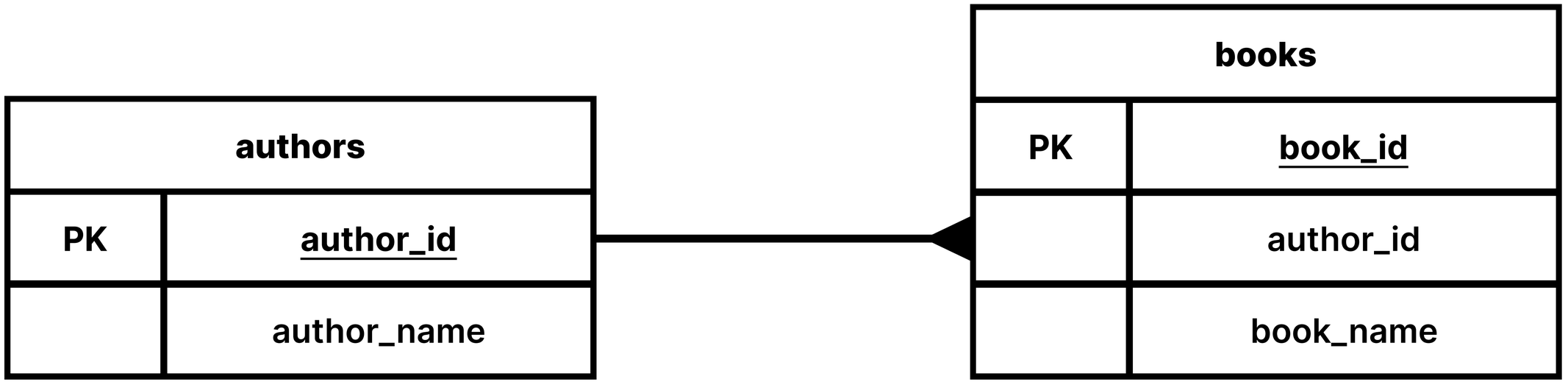
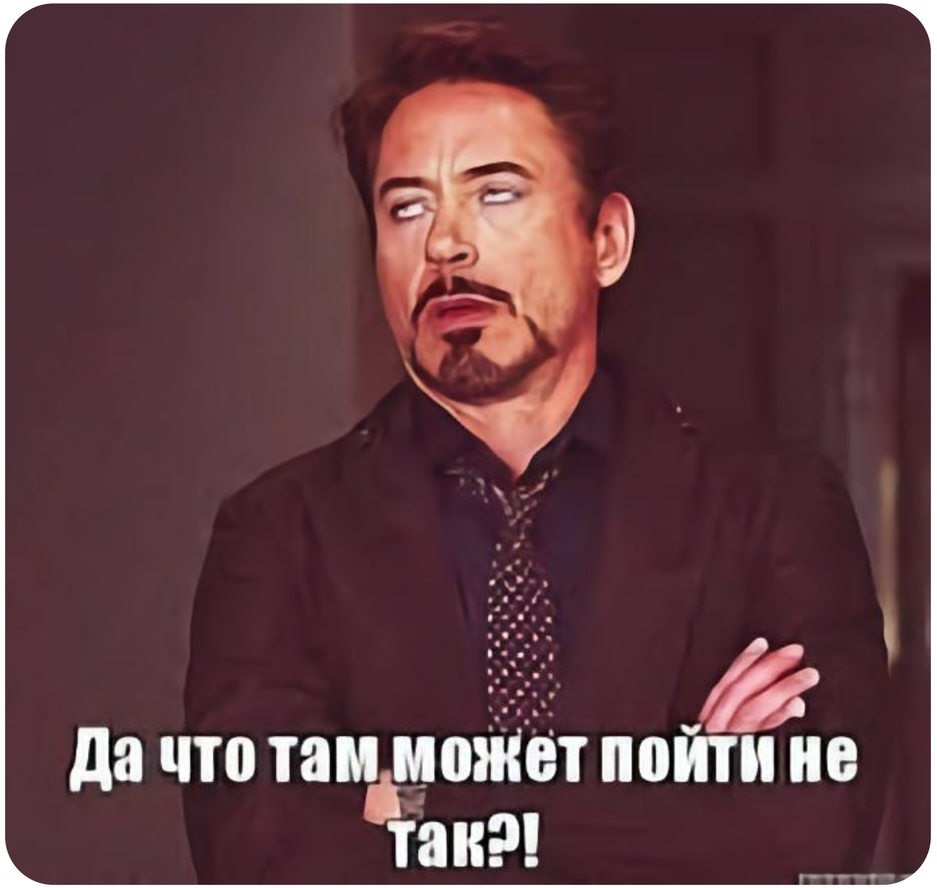
«Дух машины хранит
секреты древних.
Пренебрегая ритуалом,
ты отрекаешься от веры»

БД-укротитель


authors
PK
author_id
author_name
books
PK
book_id
author_id
book_name


Да что там может пойти не так?!

```
select
    *
from
    authors
;
```

```
select
    *
from
    authors
;
```

1 😊

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    authors
;
```

2 

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
        select
            *
        from
            books book
        where
            book.author_id=?
        ;
```

```
select
    *
from
    authors
;
```

4 😟

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    authors
;
```

7 

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    authors
;
```

N 

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

```
select
    *
from
    books book
where
    book.author_id=?
;
```

N+1

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```java
@Entity
@Table(name = "books")
public class Book {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "book_id")
    private String id;

    @Column(name = "book_name")
    private String name;

    @ManyToOne(
            cascade = CascadeType.ALL,
            fetch = FetchType.LAZY
    )
    @JoinColumn(name = "author_id")
    private Author author;

    //getters, setters...
}
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
logging.level.org.hibernate.SQL=DEBUG
```

```java
@Entity
@Table(name = "books")
public class Book {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "book_id")
    private String id;

    @Column(name = "book_name")
    private String name;

    @ManyToOne(
            cascade = CascadeType.ALL,
            fetch = FetchType.LAZY
    )
    @JoinColumn(name = "author_id")
    private Author author;

    //getters, setters...
}
```

```java
List<Author> authors = authorRepository.findAll();
for (Author author : authors) {
    List<Book> books = author.getBooks();
    System.out.println(books.size());
}
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
2
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
List<Author> authors = authorRepository.findAll();
for (Author author : authors) {
    List<Book> books = author.getBooks();
    System.out.println(books.size());
}
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_
```

```
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
2
```

```
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```
List<Author> authors = authorRepository.findAll();
for (Author author : authors) {
    List<Book> books = author.getBooks();
    System.out.println(books.size());
}
```

```
Hibernate:
select
    author0_.author_id as author_id1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?

select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
```

```
@Query("""
  select distinct a
  from Author a
  left join fetch a.books
""")
 List<Author> findAllFetch();
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    left outer join
  books books1_
    on author0_.author_id=books1_.author_id
```

```
@Query("""
  select distinct a
  from Author a
  left join fetch a.books
""")
 List<Author> findAllFetch();

@EntityGraph(
   attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph();
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    left outer join
  books books1_
    on author0_.author_id=books1_.author_id
```

```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
   attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph(
                Pageable pageable);
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    left outer join
  books books1_
    on author0_.author_id=books1_.author_id
```

```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
    attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph(
                Pageable pageable);
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as ratings3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    left outer join
  books books1_
    on author0_.author_id=books1_.author_id
```

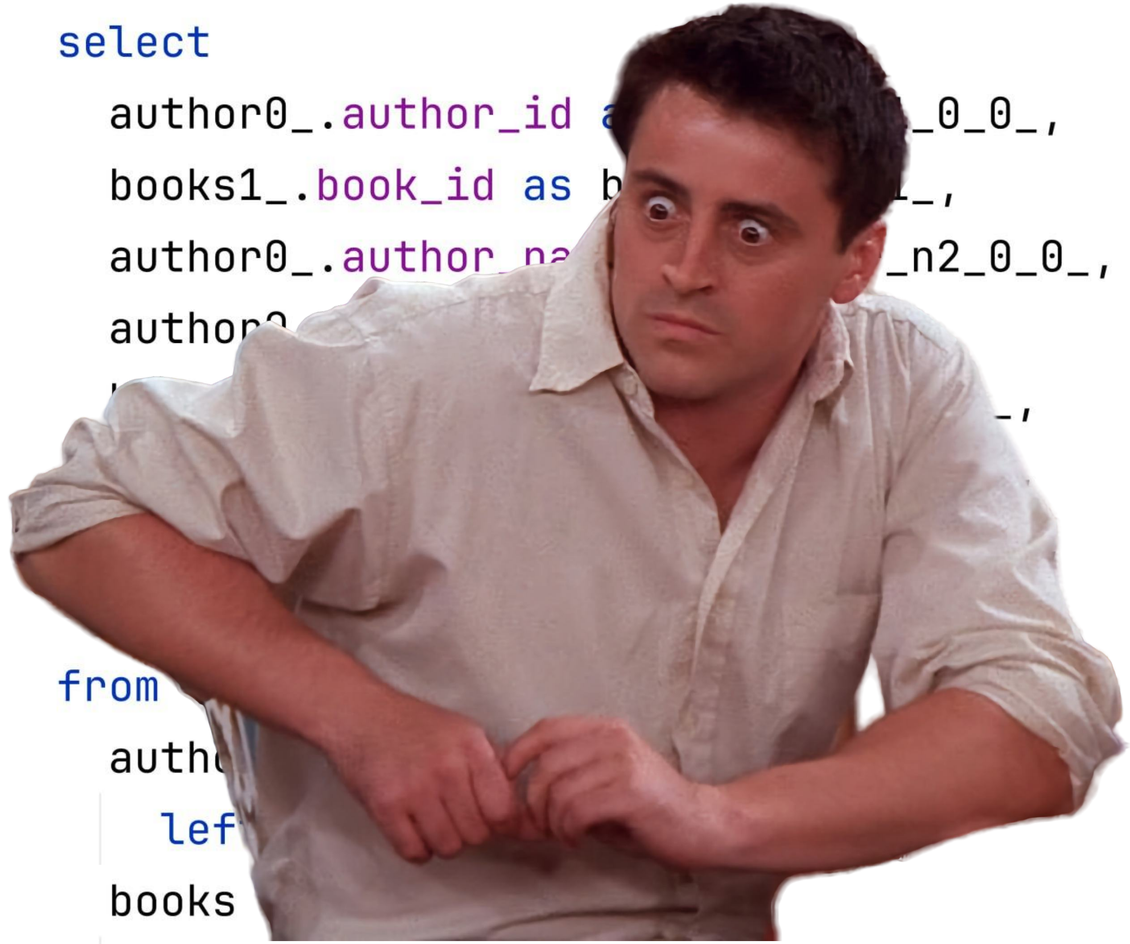
limit не завезли 😕

```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
   attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph(
                Pageable pageable);
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    left outer join
  books books1_
    on author0_.author_id=books1_.author_id
```

```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
    attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph(
                Pageable pageable);
```

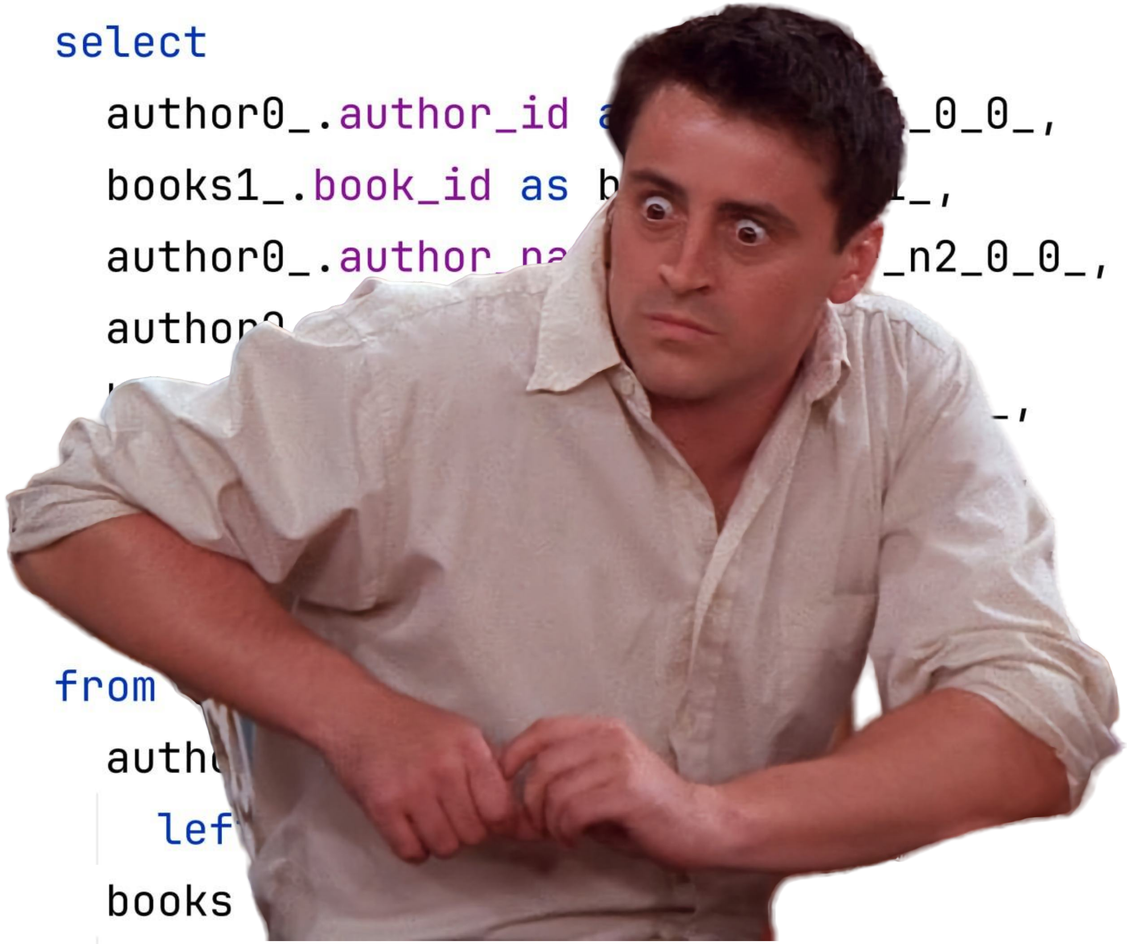
```
Hibernate:
select
  author0_.author_id a          _0_0_,
  books1_.book_id as b          _,
  author0_.author na          _n2_0_0_,
  author0
                                _,
from
  autho
    lef
  books
    on author0_.author_id=books1_.author_id
```



```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
    attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
List<Author> findAllGraph(
                Pageable pageable);
```

```
2024-04-21 13:52:02.464  WARN 95002 --- [
main] o.h.h.internal.ast.QueryTranslatorImpl
: HHH000104: firstResult/maxResults specified
with collection fetch; applying in memory!
```

```
Hibernate:
select
  author0_.author_id a
  books1_.book_id as b
  author0_.author na
  author
from
  auth
    lef
  books
    on author0_.author_id=books1_.author_id
```



```
@Query("""
  select a
  from Author a
  left join fetch a.books
""")
Slice<Author> findAllFetch(
        Pageable page);
@EntityGraph(
   attributePaths = {"books"})
@Query("""
  select a
  from Author a
""")
                Pageable pageable);
```

```
2024-04-21 13:52:02.464  WARN 95002 --- [
main] o.h.h.internal.ast.QueryTranslatorImpl
: HHH000104: firstResult/maxResults specified
with collection fetch; applying in memory!
```

```
Hibernate:
select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__
from
  authors author0_
    on author0_.author_id=books1_.author_id
```

```
spring.jpa.properties.hibernate.query.fail_on_pagination_over_collection_fetch=true
```

```
org.springframework.orm.jpa.JpaSystemException: firstResult/maxResults specified with collection fetch. In memory pagination was about to be applied. Failing because 'Fail on pagination
                                                over collection fetch' is enabled.; nested exception is org.hibernate.HibernateException: firstResult/maxResults specified with collection
                                                fetch. In memory pagination was about to be applied. Failing because 'Fail on pagination over collection fetch' is enabled.
    at org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:331)
    at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:233)
    at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExceptionIfPossible(AbstractEntityManagerFactoryBean.java:551)
    at org.springframework.dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible(ChainedPersistenceExceptionTranslator.java:61)
    at org.springframework.dao.support.DataAccessUtils.translateIfNecessary(DataAccessUtils.java:242)
    at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:152)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.data.jpa.repository.support.CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.invoke(CrudMethodMetadataPostProcessor.java:145)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:97)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:215)
    at jdk.proxy2/jdk.proxy2.$Proxy146.findAllFetch(Unknown Source)
    at fm.sazonov.dbhandler.service.AuthorServiceNPlusOneTest.testJoinFetch(AuthorServiceNPlusOneTest.java:135) <29 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
Caused by: org.hibernate.HibernateException Create breakpoint : firstResult/maxResults specified with collection fetch. In memory pagi                                  because 'Fail
                                                on pagination over collection fetch' is enabled. <8 internal lines>
    at org.springframework.orm.jpa.SharedEntityManagerCreator$DeferredQueryInvocationHandler.invoke(SharedEntityManagerCreator.ja
    at jdk.proxy2/jdk.proxy2.$Proxy166.getResultList(Unknown Source)
    at org.springframework.data.jpa.repository.query.JpaQueryExecution$CollectionExecution.doExecute(JpaQueryExecution.java:128)
    at org.springframework.data.jpa.repository.query.JpaQueryExecution.execute(JpaQueryExecution.java:90)
    at org.springframework.data.jpa.repository.query.AbstractJpaQuery.doExecute(AbstractJpaQuery.java:156)
    at org.springframework.data.jpa.repository.query.AbstractJpaQuery.execute(AbstractJpaQuery.java:144)
    at org.springframework.data.repository.core.support.RepositoryMethodInvoker.doInvoke(RepositoryMethodInvoker.java:137)
    at org.springframework.data.repository.core.support.RepositoryMethodInvoker.invoke(RepositoryMethodInvoker.java:121)
    at org.springframework.data.repository.core.support.QueryExecutorMethodInterceptor.doInvoke(QueryExecutorMethodInterceptor.java
    at org.springframework.data.repository.core.support.QueryExecutorMethodInterceptor.invoke(QueryExecutorMethodInterceptor.java:1
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.data.projection.DefaultMethodInvokingMethodInterceptor.invoke(DefaultMethodInvokingMethodInterceptor.jav
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.transaction.interceptor.TransactionInterceptor$1.proceedWithInvocation(TransactionInterceptor.java:1
    at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.j
    at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:119)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
```

```
spring.jpa.properties.hibernate.query.fail_on_pagination_over_collection_fetch=true
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
                        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
                        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

Вот так можно!

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

Вот так можно!

```
spring.jpa.properties.hibernate.default_batch_fetch_size=2
spring.jpa.properties.hibernate.batch_fetch_style=dynamic
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

Вот так можно!

это не то

```
spring.jpa.properties.hibernate.jdbc.batch_size=1000
```

```
spring.jpa.properties.hibernate.default_batch_fetch_size=2
spring.jpa.properties.hibernate.batch_fetch_style=dynamic
```

```
Hibernate:
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_ limit ?

Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
        ?, ?
    )
2
1
Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id=?
1
```

```
spring.jpa.properties.
  hibernate.default_batch_fetch_size=2
```

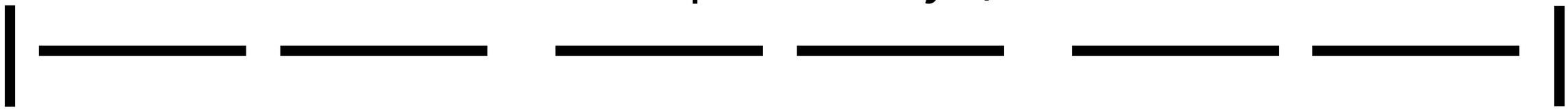
Количество родительских сущностей, для которых мы одновременно вытаскиваем дочерние коллекции

```
spring.jpa.properties.
  hibernate.jdbc.batch_size=1000
```

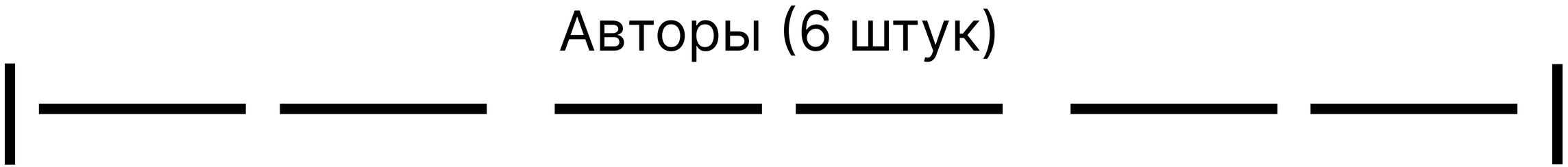
Количество апдейтов, инсертов или делитов за раз

НЕ ПЕРЕПУТАЙ!

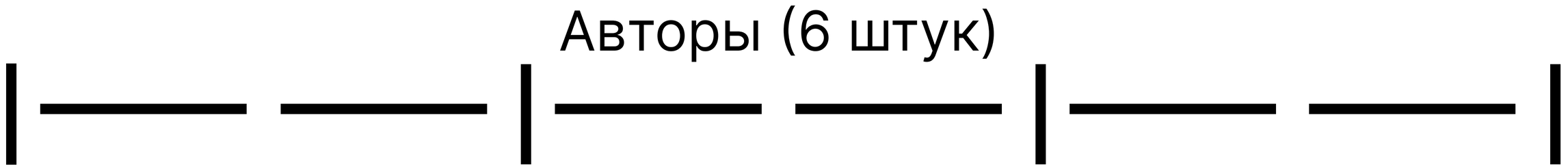
Авторы (6 штук)



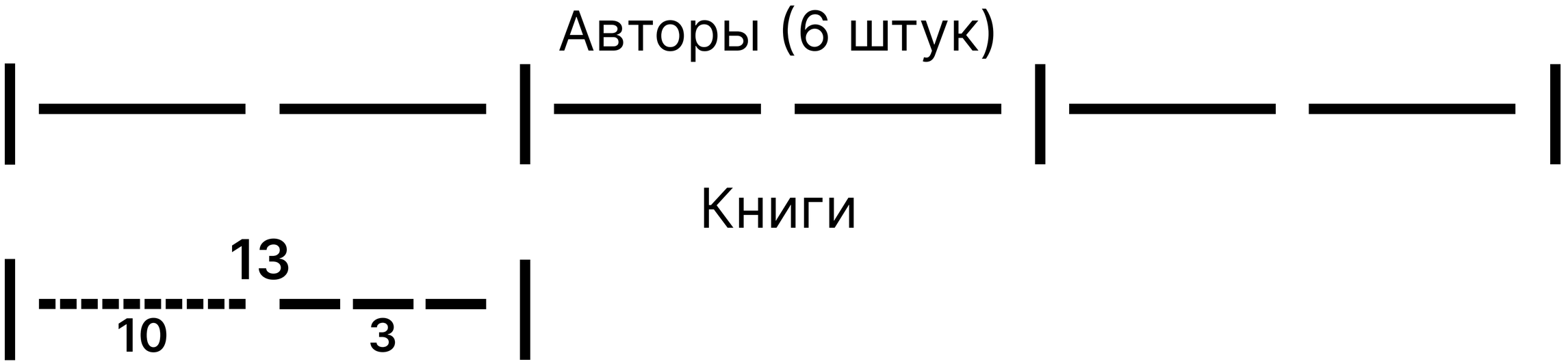
BatchSize = 2



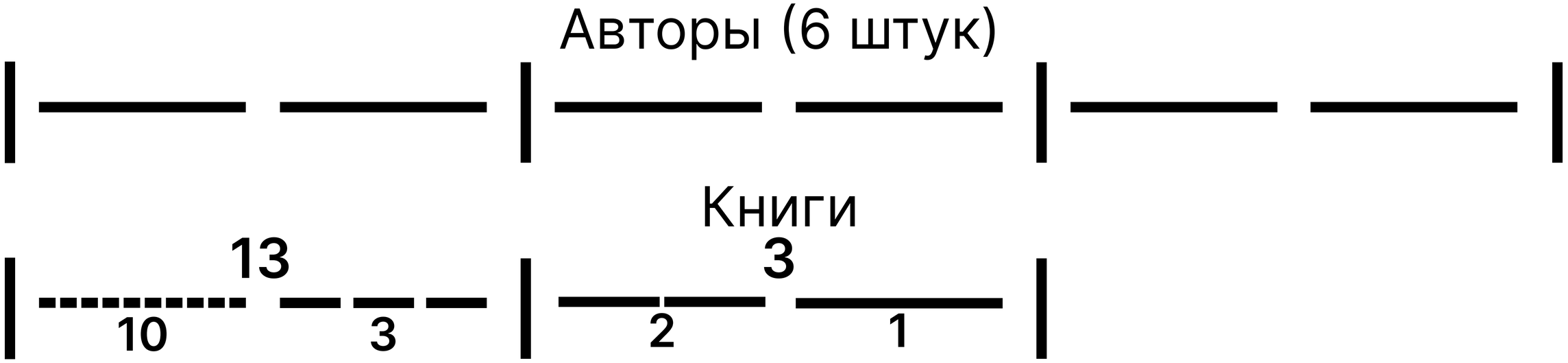
BatchSize = 2



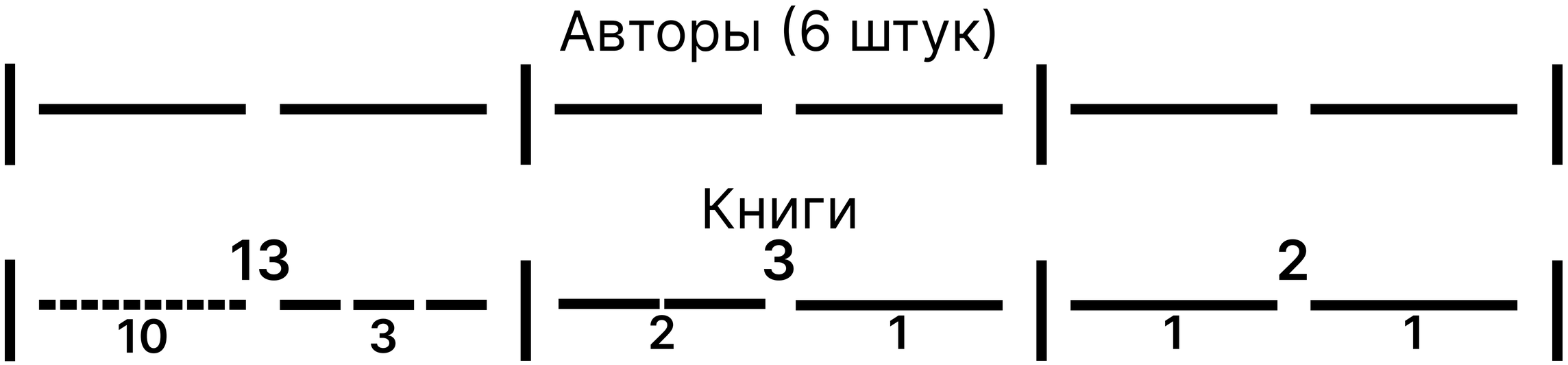
BatchSize = 2



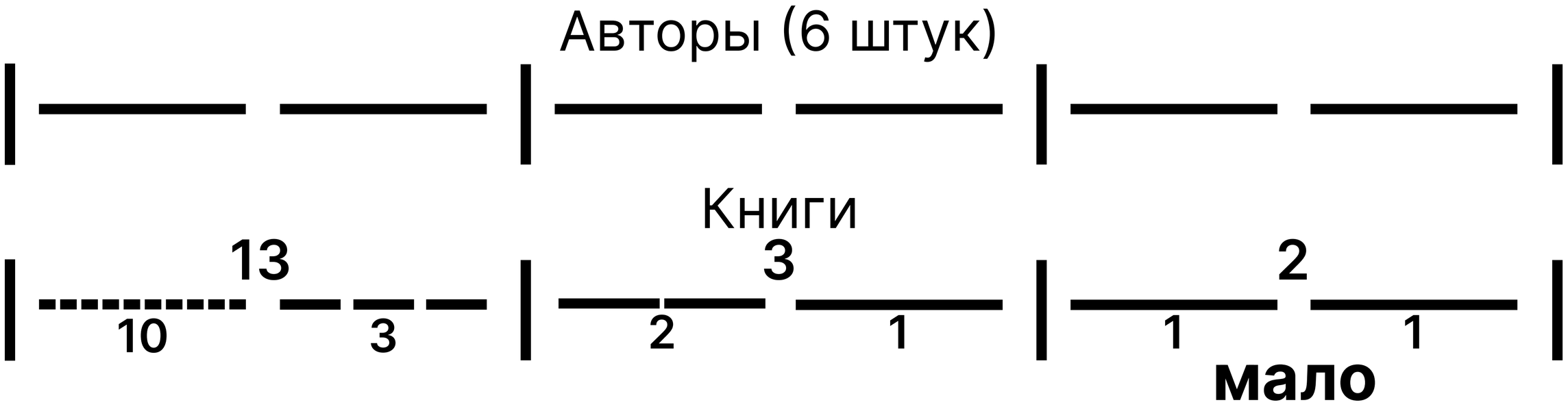
BatchSize = 2



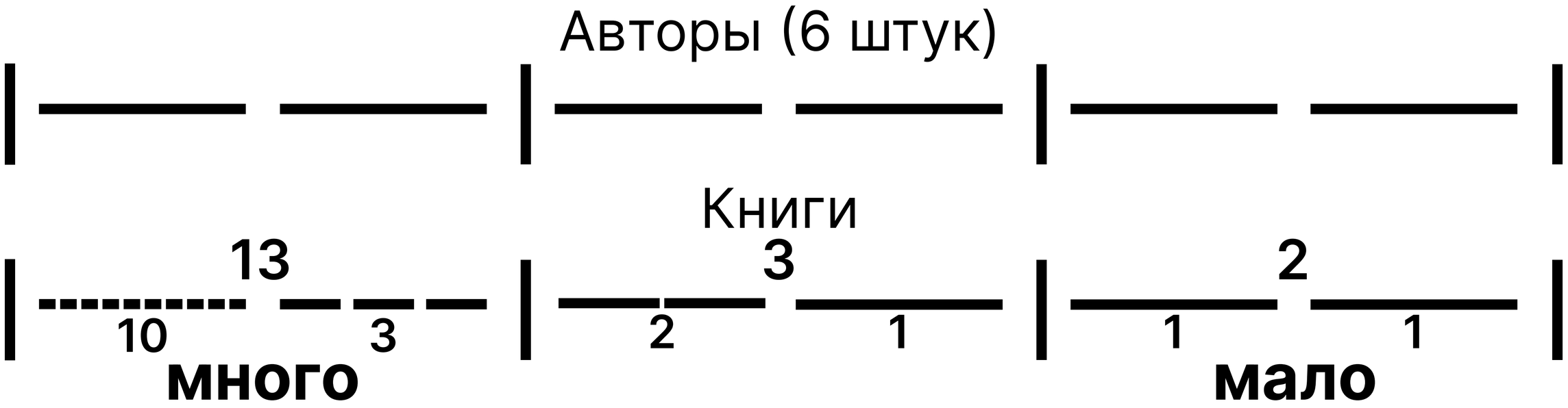
BatchSize = 2



BatchSize = 2

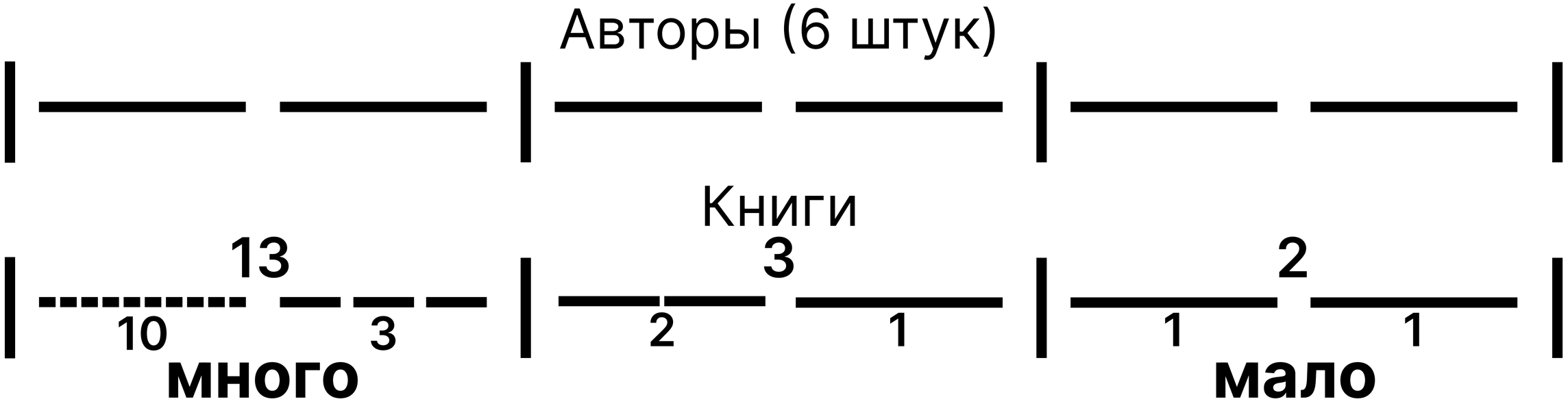


BatchSize = 2



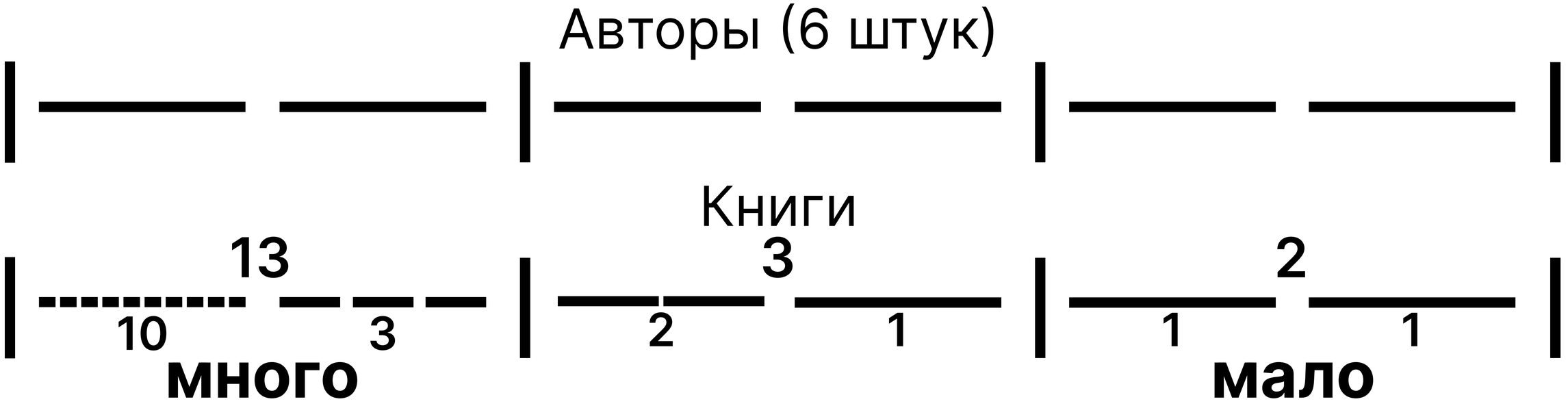
BatchSize = 2

Фатальный недостаток BatchSize



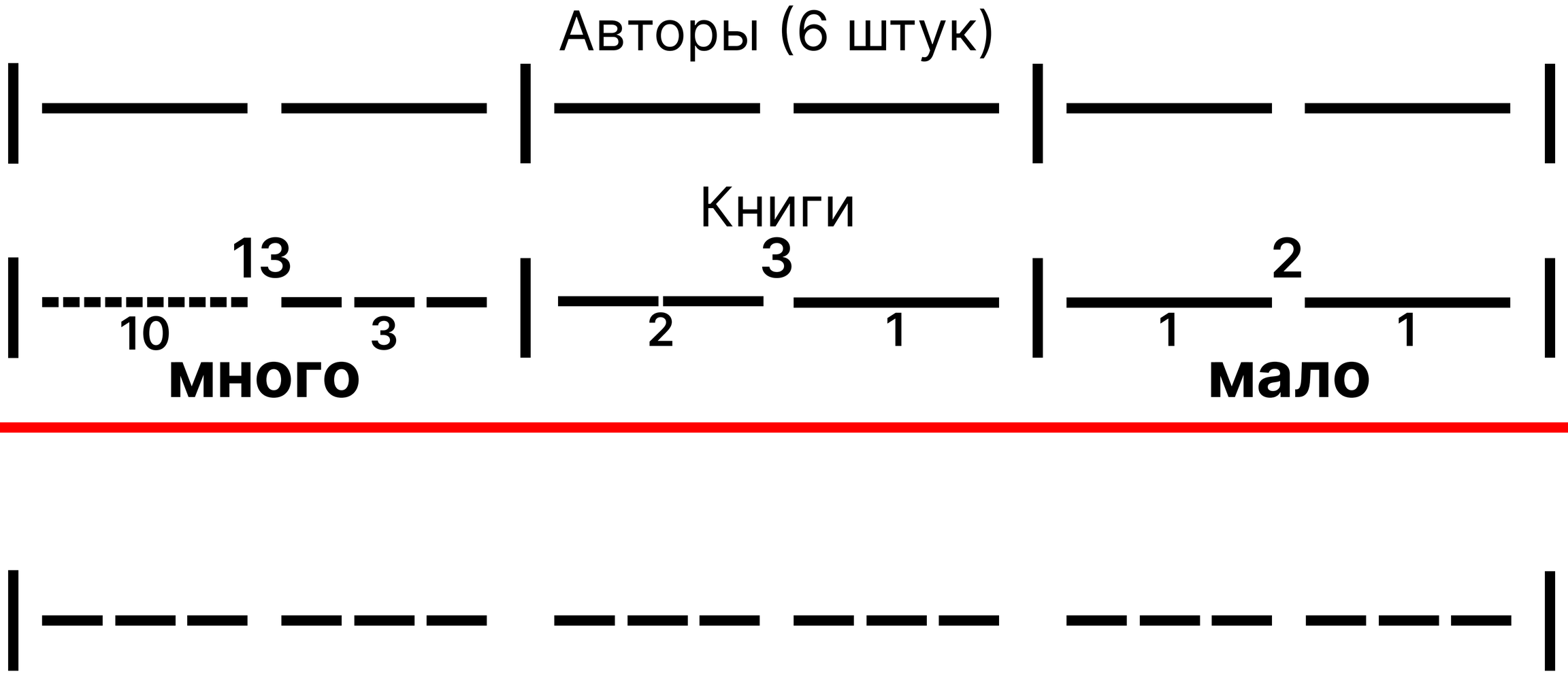
BatchSize = 2

Фатальный недостаток BatchSize



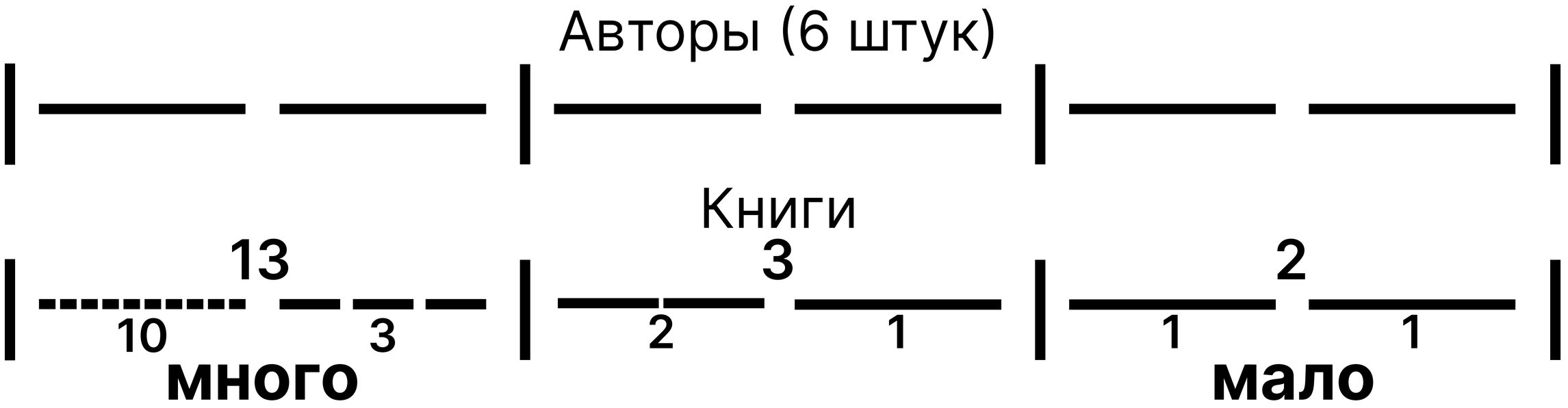
BatchSize = 2

Фатальный недостаток BatchSize

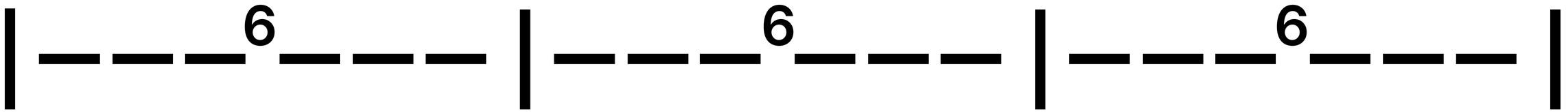


BatchSize = 2

Фатальный недостаток BatchSize



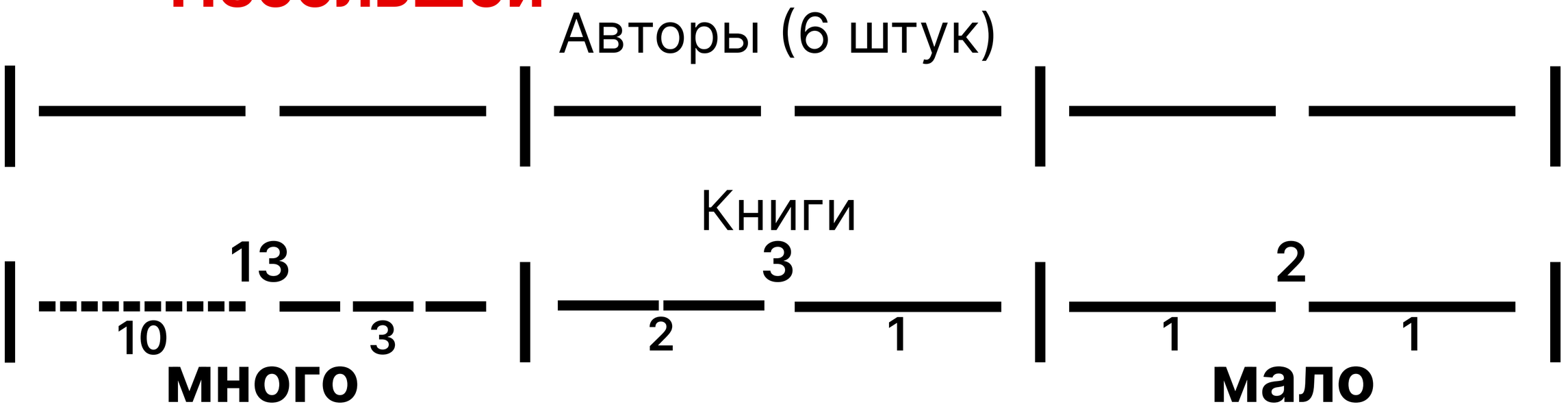
идеально



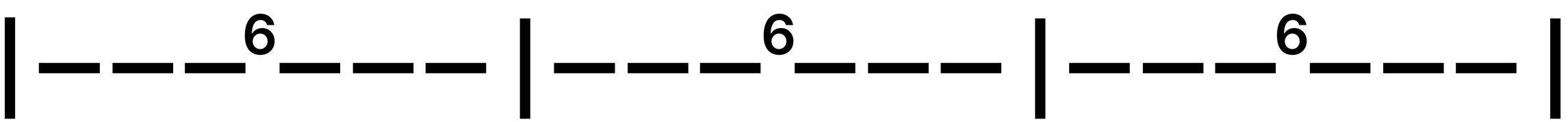
BatchSize = 2

~~Фатальный~~ недостаток BatchSize

Небольшой



идеально

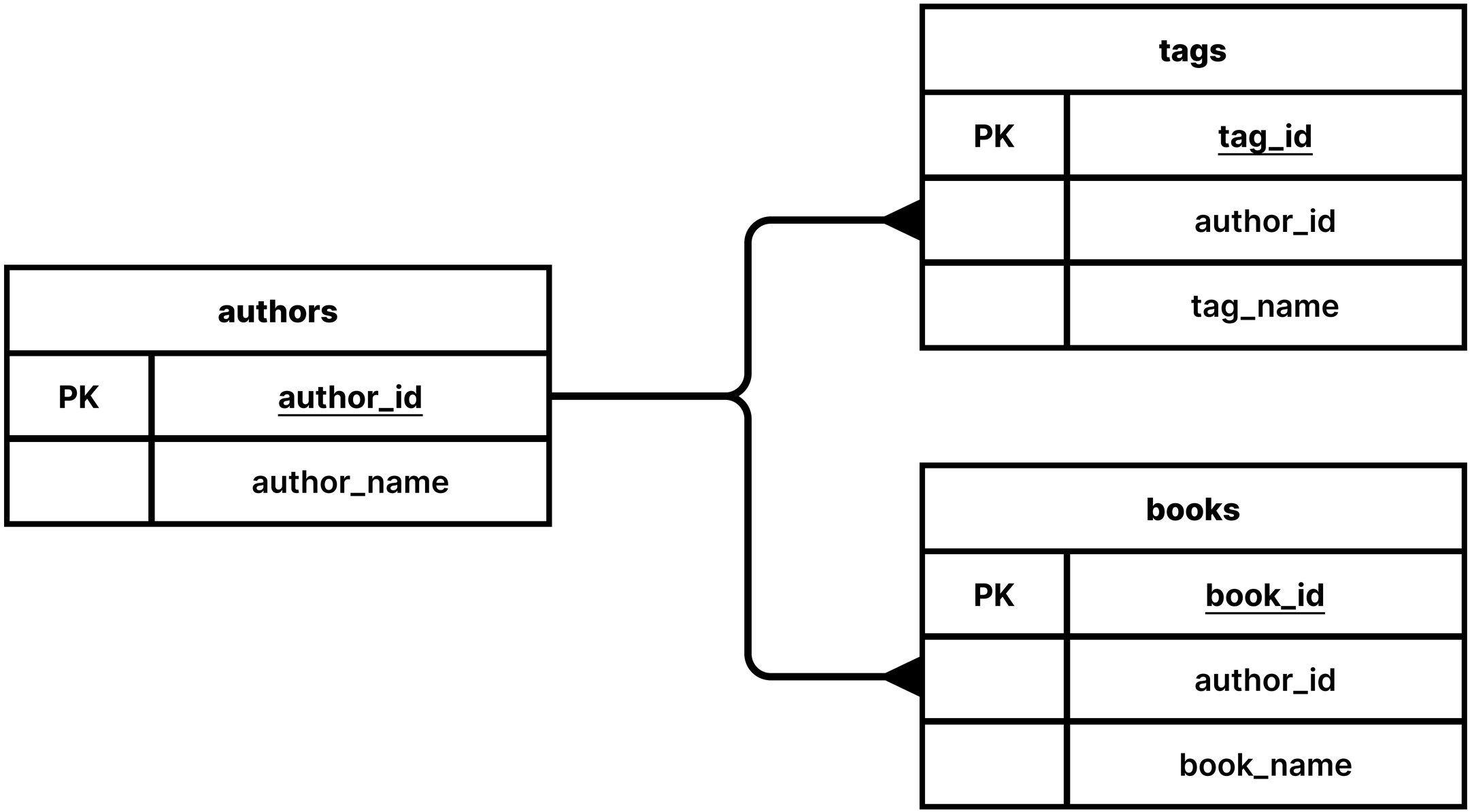


```
@Query("""
  select a.id
  from Author a
""")
List<String> findIds(
        Pageable page);
@Query("""
  select a
  from Author a
  left join fetch a.books
  where a.id in (:ids)
""")
List<String> findAuthorsByIds(
        List<String> ids);
```

```
select
 author0_.author_id as col_0_0_
from
 authors author0_ limit ?

select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  ...
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
where
  author0_.author_id in (
        ? , ?
 )
```

```
@Query("""
  select a.id
  from Author a
""")
List<String> findIds(
        Pageable page);
@Query("""
  select a
  from Author a
  left join fetch a.books
  where a.id in (:ids)
""")
List<String> findAuthorsByIds(
        List<String> ids);
```

```
select
 author0_.author_id as col_0_0_
from
 authors author0_ limit ?

select
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  ...
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
where
  author0_.author_id in (
    ? , ?
  )
```


tags
PK
tag_id
author_id
tag_name
authors
PK
author_id
author_name
books
PK
book_id
author_id
book_name

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Entity
@Table(name = "books")
public class Book {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "book_id")
    private String id;

    @Column(name = "book_name")
    private String name;

    @ManyToOne(
            cascade = CascadeType.ALL,
            fetch = FetchType.LAZY
    )
    @JoinColumn(name = "author_id")
    private Author author;

    //getters, setters...
}
```

```java
@Entity
@Table(name = "tags")
public class Tag {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "tag_id")
    private String id;

    @Column(name = "tag_name")
    private String name;

    @ManyToOne(
            cascade = CascadeType.ALL,
            fetch = FetchType.LAZY
    )
    @JoinColumn(name = "author_id")
    private Author author;

    //getters, setters...
}
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```

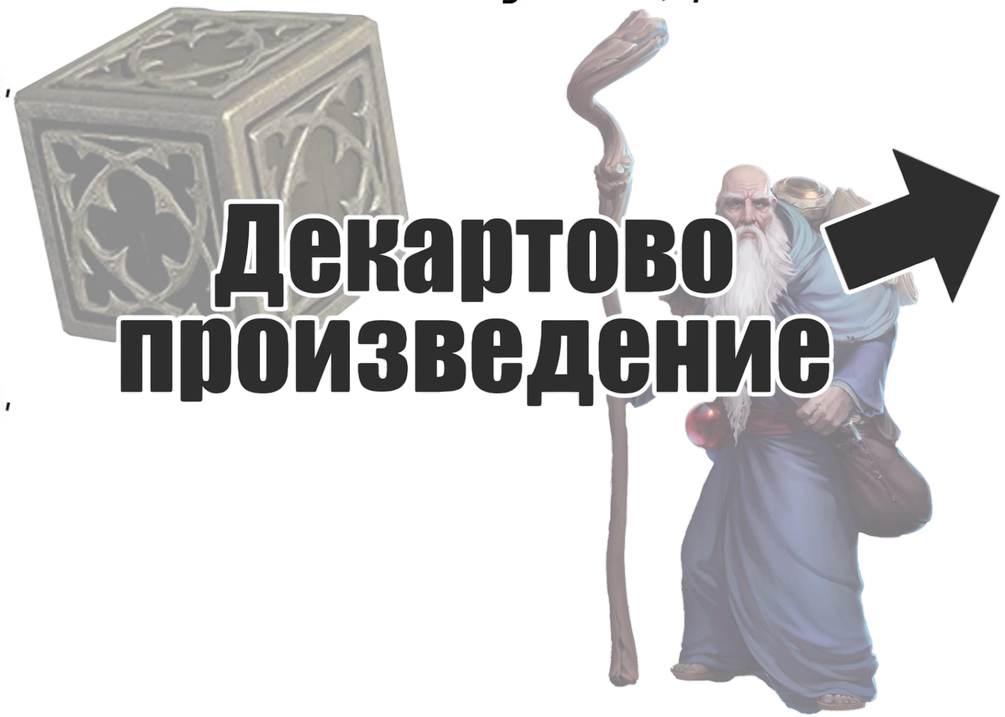
```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```

```
select distinct
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  tags2_.tag_id as tag_id1_2_2_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__,
  tags2_.author_id as author_i3_2_2_,
  tags2_.tag_name as tag_name2_2_2_,
  tags2_.author_id as author_i3_2_1__,
  tags2_.tag_id as tag_id1_2_1__
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
    left outer join
  tags tags2_
  on author0_.author_id=tags2_.author_id
where
  author0_.author_id in (
      ? , ?
  )
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

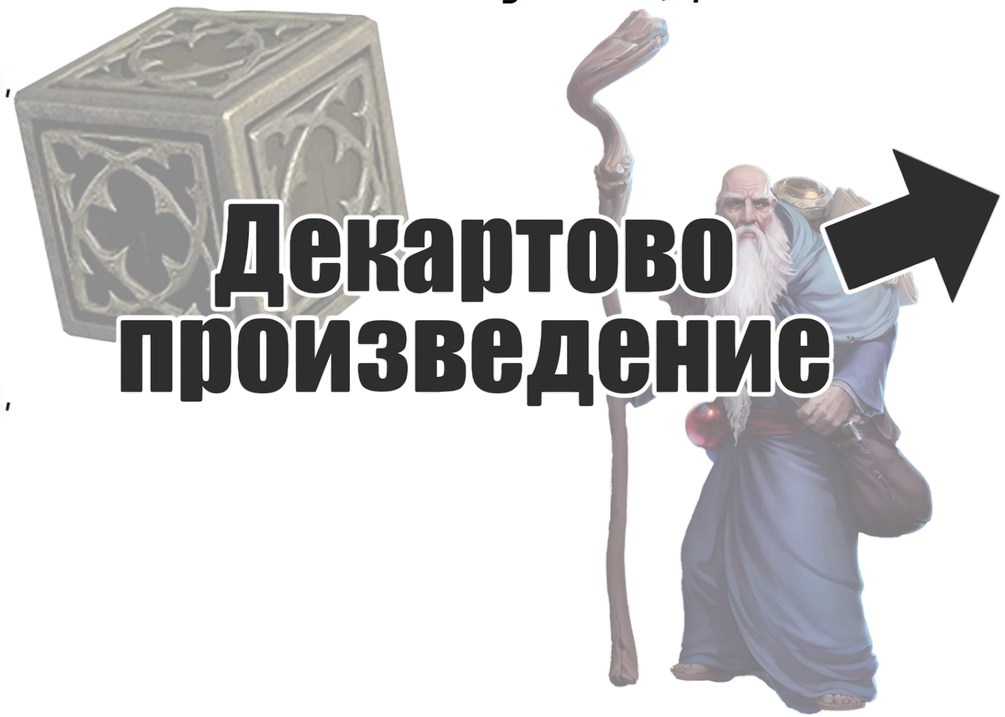
```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```




```sql
select distinct
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  tags2_.tag_id as tag_id1_2_2_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__,
  tags2_.author_id as author_i3_2_2_,
  tags2_.tag_name as tag_name2_2_2_,
  tags2_.author_id as author_i3_2_1__,
  tags2_.tag_id as tag_id1_2_1__
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
    left outer join
  tags tags2_
  on author0_.author_id=tags2_.author_id
where
  author0_.author_id in (
      ? , ?
  )
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;


    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

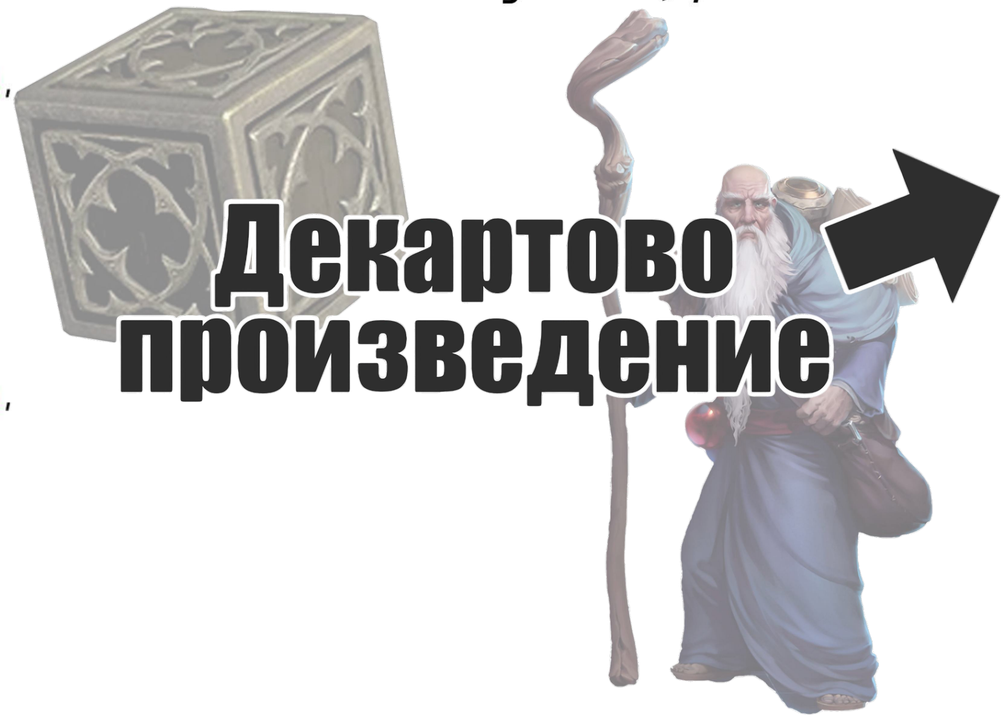
```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```




```sql
select distinct
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  tags2_.tag_id as tag_id1_2_2_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__,
  tags2_.author_id as author_i3_2_2_,
  tags2_.tag_name as tag_name2_2_2_,
  tags2_.author_id as author_i3_2_1__,
  tags2_.tag_id as tag_id1_2_1__
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
    left outer join
  tags tags2_
  on author0_.author_id=tags2_.author_id
where
  author0_.author_id in (
      ? , ?
  )
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```




```sql
select distinct
  author0_.author_id as author_i1_0_0_,
  books1_.book_id as book_id1_1_1_,
  tags2_.tag_id as tag_id1_2_2_,
  author0_.author_name as author_n2_0_0_,
  author0_.rating as rating3_0_0_,
  books1_.author_id as author_i3_1_1_,
  books1_.book_name as book_nam2_1_1_,
  books1_.author_id as author_i3_1_0__,
  books1_.book_id as book_id1_1_0__,
  tags2_.author_id as author_i3_2_2_,
  tags2_.tag_name as tag_name2_2_2_,
  tags2_.author_id as author_i3_2_1__,
  tags2_.tag_id as tag_id1_2_1__
from
  authors author0_
    left outer join
  books books1_
  on author0_.author_id=books1_.author_id
    left outer join
  tags tags2_
  on author0_.author_id=tags2_.author_id
where
  author0_.author_id in (
      ? , ?
  )
```

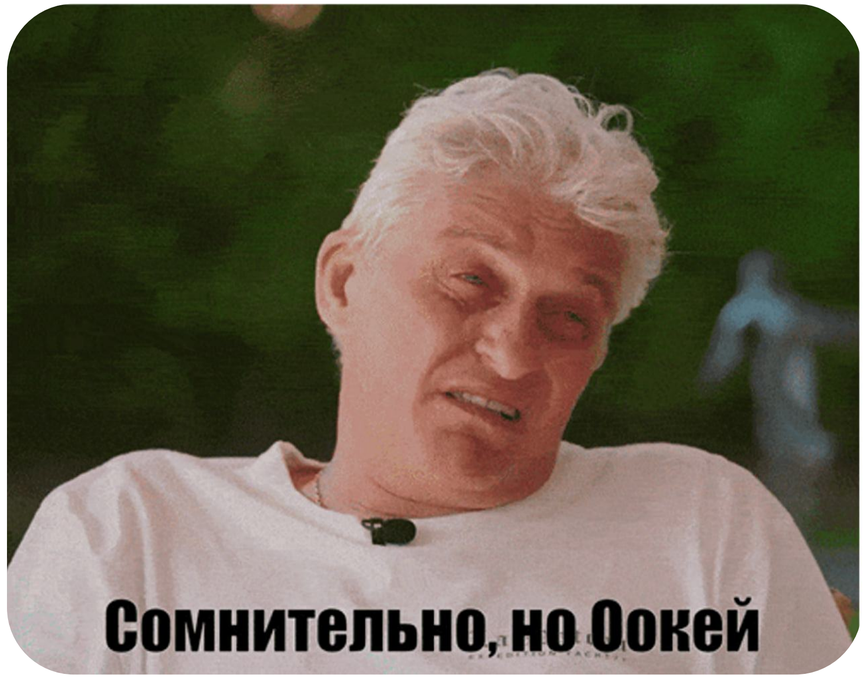
```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```

```
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_
where
    author0_.author_id in (
                          ? , ?
        )
    Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
                         ?,?
        )
```

```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Tag> tags;

    //getters, setters...
}
```

```java
@Query("""
select distinct a
from Author a
left join fetch a.books
left join fetch a.tags
where a.id in (:ids)
""")
List<Author> findAuthorsByIds(
    List<String> ids);
```




```sql
select
    author0_.author_id as author_i1_0_,
    author0_.author_name as author_n2_0_
from
    authors author0_
where
    author0_.author_id in (
                          ? , ?
        )
    Hibernate:
select
    books0_.author_id as author_i3_1_1_,
    books0_.book_id as book_id1_1_1_,
    books0_.book_id as book_id1_1_0_,
    books0_.author_id as author_i3_1_0_,
    books0_.book_name as book_nam2_1_0_
from
    books books0_
where
    books0_.author_id in (
                         ?,?
        )
```

Hibernate
проблема декартова произведения при запросах с пагинацией

Артём Гордиенко

```sql
select
  a.author_id,
  a.author_name,
  b.book_id,
  b.book_name
from
  authors a
    left join books b
      on b.author_id = a.author_id
where
  a.author_id in (
    select a1.author_id
    from authors a1
    where a1.rating = :rating
    order by a1.author_id
    limit :limit
    offset :offset
  )
order by
    a.author_id,
    b.book_id
```

```java
List<Author> authors = authorRepository
  .findAuthorsSqlSubquery(
    Pageable.ofSize(10).withPage(1), 5);

for (Author author : authors) {
  Collection<Book> books = author.getBooks();
  System.out.println(books.size());
}
```

```sql
select
  a.author_id,
  a.author_name,
  b.book_id,
  b.book_name
from
  authors a
    left join books b
      on b.author_id = a.author_id
where
  a.author_id in (
    select a1.author_id
    from authors a1
    where a1.rating = :rating
    order by a1.author_id
    limit :limit
    offset :offset
  )
order by
    a.author_id,
    b.book_id
```

```
@Fetch(FetchMode.SUBSELECT)
@OneToMany(
        cascade = CascadeType.ALL,
        mappedBy = "author",
        fetch = FetchType.LAZY
)
@ToString.Exclude
Set<Book> books;
```

```
List<Author> authors = authorRepository
  .findAuthorsSqlSubquery(
    Pageable.ofSize(10).withPage(1), 5);

for (Author author : authors) {
  Collection<Book> books = author.getBooks();
  System.out.println(books.size());
}
```

```java
@Fetch(FetchMode.SUBSELECT)
@OneToMany(
        cascade = CascadeType.ALL,
        mappedBy = "author",
        fetch = FetchType.LAZY
)
@ToString.Exclude
Set<Book> books;


List<Author> authors = authorRepository
  .findAuthorsSqlSubquery(
    Pageable.ofSize(10).withPage(1), 5);

for (Author author : authors) {
  Collection<Book> books = author.getBooks();
  System.out.println(books.size());
}
```

```sql
select
  a1_0.*
from authors a1_0
where a1_0.rating=?
offset ?
limit ?
```

```java
@Fetch(FetchMode.SUBSELECT)
@OneToMany(
        cascade = CascadeType.ALL,
        mappedBy = "author",
        fetch = FetchType.LAZY
)
@ToString.Exclude
Set<Book> books;


List<Author> authors = authorRepository
  .findAuthorsSqlSubquery(
    Pageable.ofSize(10).withPage(1), 5);

for (Author author : authors) {
  Collection<Book> books = author.getBooks();
  System.out.println(books.size());
}
```

```sql
select
  a1_0.*
from authors a1_0
where a1_0.rating=?
offset ?
limit ?

select
  b1_0.*
from
  books b1_0
where
  b1_0.author_id in (
    select a1_0.author_id
    from authors a1_0
    where a1_0.rating=?
)
```

```
@Query(value = """
select a
from Author a
  join fetch a.books
where a.id in (
  select a1.id
  from Author a1
  where
    a1.rating >= :rating
  order by a1.id
  limit :limit
  offset :offset
)""")
List<Author> findByRatingHql(
  int limit, int offset,
  int rating);
```

```
@Query(value = """
select a
from Author a
  join fetch a.books
where a.id in (
  select a1.id
  from Author a1
  where
    a1.rating >= :rating
  order by a1.id
  limit :limit
  offset :offset
)""")
List<Author> findByRatingHql(
  int limit, int offset,
  int rating);
```


```
@Query(value = """
select a
from Author a
  join fetch a.books
where a.id in (
  select a1.id
  from Author a1
  where
    a1.rating >= :rating
  order by a1.id
  limit :limit
  offset :offset
)""")
List<Author> findByRatingHql(
  int limit, int offset,
  int rating);
```


```
@Query(value = """
select a
from Author a
  join fetch a.books
where a.id in (
  select a1.id
  from Author a1
  where
    a1.rating >= :rating
  order by a1.id
  limit :limit
  offset :offset
)""")
List<Author> findByRatingHql(
  int limit, int offset,
  int rating);
```

```
select
  a1_0.author_id,
  b1_0.author_id,
  ...
from
  authors a1_0 join books b1_0
  on a1_0.author_id=b1_0.author_id
where
  a1_0.author_id in (
  select
    a2_0.author_id
  from
    authors a2_0
  where
    a2_0.rating>=?
  order by
    a2_0.author_id
  offset ? rows
  fetch first ? rows only
  )
```

```
@Query(value = """
select a
from Author a
  join fetch a.books
where a.id in (
  select a1.id
  from Author a1
  where
    a1.rating >= :rating
  order by a1.id
  limit :limit
  offset :offset
)""")
List<Author> findByRatingHql(
  int limit, int offset,
  int rating);
```

HIBERNATE 6

```
select
  a1_0.author_id,
  b1_0.author_id,
  ...
from
  authors a1_0 join books b1_0
  on a1_0.author_id=b1_0.author_id
where
  a1_0.author_id in (
  select
    a2_0.author_id
  from
    authors a2_0
  where
    a2_0.rating>=?
  order by
    a2_0.author_id
  offset ? rows
  fetch first ? rows only
  )
```


```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```


```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
select
  a1_0.author_id,
  a1_0.author_name,
  a1_0.rating
from
  authors a1_0
where
  a1_0.rating>?
offset ? rows
fetch
first ? rows only
```


```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
select
  b1_0.author_id,
  b1_0.book_id,
  b1_0.book_name
from
  books b1_0
where
  b1_0.author_id = any (?)
```

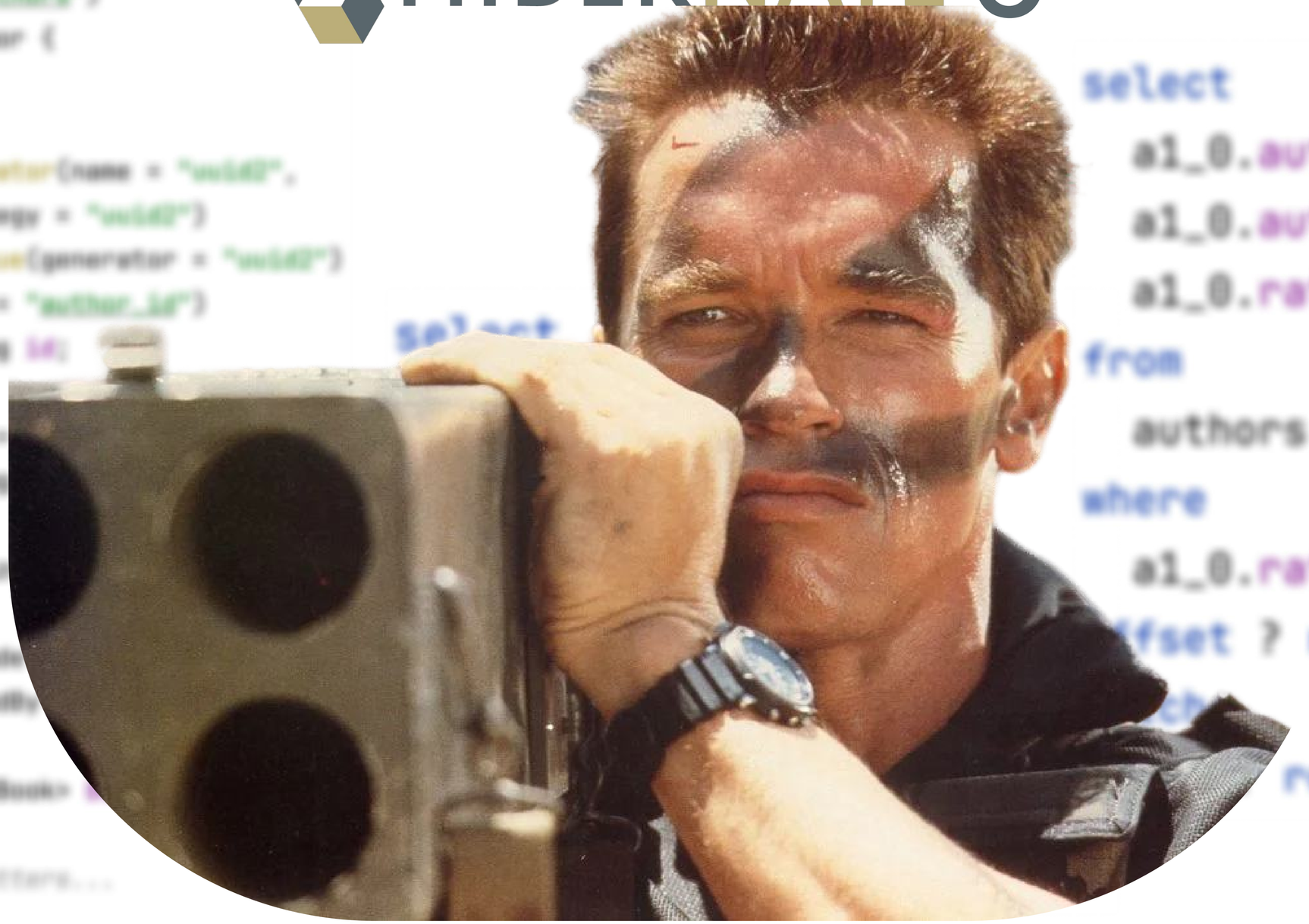
```
select
  a1_0.author_id,
  a1_0.author_name,
  a1_0.rating
from
  authors a1_0
where
  a1_0.rating>?
offset ? rows
fetch
first ? rows only
```


```java
@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GenericGenerator(name = "uuid2",
            strategy = "uuid2")
    @GeneratedValue(generator = "uuid2")
    @Column(name = "author_id")
    private String id;

    @Column(name = "author_name")
    private String name;

    @BatchSize(size = 2)
    @OneToMany(
            cascade = CascadeType.ALL,
            mappedBy = "author"
    )
    private List<Book> books;

    //getters, setters...
}
```

```
select
  b1_0.author_id,
  b1_0.book_id,
  b1_0.book_name
from
  books b1_0
where
  b1_0.author_id = any (?)
```

```
select
  a1_0.author_id,
  a1_0.author_name,
  a1_0.rating
from
  authors a1_0
where
  a1_0.rating>?
offset ? rows
fetch
first ? rows only
```

HIBERNATE 6



Артём Гордиенко

спикер

Артём
Гордиенко
спикер

[HHH-16427] Performance improvement for the @BatchSize

Артём Гордиенко
спикер

Алексей Стукалов
пк

Владимир Ситников
пк

[HHH-16427] Performance improvement for the @BatchSize

Артём Гордиенко
спикер

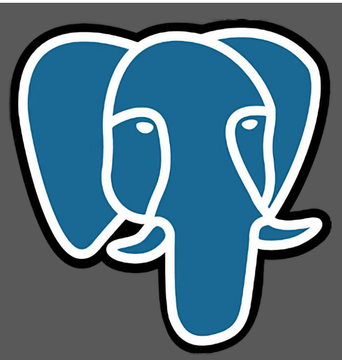
Алексей Стукалов
пк

Владимир Ситников
пк

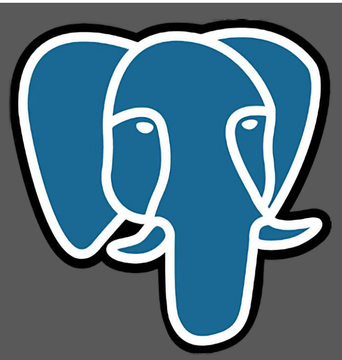
[HHH-16427] Performance improvement for the @BatchSize

[HHH-16466] ARRAY parameter support for multi-key loads

Application

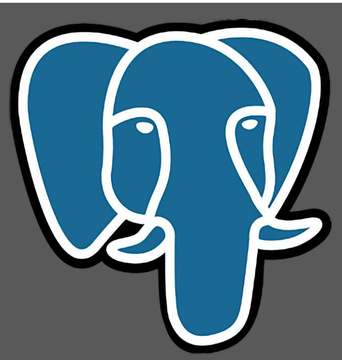


Application



SQL(jdbc)

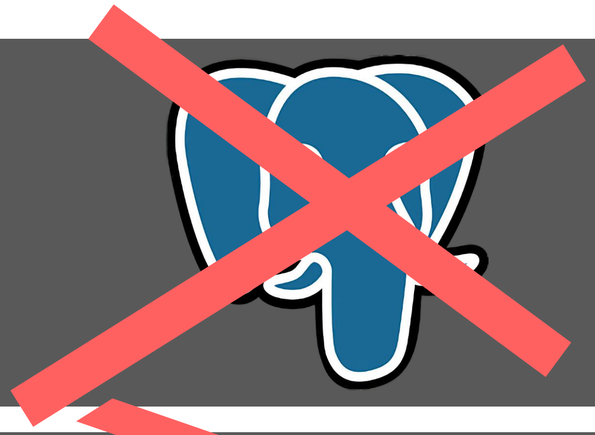
Application



SQL(jdbc)

HIBERNATE

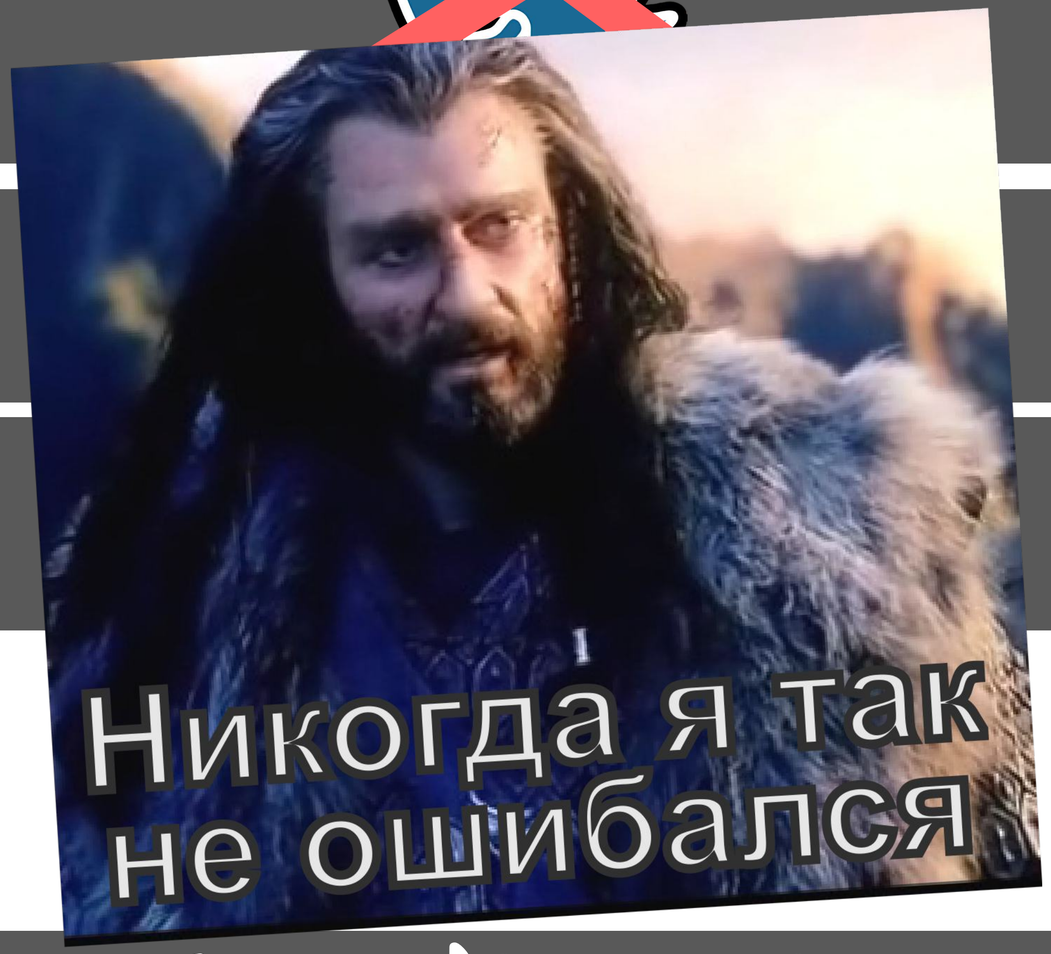
Application



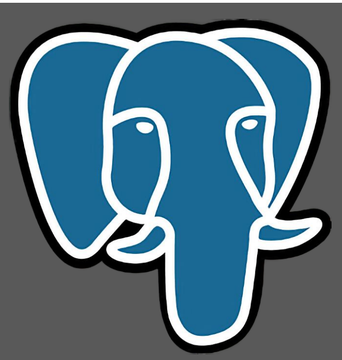
~~SQL (jdbc)~~

HIBERNATE

Application



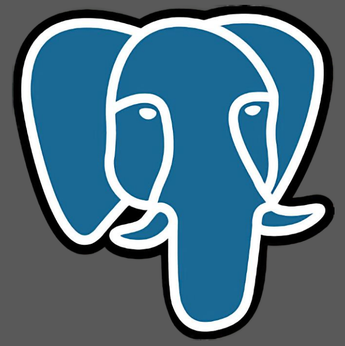
Application



SQL(jdbc)

HIBERNATE

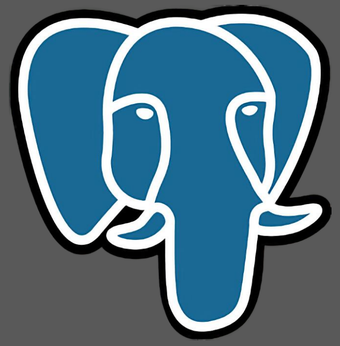
Application



SQL(jdbc)

HIBERNATE

Application



SQL(jdbc)

HIBERNATE

Облегчает жизнь,
а не абстрагирует
от неё

Application

Java Object Oriented Querying
https://www.jooq.org

jOOQ: The easiest way to write SQL in Java

jOOQ generates Java code from your database and lets you build type safe SQL queries through its fluent API. · Great Reasons for Using **jOOQ** · Database First.

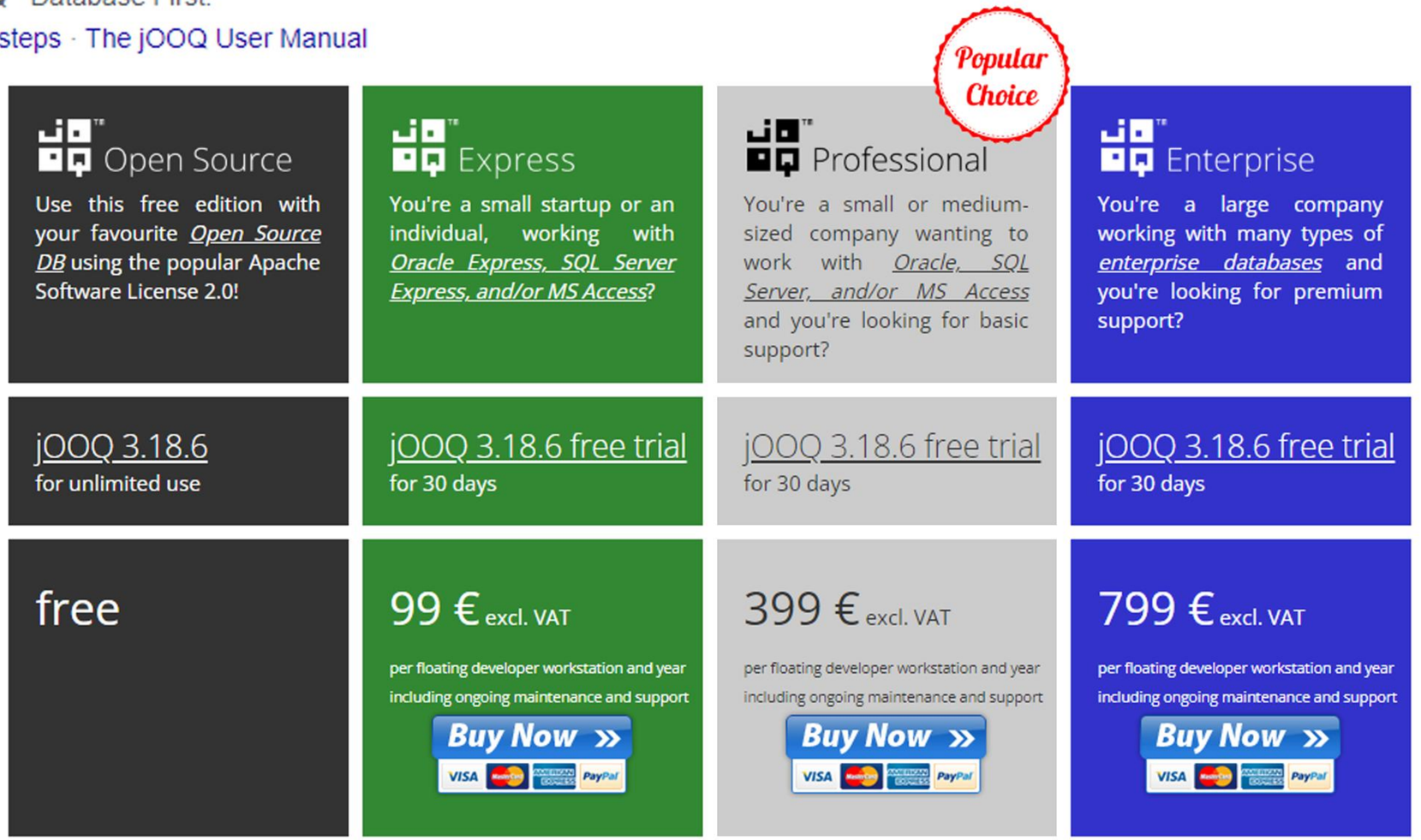
Download / Pricing · Documentation · jOOQ in 7 easy steps · The jOOQ User Manual

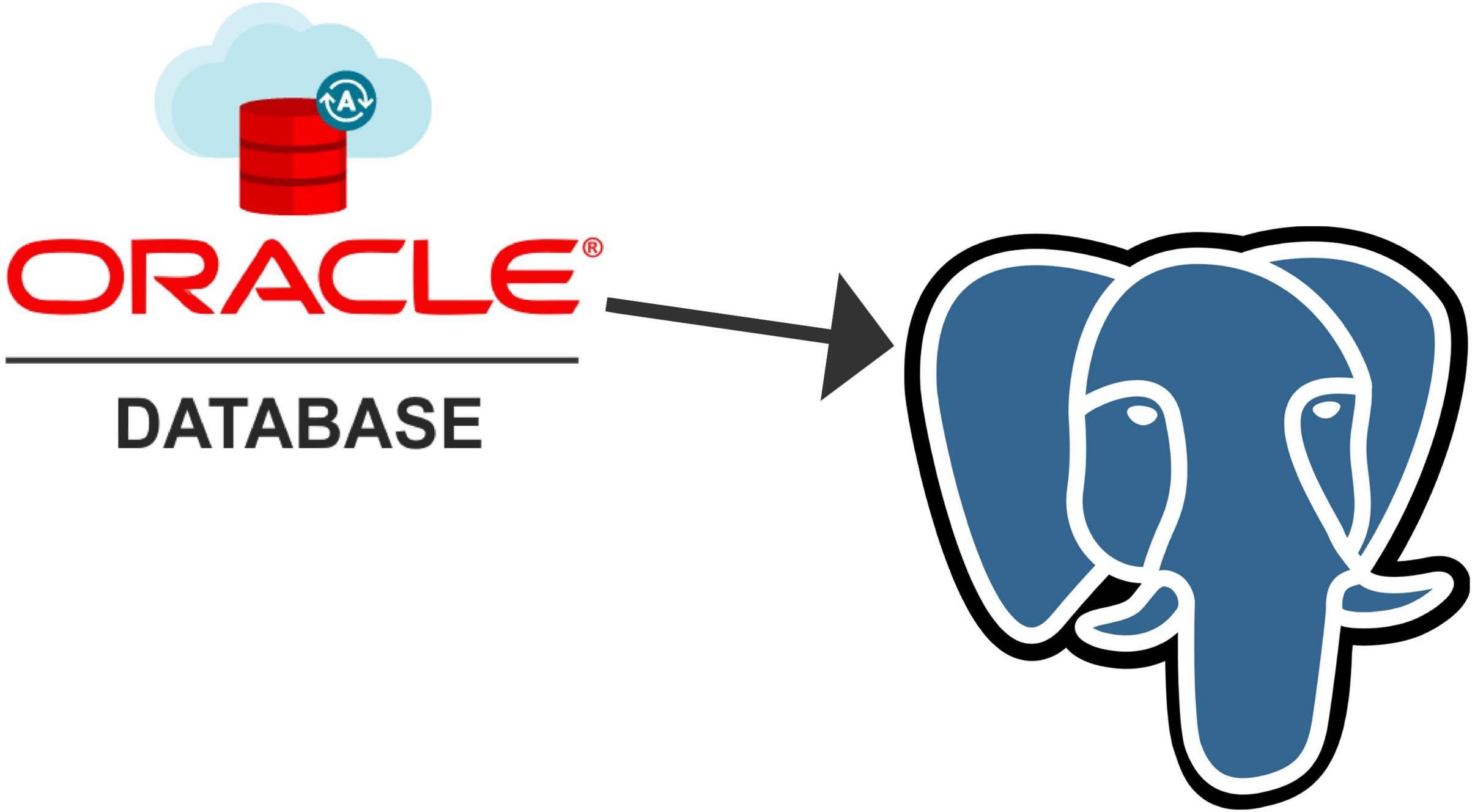
Java Object Oriented Querying
https://www.jooq.org

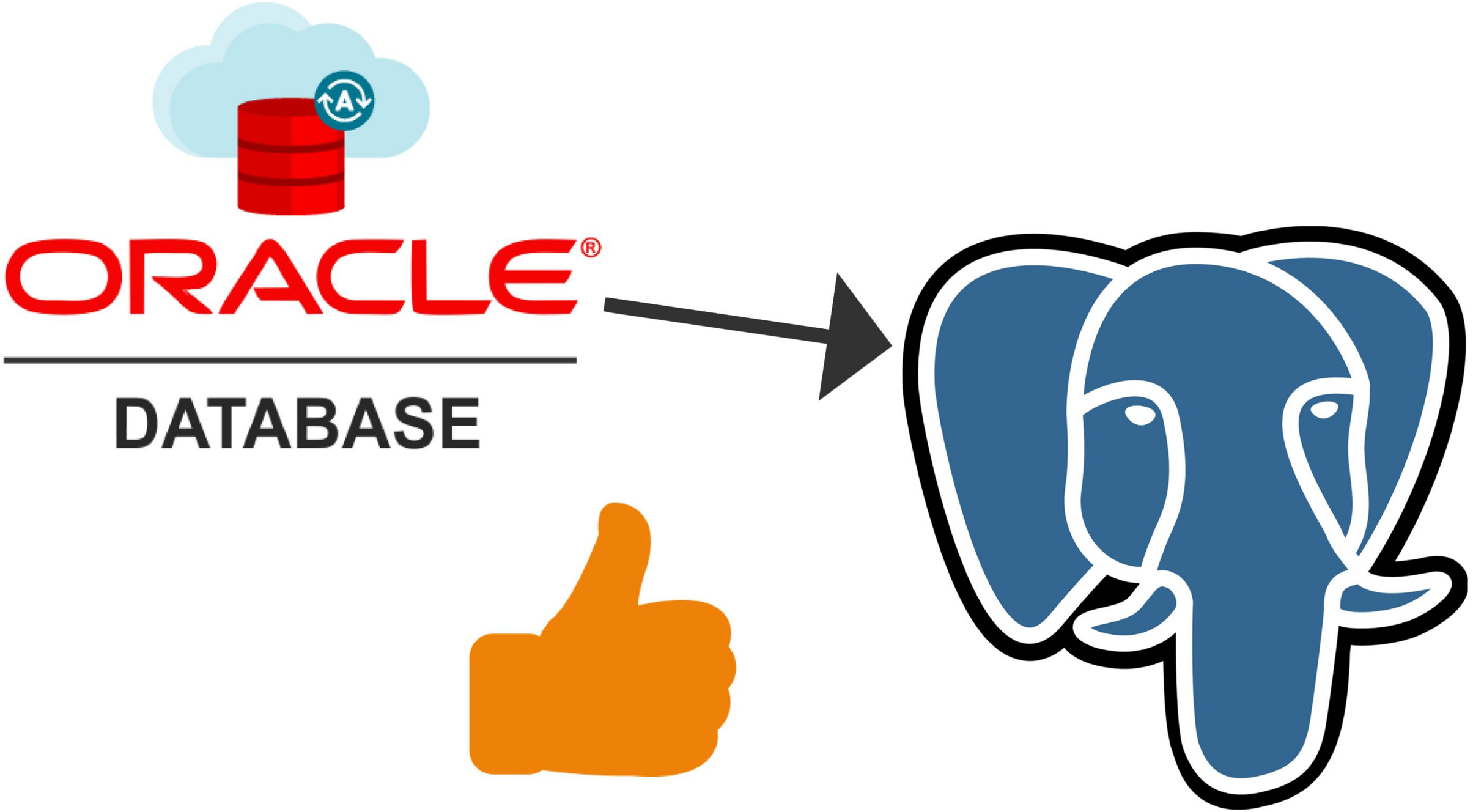
jOOQ: The easiest way to write SQL in Java

jOOQ generates Java code from your database and lets you build type safe SQL queries through its fluent API. · Great Reasons for Using **jOOQ** · Database First.

Download / Pricing · Documentation · jOOQ in 7 easy steps · The jOOQ User Manual




ORACLE®
DATABASE


ORACLE®
DATABASE


HIBERNATE
jOOQ™

id	author id	title	rating
11	1	Властелин колец	96
23	1	Сильмариллион	32
37	1	Хоббит	14
41	2	Гарри Поттер	46
53	2	Фантастические звери	28
57	3	Солярис	55
73	3	Фиаско	34
77	5	Игра престолов	95
89	7	Чистая архитектура	14
89	7	Чистый кот	9
89	7	Чистый кодер	19


HIBERNATE
ENVERS

rev_id	id	author id	title	rating	last updated
6	11	1	Властелин колец	96	3 марта
7	11	1	Властелин колец	97	5 марта
9	11	1	Властелин колец	98	18 марта
3	41	2	Гарри Поттер	46	27 февраля
5	41	2	Гарри Поттер	45	30 февраля
2	57	3	Солярис	55	24 января
11	57	3	Солярис	54	8 апреля
10	77	5	Игра престолов	95	24 марта
1	89	7	Чистая архитектура	14	19 января
4	89	7	Чистая архитектура	15	28 февраля
8	89	7	Чистая архитектура	16	5 марта

```
repository.deleteAll();
```

```
repository.deleteAll();
```

```
select bh.*
from book_hist bh
```

1

```
delete
from book_hist bh
where bh.rev_id = ?
```

N

```
repository.deleteAll();
```

```
select bh.*
from book_hist bh
```

1

```
delete
from book_hist bh
where bh.rev_id = ?
```

N

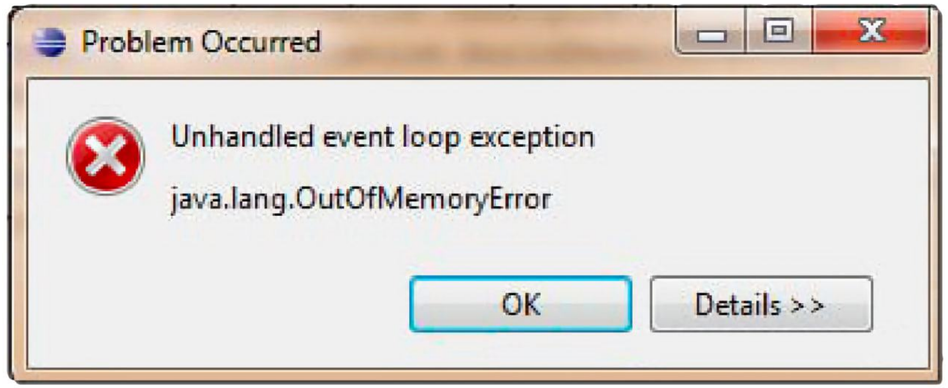
```
repository.deleteAll();
```

```
select bh.*
from book_hist bh
```

1

```
delete
from book_hist bh
where bh.rev_id = ?
```

N



```
void deleteAllInBatch();
```

```
@Query("delete from BookHist")
@Modifying
@Transactional
void deleteAllRows();
```

```
void deleteAllInBatch();
```

```
@Query("delete from BookHist")
@Modifying
@Transactional
void deleteAllRows();
```

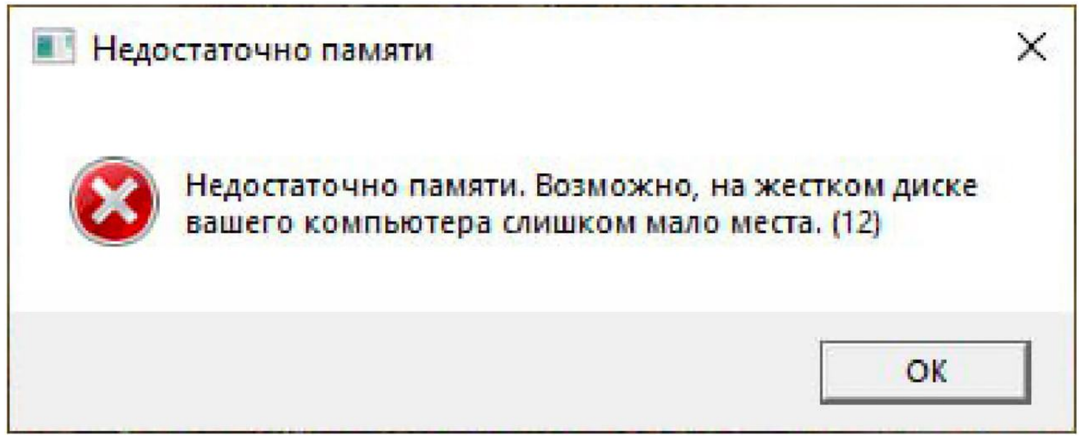
```
void deleteAllInBatch();
```

```
delete
from
book_hist
```

```
@Query("delete from BookHist")
@Modifying
@Transactional
void deleteAllRows();
```

```
void deleteAllInBatch();
```

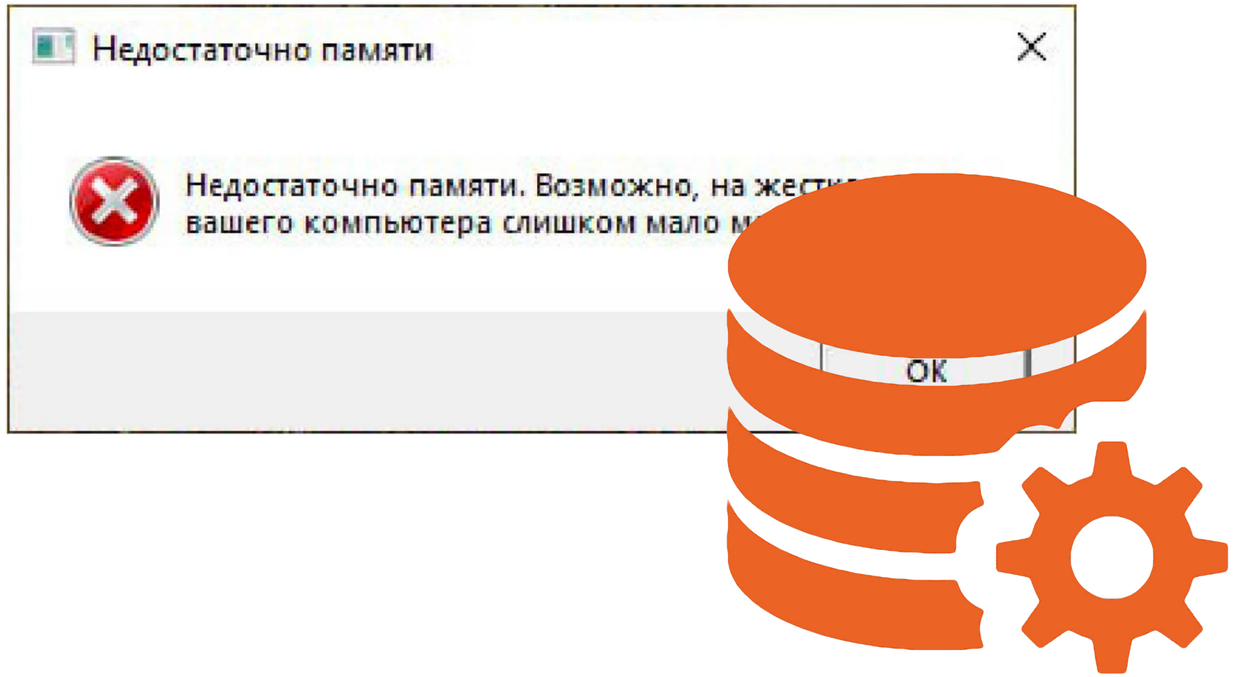
```
delete
from
book_hist
```



```java
@Query("delete from BookHist")
@Modifying
@Transactional
void deleteAllRows();
```

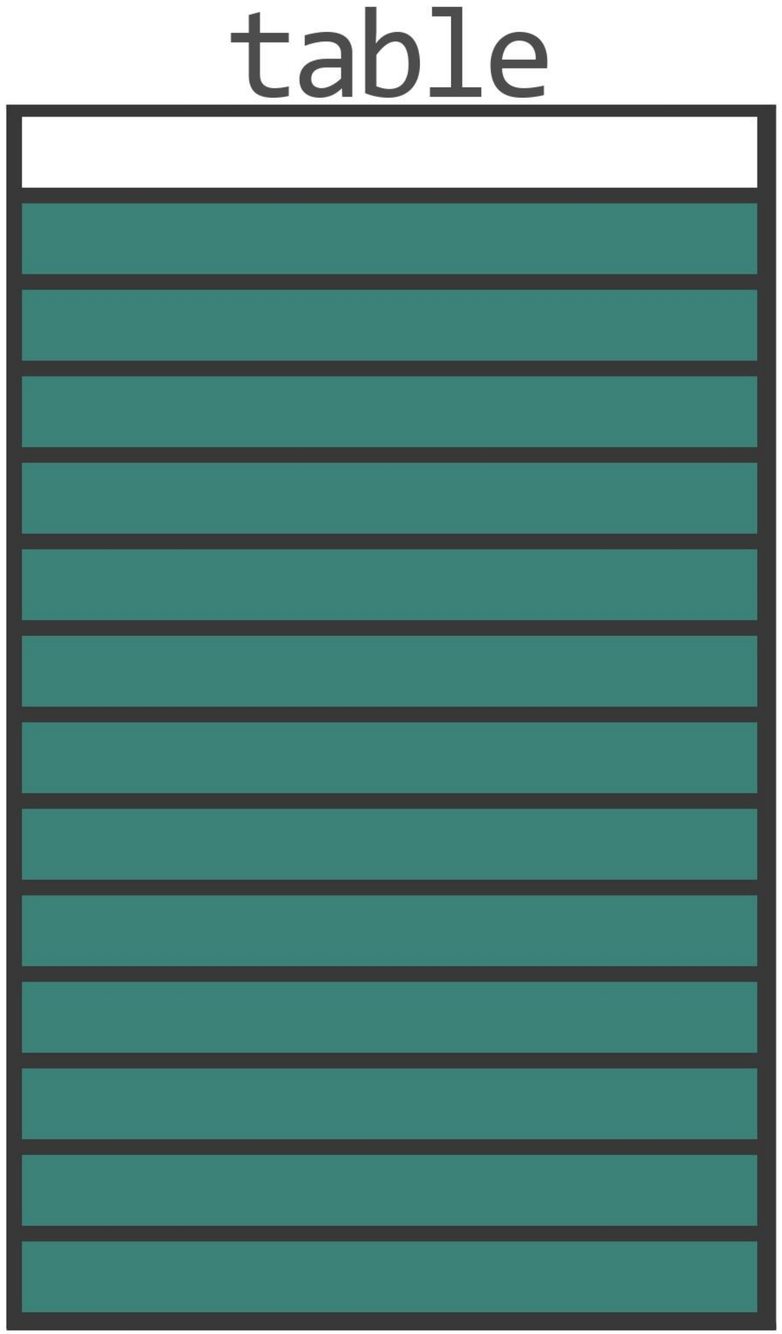
```java
void deleteAllInBatch();
```

```
delete
from
book_hist
```

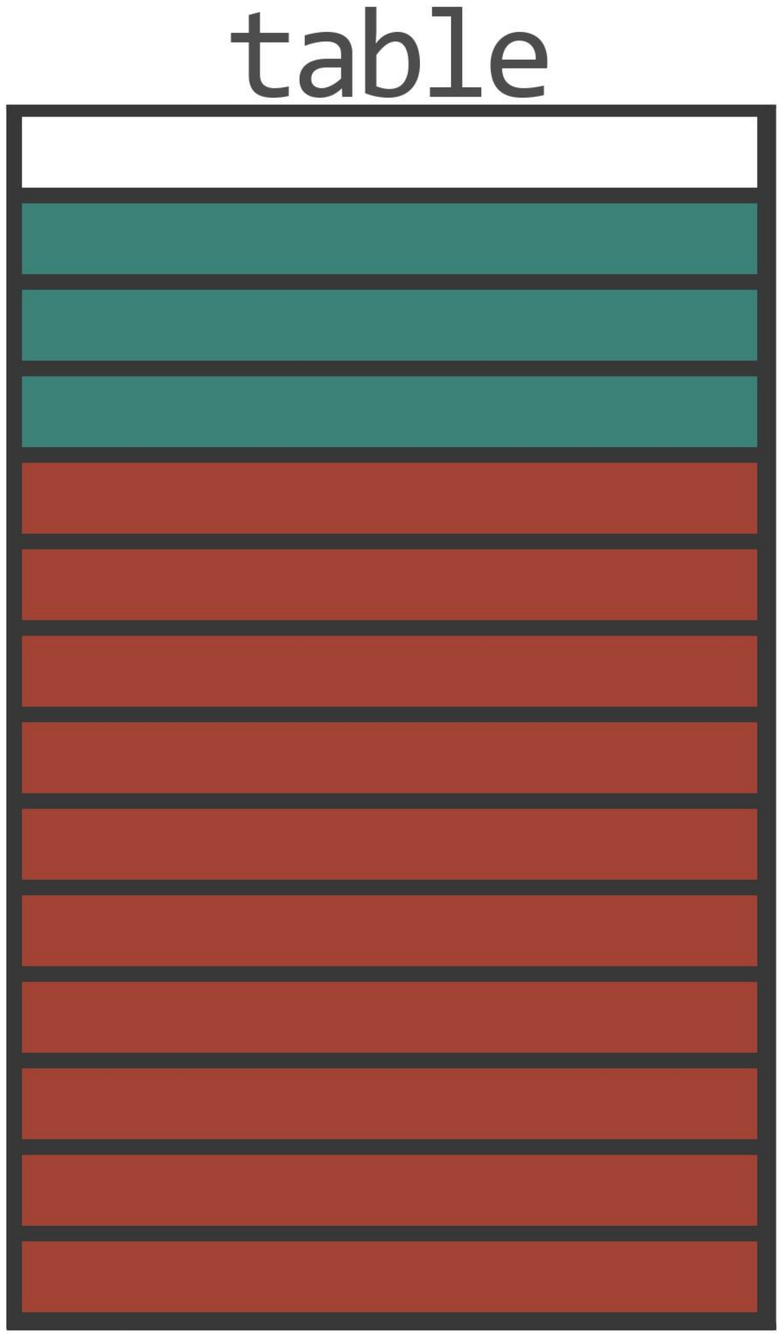


```
@Query("delete from BookHist")
@Modifying
@Transaction
                                    Batch();
void deleteA
```

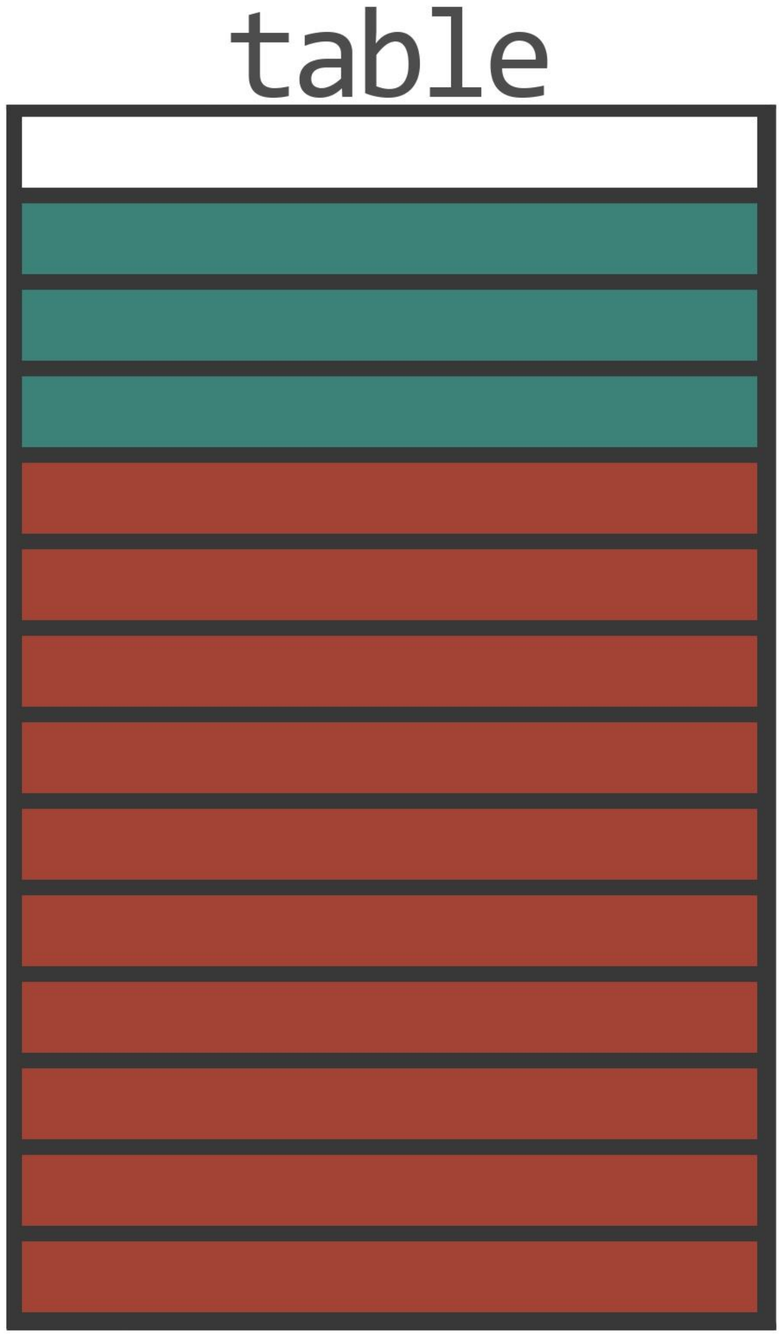
```
delete
from
book_h
```



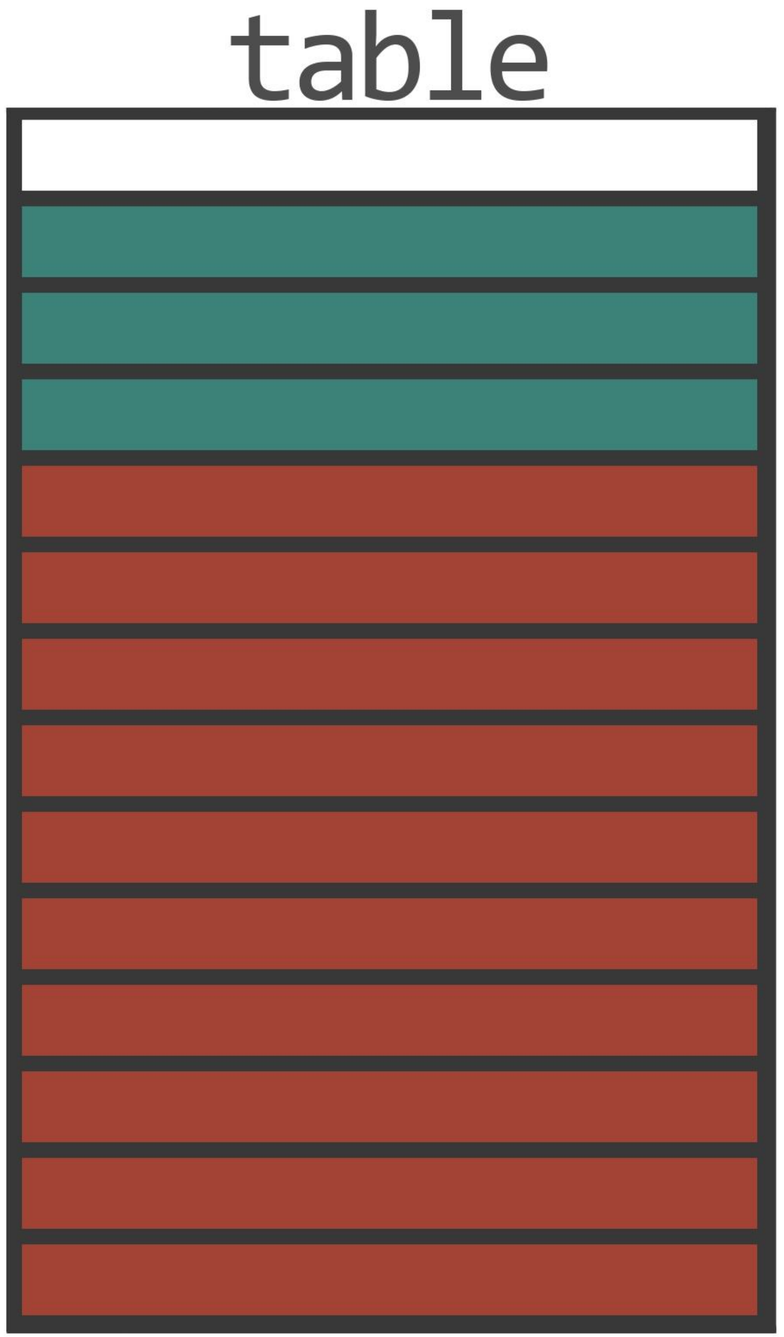
table



table

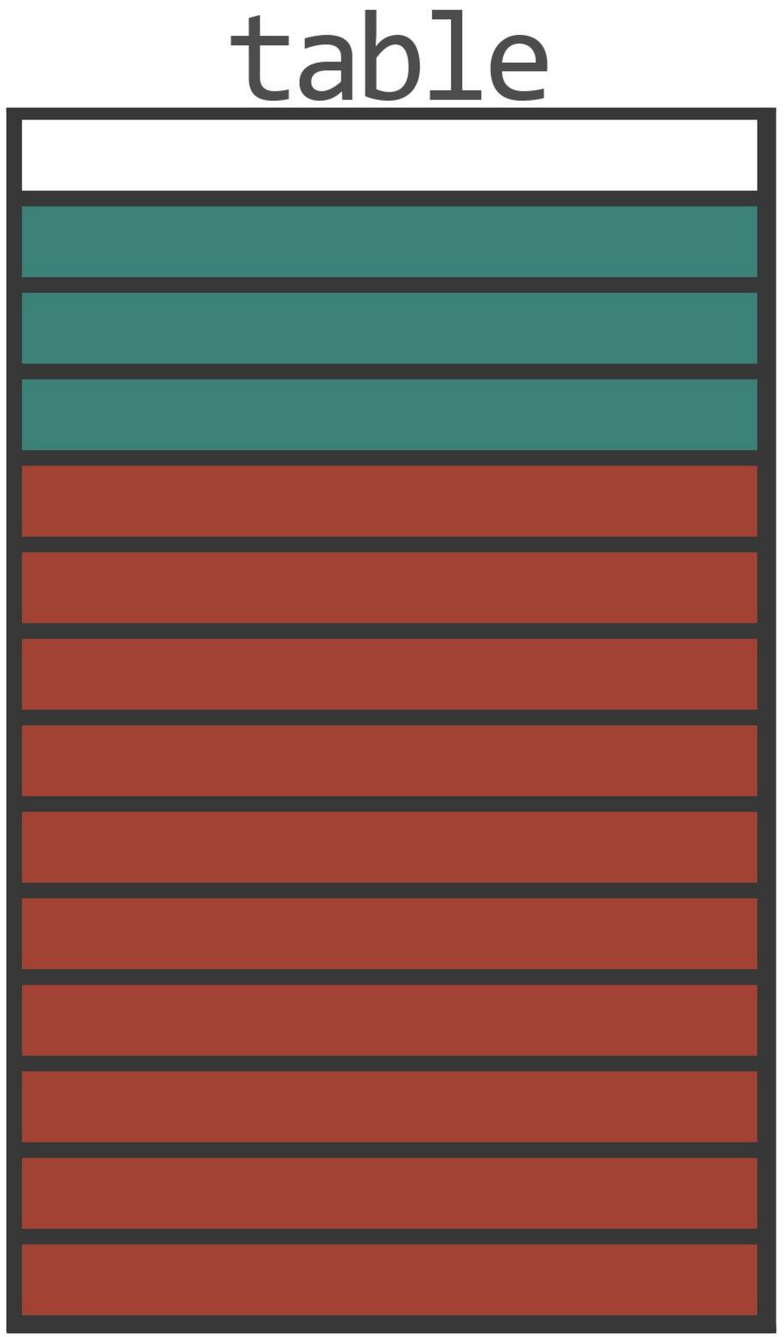


ACID



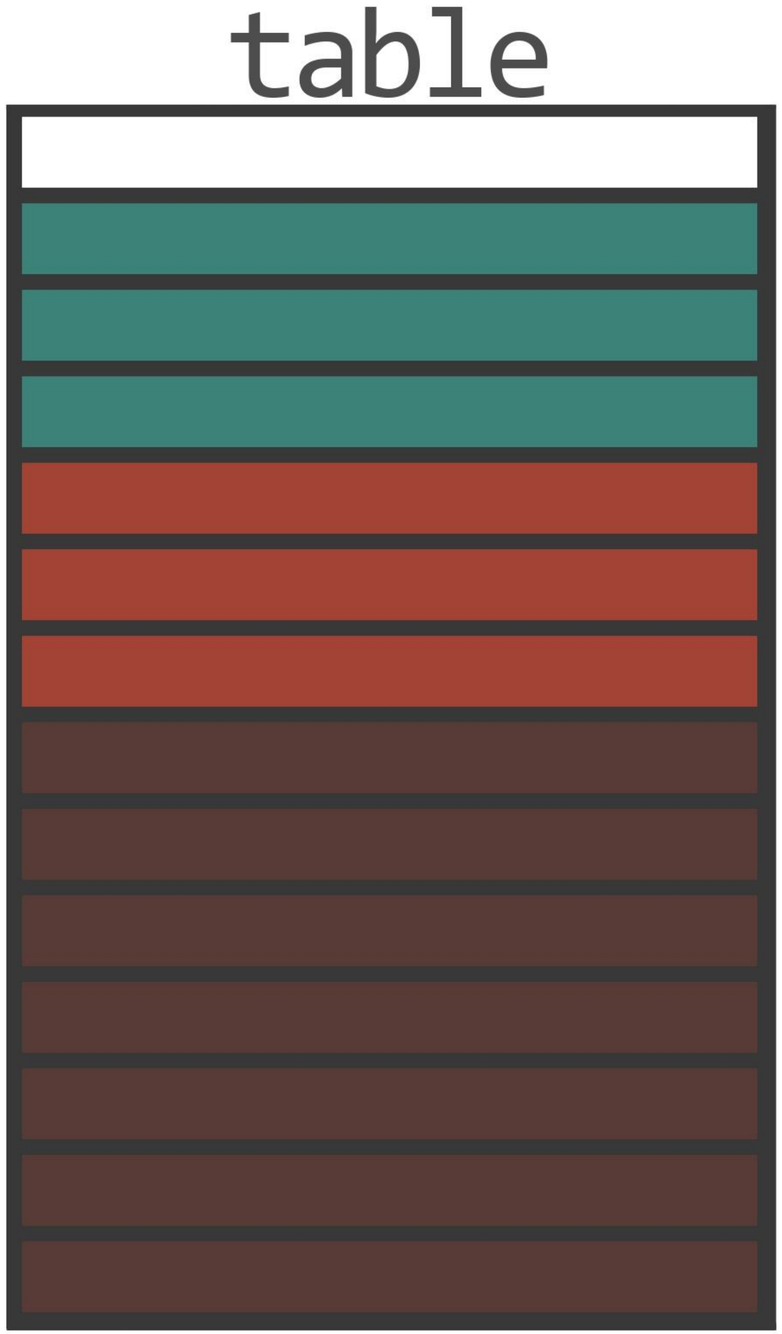
ACID

```
delete
from table
where
...
```



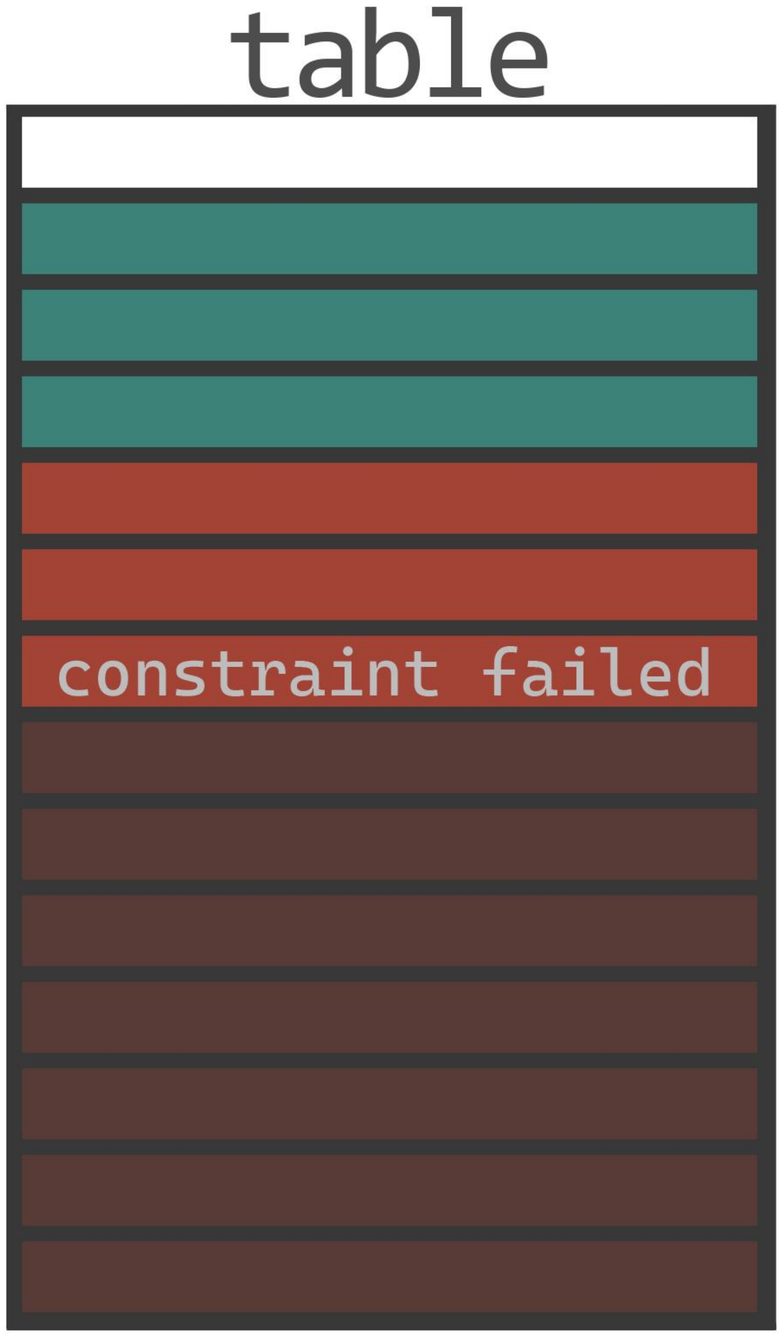
ACID

```
delete
from table
where
...
```



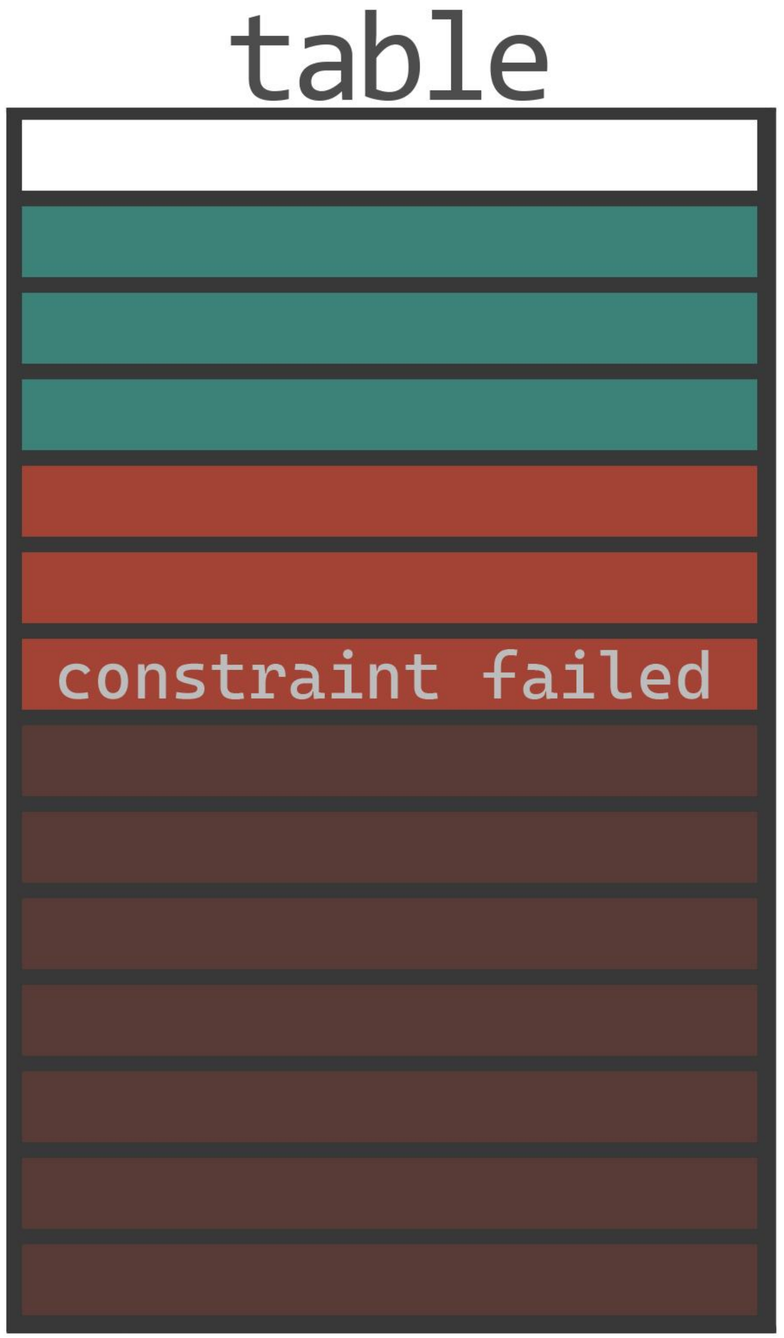
ACID

```
delete
from table
where
...
```



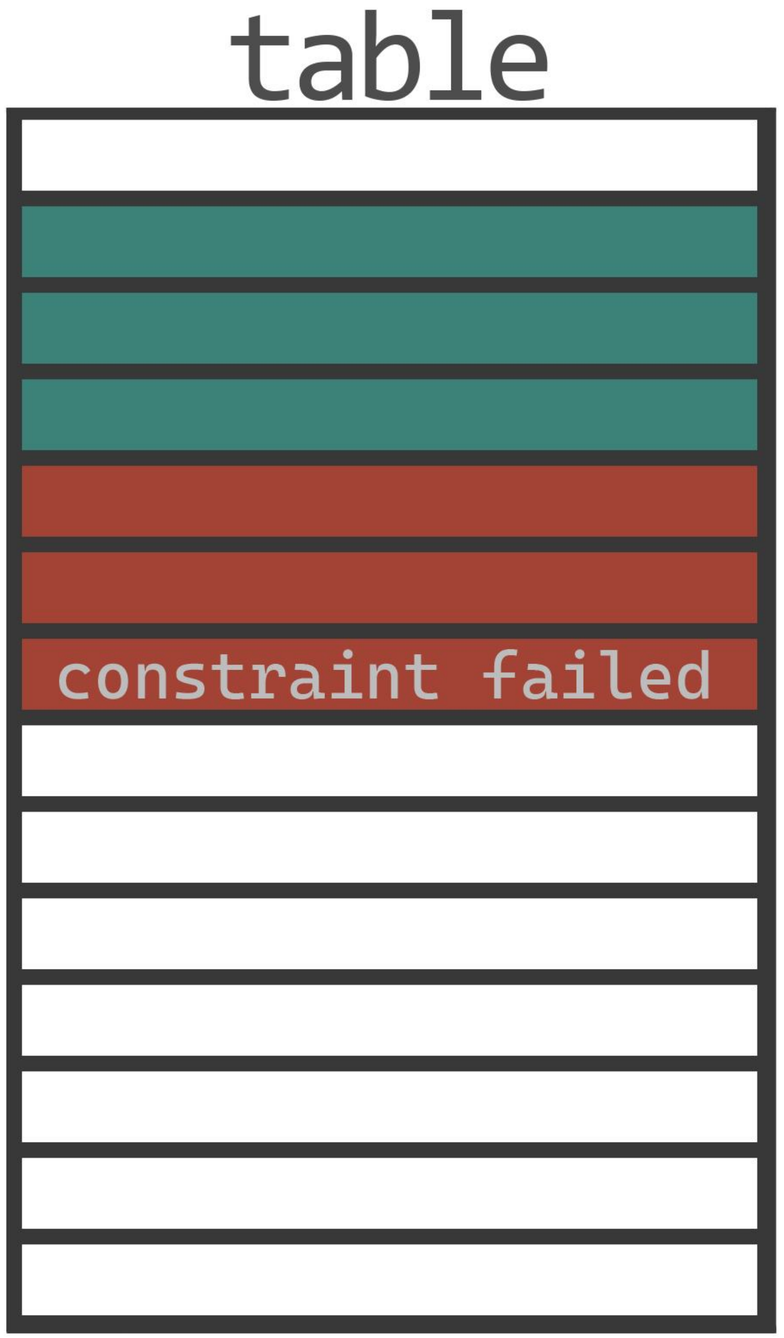
ACID

```
delete
from table
where
...
```



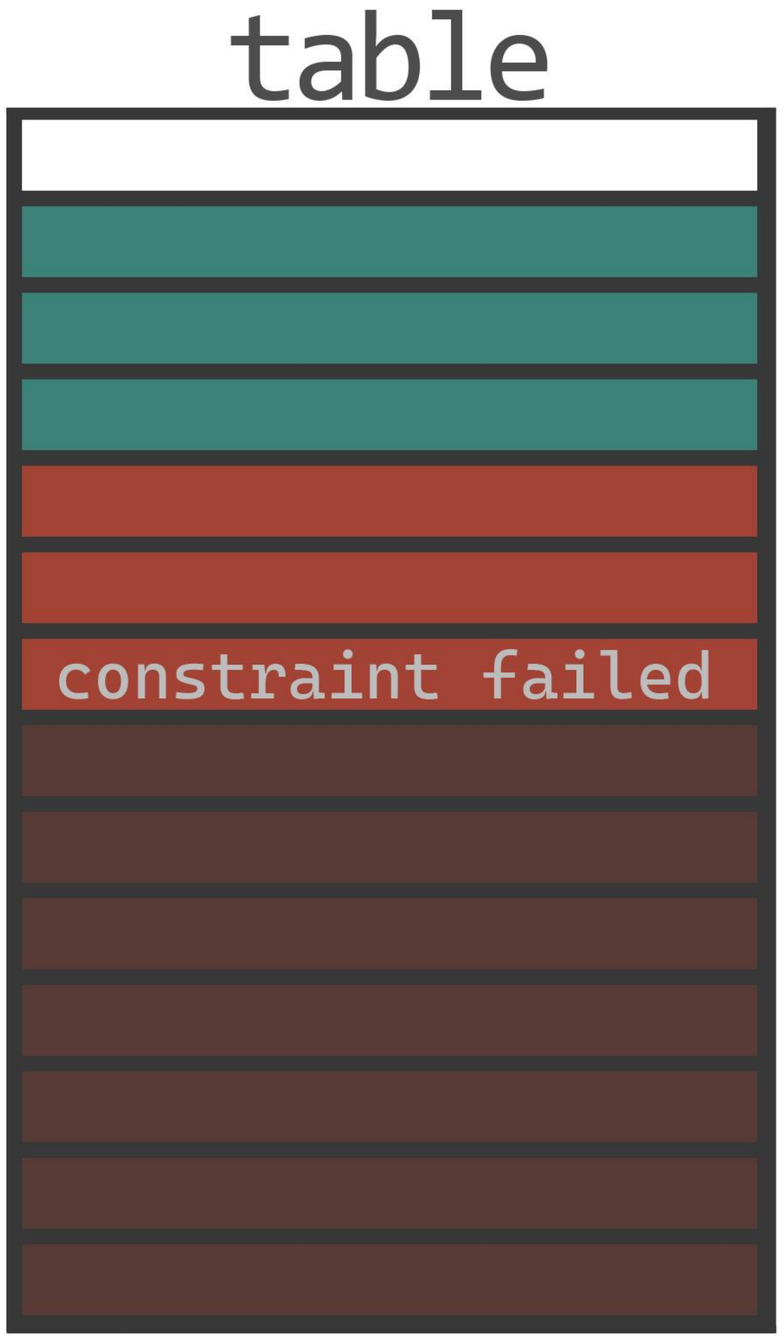
ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```

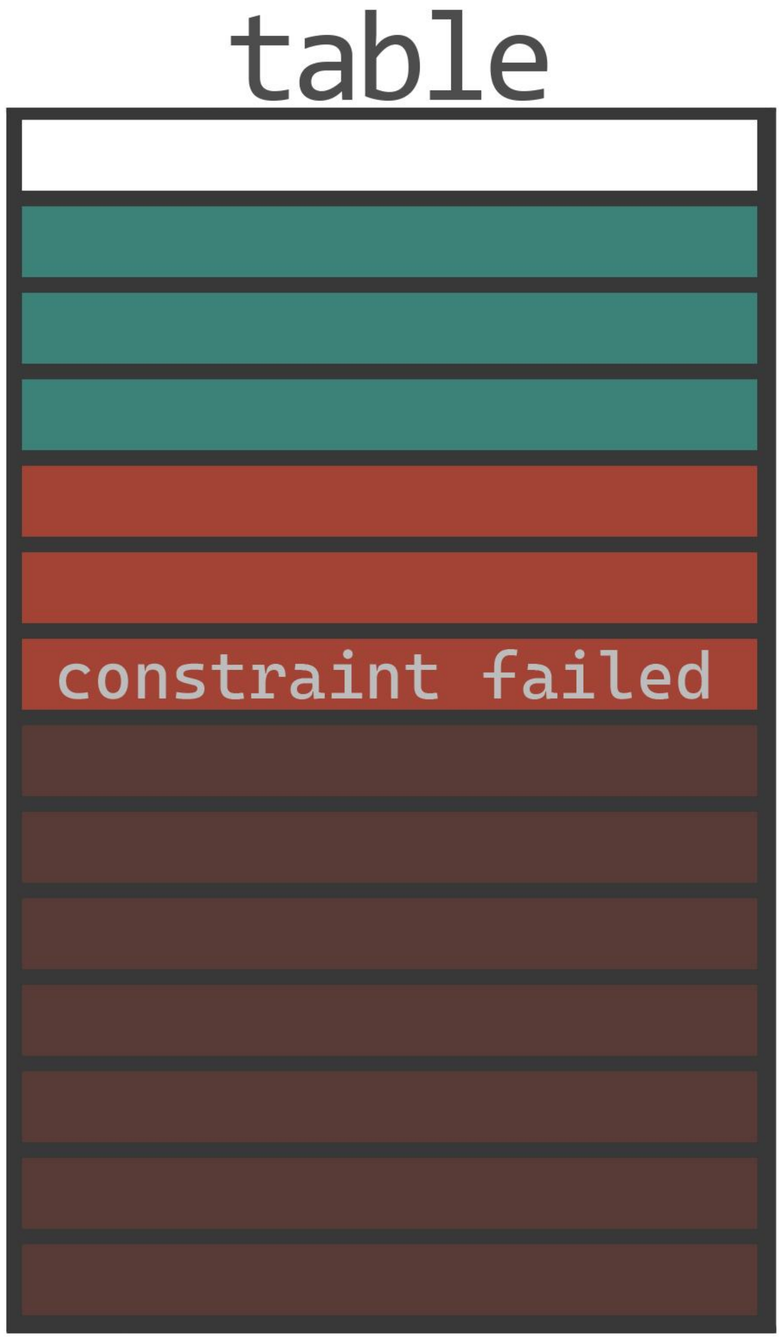
table

ACID

te

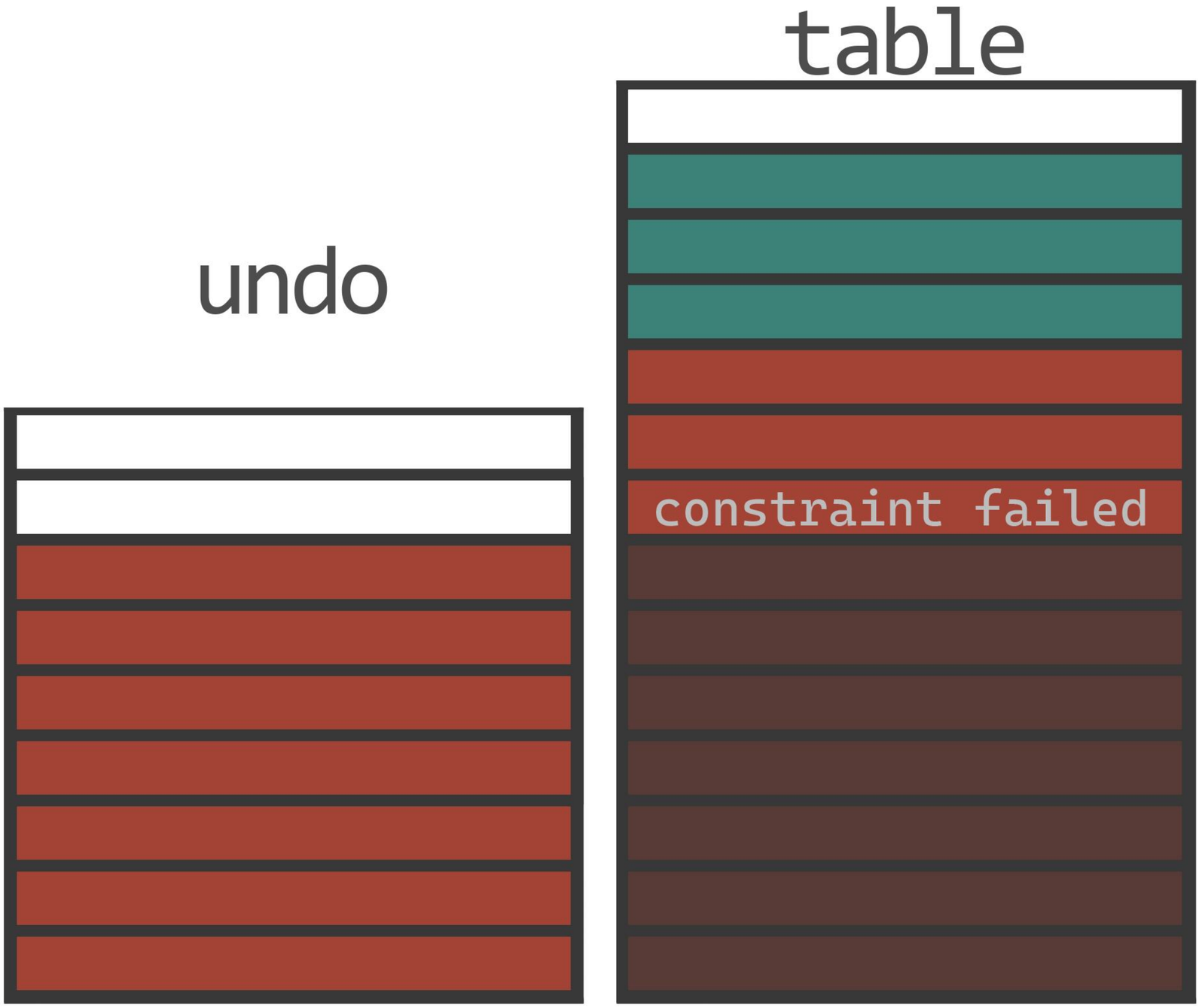
table

e



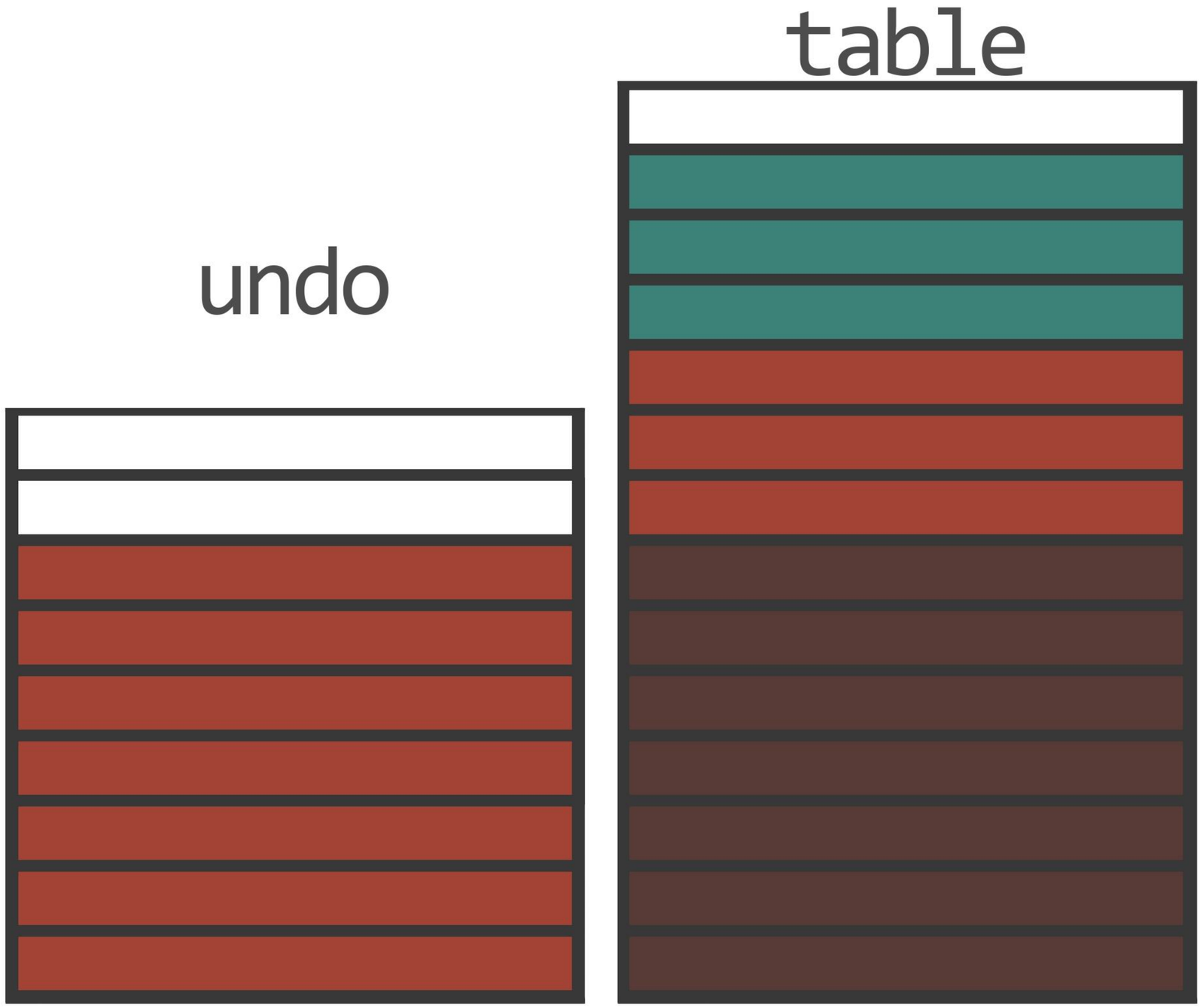
ACID

```
delete
from table
where
...
```



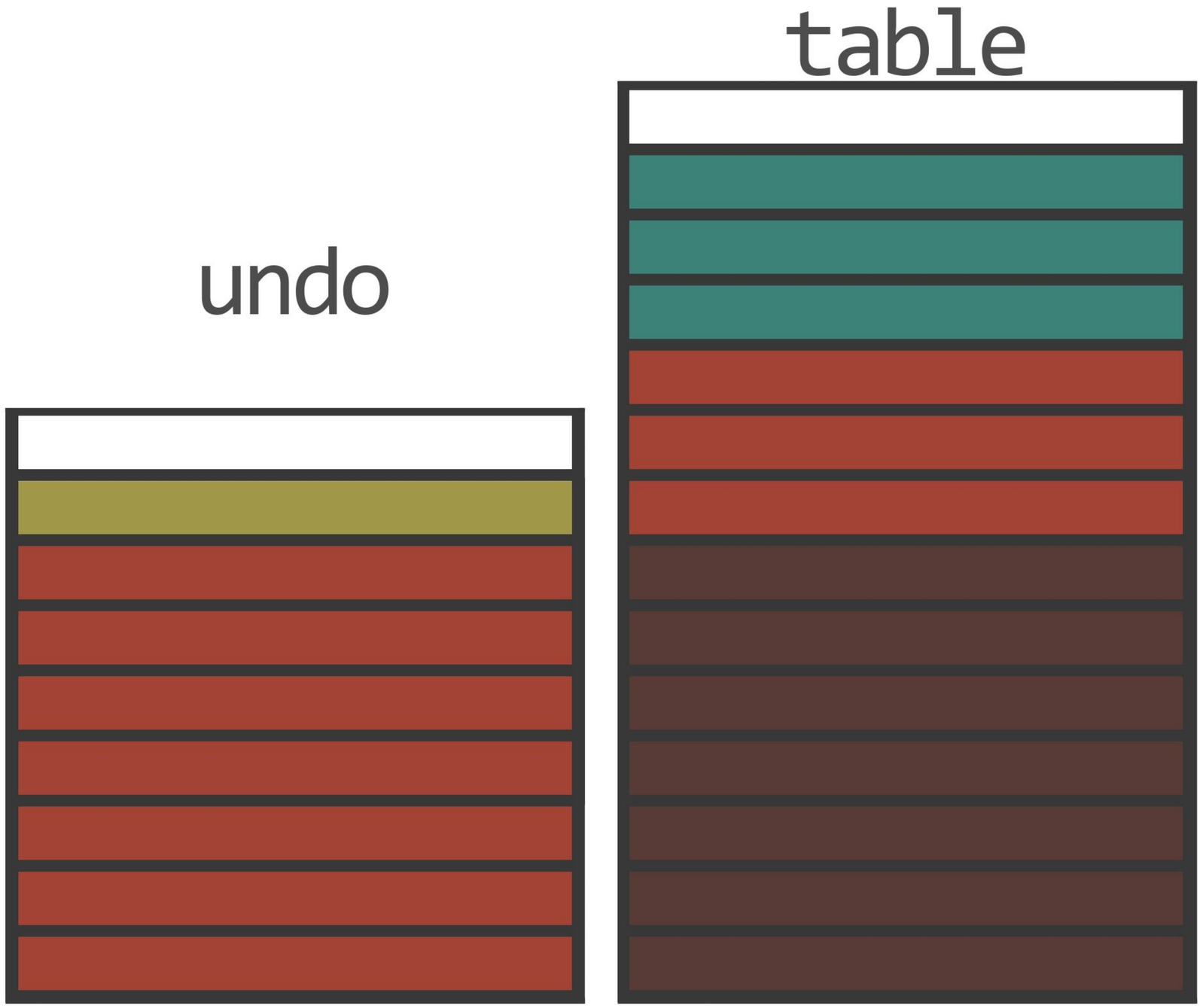
ACID

```
delete
from table
where
...
```



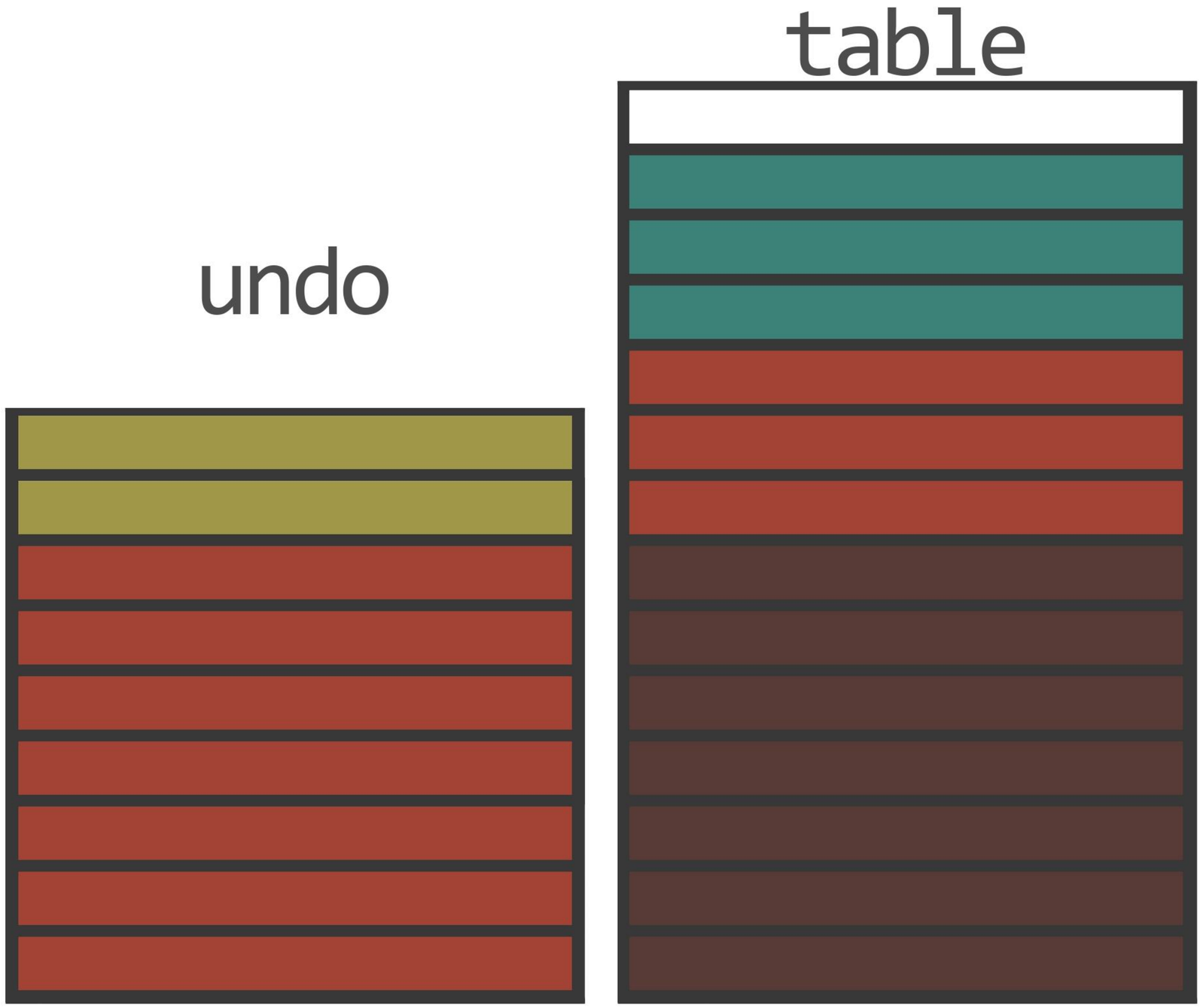
ACID

```
delete
from table
where
...
```



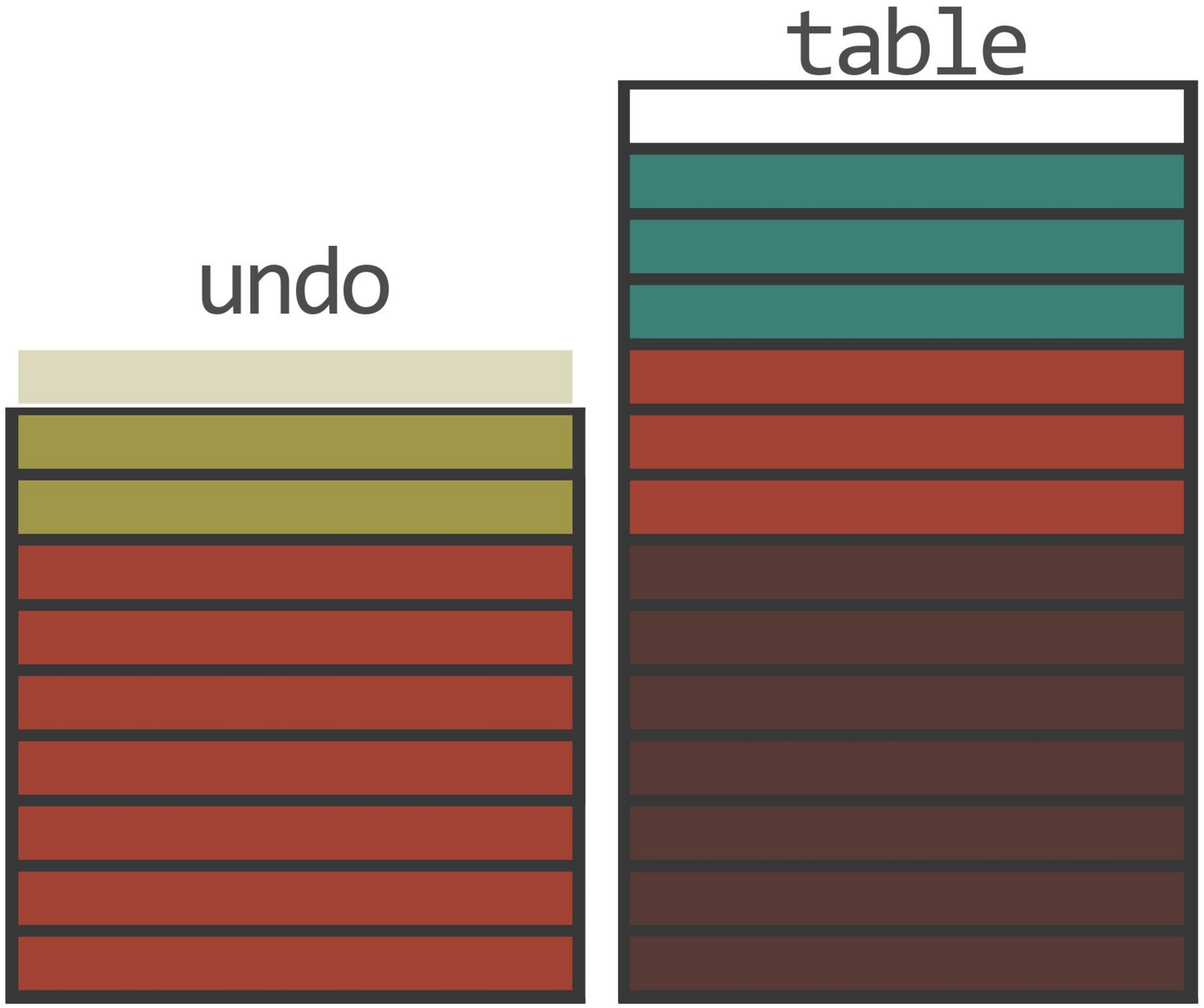
ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```



ACID

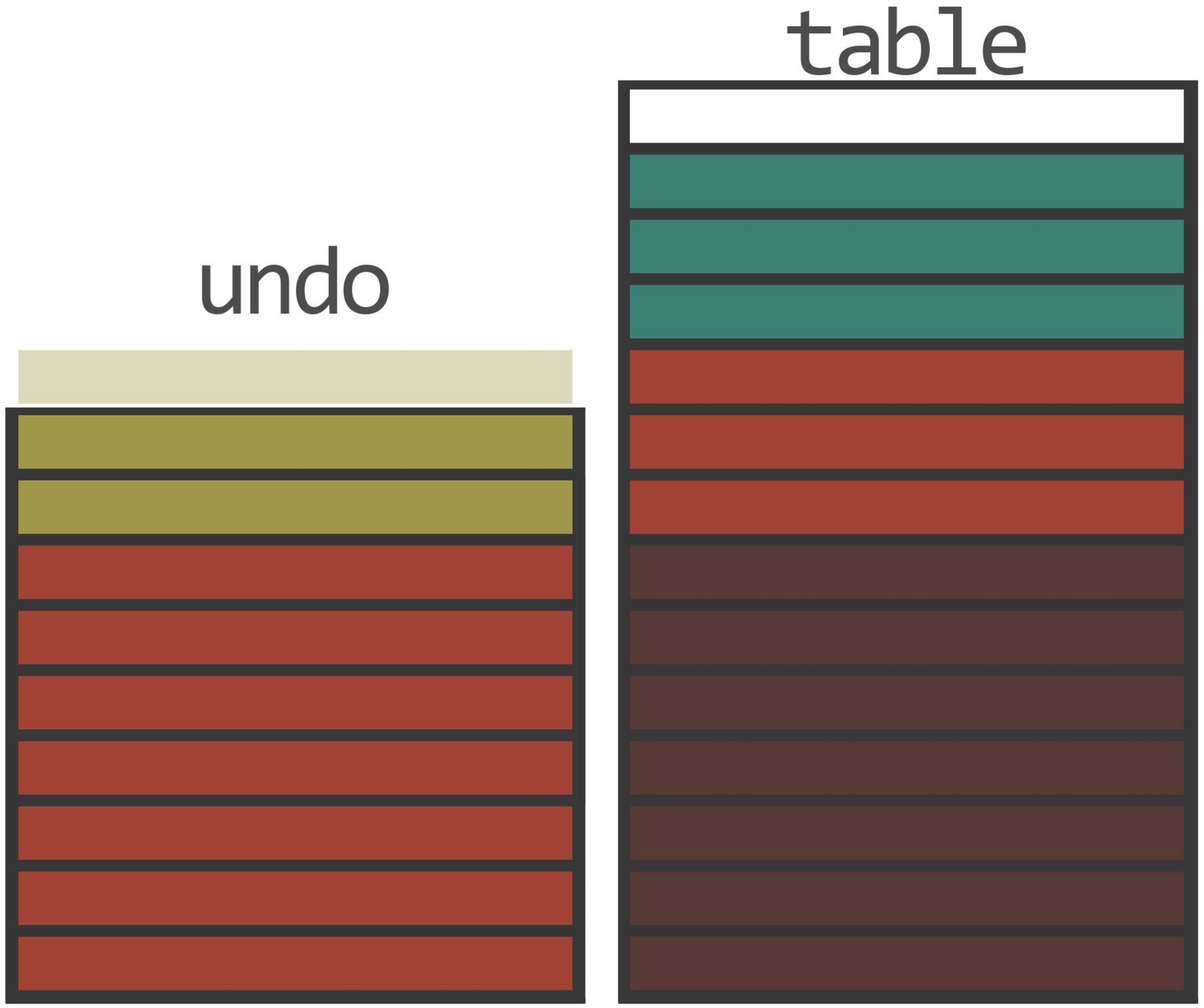
```
delete
from table
where
...
```

table

undo

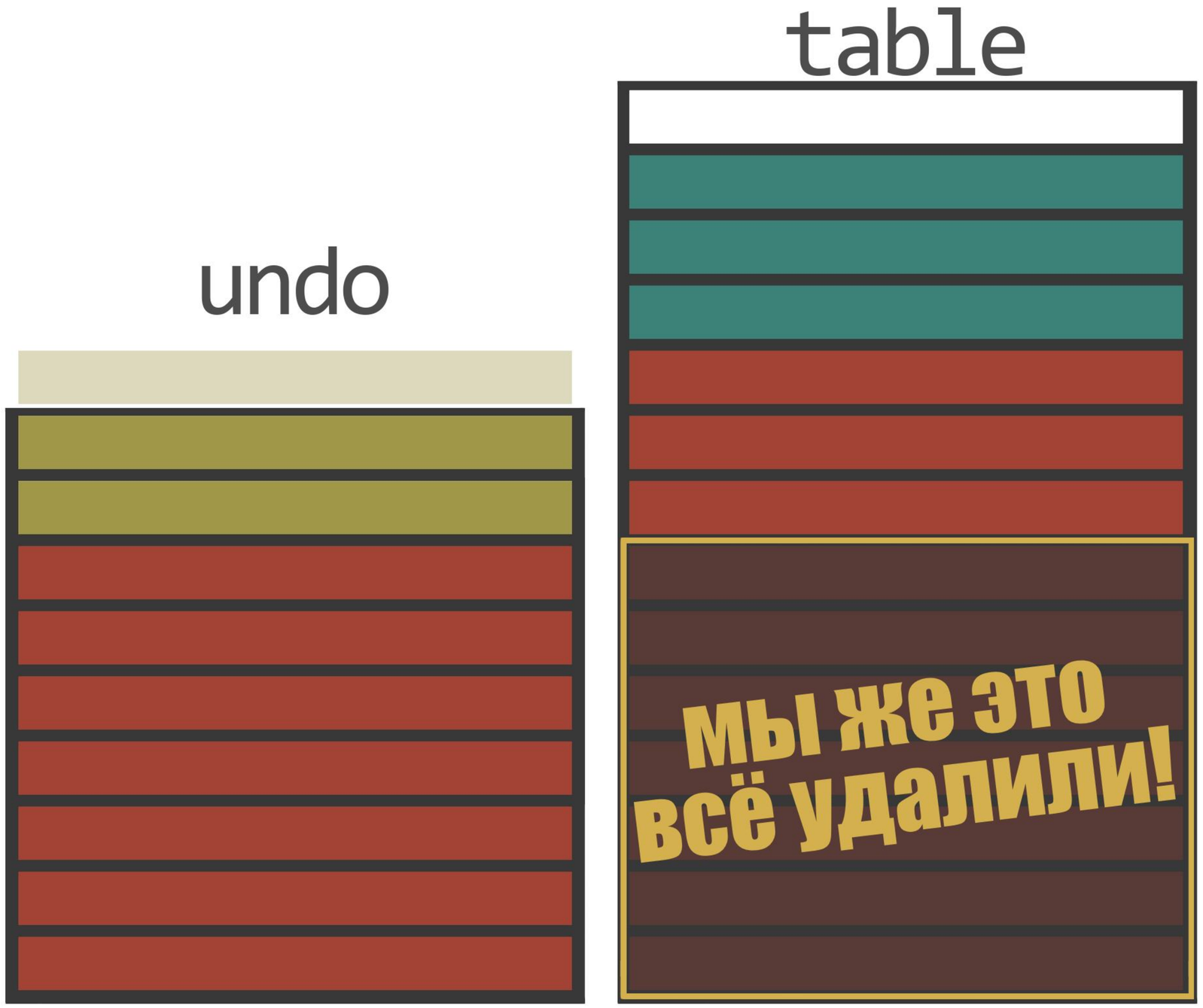
ACID

```
elete
rom table
where
...
```



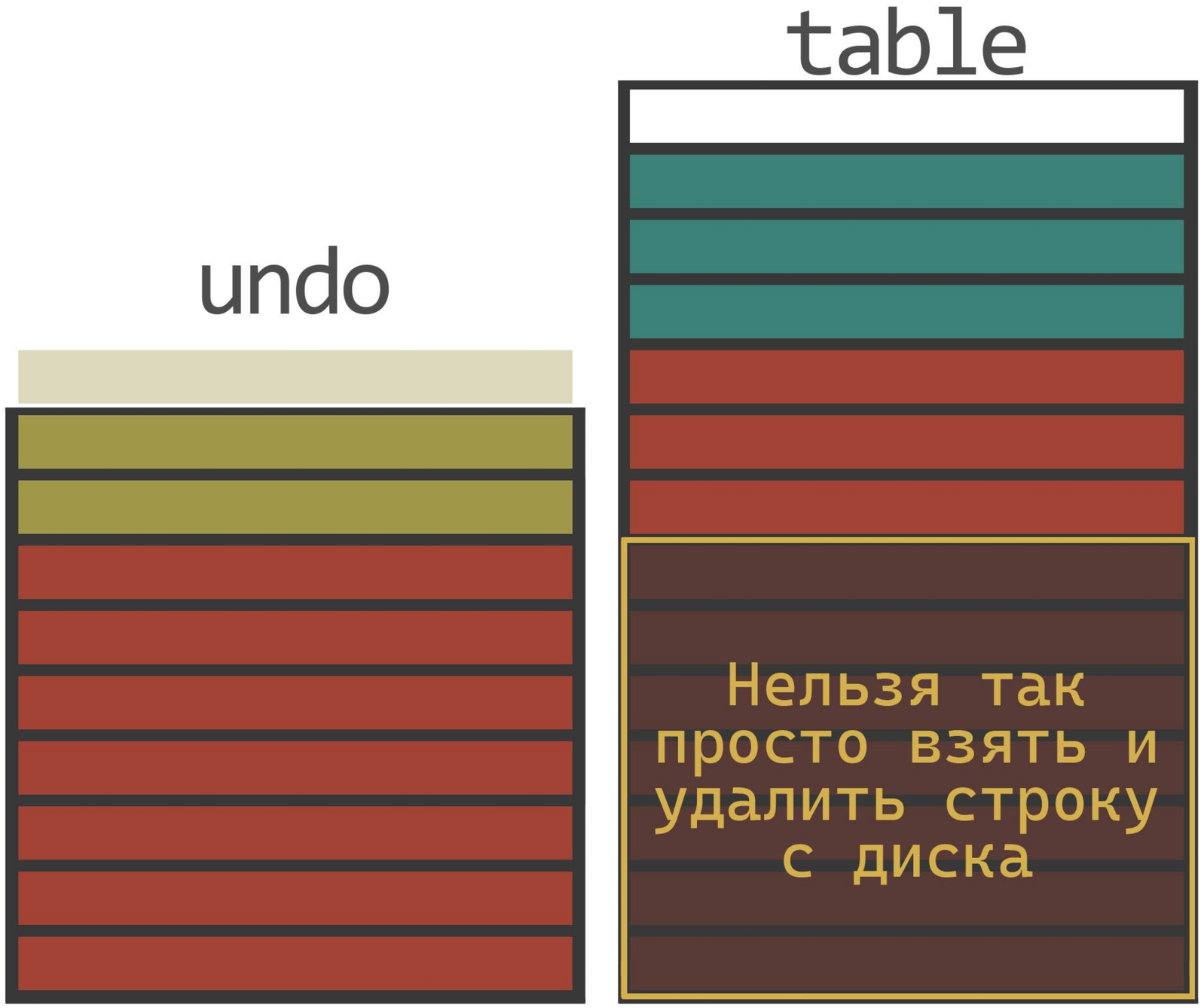
ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```



ACID

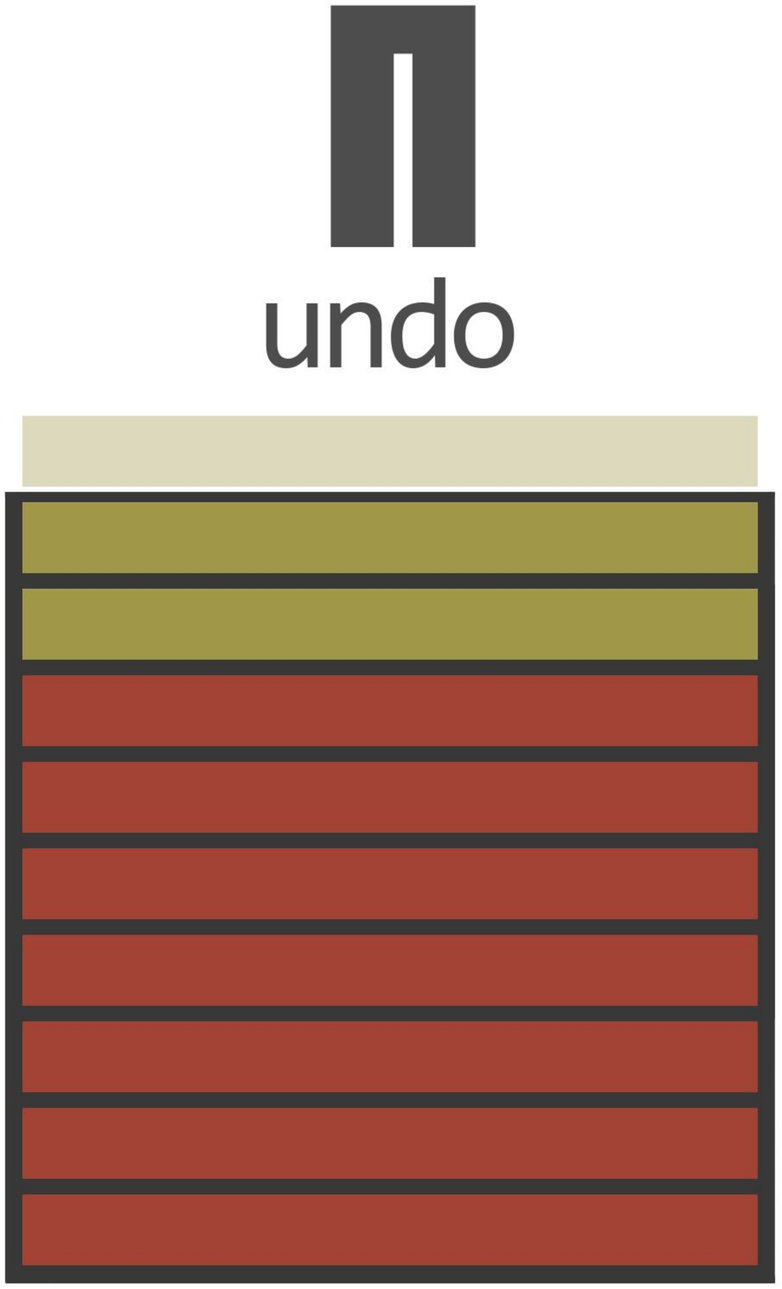
```
delete
from table
where
...
```

table

undo

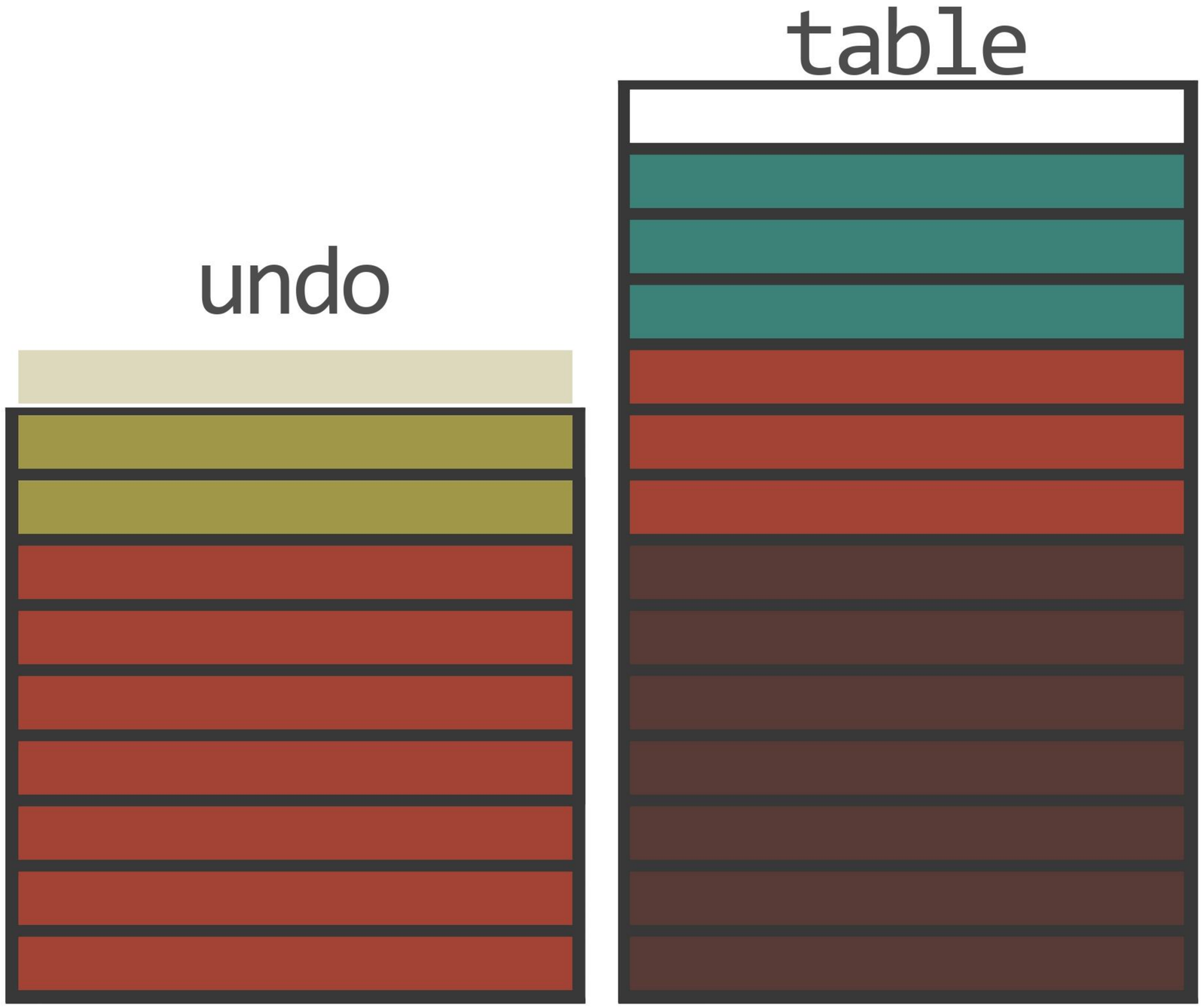
ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```



ACID

```
delete
from table
where
...
```

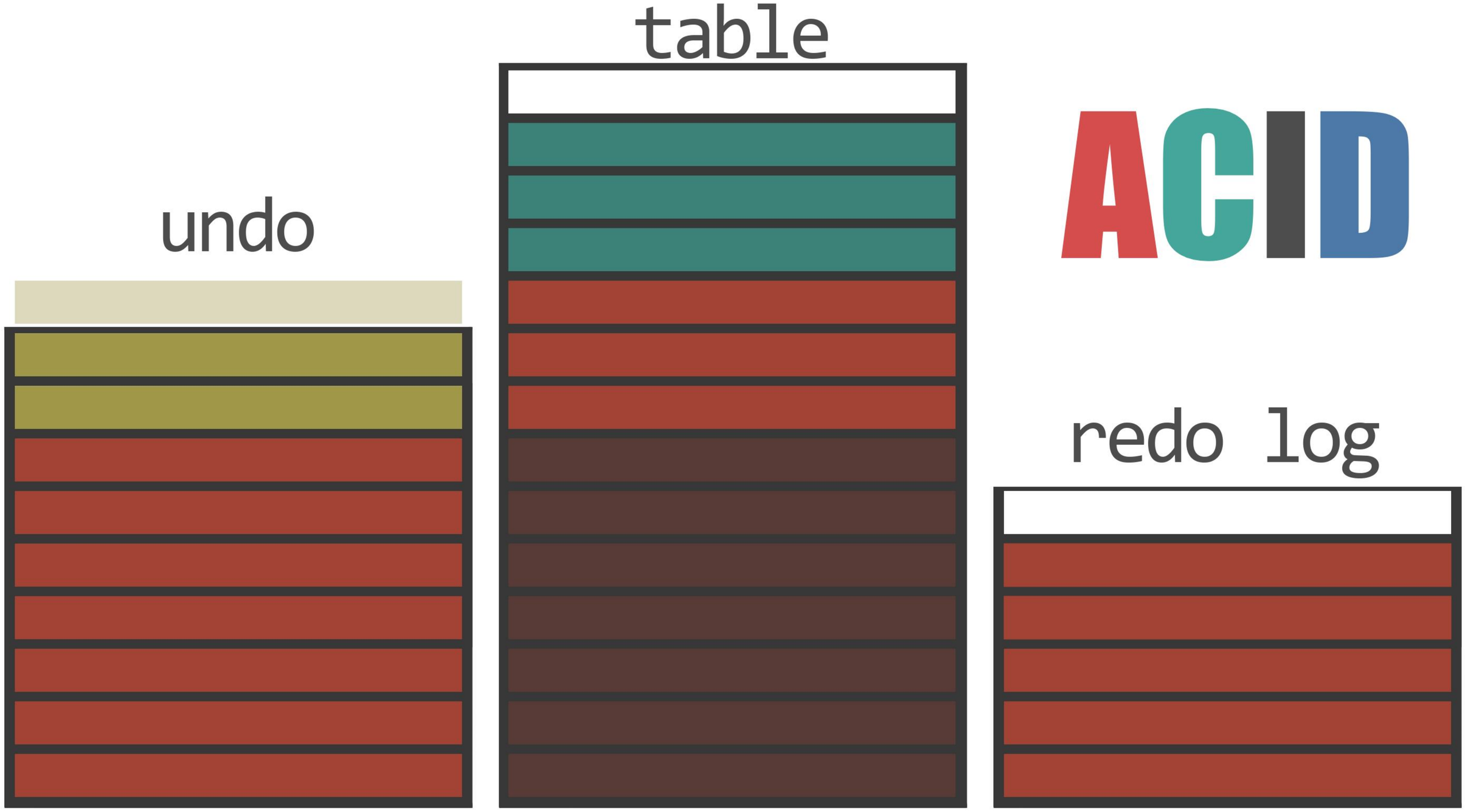
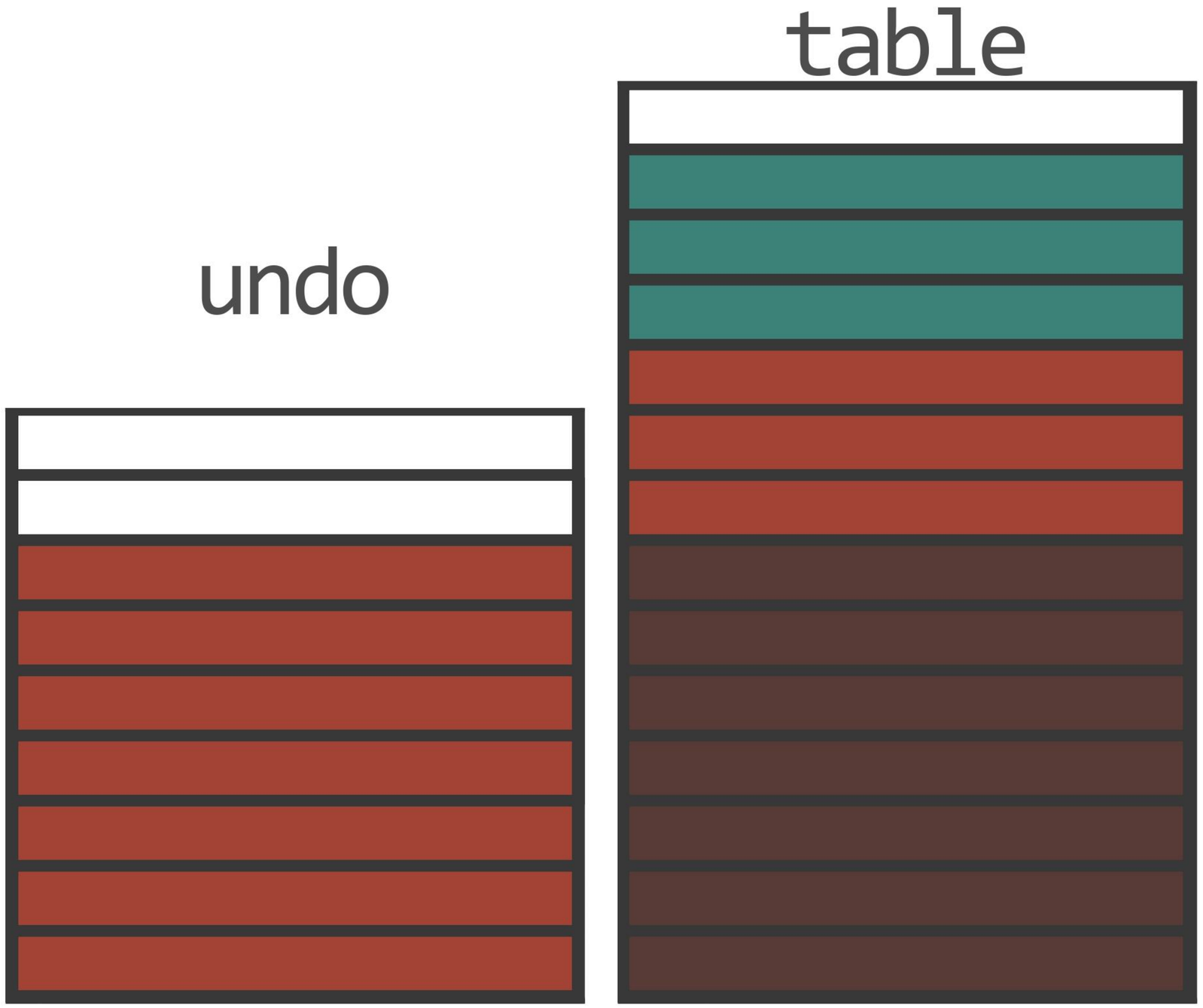
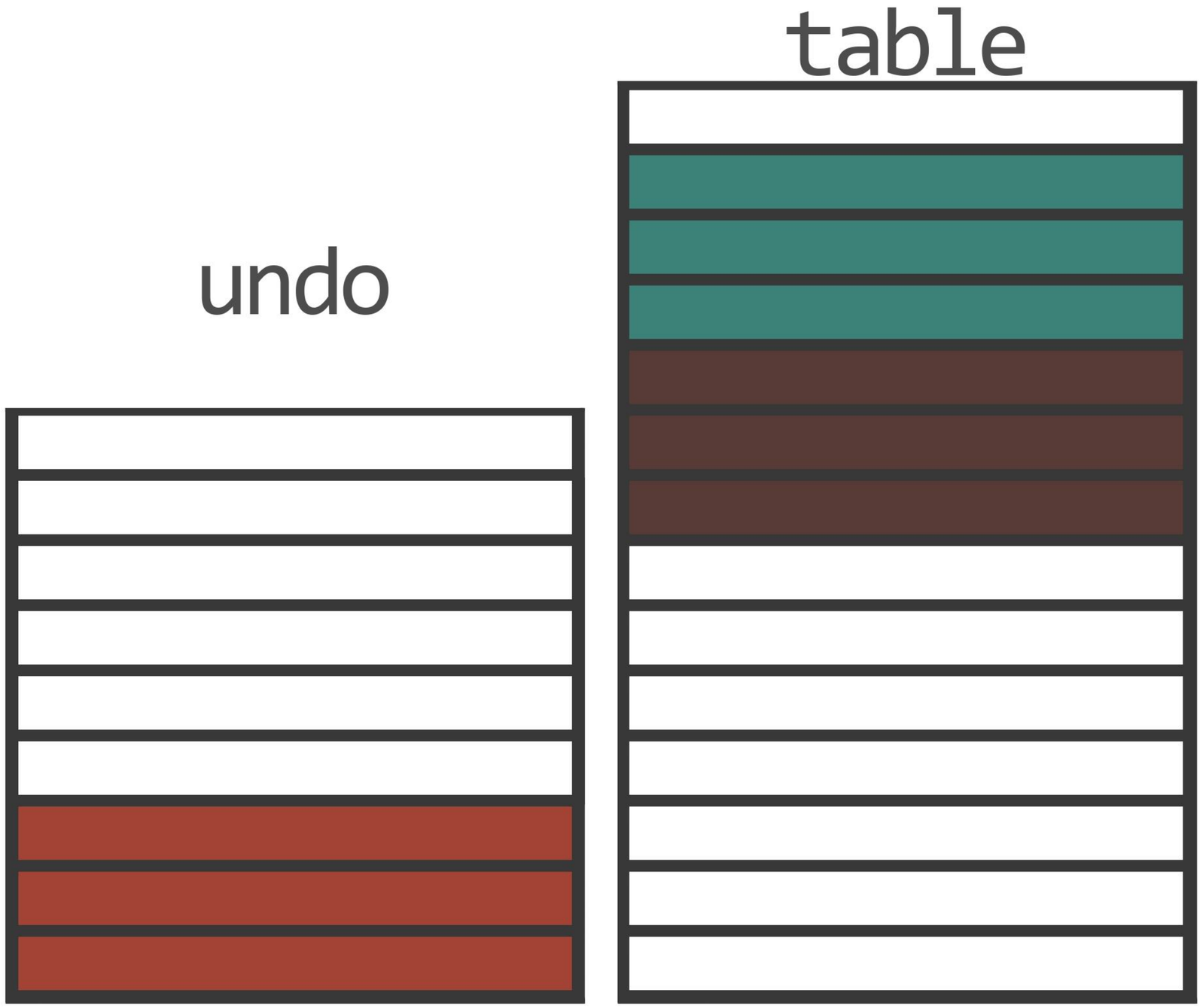


table
undo
ACID
redo log



ACID

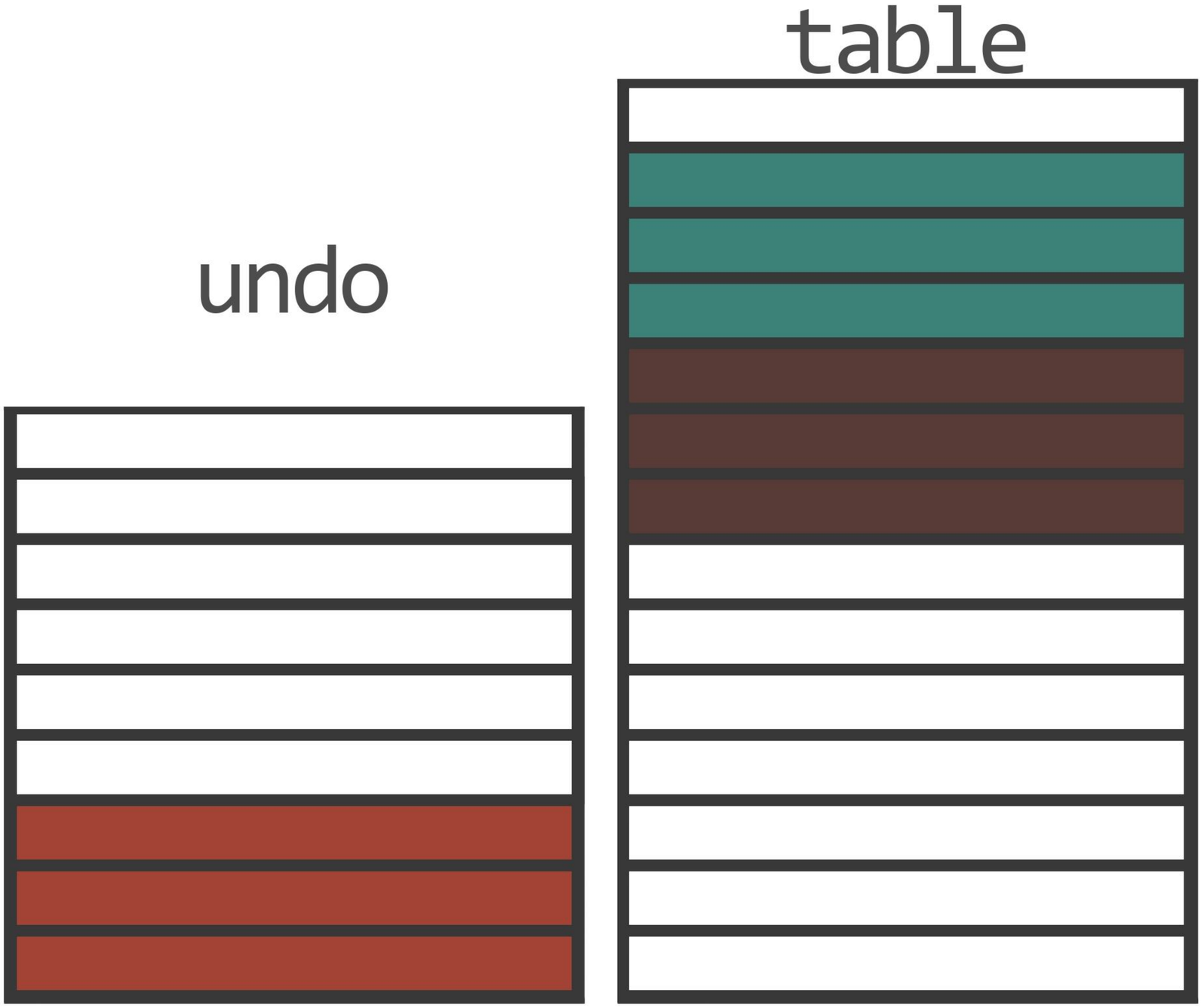
```
delete
from table
where
...
limit 7
```



ACID

```
delete
from table
where
...
limit 7
```

И так 50 раз



ACID

```
delete
from table t
where t.id in (
  select t1.id
  from table t1
  where
  ...
  limit 7
)
```

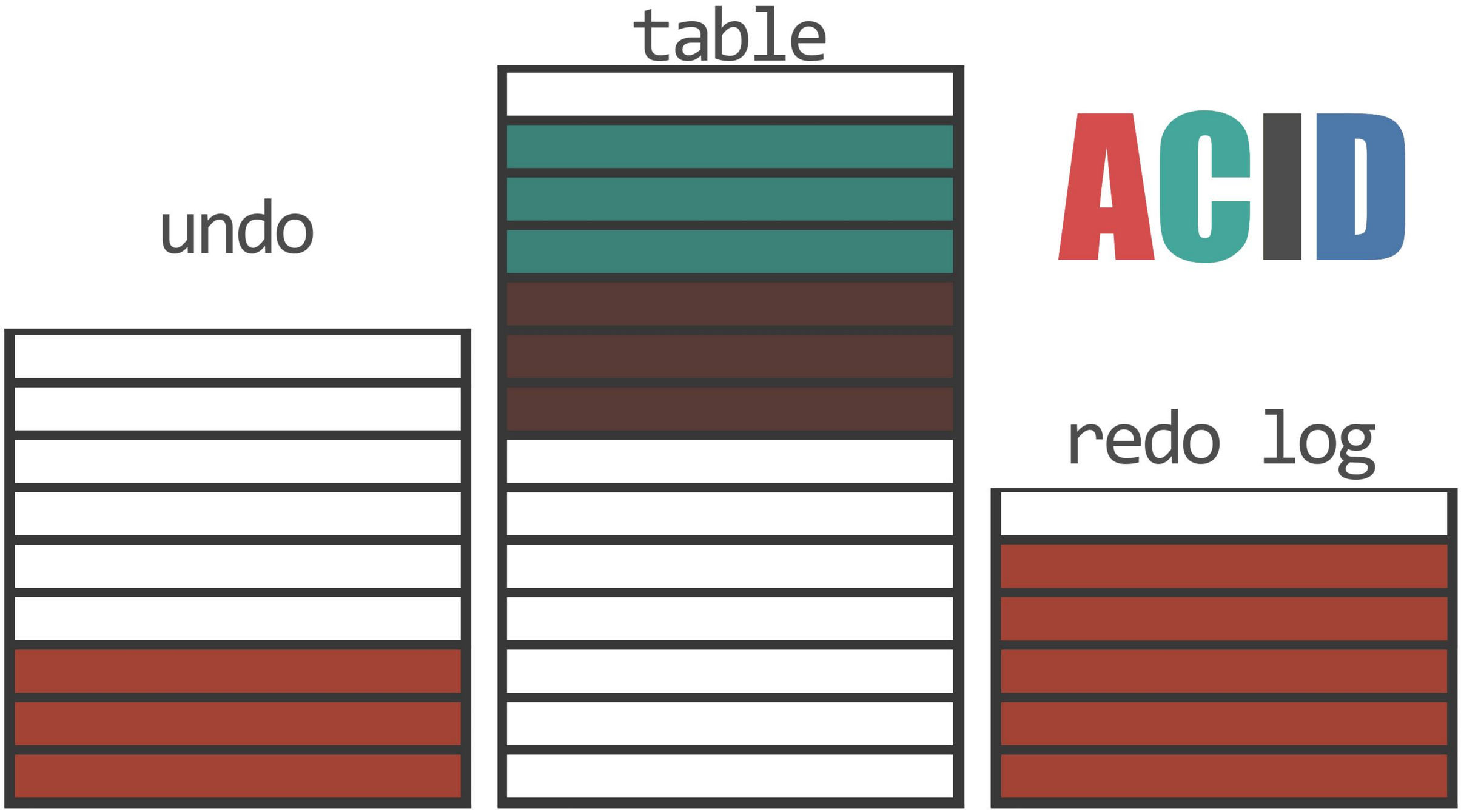


table
undo
ACID
redo log

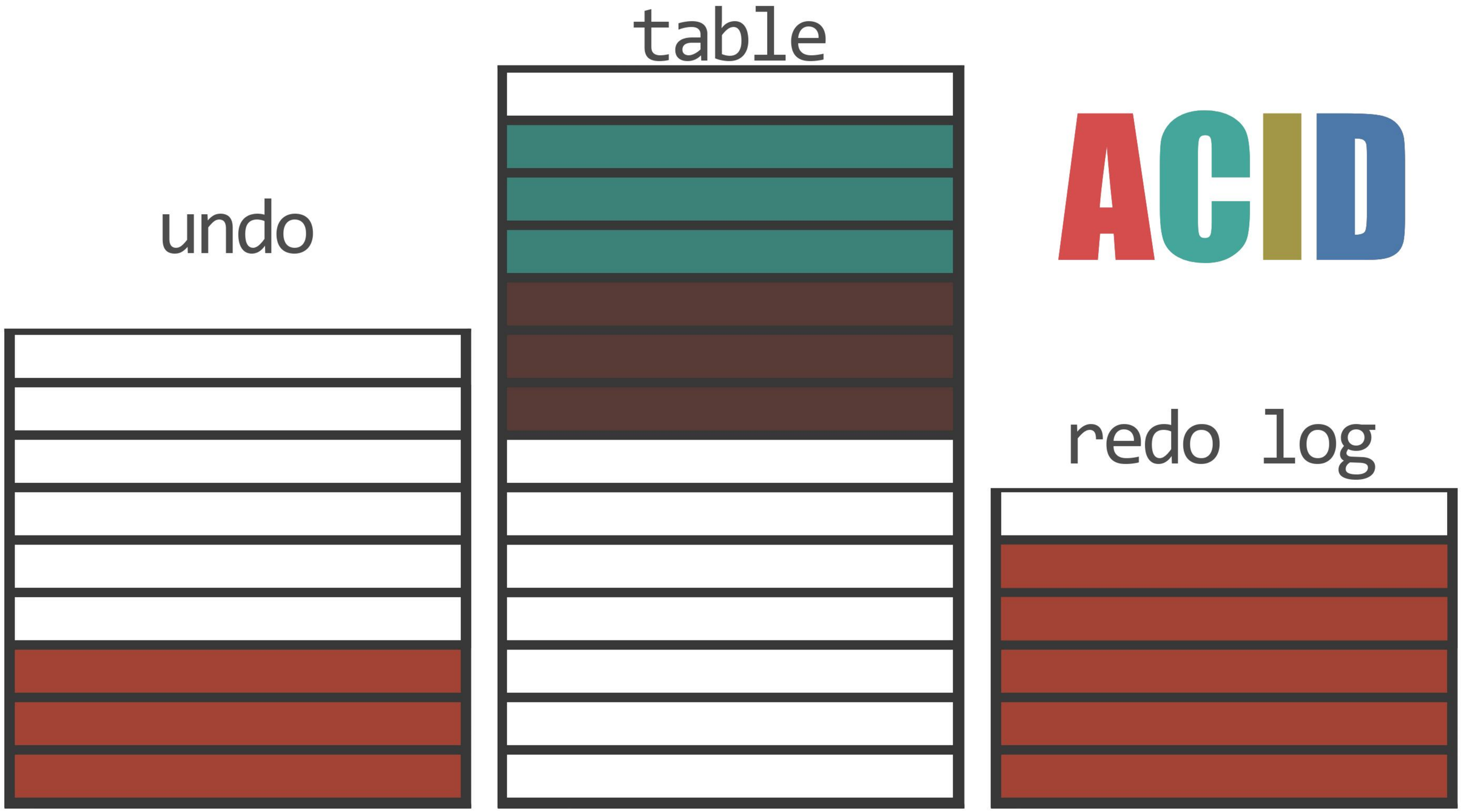


table
undo
ACID
redo log

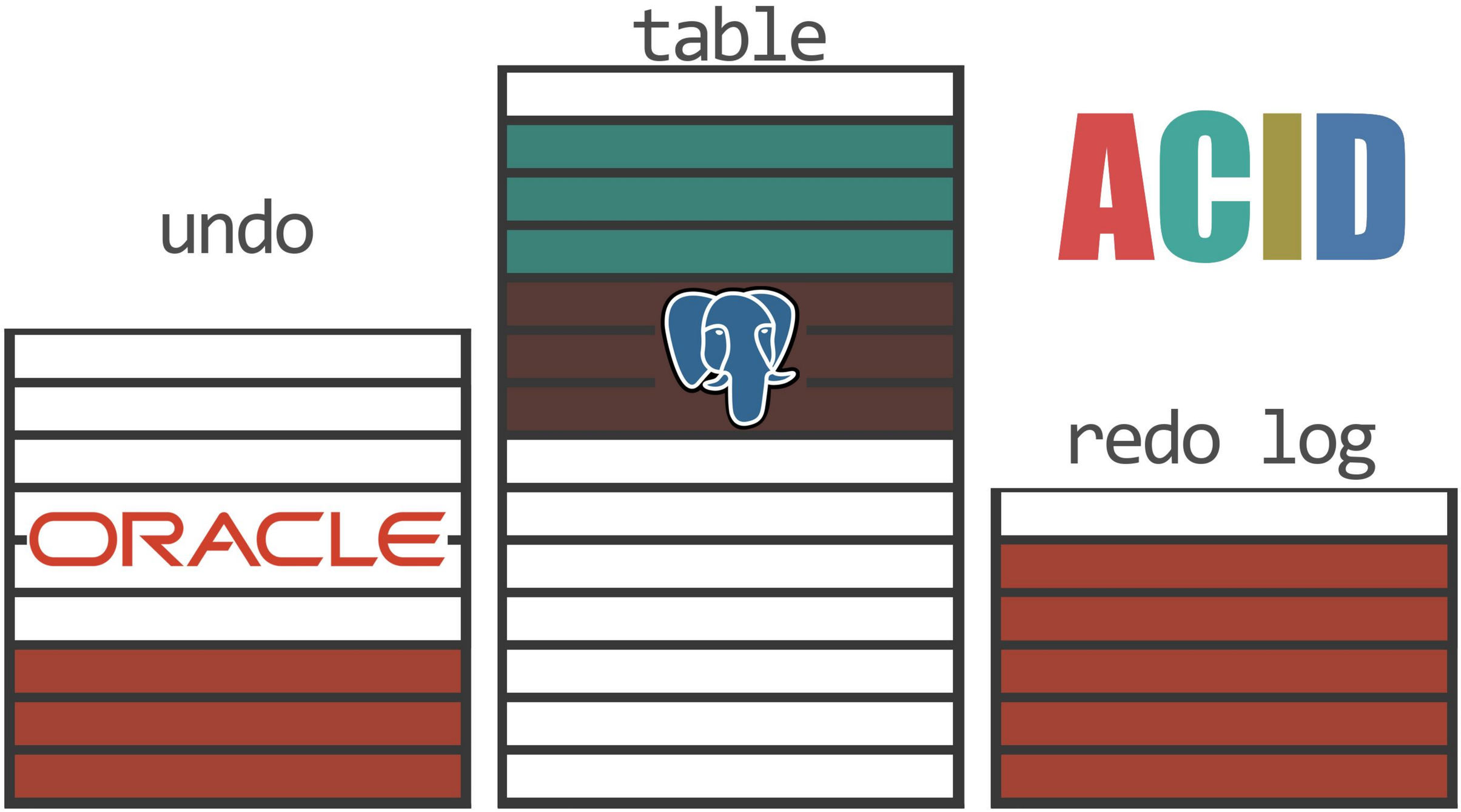


table
undo
ORACLE
redo log
ACID

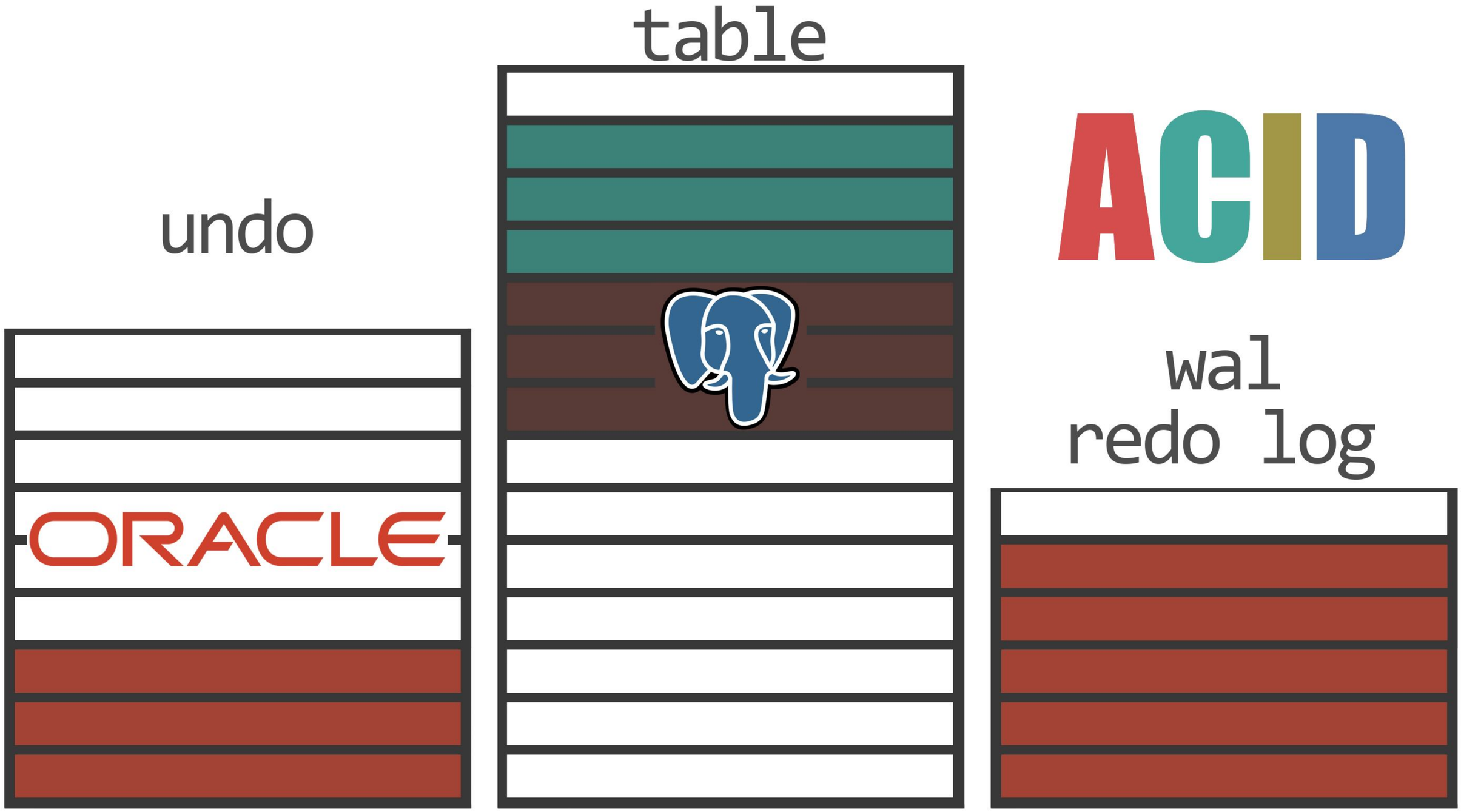


table
undo
ORACLE
ACID
wal
redo log

```
... 38 common frames omitted

2024-02-29T20:43:54.718+03:00  WARN 20412 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 0, SQLState: 53100
2024-02-29T20:43:54.718+03:00 ERROR 20412 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : PANIC: could not write to file "pg_wal/xlogtemp.30": No space left on device
2024-02-29T20:43:54.718+03:00  WARN 20412 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 0, SQLState: 08006
2024-02-29T20:43:54.718+03:00 ERROR 20412 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : An I/O error occurred while sending to the backend.
2024-02-29T20:43:54.730+03:00 ERROR 20412 --- [           main] o.s.t.l.TransactionInterceptor           : Application exception overridden by rollback exception

org.hibernate.exception.GenericJDBCException: JDBC exception executing SQL [delete from books] [PANIC: could not write to file "pg_wal/xlogtemp.30": No space left on device] [n/a]
        at org.hibernate.exception.internal.StandardSQLExceptionConverter.convert(StandardSQLExceptionConverter.java:63) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.engine.jdbc.spi.SqlExceptionHelper.convert(SqlExceptionHelper.java:108) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.engine.jdbc.spi.SqlExceptionHelper.convert(SqlExceptionHelper.java:94) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.sql.exec.internal.StandardJdbcMutationExecutor.execute(StandardJdbcMutationExecutor.java:96) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.query.sqm.internal.SimpleDeleteQueryPlan.executeUpdate(SimpleDeleteQueryPlan.java:201) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.query.sqm.internal.QuerySqmImpl.doExecuteUpdate(QuerySqmImpl.java:705) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.hibernate.query.sqm.internal.QuerySqmImpl.executeUpdate(QuerySqmImpl.java:675) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
        at org.springframework.data.jpa.repository.support.SimpleJpaRepository.deleteAllInBatch(SimpleJpaRepository.java:296) ~[spring-data-jpa-3.2.0.jar!/:3.2.0]
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na]
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) ~[na:na]
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
        at java.base/java.lang.reflect.Method.invoke(Method.java:568) ~[na:na]
        at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:352) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.data.repository.core.support.RepositoryMethodInvoker$RepositoryFragmentMethodInvoker.lambda$new$0(RepositoryMethodInvoker.java:277) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.RepositoryMethodInvoker.doInvoke(RepositoryMethodInvoker.java:170) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.RepositoryMethodInvoker.invoke(RepositoryMethodInvoker.java:158) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.RepositoryComposition$RepositoryFragments.invoke(RepositoryComposition.java:516) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.RepositoryComposition.invoke(RepositoryComposition.java:285) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.RepositoryFactorySupport$ImplementationMethodExecutionInterceptor.invoke(RepositoryFactorySupport.java:628) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.data.repository.core.support.QueryExecutorMethodInterceptor.doInvoke(QueryExecutorMethodInterceptor.java:168) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.data.repository.core.support.QueryExecutorMethodInterceptor.invoke(QueryExecutorMethodInterceptor.java:143) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.data.projection.DefaultMethodInvokingMethodInterceptor.invoke(DefaultMethodInvokingMethodInterceptor.java:70) ~[spring-data-commons-3.2.0.jar!/:3.2.0]
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.transaction.interceptor.TransactionInterceptor$1.proceedWithInvocation(TransactionInterceptor.java:123) ~[spring-tx-6.1.1.jar!/:6.1.1]
        at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:385) ~[spring-tx-6.1.1.jar!/:6.1.1]
        at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:119) ~[spring-tx-6.1.1.jar!/:6.1.1]
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:137) ~[spring-tx-6.1.1.jar!/:6.1.1]
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.1.jar!/:6.1.1]
        at org.springframework.data.repository.core.support.SurroundingTransactionDetectorMethodInterceptor.invoke(SurroundingTransactionDetectorMethodInterceptor.java:57) ~[spring-data-commons-3.2.0.jar!/:3.2.0
]
```

```
 SQL [delete from books] [PANIC: could not write to file "pg_wal/xlogtemp.30": No space left on device] [n/a]
ter.convert(StandardSQLExceptionConverter.java:63) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
lExceptionHelper.java:108) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
lExceptionHelper.java:94) ~[hibernate-core-6.3.1.Final.jar!/:6.3.1.Final]
```

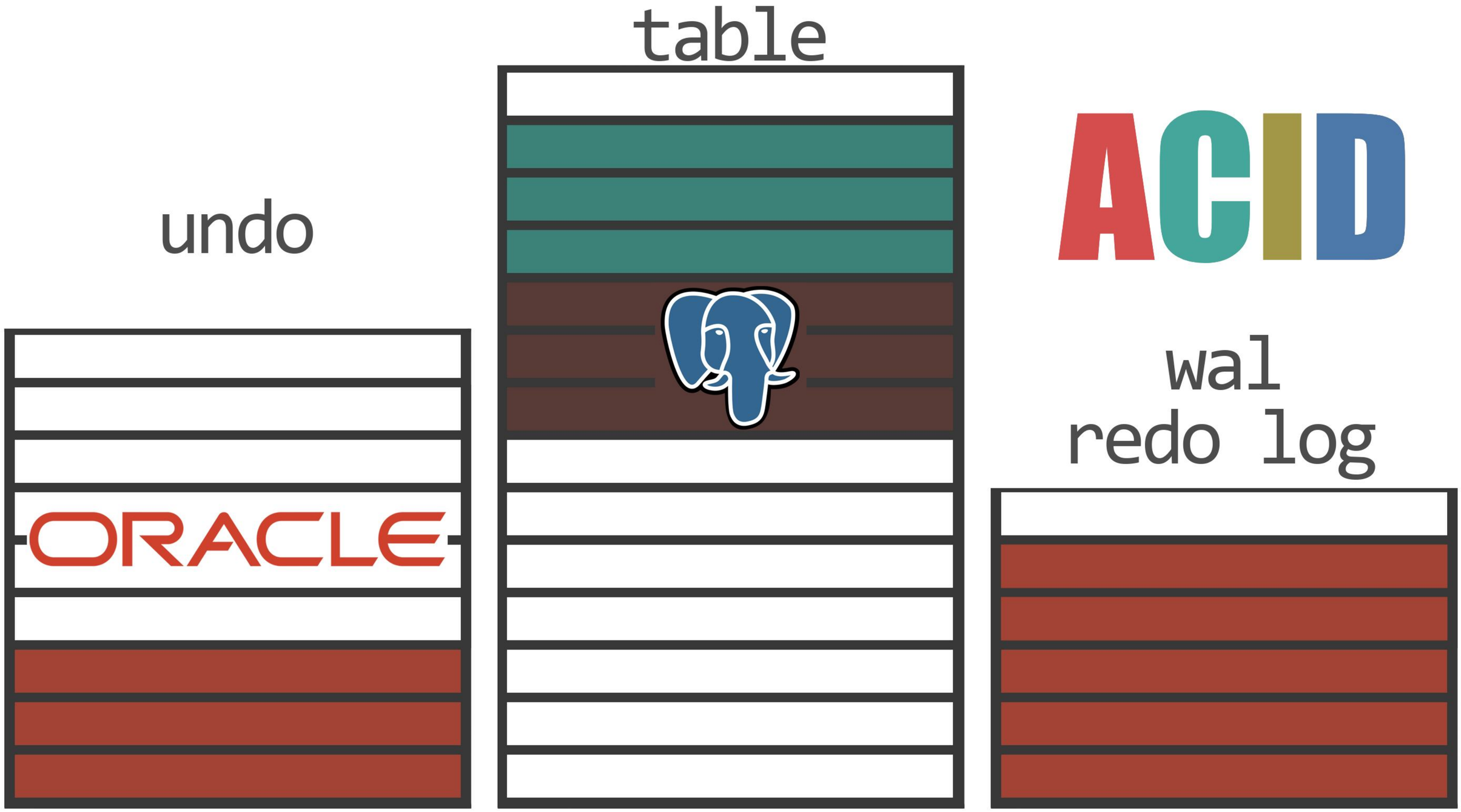
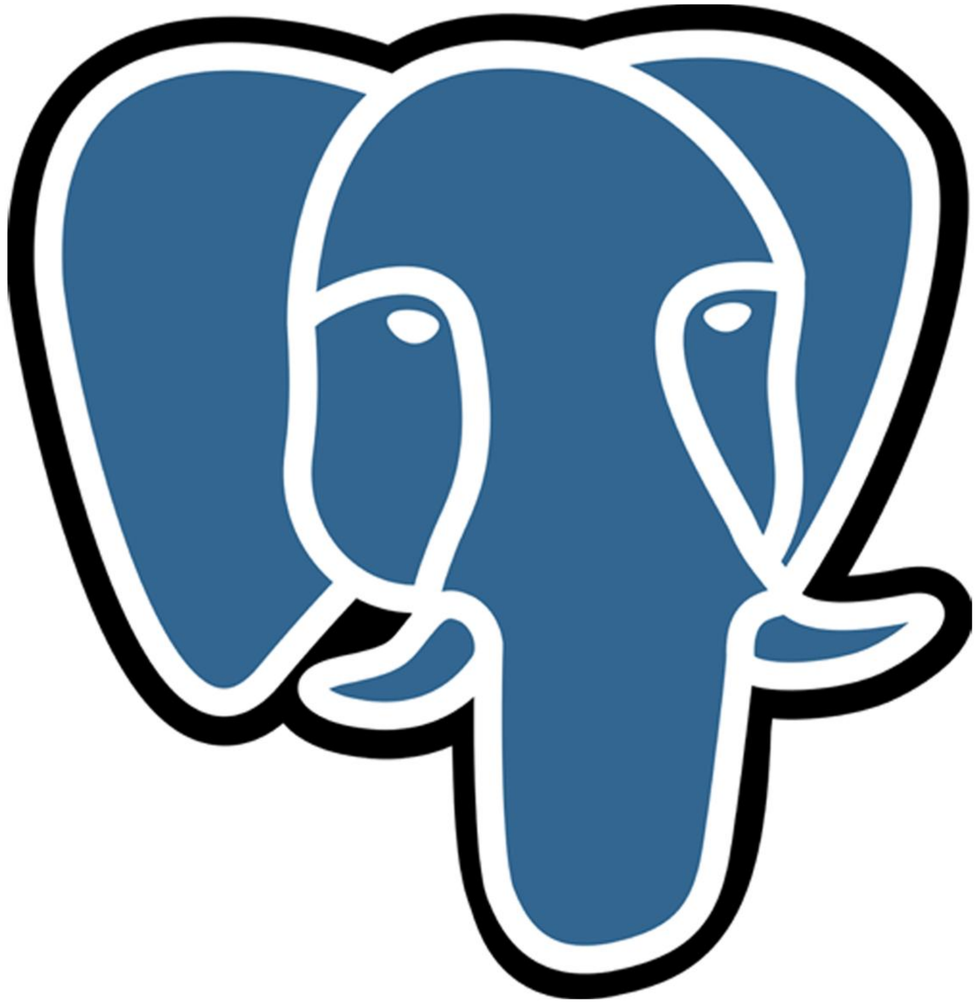


table
undo
ORACLE
ACID
wal
redo log



Слона нужно есть по кускам

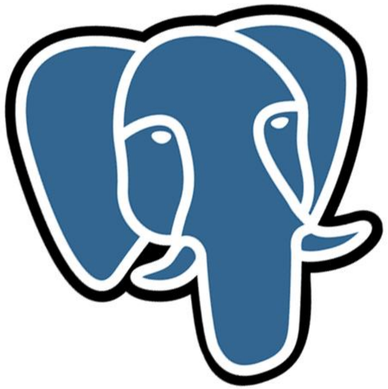
Проставляй границы транзакций

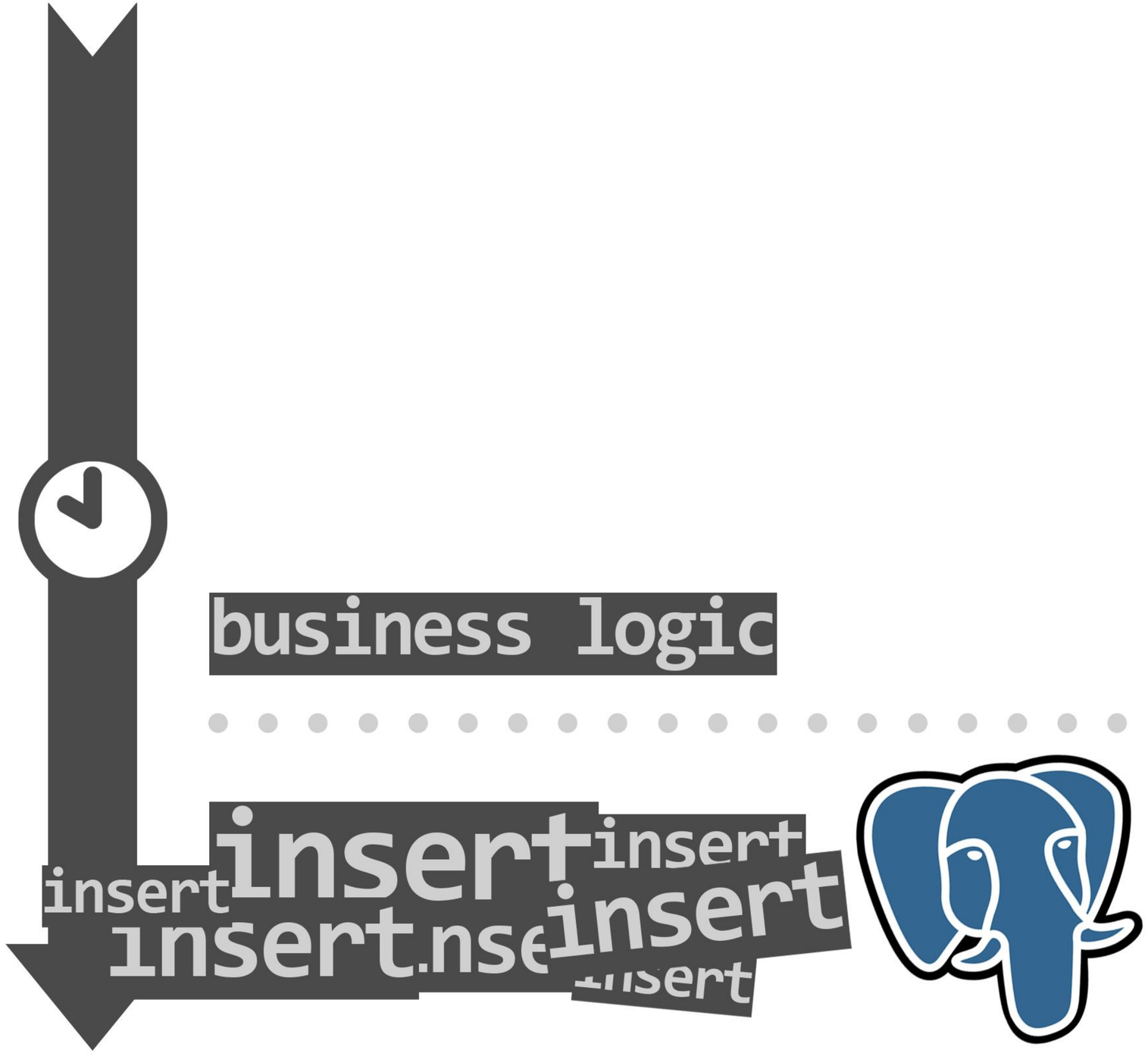
business logic

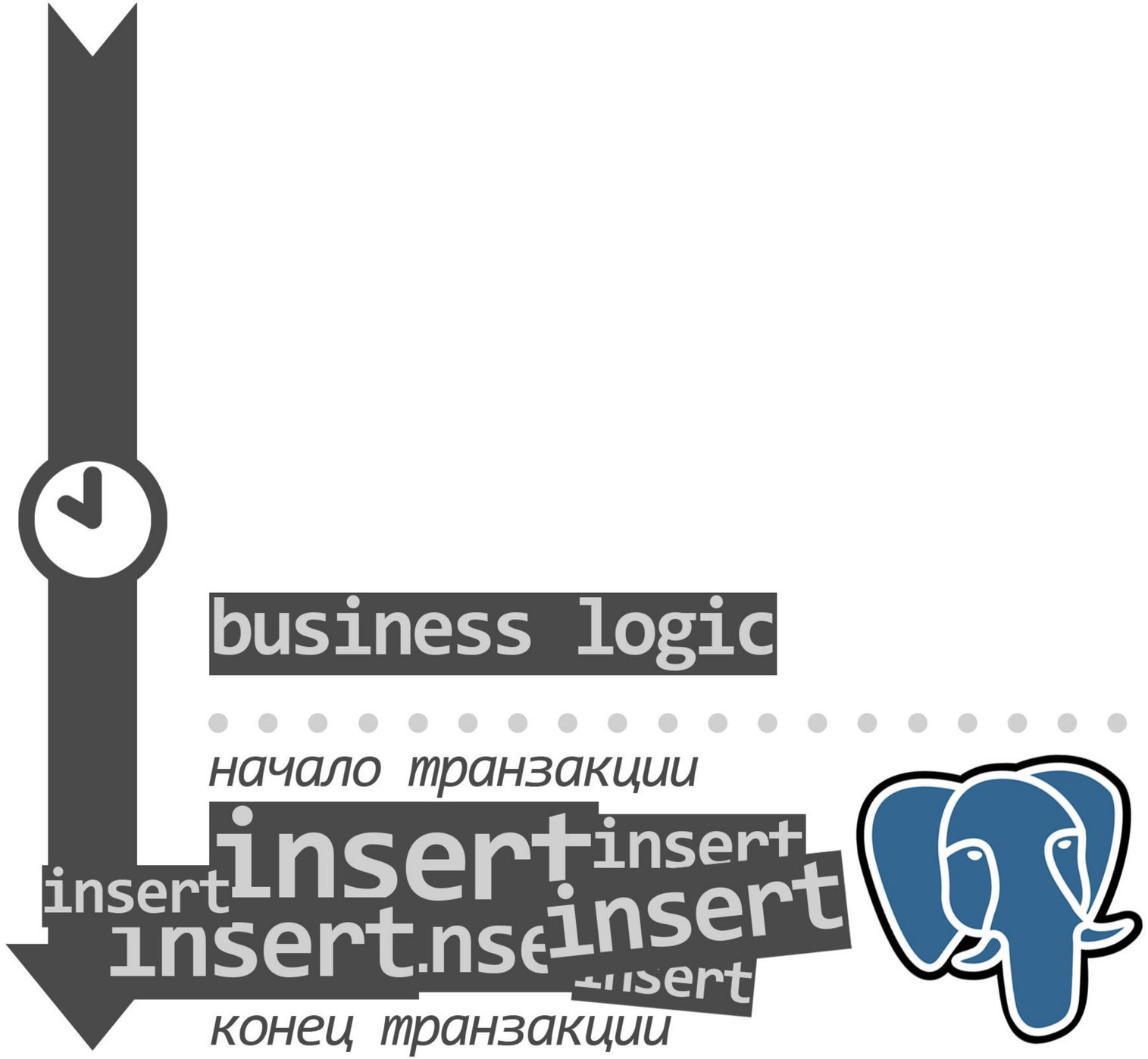
insert

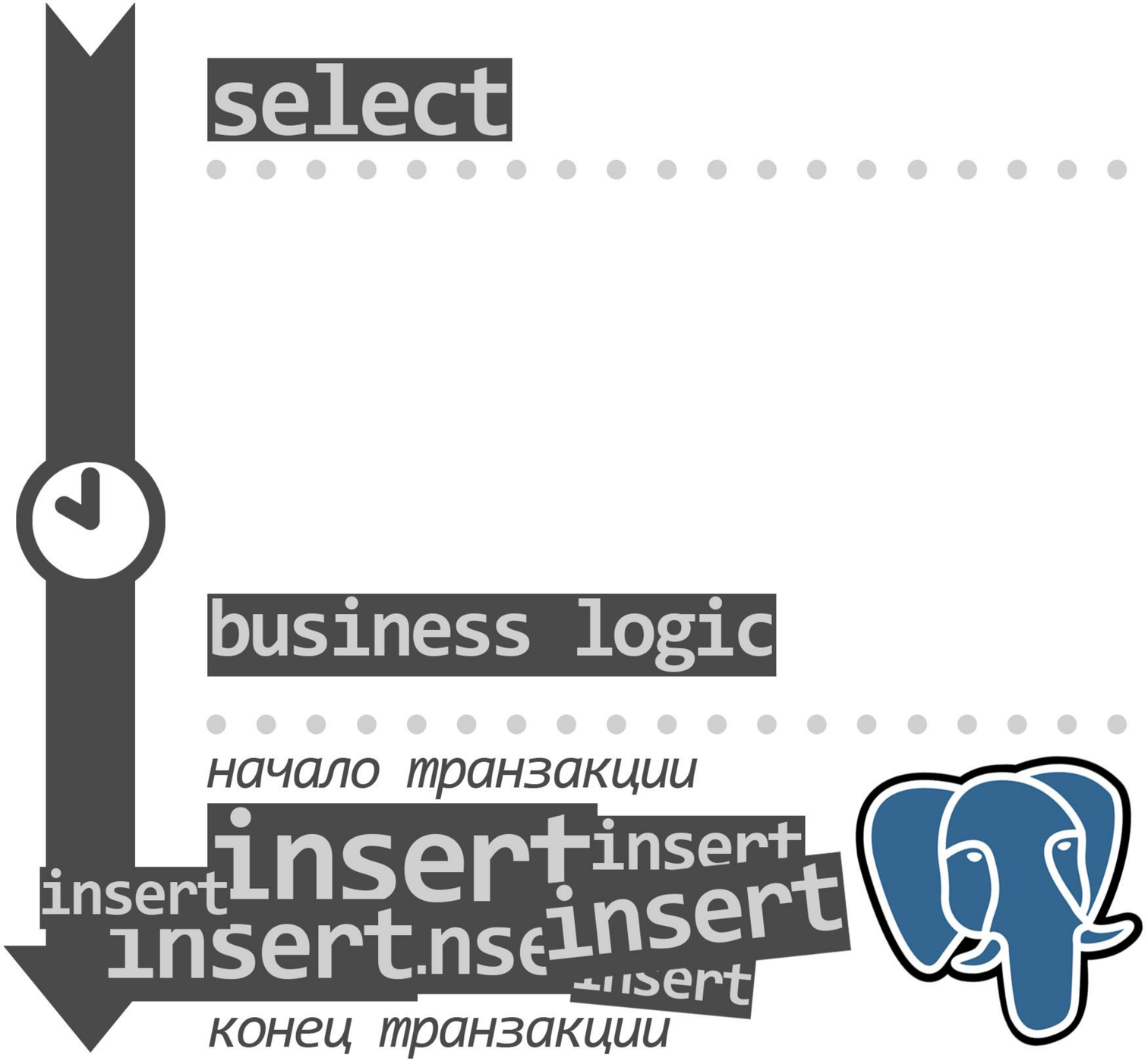
business logic

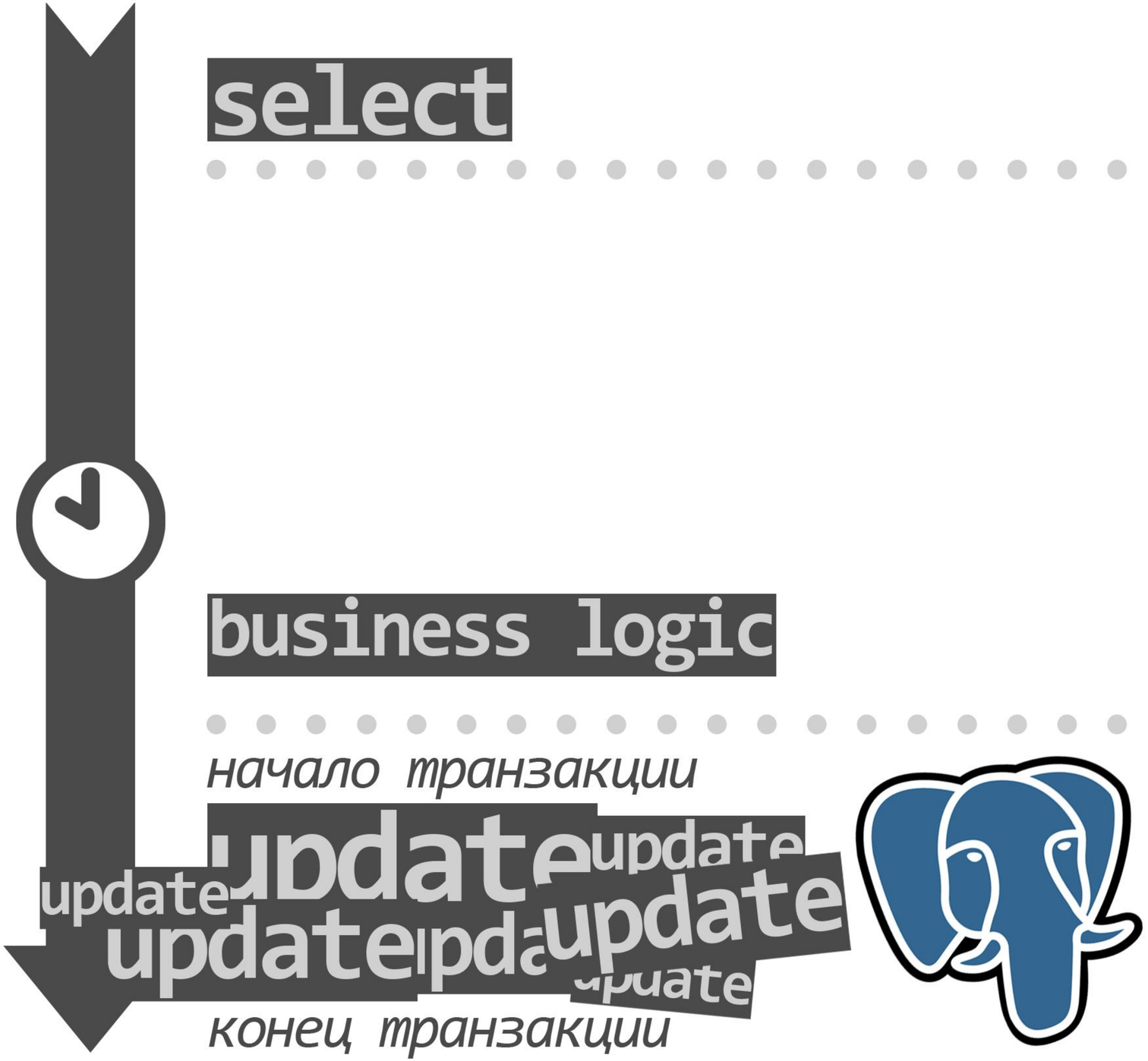
insert

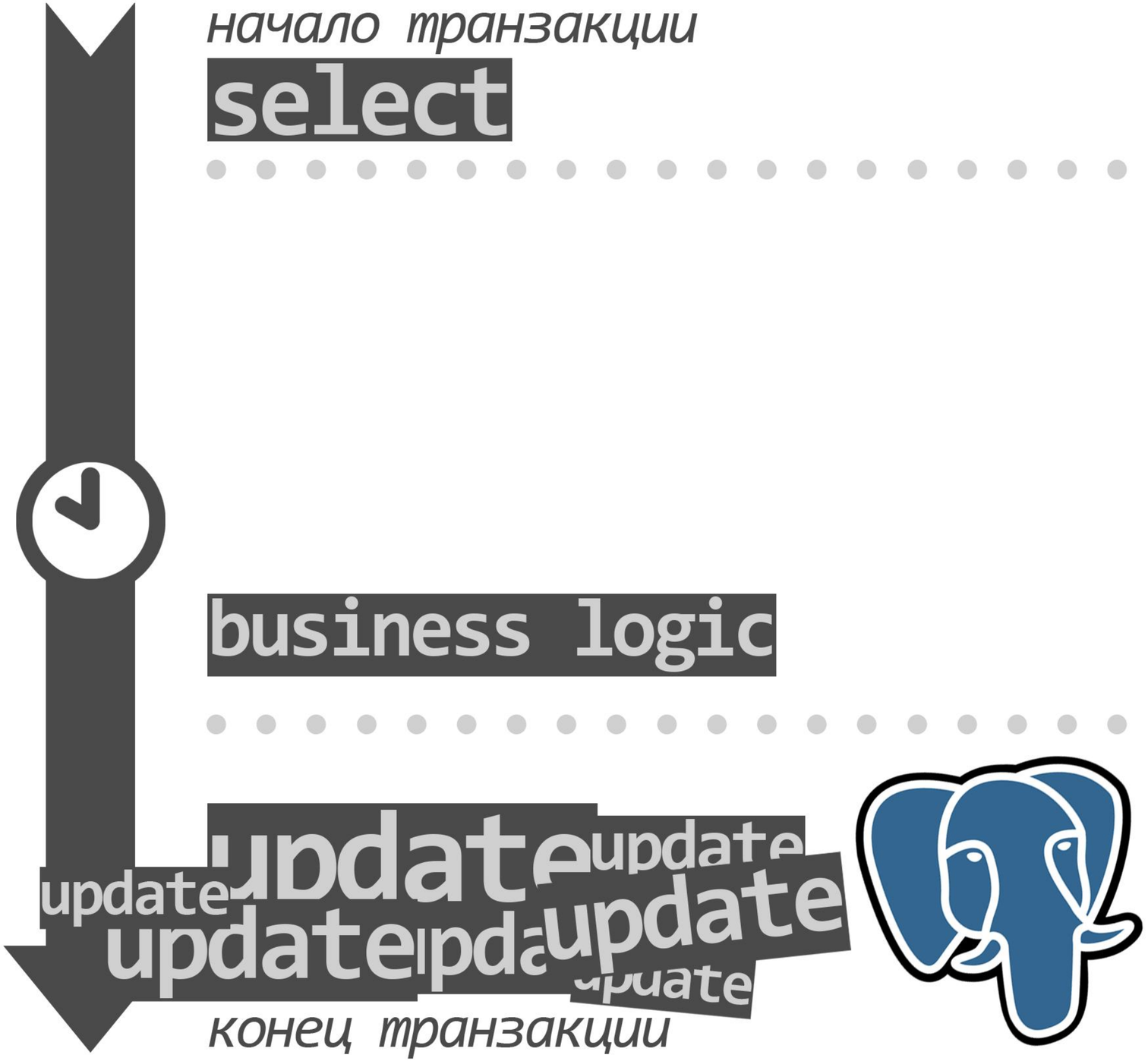


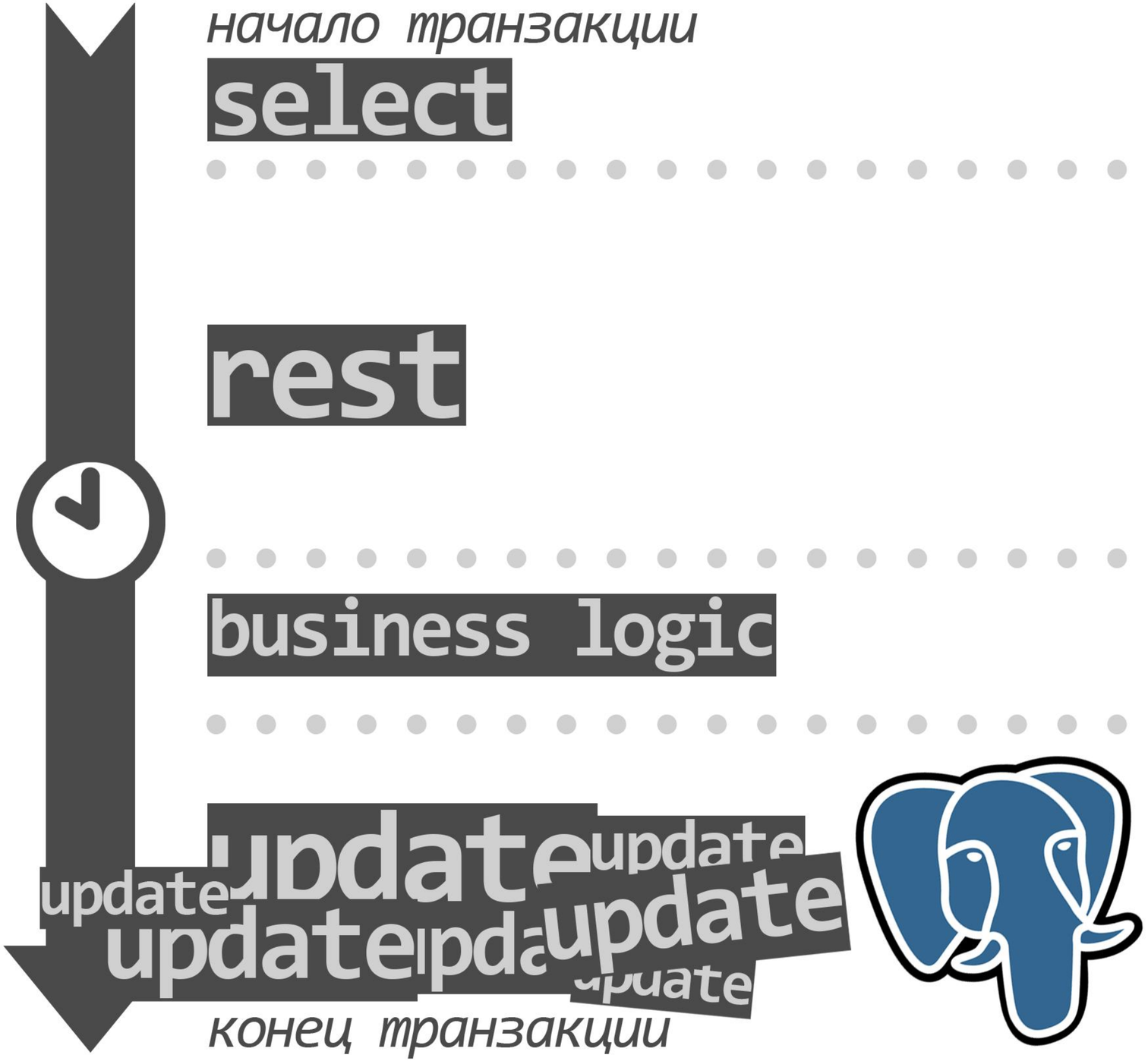

business logic
insert
insert
insert
insert
insert
insert


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
update
конец транзакции

начало транзакции

select

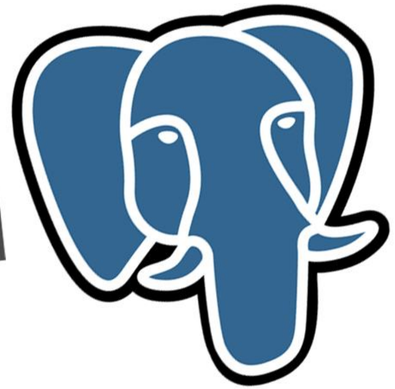
rest

очень долго

business logic

update update update update update update

конец транзакции



начало транзакции

select

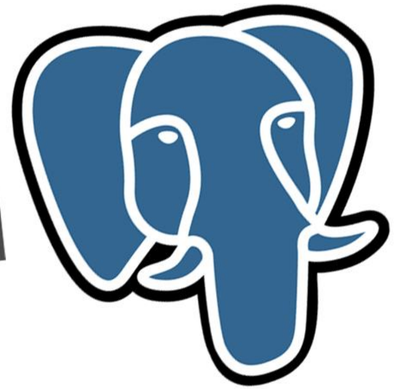
rest

очень долго

business logic

update update update update update update

конец транзакции



начало транзакции

select

rest

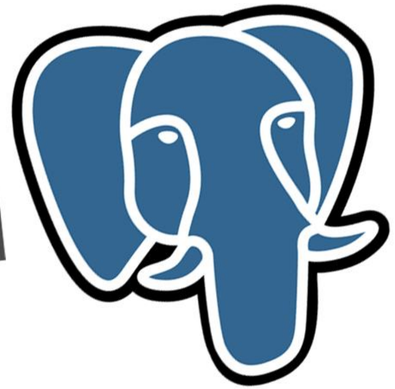
НУ И ЧТО?

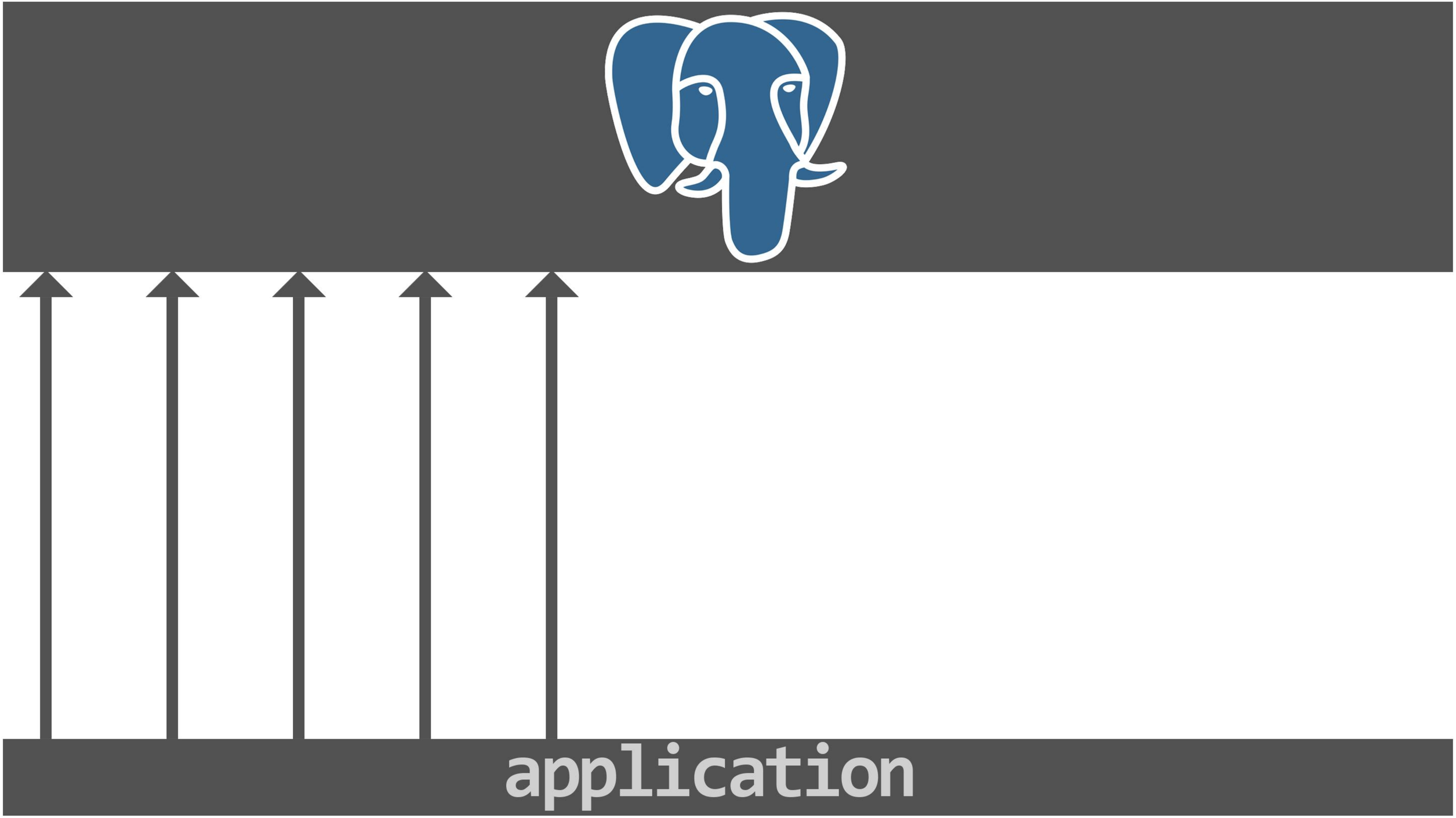
очень долго

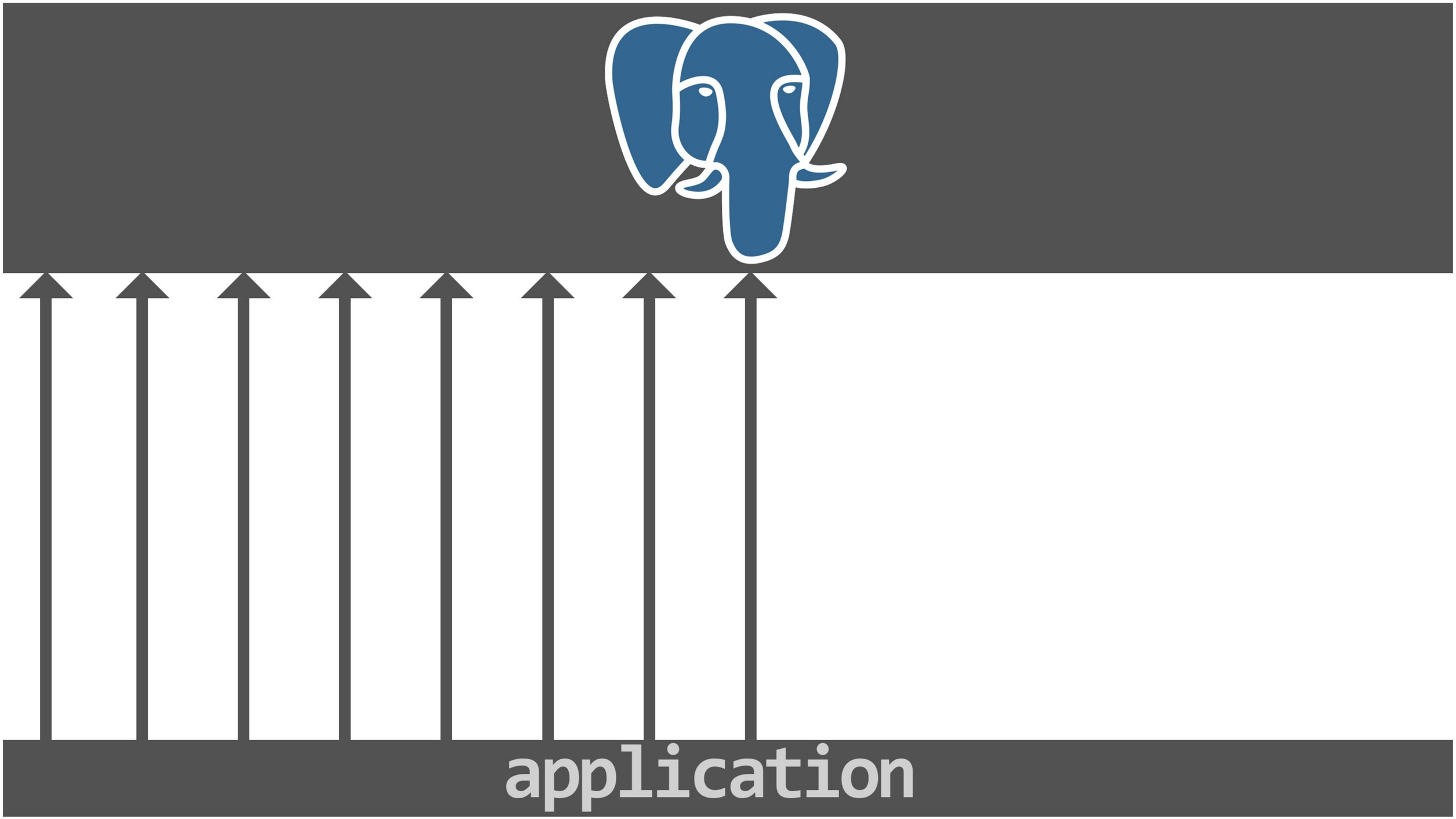
business logic

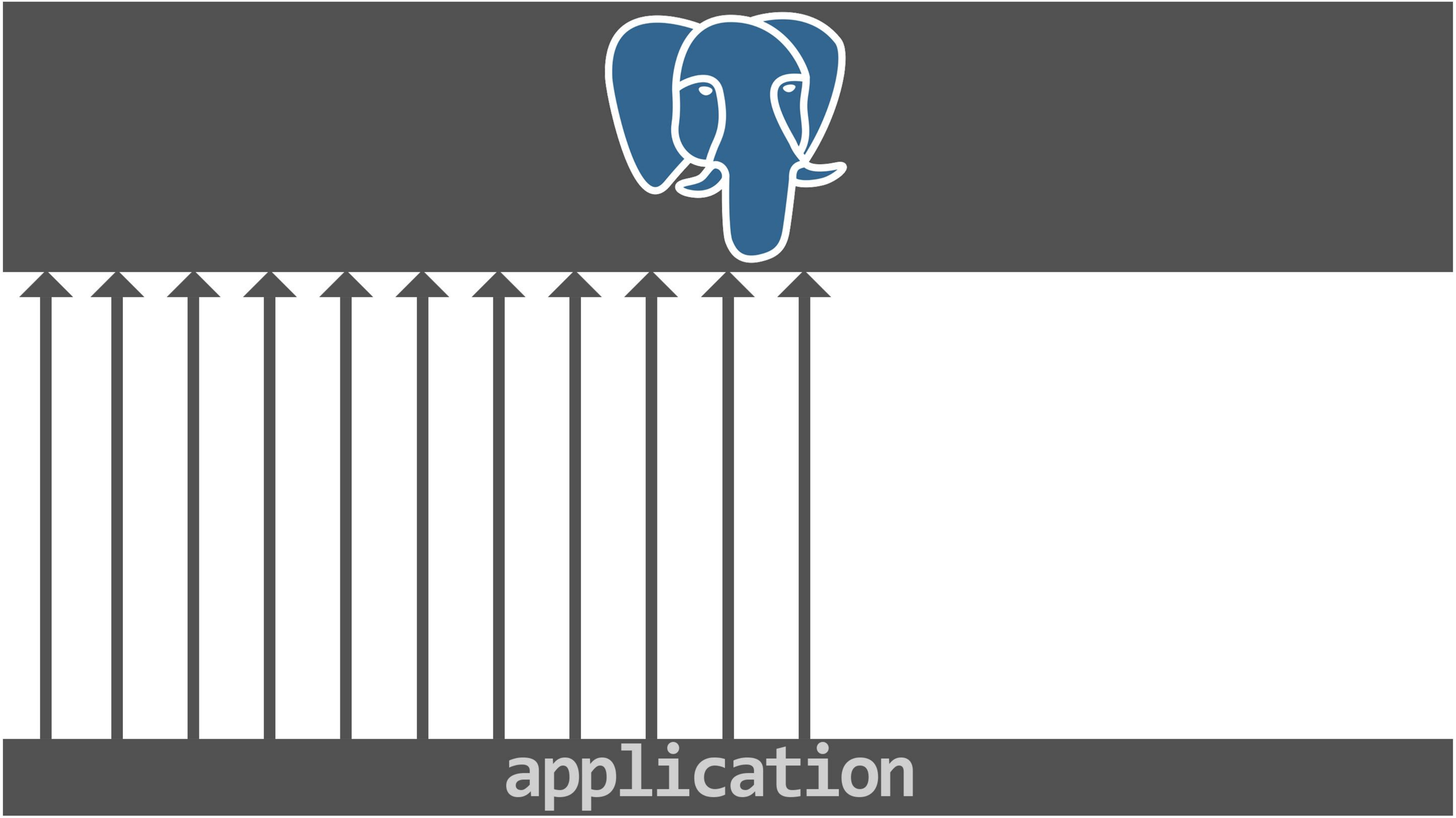
update update update update update update

конец транзакции

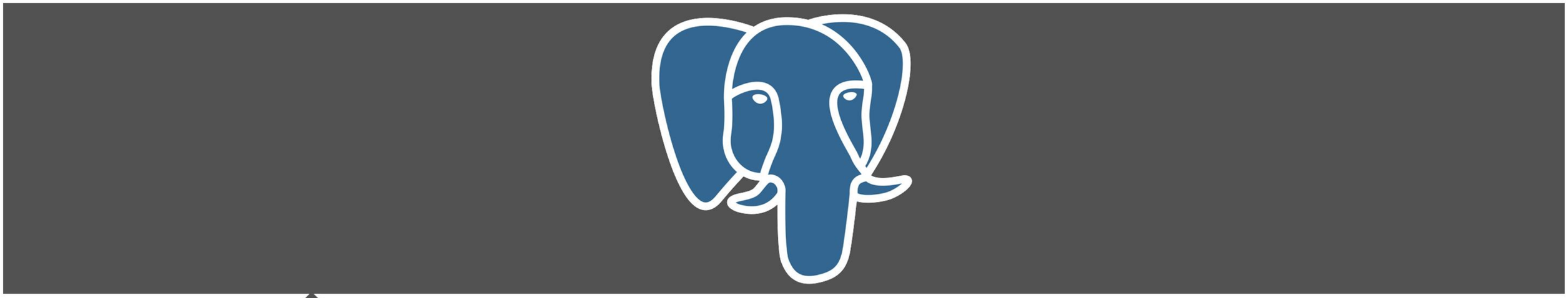



application


application


application

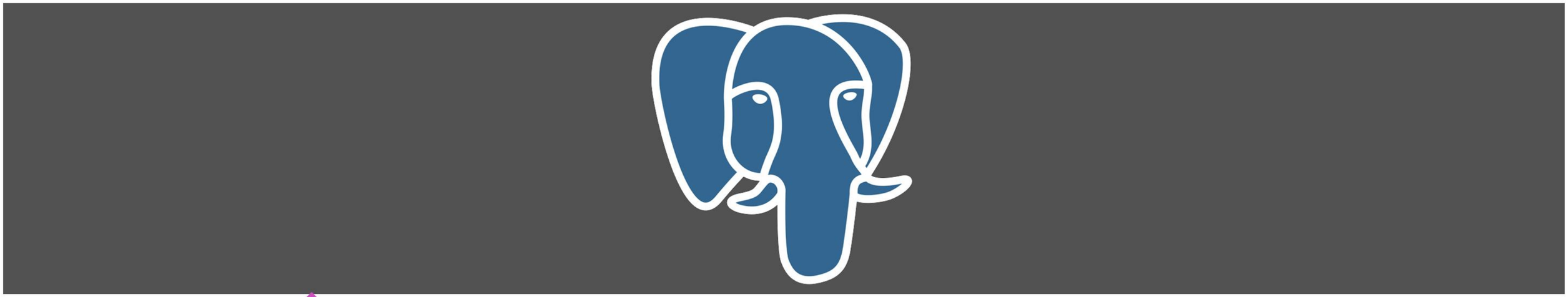
application



connection pool

запросы

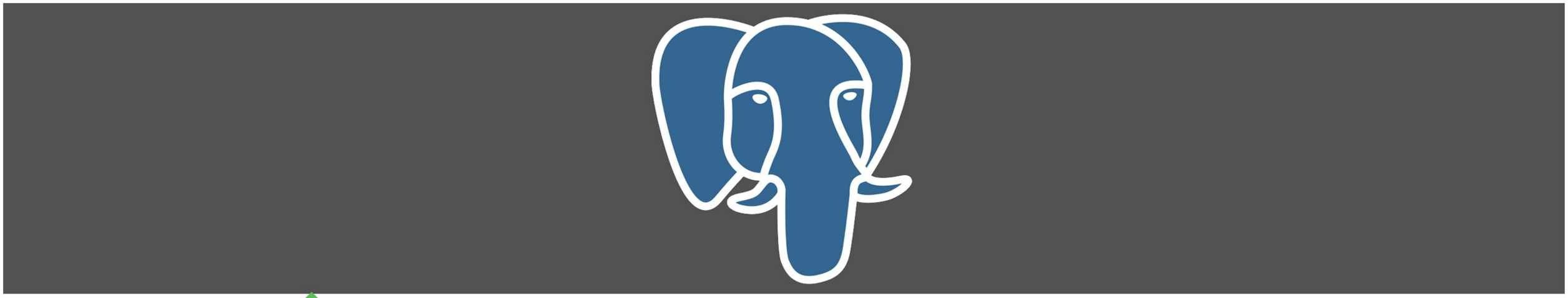
application



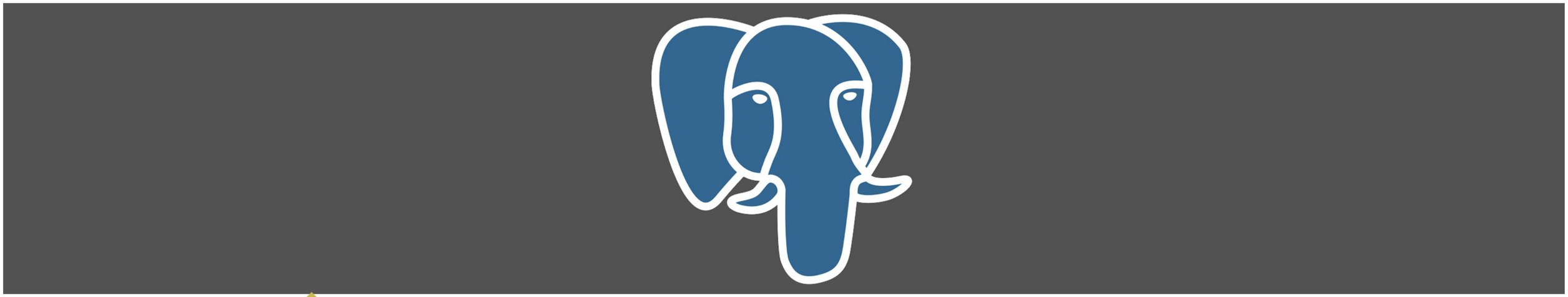
connection pool

запросы

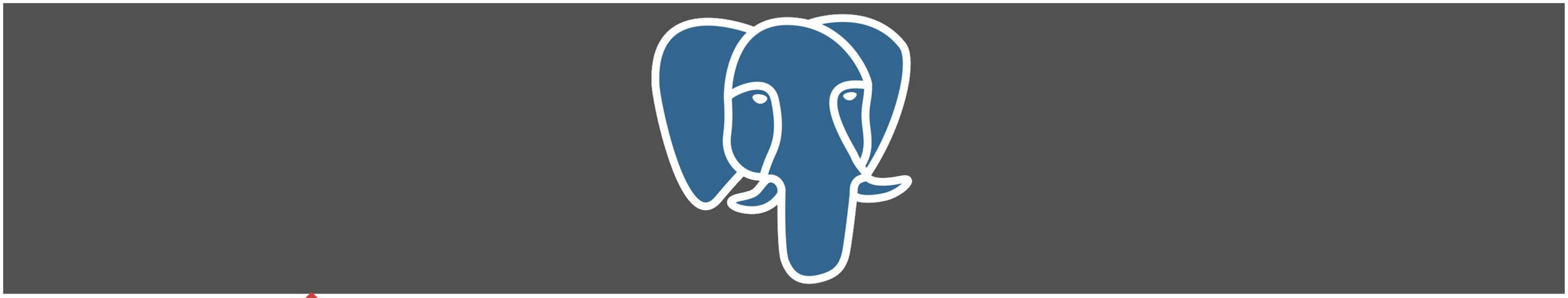
application




connection pool
запросы
application



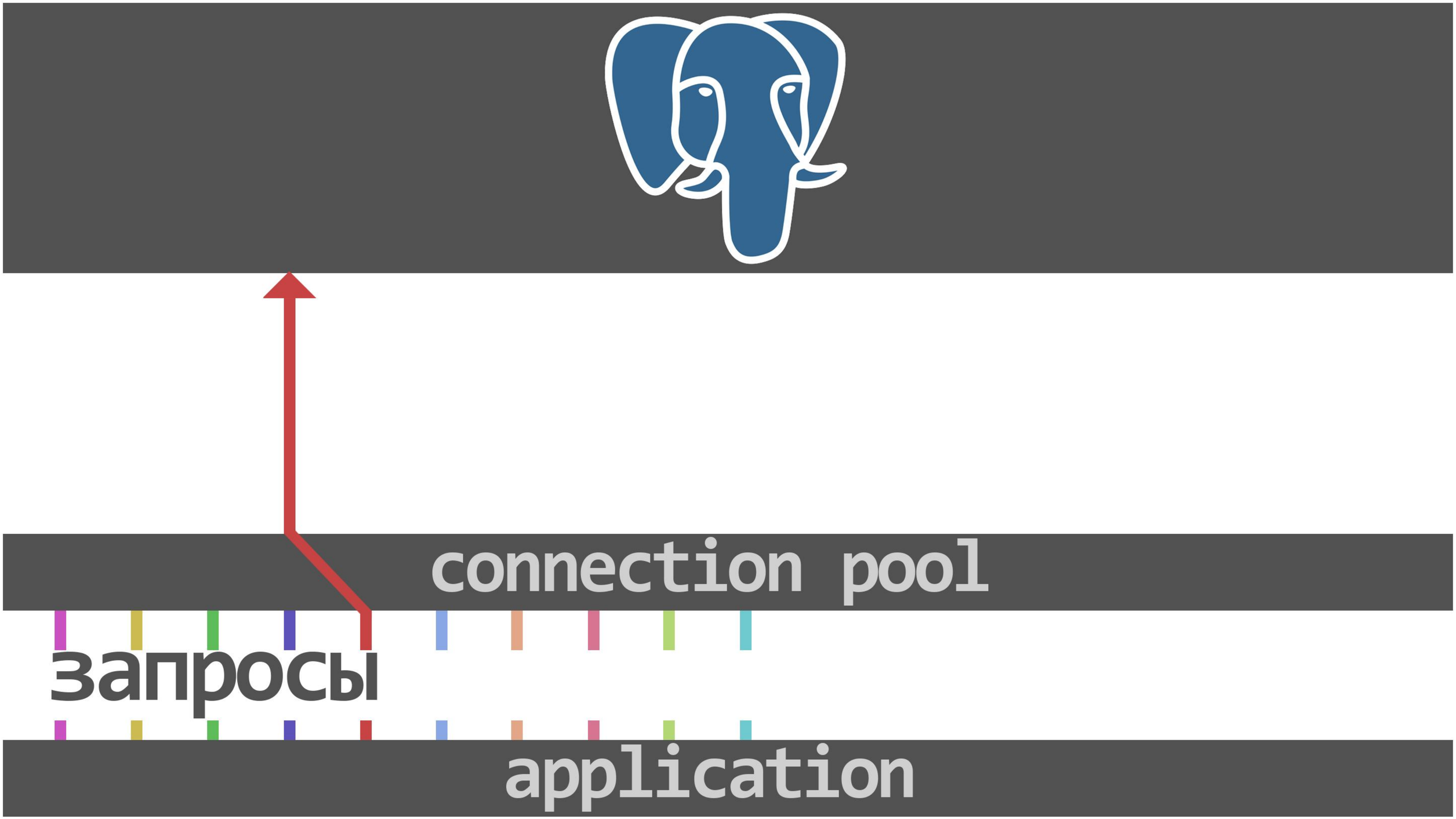

connection pool
запросы
application

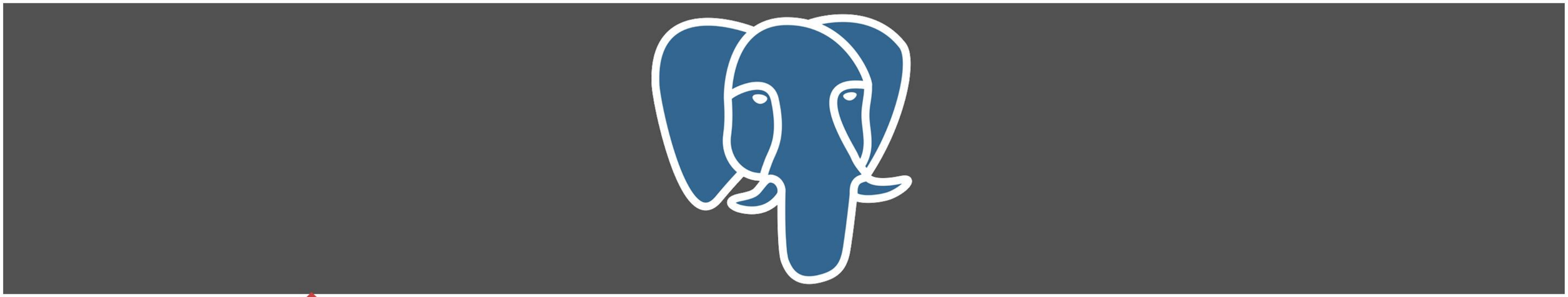


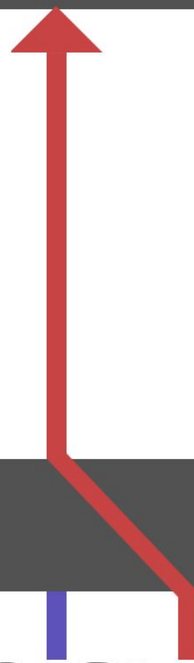
connection pool

запросы

application


connection pool
запросы
application

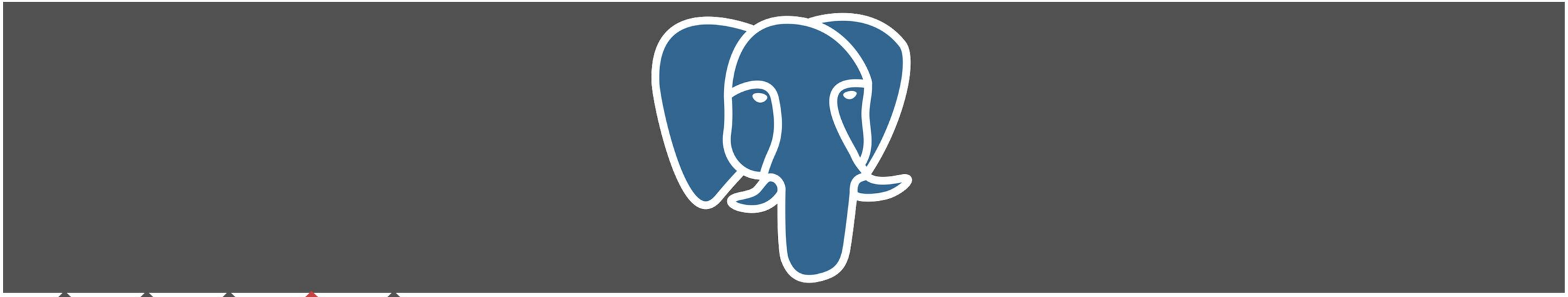


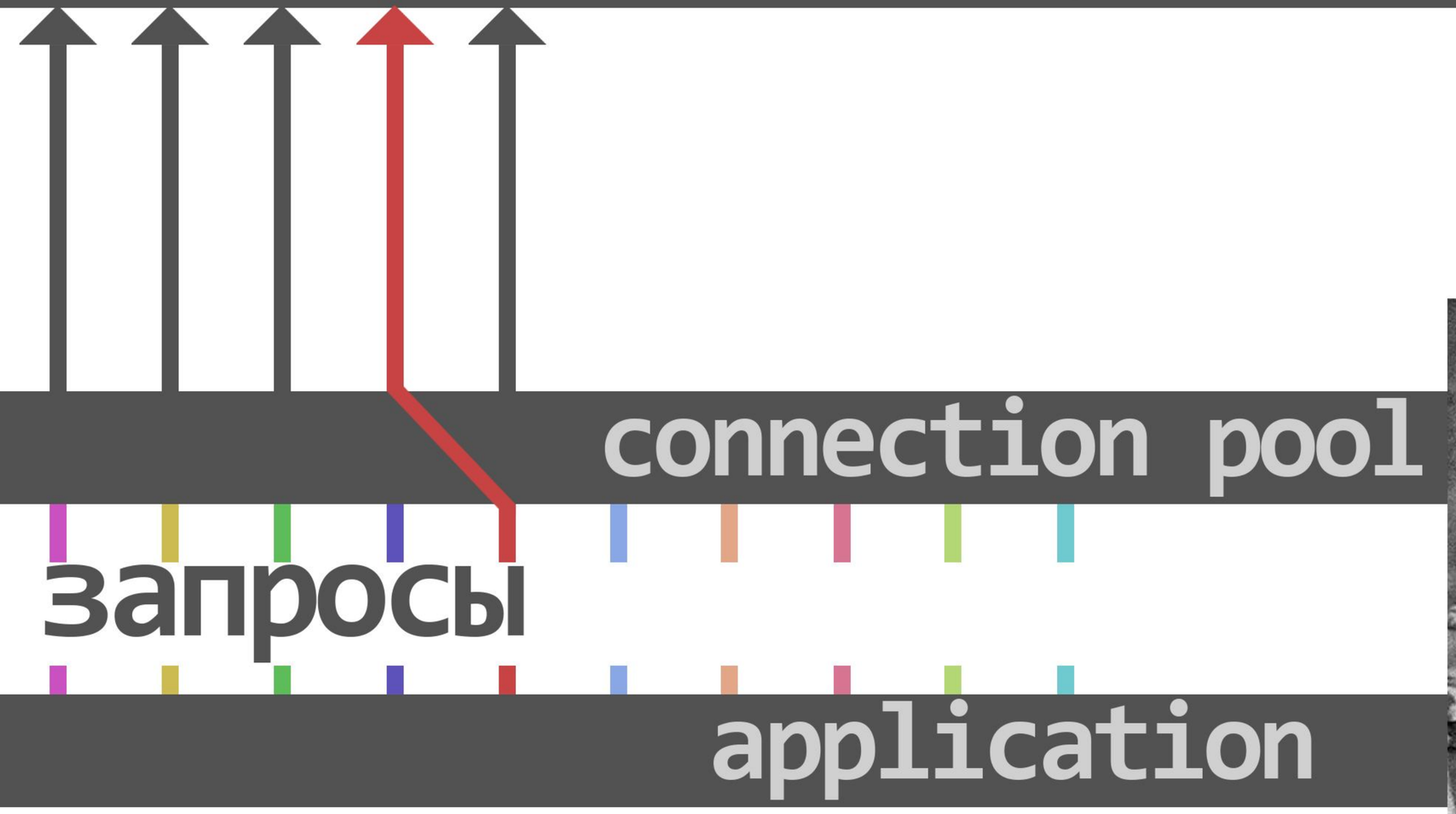


connection pool

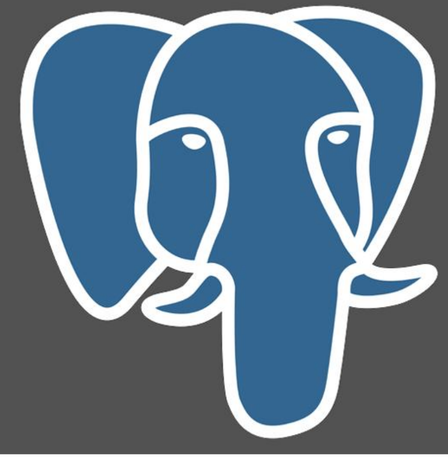
запросы

application




connection pool
запросы
application


РОГОВ и КОПЫТ
Закрыто
на обед



connection pool

запросы

application

начало транзакции

select

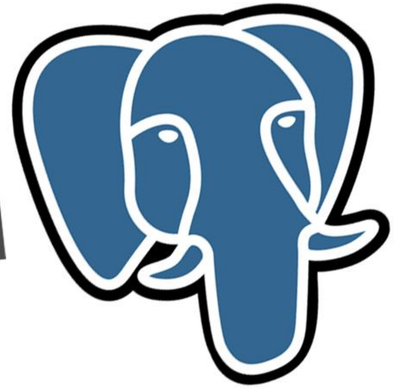
rest

очень долго

business logic

update update update update update update

конец транзакции



начало транзакции

select

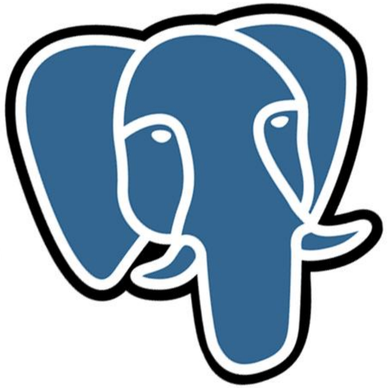
rest

очень долго

business logic

update update update update update

конец транзакции



очень долго
да и пофиг

rest

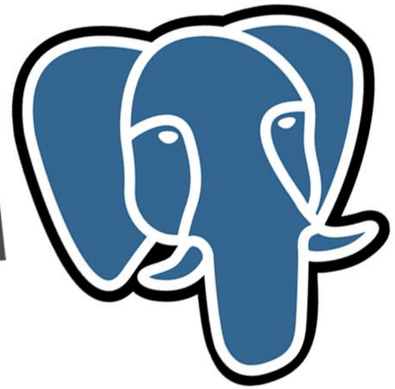
начало транзакции

select

business logic

update update update update update update

конец транзакции



начало транзакции

select

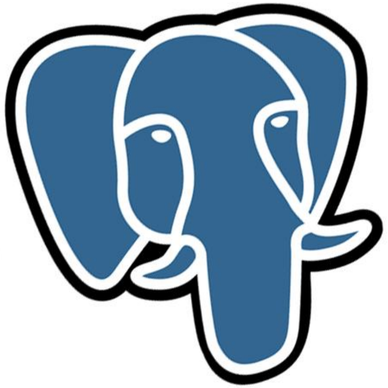
rest

очень долго

business logic

update update update update update

конец транзакции



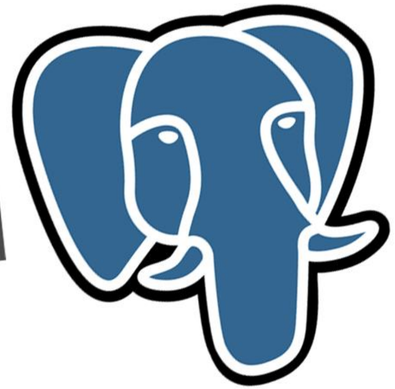
начало транзакции

select

rest

business logic

update update update update update update



конец транзакции

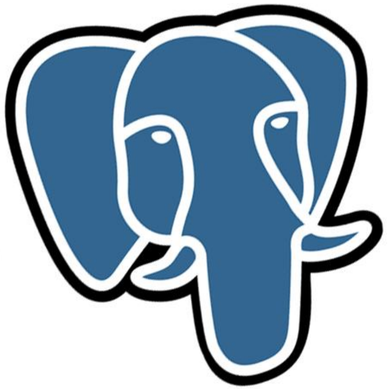
начало транзакции

select for update

rest

business logic

update update update update update update



конец транзакции

select

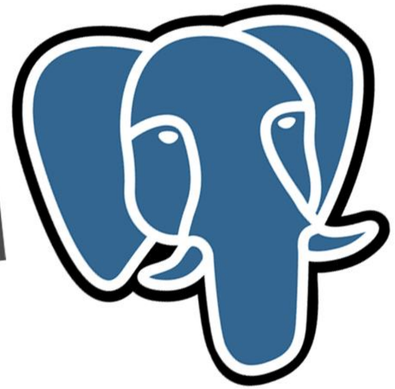
rest

business logic

начало транзакции

update update update update update update

конец транзакции

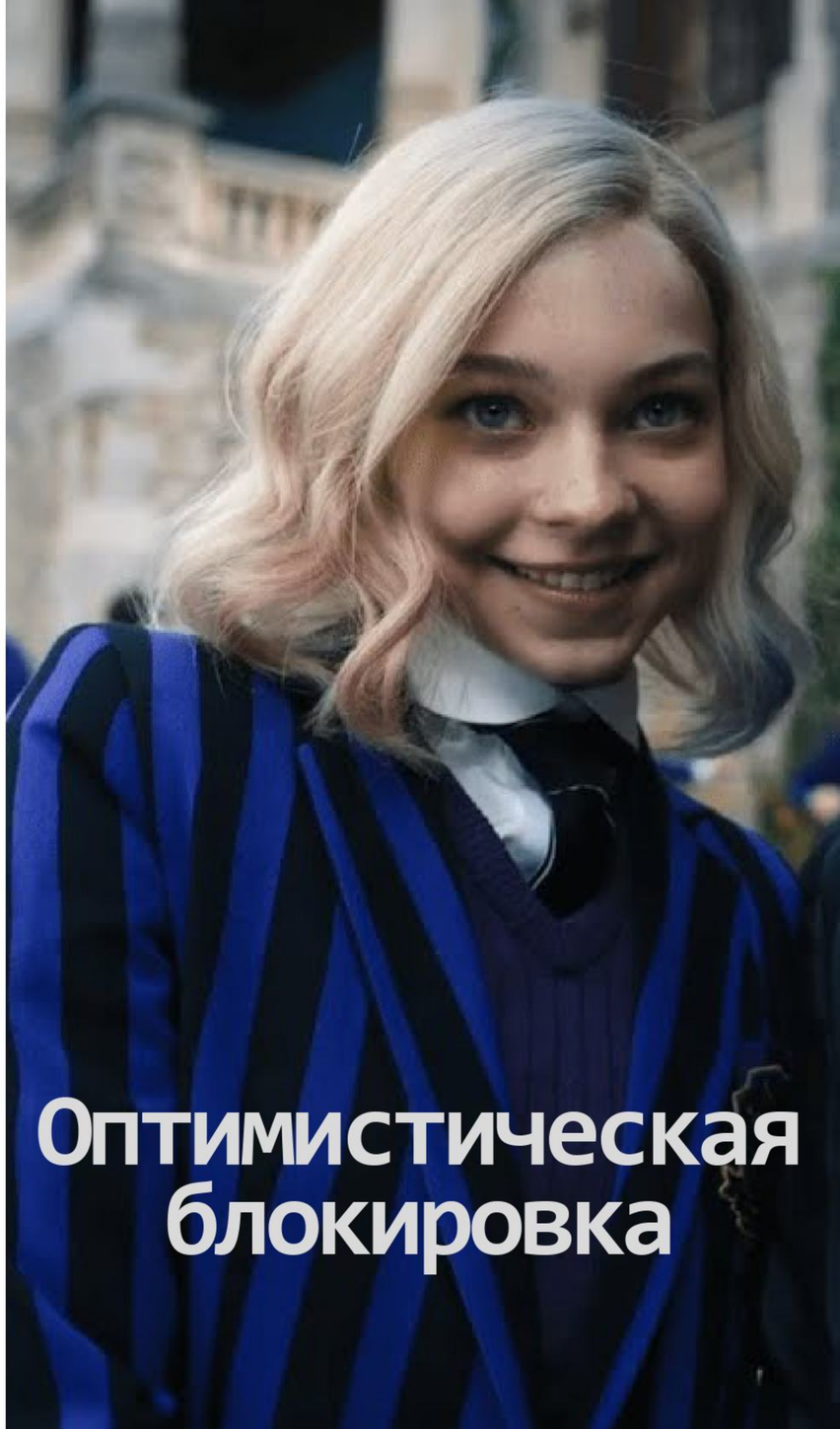


books

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
```

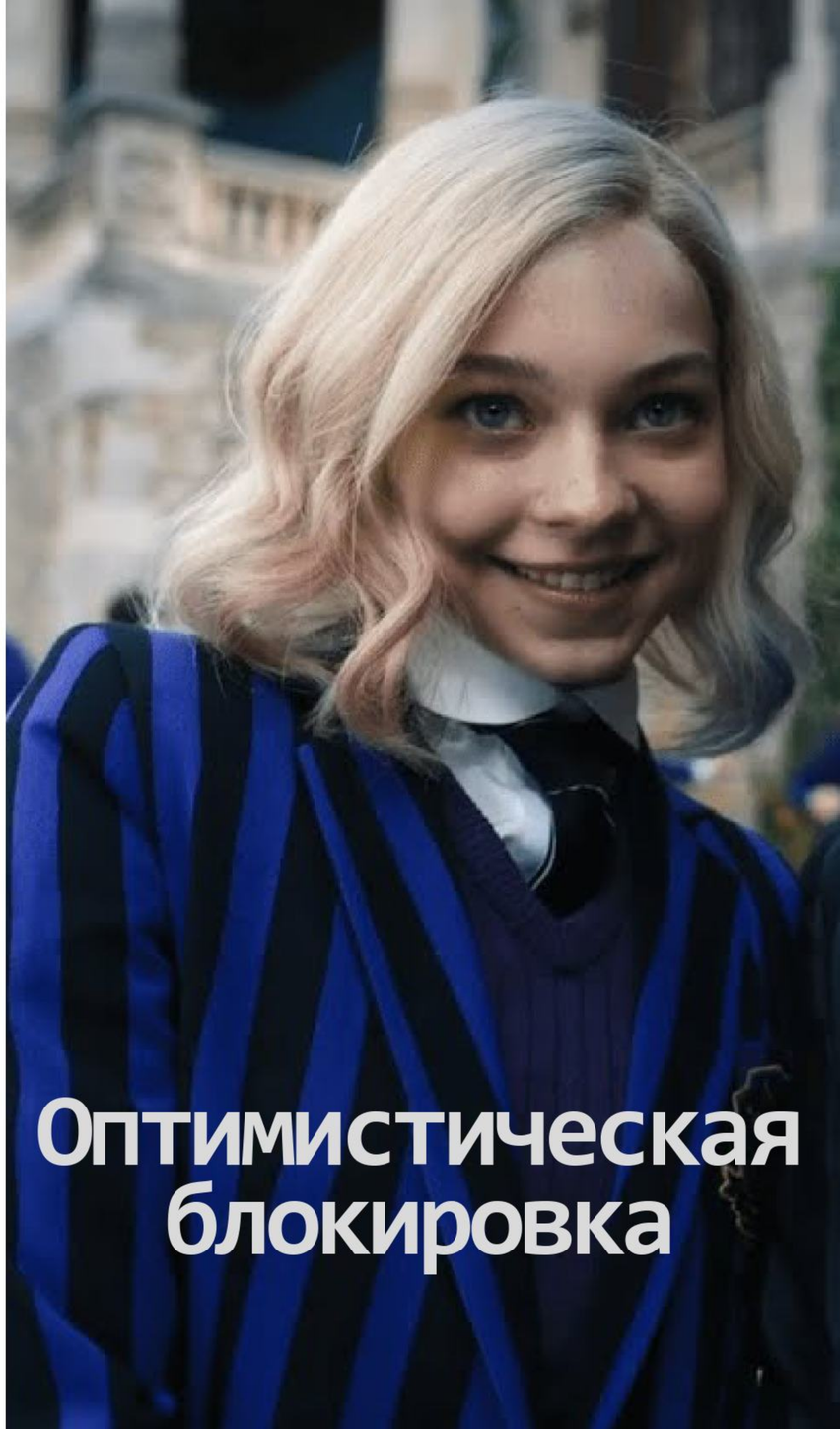


books

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
```

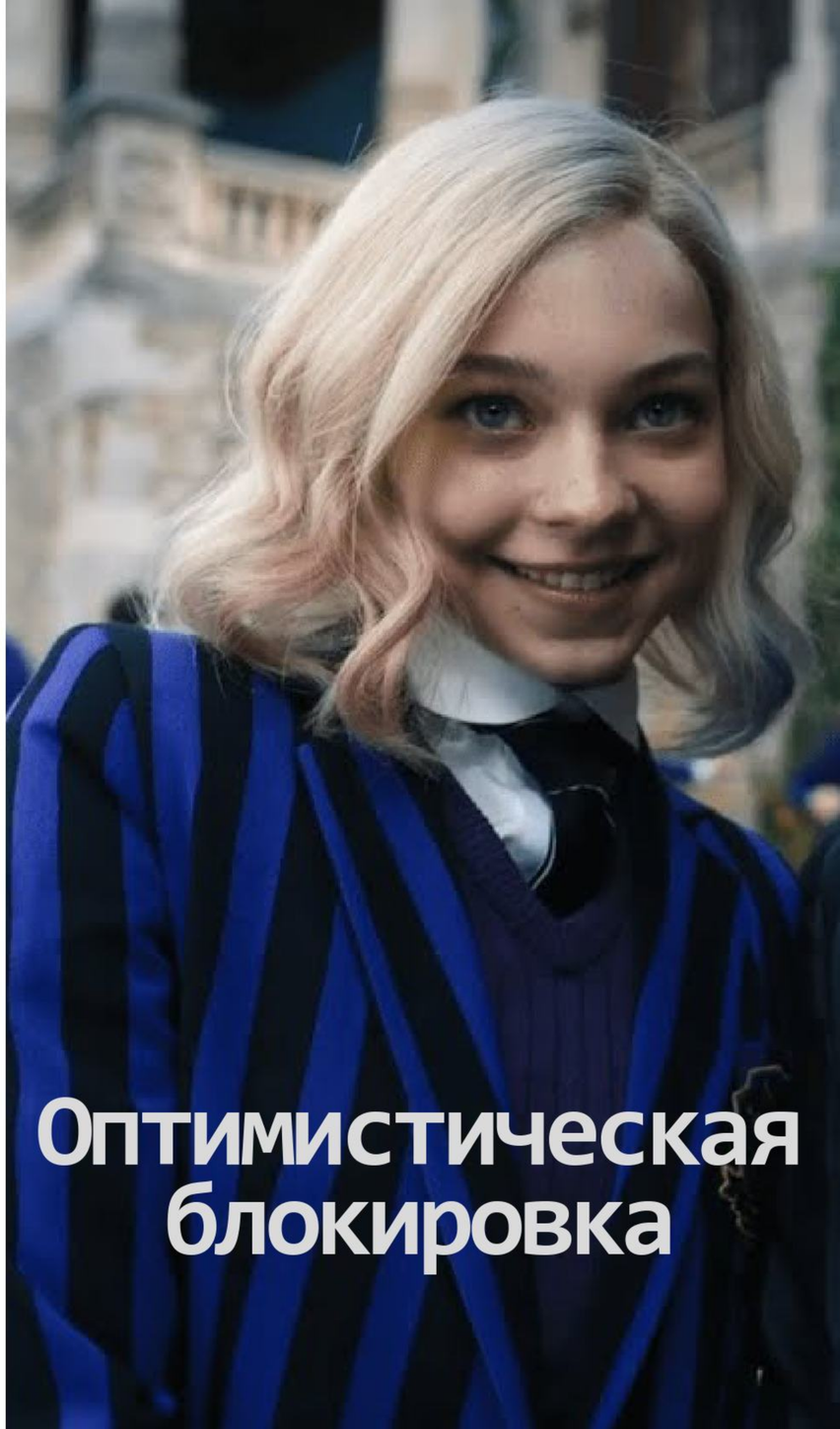


books

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
```

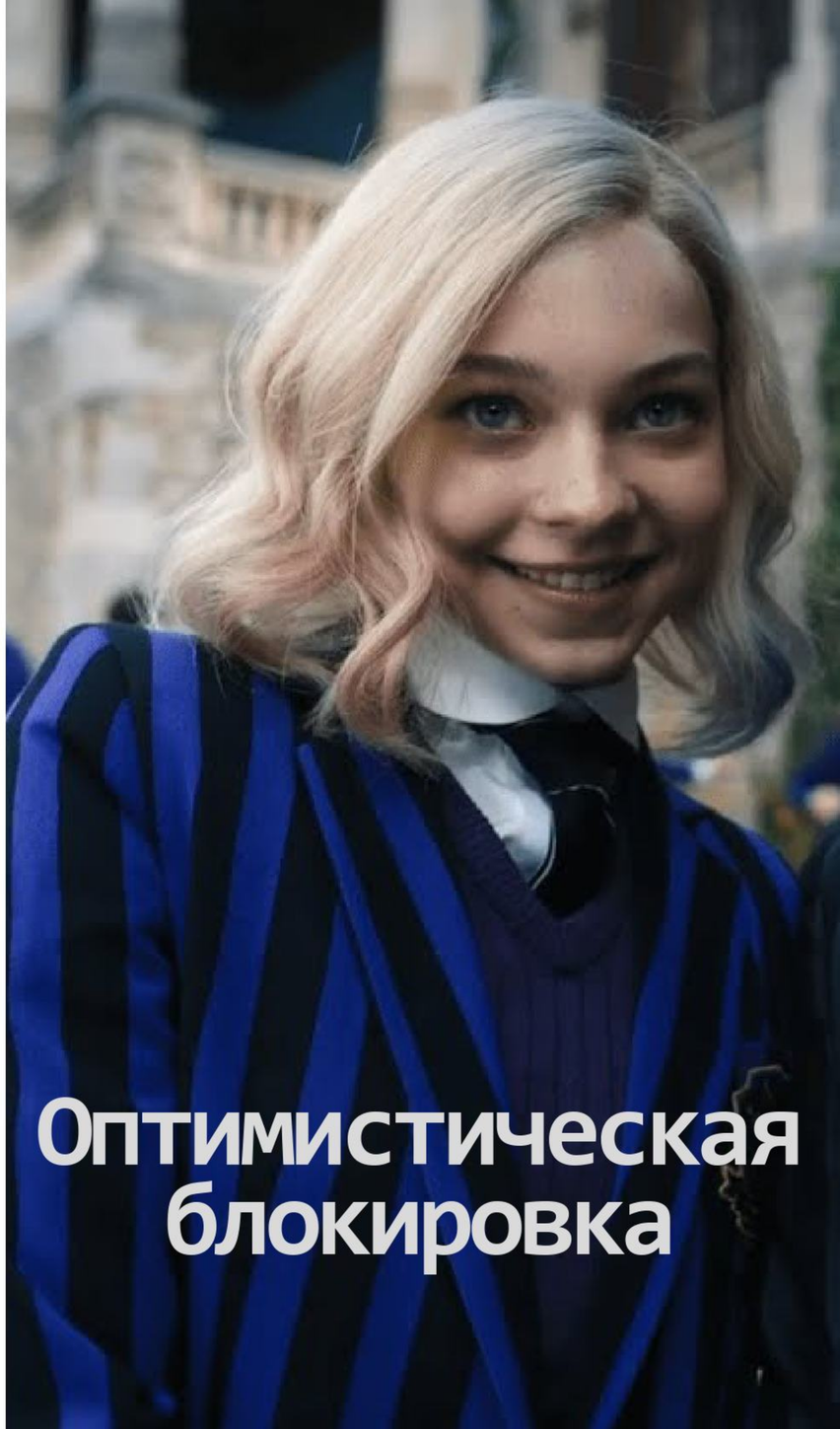


books

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

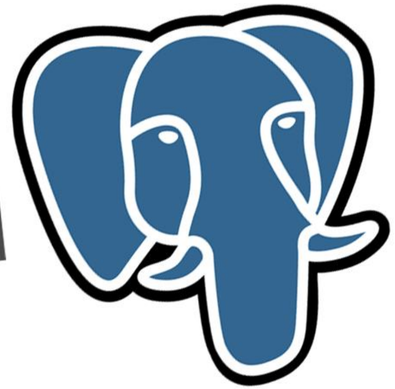
rest

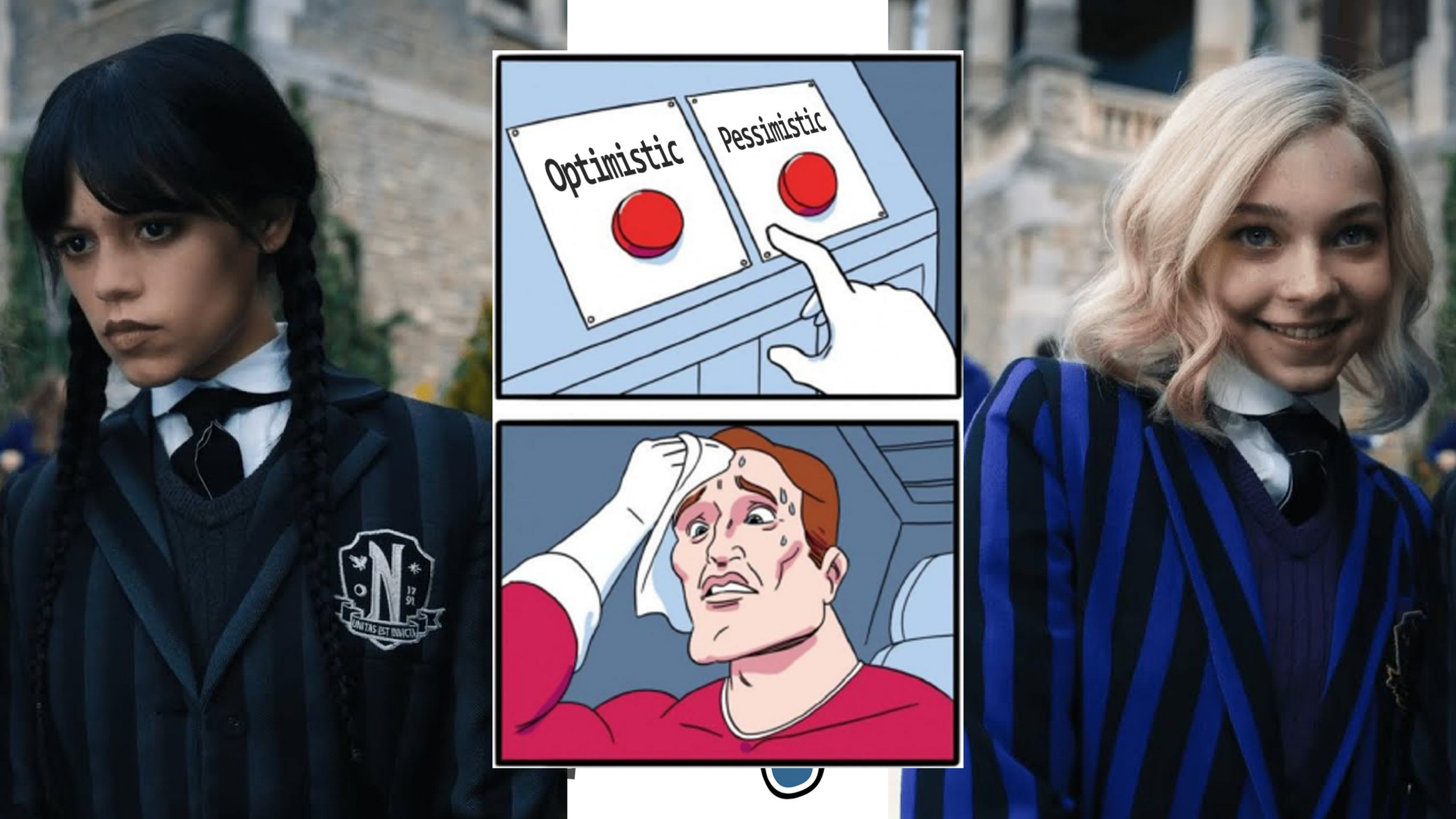
business logic

начало транзакции

update update update update update update

конец транзакции




Optimistic
Pessimistic

select

rest

business logic

начало транзакции

update update update update update update

конец транзакции

открытие коннекшона

select

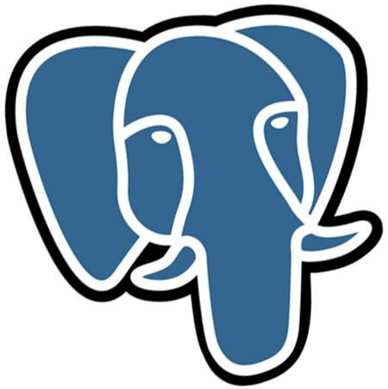
rest

business logic

начало транзакции

update update update update update

конец транзакции



OSIV

открытие коннекшона

select

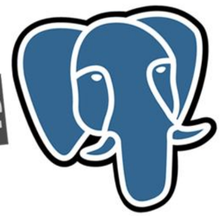
rest

business logic

начало транзакции

update update update update

конец транзакции



закрытие коннекшона

открытие коннекшона

select

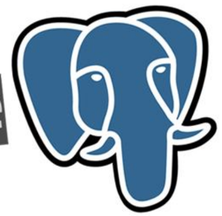
rest

business logic

начало транзакции

update update update update

конец транзакции



рендеринг веб страницы | сериализация json

закрытие коннекшона

открытие коннекшона

select

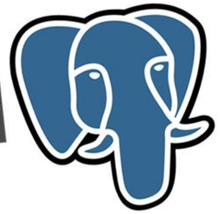
rest

business logic

начало транзакции

update update update update

конец транзакции



рендеринг веб страницы | сериализация json

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

закрытие коннекшона

OSIV

~~открытие коннекшена~~

select

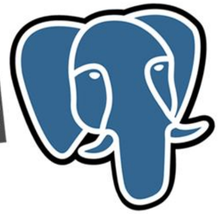
rest

business logic

начало транзакции

update update update update

конец транзакции



рендеринг веб страницы | сериализация json

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

~~закрытие коннекшена~~

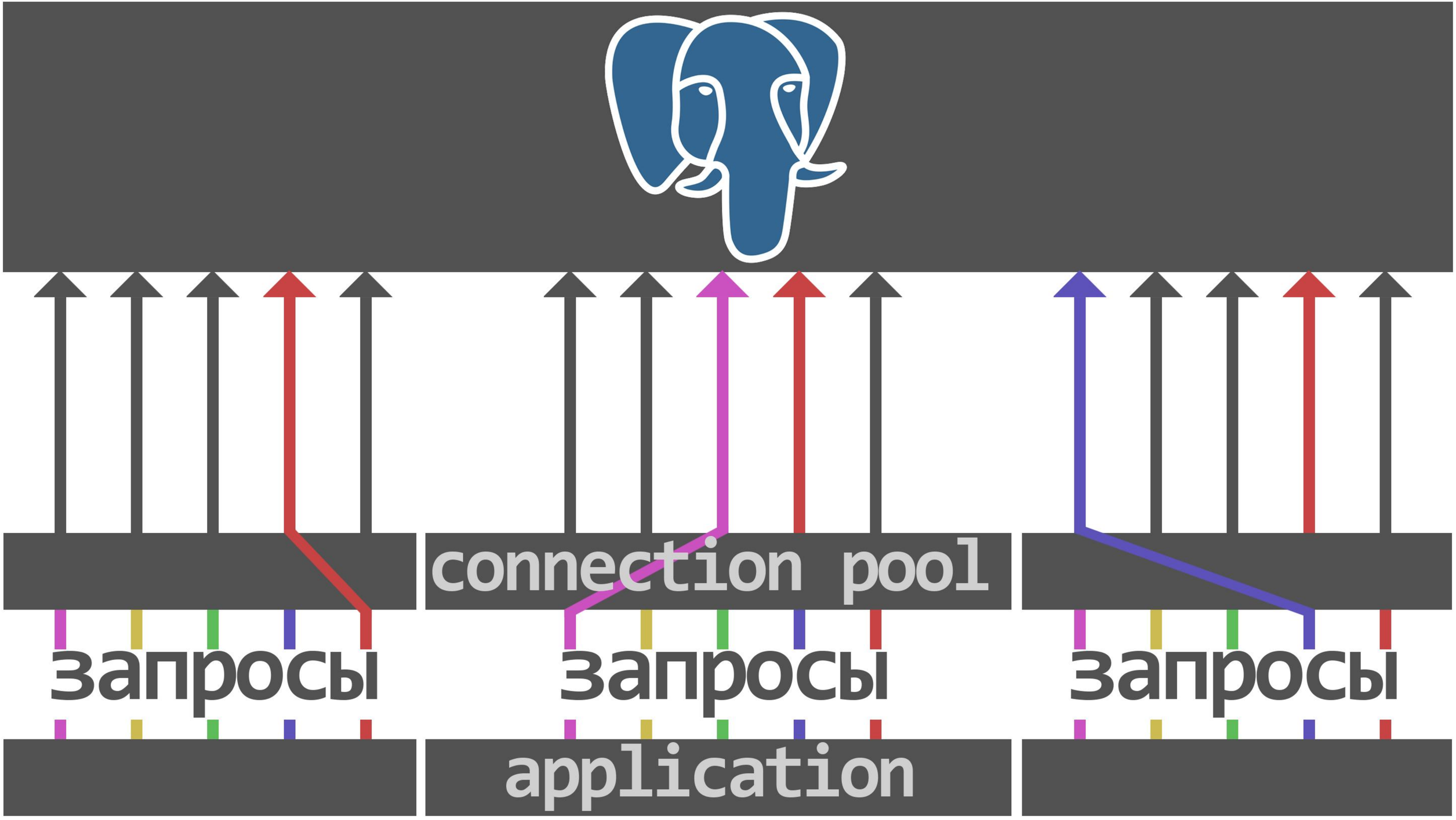

Да не...
Дефолты лучше не трогать...

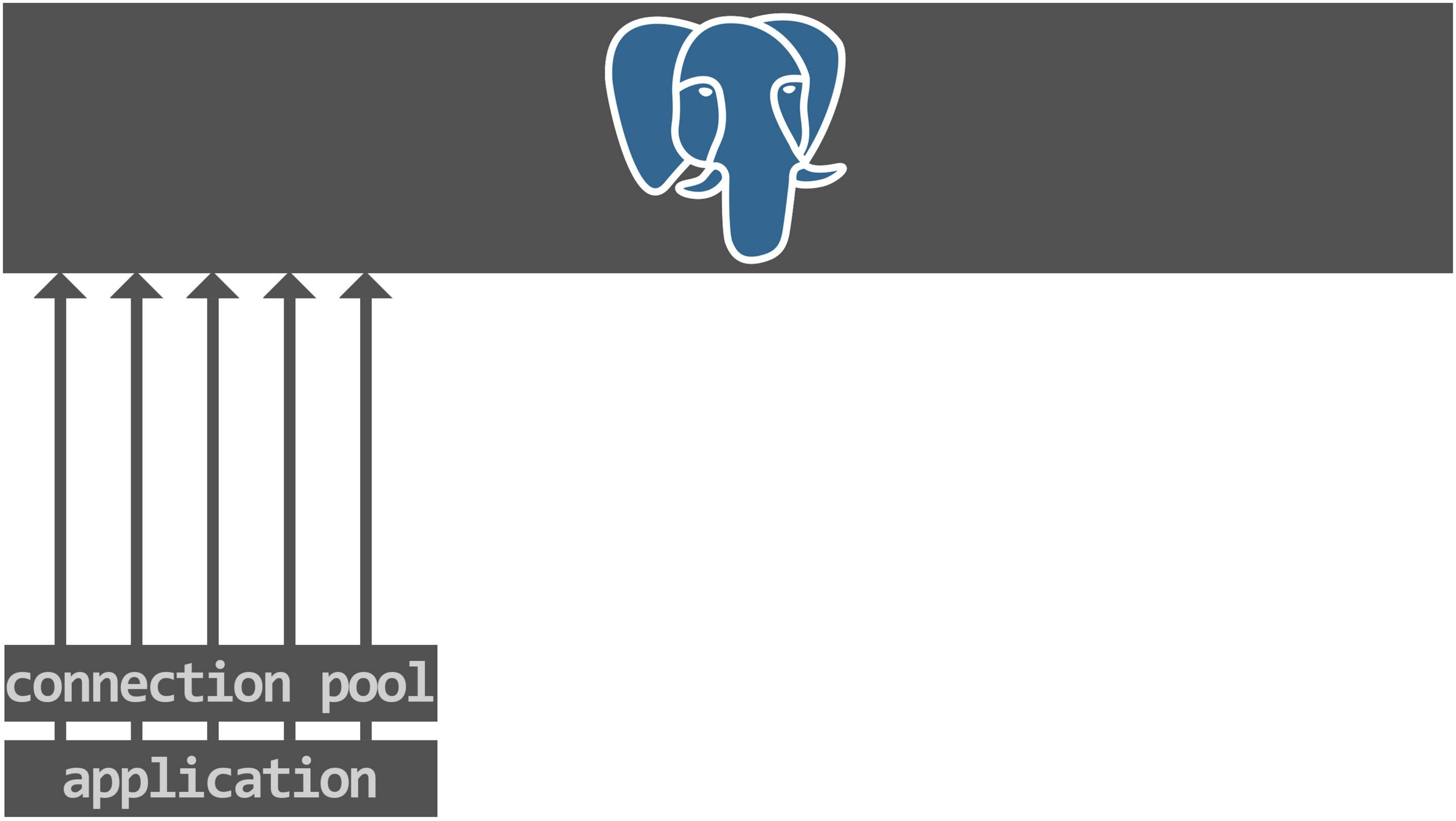
JPoint

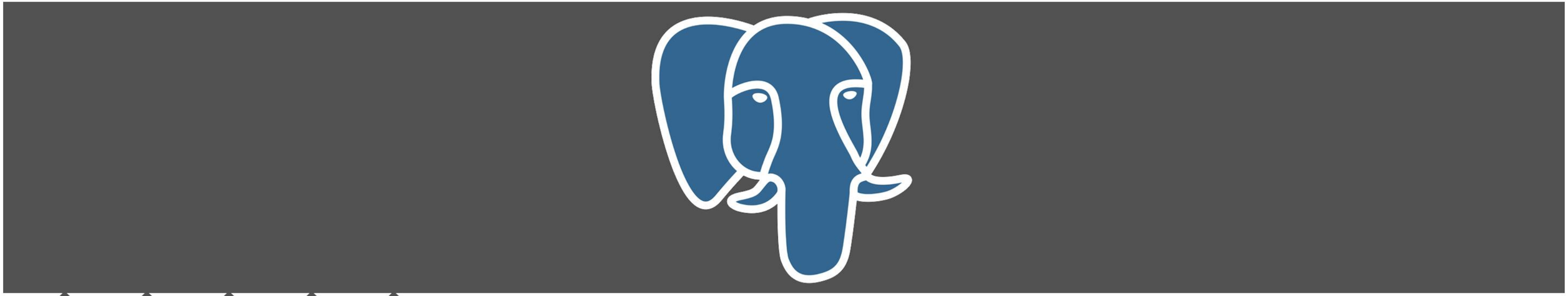
PG для Java-разработчиков

Сергей Вальков

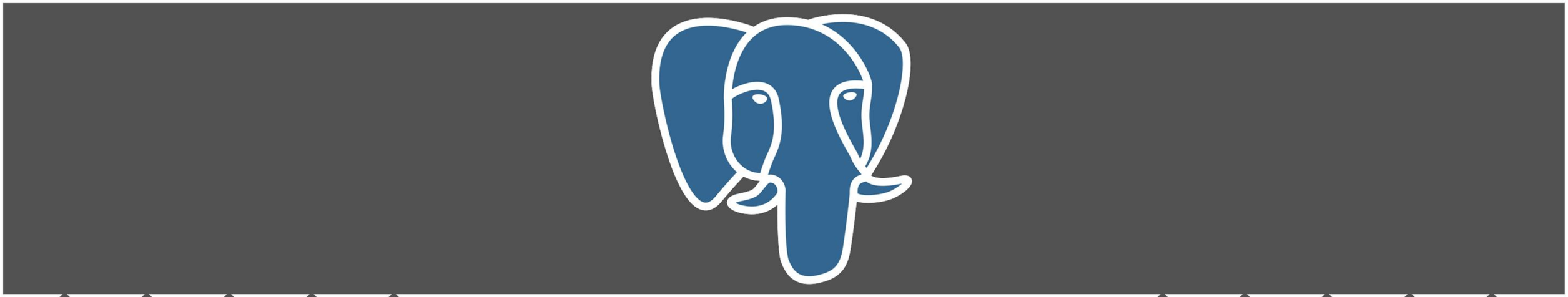
Яндекс Маркет


connection pool
запросы
запросы
запросы
application

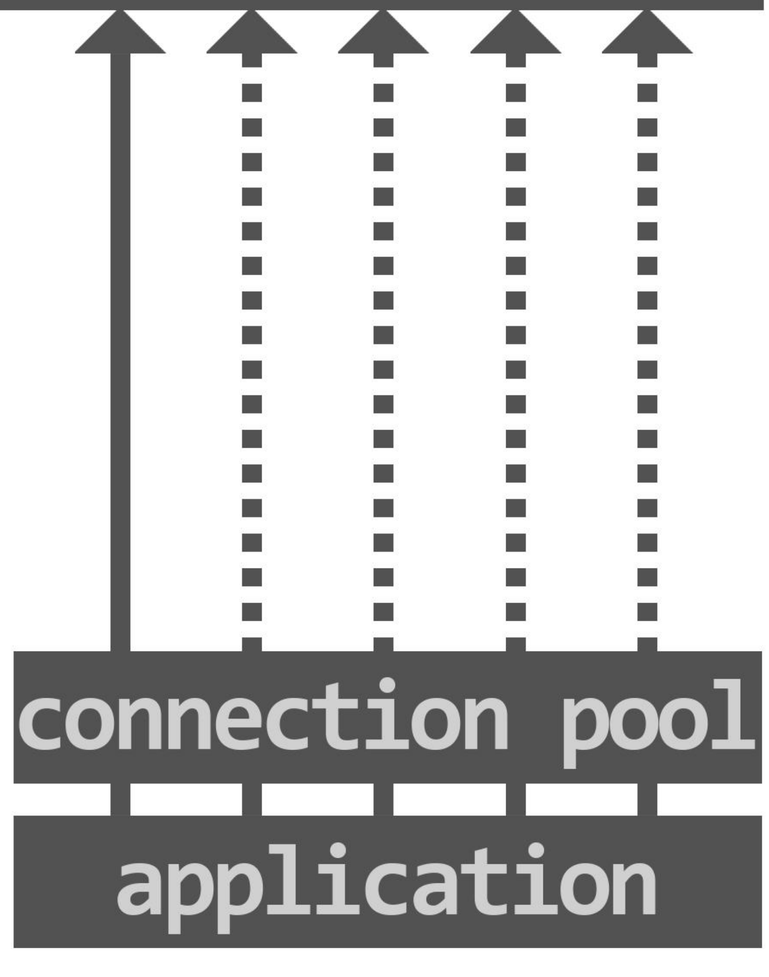

connection pool
application

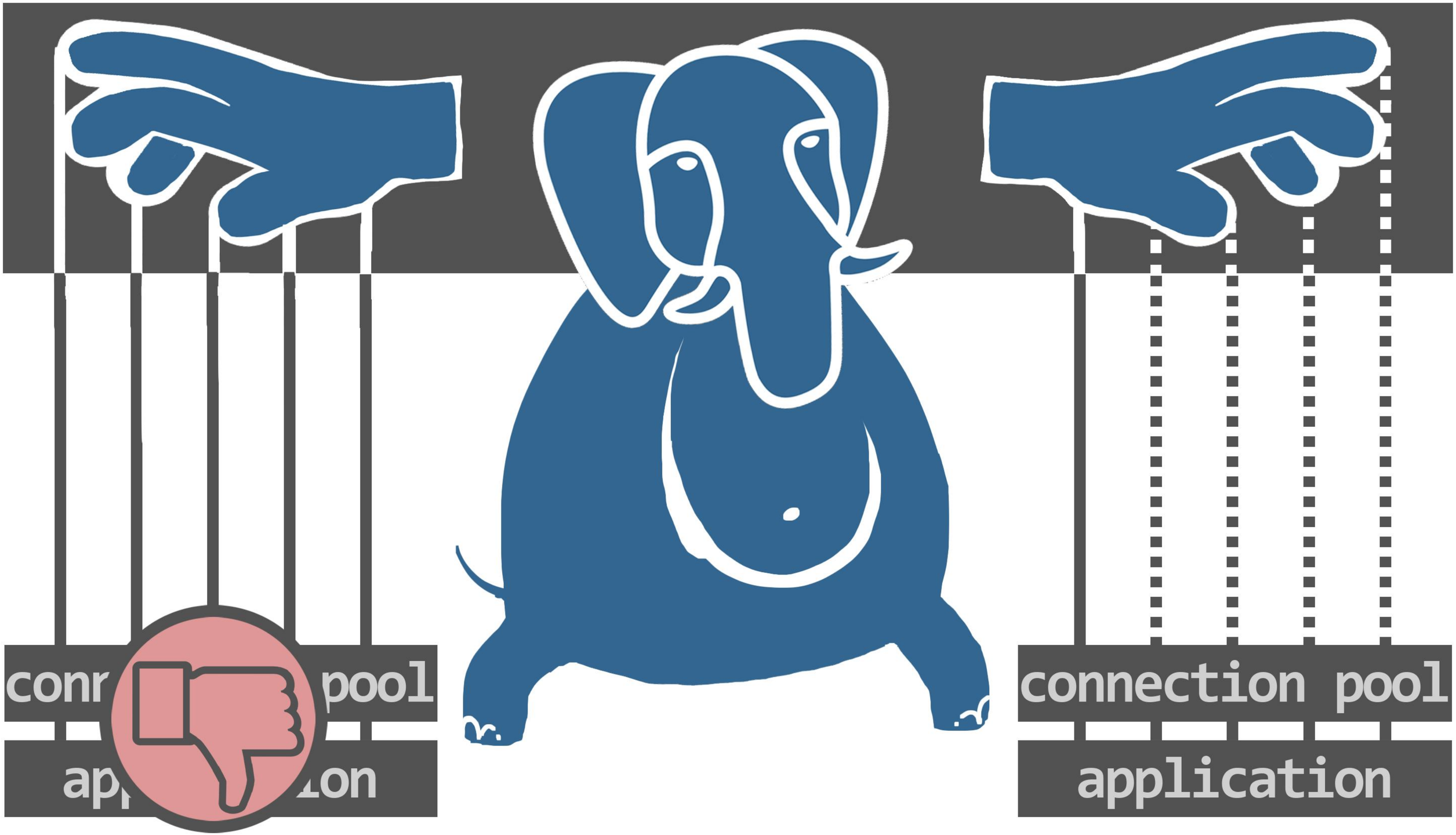


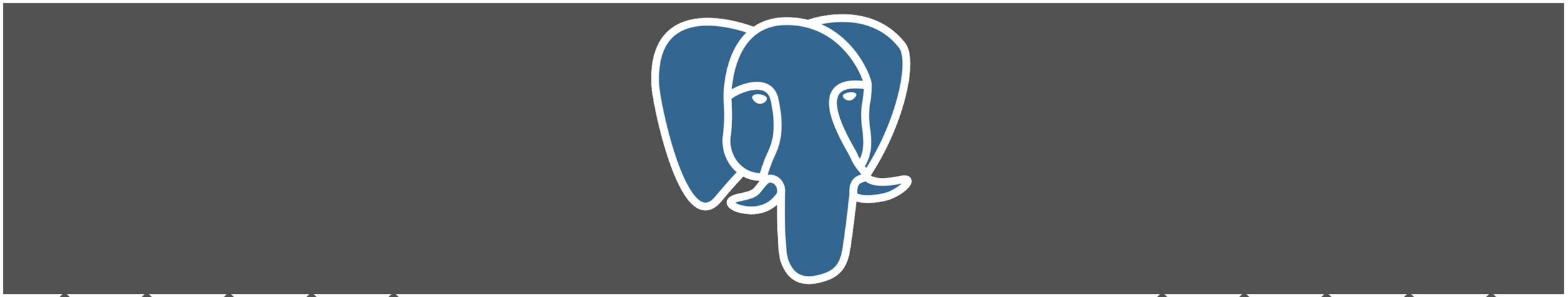

conr
pool
ap
on

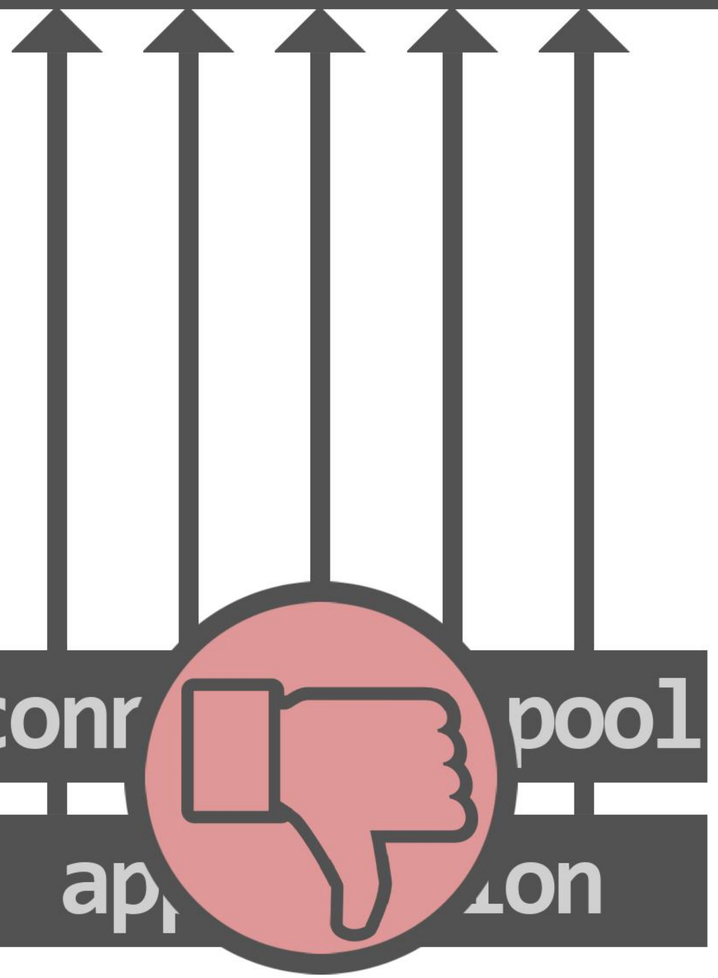



pool
application

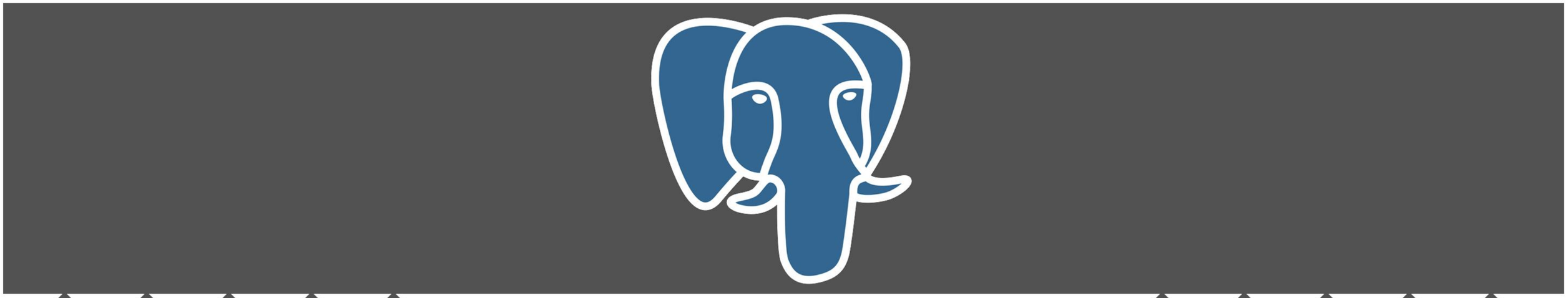

connection pool
application


connection pool
application
connection pool
application




conn pool
application

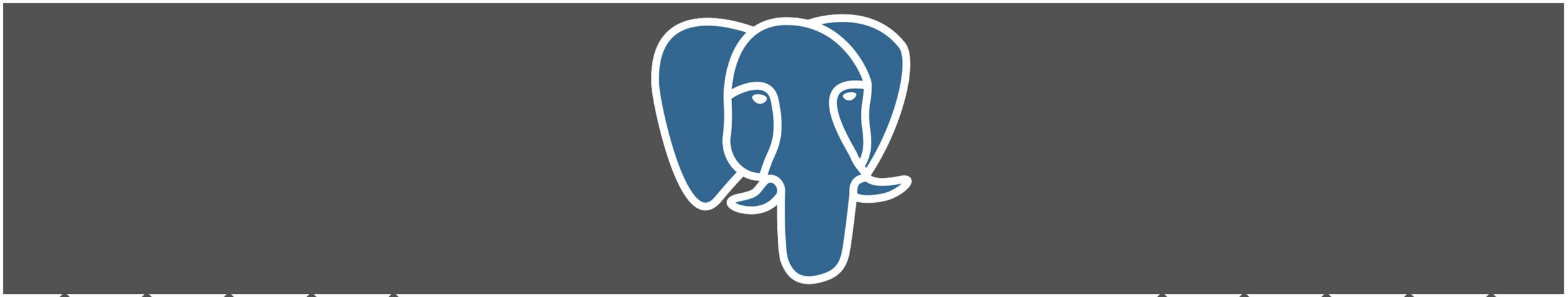

минимальное
количество
коннекшнов
максимальное
количество
коннекшнов
connection pool
application



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

минимальное количество коннекшнов

максимальное количество коннекшнов

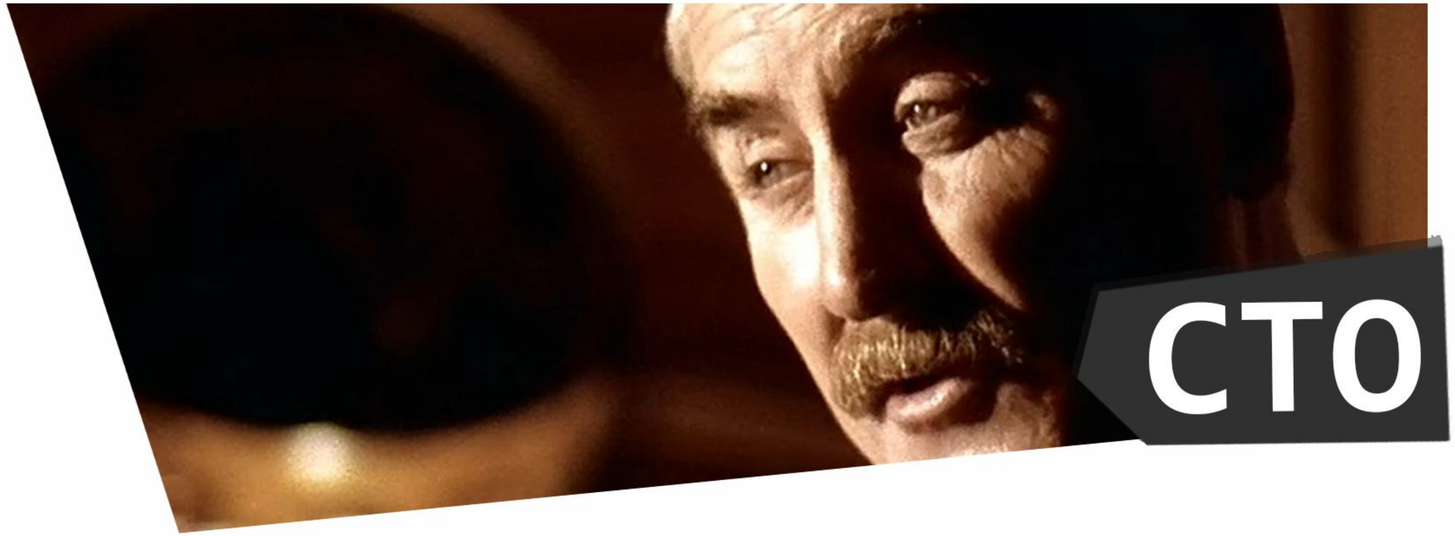



```
spring:
  datasource:
    hikari:
      minimumIdle: 1
      maximum-pool-size: 5
      idle-timeout: 60000
```

минимальное количество коннекшнов

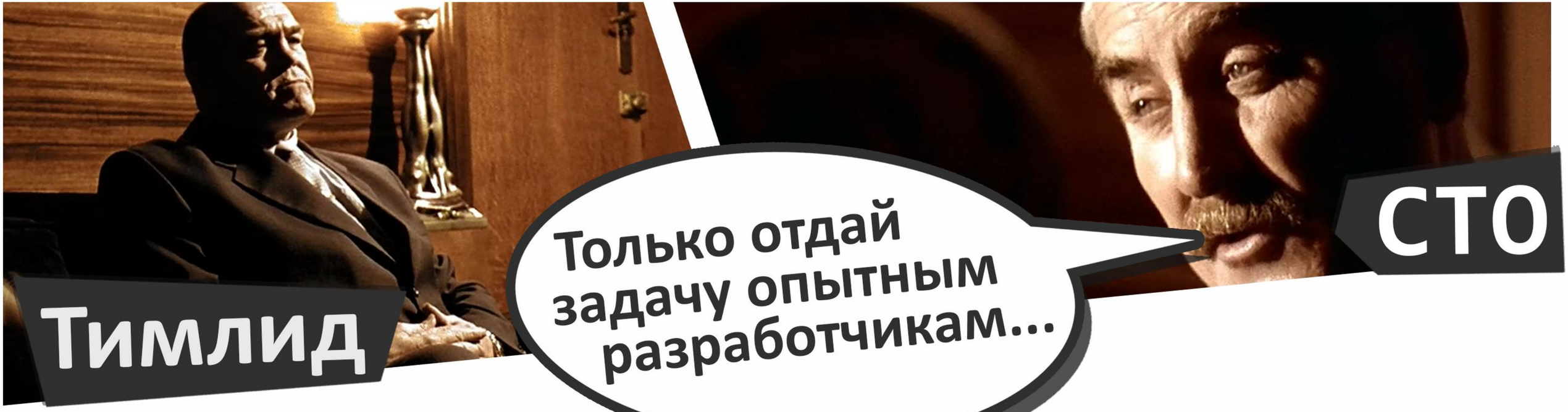
максимальное количество коннекшнов

время до неактивности


CTO

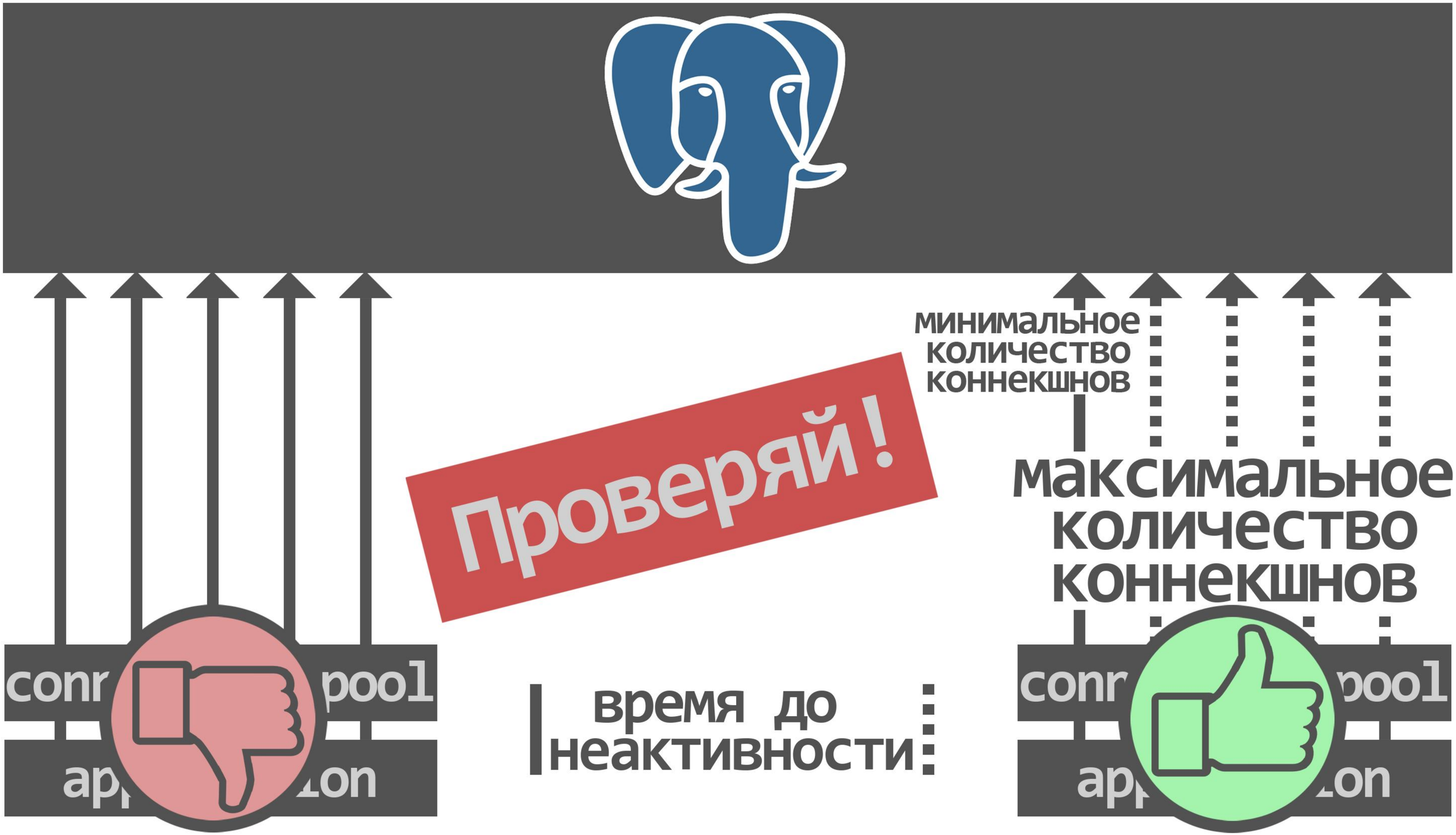

Тимлид


СТО


Тимлид
Только отдай задачу опытным разработчикам...
СТО


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

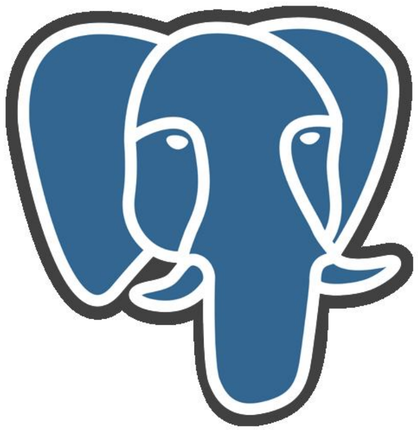

Тимлид
Только отдай задачу опытным разработчикам...
СТО
Senior Developer
Lead Developer
ДБА


Проверяй!
минимальное количество коннекшнов
максимальное количество коннекшнов
время до неактивности
conn pool
conn pool


Liquibase


Liquibase

Liquibase

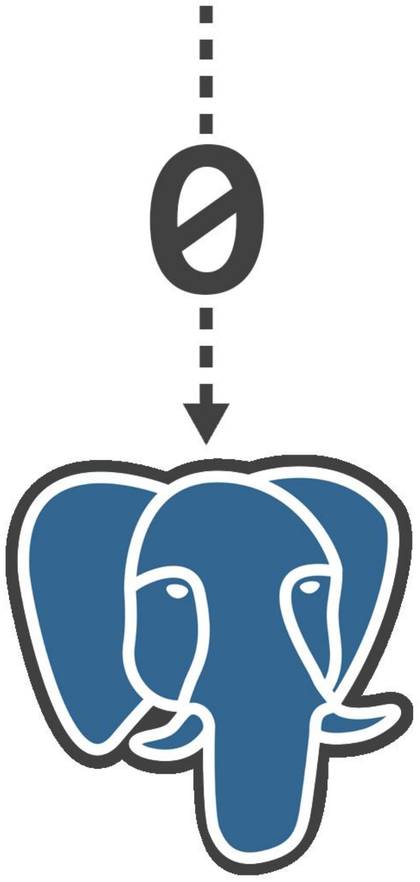



Liquibase
10

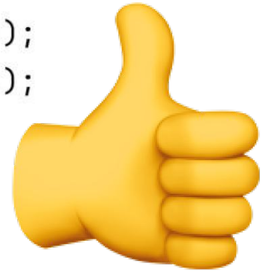
```
management.endpoint.liquibase.enabled=false
```

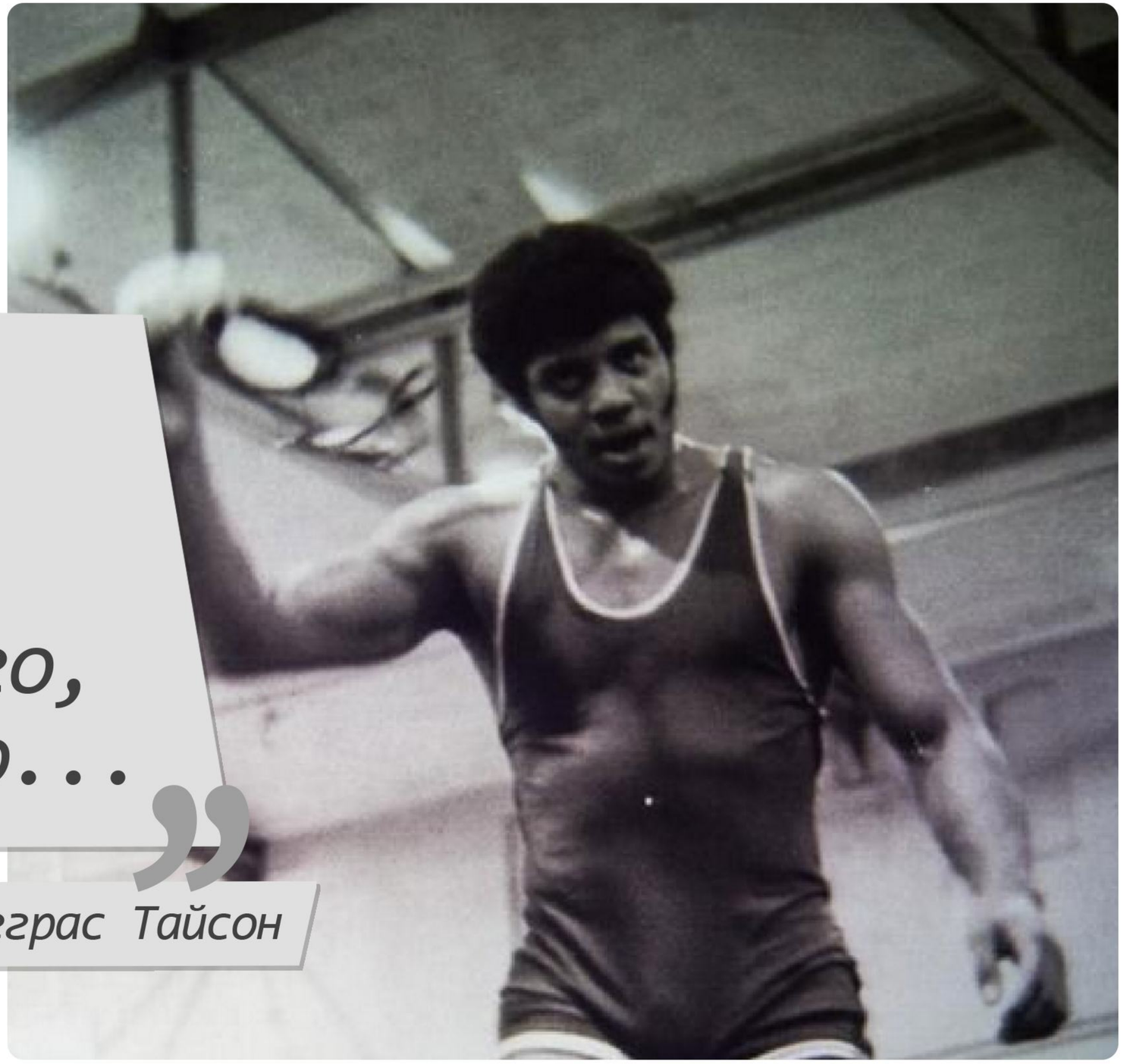

```
management.endpoint.liquibase.enabled=false
```

```
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();
    }
}
```



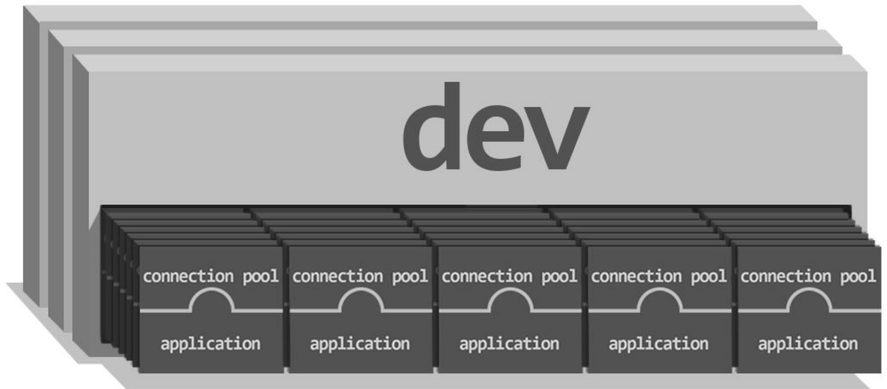
```java
@Configuration
public class JpaConfiguration {
    @Bean
    @Primary
    @LiquibaseDataSource
    @ConfigurationProperties(
            prefix = "spring.datasource.hikari")
    public HikariDataSource dataSource(
            DataSourceProperties properties
    ) {
        return newHikariDataSource(properties);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

    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;
    }
}
```

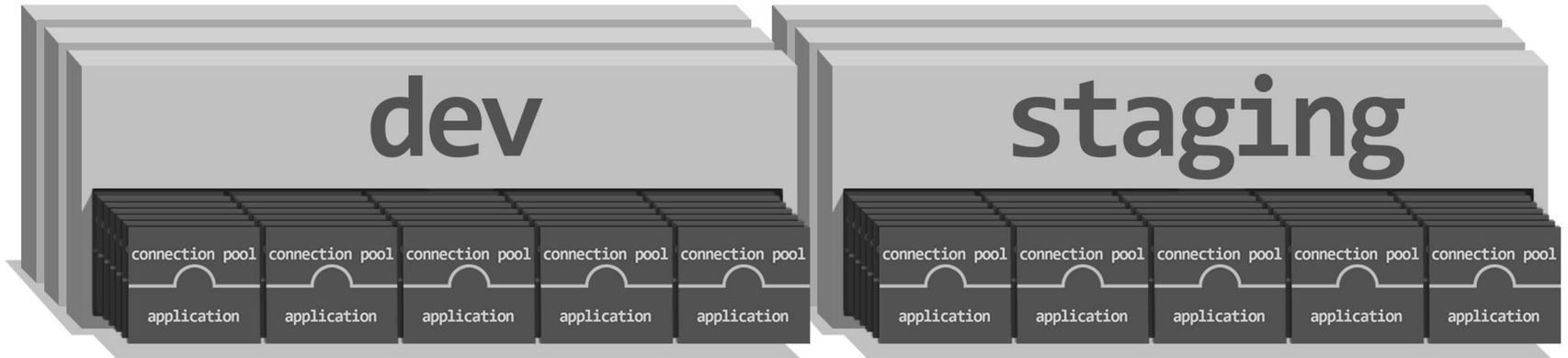


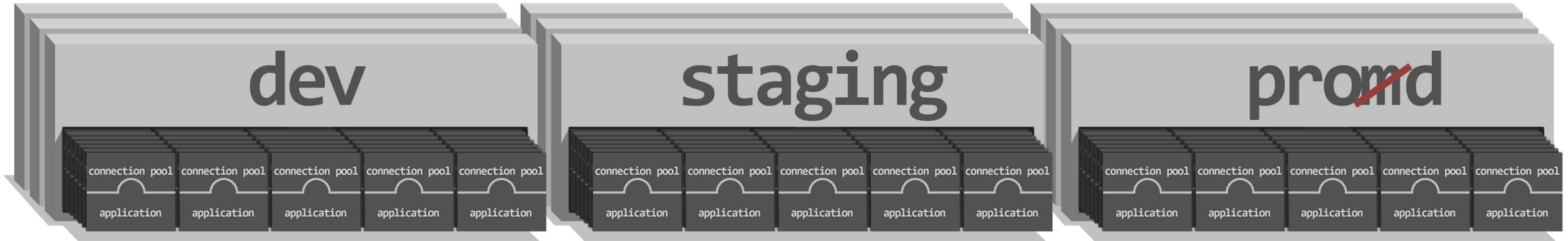


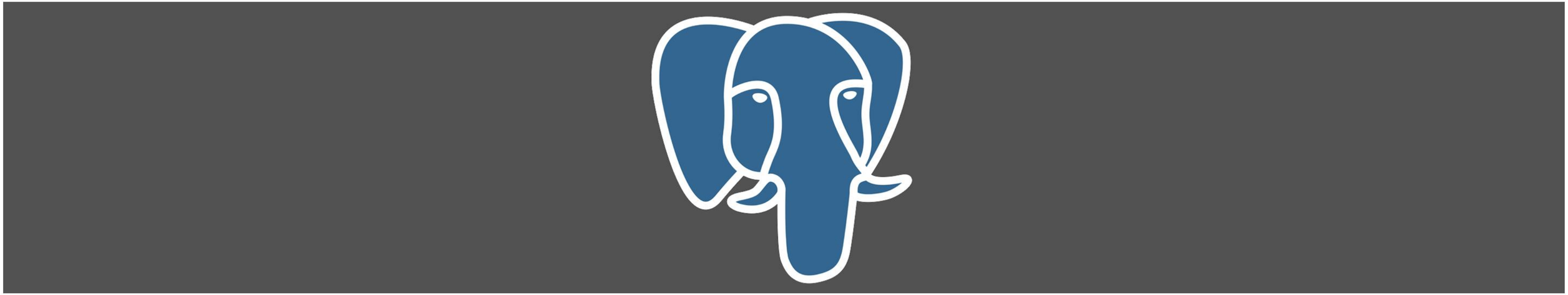
“Если вы нашли какую-то одну штуку, то, скорее всего, таких штук много...”

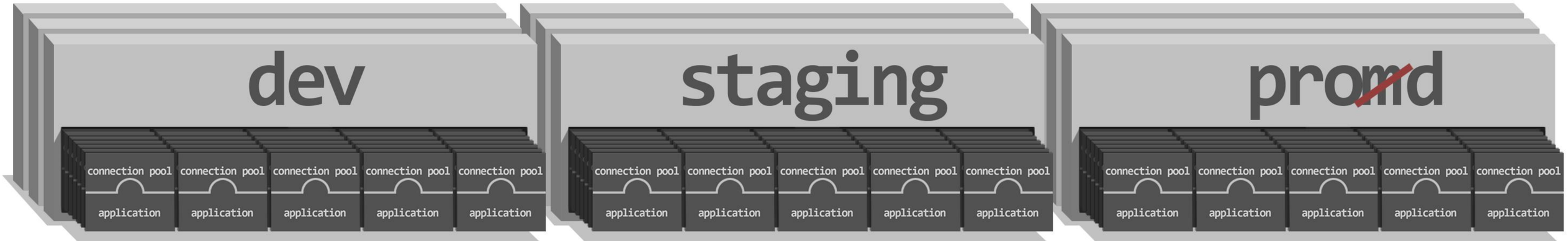
Неизвестный Нил Деграс Тайсон

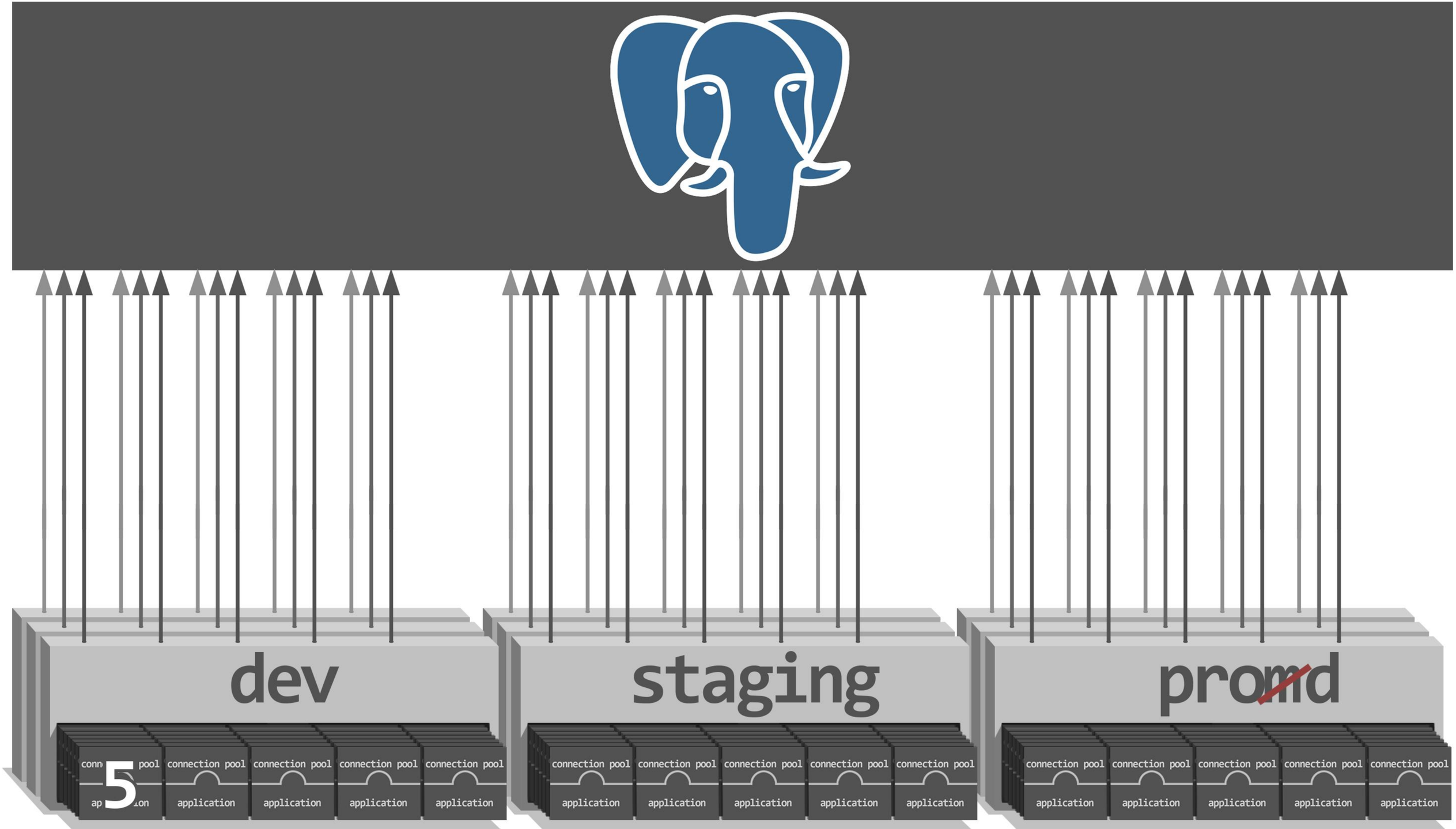

dev
connection pool
connection pool
connection pool
connection pool
connection pool
application
application
application
application
application

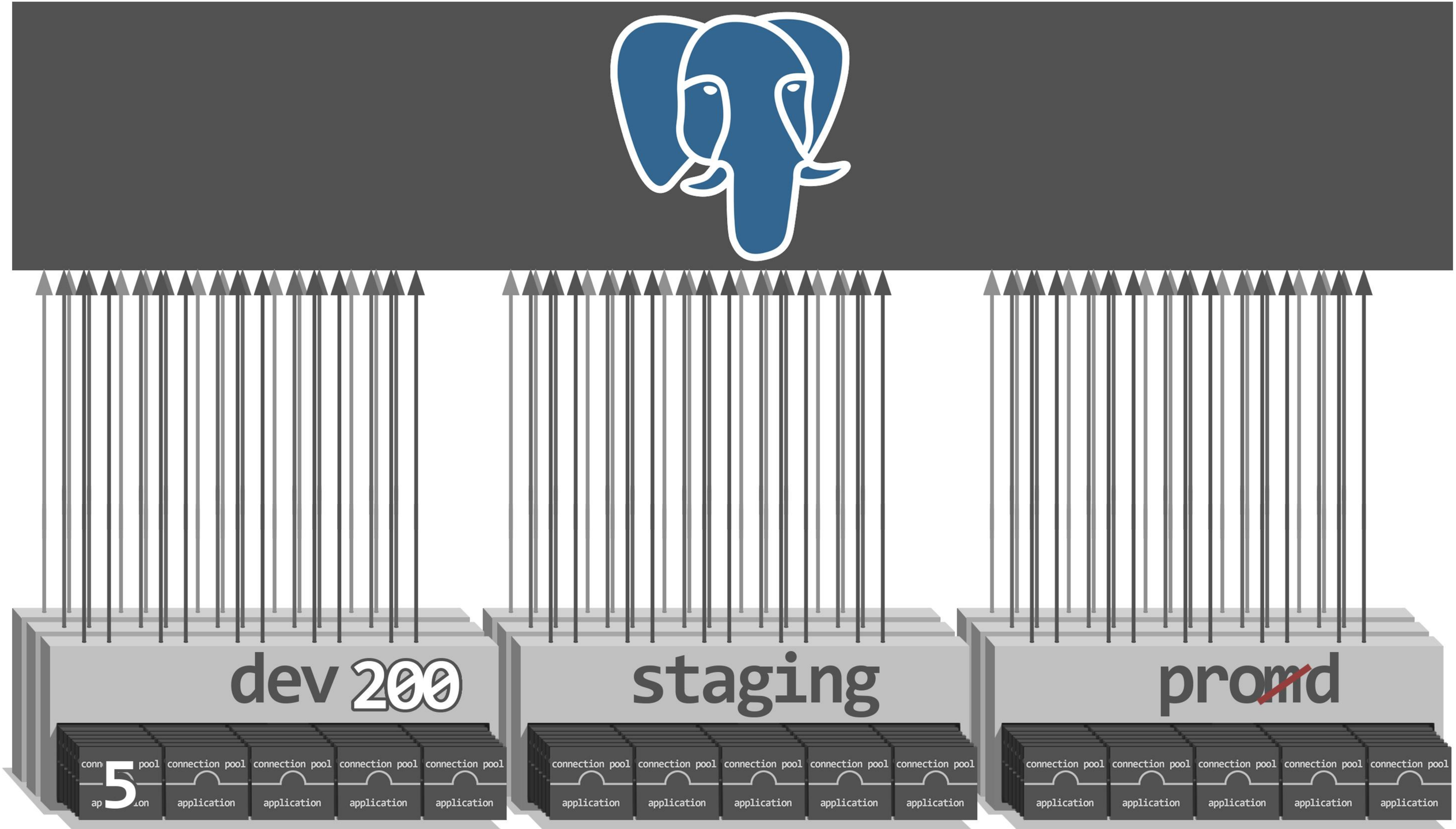

dev
connection pool
connection pool
connection pool
connection pool
connection pool
application
application
application
application
application
staging
connection pool
connection pool
connection pool
connection pool
connection pool
application
application
application
application
application

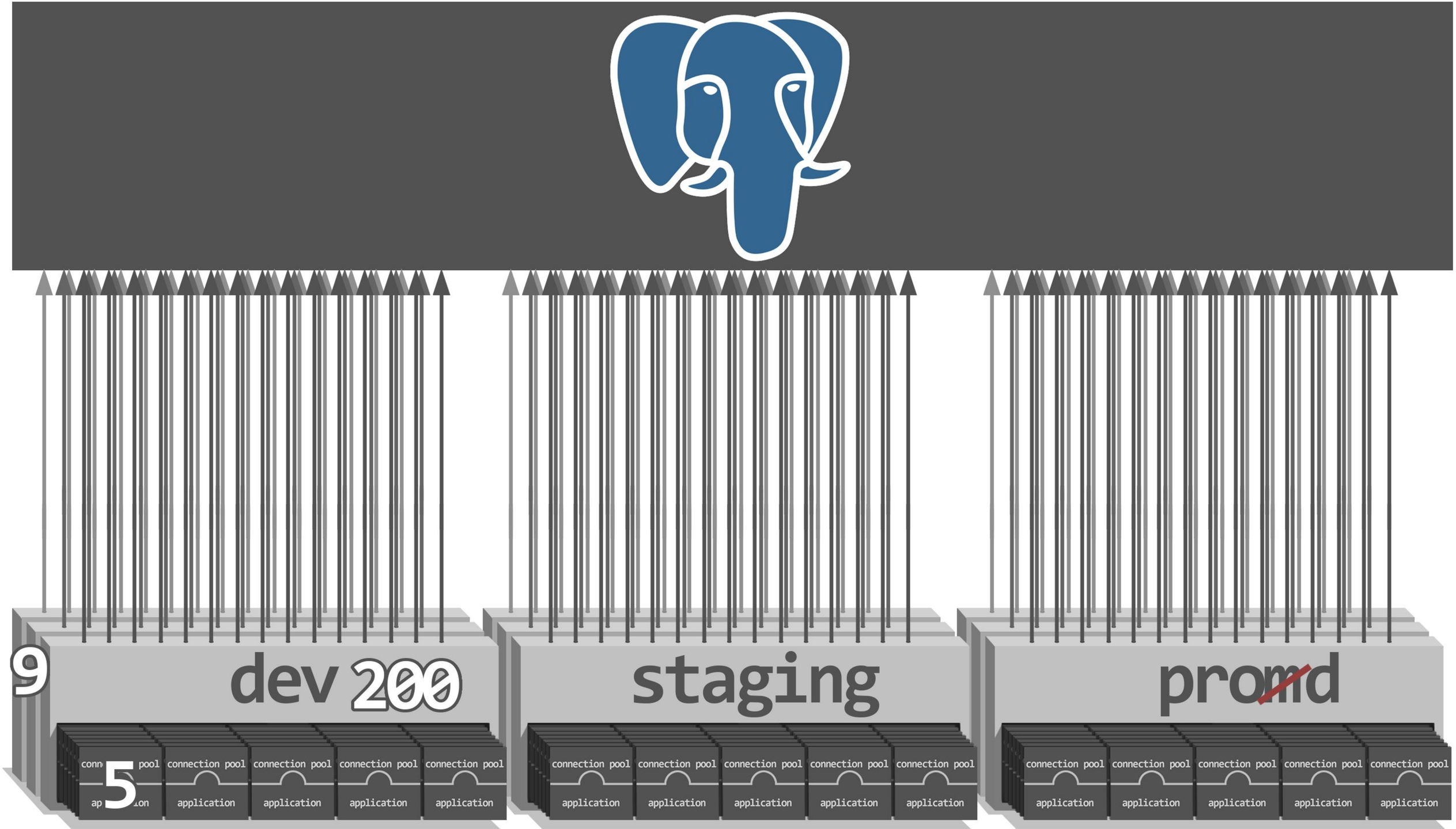

dev
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
staging
connection pool
application
connection pool
application
connection pool
application
connection pool
application
connection pool
application
prod
connection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

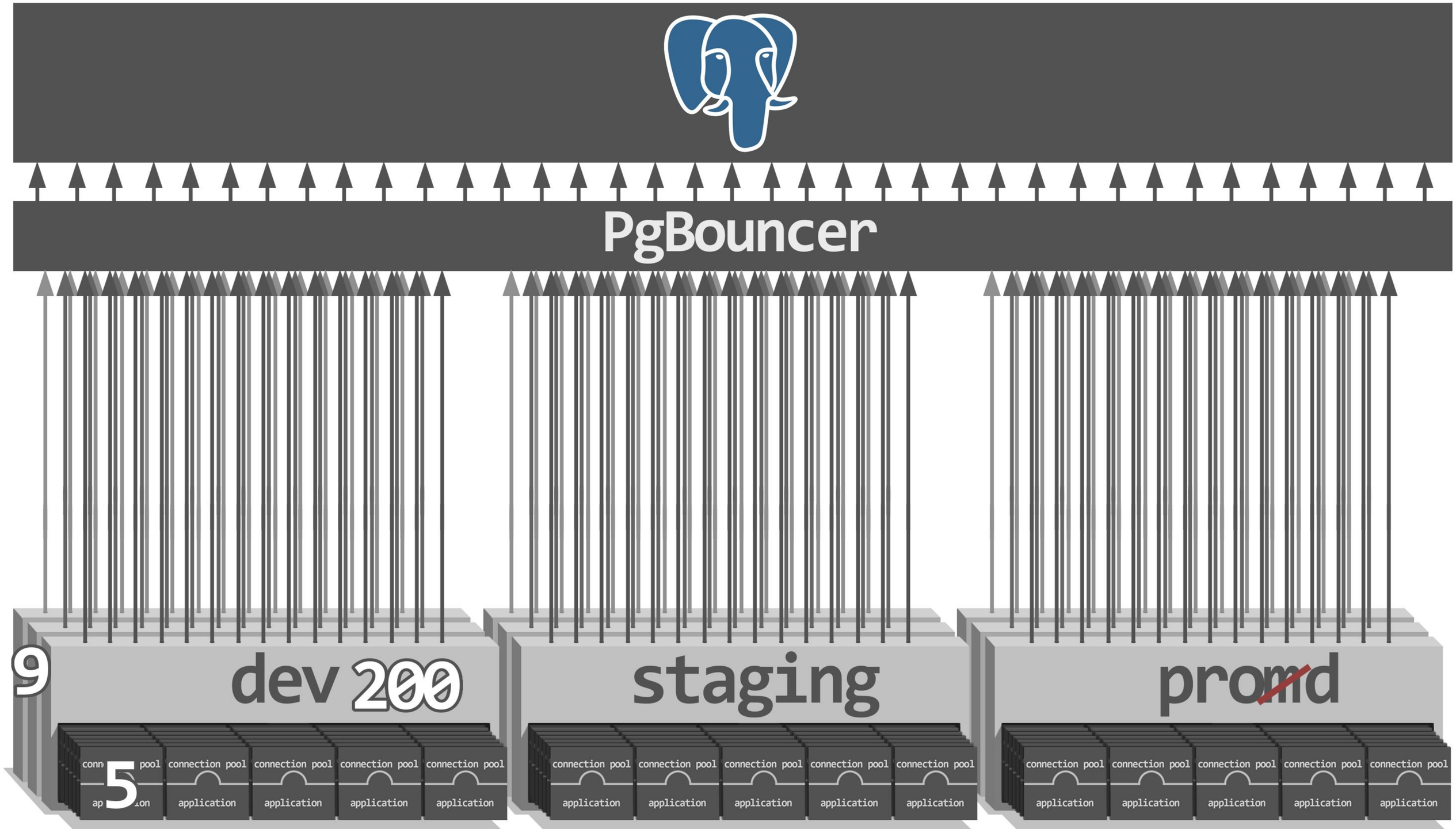



dev
connection pool
application
staging
connection pool
application
prod
connection pool
application


dev
staging
prod
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

Три типа ритуалов

1. Оптимизация запросов
2. Простановка границ транзакций
3. Экономия коннекшнов
4. 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

Offset и keyset

почём пагинация для продакшена?

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

Антипаттерн orisnull:

коварство иллюзорной простоты