с его личностью, сохранились также в полумифической...

10Mb

Мегабайты текста

Как хранятся большие объекты

PERSON

Name	About
Илья	Весёлый
Фёдор	Умный
Александр	Творческий
Оксана	Лучшая
Олег	

10Mb

TOAST

Согласно «Повести временных лет», (8 kb)
Олег был родичем (соплеменником) (8 kb)
Рюрика. В.Н.Татищев со ссылкой на (8 kb)
Иоакимовскую летопись считает его (8 kb)
шурином — братом жены Рюрика, (8 kb)
которую называет Ефандой[7]. Точное (8 kb)
происхождение Олега в «Повести (8 kb)
временных лет» не указывается. (8 kb)
Предания, связанные с его личностью, (8 kb)
сохранились также в полумифической (8 kb)

Мегабайты текста
(до одного гигабайта)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. During an UPDATE operation, values of unchanged fields are normally preserved as-is; so an UPDATE of a row with out-of-line values incurs no TOAST costs if none of the out-of-line values change.

The TOAST management code recognizes four different strategies for storing TOAST-able columns on disk:

- `PLAIN` prevents either compression or out-of-line storage; furthermore it disables use of single-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. During an UPDATE operation, values of unchanged fields are normally preserved as-is; so an UPDATE of a row with out-of-line values incurs no TOAST costs if none of the out-of-line values change.

The TOAST management code recognizes four different strategies for storing TOAST-able columns on disk:

- `PLAIN` prevents either compression or out-of-line storage; furthermore it disables use of single-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. Durin DATE of a row with out-

https://github.com/postgres/postgres/blob/master/src/include/access/heaptoast.h

```
#define TOAST_TUPLES_PER_PAGE   4

#define TOAST_TUPLE_THRESHOLD   MaximumBytesPerTuple(TOAST_TUPLES_PER_PAGE)

#define TOAST_TUPLE_TARGET      TOAST_TUPLE_THRESHOLD
```

The TOAST m sk:

- Pl gle-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

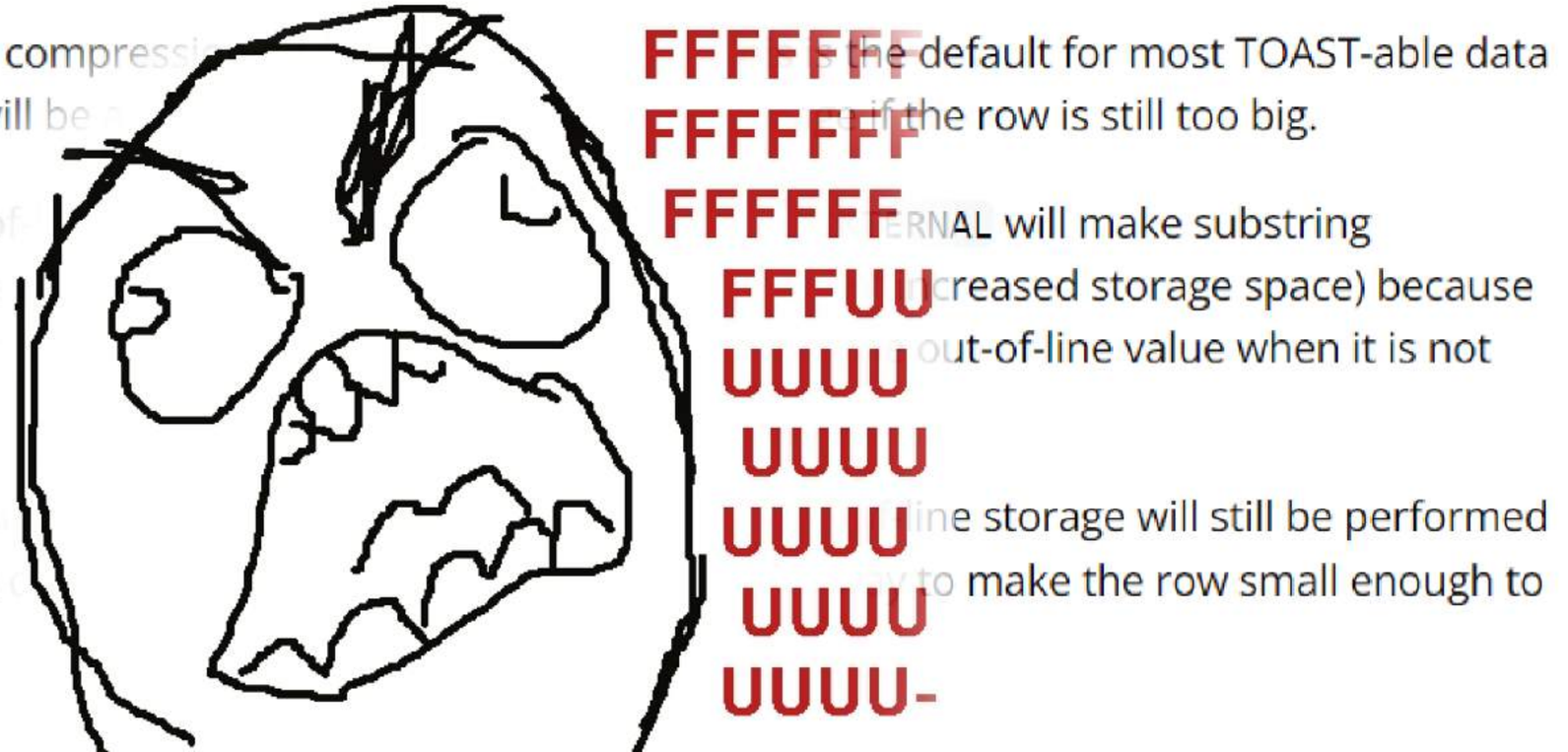
The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. Durin DATE of a row with out-

https://github.com/postgres/postgres/blob/master/src/include/access/heaptoast.h

```
#define TOAST_TUPLES_PER_PAGE   4

#define TOAST_TUPLE_THRESHOLD   MaximumBytesPerTuple(TOAST_TUPLES_PER_PAGE)

#define TOAST_TUPLE_TARGET      TOAST_TUPLE_THRESHOLD
```

The TOAST ma sk:

- Pl gle-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compressi the default for most TOAST-able data types. Compression will be if the row is still too big.
- `EXTERNAL` allows out-of- ERNAL will make substring operations on wide te creased storage space) because these operations are out-of-line value when it is not compressed.
- `MAIN` allows compress ine storage will still be performed for such columns, but to make the row small enough to fit on a page.)



Как хранить JSON в базе данных

Name

Author

Rating

Country

Name

Author

Rating

Country

name	author	country	rating
Empire V	Pelevin	Russia	5
Lord of the rings	Tolkien	Britain	6
Invincible	Lem	Poland	5
Worm	Wildbow	Canada	4
Same old story	Goncharov	Russia	5
Harry Potter	Rawling	Britain	3
Clean Code	Martin	USA	5
Spectre general	Cogswell	USA	4
Diaspora	Egan	Australia	4
Harry Potter	Yudkovski	USA	5

Name

Author

Rating

5

Country

name	author	country	rating
Empire V	Pelevin	Russia	5
Lord of the rings	Tolkien	Britain	6
Invincible	Lem	Poland	5
Worm	Wildbow	Canada	4
Same old story	Goncharov	Russia	5
Harry Potter	Rawling	Britain	3
Clean Code	Martin	USA	5
Spectre general	Cogswell	USA	4
Diaspora	Egan	Australia	4
Harry Potter	Yudkovski	USA	5

name = null
author = null
country = null
rating = 5

Name

Author

Pelevin

Rating

5

Country

name	author	country	rating
Empire V	Pelevin	Russia	5
Lord of the rings	Tolkien	Britain	6
Invincible	Lem	Poland	5
Worm	Wildbow	Canada	4
Same old story	Goncharov	Russia	5
Harry Potter	Rawling	Britain	3
Clean Code	Martin	USA	5
Spectre general	Cogswell	USA	4
Diaspora	Egan	Australia	4
Harry Potter	Yudkovski	USA	5

name = null
author = Pelevin
country = null
rating = 5

Name

Empire V

Author

Pelevin

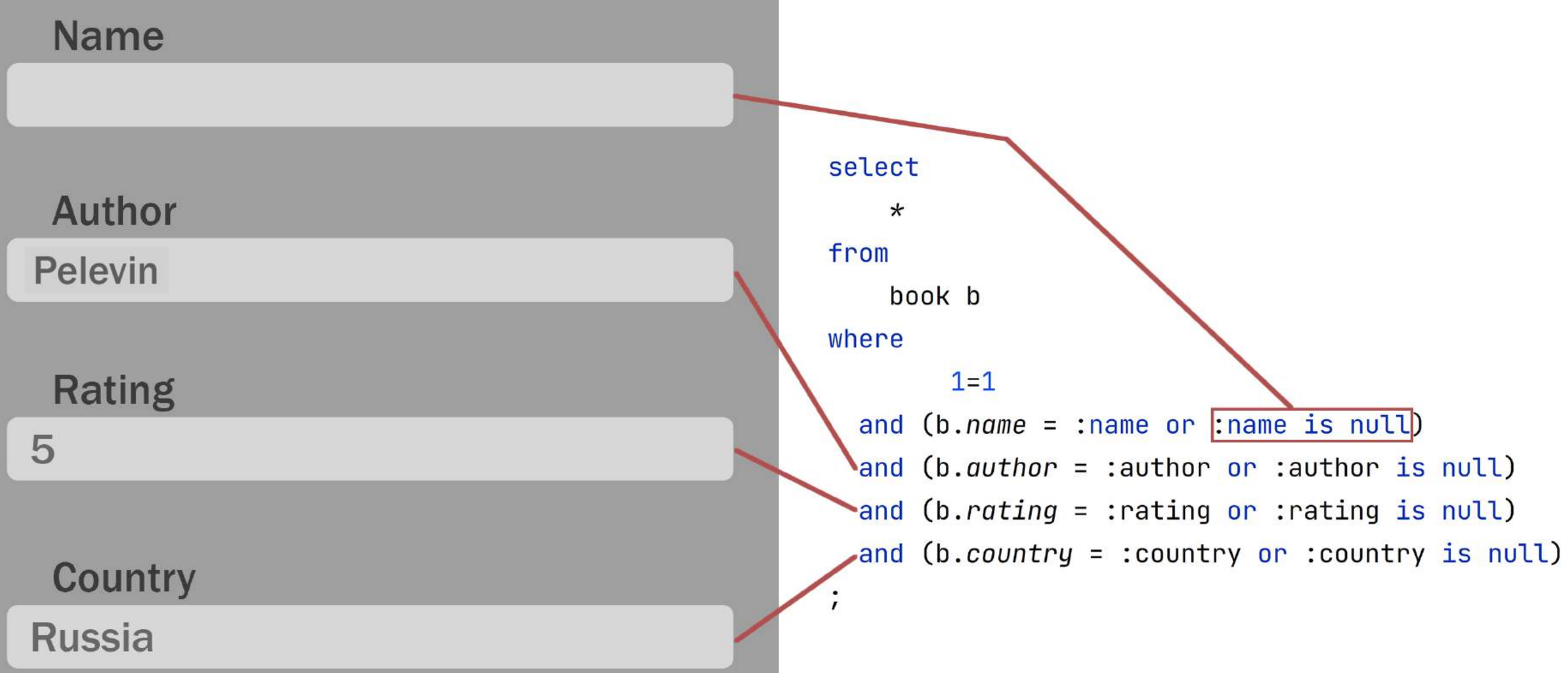
Rating

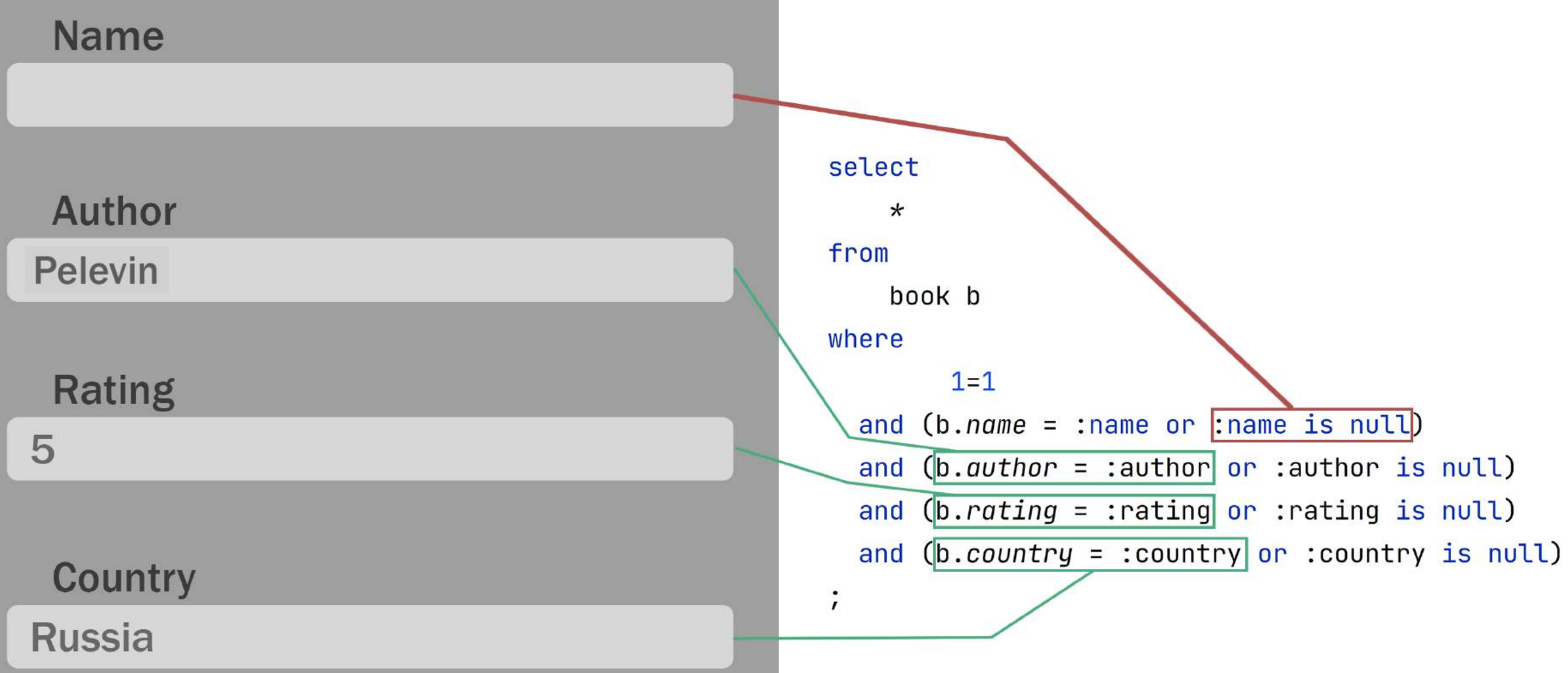
5

Country

Russia

```
select
    *
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```


Name
Author
Pelevin
Rating
5
Country
Russia
select
*
from
book b
where
1=1
and (b.name = :name or :name is null)
and (b.author = :author or :author is null)
and (b.rating = :rating or :rating is null)
and (b.country = :country or :country is null)
;


Name
Author
Pelevin
Rating
5
Country
Russia
select
*
from
book b
where
1=1
and (b.name = :name or :name is null)
and (b.author = :author or :author is null)
and (b.rating = :rating or :rating is null)
and (b.country = :country or :country is null)
;

Прекрасный рефакторинг

Было:

```
String sql = " select b from Book b where 1=1  "
  + (filter.getAuthor() == null ? "" :
  " and b.author = :author  "
)
  + (filter.getCountry() == null ? "" :
  " and b.country = :country"
)
  + (filter.getRating() == null ? "" :
  " and b.rating = :rating  "
)
  + (filter.getName() == null ? "" :
  " and b.name = :name      "
);
```

Стало:

```
" select b from Book b where            " +
"   1=1                                 " +
" and (b.author = :#{#filter.author}    " +
"   or :#{#filter.author} is null)      " +
" and (b.rating = :#{#filter.rating}    " +
"   or :#{#filter.rating} is null)      " +
" and (b.country = :#{#filter.country}  " +
"   or :#{#filter.country} is null)     " +
" and (b.name = :#{#filter.name}        " +
"   or :#{#filter.name} is null)        "
```

Краткость - сестра таланта

Прекрасный рефакторинг

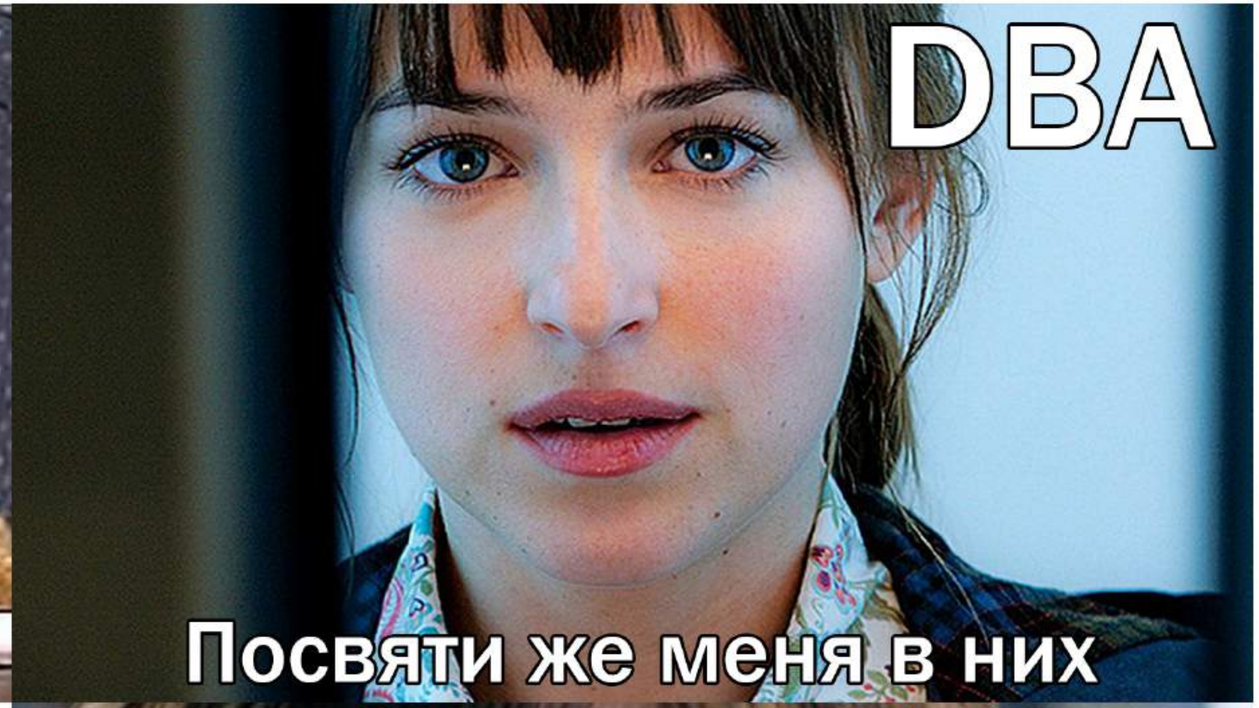
Было:

```java
@Override
public List<Book> findByFilterJpqlTypical(BookFilter filter) {
    if (filter.isEmpty()) {
        return Collections.emptyList();
    }
    String sql = " select b from Book b where 1=1 "
            + (filter.getAuthor() == null ? "" :
            " and b.author = :author "
    )
            + (filter.getCountry() == null ? "" :
            " and b.country = :country"
    )
            + (filter.getRating() == null ? "" :
            " and b.rating = :rating "
    )
            + (filter.getName() == null ? "" :
            " and b.name = :name "
    );

    final var em = emf.createEntityManager();
    final var query = em.createQuery(sql, Book.class);

    if (filter.getAuthor() != null) {
        query.setParameter("author", filter.getAuthor());
    }
    if (filter.getCountry() != null) {
        query.setParameter("country", filter.getCountry());
    }
    if (filter.getName() != null) {
        query.setParameter("name", filter.getName());
    }
    if (filter.getRating() != null) {
        query.setParameter("rating", filter.getRating());
    }

    return query.getResultList();
}
```

Стало:

```
" select b from Book b where                " +
"   1=1                                     " +
" and (b.author = :#{#filter.author}        " +
"   or :#{#filter.author} is null)          " +
" and (b.rating = :#{#filter.rating}        " +
"   or :#{#filter.rating} is null)          " +
" and (b.country = :#{#filter.country}      " +
"   or :#{#filter.country} is null)         " +
" and (b.name = :#{#filter.name}            " +
"   or :#{#filter.name} is null)            "
```

Краткость - сестра таланта




```
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
```

3.3. Ignoring *null* Parameters Using the *@Query* Annotation

We can avoid creating additional methods by using the *@Query* annotation and adding a small complication to the JPQL statement:

```
@Query("SELECT c FROM Customer c WHERE (:name is null or c.name = :name) and (:email is null"
  + " or c.email = :email)")
List<Customer> findCustomerByNameAndEmail(@Param("name") String name, @Param("email") String
email);
```

Notice that if the: *email* parameter is *null*:

```
:email is null or s.email = :email
```

Then the clause is always *true* and so doesn't influence the whole *WHERE* clause.

Let's make sure that this works:

```
List<Customer> customers = repository.findCustomerByNameAndEmail("D", null);

assertEquals(2, customers.size());
```

```
where (? is null or customer0_.name=?) and (? is null or customer0_.email=?)
```

With this method, we are putting trust in the database server to recognize the clause regarding our query parameter being *null* and optimize the execution plan of the query so that it doesn't have a significant performance overhead. For some queries or database servers, especially involving a huge table scan, there could be a performance overhead.

```
where (? is null or customer0_.name=?) and (? is null or customer0_.email=?)
```

With this method, we are putting trust in the database server to recognize the clause regarding our query parameter being *null* and optimize the execution plan of the query so that it doesn't have a significant performance overhead. For some queries or database servers, especially involving a huge table scan, there could be a performance overhead.

App

```
@Query("" +
 " select b from Book b where                " +
 "   1=1                                     " +
 " and (b.author = :#{#filter.author}        " +
 "   or :#{#filter.author} is null)          " +
 " and (b.rating = :#{#filter.rating}        " +
 "   or :#{#filter.rating} is null)          " +
 " and (b.country = :#{#filter.country}      " +
 "   or :#{#filter.country} is null)         " +
 " and (b.name = :#{#filter.name}            " +
 "   or :#{#filter.name} is null)            "
) List<Book> findByFilterOrIsNull(@Param("filter")
                                  BookFilter filter);
```

name = null
author = Pelevin
country = Russia
rating = 5

1

2

DB

Планы запросов

3

```
select
    *
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```

name	author	country	rating
Empire V	Pelevin	Russia	5
Lord of the rings	Tolkien	Britain	6
Invincible	Lem	Poland	5
Worm	Wildbow	Canada	4
Same old story	Goncharov	Russia	5
Harry Potter	Rawling	Britain	3
Clean Code	Martin	USA	5
Spectre general	Cogswell	USA	4
Diaspora	Egan	Australia	4
Harry Potter	Yudkovski	USA	5

name = null
author = null
country = null
rating = 5

DB

Планы запросов

```
select
    *
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```

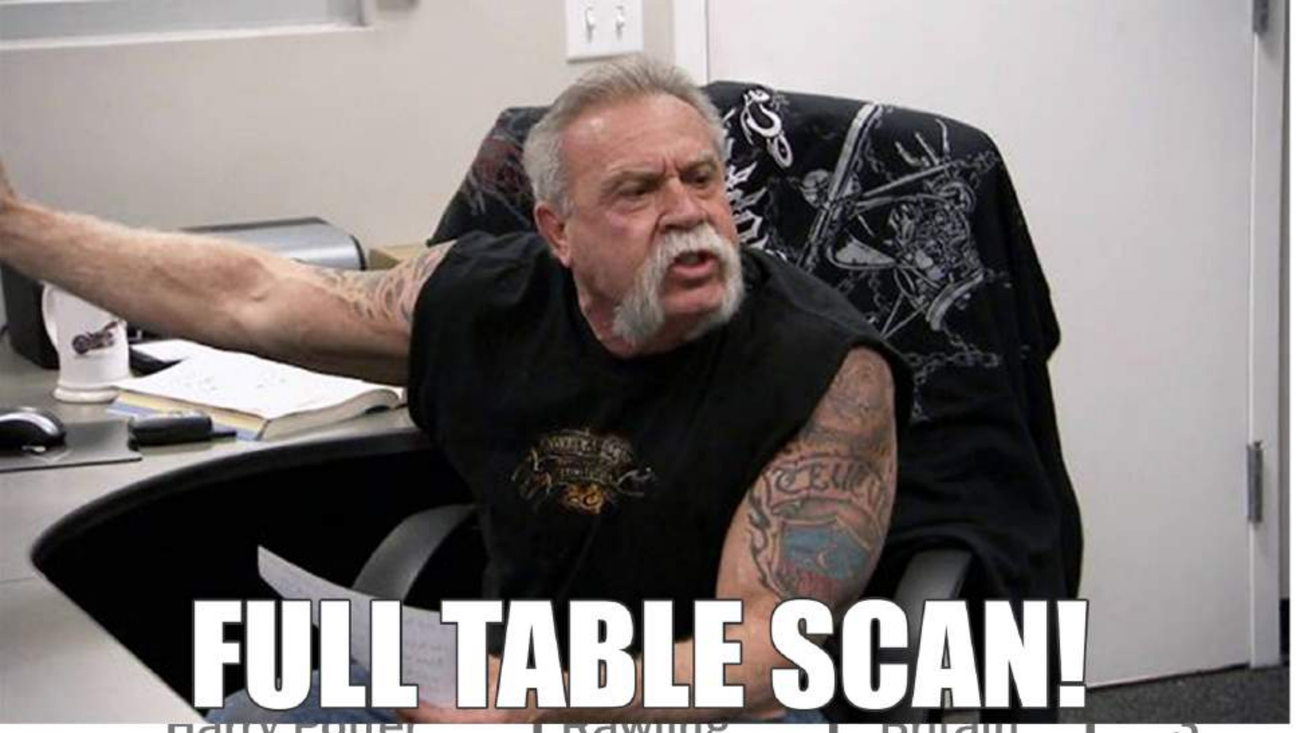
name	author	country	rating
Lord of the rings	Tolkien	Britain	6
Empire V	Pelevin	Russia	5
Invincible	Lem	Poland	5
Clean Code	Martin	USA	5
Same old story	Goncharov	Russia	5
Harry Potter	Yudkovski	USA	5
Worm	Wildbow	Canada	4
Spectre general	Cogswell	USA	4
Diaspora	Egan	Australia	4
Harry Potter	Rawling	Britain	3

name = null
author = null
country = null
rating = 5

DB

Планы запросов

```
select
    *
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```



Harry Potter | Rawling | Britain | 3

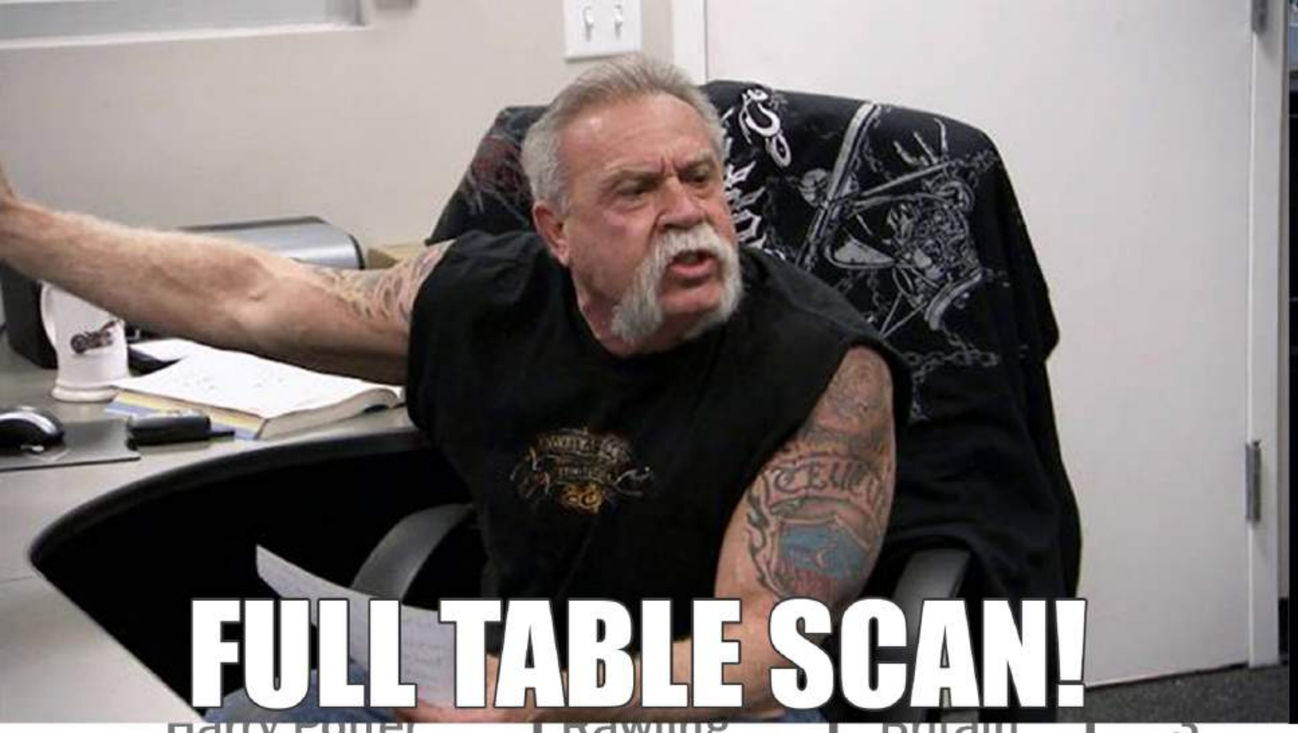
индекс

name = null
author = null
country = null
rating = 5

DB

Планы запросов

```
select
    *
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```

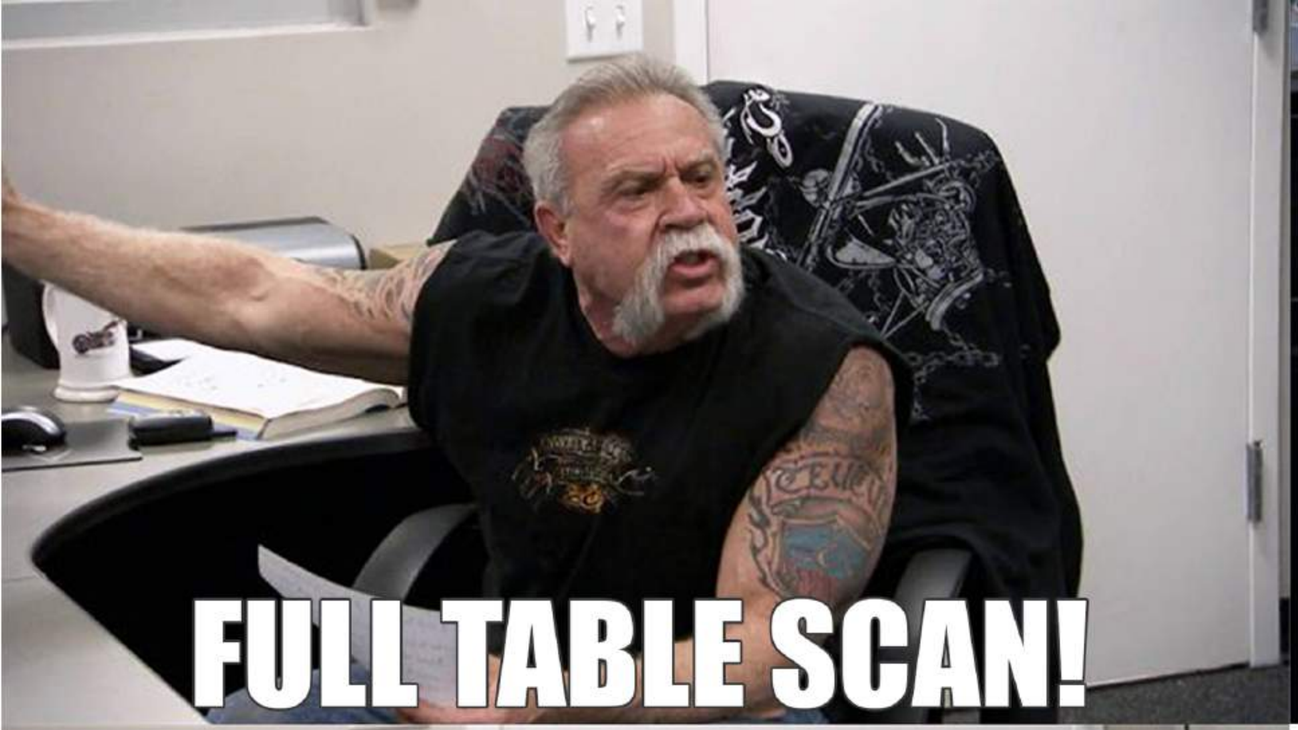


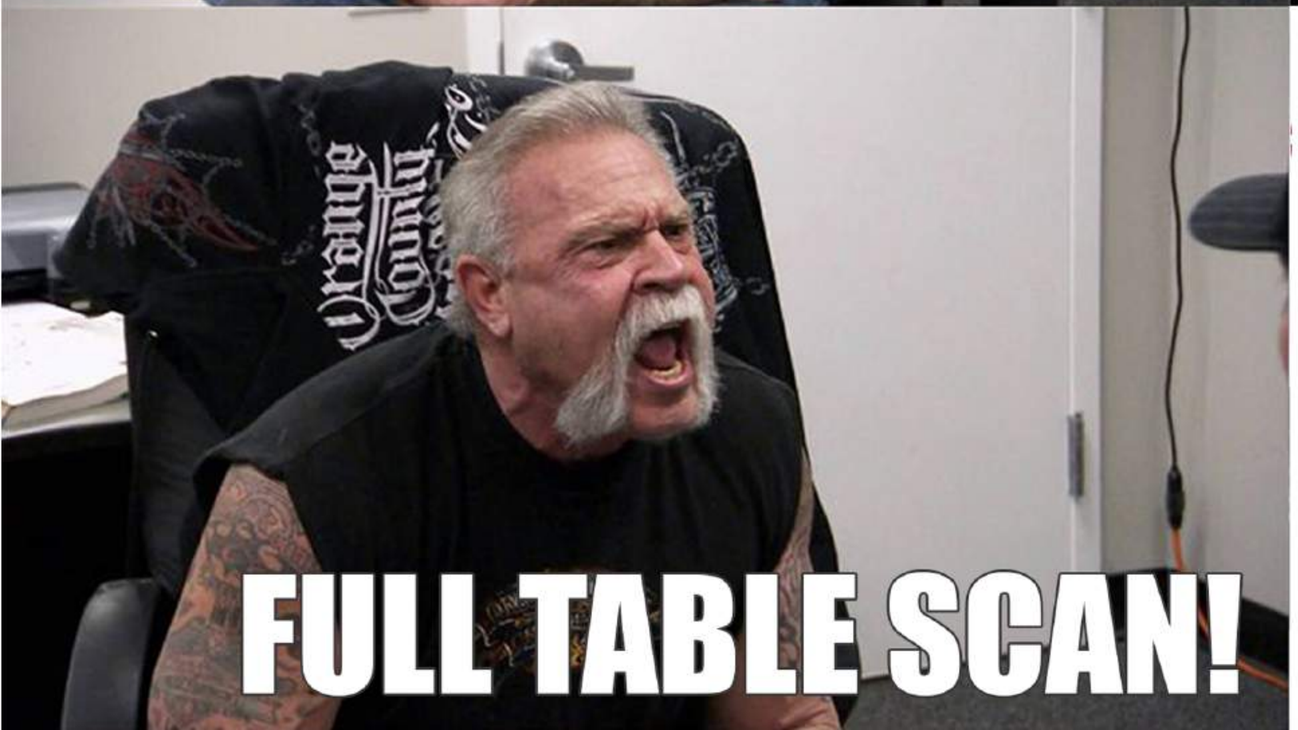
Harry Potter | Rawling | Britain | 3

индекс

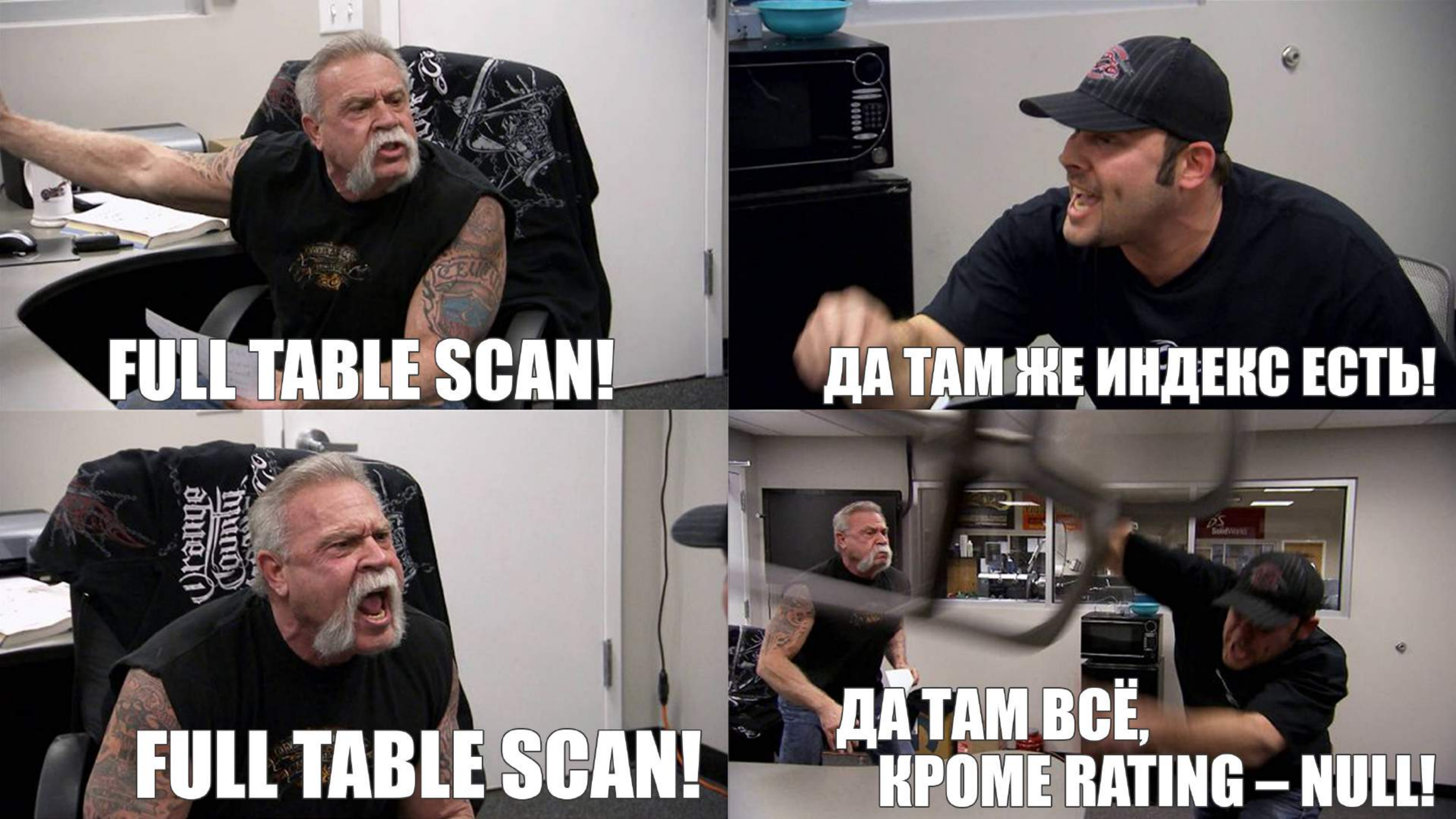
name = null
author = null
country = null
rating = 5

```
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```

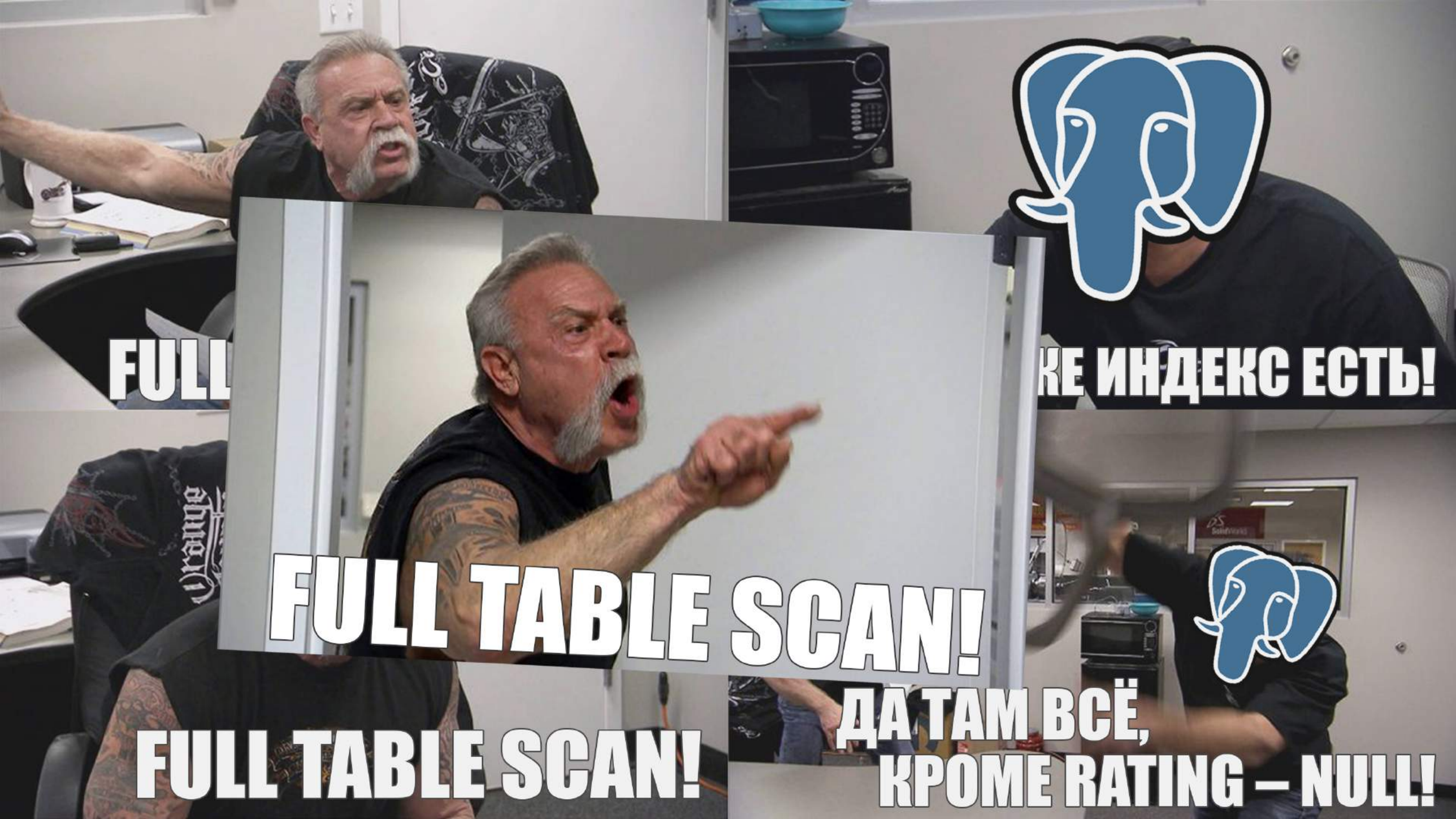





```
from
    book b
where
        1=1
  and (b.name = :name or :name is null)
  and (b.author = :author or :author is null)
  and (b.rating = :rating or :rating is null)
  and (b.country = :country or :country is null)
;
```


FULL TABLE SCAN!
ДА ТАМ ЖЕ ИНДЕКС ЕСТЬ!
FULL TABLE SCAN!
ДА ТАМ ВСЁ,
КРОМЕ RATING – NULL!


FULL
КЕ ИНДЕКС ЕСТЬ!
FULL TABLE SCAN!
FULL TABLE SCAN!
ДА ТАМ ВСЁ,
КРОМЕ RATING – NULL!


FULL
КЕ ИНДЕКС ЕСТЬ!
FULL TABLE SCAN!
FULL TABLE SCAN!
ДА ТАМ ВСЁ,
КРОМЕ RATING – NULL!


FULL
КЕ ИНДЕКС ЕСТЬ!
Oracle
FULL TABLE SCAN!
FULL TABLE SCAN!
ДА ТАМ ВСЁ,
КРОМЕ RATING – NULL!

Table 1-6 Optimizer Features for Oracle Database 11g Releases

Features	11.1.0.6	11.1.0.7	11.2.0.1	11.2.0.2	11.2.0.3	11.2.0.4
Adaptive cursor sharing	X	X	X	X	X	X
Join predicate pushdown			X	X	X	X
Use extended statistics to estimate selectivity						X
Use native implementation for full outer joins						
Partition pruning using join filtering	X	X	X			
Group by placement optimization	X	X	X	X	X	X
Null aware antijoins	X	X	X	X	X	X
Join predicate pushdown	X	X	X	X		
Join Factorization			X	X		
Cardinality Feedback			X	X	X	
Subquery Unnesting			X	X	X	X
Subquery Coalescing			X	X	X	X
Table Expansion			X	X	X	X
Filtering Join Elimination			X	X	X	X
Dynamic statistics enhancements						X

Table 1-7 Optimizer Features for Oracle Database 12c and Later Releases

Features	12.1.0.1	12.1.0.2	12.2.0.1	18c	19c
All optimizer features listed in Table 1-6	X	X	X	X	X
Adaptive Query Optimization	X	X	X	X	X
Online statistics gathering for bulk loads	X	X	X	X	X
Session level statistics for Global Temporary Tables	X	X	X	X	X
Multi-table left outer joins	X	X	X	X	X
Lateral views	X	X	X	X	X
Batch table access by rowid	X	X	X	X	X
Null accepting semi joins	X	X	X	X	X
Scalar subquery unnesting	X	X	X	X	X
Conversion of joins that produce unnecessary duplicates to semi-joins	X	X	X	X	X
[illegible]	X	X	X	X	X
[illegible]	X	X	X	X	X
[illegible]			X	X	X
Support [illegible] acle Database In-Memory			X	X	X
[illegible] [illegible] imination		X	X	X	X
Query [illegible] query processing			X	X	X
[illegible]			X	X	X
Support for shard[illegible] eases			X	X	X
Expression tracking			X	X	X
Space-saving algorithm for partition synopses			X	X	X
Oracle In-Memory Database statistics			X	X	X
Support for sharding			X	X	X
Cost-based OR expansion			X	X	X
Sub-query elimination			X	X	X
Multi-column key join elimination			X	X	X
SQL Quarantine					X
Gathering and use of real-time statistics					X
Use of automatic indexes					X

Немного настроек oracle

20.7. Query Planning

20.7.1. Planner Method Configuration
20.7.2. Planner Cost Constants
20.7.3. Genetic Query Optimizer
20.7.4. Other Planner Options

20.7.1. Planner Method Configuration

These configuration parameters pr... method of influencing the query plans chosen by the query optimizer. If the default plan chosen by the optimi... not optimal, a *temporary* solution is to use one of these configuration parameters to force... to improve the quality of the plans chosen by the optimizer include a... ANALYZE ... equally... increasing the value of the defau... for specific columns using ALTER ... STA... CS...

`enable_async_append (boolean)`

Enables or disables the query planner's use of async-aware append plan types. The default is ...

`enable_bitmapscan (boolean)`

Enables or disables the query planner's use of bi... ...ype... ...t is on.

`enable_gathermerge (boolean)`

Enables or disables the query planner's use of g... ...erge plan ... The ... ...t is o...

`enable_hashagg (boolean)`

Enables or disables the query planner's use of hashed aggregation plan types. The default is on.

`enable_hashjoin (boolean)`

Enables or disables the query planner's use of hash-join plan types. The default is on.

`enable_incremental_sort (boolean)`

Enables or disables the query planner's use of incremental sort steps. The default is on.

`enable_indexscan (boolean)`

Enables or disables the query planner's use of index-scan plan types. The default is on.

`enable_indexonlyscan (boolean)`

Enables or disables the query planner's use of index-only-scan plan types (see Section 11.9). The default is on.

`enable_material (boolean)`

Enables or disables the query planner's use of materialization. It is impossible to suppress materialization entirely, but turning this variable off prevents the planner from inserting materialize nodes except in cases where it is required for correctness. The default is on.

`enable_memoize (boolean)`

Enables or disables the query planner's use of memoize plans for caching results from parameterized scans inside nested-loop joins. This plan type allows scans to the underlying plans to be skipped when the results for the current parameters are already in the cache. Less commonly looked up results may be evicted from the cache when more space is required for new entries. The default is on.

`constraint_exclusion (enum)`

Controls the query planner's use of table constraints to optimize queries. The allowed values of `constraint_exclusion` are `on` (examine constraints for all tables), `off` (never examine constraints), and `partition` (examine constraints only for inheritance child tables and `UNION ALL` subqueries). `partition` is the default setting. It is often used with traditional inheritance trees to improve performance.

When this parameter allows it for a particular table, the planner compares query conditions with the table's `CHECK` constraints, and omits scanning tables for which the conditions contradict the constraints. For example:

```
CREATE TABLE parent(key integer, ...);
CREATE TABLE child1000(check (key between 1000 and 1999)) INHERITS(parent);
CREATE TABLE child2000(check (key between 2000 and 2999)) INHERITS(parent);
...
SELECT * FROM parent WHERE key = 2400;
```

With constraint exclusion enabled, this `SELECT` will not scan `child1000` at all, improving performance.

Currently, constraint exclusion is enabled by default only for cases that are often used to implement table partitioning via inheritance trees. Turning it on for all tables imposes extra planning overhead that is quite noticeable on simple queries, and most often will yield no benefit for simple queries. If you have no tables that are partitioned using traditional inheritance, you might prefer to turn it off entirely. (Note that the equivalent feature for partitioned tables is controlled by a separate parameter, enable_partition_pruning.)

Refer to Section 5.11.5 for more information on using constraint exclusion to implement partitioning.

`cursor_tuple_fraction (floating point)`

Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved. The default is 0.1. Smaller values of th... ...g bias the planner towards using "fast start" plans for cursors, which will retrieve the first few rows q... ...time to fetch all rows. Larger values put more emphasis on the total estimated t... ...axim... ...retur... ...ors... ...ed exactly like regular queries, considering only the total ... ...ted time ar... ...how... ...th... ...e q...

`coll... ... (i... ...r)`

The planner will merge sub... ...up... ...if ... ...FR... ...) have no more than this many it... ...ller values reduce p...nning time but might yield ... ...default is eight. For more ... Sect...

(... ...) ... ...may trigger use of the GEQO planner, resulting in non-optimal plans. ... 20.7.

(... ...)

...r... ...he ... co... ...on ... used by PostgreSQL, if available (see Chapter 32). The default is on.

`join_collapse_limit (int...`

The planner will rewrite ex... JOIN constructs (except `FULL JOIN`s) into lists of `FROM` items whenever a list of no more than this many items would result. Smaller values reduce planning time but might yield inferior query plans.

By default, this variable is set the same as `from_collapse_limit`, which is appropriate for most uses. Setting it to 1 prevents any reordering of explicit `JOIN`s. Thus, the explicit join order specified in the query will be the actual order in which the relations are joined. Because the query planner does not always choose the optimal join order, advanced users can elect to temporarily set this variable to 1, and then specify the join order they desire explicitly. For more information see Section 14.3.

Setting this value to geqo_threshold or more may trigger use of the GEQO planner, resulting in non-optimal plans. See Section 20.7.3.

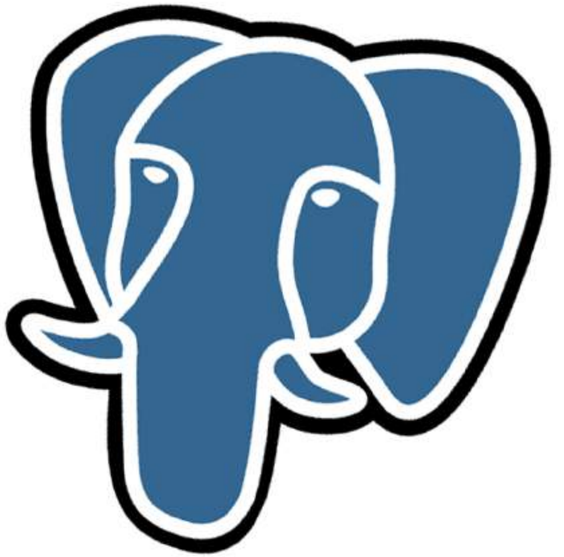
`plan_cache_mode (enum)`

Prepared statements (either explicitly prepared or implicitly generated, for example by PL/pgSQL) can be executed using custom or generic plans. Custom plans are made afresh for each execution using its specific set of parameter values, while generic plans do not rely on the parameter values and can be re-used across executions. Thus, use of a generic plan saves planning time, but if the ideal plan depends strongly on the parameter values then a generic plan may be inefficient. The choice between these options is normally made automatically, but it can be overridden with `plan_cache_mode`. The allowed values are `auto` (the default), `force_custom_plan` and `force_generic_plan`. This setting is considered when a cached plan is to be executed, not when it is prepared. For more information see PREPARE.

Немного настроек PostgreSQL

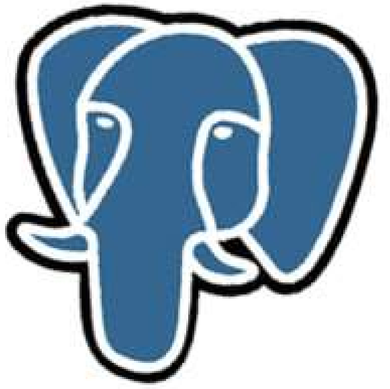
`plan_cache_mode` (enum)

Prepared statements (either explicitly prepared or implicitly generated, for example by PL/pgSQL) can be executed using custom or generic plans. Custom plans are made afresh for each execution using its specific set of parameter values, while generic plans do not rely on the parameter values and can be re-used across executions. Thus, use of a generic plan saves planning time, but if the ideal plan depends strongly on the parameter values then a generic plan may be inefficient. The choice between these options is normally made automatically, but it can be overridden with `plan_cache_mode`. The allowed values are `auto` (the default), `force_custom_plan` and `force_generic_plan`. This setting is considered when a cached plan is to be executed, not when it is prepared. For more information see PREPARE.



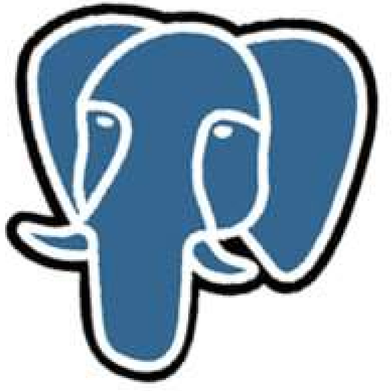
```
22  set plan_cache_mode=force_custom_plan;
23  prepare orisnull(
24      varchar,
25      integer,
26      varchar,
27      varchar
28  ) as select
29      *
30  from
31      books b
32  where
33      1=1
34    and (b.name = $1 or $1 is null)
35    and (b.rating = $2 or $2 is null)
36    and (b.author = $3 or $3 is null)
37    and (b.country = $4 or $4 is null)
38  ;
39  explain execute orisnull('Lord of the Rings',
40      null, null, null);
```

```
QUERY PLAN
1 Index Scan using idx_name on books b  (cost=0.28..8…
2   Index Cond: ((name)::text = 'Lord of the Rings'::…
```

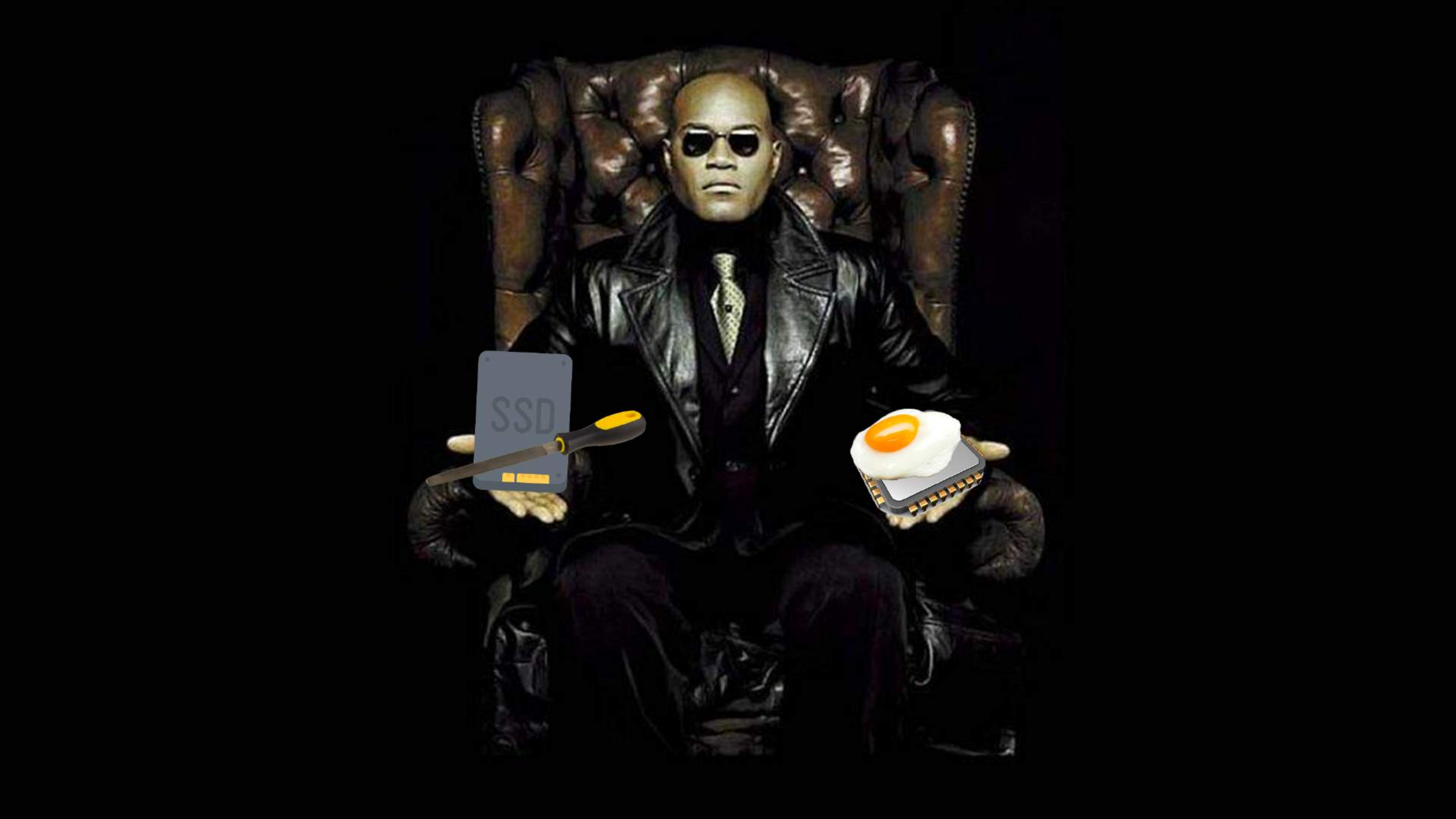


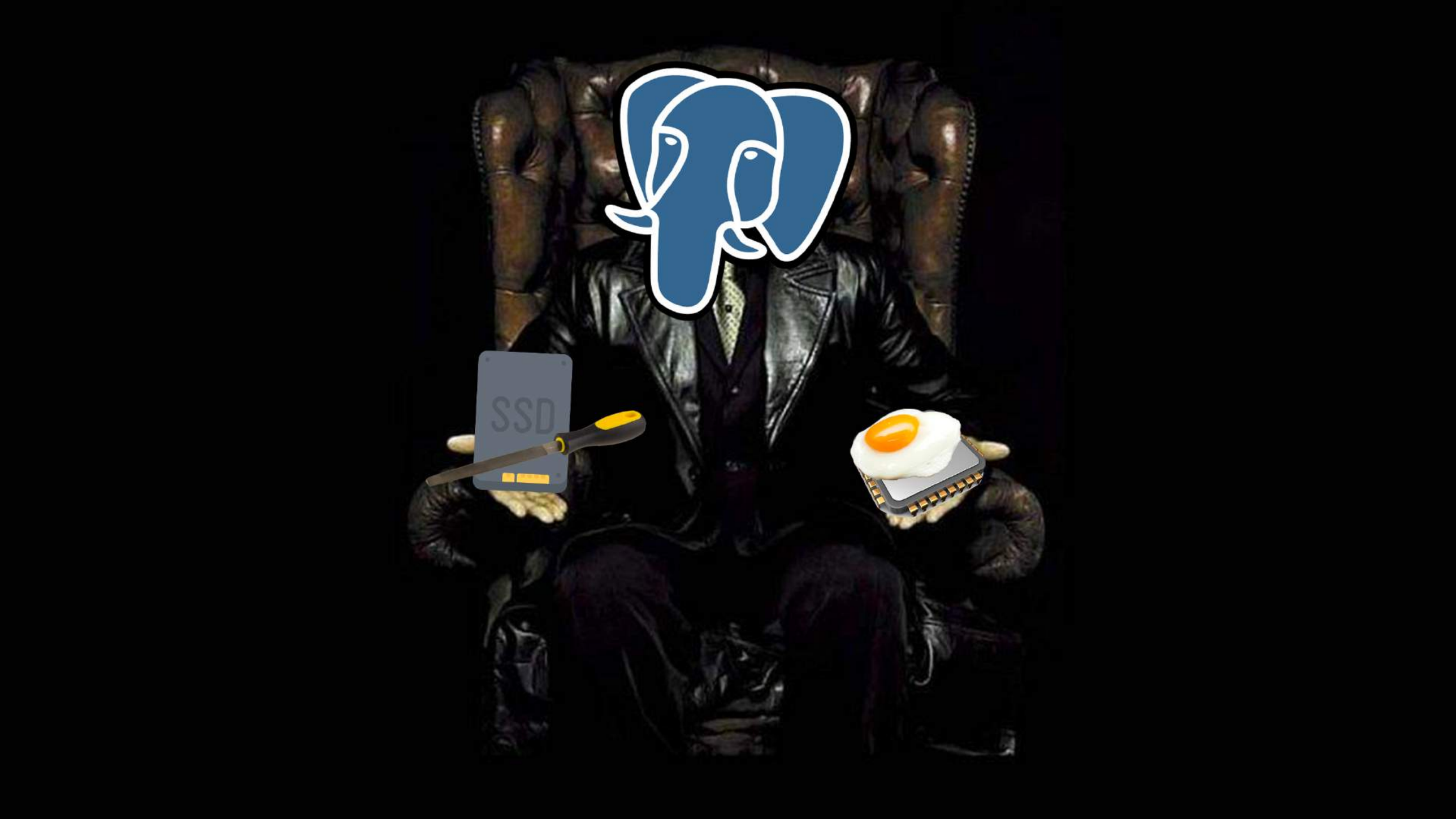
```
22  set plan_cache_mode=force_generic_plan;
23  prepare orisnull(
24      varchar,
25      integer,
26      varchar,
27      varchar
28  ) as select
29      *
30  from
31      books b
32  where
33      1=1
34    and (b.name = $1 or $1 is null)
35    and (b.rating = $2 or $2 is null)
36    and (b.author = $3 or $3 is null)
37    and (b.country = $4 or $4 is null)
38  ;
39  explain execute orisnull('Lord of the Rings',
40      null, null, null);
```

```
QUERY PLAN
1 Seq Scan on books b  (cost=0.00..81.40 rows=1 width=12…
2   Filter: ((((name)::text = ($1)::text) OR ($1 IS NULL…
```



2 rows retrieved starting from 1 in 22 ms (execution: 3 ms, fetching: 19 ms)


SSD


SSD

Прекрасный рефакторинг

Было:

```
" select b from Book b where                " +
"   1=1                                     " +
" and (b.author = :#{#filter.author}        " +
"   or :#{#filter.author} is null)          " +
" and (b.rating = :#{#filter.rating}        " +
"   or :#{#filter.rating} is null)          " +
" and (b.country = :#{#filter.country}      " +
"   or :#{#filter.country} is null)         " +
" and (b.name = :#{#filter.name}            " +
"   or :#{#filter.name} is null)            "
```

Стало:

```
String sql = " select b from Book b where 1=1   "
  + (filter.getAuthor() == null ? "" :
  " and b.author = :author  "
)
  + (filter.getCountry() == null ? "" :
  " and b.country = :country"
)
  + (filter.getRating() == null ? "" :
  " and b.rating = :rating  "
)
  + (filter.getName() == null ? "" :
  " and b.name = :name      "
);
```

Не умничай

Name

Empire V

Author

Pelevin

Rating

5

Country

Russia

Мы

Name

Empire V

Author

Pelevin

Rating

5

Country

Russia

Мы

Партнёр

Name

Empire V

Author

Pelevin

Rating

5

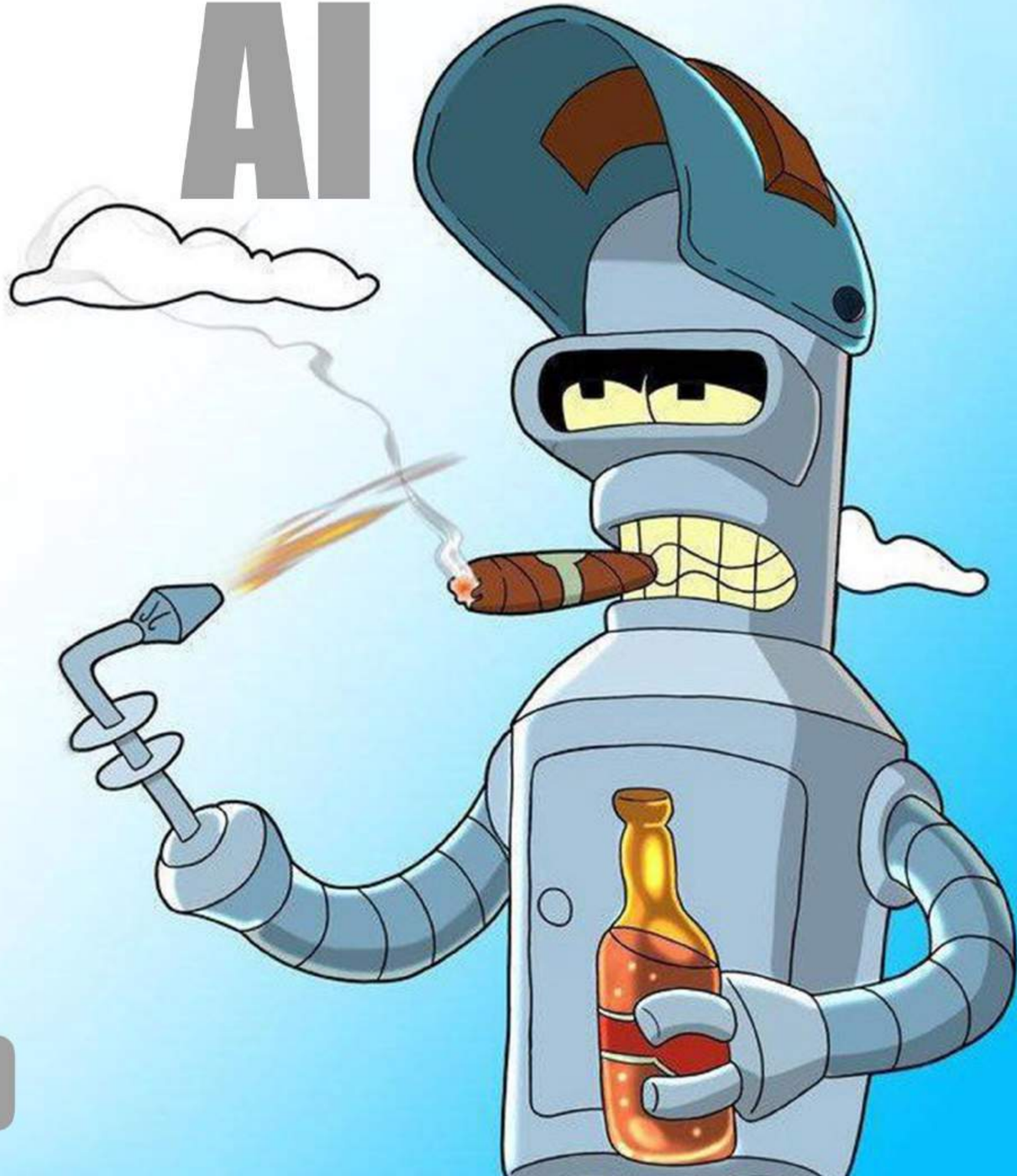
Country

Russia

Мы

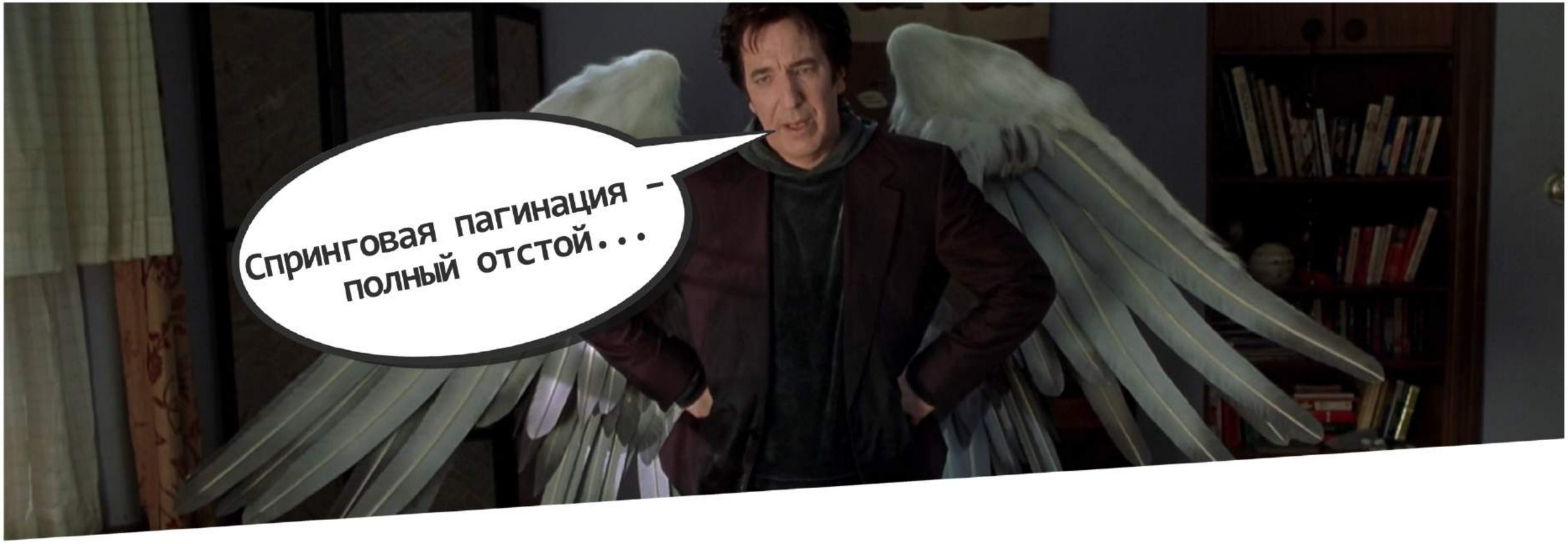
Партнёр

AI



Антипаттерн orisnull:

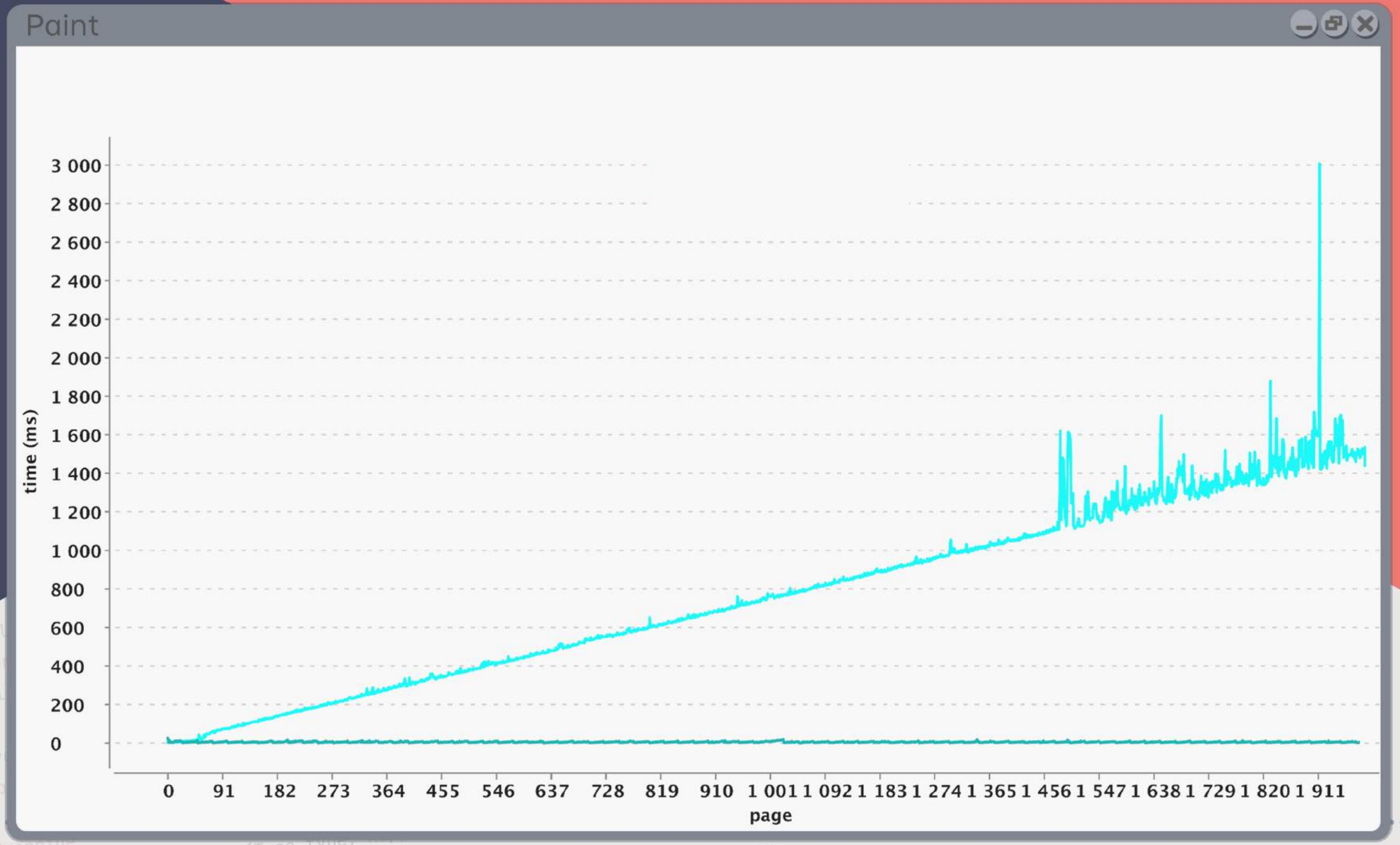
коварство иллюзорной простоты


Спринговая пагинация –
полный отстой...


Используй
keyset,
а не offset!


Используй
keyset,
а не offset!


Используй keyset, а не offset!
Во народ! Для чего нет ключевого слова в SQL, то и знать не надо?


Paint
time (ms)
3 000
2 800
2 600
2 400
2 200
2 000
1 800
1 600
1 400
1 200
1 000
800
600
400
200
0
0
91
182
273
364
455
546
637
728
819
910
1 001
1 092
1 183
1 274
1 365
1 456
1 547
1 638
1 729
1 820
1 911
page

Offset и keyset

почём пагинация для продакшена?

names.add(fieldName);
values.add(fieldValue);

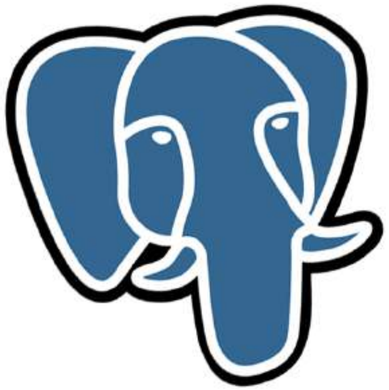
Проставляй границы транзакций

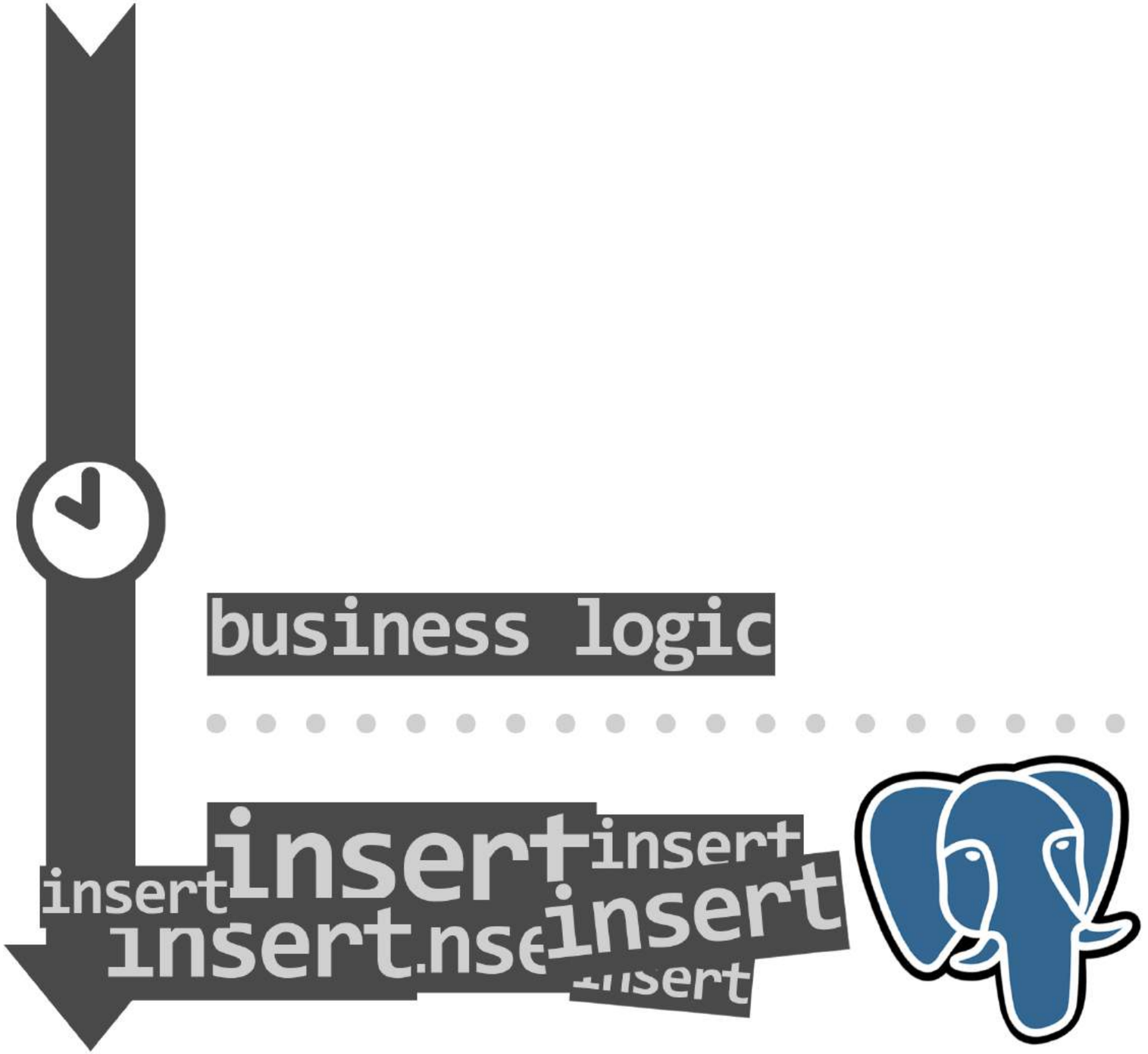
business logic

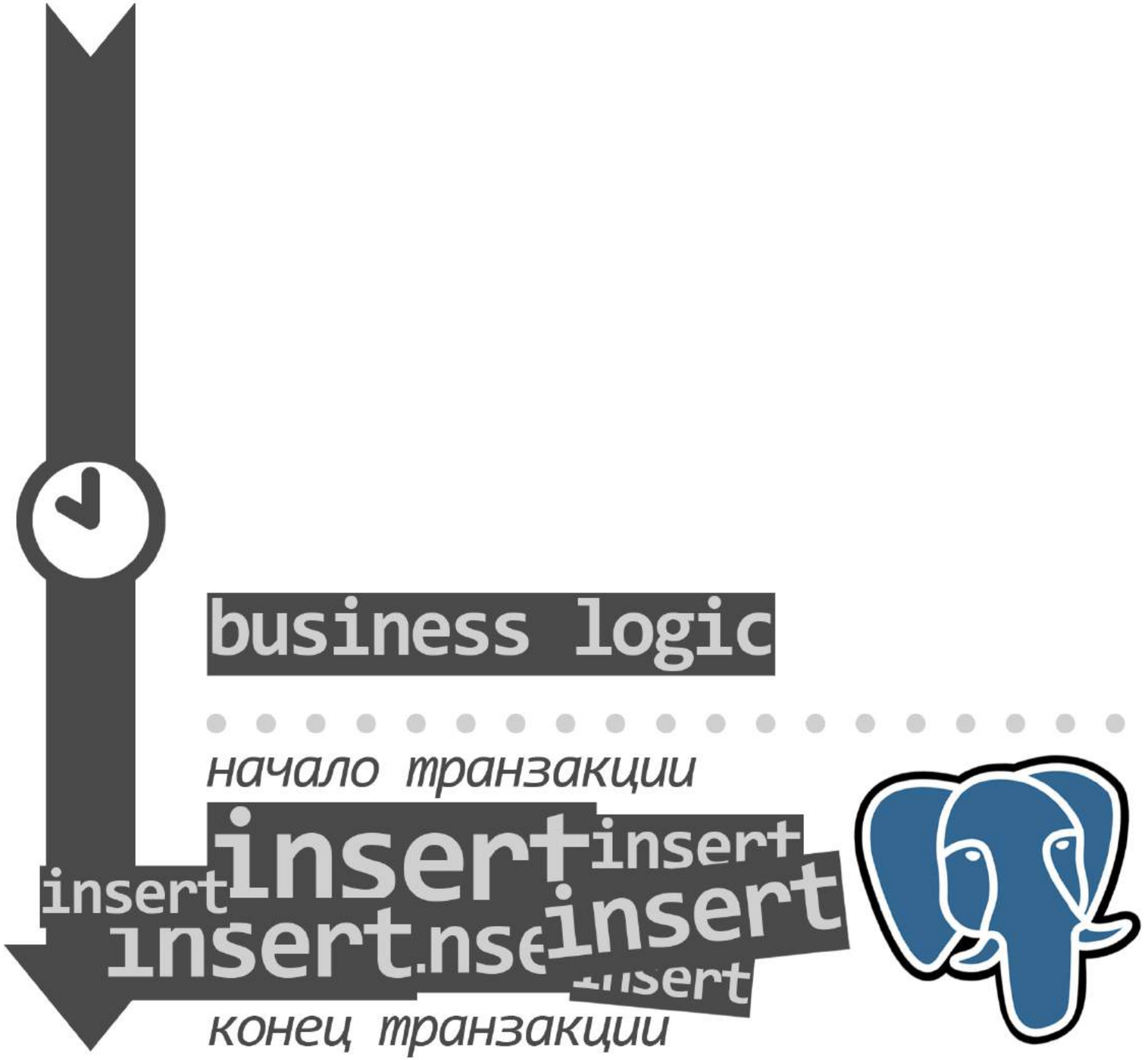
insert

```
business logic
```

```
insert
```




business logic
insert
insert
insert
insert
insert
insert


business logic
начало транзакции
insert
insert
insert
insert
insert
insert
конец транзакции

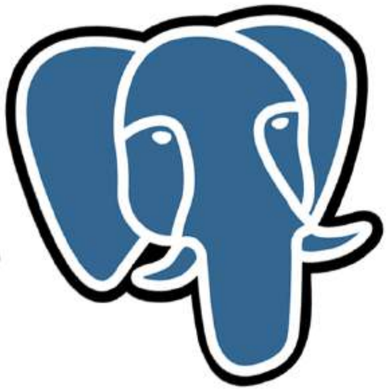
select

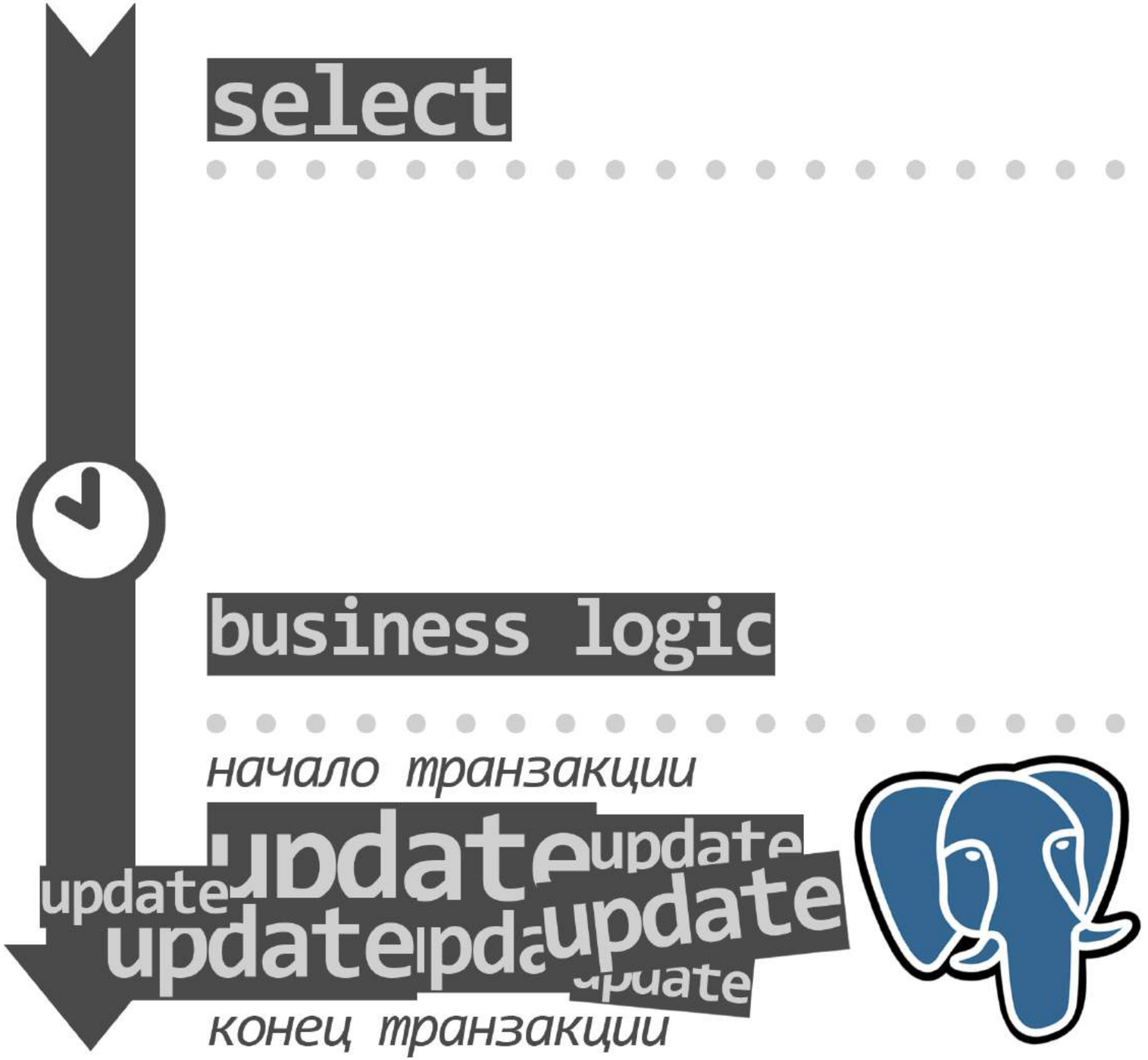
business logic

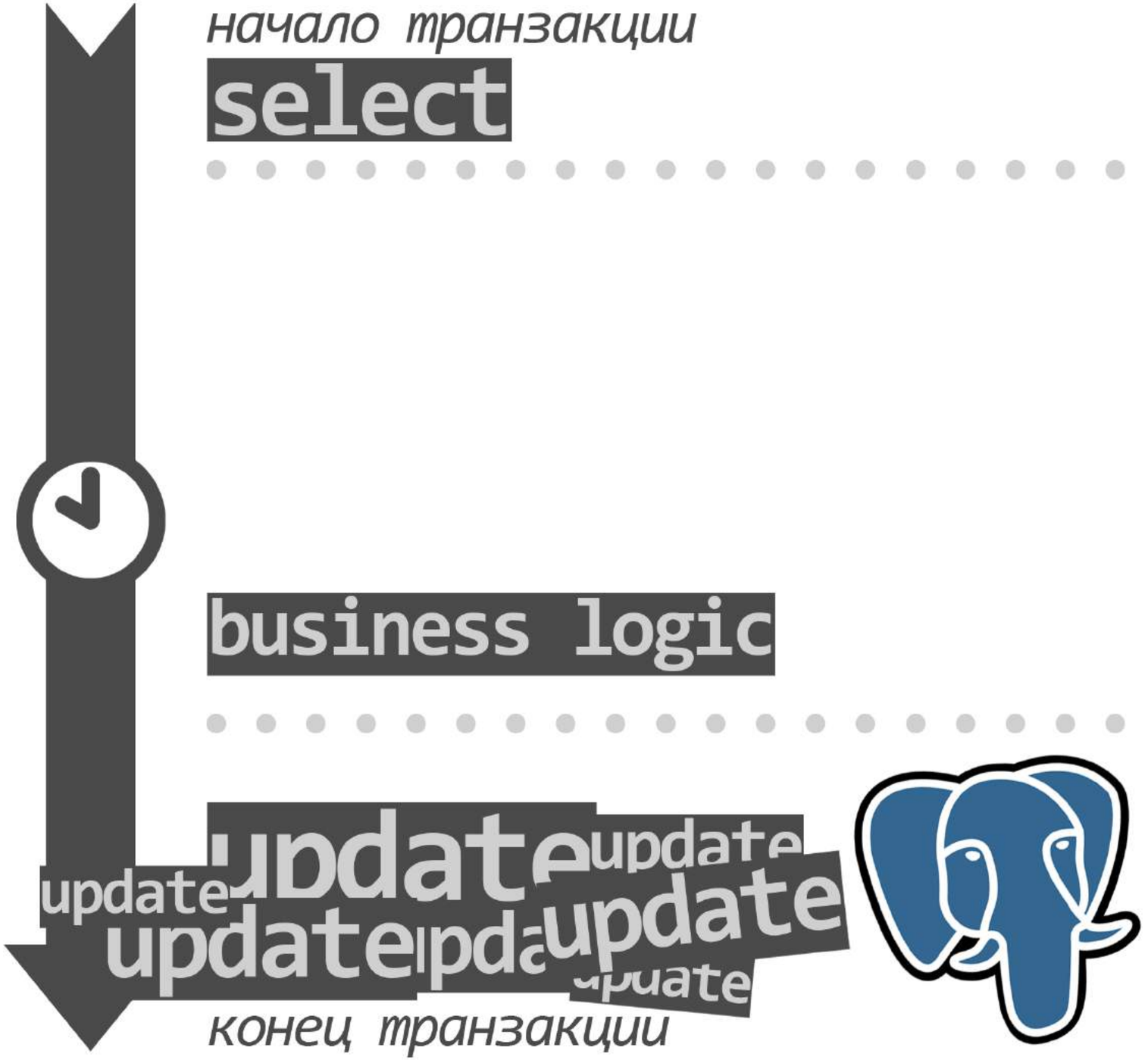
начало транзакции

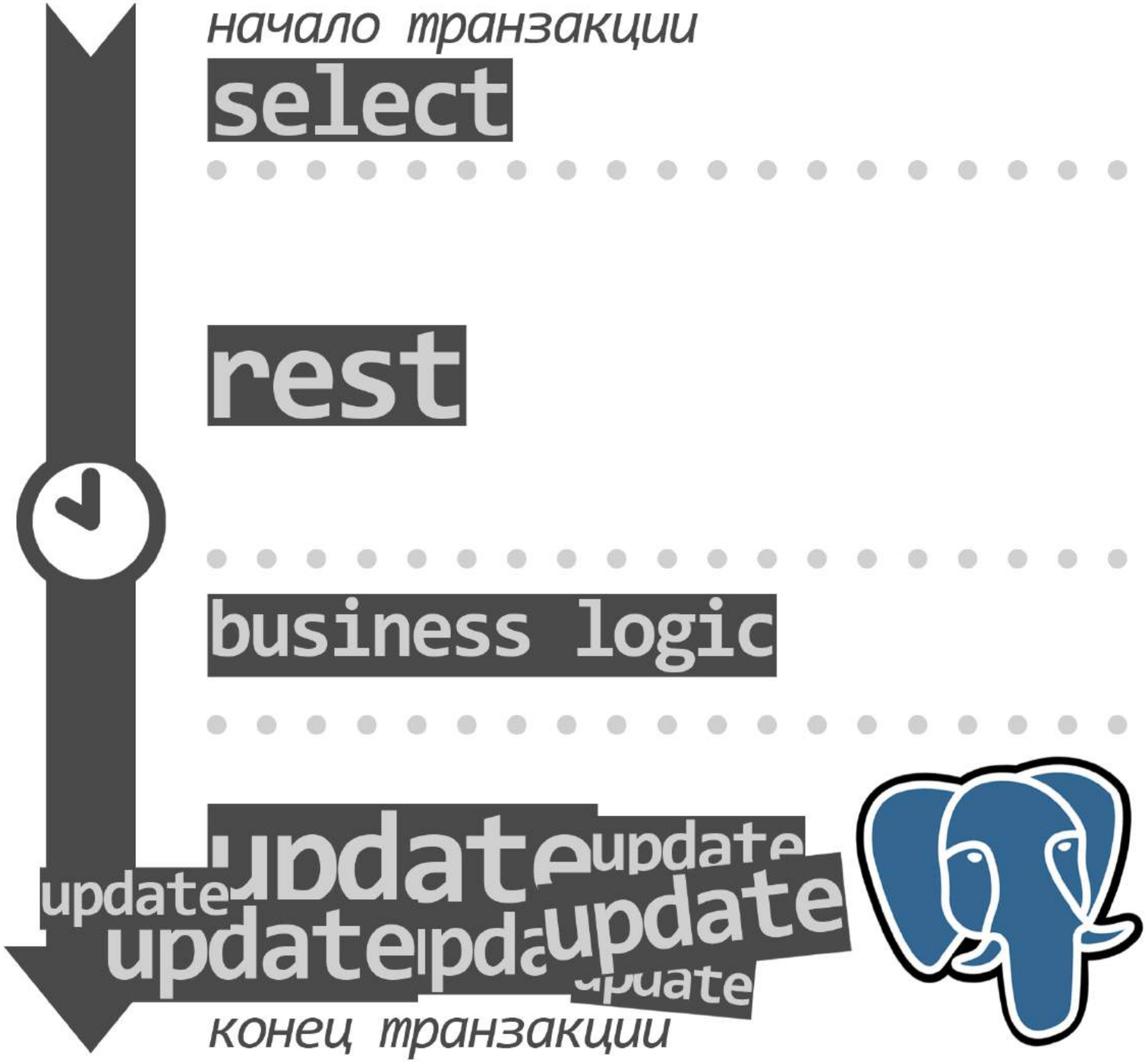
insert insert insert insert insert insert

конец транзакции




select
business logic
начало транзакции
update
update
update
update
update
update
конец транзакции


начало транзакции
select
business logic
update
update
update
update
update
конец транзакции


начало транзакции
select
rest
business logic
update
update
update
update
update
конец транзакции

начало транзакции

select

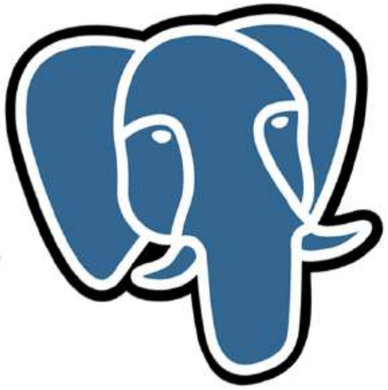
очень долго

rest

business logic

update update update update update update

конец транзакции



начало транзакции

select

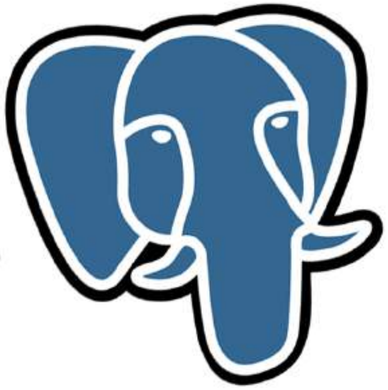
rest

очень долго

business logic

update update update update update update

конец транзакции



начало транзакции

select

rest

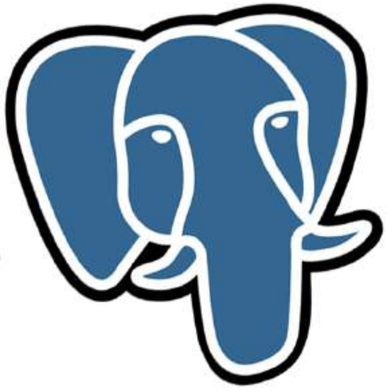
НУ И ЧТО?

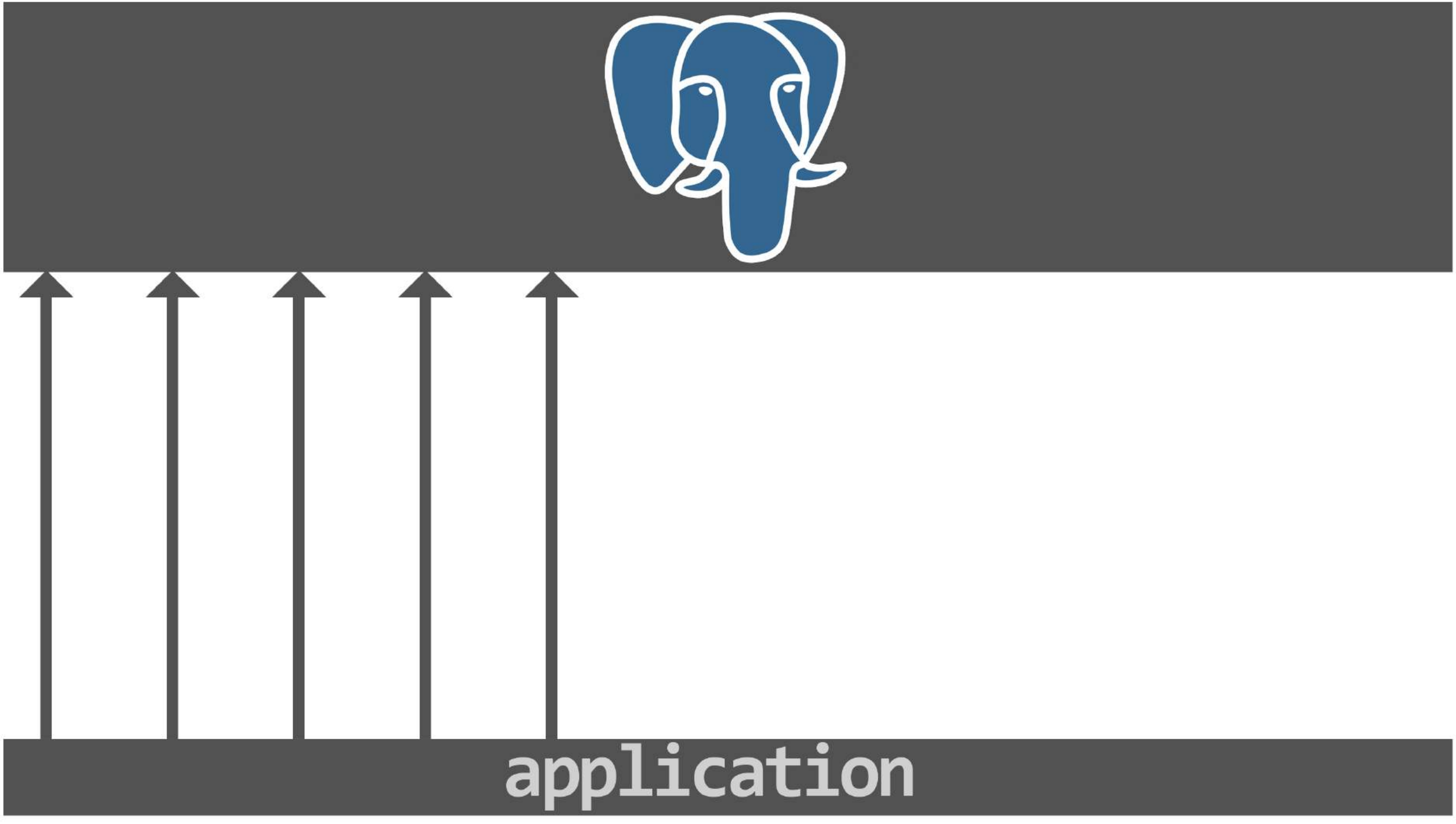
очень долго

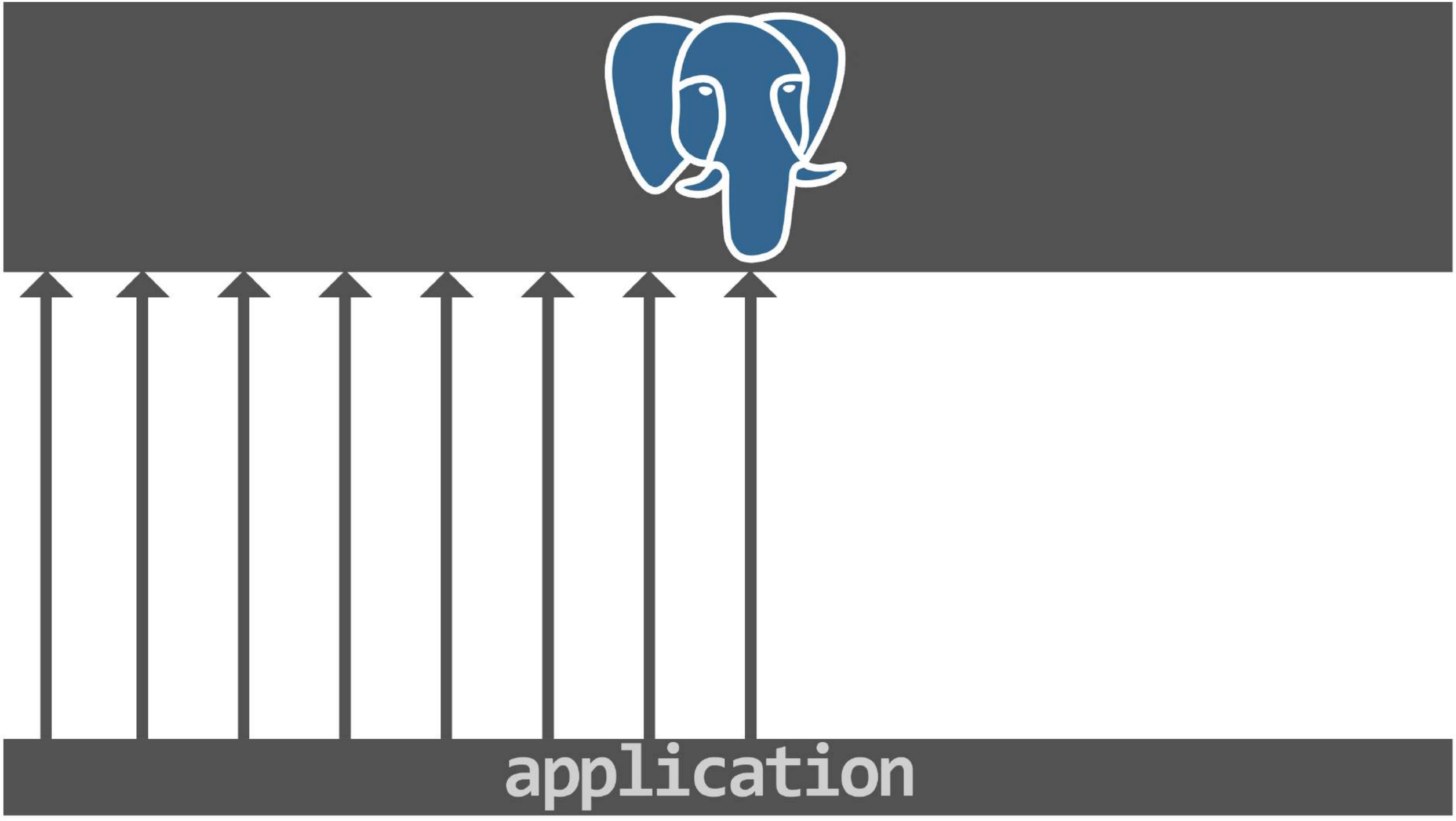
business logic

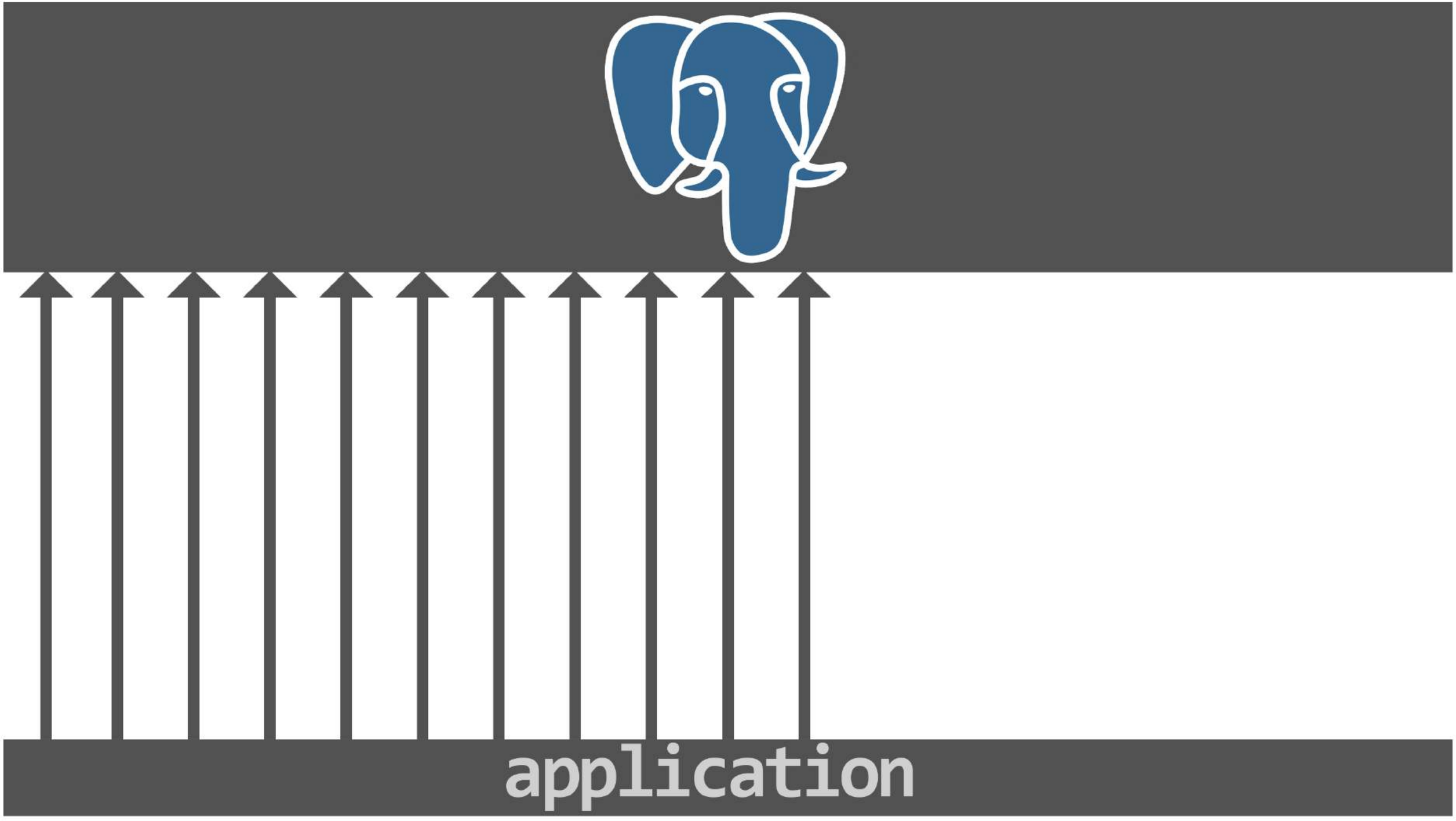
update update update update update

конец транзакции

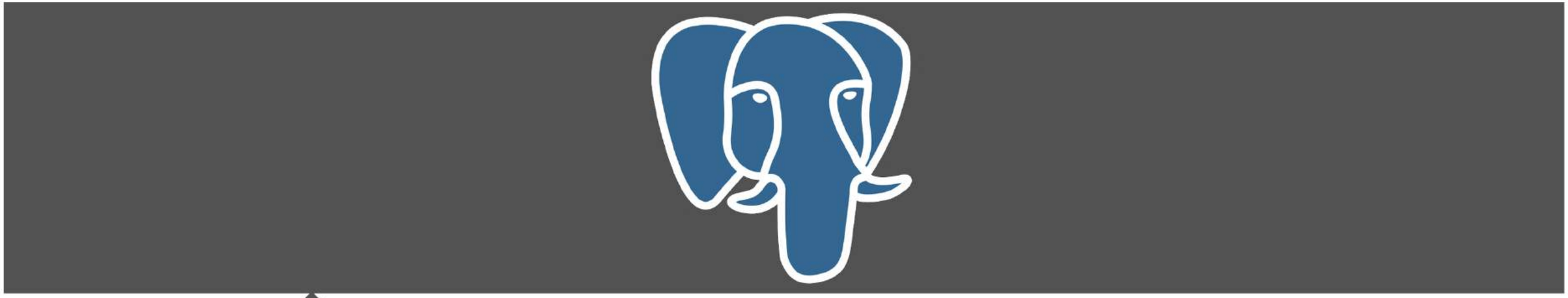



application


application


application

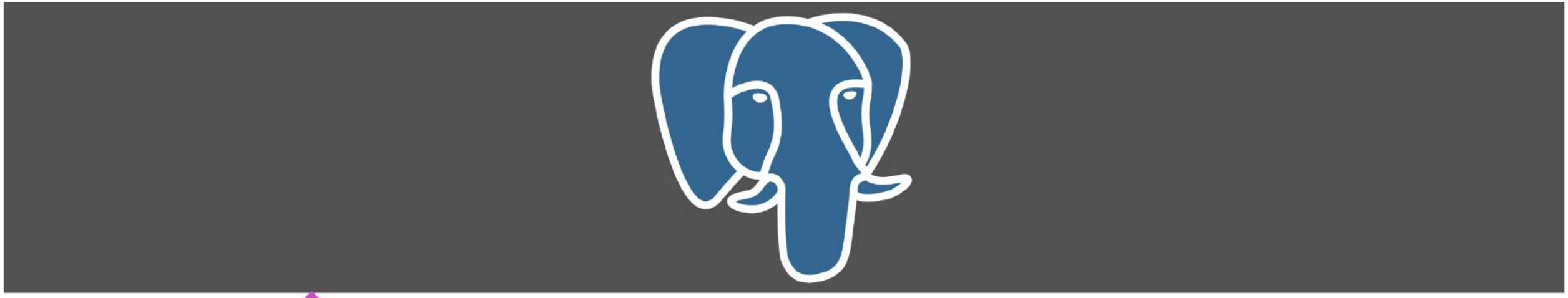
application



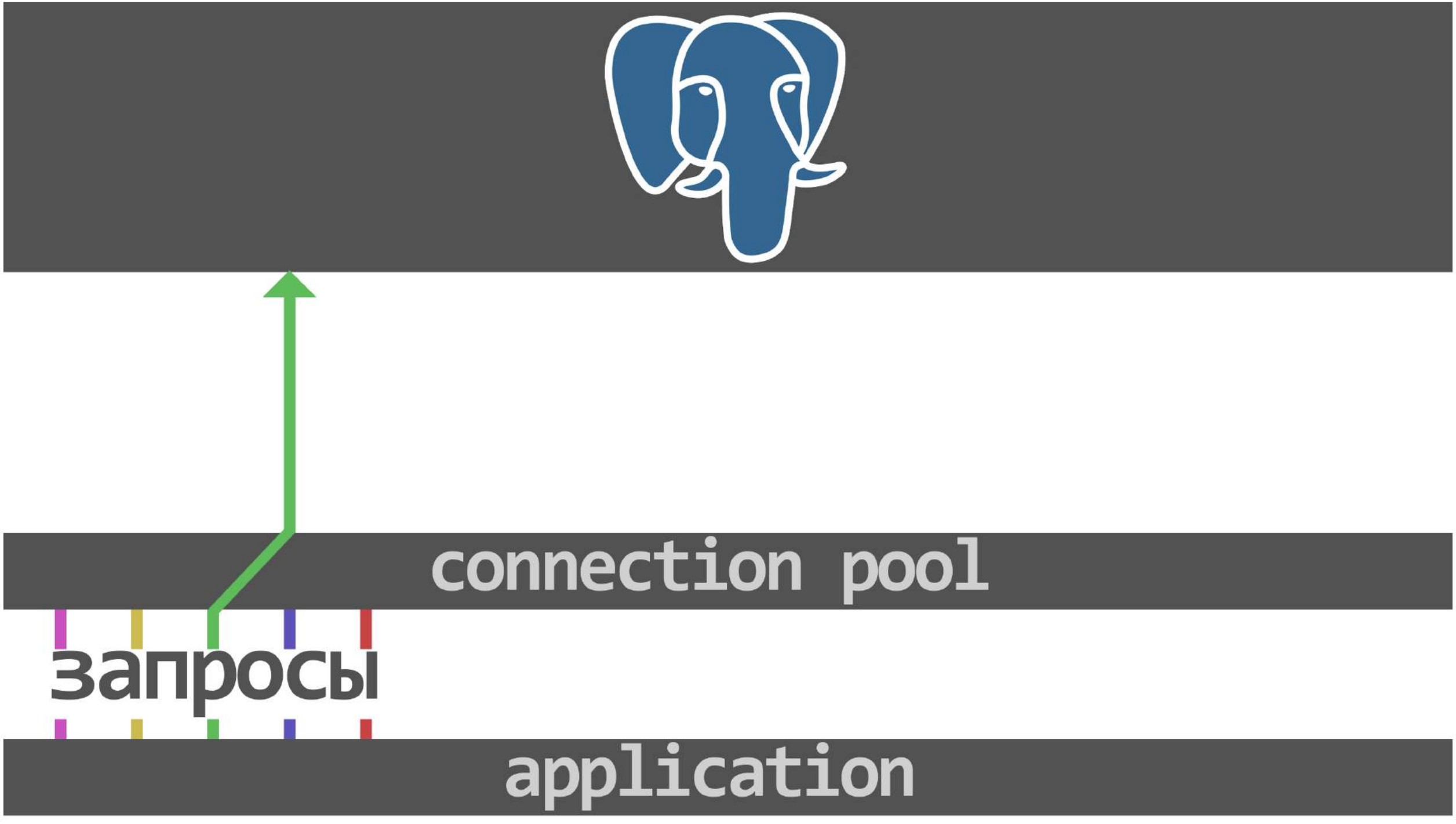
connection pool

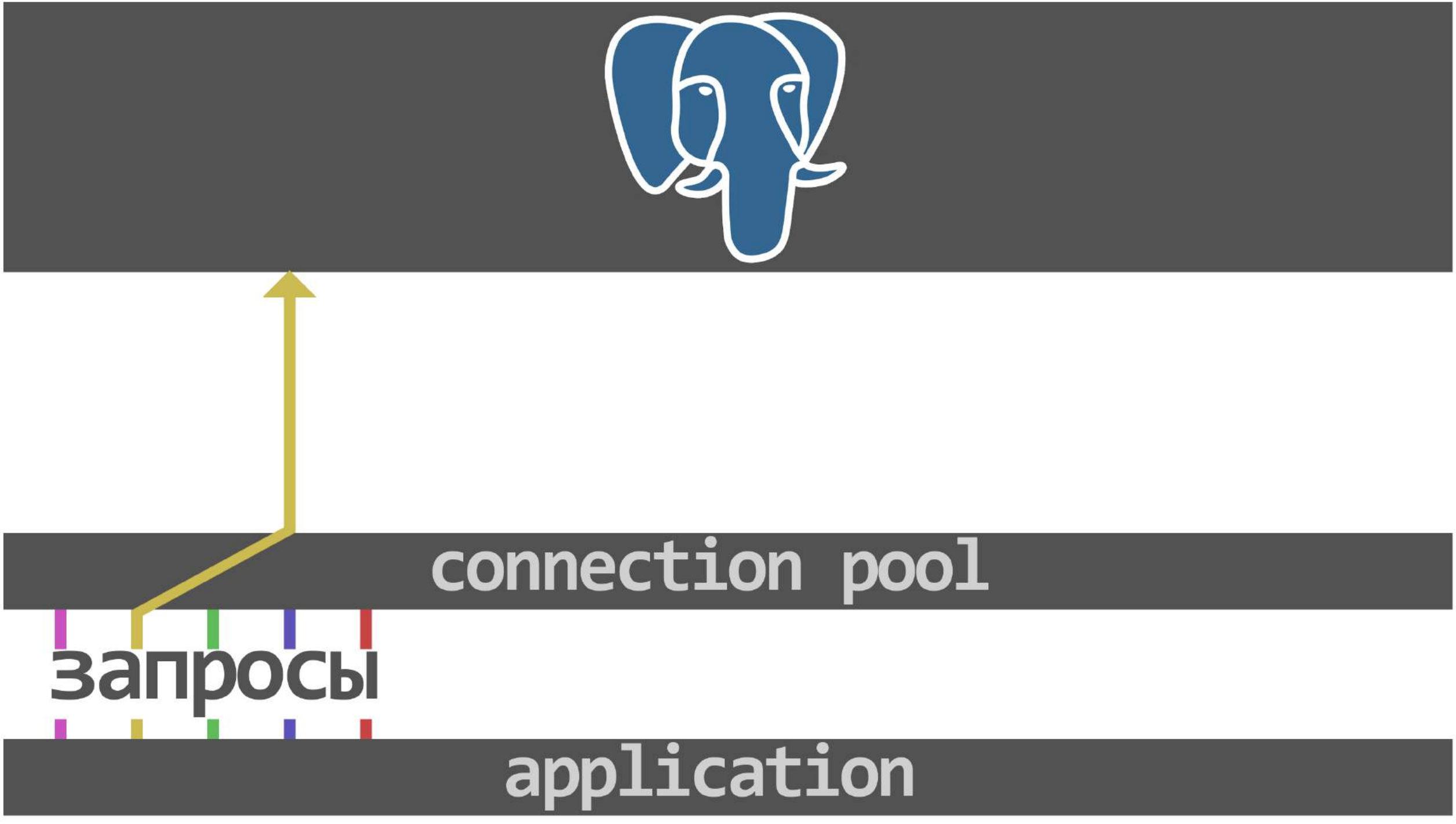
запросы

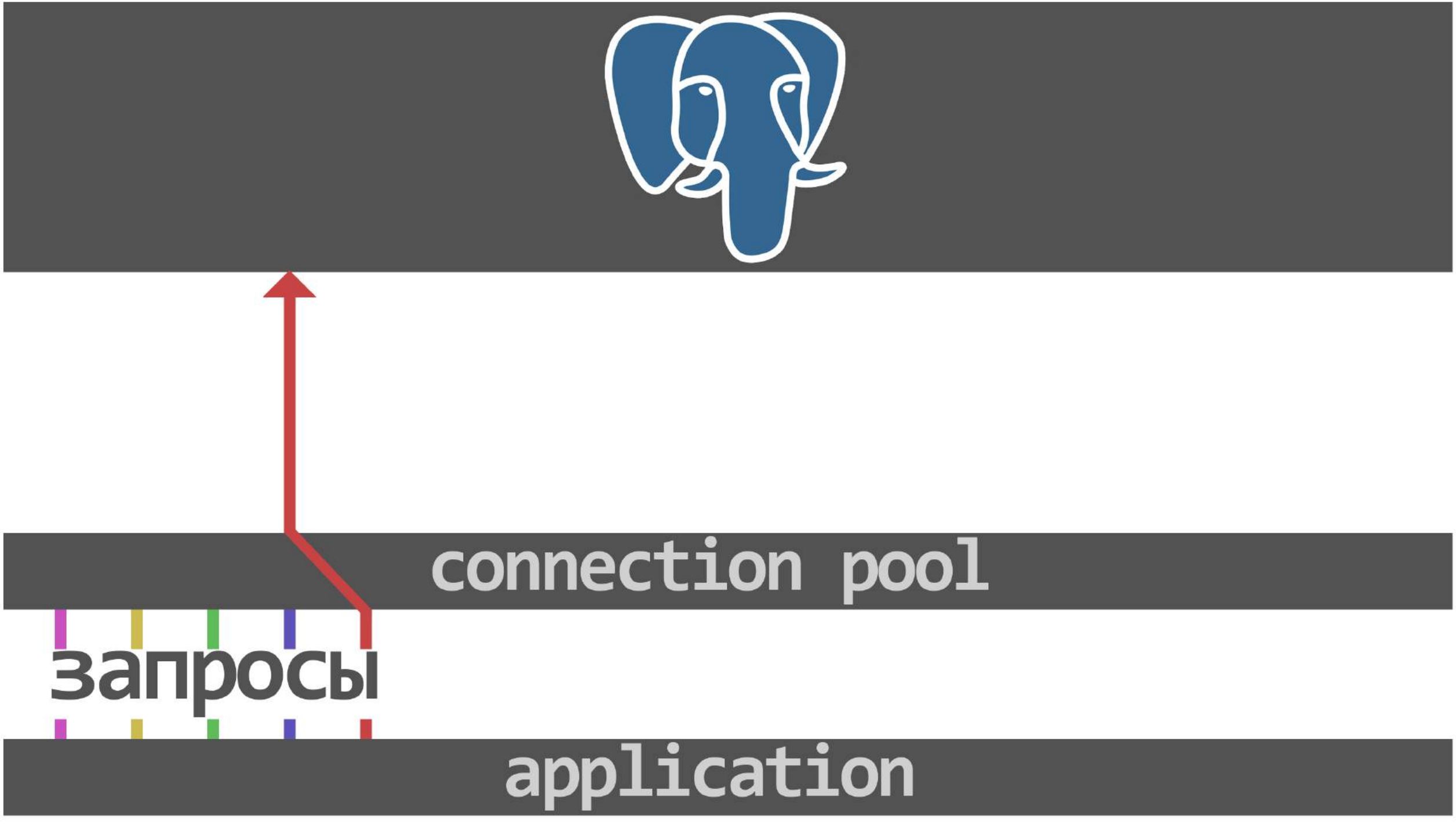
application

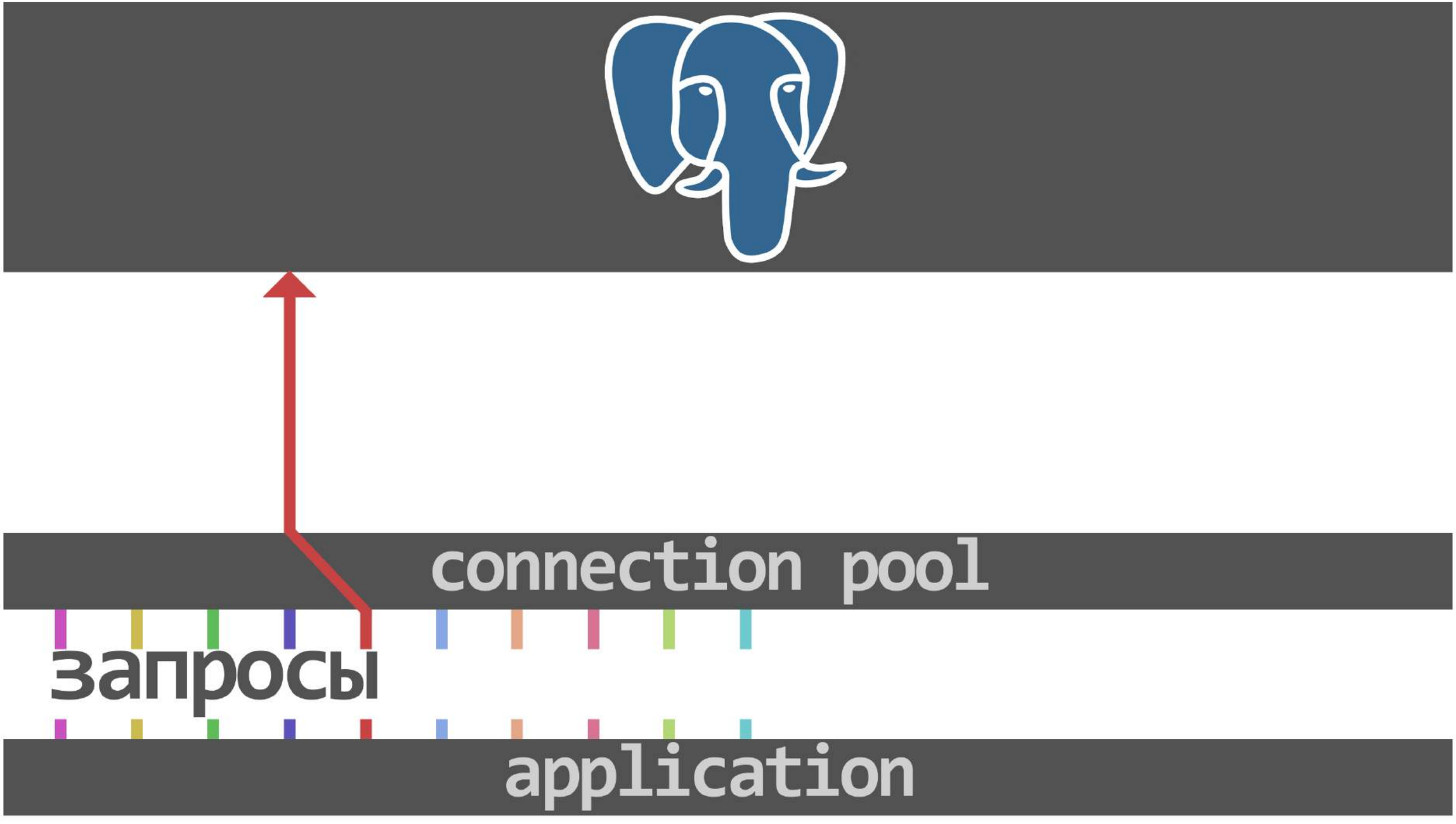


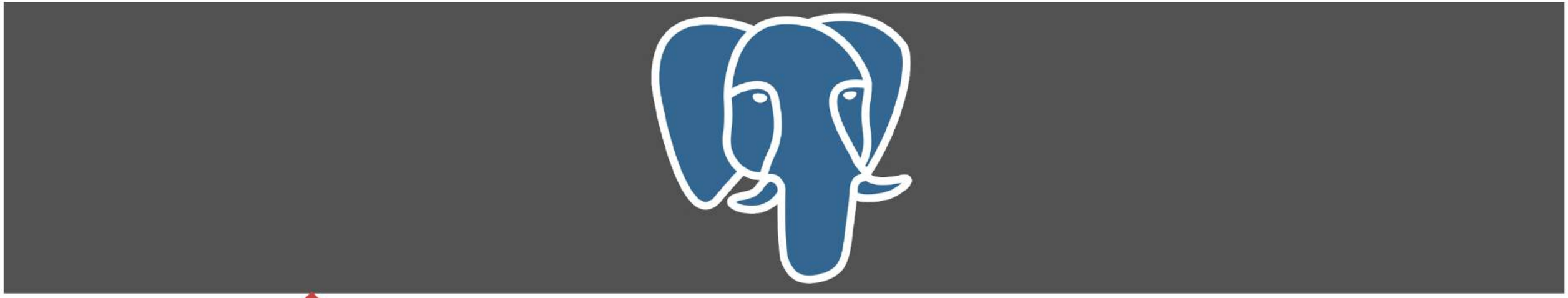

connection pool
запросы
application

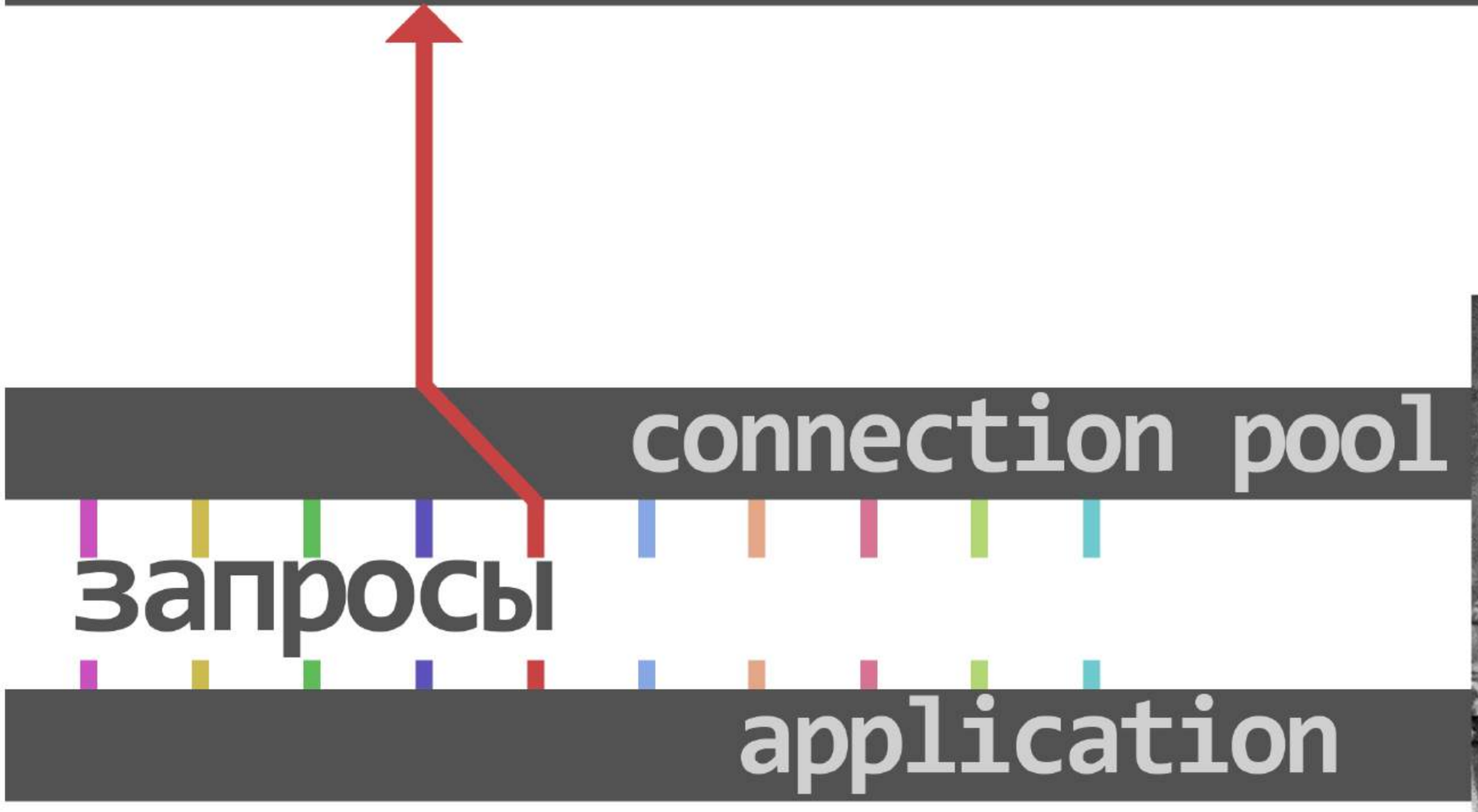

connection pool
запросы
application


connection pool
запросы
application

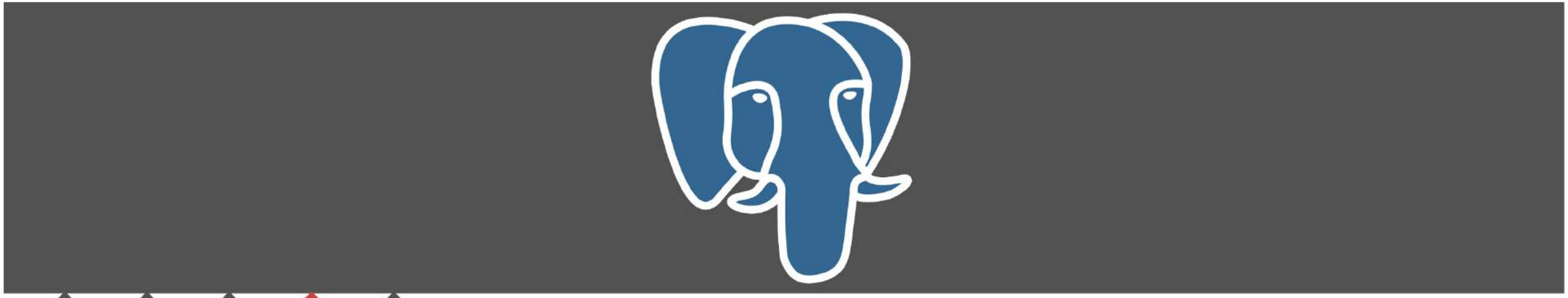

connection pool
запросы
application

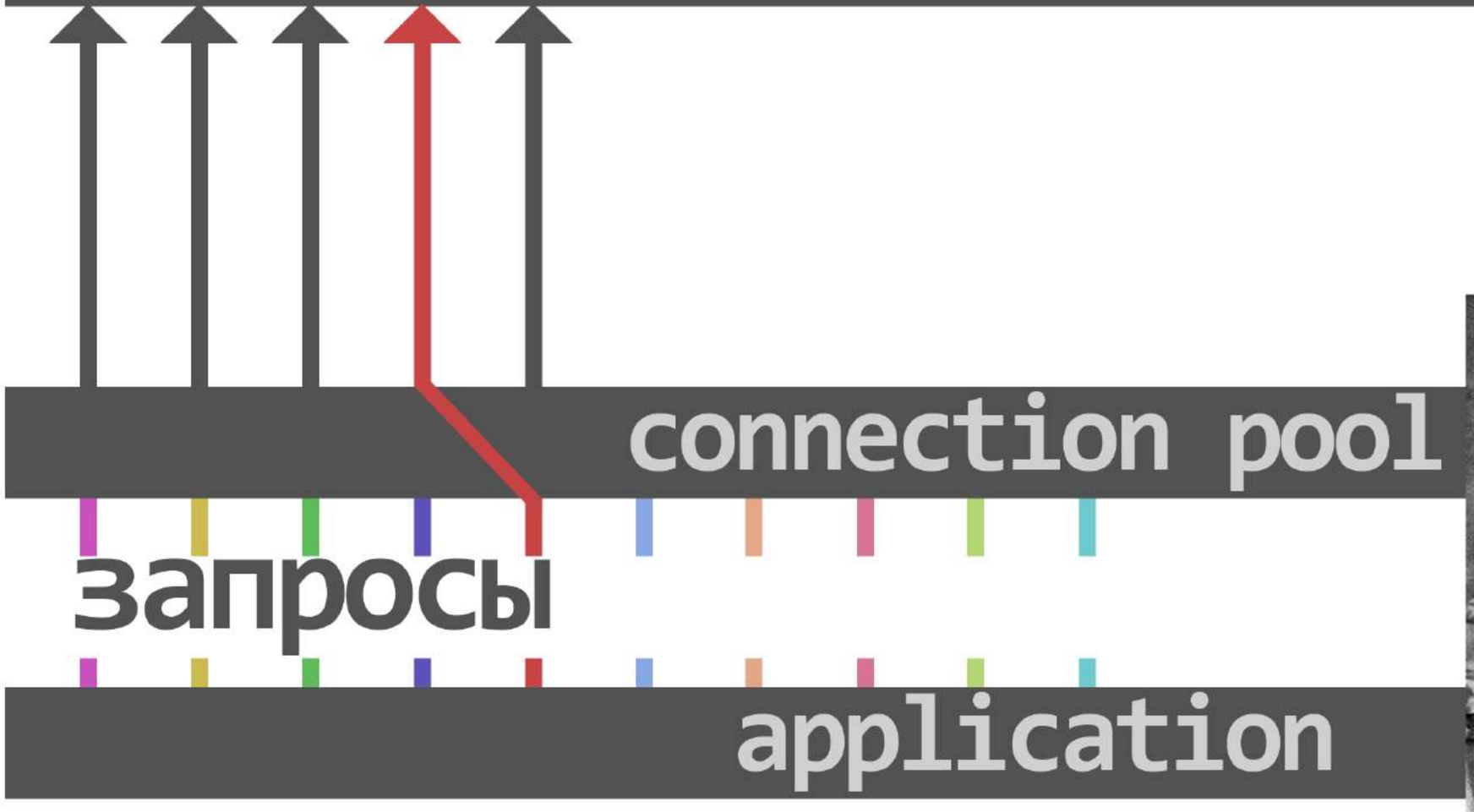

connection pool
запросы
application



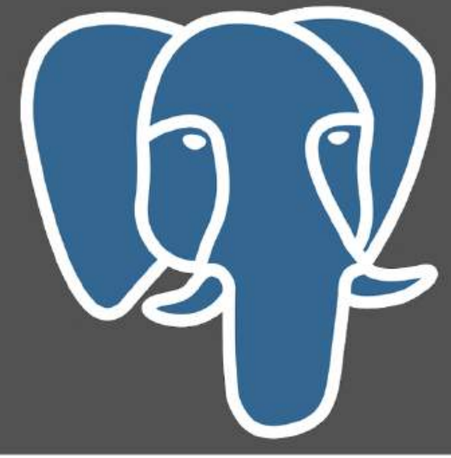

connection pool
запросы
application


РОГОВ и КОПЫТ
Закрыто
на обед




connection pool
запросы
application


РОГОВ и КОПЫТ
Закрыто
на обед



connection pool

запросы

application

начало транзакции

select

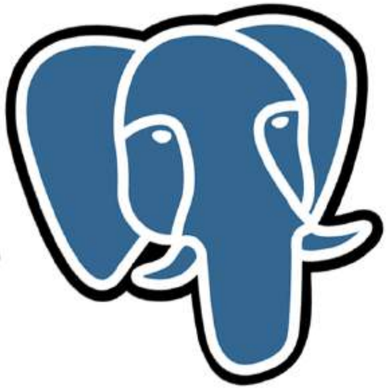
rest

очень долго

business logic

update update update update update

конец транзакции



начало транзакции

select

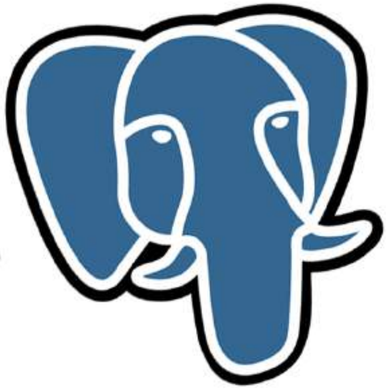
rest

очень долго

business logic

update update update update update

конец транзакции



очень долго
да и пофиг

rest

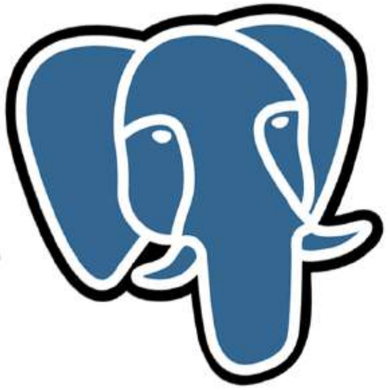
начало транзакции

select

business logic

update update update update update update

конец транзакции



начало транзакции

select

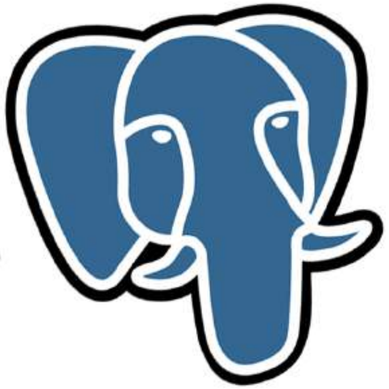
rest

очень долго

business logic

update update update update update

конец транзакции



начало транзакции

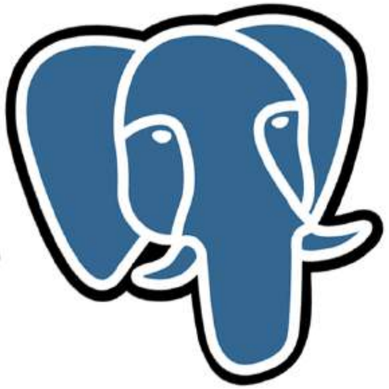
select

rest

business logic

update update update update update update

конец транзакции



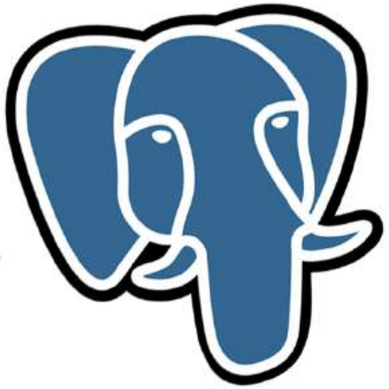
начало транзакции

select for update

rest

business logic

update update update update update update



конец транзакции

select

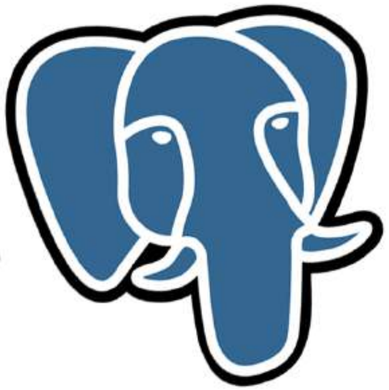
rest

business logic

начало транзакции

update update update update update update

конец транзакции

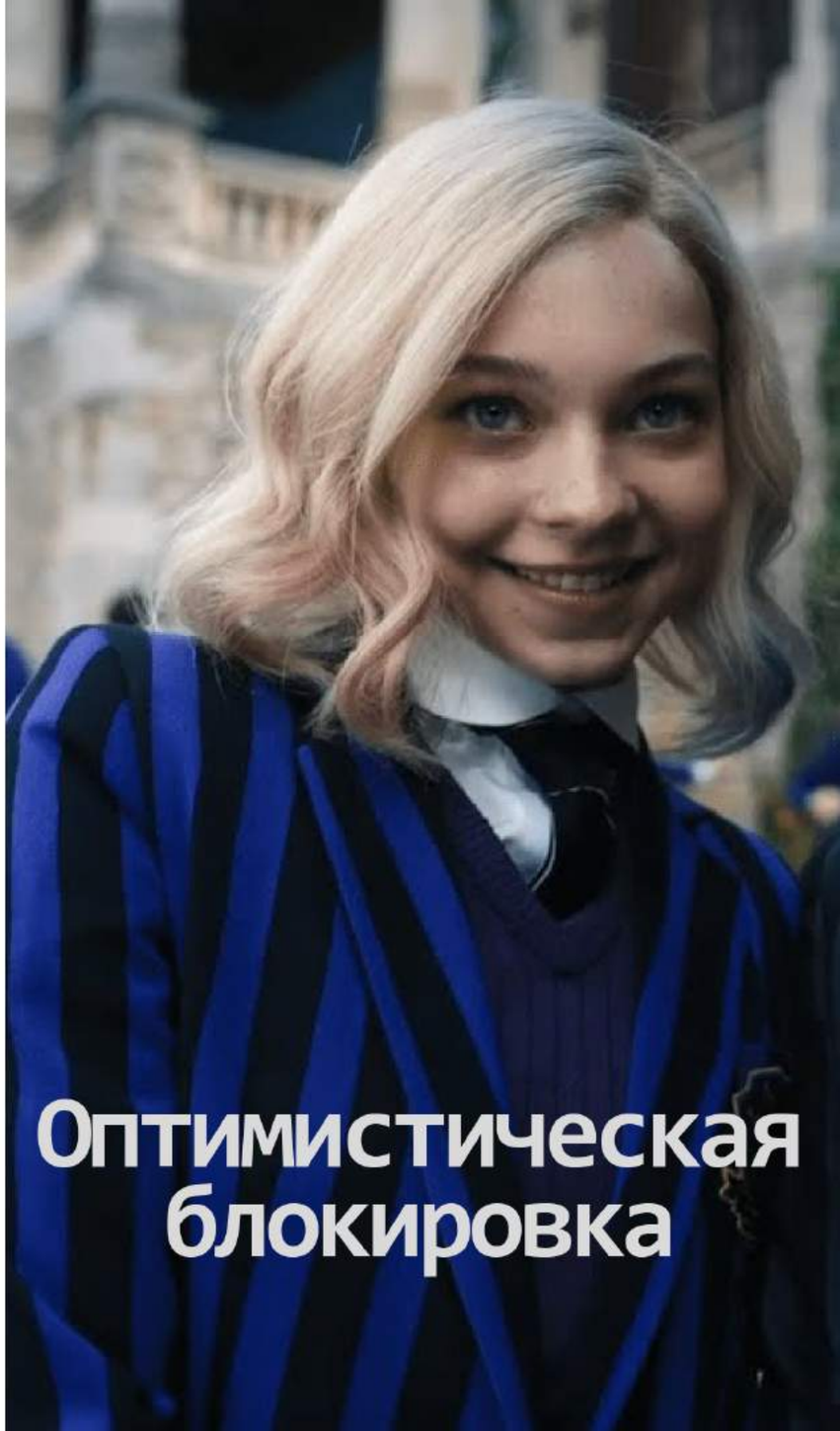


books

id	author id	title	rating	version
11	1	Властелин колец	6	1
23	1	Сильмариллион	5	1
37	1	Хоббит	4	1
41	2	Гарри Поттер	3	1
53	2	Фантастические звери	3	1
57	3	Солярис	5	1
73	3	Фиаско	5	1
77	5	Игра престолов	2	1
89	7	Чистая архитектура	5	1
97	7	Чистый кот	5	1
101	7	Чистый кодер	5	1

```
update books
set
  version=2,
  rating=7
where
  id=11 and
  version=1
```

```
update books
set
  version=2,
  title='Идеальный программист'
where
  id=101 and
  version=1
```

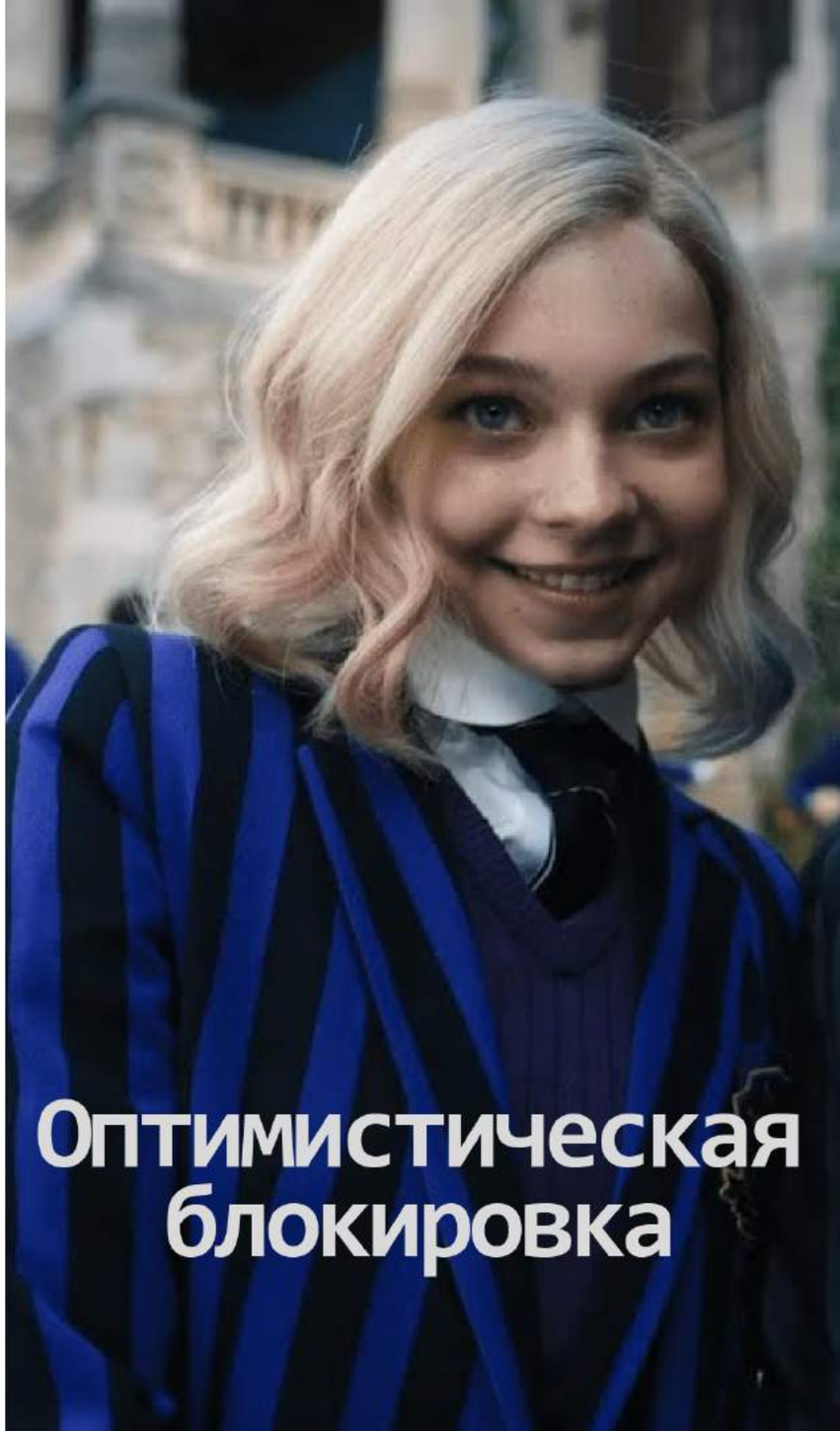


books

id	author id	title	rating	version
11	1	Властелин колец	6	1
23	1	Сильмариллион	5	1
37	1	Хоббит	4	1
41	2	Гарри Поттер	3	1
53	2	Фантастические звери	3	1
57	3	Солярис	5	1
73	3	Фиаско	5	1
77	5	Игра престолов	2	1
89	7	Чистая архитектура	5	1
97	7	Чистый кот	5	1
101	7	Чистый кодер	5	1

```
update books
set
  version=2,
  rating=7
where
  id=11 and
  version=1
```

```
update books
set
  version=2,
  title='Идеальный программист'
where
  id=101 and
  version=1
```

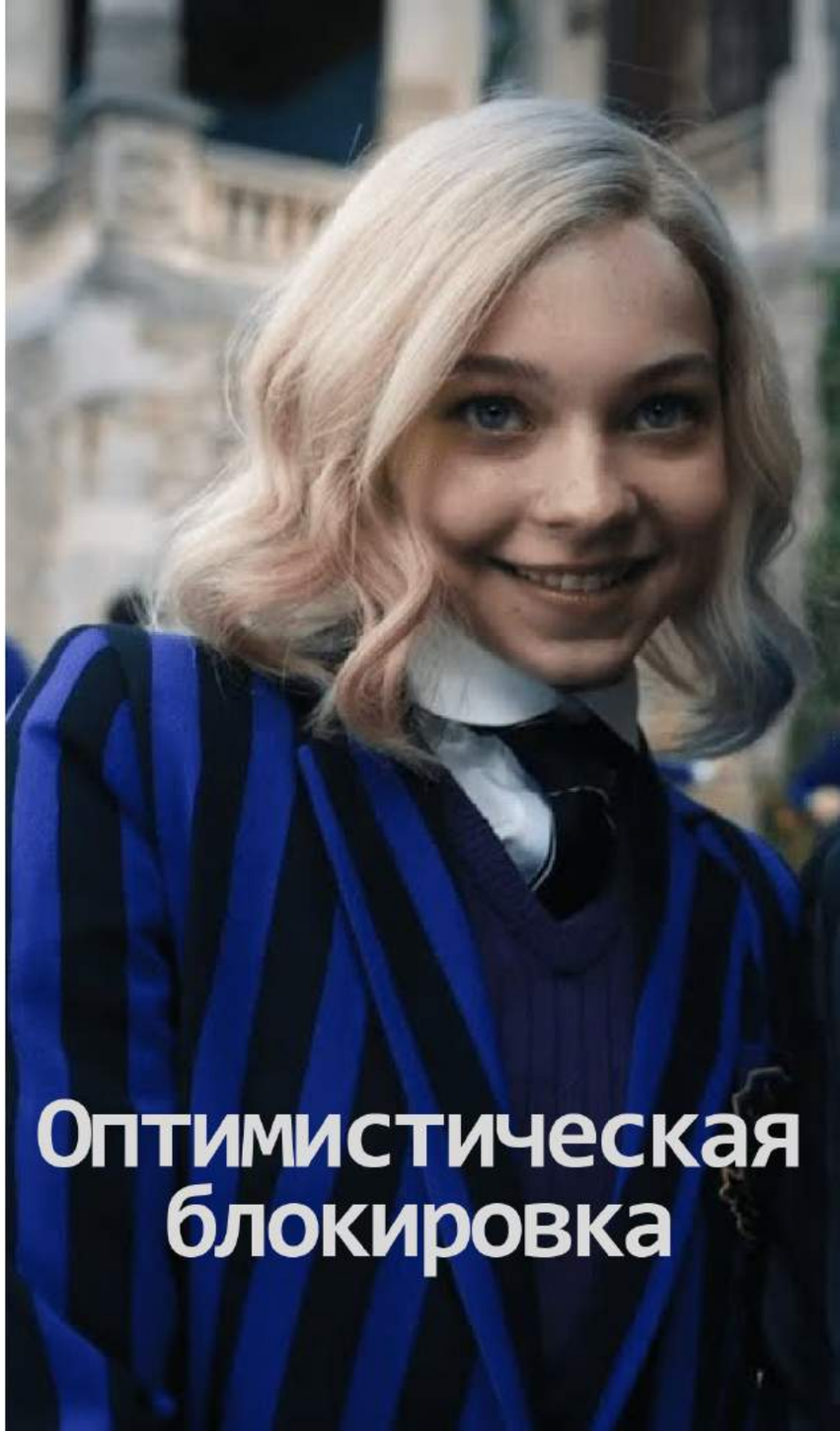


books

id	author id	title	rating	version
11	1	Властелин колец	6	1
23	1	Сильмариллион	5	1
37	1	Хоббит	4	1
41	2	Гарри Поттер	3	1
53	2	Фантастические звери	3	1
57	3	Солярис	5	1
73	3	Фиаско	5	1
77	5	Игра престолов	2	1
89	7	Чистая архитектура	5	1
97	7	Чистый кот	5	1
101	7	Чистый кодер	5	1

```
update books
set
  version=2,
  rating=7
where
  id=11 and
  version=1
```

```
update books
set
  version=2,
  title='Идеальный программист'
where
  id=101 and
  version=1
```

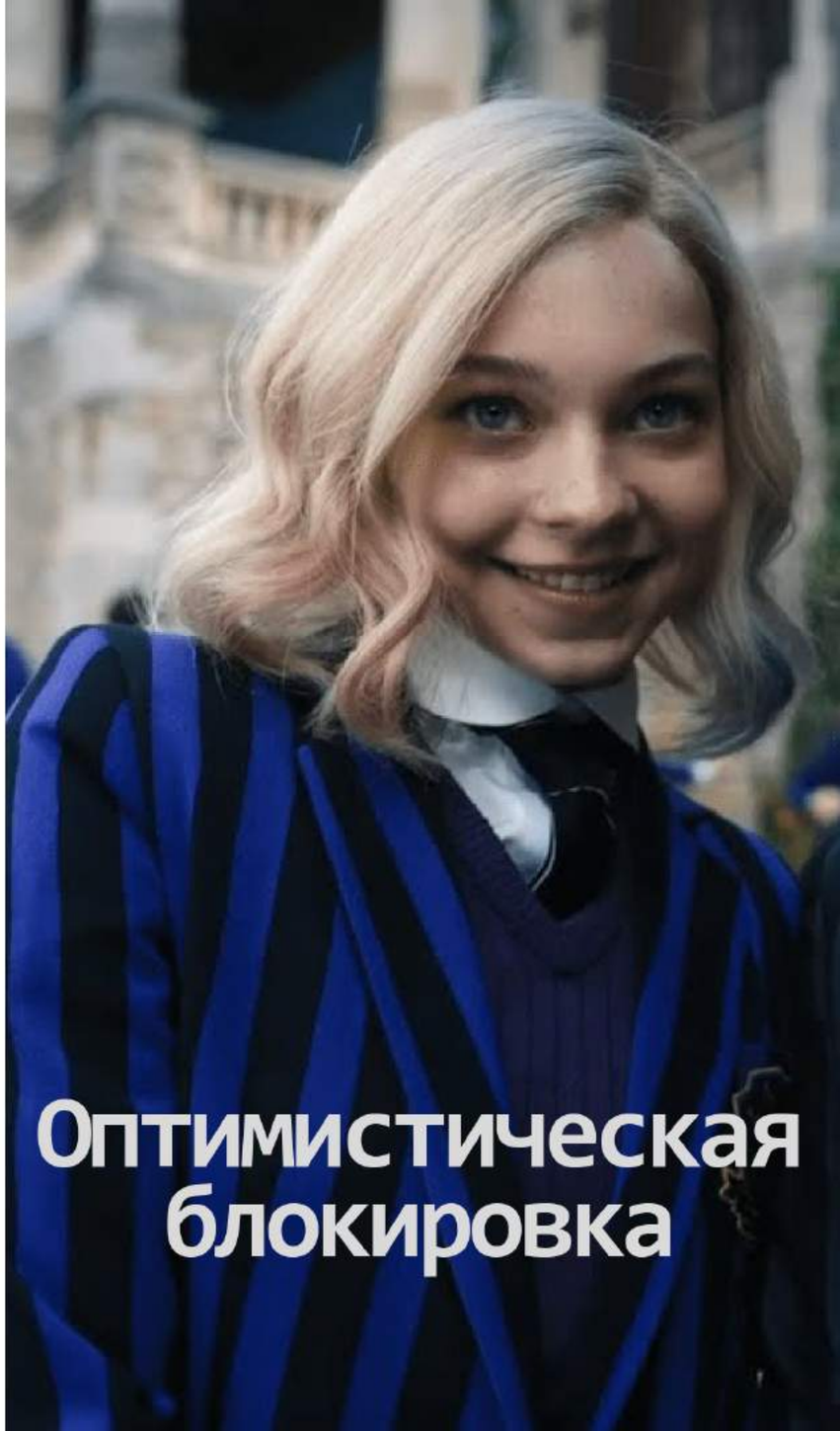


books

id	author id	title	rating	version
11	1	Властелин колец	6	1
23	1	Сильмариллион	5	1
37	1	Хоббит	4	1
41	2	Гарри Поттер	3	1
53	2	Фантастические звери	3	1
57	3	Солярис	5	1
73	3	Фиаско	5	1
77	5	Игра престолов	2	1
89	7	Чистая архитектура	5	1
97	7	Чистый кот	5	1
101	7	Чистый кодер	5	1

```
update books
set
  version=2,
  rating=7
where
  id=11 and
  version=1
```

```
update books
set
  version=2,
  title='Идеальный программист'
where
  id=101 and
  version=1
```



select

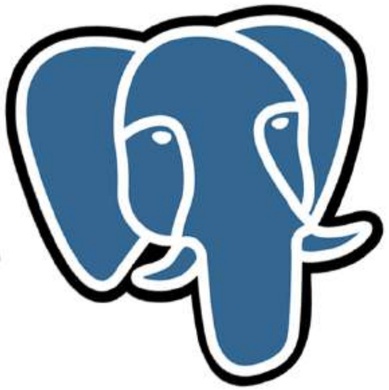
rest

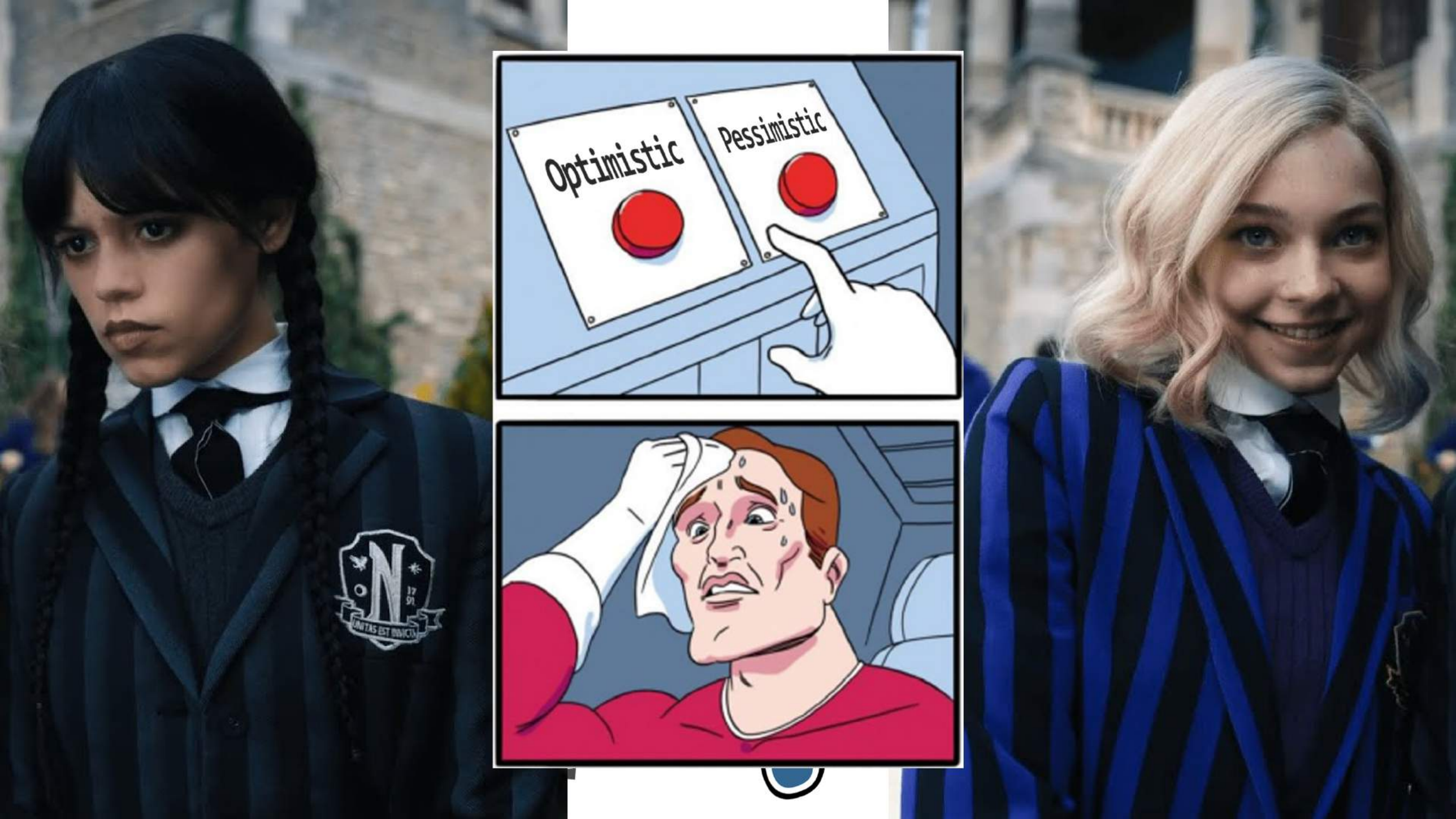
business logic

начало транзакции

update update update update update update

конец транзакции




Optimistic
Pessimistic


Да не...
Дефолты лучше не трогать...

select

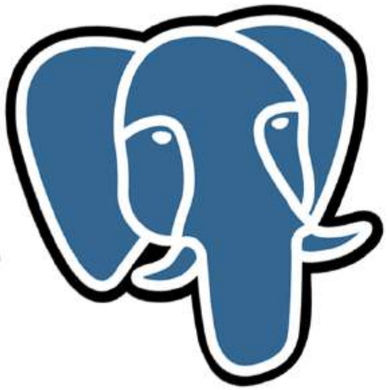
rest

business logic

начало транзакции

update update update update update

конец транзакции



OSIV

открытие коннекшона

select

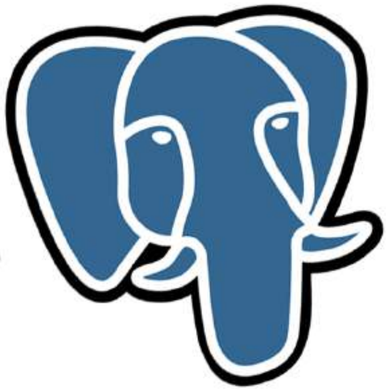
rest

business logic

начало транзакции

update update update update update

конец транзакции



OSIV

открытие коннекшона

select

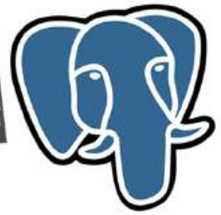
rest

business logic

начало транзакции

update update update update

конец транзакции



закрытие коннекшона

открытие коннекшона

select

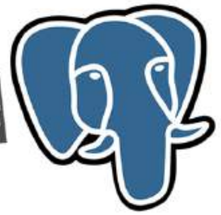
rest

business logic

начало транзакции

update update update update

конец транзакции



рендеринг веб страницы | сериализация json

закрытие коннекшона

открытие коннекшона

select

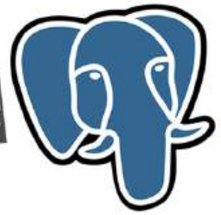
rest

business logic

начало транзакции

update update update update

конец транзакции



рендеринг веб страницы | сериализация json

```
spring.jpa.open-in-view=false
```

закрытие коннекшона

~~открытие коннекшена~~

select

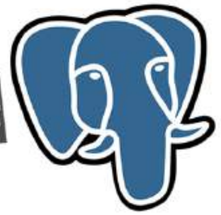
rest

business logic

начало транзакции

update update update update

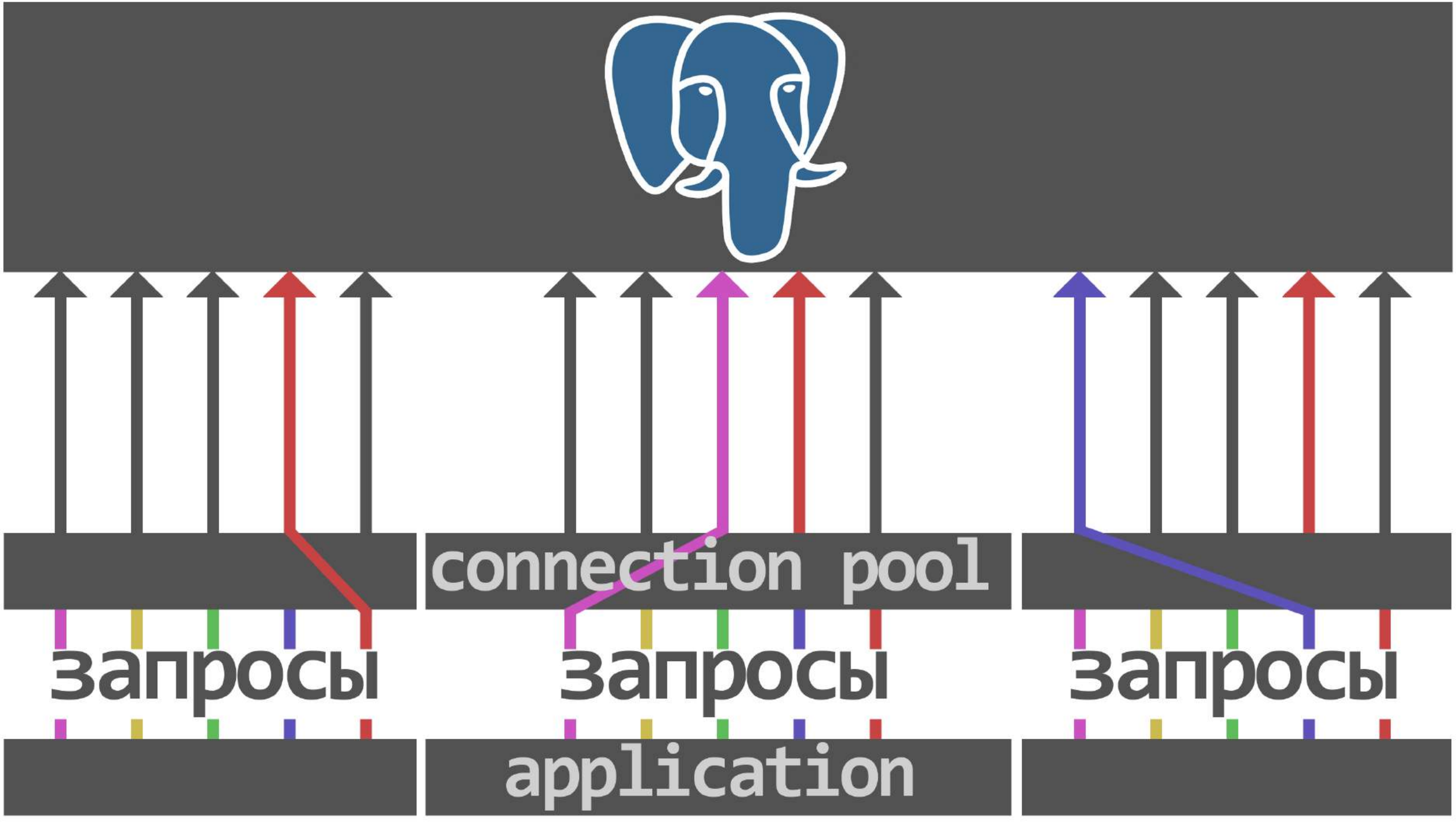
конец транзакции

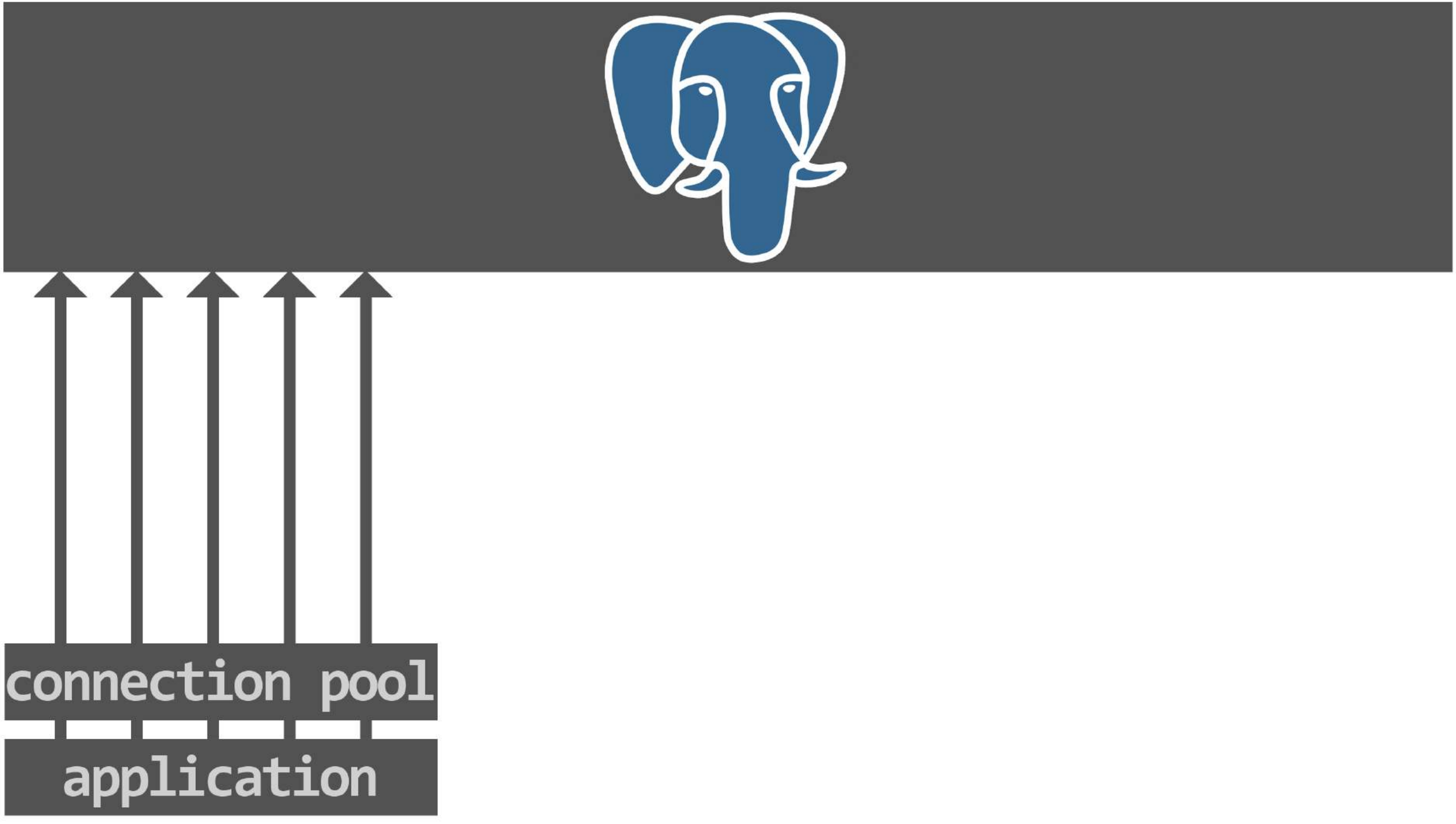


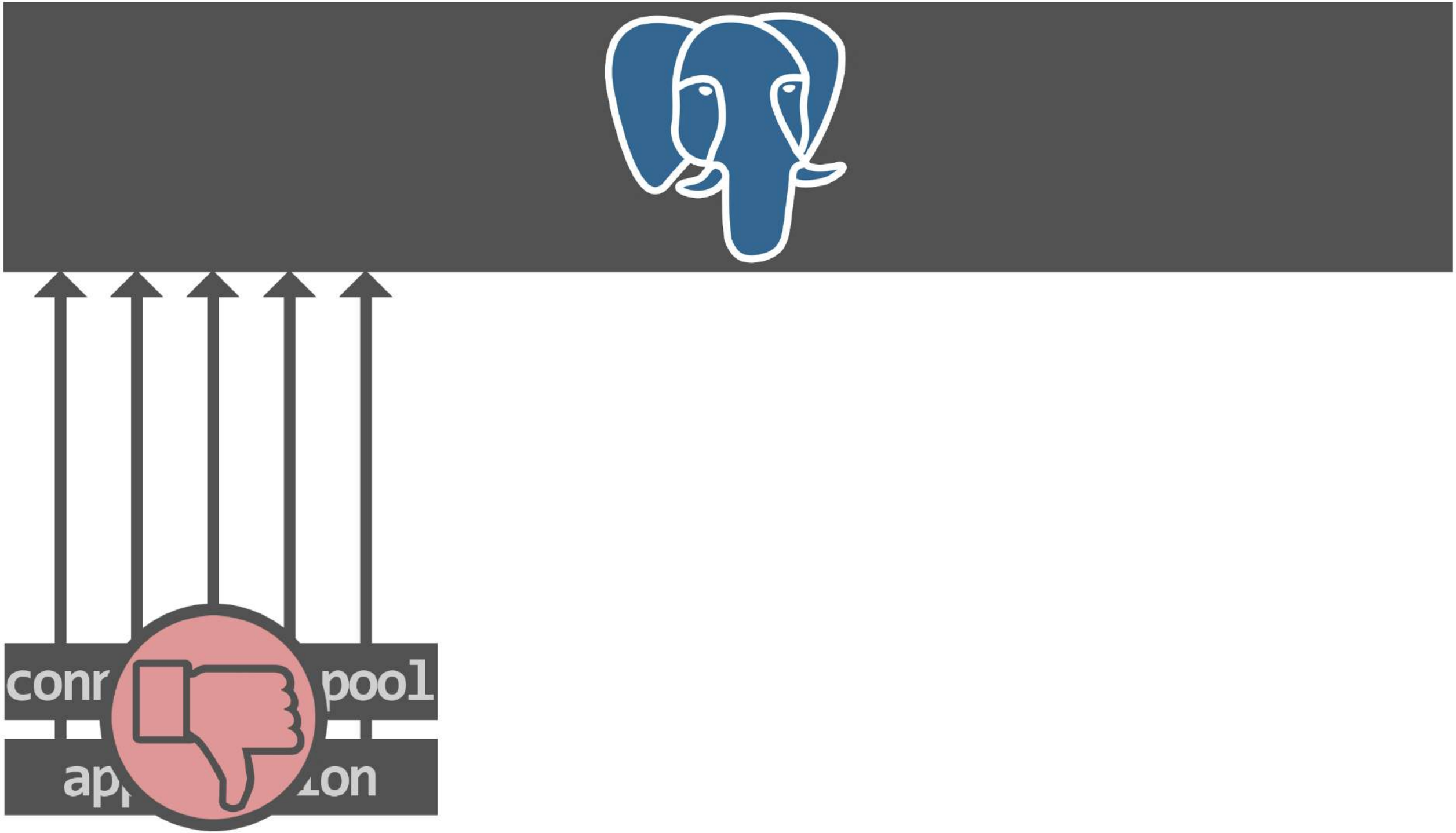
рендеринг веб страницы | сериализация json

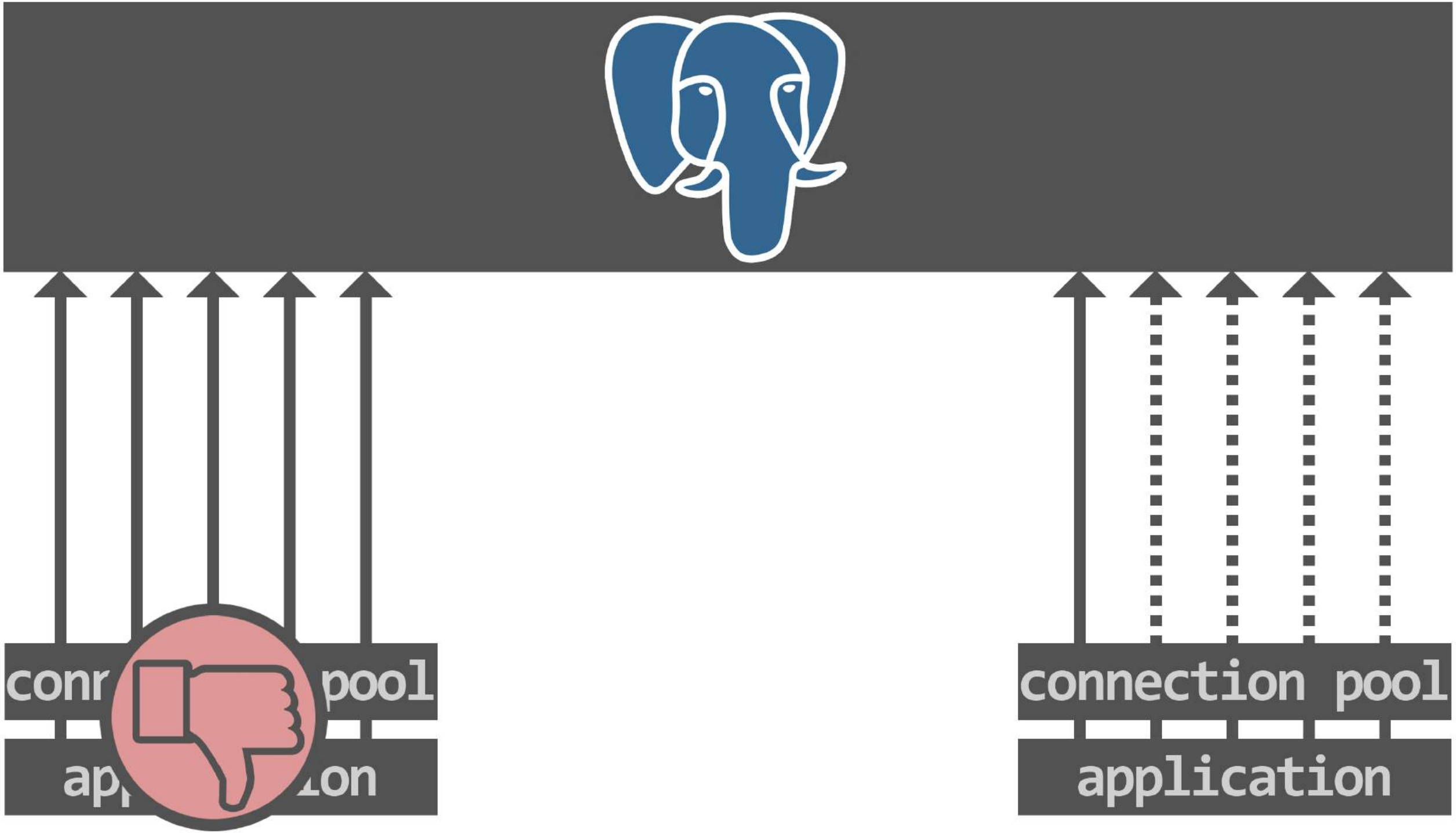
```
spring.jpa.open-in-view=false
```

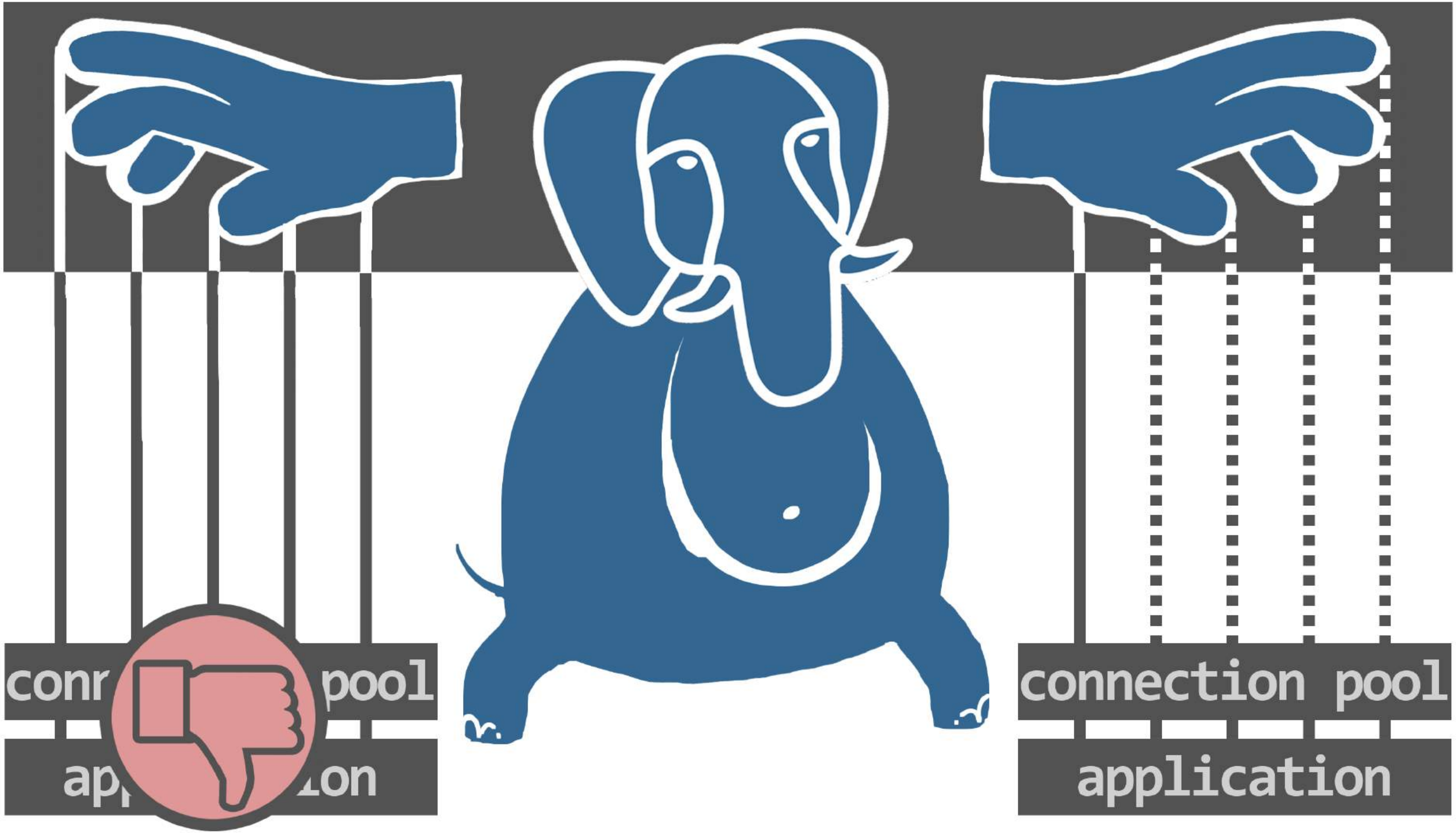
~~закрытие коннекшена~~

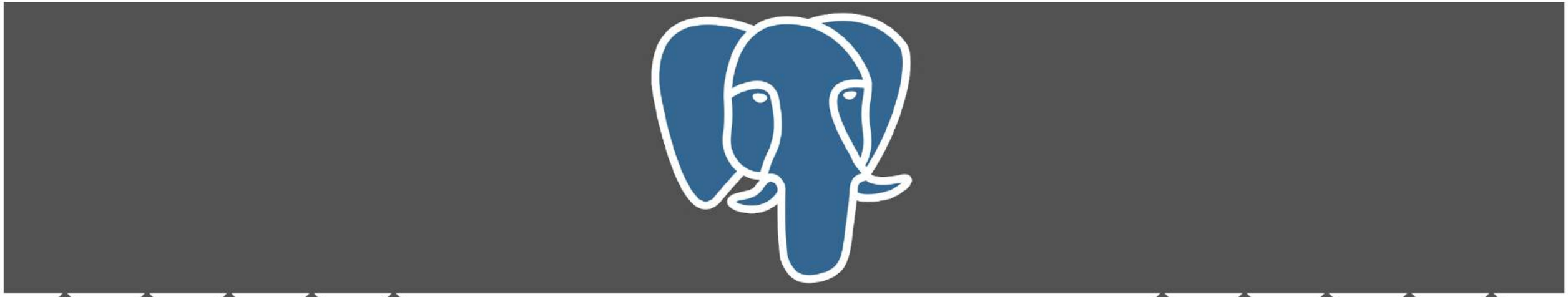

connection pool
запросы
запросы
запросы
application


connection pool
application


conr
pool
ap
on

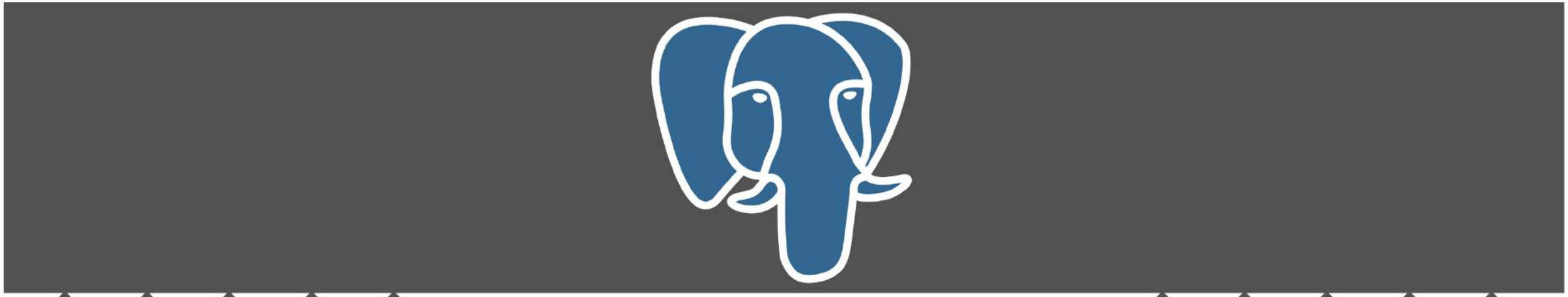

connection pool
application
connection pool
application


connection pool
application
connection pool
application




conr
pool
ap
on

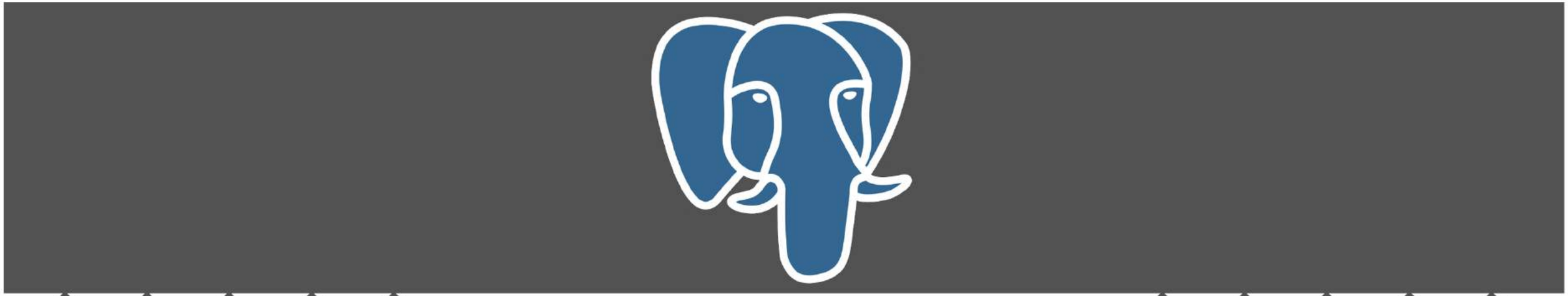

минимальное
количество
коннекшнов
максимальное
количество
коннекшнов
connection pool
application




```yaml
spring:
  datasource:
    hikari:
      minimumIdle: 1
      maximum-pool-size: 5
      idle-timeout: 60000
```

минимальное количество коннекшнов

максимальное количество коннекшнов

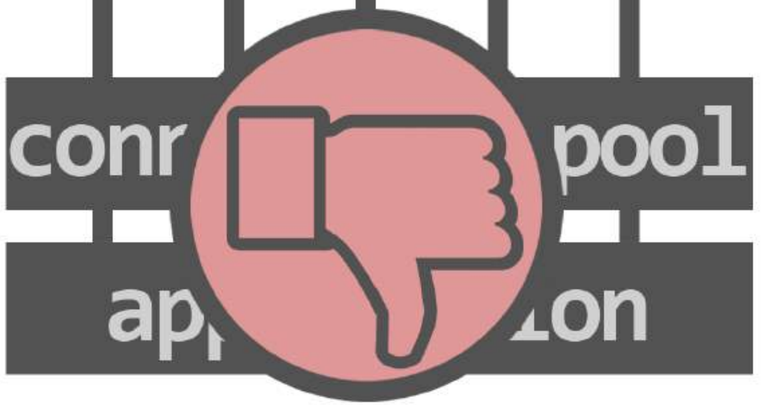


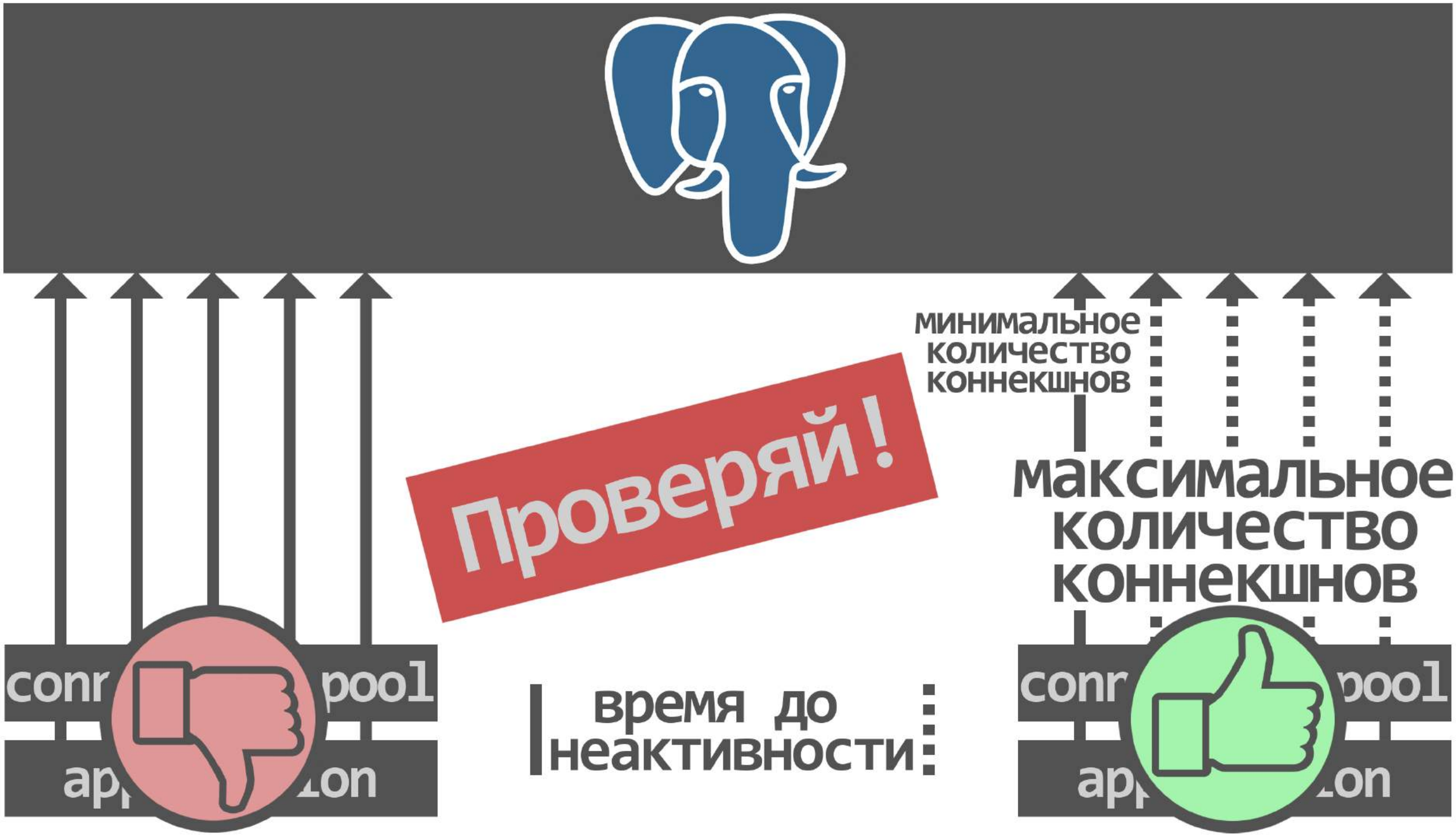
```yaml
spring:
  datasource:
    hikari:
      minimumIdle: 1
      maximum-pool-size: 5
      idle-timeout: 60000
```

минимальное количество коннекшнов

максимальное количество коннекшнов

время до неактивности

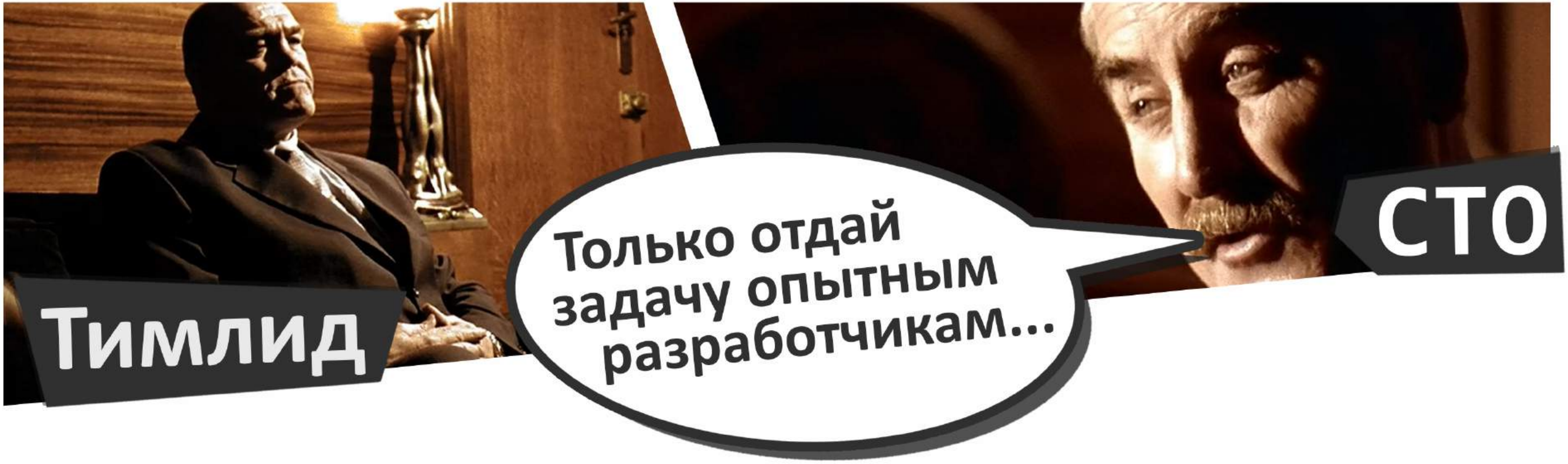



минимальное количество коннекшнов
максимальное количество коннекшнов
Проверяй!
время до неактивности


CTO


Тимлид


СТО


Тимлид
Только отдай задачу опытным разработчикам...
СТО

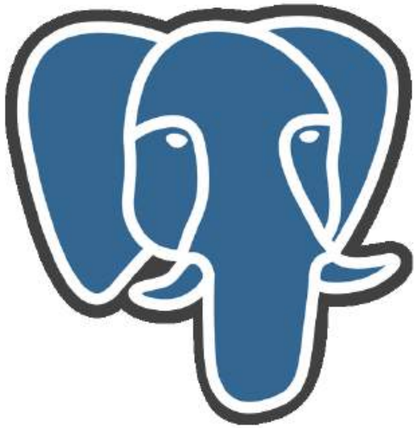

Тимлид
Только отдай задачу опытным разработчикам...
CTO
Senior Developer
Lead Developer


Тимлид
Только отдай задачу опытным разработчикам...
СТО
Senior Developer
Lead Developer
ДБА

Liquibase


Liquibase

Liquibase




Liquibase
10


```
management.endpoint.liquibase.enabled=false
```


```
management.endpoint.liquibase.enabled=false
```

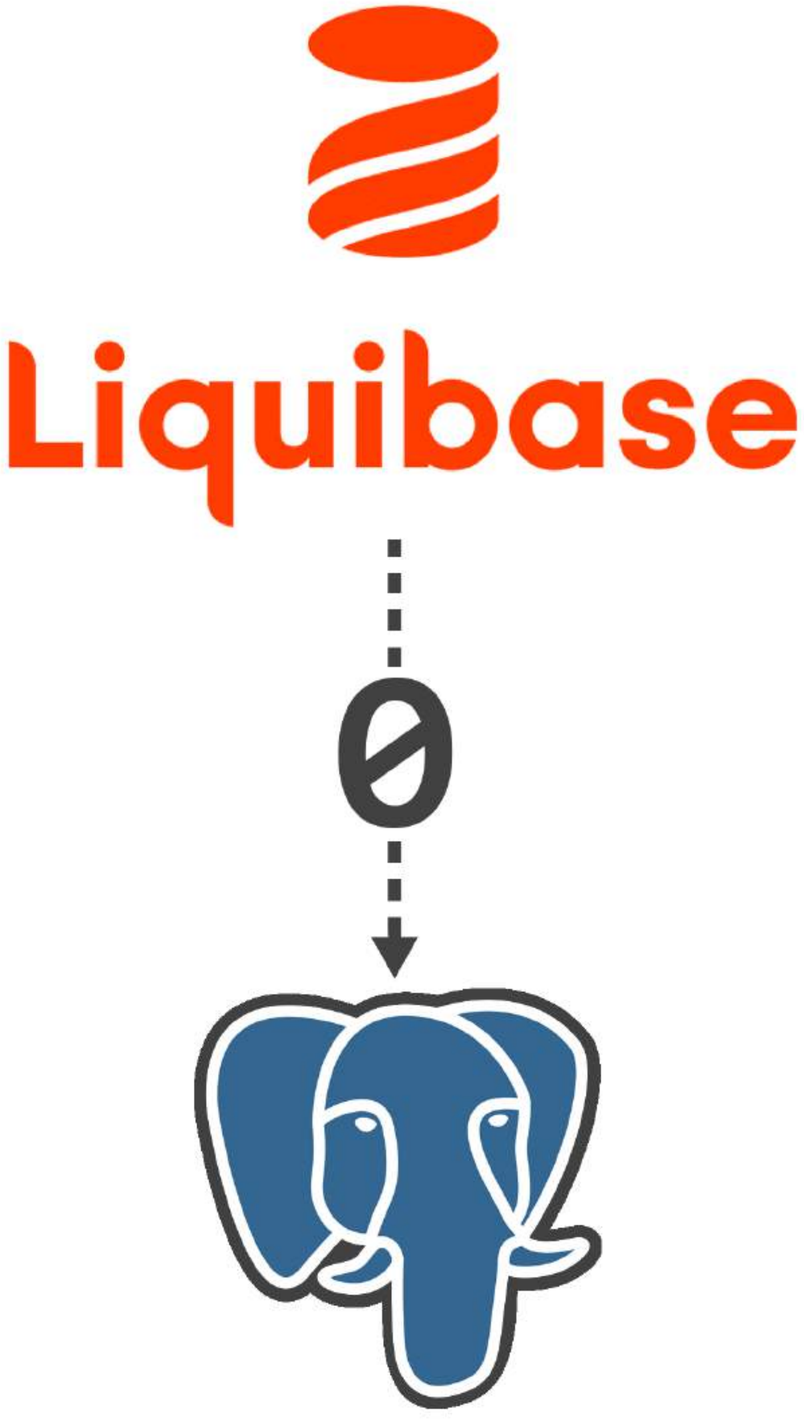
```
public class JpaConfiguration {
    @Bean
    public DataSource wrappedDataSource(
            @Value("${spring.datasource.url}") String url,
            @Value("${spring.datasource.driver-class-name}") String driverClassName,
            @Value("${spring.datasource.username}") String username,
            @Value("${spring.datasource.password}") String password) {
        return DataSourceBuilder.create()
                .driverClassName(driverClassName)
                .url(url)
                .username(username)
                .password(password)
                .build();
    }
}
```


```
management.endpoint.liquibase.enabled=false
```

```
public class JpaConfiguration {
    @LiquibaseDataSource
    @Bean
    public DataSource wrappedDataSource(
            @Value("${spring.datasource.url}") String url,
            @Value("${spring.datasource.driver-class-name}") String driverClassName,
            @Value("${spring.datasource.username}") String username,
            @Value("${spring.datasource.password}") String password,
            @Value("${spring.datasource.hikari.maximum-pool-size:5}") int poolSize,
            @Value("${spring.datasource.hikari.minimum-idle:1}") int minimumIdle,
            @Value("${spring.datasource.hikari.idle-timeout:60000}") long idleTimeout
    ) {
        DataSource ds = DataSourceBuilder.create()
                .driverClassName(driverClassName)
                .username(username)
                .password(password)
                .url(url)
                .build();
        if (ds instanceof HikariDataSource) {
            ((HikariDataSource) ds).setMaximumPoolSize(poolSize);
            ((HikariDataSource) ds).setMinimumIdle(minimumIdle);
            ((HikariDataSource) ds).setIdleTimeout(idleTimeout);
        }
        return ds;
    }
}
```

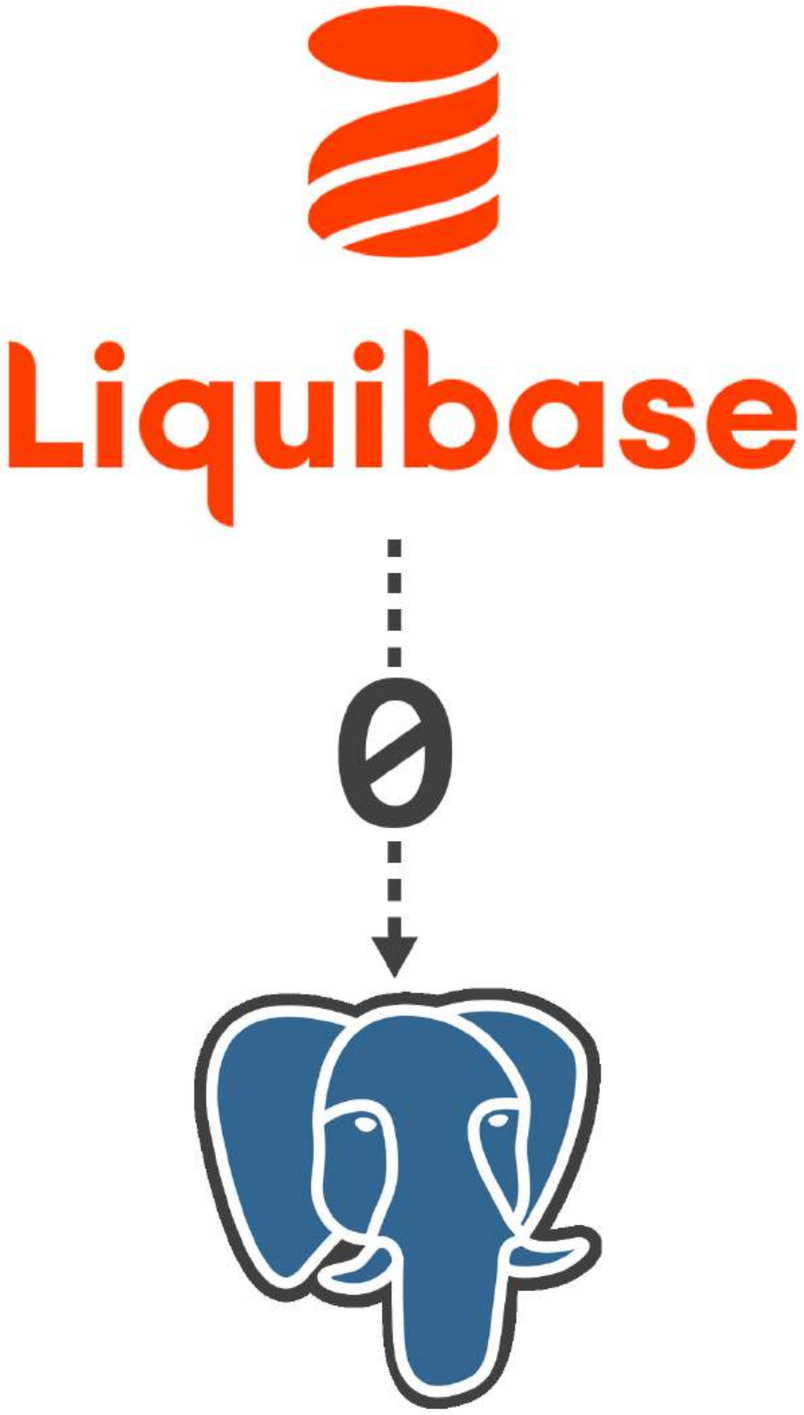

```
management.endpoint.liquibase.enabled=false
```

```
@EnableJpaAuditing
public class JpaConfiguration {
    @LiquibaseDataSource
    @Bean
    public DataSource wrappedDataSource(
            @Value("${spring.datasource.url}") String url,
            @Value("${spring.datasource.driver-class-name}") String driverClassName,
            @Value("${spring.datasource.username}") String username,
            @Value("${spring.datasource.password}") String password) {
        return DataSourceBuilder.create()
            @Value("${spring.datasource.password}") String password,
            @Value("${spring.datasource.hikari.maximum-pool-size:5}") int poolSize,
            @Value("${spring.datasource.hikari.minimum-idle:1}") int minimumIdle,
            @Value("${spring.datasource.hikari.idle-timeout:60000}") long idleTimeout
    ) {
        DataSource ds = DataSourceBuilder.create()
                .driverClassName(driverClassName)
                .url(url)
                .username(username)
                .password(password)
                .url(url)
                .build();
        if (ds instanceof HikariDataSource) {
            ((HikariDataSource) ds).setMaximumPoolSize(poolSize);
            ((HikariDataSource) ds).setMinimumIdle(minimumIdle);
            ((HikariDataSource) ds).setIdleTimeout(idleTimeout);
        }
        return ds;
    }
}
```

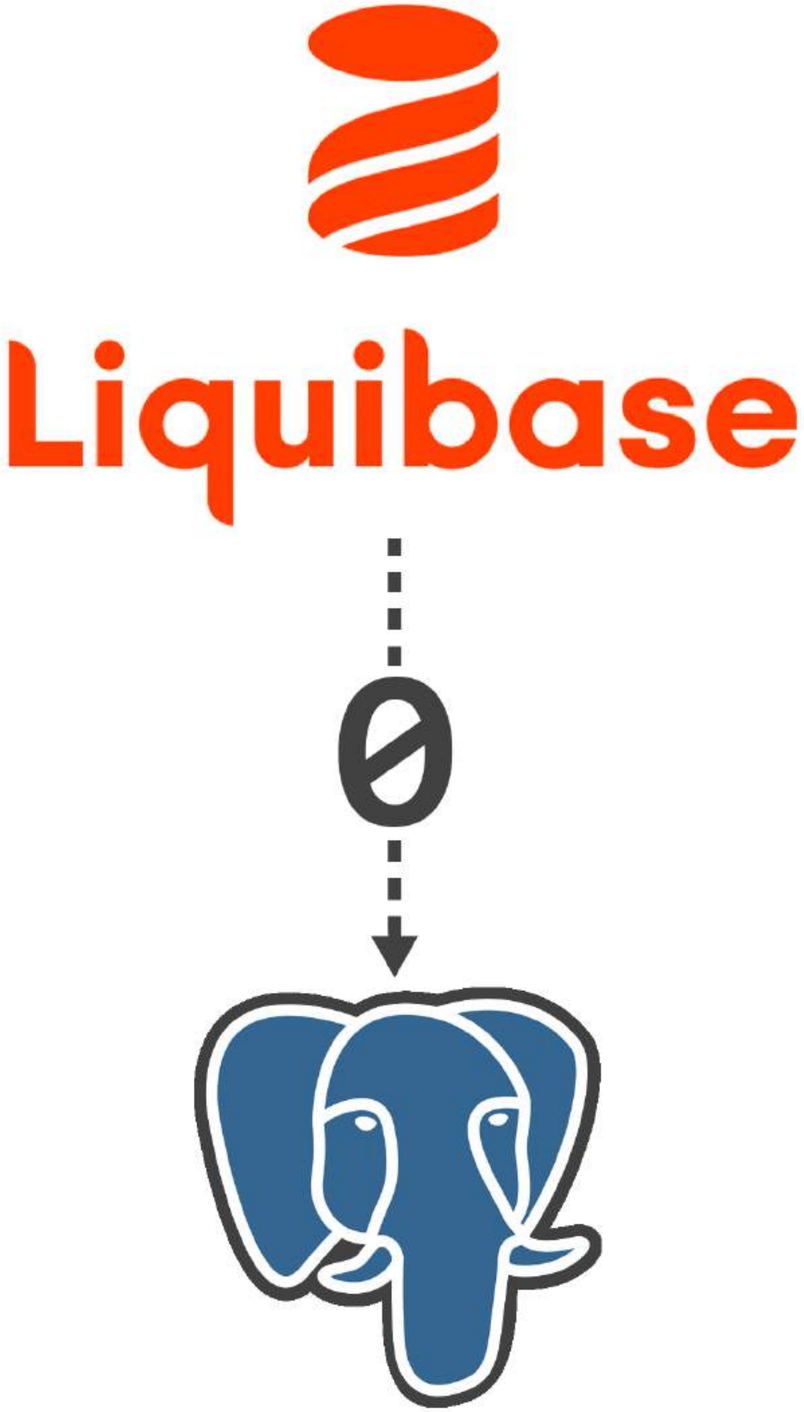


management.endpoint.liquibase.enabled=false

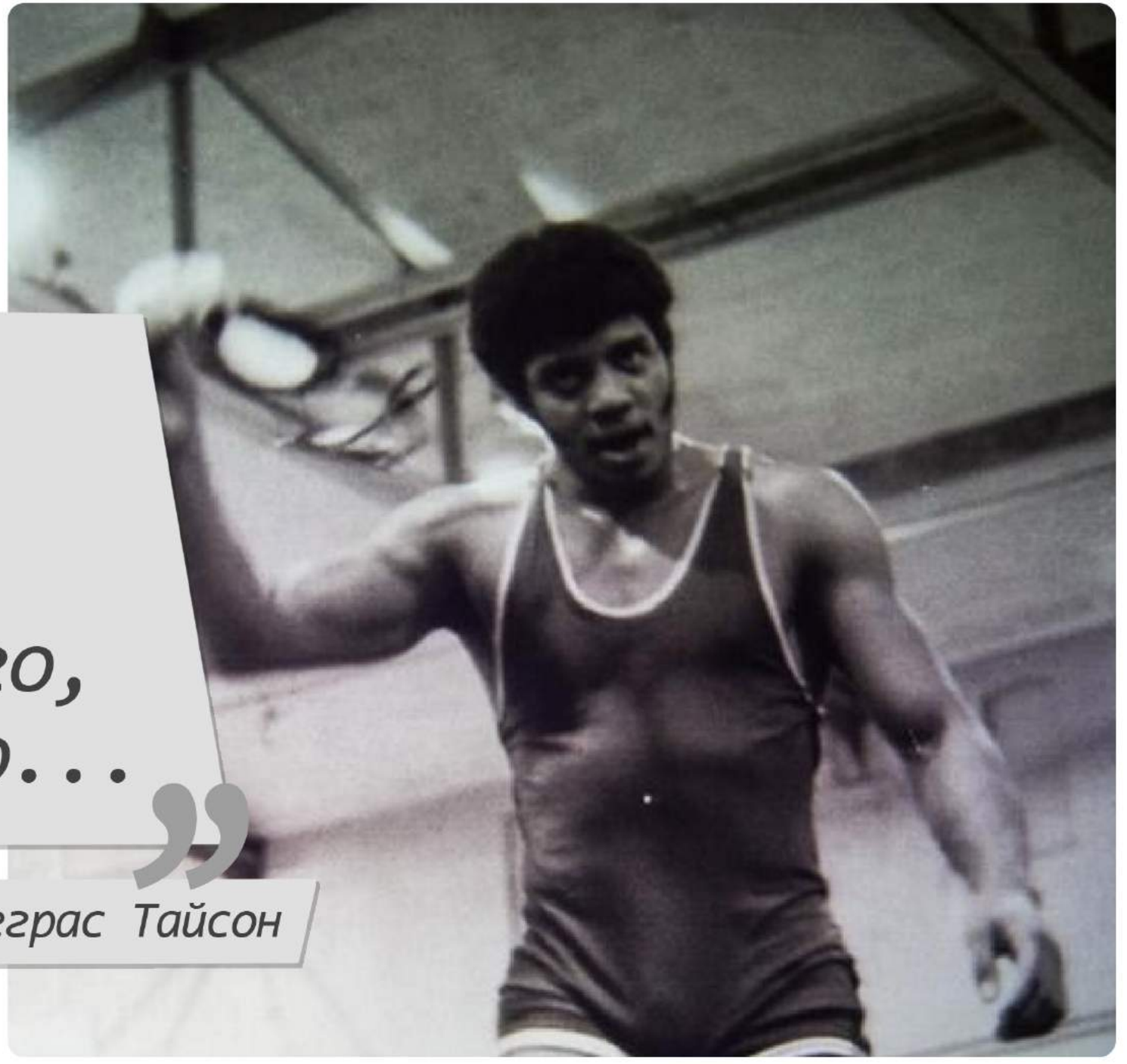
```
12 14   @EnableJpaAuditing
13 15   public class JpaConfiguration {
   16 +        @LiquibaseDataSource
14 17       @Bean
15 18       public DataSource wrappedDataSource(
16 19               @Value("${spring.datasource.url}") String url,
17 20               @Value("${spring.datasource.driver-class-name}") String driverClassName,
18 21               @Value("${spring.datasource.username}") String username,
19    -             @Value("${spring.datasource.password}") String password) {
20    -         return DataSourceBuilder.create()
   22 +             @Value("${spring.datasource.password}") String password,
   23 +             @Value("${spring.datasource.hikari.maximum-pool-size:5}") int poolSize,
   24 +             @Value("${spring.datasource.hikari.minimum-idle:1}") int minimumIdle,
   25 +             @Value("${spring.datasource.hikari.idle-timeout:60000}") long idleTimeout
   26 +     ) {
   27 +         DataSource ds = DataSourceBuilder.create()
21 28                   .driverClassName(driverClassName)
22    -              .url(url)
23 29                   .username(username)
24 30                   .password(password)
   31 +              .url(url)
25 32                   .build();
   33 +         if (ds instanceof HikariDataSource) {
   34 +             ((HikariDataSource) ds).setMaximumPoolSize(poolSize);
   35 +             ((HikariDataSource) ds).setMinimumIdle(minimumIdle);
   36 +             ((HikariDataSource) ds).setIdleTimeout(idleTimeout);
   37 +         }
   38 +         return ds;
26 39       }
27 40   }
```




```java
@Configuration
public class JpaConfiguration {

    @Primary
    @Bean(name = "dsProperties")
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSourceProperties getDataSourceProperties() {
        return new DataSourceProperties();
    }

    @Bean(name = "hikariProperties")
    @ConfigurationProperties(prefix = "spring.datasource.hikari")
    public Properties hikariProperties() {
        return new Properties();
    }

    @LiquibaseDataSource
    @Bean
    public DataSource wrappedDataSource(
            @Qualifier("dsProperties")
            DataSourceProperties dataSourceProperties,
            @Qualifier("hikariProperties") Properties properties) {
        HikariDataSource ds = dataSourceProperties
                .initializeDataSourceBuilder().type(HikariDataSource.class)
                .build();
        if (ds != null) {
            Properties hikariProperties = new Properties();
            properties.forEach((o, o2) -> hikariProperties.setProperty(
                    toCamelCase((String) o, false), (String) o2));
            PropertyElf.setTargetFromProperties(ds, hikariProperties);
        }
        return ds;
    }
}
```

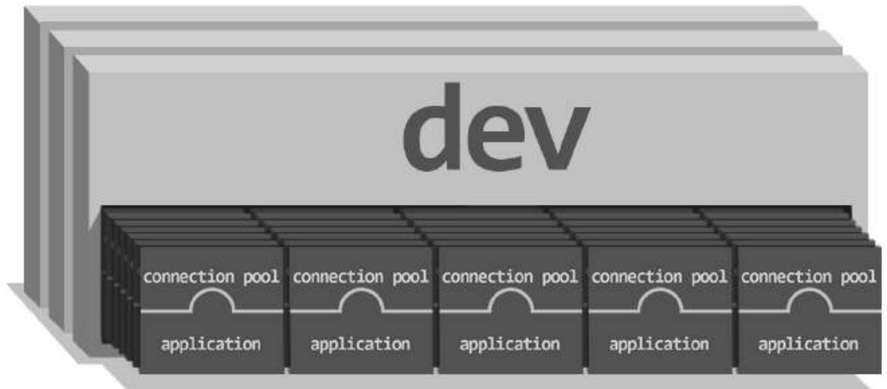


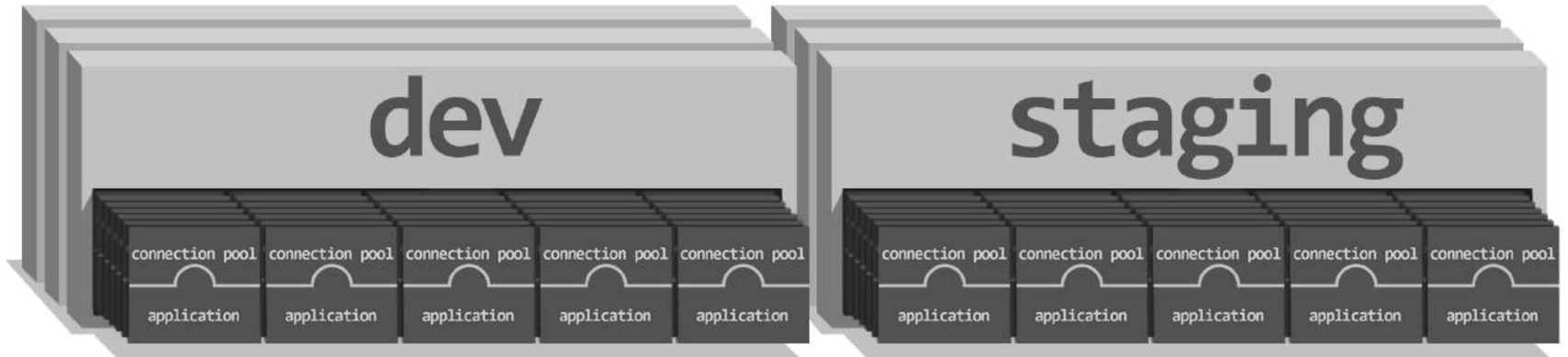

```java
@Configuration
public class JpaConfiguration {
    @Bean
    @Primary
    @LiquibaseDataSource
    @ConfigurationProperties(
            prefix = "spring.datasource.hikari")
    public HikariDataSource dataSource(
            DataSourceProperties properties
    ) {
        return newHikariDataSource(properties);
    }

    private HikariDataSource newHikariDataSource(
            DataSourceProperties props
    ) {
        var dataSource = new HikariDataSource();
        String driverClassName = props.getDriverClassName();
        if (driverClassName != null) {
            dataSource.setDriverClassName(driverClassName);
        }
        dataSource.setJdbcUrl(props.getUrl());
        dataSource.setUsername(props.getUsername());
        dataSource.setPassword(props.getPassword());
        return dataSource;
    }
}
```

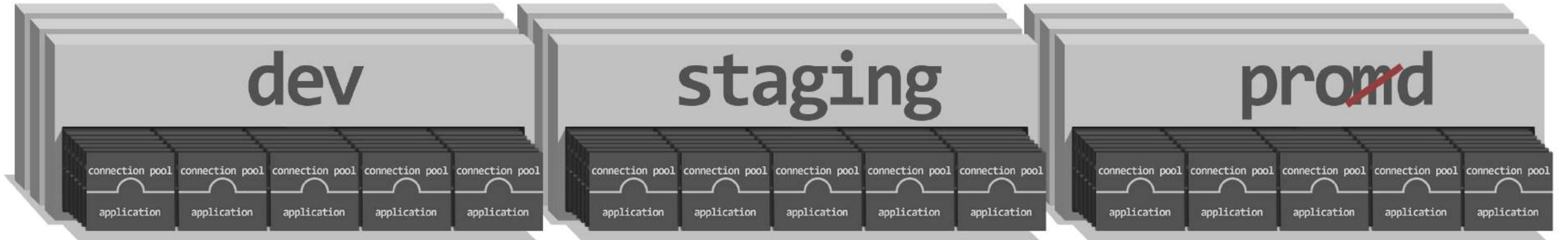


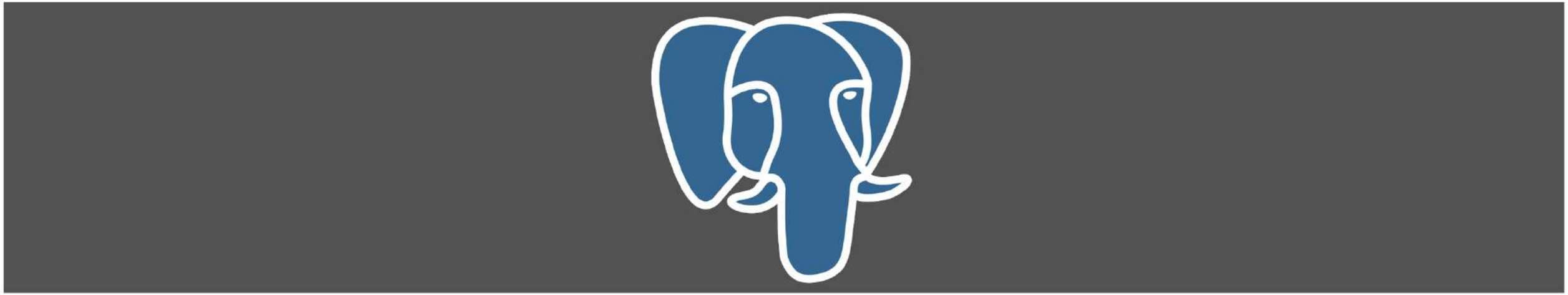
“Если вы нашли какую-то одну штуку, то, скорее всего, таких штук много...”

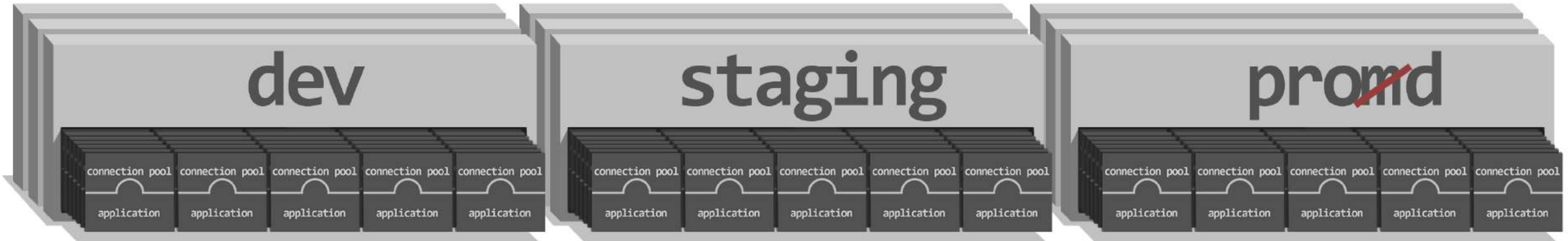
Неизвестный Нил Деграс Тайсон

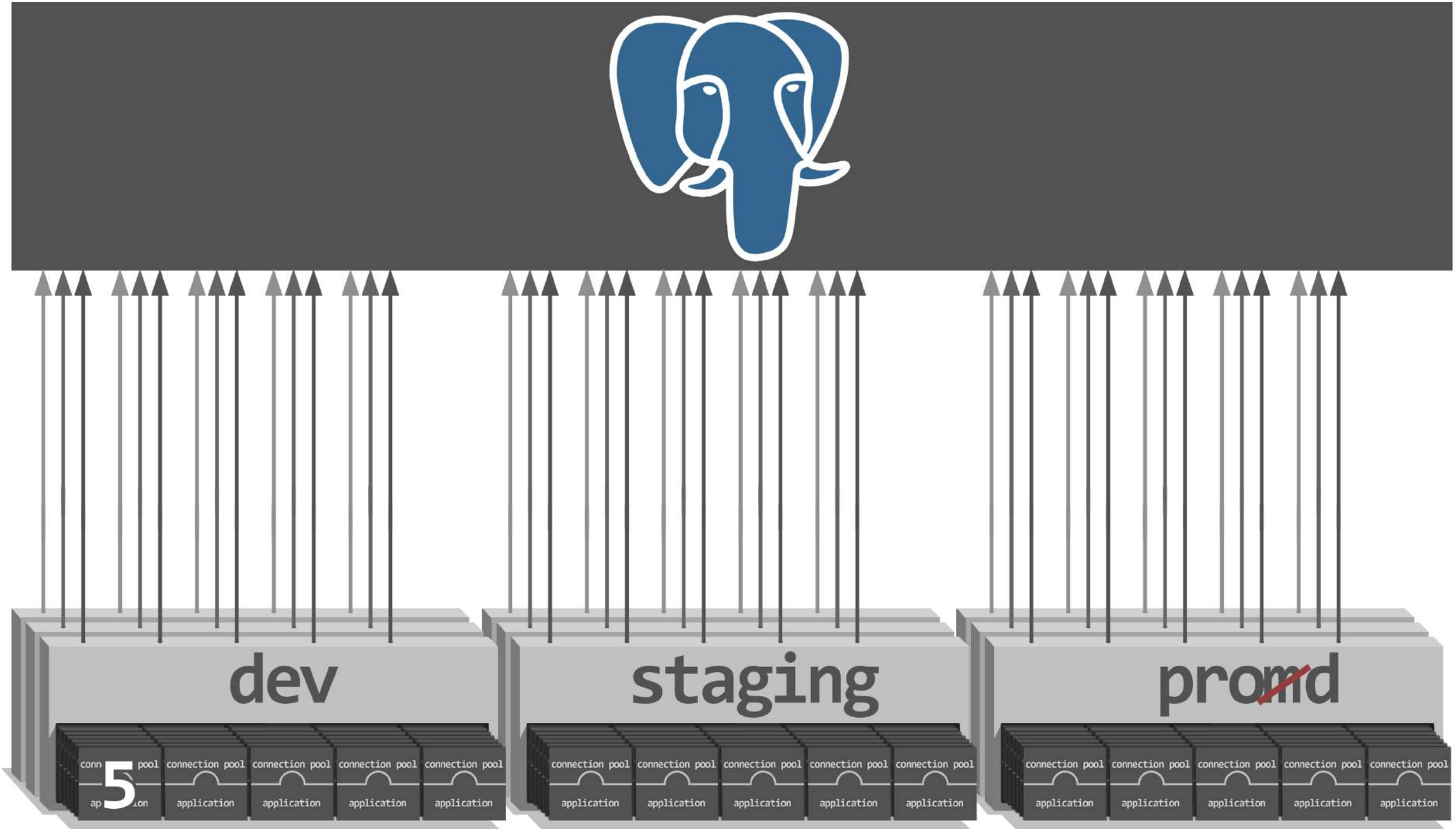

dev
connection pool
connection pool
connection pool
connection pool
connection pool
application
application
application
application
application

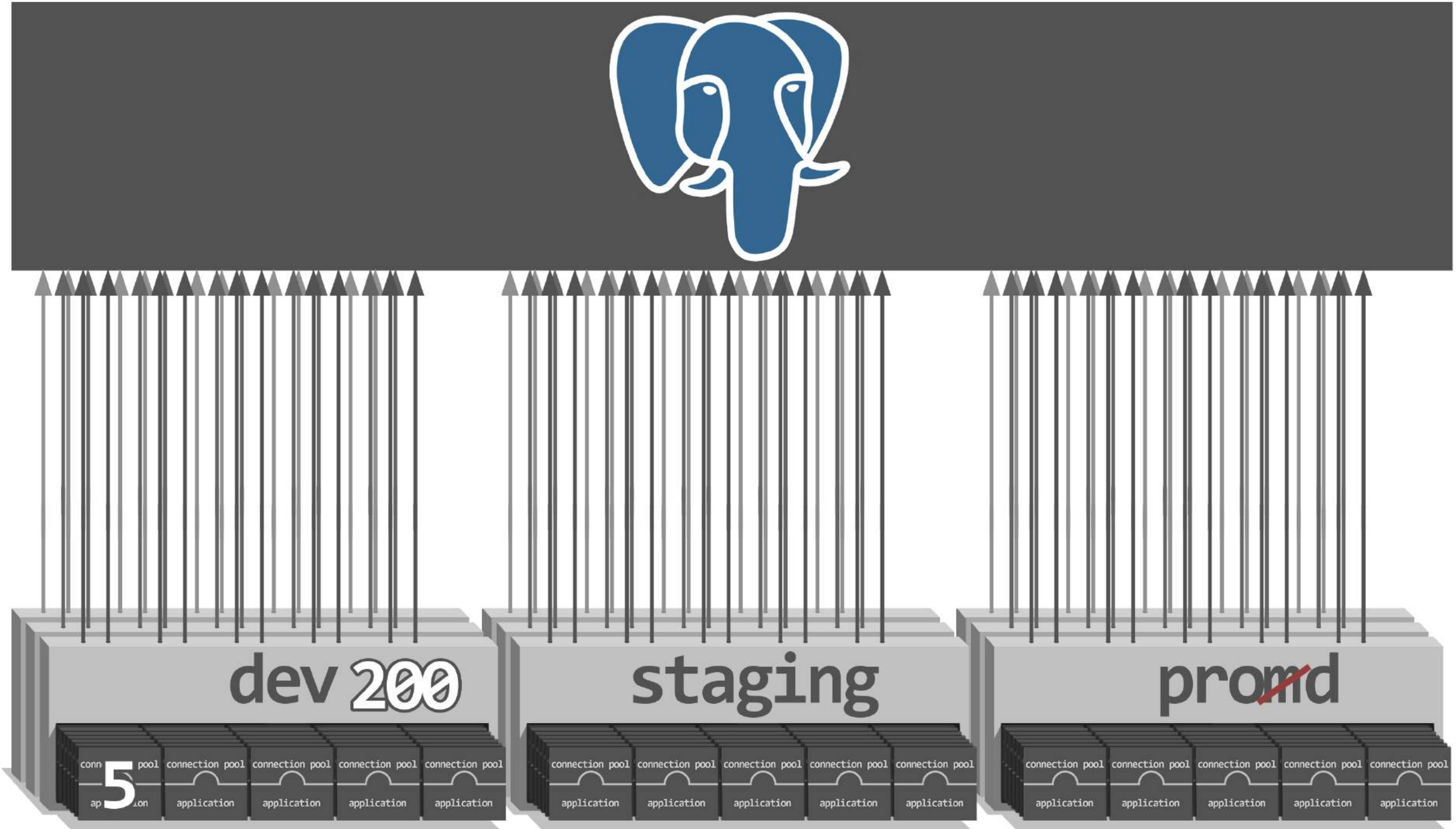

dev
staging
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application

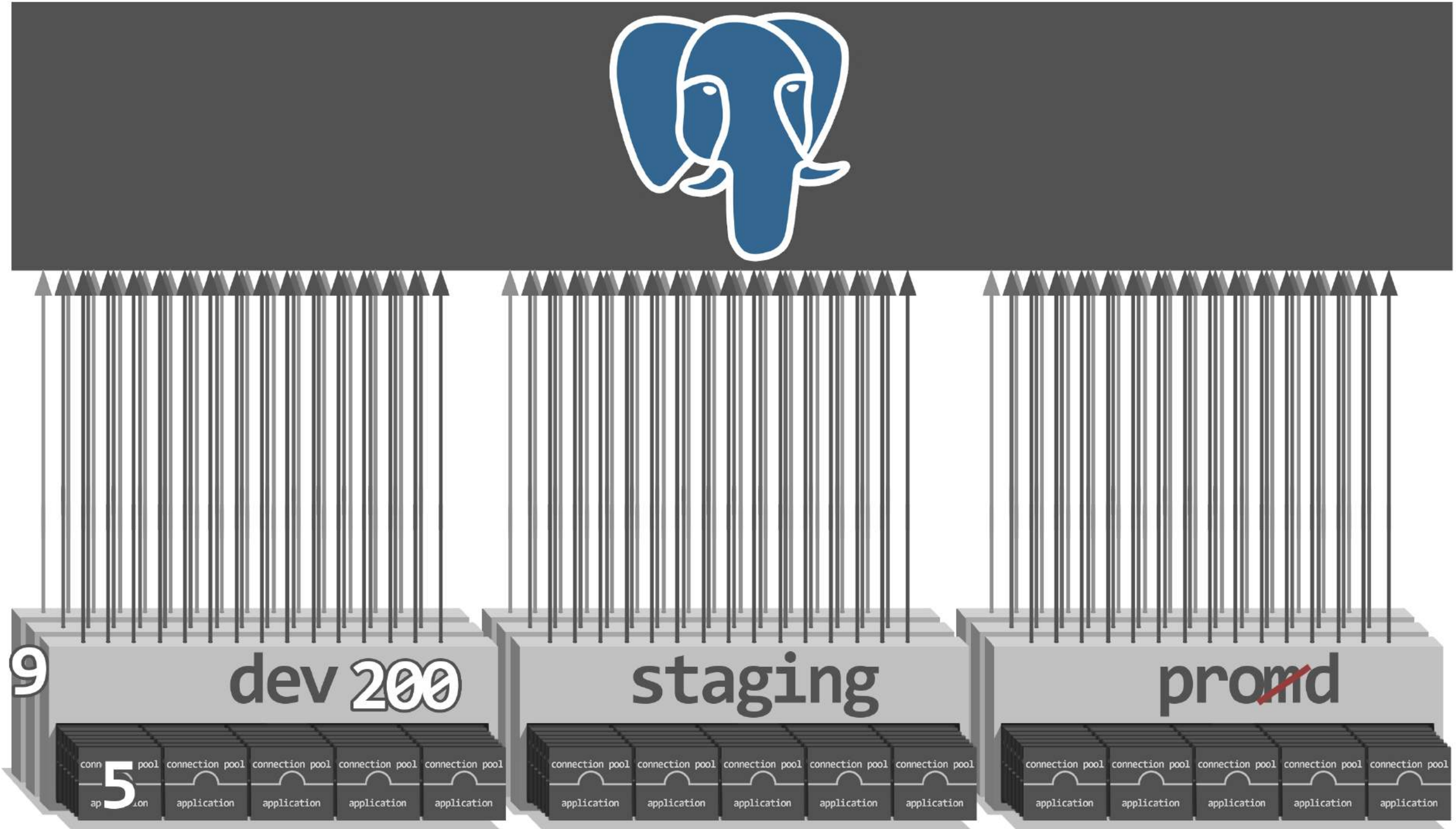

dev
staging
prod
connection pool
application

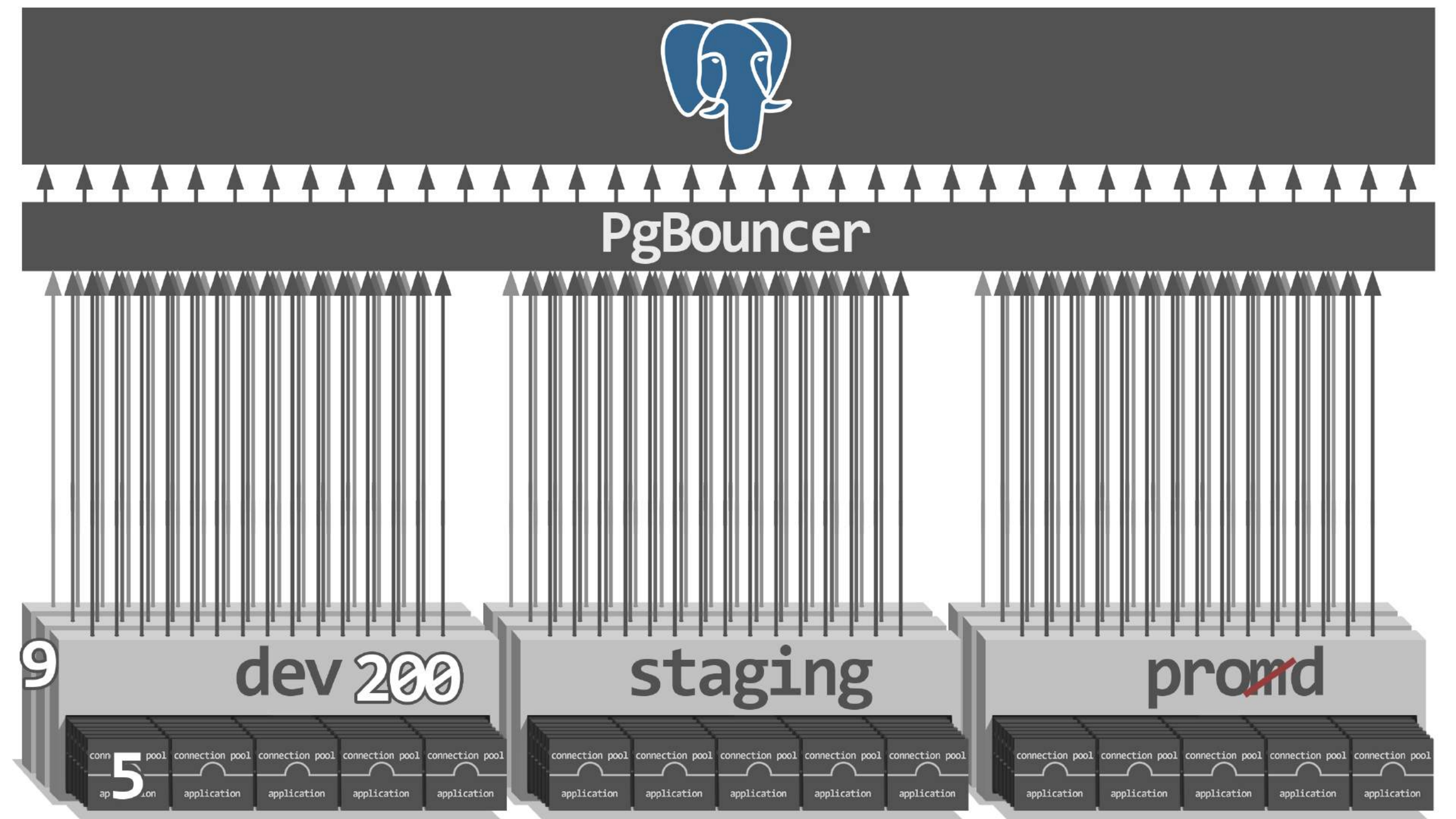



dev
connection pool
application
staging
connection pool
application
prod
connection pool
application


dev
staging
promd
connection pool
application
5


dev 200
staging
prod
5
connection pool
application


9
dev 200
staging
prod
5
connection pool
application


PgBouncer
9
dev 200
staging
prod
5
connection pool
application

Итого:

1. Использование фич СУБД
2. Простановка границ транзакций
3. Оптимизация запросов
4. Экономия коннекшнов
5. Медитация на внутреннее устройство СУБД

Илья Сазонов

@imsazonov

poxvuibr@gmail

@sazonovfm

sazonovfm@gmail

Фёдор Сазонов

Илья Сазонов

@imsazonov

poxvuibr@gmail

Спасибо!

@sazonovfm

sazonovfm@gmail

Фёдор Сазонов