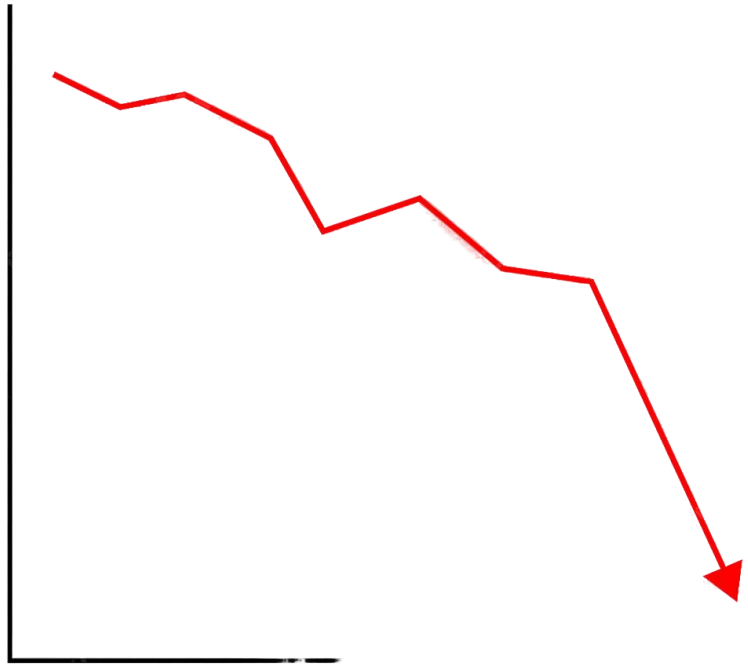
Высоконагруженные системы

Лекция

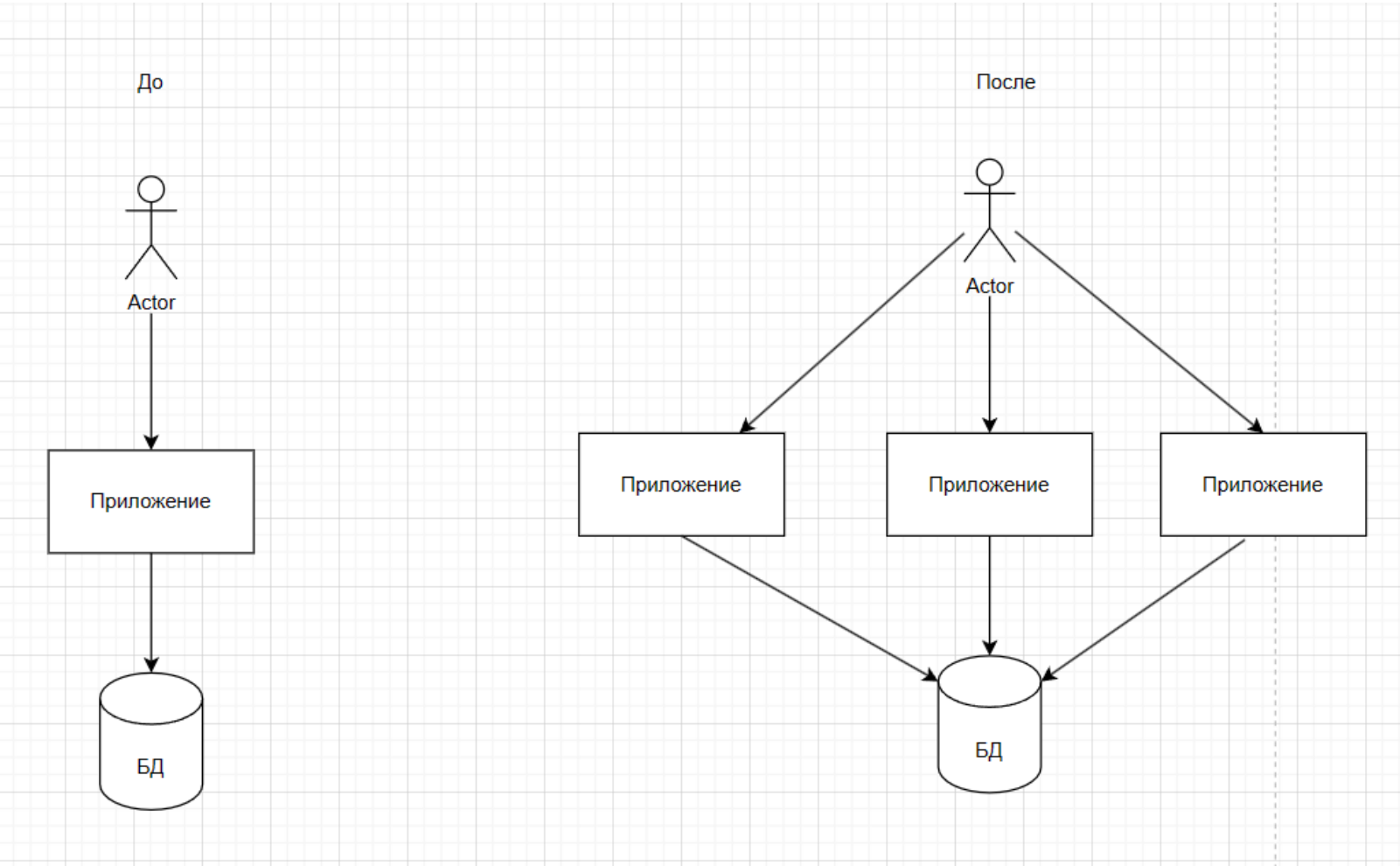
RPS. Масштабирование. Базы данных

RPS – requests per second (запросы в секунду)

При невозможности обработать большие нагрузки бизнес теряет клиентов



Масштабирование приложения



База данных – слабое место

Настройки БД

Настройка БД для обработки запросов

Индексы

Оптимизация запросов

Оптимизируем запросы которые идут к БД

Де нормализация

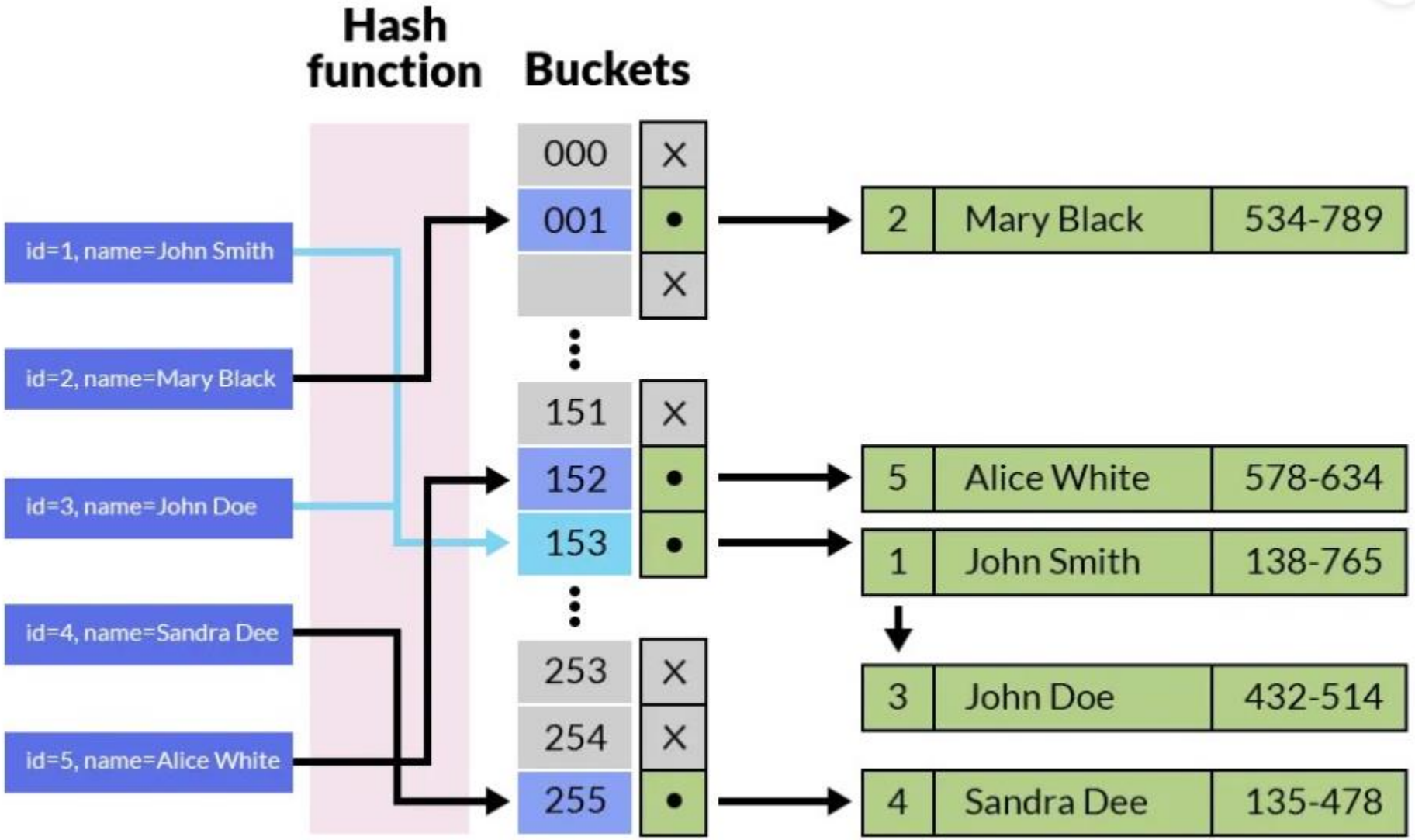
Денормализация данных

Настройка БД

- WORK_MEM – сколько ОЗУ будет выделено для обработки запросов

```
testdb=# SET work_mem TO "2MB";
testdb=# EXPLAIN SELECT * FROM bar ORDER BY bar.b;
                                    QUERY PLAN
-------------------------------------------------------------------------------
Gather Merge  (cost=509181.84..1706542.14 rows=10000116 width=24)
   Workers Planned: 4
   ->  Sort  (cost=508181.79..514431.86 rows=2500029 width=24)
         Sort Key: b
         ->  Parallel Seq Scan on bar  (cost=0.00..88695.29 rows=2500029 width=24)
(5 rows)
```

Индексы (пример hash-индекса)

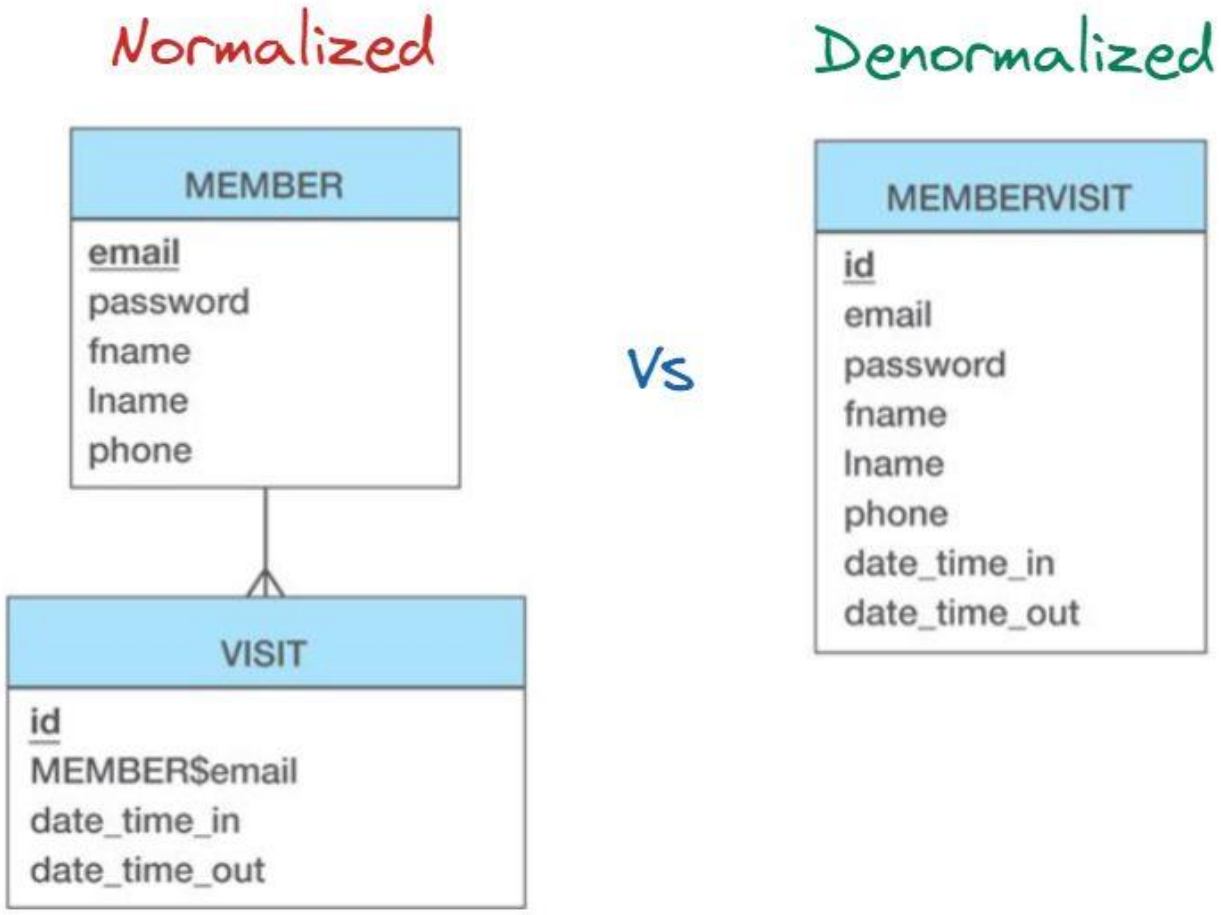


Оптимизация запросов

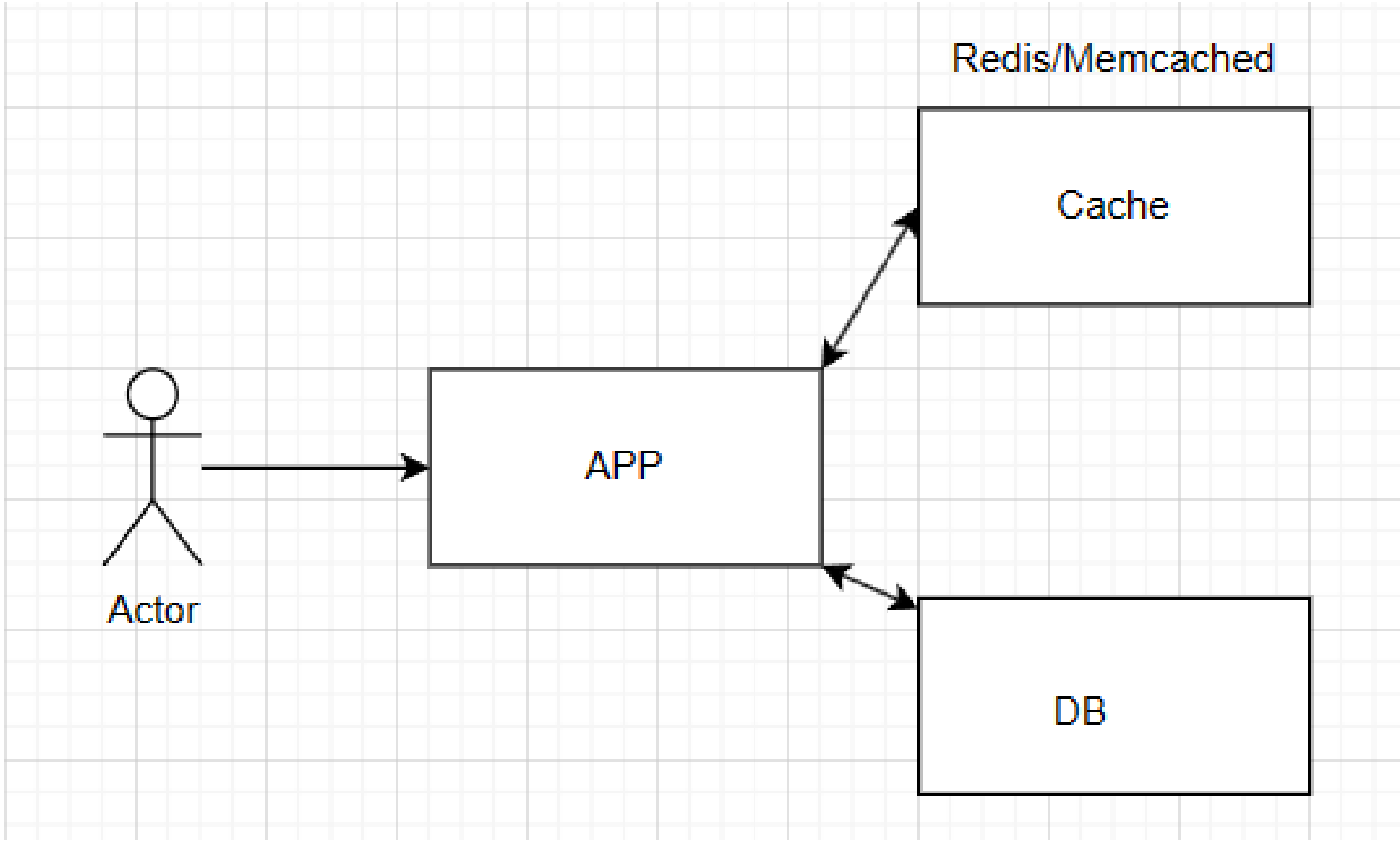
```
EXPLAIN SELECT * FROM tenk1 WHERE unique1 < 100 AND unique2 > 9000;

                                     QUERY PLAN
-------------------------------------------------------------------------------------
 Bitmap Heap Scan on tenk1  (cost=11.27..49.11 rows=11 width=244)
   Recheck Cond: ((unique1 < 100) AND (unique2 > 9000))
   ->  BitmapAnd  (cost=11.27..11.27 rows=11 width=0)
         ->  Bitmap Index Scan on tenk1_unique1  (cost=0.00..2.37 rows=106 width=0)
               Index Cond: (unique1 < 100)
         ->  Bitmap Index Scan on tenk1_unique2  (cost=0.00..8.65 rows=1042 width=0)
               Index Cond: (unique2 > 9000)
```

```
EXPLAIN SELECT * FROM tenk1 t1, tenk2 t2 WHERE t1.unique1 < 100 AND t1.unique2 = t2.unique2;

                                     QUERY PLAN
-------------------------------------------------------------------------------------
 Nested Loop  (cost=2.37..553.11 rows=106 width=488)
   ->  Bitmap Heap Scan on tenk1 t1  (cost=2.37..232.35 rows=106 width=244)
         Recheck Cond: (unique1 < 100)
         ->  Bitmap Index Scan on tenk1_unique1  (cost=0.00..2.37 rows=106 width=0)
               Index Cond: (unique1 < 100)
   ->  Index Scan using tenk2_unique2 on tenk2 t2  (cost=0.00..3.01 rows=1 width=244)
         Index Cond: (t2.unique2 = t1.unique2)
```

Денормализация



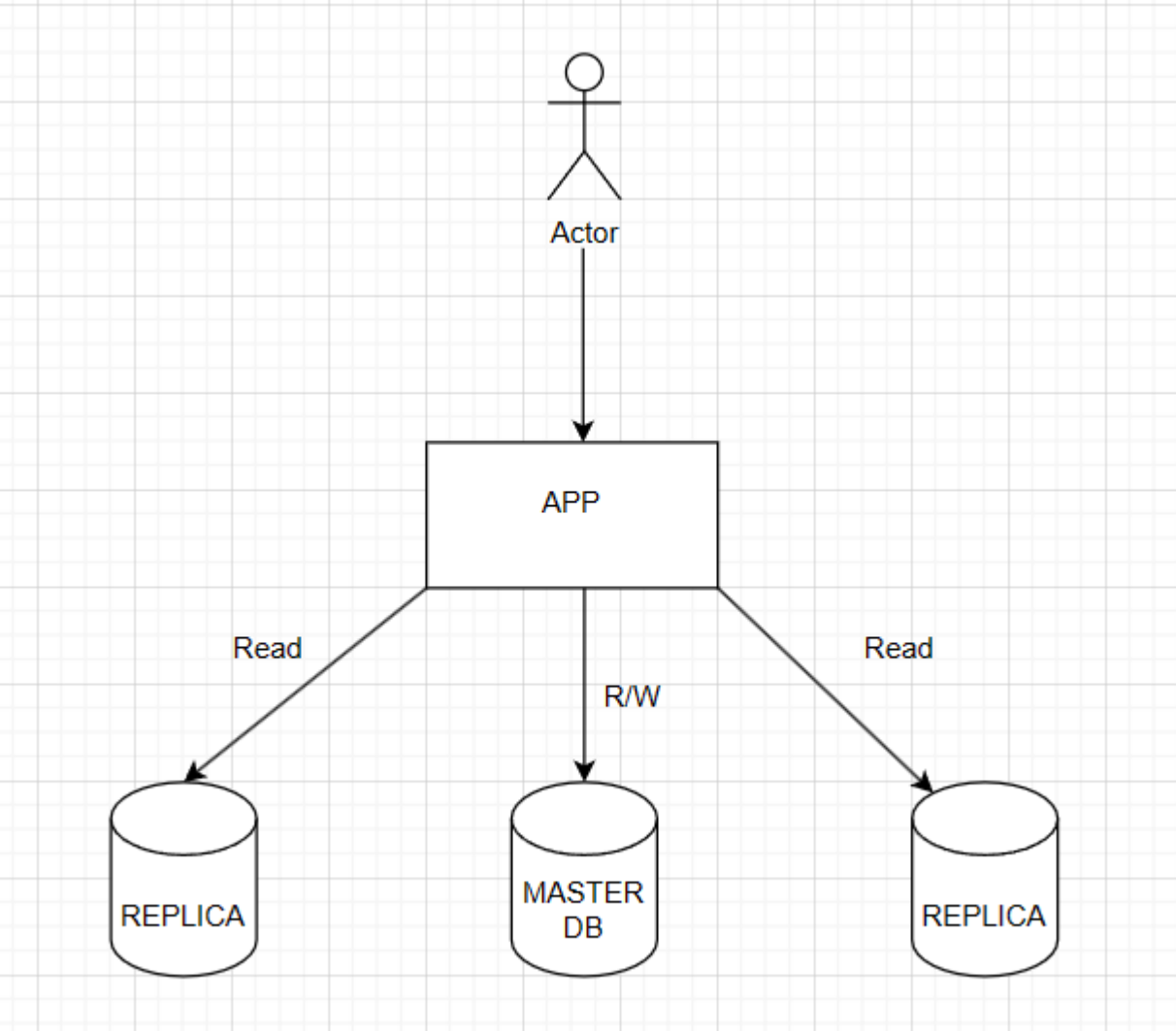
Кеширование



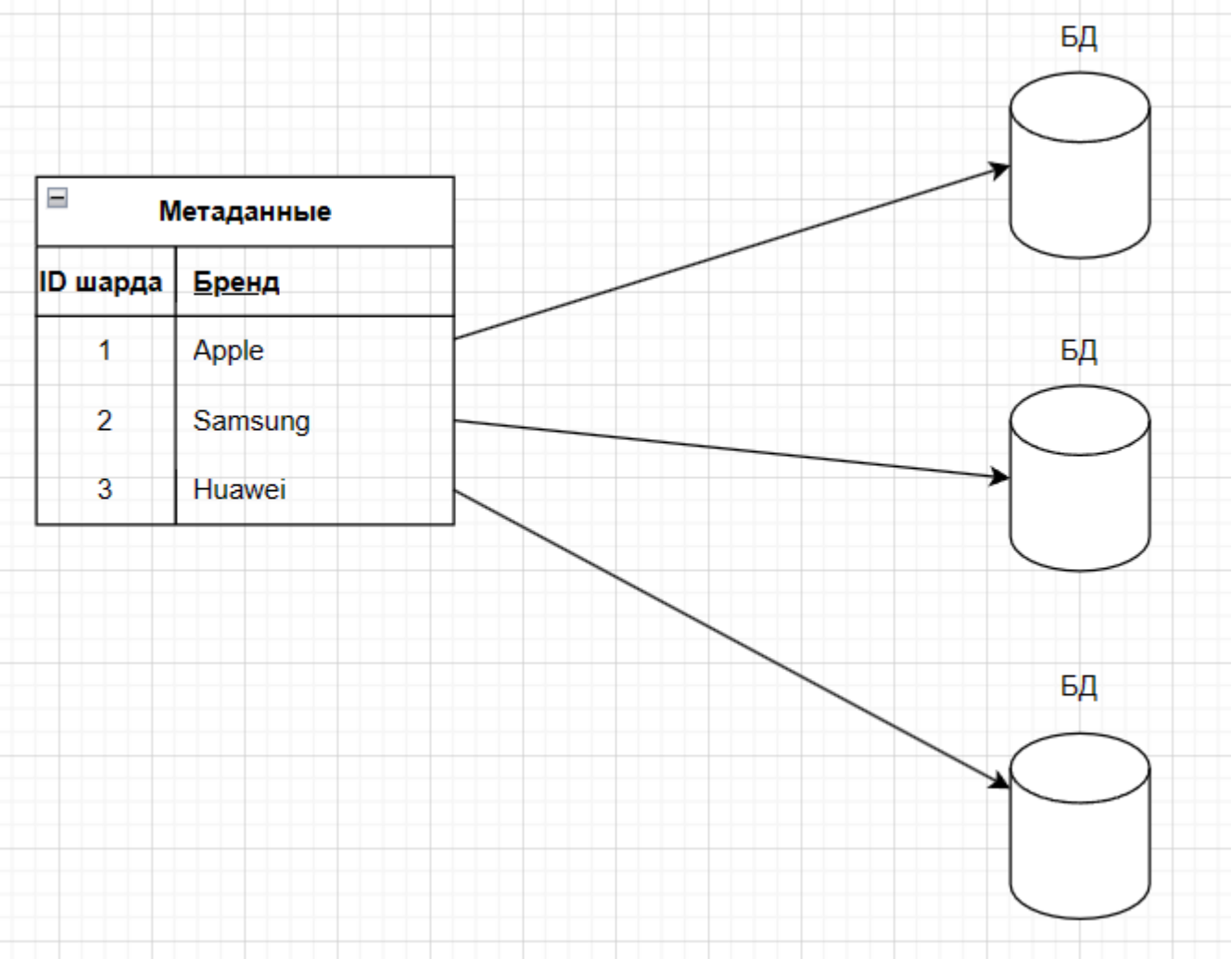
Декоратор кеширования - пример

```python
def cache_result(func):
    cache = {}

    def wrapper(*args, **kwargs):
        key = (*args, *kwargs.items())

        if key in cache:
            print("Retrieving result from cache...")
            return cache[key]

        result = func(*args, **kwargs)
        cache[key] = result

        return result

    return wrapper
```

Репликация (Масштабирование на чтение)

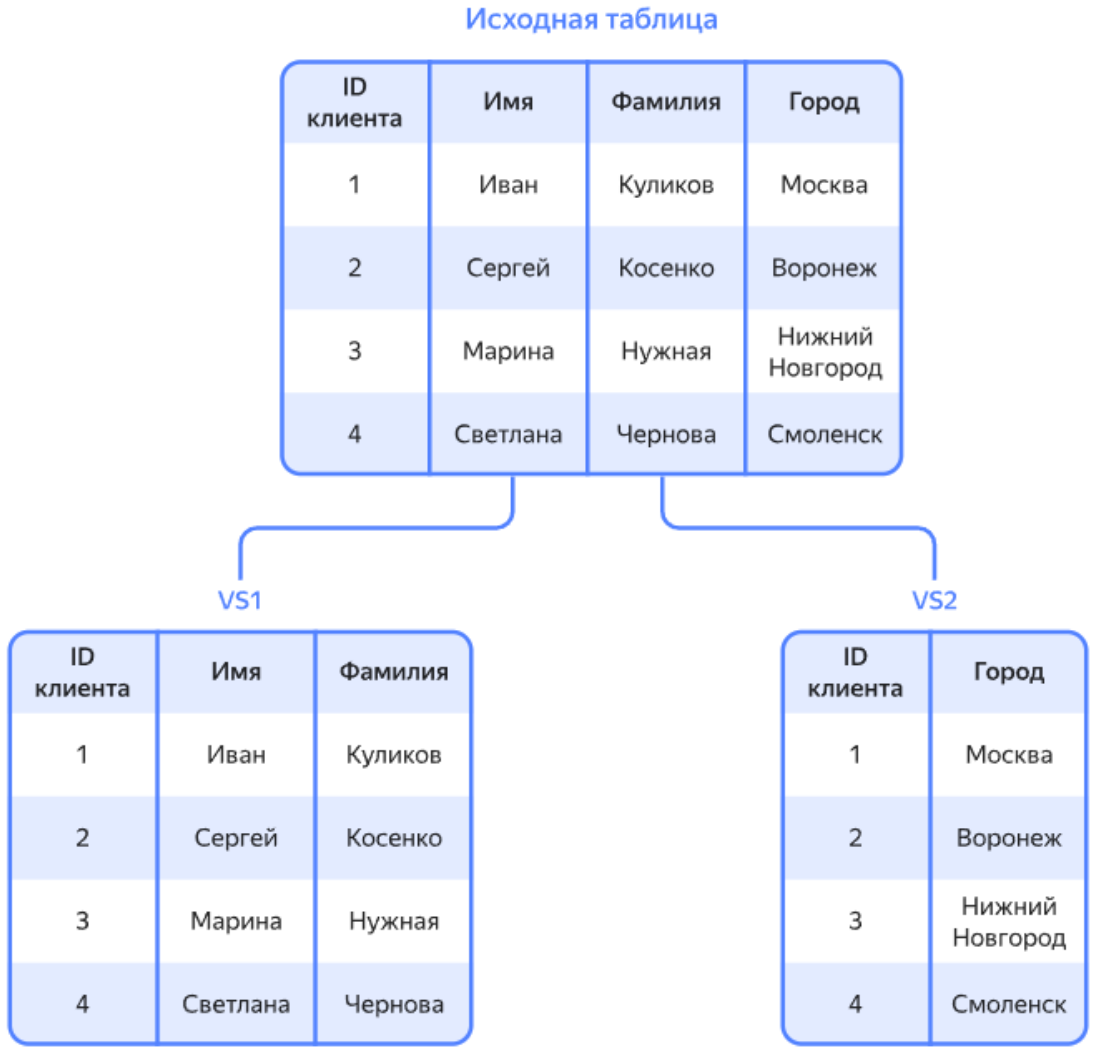


Шардирование



Шардирование — принцип проектирования базы данных, при котором логически независимые строки таблицы базы данных хранятся раздельно, заранее сгруппированные в секции. Эти секции, в свою очередь, размещаются на разных, физически и логически независимых серверах базы данных.

Вертикальное шардирование (изменение структуры БД)



Горизонтальное шардирование (изменение структуры БД)