PostgresPro

Ускоряем ваше приложение без смс и регистрации или как кэшировать данные правильно

Ахмедов Влад

- Работаю в PostgresPro
- Студент НГУ
- Разрабатываю кэш

PostgresPro

Postgres Professional сегодня

10 лет

на рынке с 2015

ТОП-5

в мире по вкладу
в PostgreSQL

(100+ патчей ежегодно)

№1

в России среди
разработчиков СУБД,
по данным ЦСР (2025)

500+

специалистов в команде,
включая Major Contributors
PostgreSQL

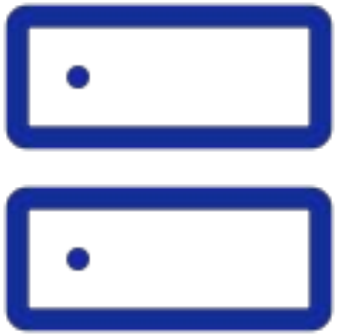
3000+ заказчиков

крупнейшие госкомпании,
банки, телеком, критическая
инфраструктура

Зачем нужно кэширование?

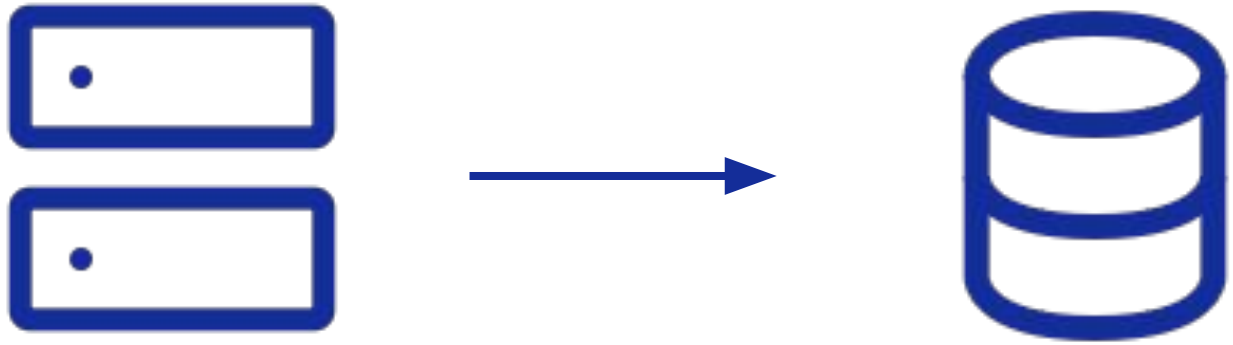
Ваше крутое приложение

PostgresPro



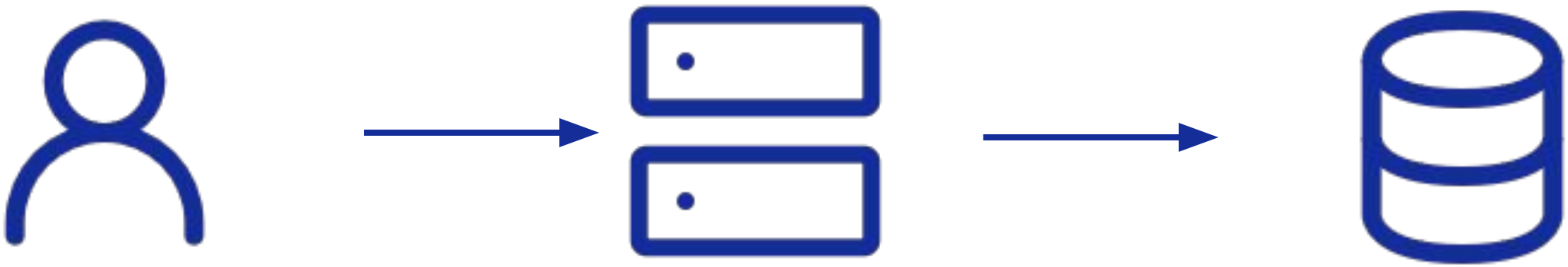
Ваше крутое приложение

PostgresPro



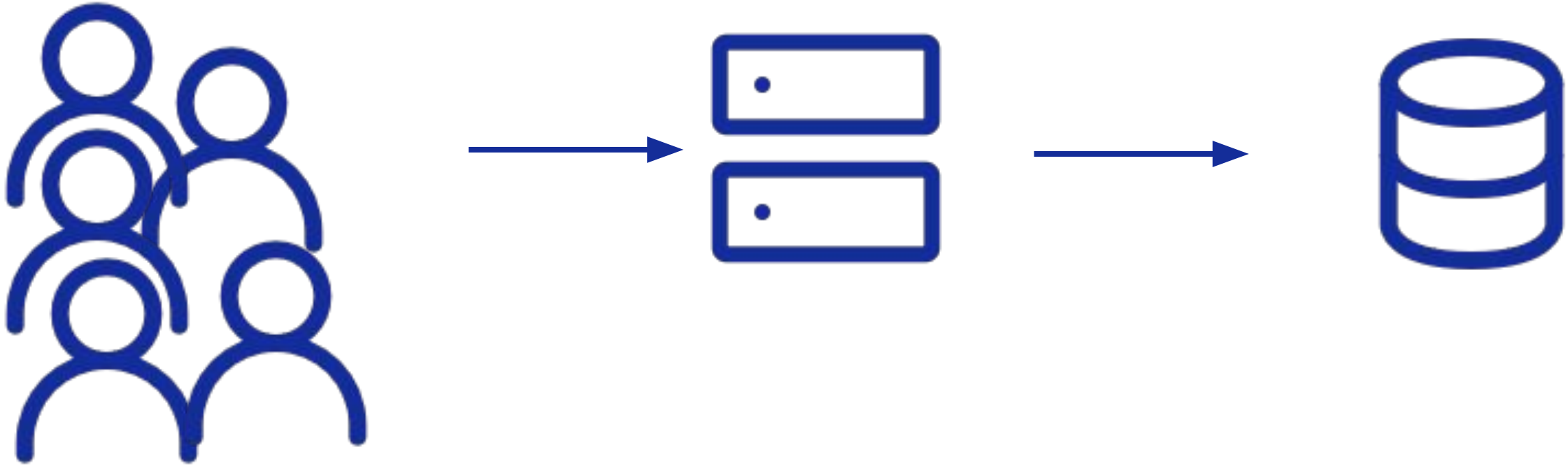
Ваше крутое приложение

PostgresPro



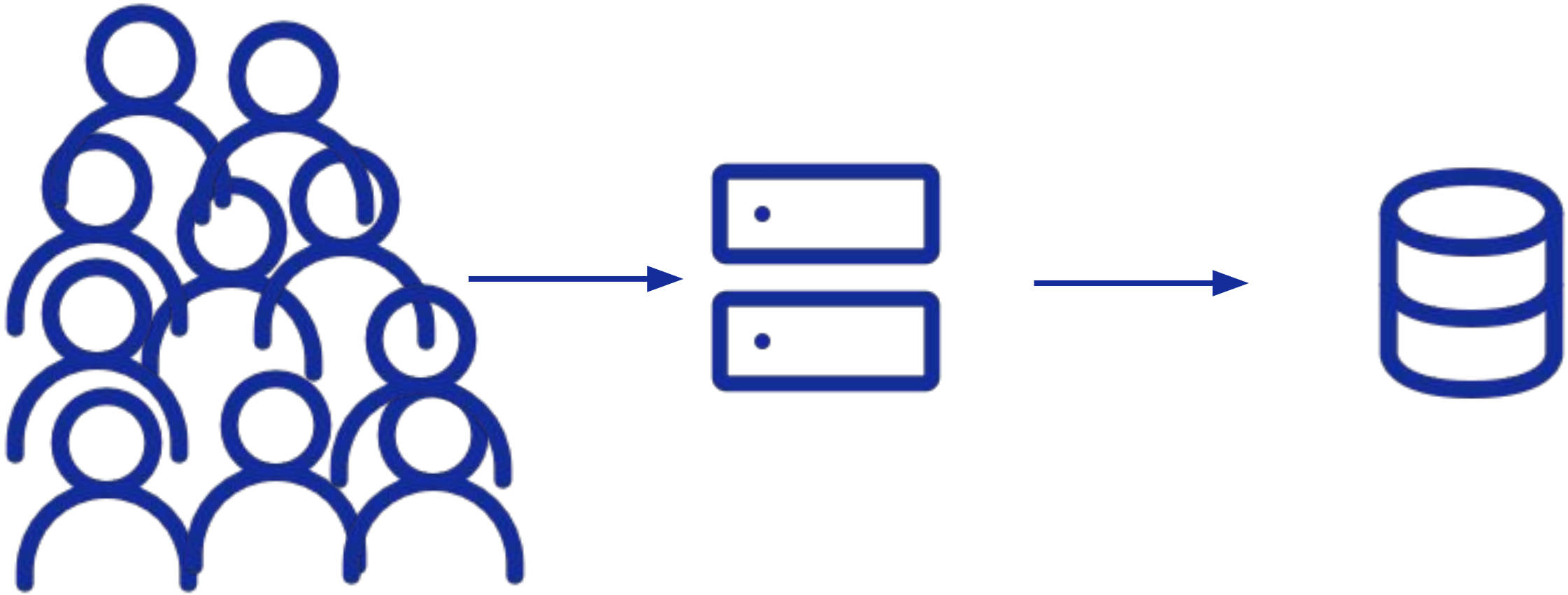
Ваше крутое приложение

PostgresPro



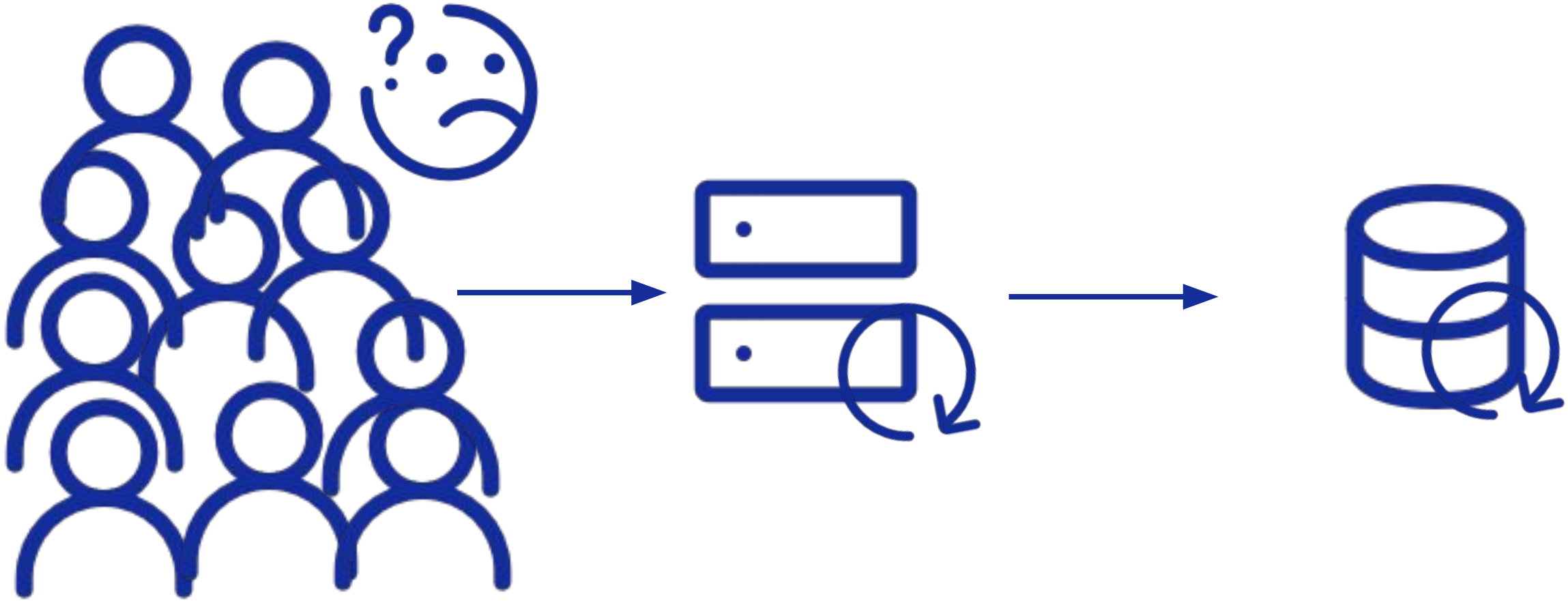
Ваше крутое приложение

PostgresPro



Ваше крутое приложение

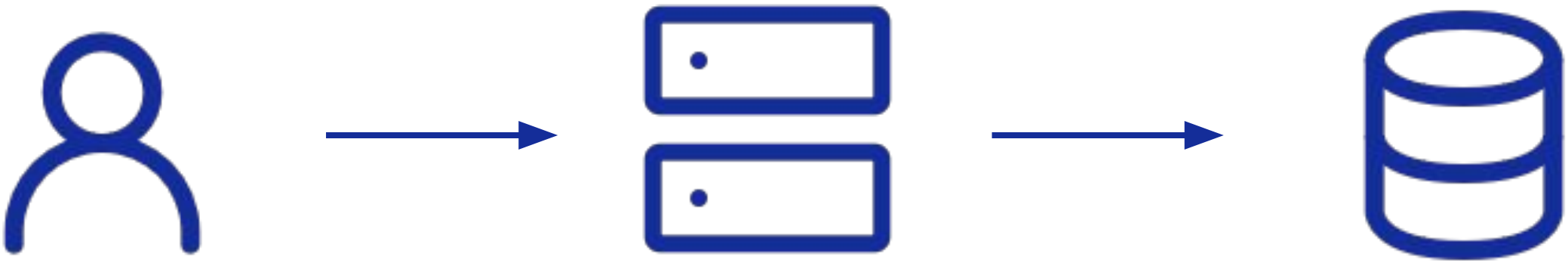
PostgresPro



Почему так?

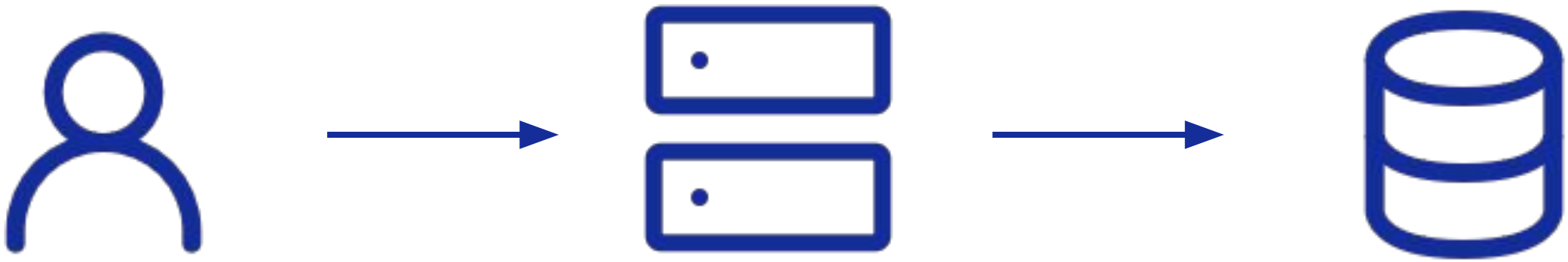
Обработка запроса пользователя

PostgresPro



Обработка запроса пользователя

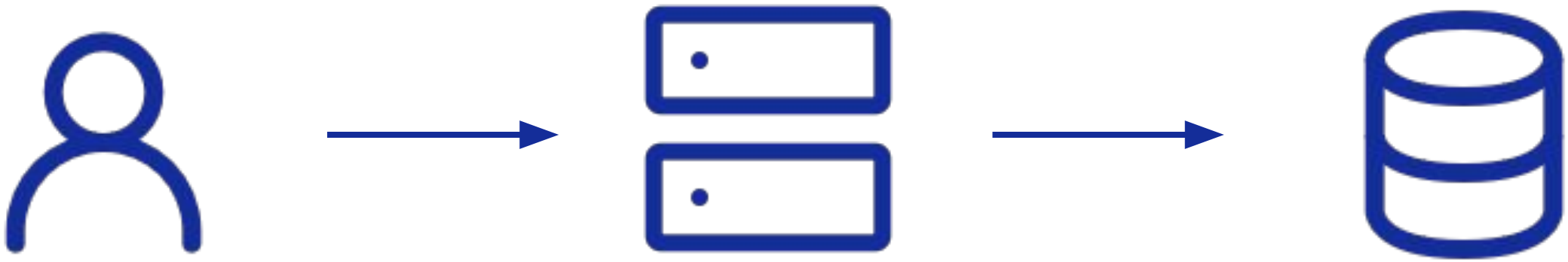
PostgresPro



Дай мне
товар 1

Обработка запроса пользователя

PostgresPro

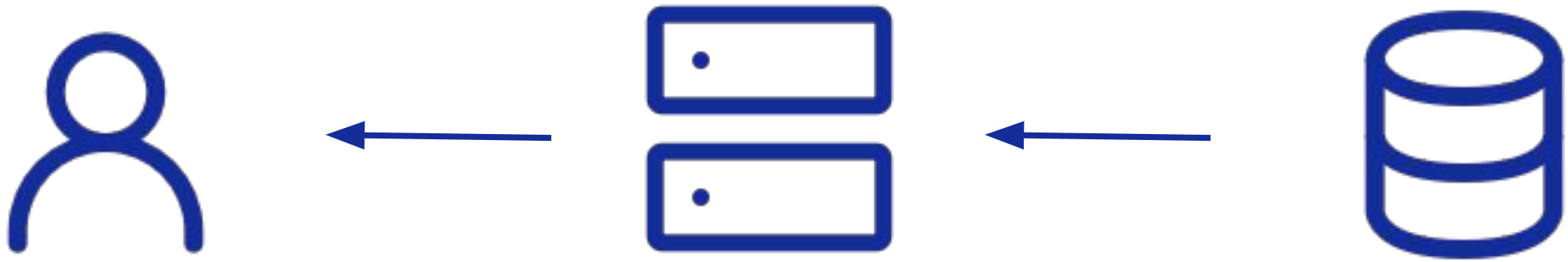


Дай мне
товар 1

Select from table
where id = 1

Обработка запроса пользователя

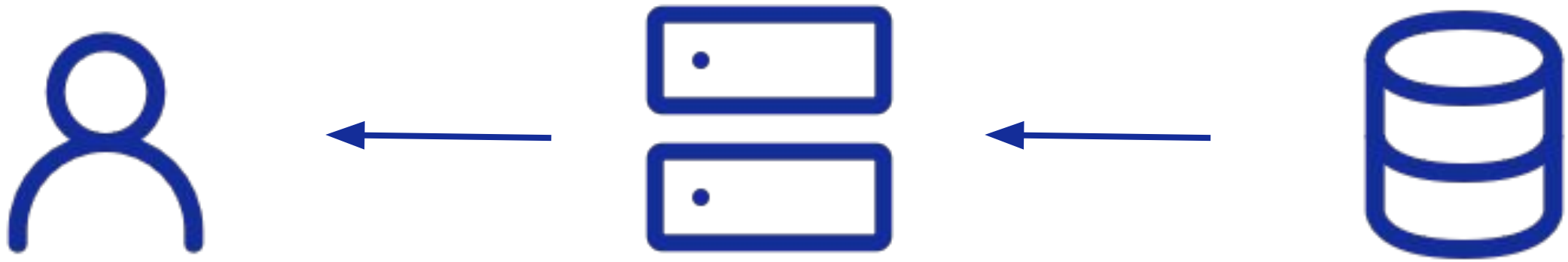
PostgresPro



product where id = 1

Обработка запроса пользователя

PostgresPro



json:
id :1
des: tovar …

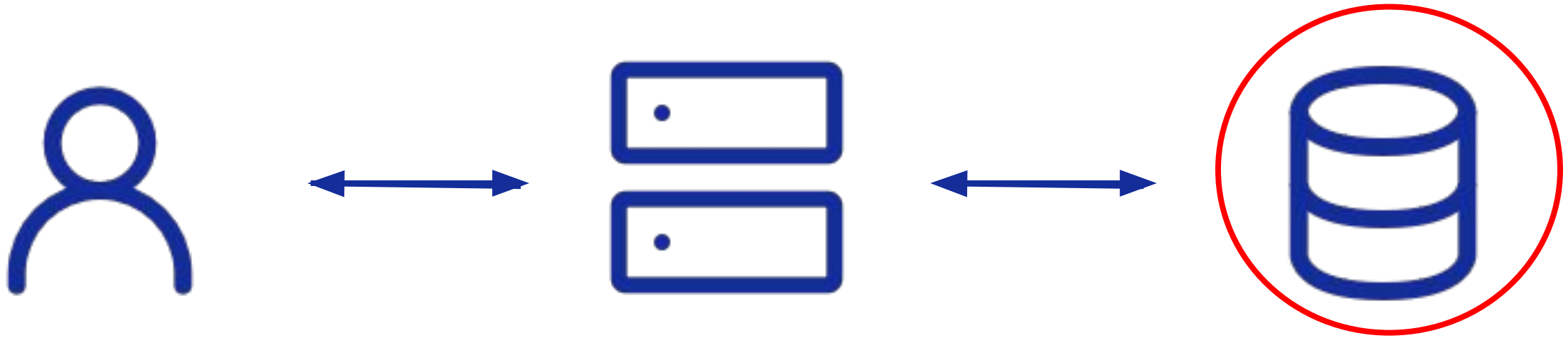
product where id = 1

```
→  db pgbench -S -T 30 -c 5 postgres
pgbench (18.1)
starting vacuum...end.
transaction type: <builtin: select only>
scaling factor: 1000
query mode: simple
number of clients: 5
number of threads: 1
maximum number of tries: 1
duration: 30 s
number of transactions actually processed: 1386369
number of failed transactions: 0 (0.000%)
latency average = 0.108 ms
initial connection time = 24.571 ms
tps = 46249.989116 (without initial connection time)
→  db
```

Где проблема?

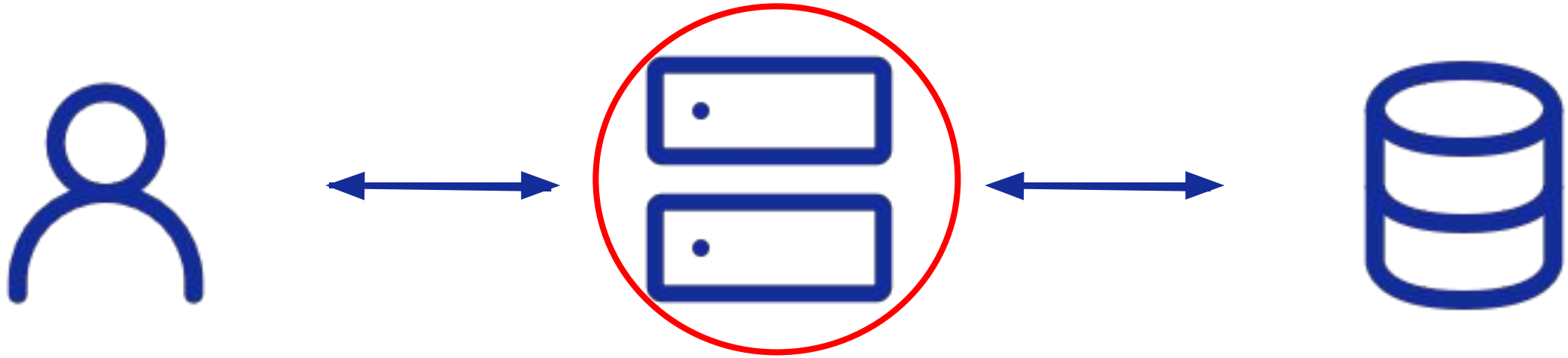
Поиск проблемы

PostgresPro



Поиск проблемы

PostgresPro

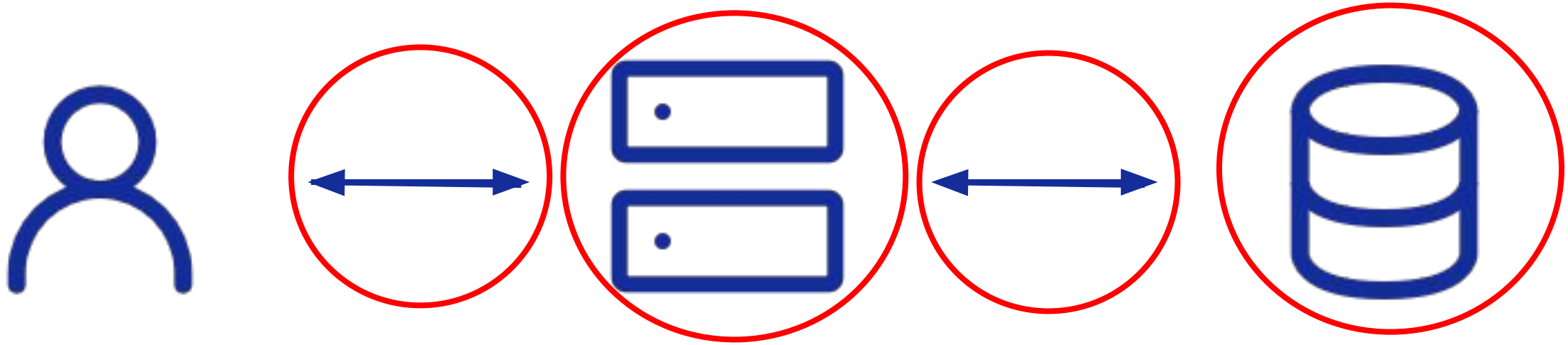


Поиск проблемы

PostgresPro

Поиск проблемы

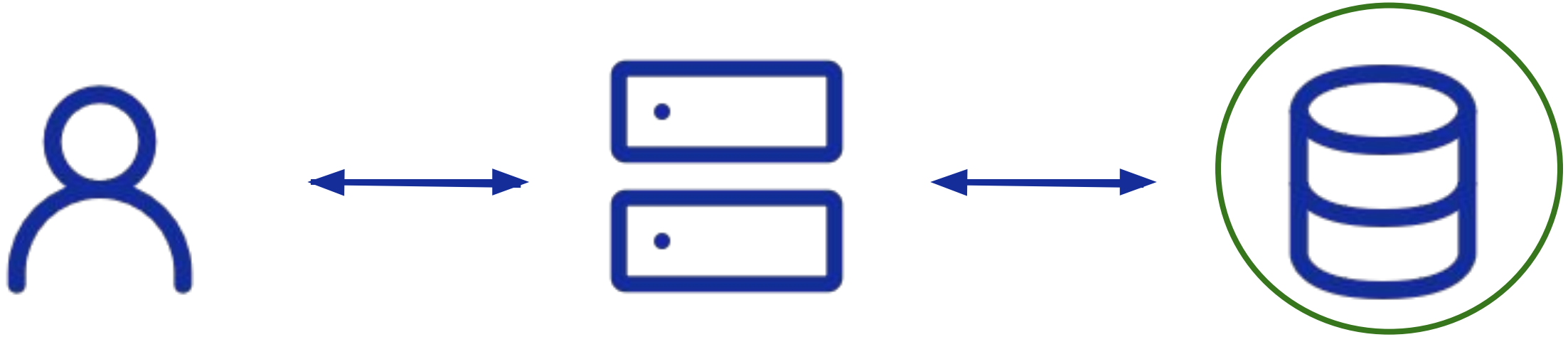
PostgresPro



Можно ли избежать этих проблем?

Решение проблемы

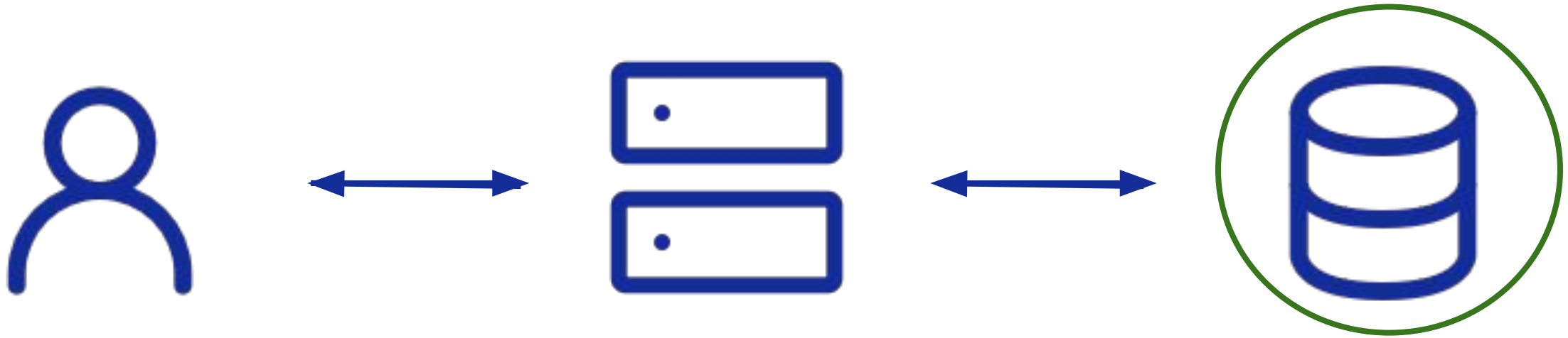
PostgresPro



Дай мне
товар 1

Решение проблемы

PostgresPro

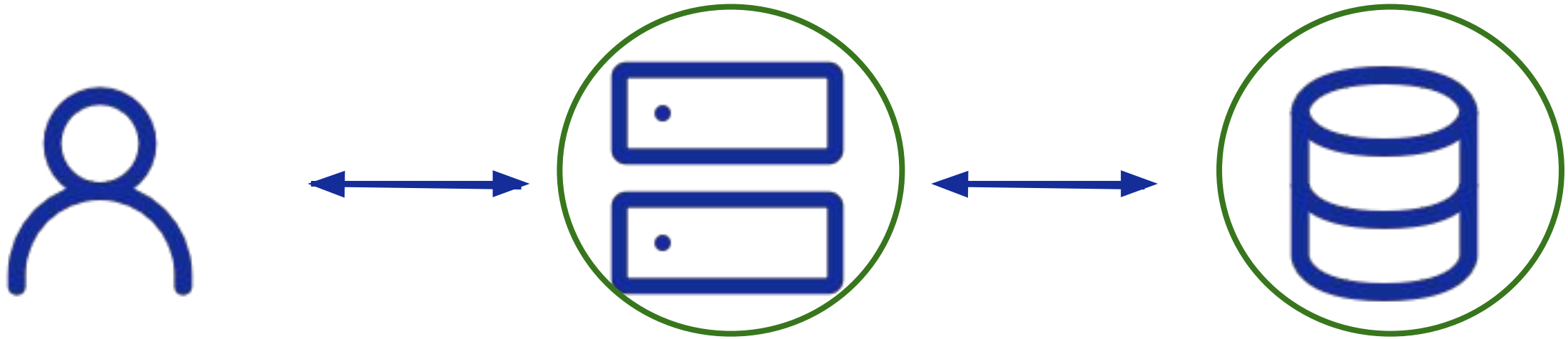


Дай мне
товар 1

Select from table
where id = 1

Решение проблемы

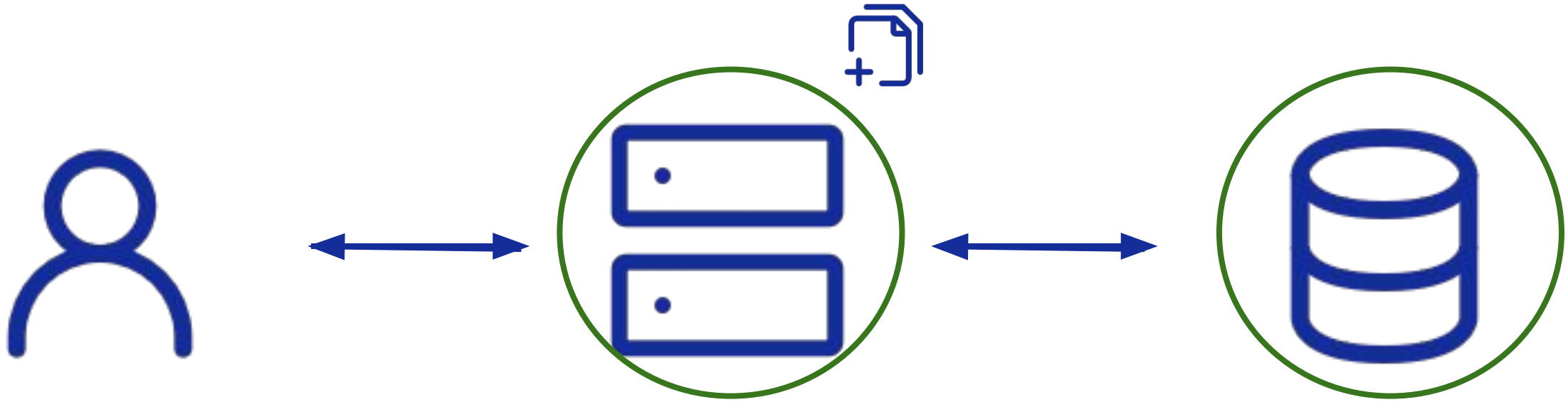
PostgresPro



product where id = 1

Решение проблемы

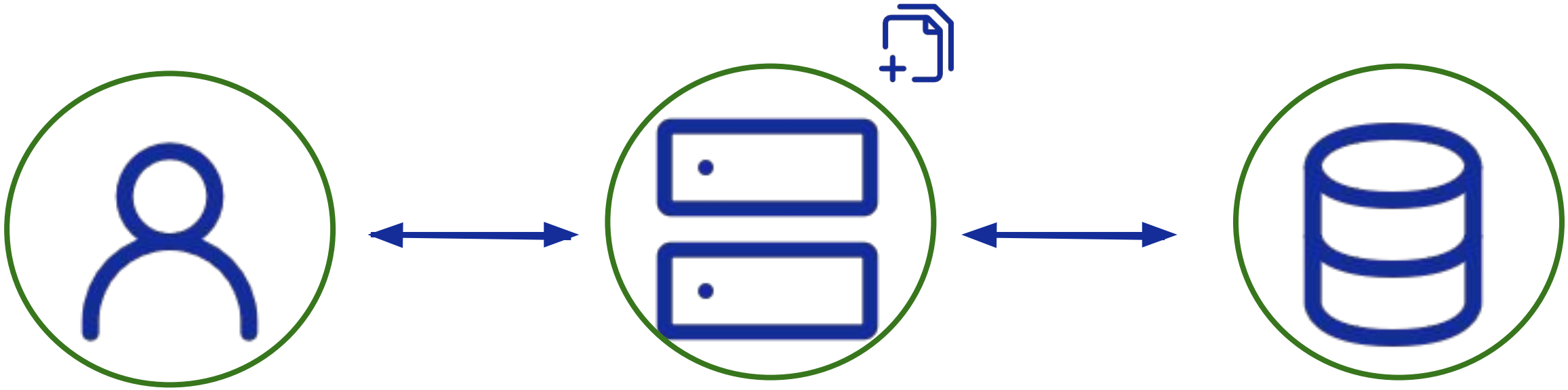
PostgresPro



product where id = 1

Решение проблемы

PostgresPro

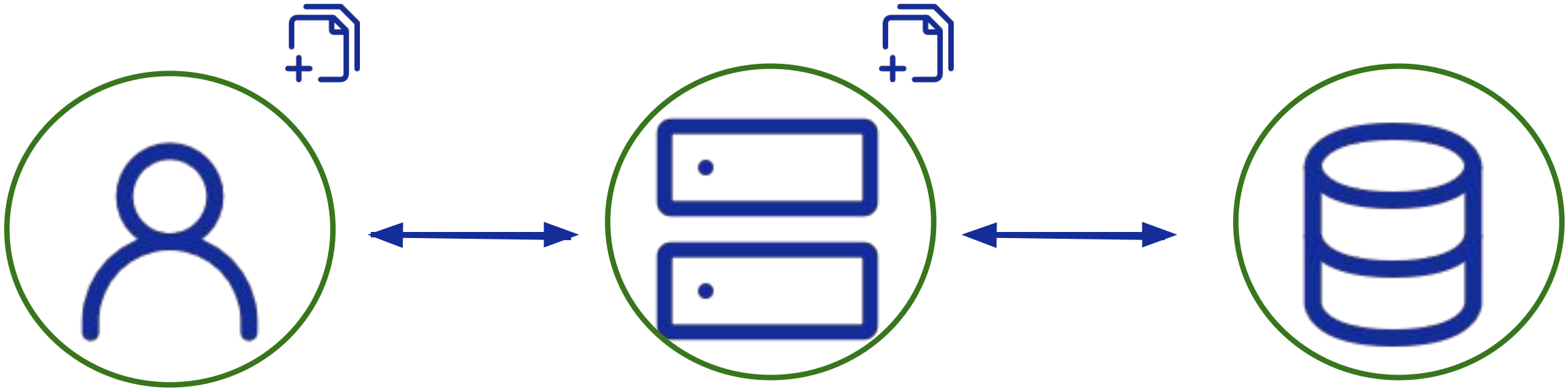


json:
id :1
des: tovar …

product where id = 1

Решение проблемы

PostgresPro

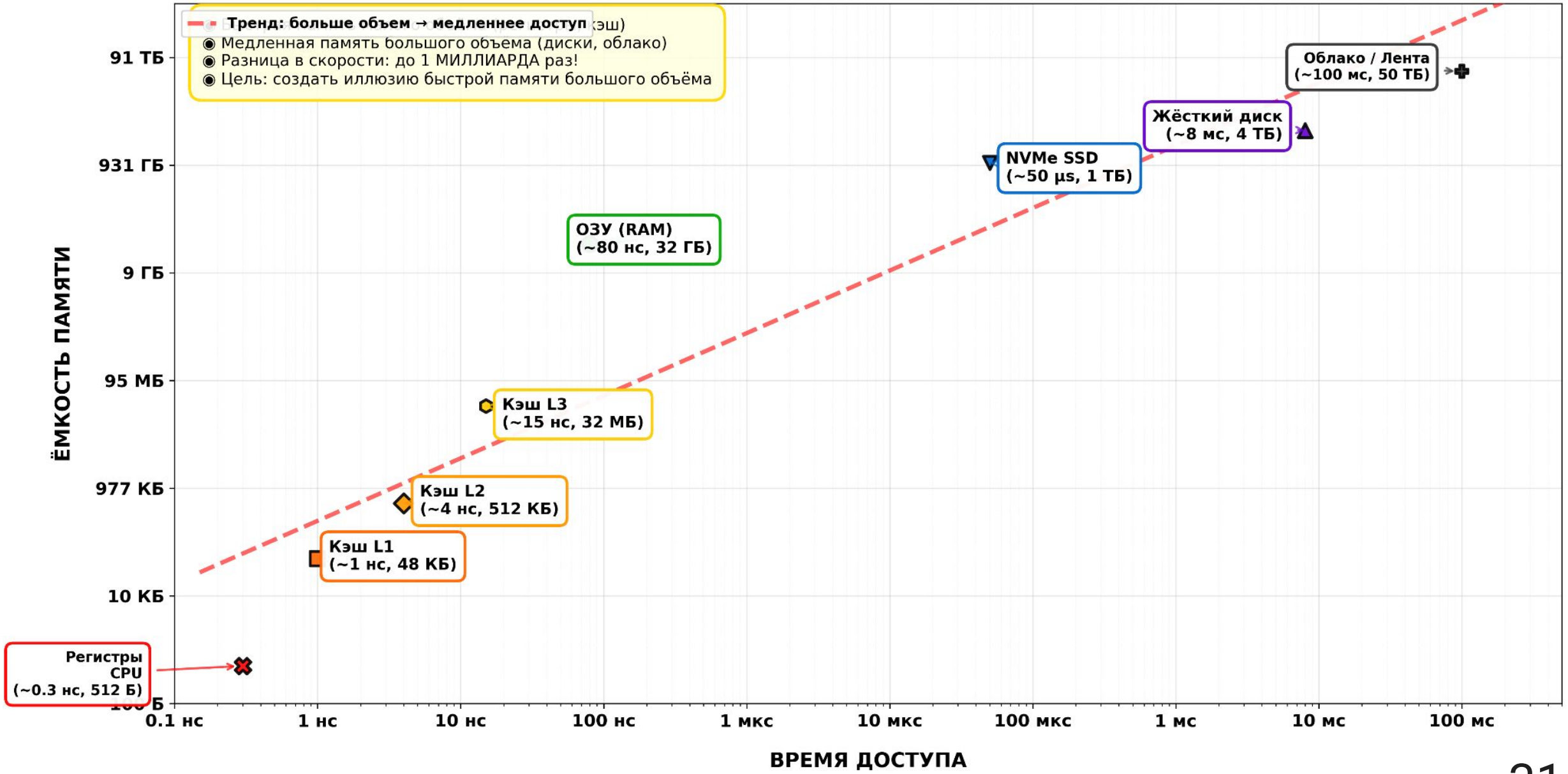


json:
id :1
des: tovar …

product where id = 1

Теория кэширования

ИЕРАРХИЯ ПАМЯТИ КОМПЬЮТЕРА: КОМПРОМИСС МЕЖДУ БЫСТРОДЕЙСТВИЕМ И ОБЪЁМОМ



Какие бывают кэши?

Схема работы с кэшом

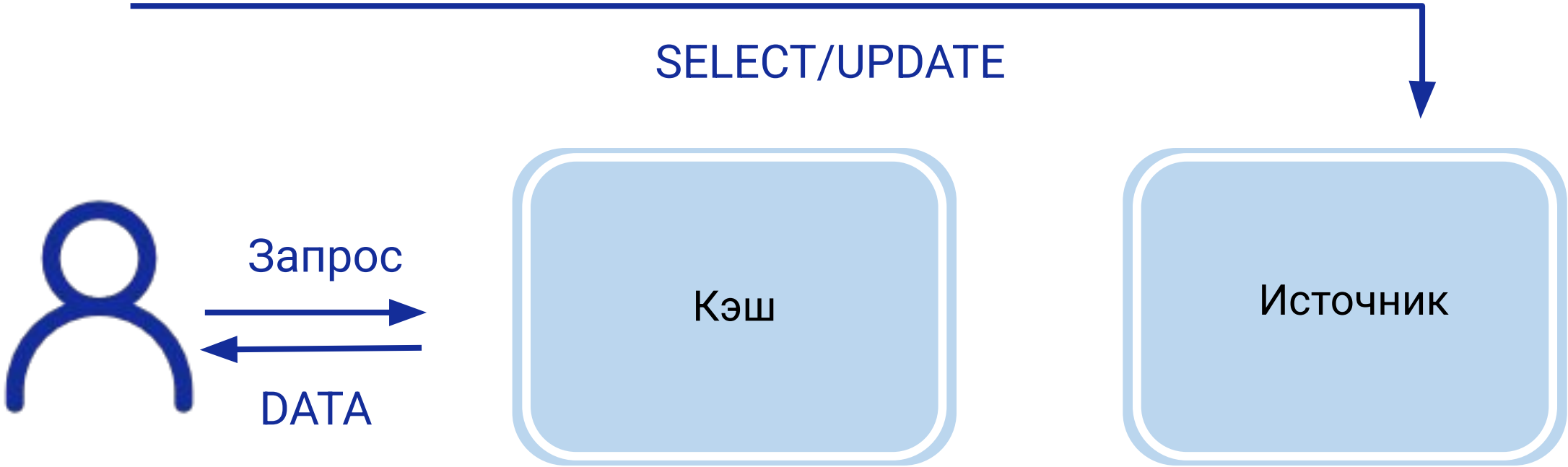
PostgresPro

Кэш

Источник

Cache Aside

PostgresPro



Плюсы и минусы Cache aside

PostgresPro

+ Кэш и источник легко использовать по-отдельности	— Усложнение бизнес логики прикладного приложения
+ Проще по своему устройству	— Кэш ничего не знает про источник данных
+ Гибче в использовании	

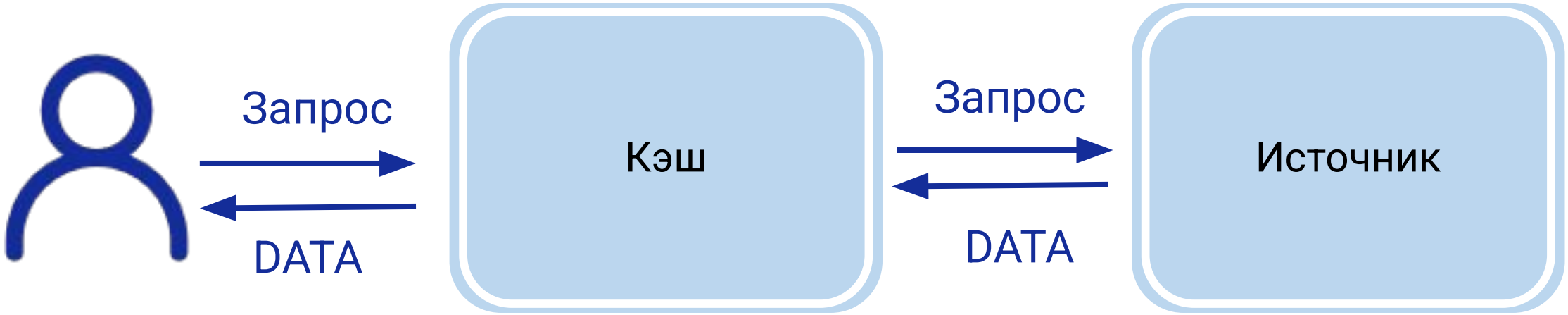
Примеры Cache aside

PostgresPro

- Redis
- garnet
- keyvalueDB
- Memcached
- java: Caffeine

Cache through

PostgresPro



Плюсы и минусы Cache through

PostgresPro

+ Простая бизнес логика	— Сложнее использовать кэш и источник по отдельности
+ Меньше администрируемых единиц	— У пользователя меньше контроля
+ Потенциальная оптимизация сетевых ресурсов	

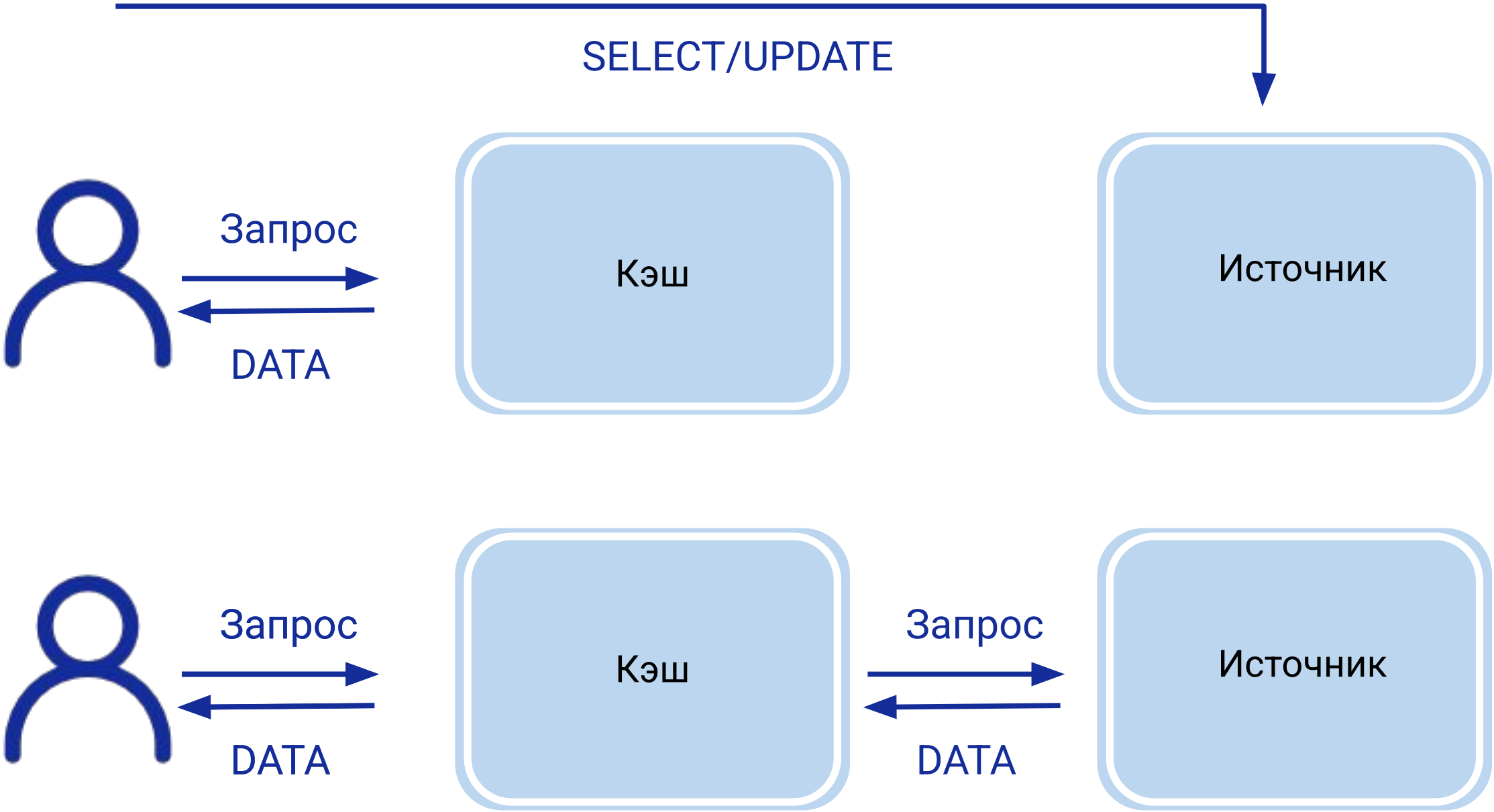
Примеры Cache through

PostgresPro

- Кэши процессора
- java: Caffeine с CacheProvider
- Декоратор @lru_cache в python
- CDN (Content Delivery Network)

Сравнение стратегий

PostgresPro



Проблемы кэширования

Проблема вытеснения

Схема работы с кэшом

PostgresPro

Кэш

Источник

Схема работы с кэшом

PostgresPro

Кэш

Источник

Вытеснение. Алгоритм Белади

PostgresPro

3 с

100 мс

1 день

3 часа

2 года

16 минут

Вытеснение. Рабочие алгоритмы

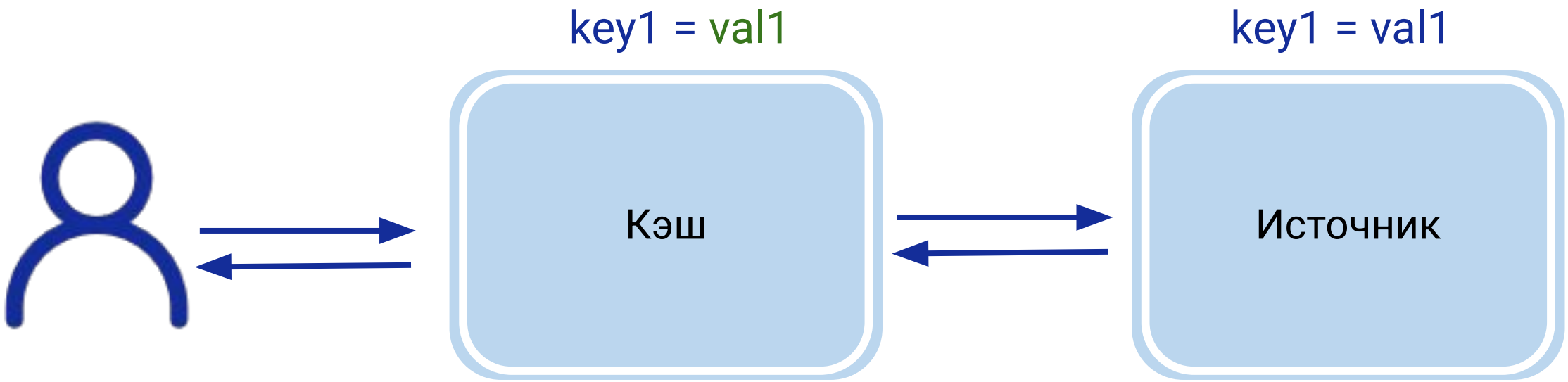
PostgresPro

- Random
- FIFO (First-In, First-Out) – последний пришедший
- LIFO (Last In, First Out) – последняя добавленная
- LRU (Least Recently Used) – наиболее давно не используемый
- MRU (Most Recently Used) – последний используемый
- LFU (Least Frequently Used) – наименее часто используемый
- Clock

Проблема инвалидации

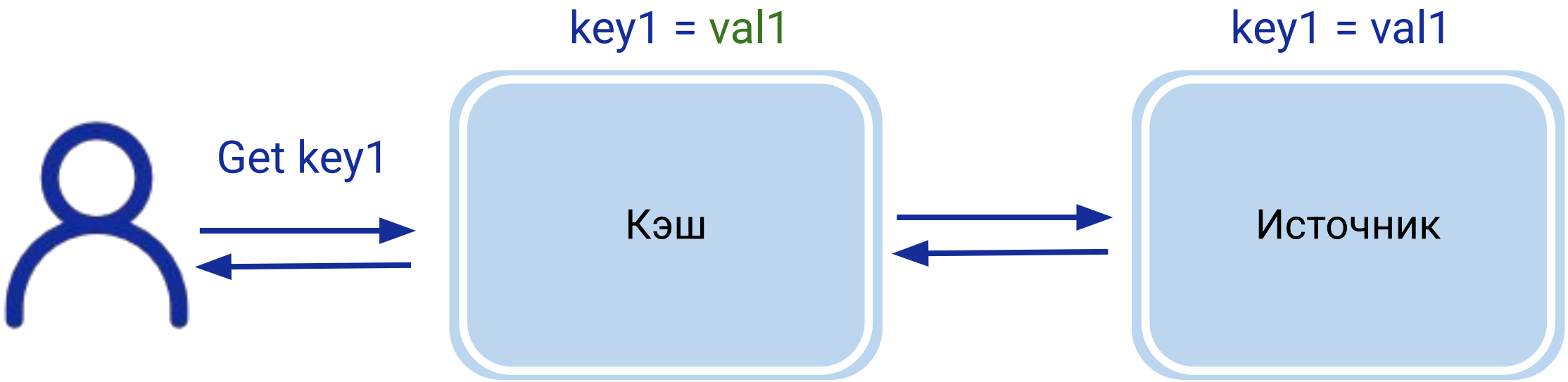
Инвалидация

PostgresPro



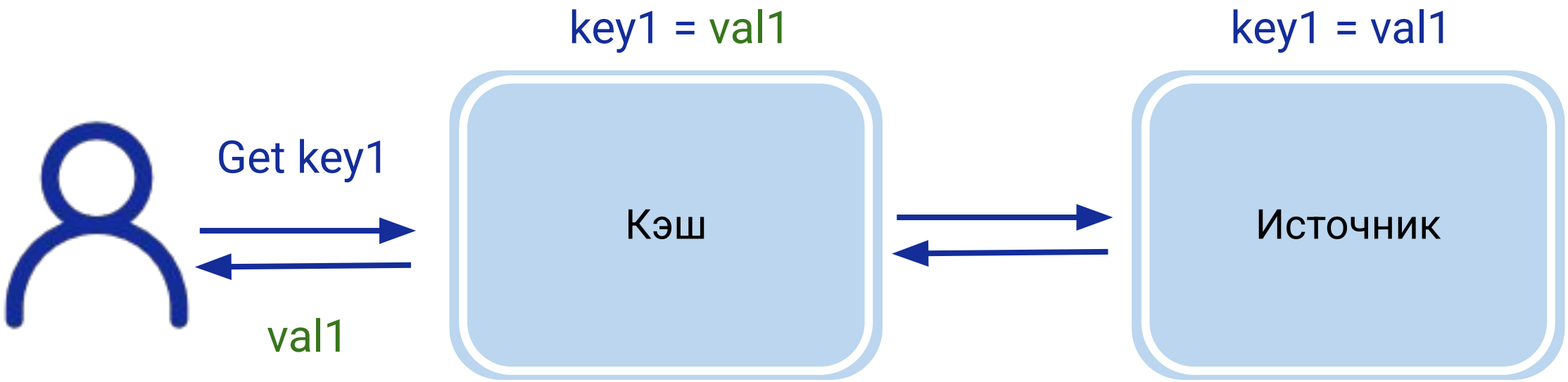
Инвалидация

PostgresPro



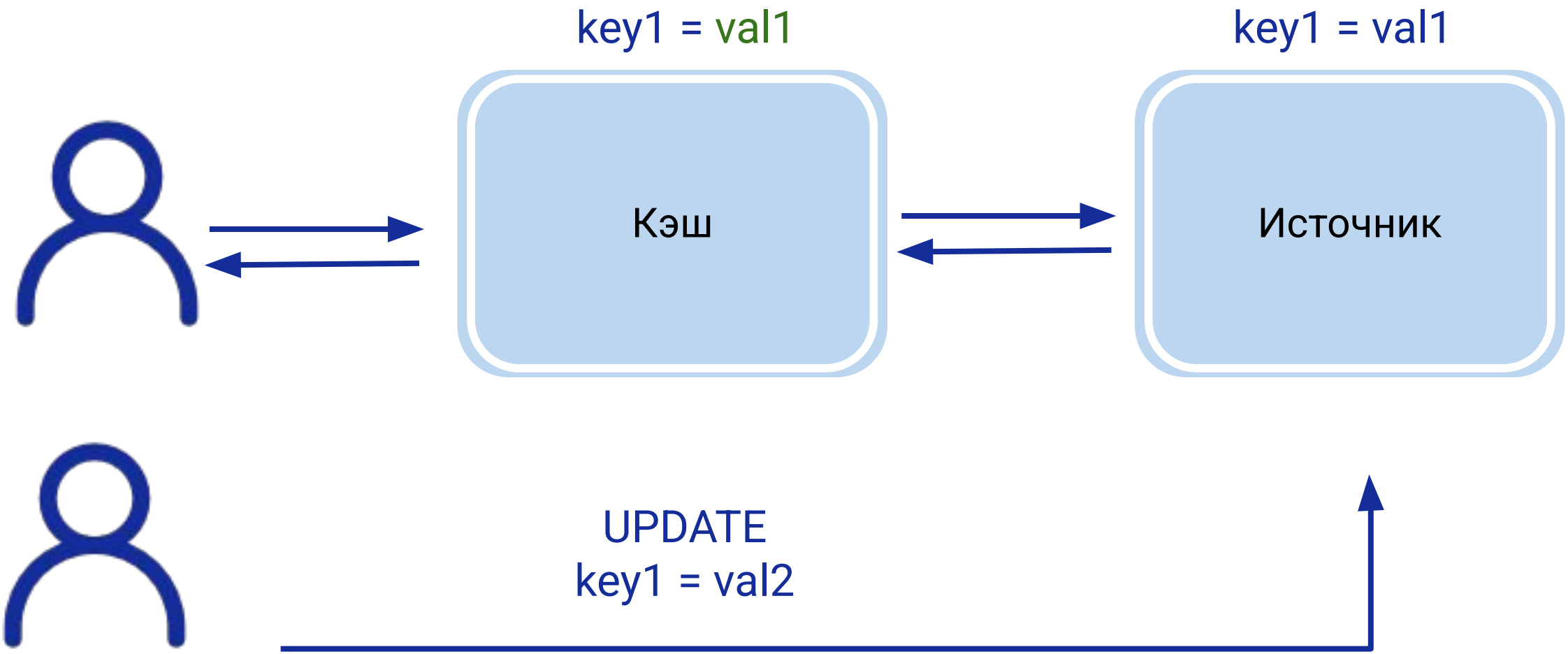
Инвалидация

PostgresPro



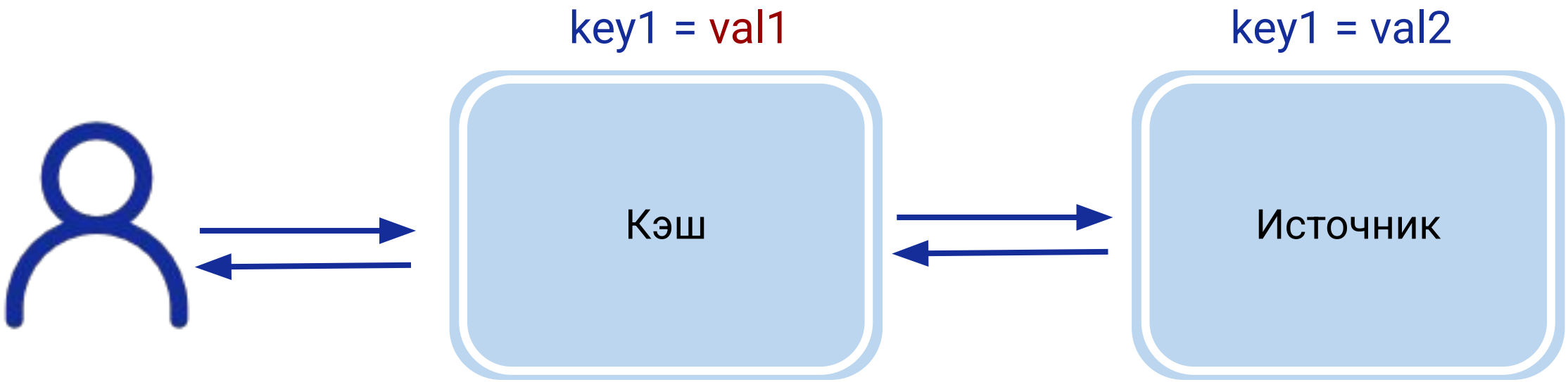
Инвалидация

PostgresPro



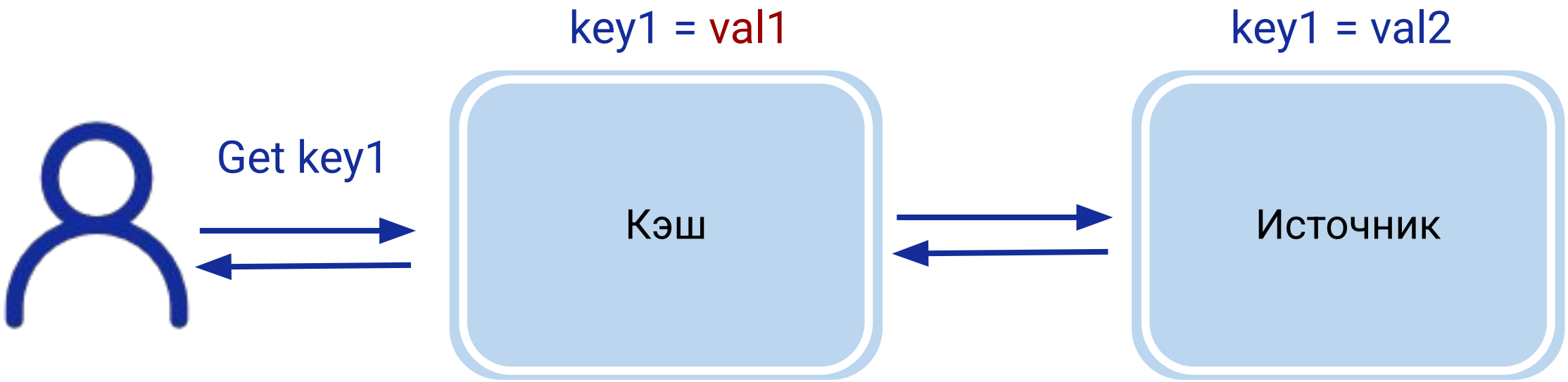
Инвалидация

PostgresPro



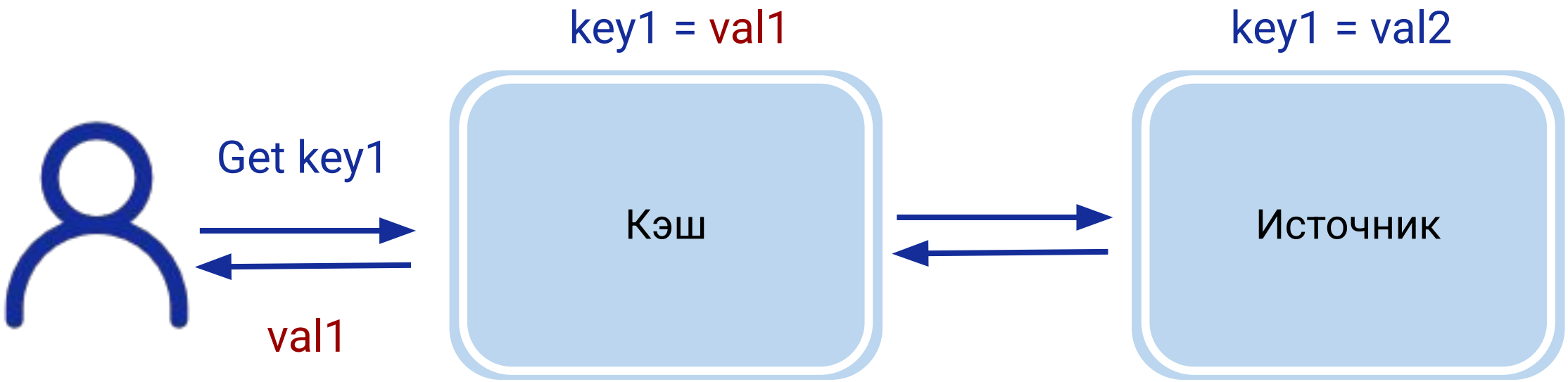
Инвалидация

PostgresPro



Инвалидация

PostgresPro

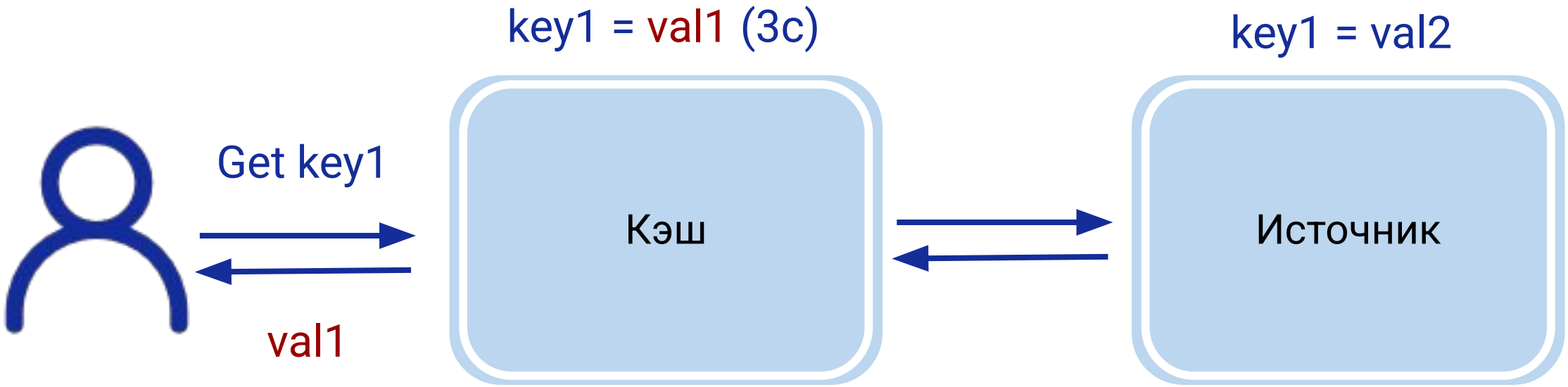


Как решить эту проблему?

Инвалидация по времени

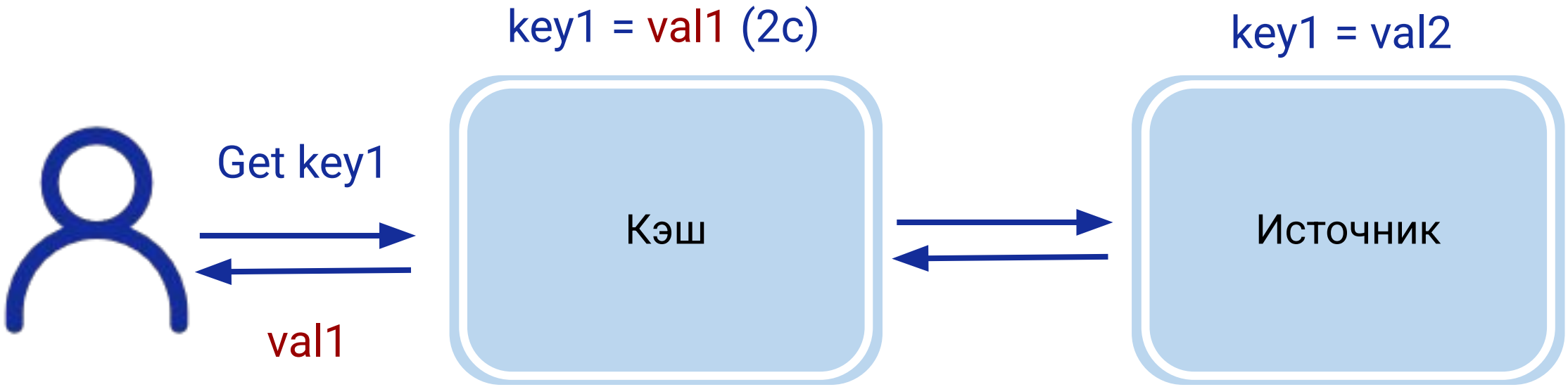
Инвалидация по времени

PostgresPro



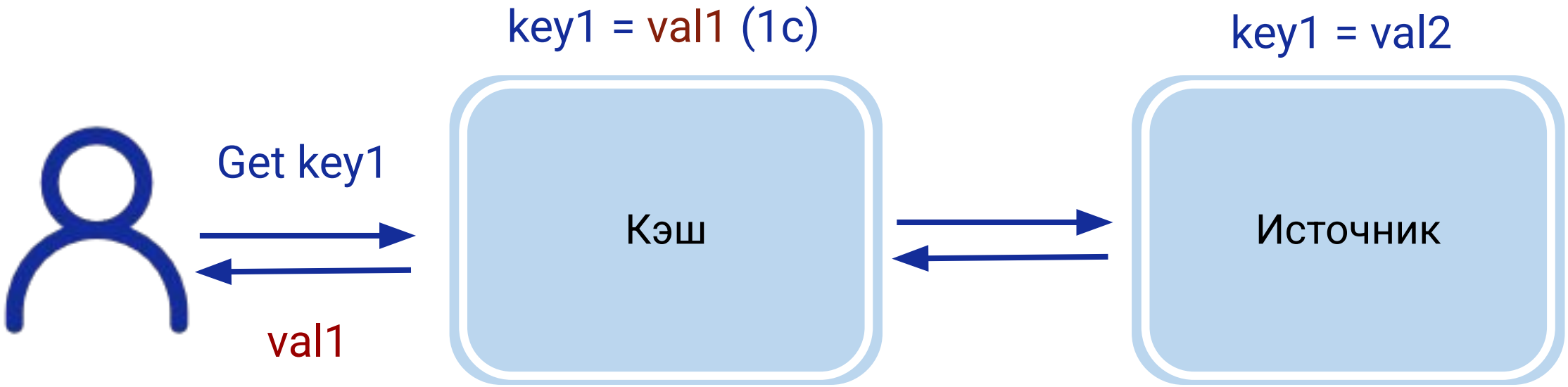
Инвалидация по времени

PostgresPro



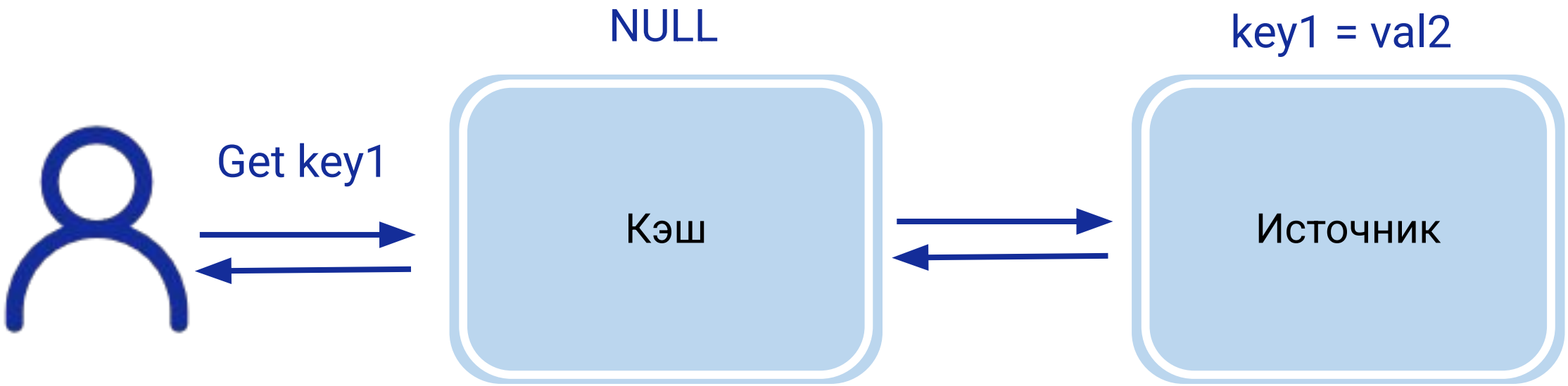
Инвалидация по времени

PostgresPro



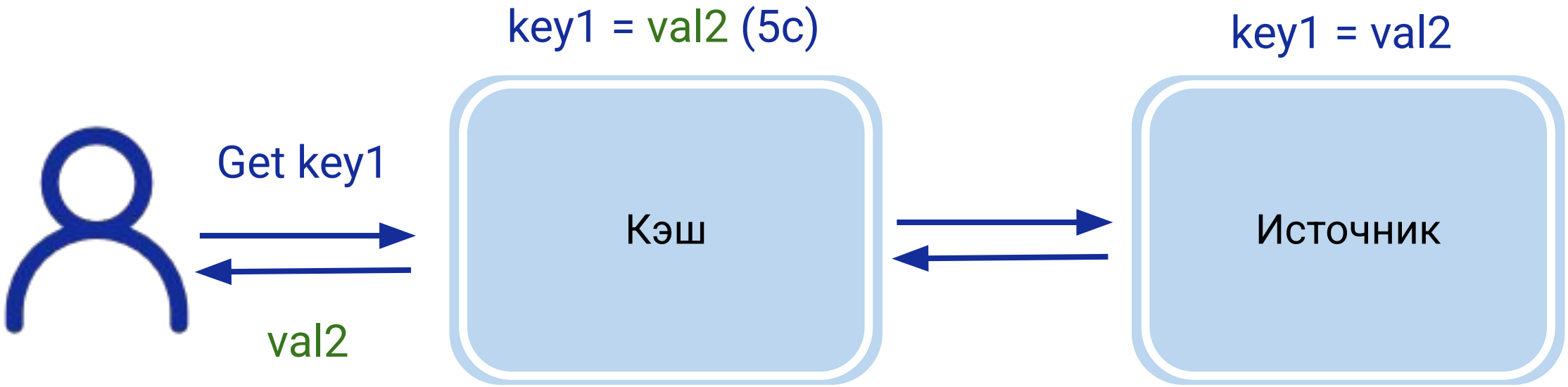
Инвалидация по времени

PostgresPro



Инвалидация по времени

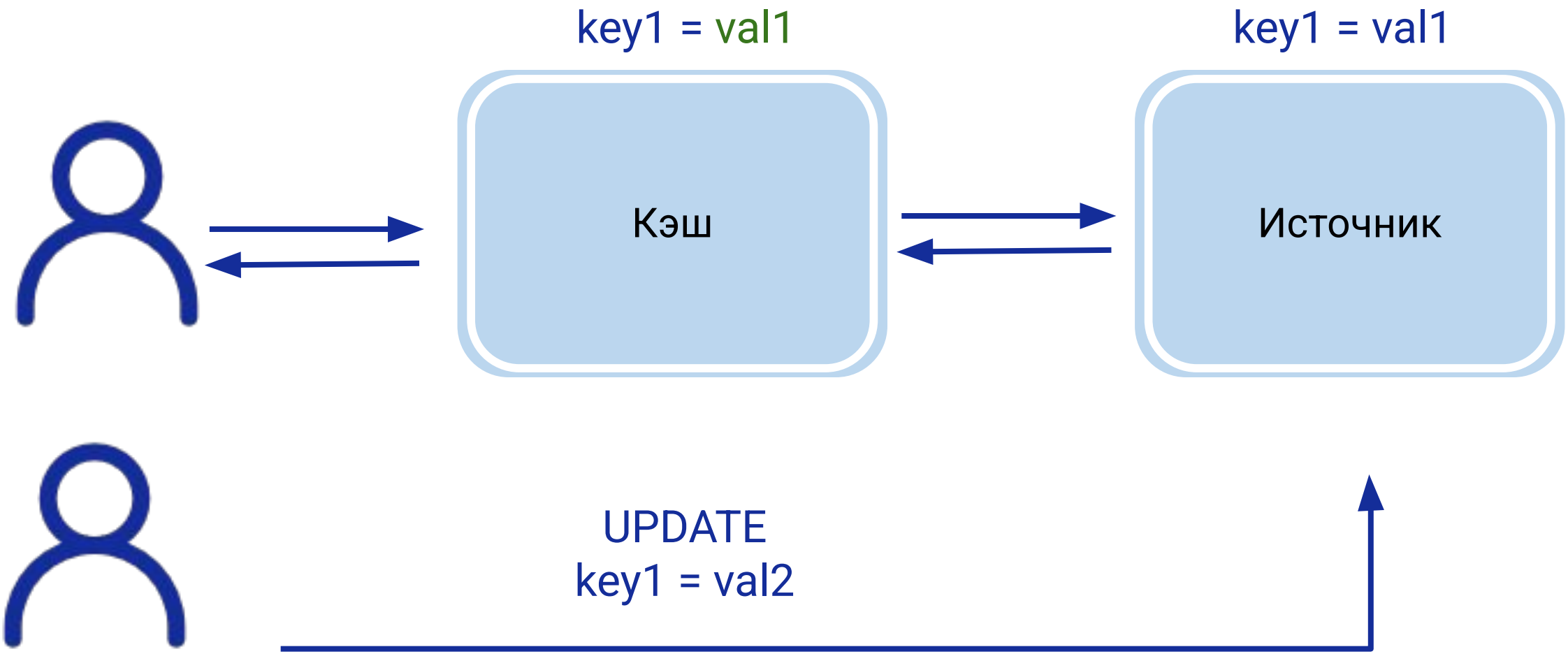
PostgresPro



Инвалидация по событию

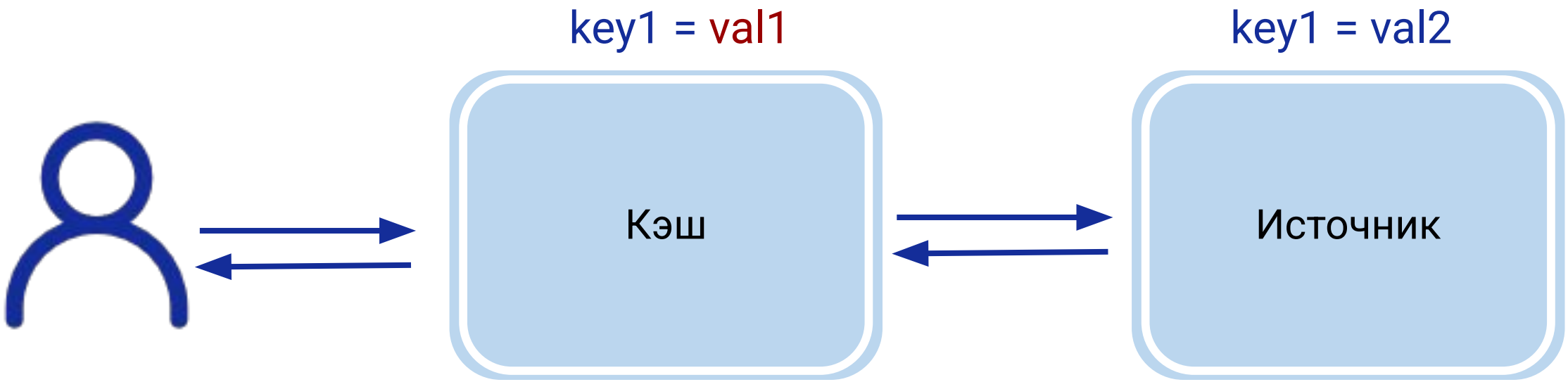
Инвалидация по событию

PostgresPro



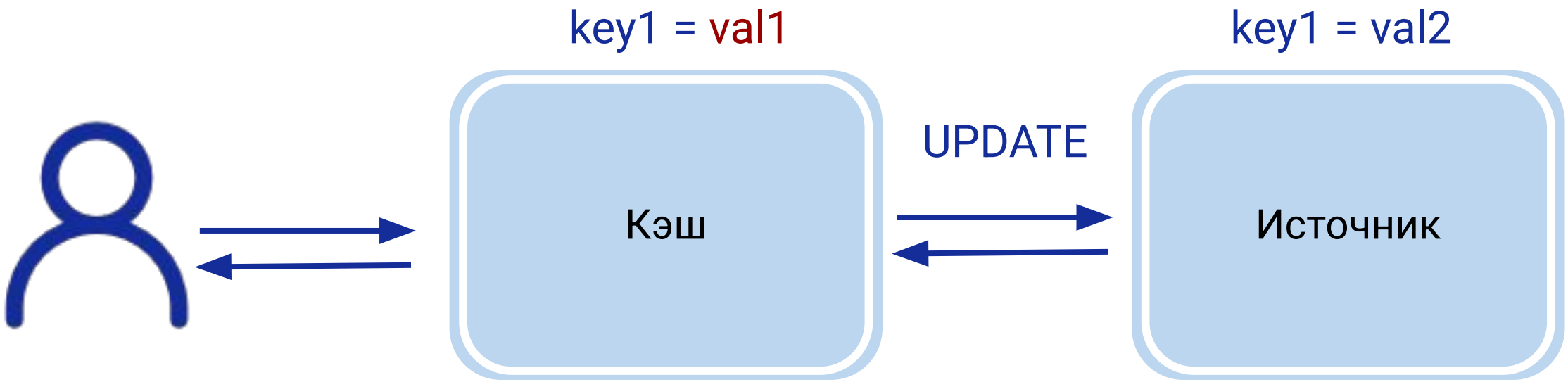
Инвалидация по событию

PostgresPro



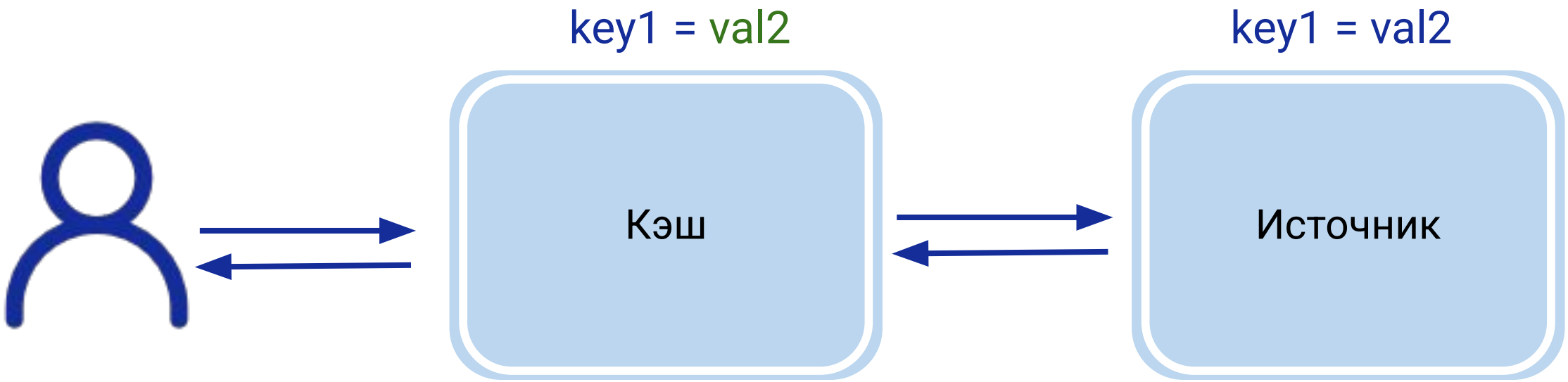
Инвалидация по событию

PostgresPro



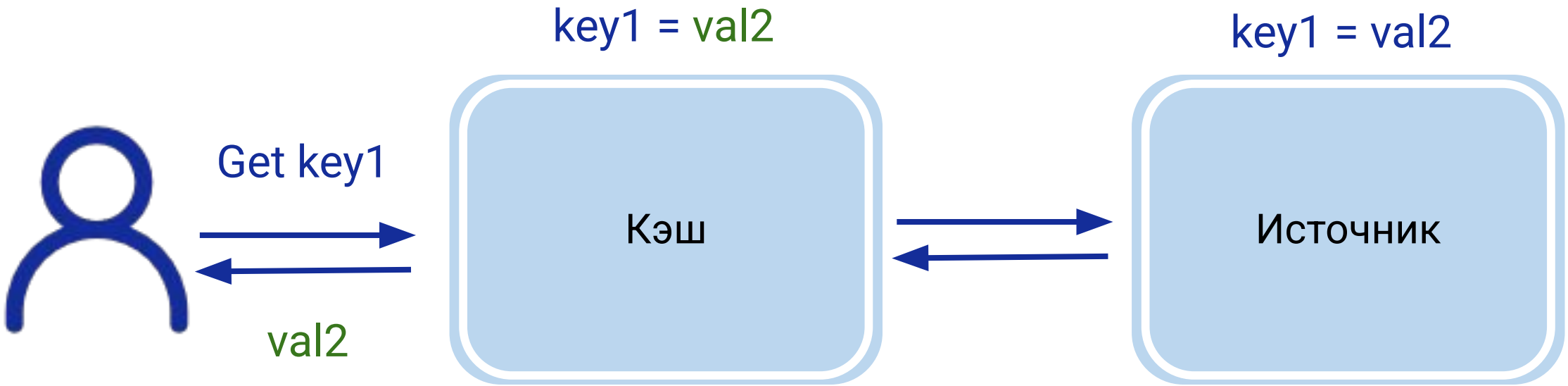
Инвалидация по событию

PostgresPro



Инвалидация по событию

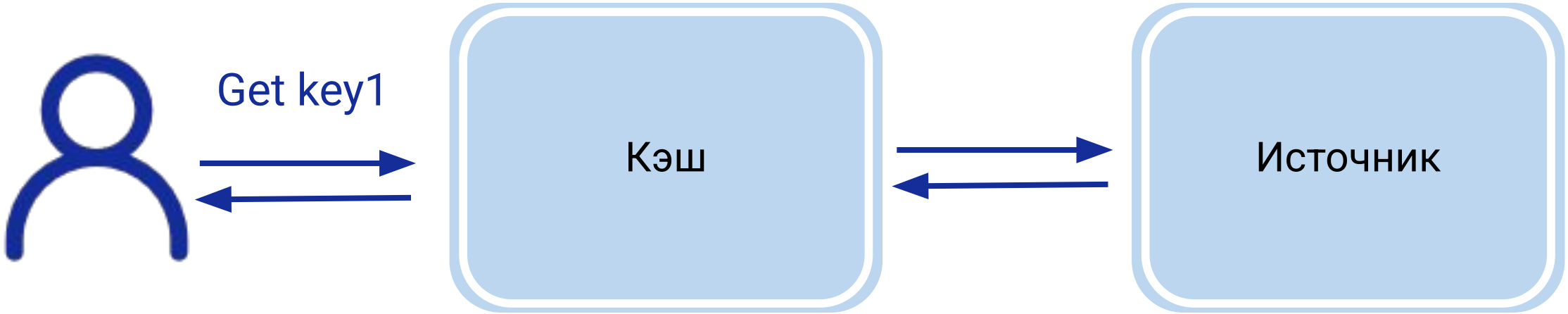
PostgresPro



Проблема формирования ключа

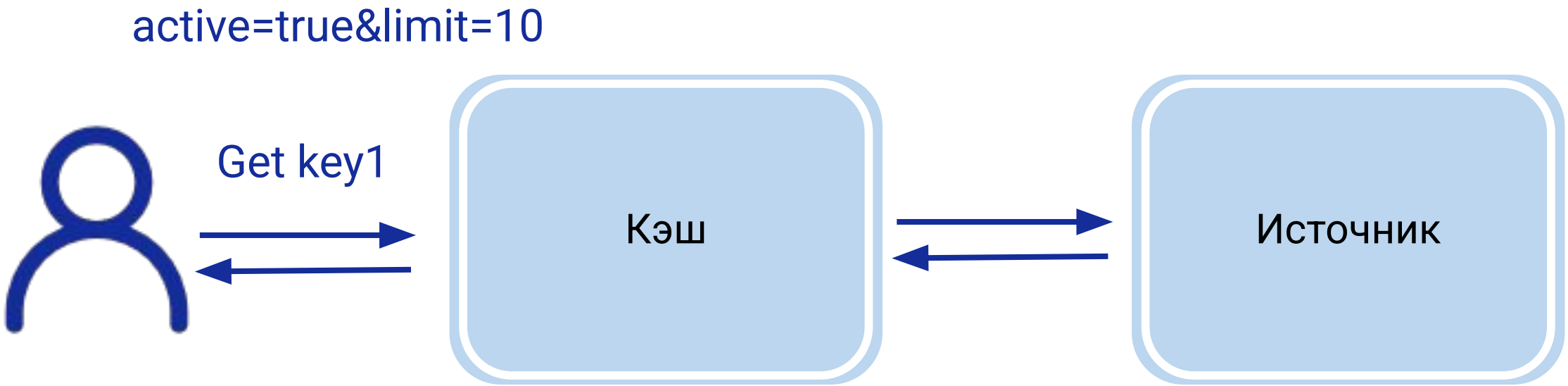
Формирование ключа

PostgresPro



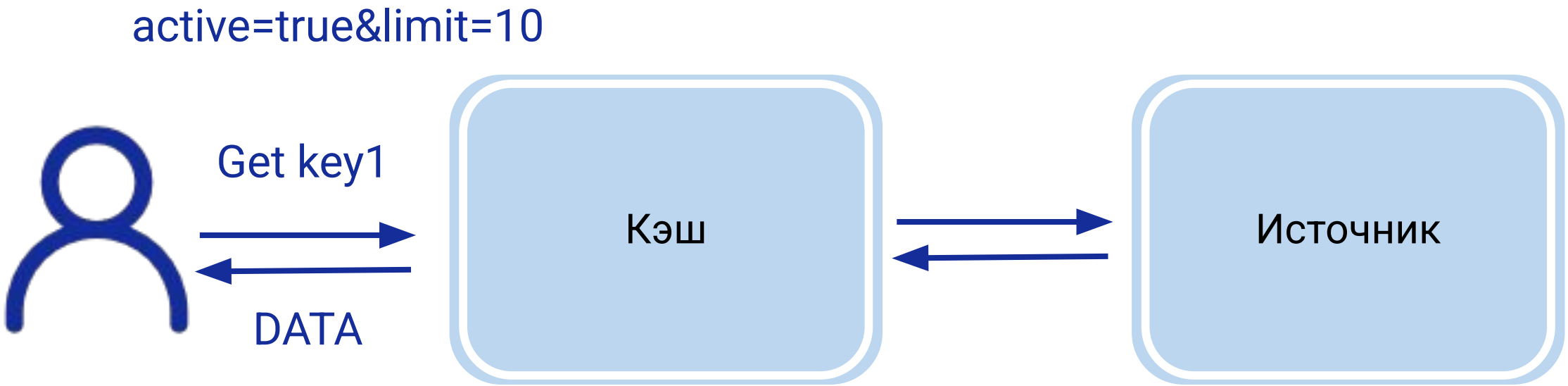
Формирование ключа

PostgresPro



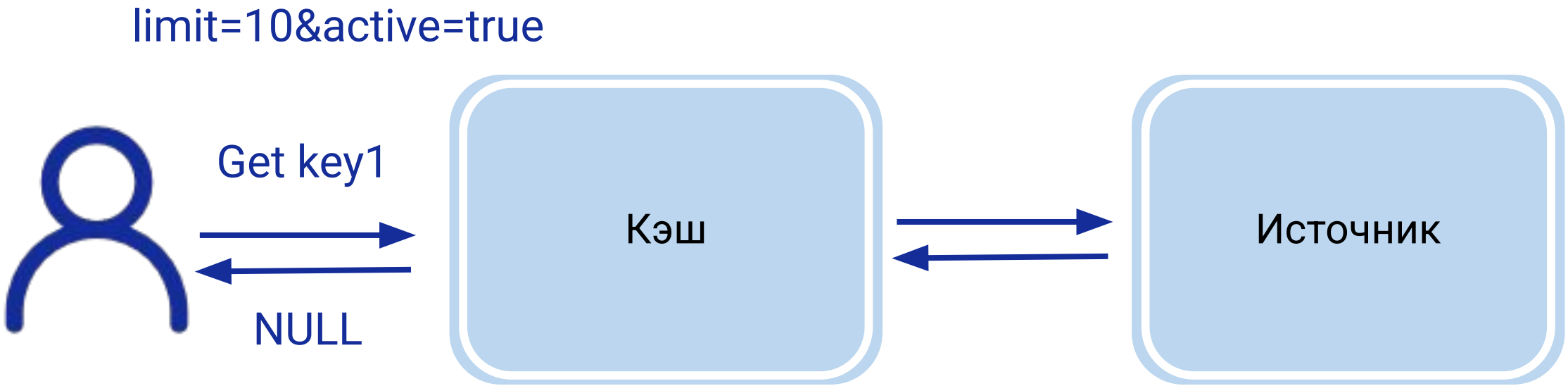
Формирование ключа

PostgresPro



Формирование ключа

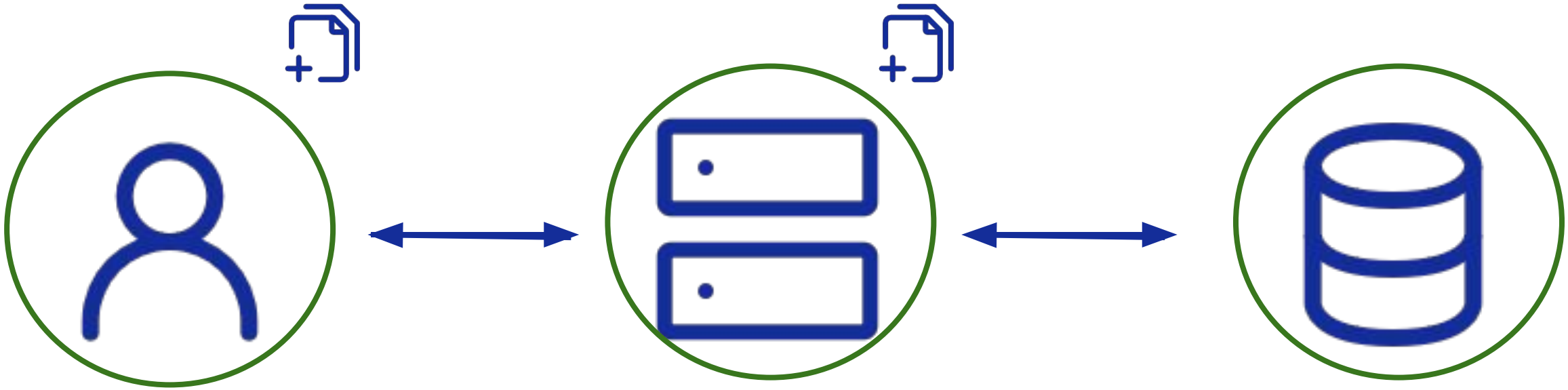
PostgresPro



Вернемся к нашей проблеме

Где взять данные?

PostgresPro



Кэширование на клиенте

Обработка запроса пользователя

PostgresPro

Дай мне
товар 1
еще раз

Обработка запроса пользователя

PostgresPro

А он уже есть!

Плюсы и минусы кэширования на клиенте

PostgresPro

+ Никаких сетевых задержек	— Пользу от кэширования получает только один клиент
+ Снижение нагрузки на сервер	— Проблема инвалидации данных
+ Работа оффлайн	

Когда стоит использовать кэширование на клиенте

PostgresPro

- Доступ к данным нужен всегда, даже без сети
- Один и тот же пользователь обращается к одним и тем же данным

Когда не стоит использовать кэширование на клиенте

PostgresPro

- Кэшировать нужно большой объем информации
- Разные пользователи обращаются к одному подмножеству данных

Примеры кэширования на клиенте

- Браузер
- Мобильные приложения
- Декоратор @lru_cache в python
- Любая самописная логика на стороне клиентского приложения

Примеры кэширования на клиенте

- Браузер
- Мобильные приложения
- Декоратор @lru_cache в python
- Любая самописная логика на стороне клиентского приложения
- Сервисы заметок

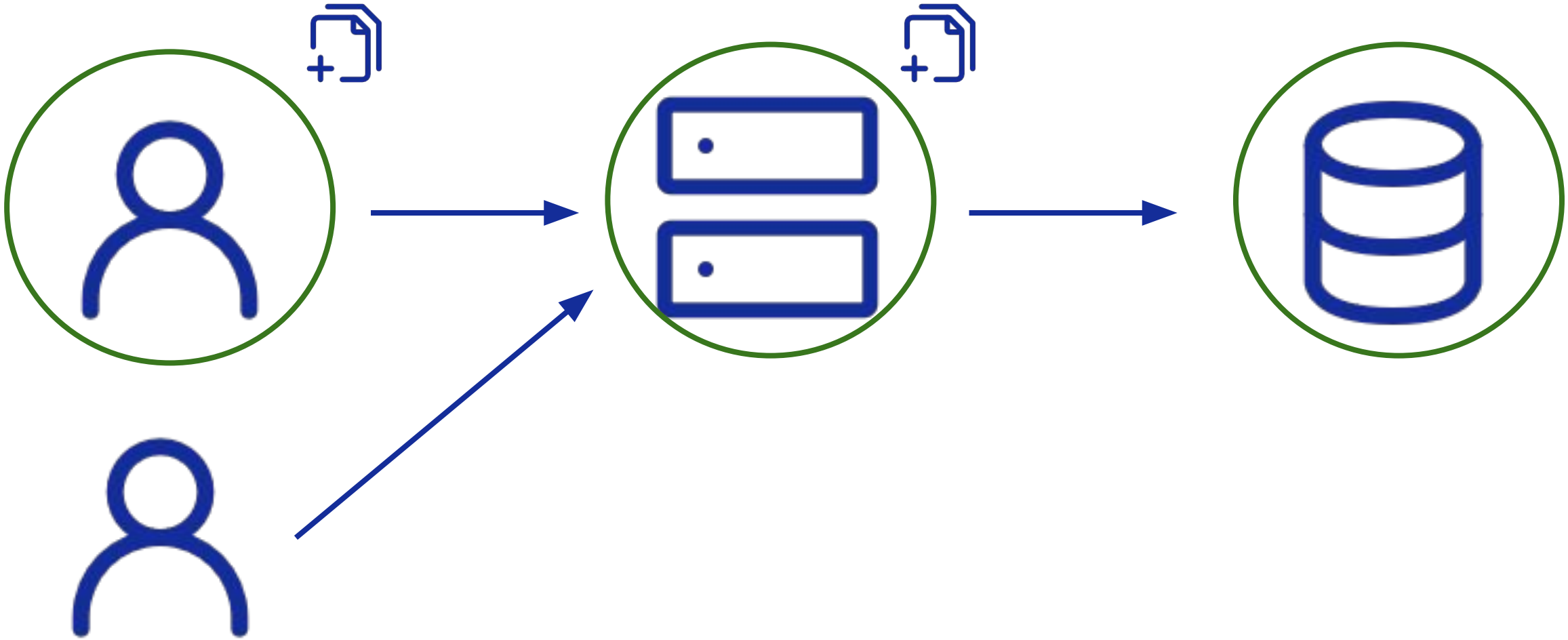
Кэширование на сервере

Обработка запроса пользователя

PostgresPro

Обработка запроса пользователя

PostgresPro



Плюсы и минусы кэширования на сервера

PostgresPro

+ Разные клиенты влияют друг на друга	— Разные клиент влияют друг на друга
+ Снижение нагрузки на СУБД	— Проблема инвалидации данных
+ Больше мощности, относительно клиента	

Когда стоит использовать кэширование на сервере

PostgresPro

- Необходима поддержка работы с несколькими пользователями
- Данные специфичные для экземпляра приложения

Когда не стоит использовать кэширование на сервере

PostgresPro

- Распределенные системы с требованием консистентности
- Большой объем данных, которые нужно кэшировать
- Нужно масштабировать кэш
- Нужно эффективное использование кэша
- Нужны сложная логика взаимодействия с кэшом

Примеры кэширования на сервере

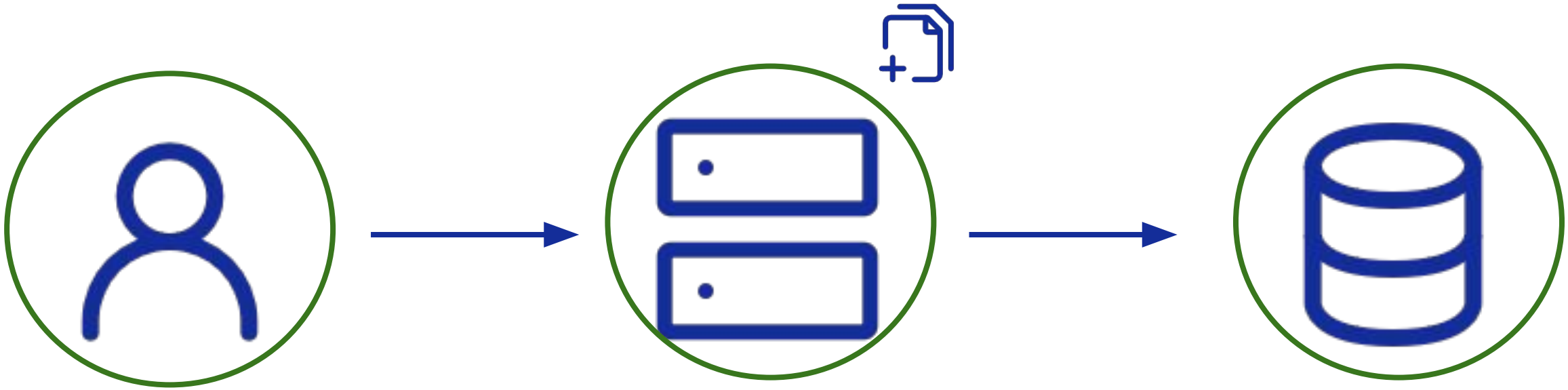
PostgresPro

- Java:Caffeine - Apache Kafka, Spring Boot приложения
- Go:Ristretto - распределенная база данных Dgraph
- Python:Theine - пока не получила широкого применения
- Кэш маршрутизатора - локальность данных

Кэширование на стороннем сервисе

Обработка запроса пользователя

PostgresPro

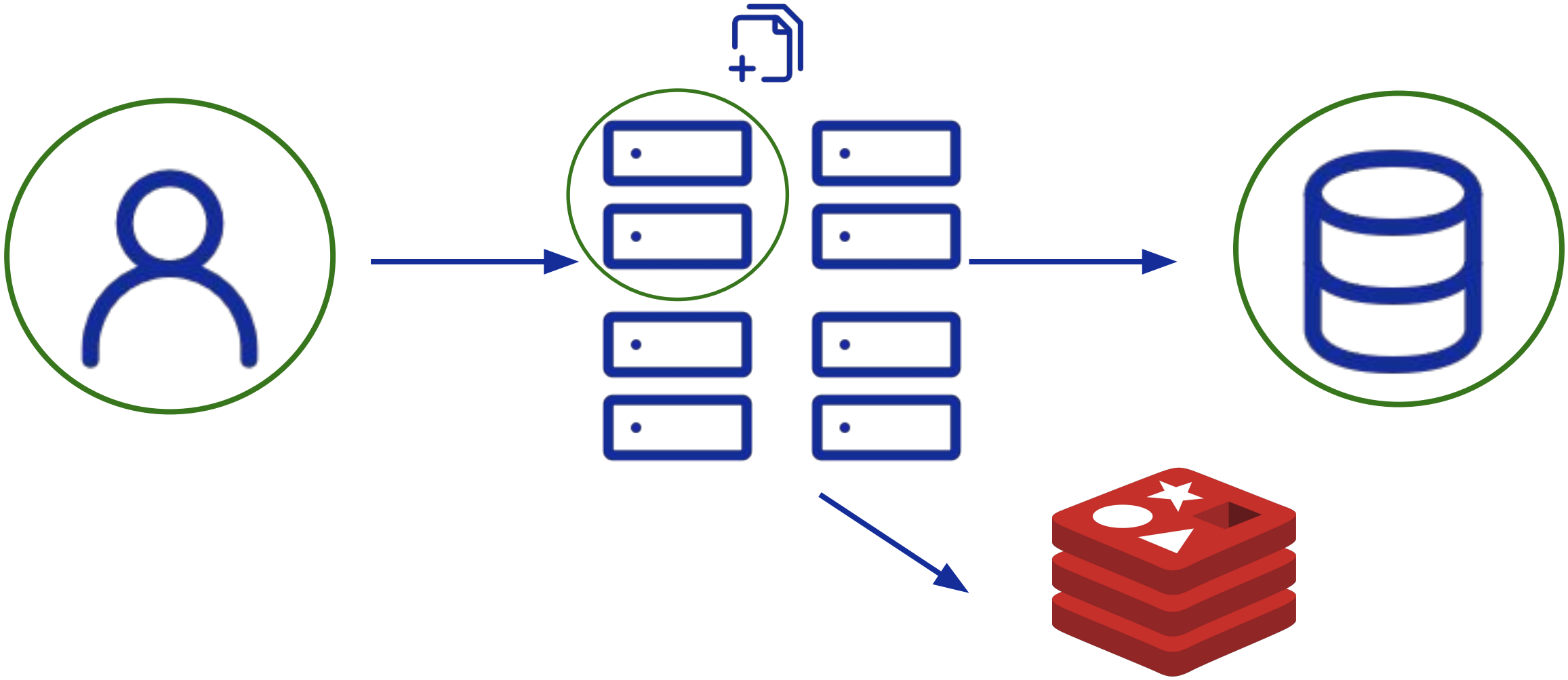


Обработка запроса пользователя

PostgresPro

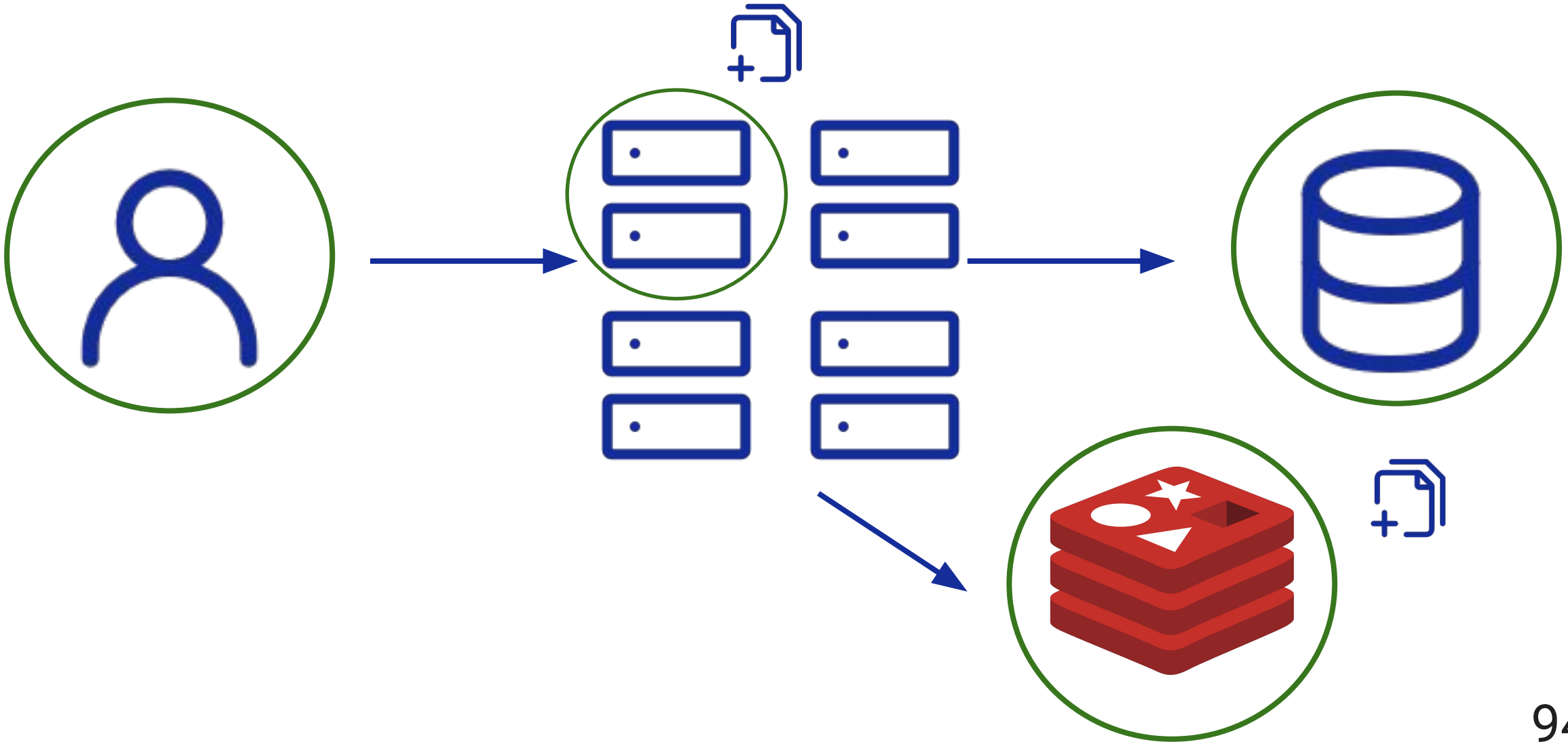
Обработка запроса пользователя

PostgresPro



Обработка запроса пользователя

PostgresPro



Плюсы и минусы кэширования на стороннем сервера

PostgresPro

+ Встроенные сложные типы данных	— Дополнительная точка отказа и администрирования
+ Встроенная репликация, шардирование, персистентность	— Проблема инвалидации данных
+ Кэш не зависит от приложения и его конкретной версии	— Дополнительное сетевое взаимодействие

Когда стоит использовать кэширование на стороннем сервере

PostgresPro

- Есть дополнительные ресурсы на сервер
- Требуются сложные структуры данных
- Требуется распределенность, шардирование, персистентность
- Требуется согласованный кэш

Когда не стоит использовать кэширование на стороннем сервере

PostgresPro

- Простые приложения
- Требуется сильная консистентность данных в кэше и СУБД

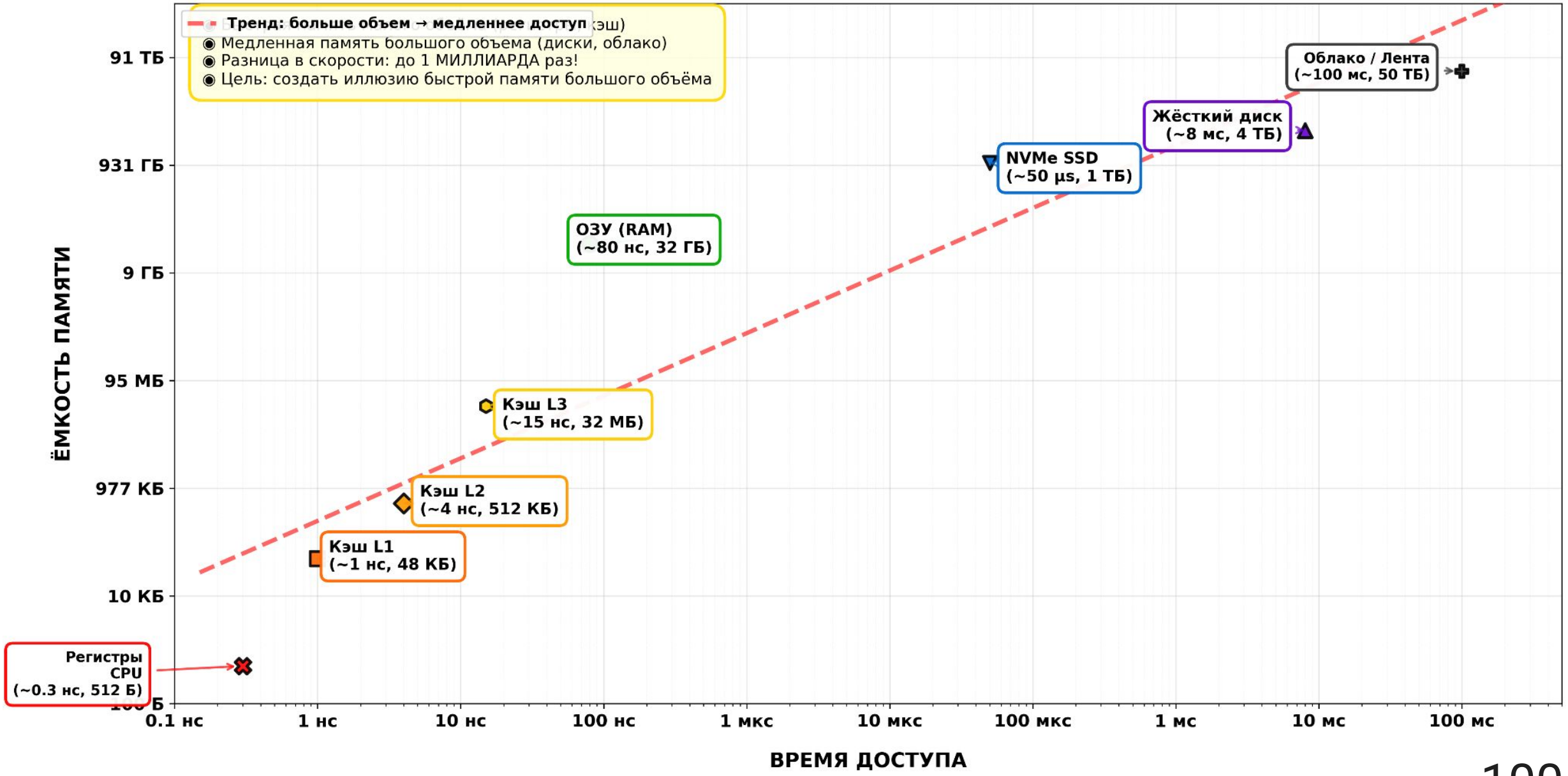
Примеры кэширования на стороннем сервере

PostgresPro

- Redis - гибкий кэш, брокер сообщений
- memcached - простое решение
- garnet - высокая производительность
- tarantool - гибридная in-memory СУБД
- picodata - распределенная in-Memory Data Grid

Кэширование в СУБД

ИЕРАРХИЯ ПАМЯТИ КОМПЬЮТЕРА: КОМПРОМИСС МЕЖДУ БЫСТРОДЕЙСТВИЕМ И ОБЪЁМОМ



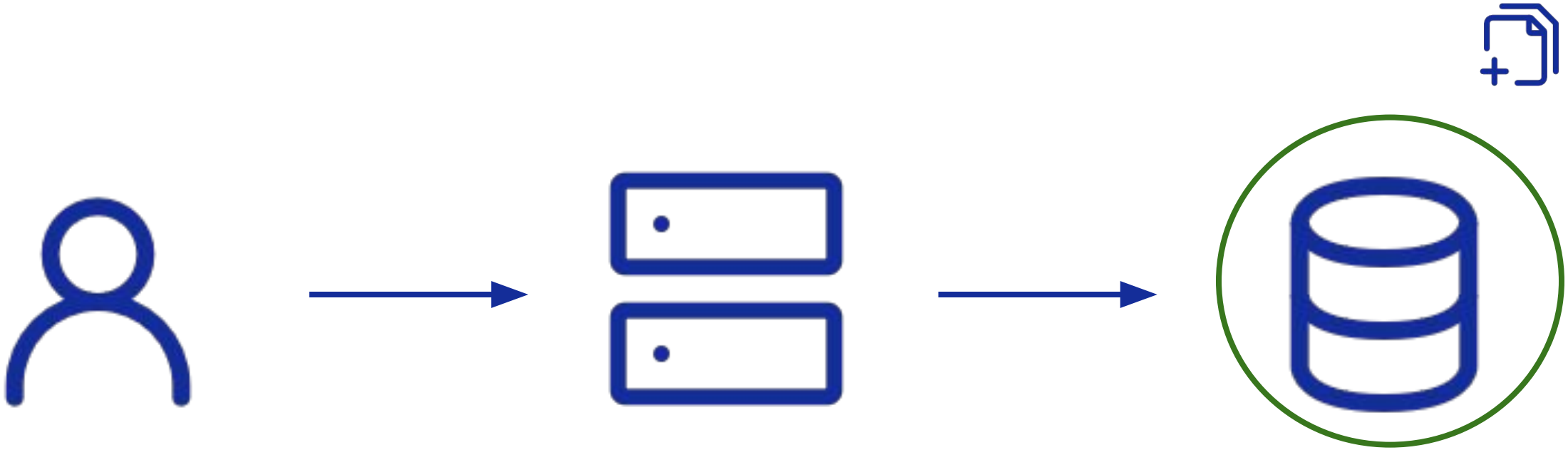
Обработка запроса пользователя

PostgresPro

Дай мне
товар 1

Обработка запроса пользователя

PostgresPro

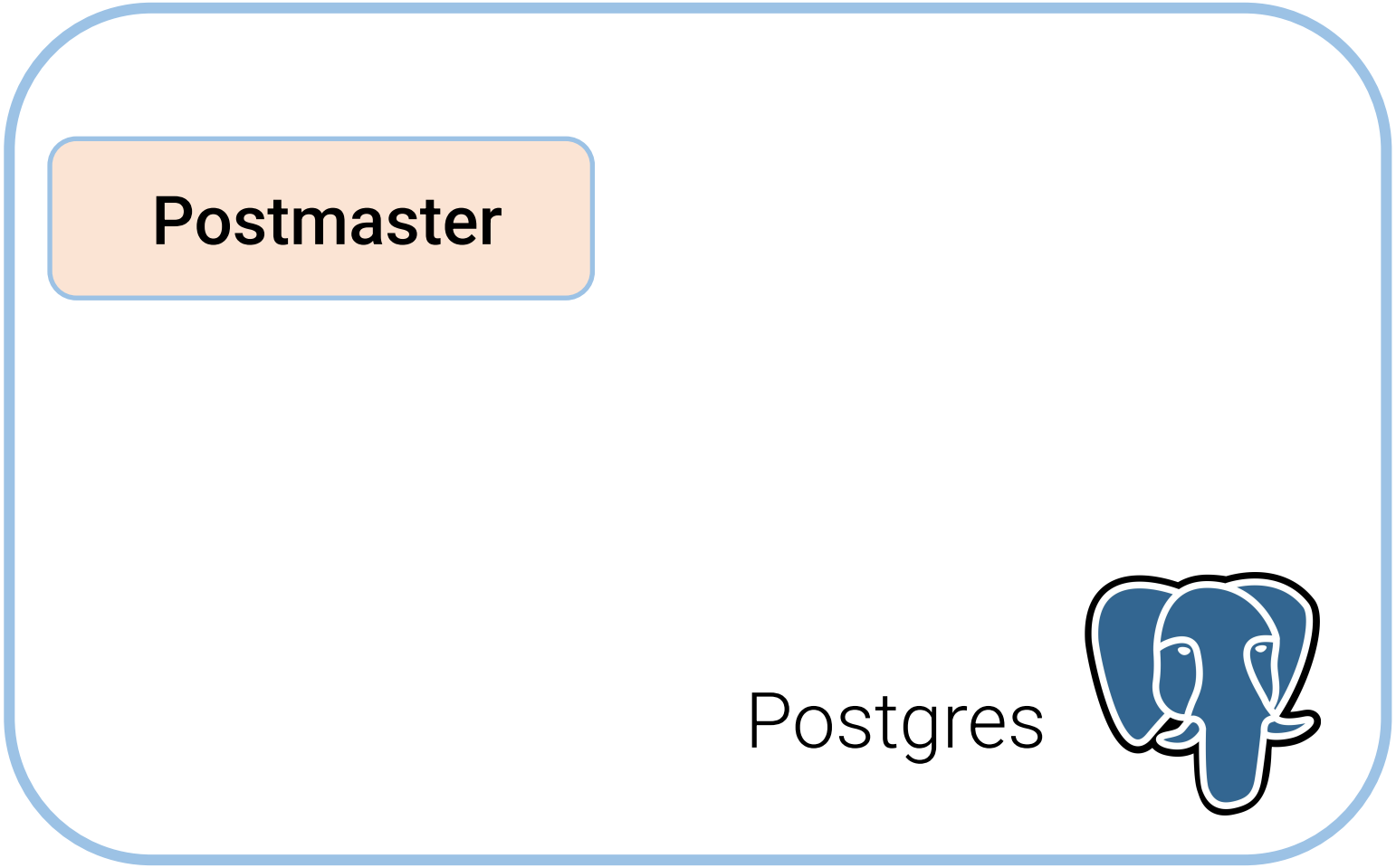


Дай мне
товар 2

кэш в PostgreSQL

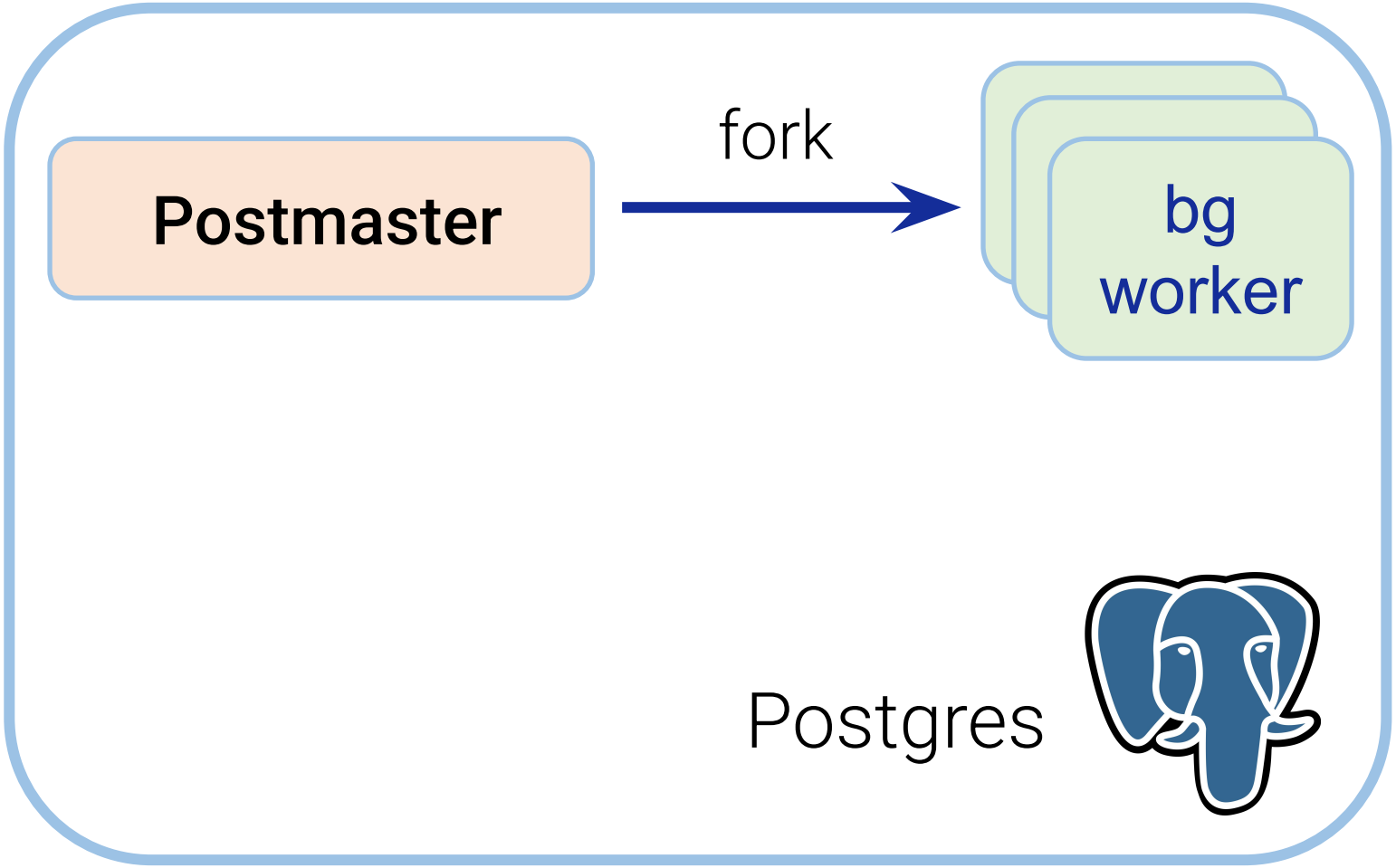
Процессная модель Postgres

PostgresPro



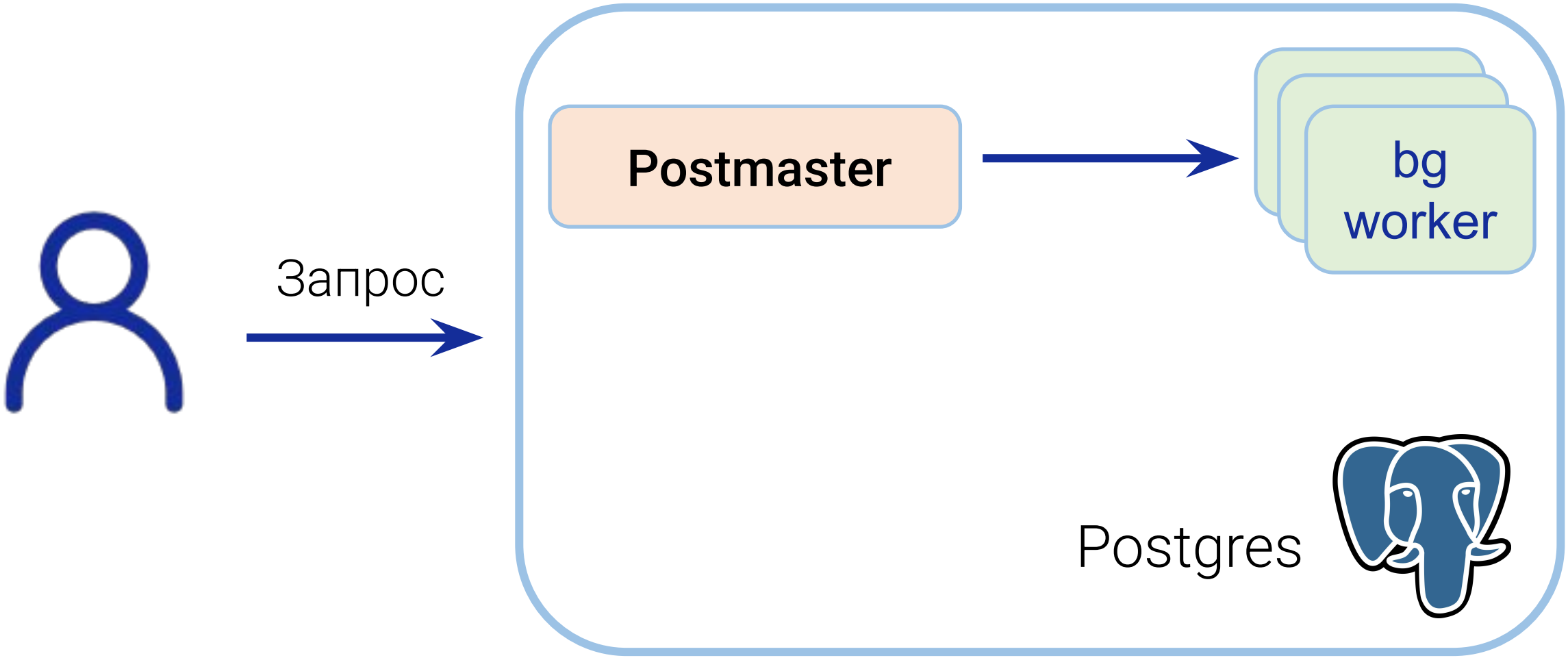
Процессная модель Postgres

PostgresPro



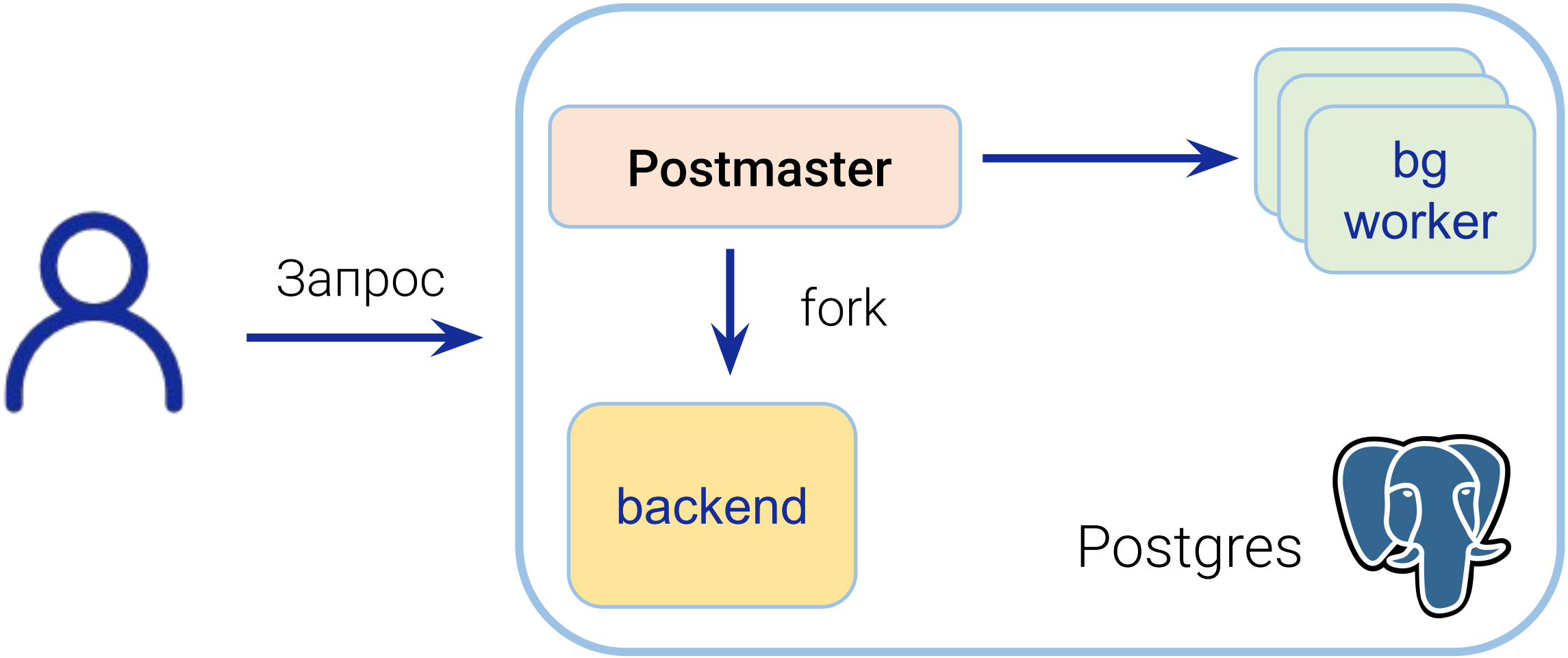
Процессная модель Postgres

PostgresPro



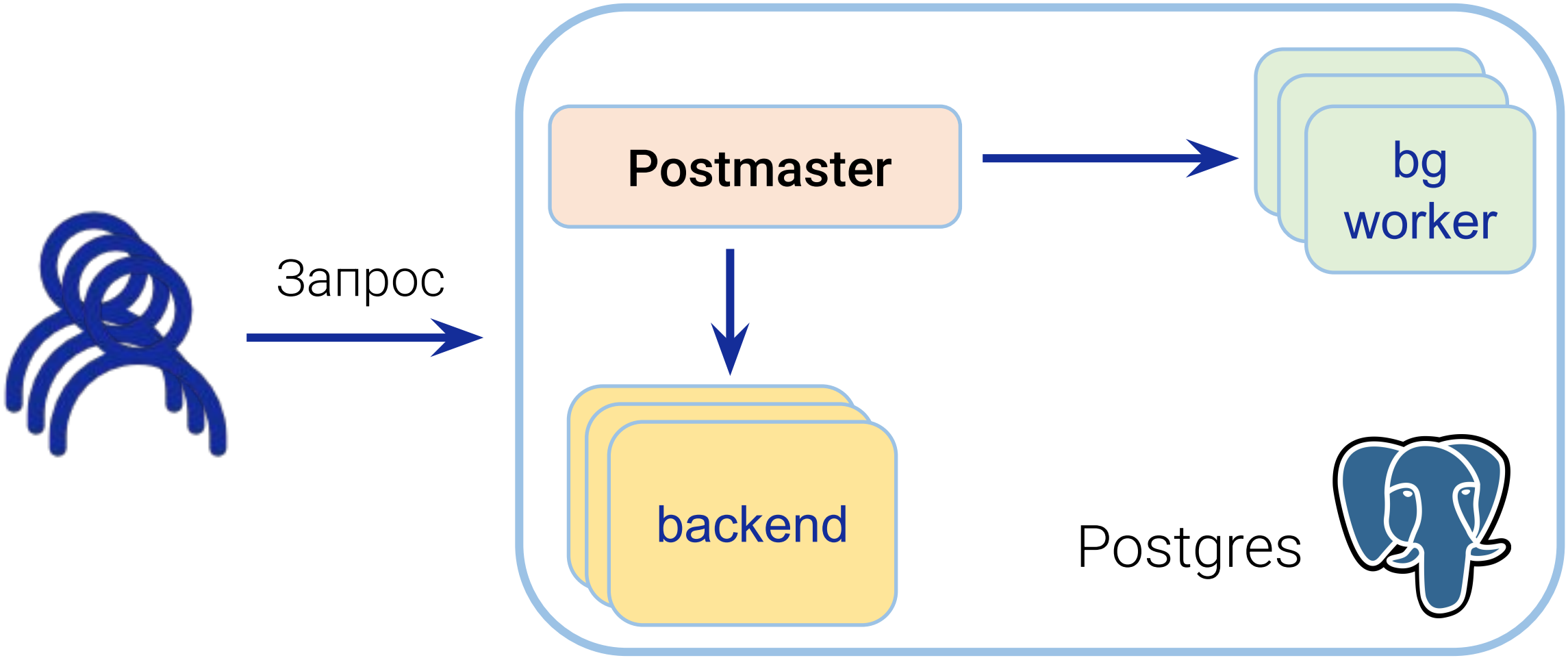
Процессная модель Postgres

PostgresPro



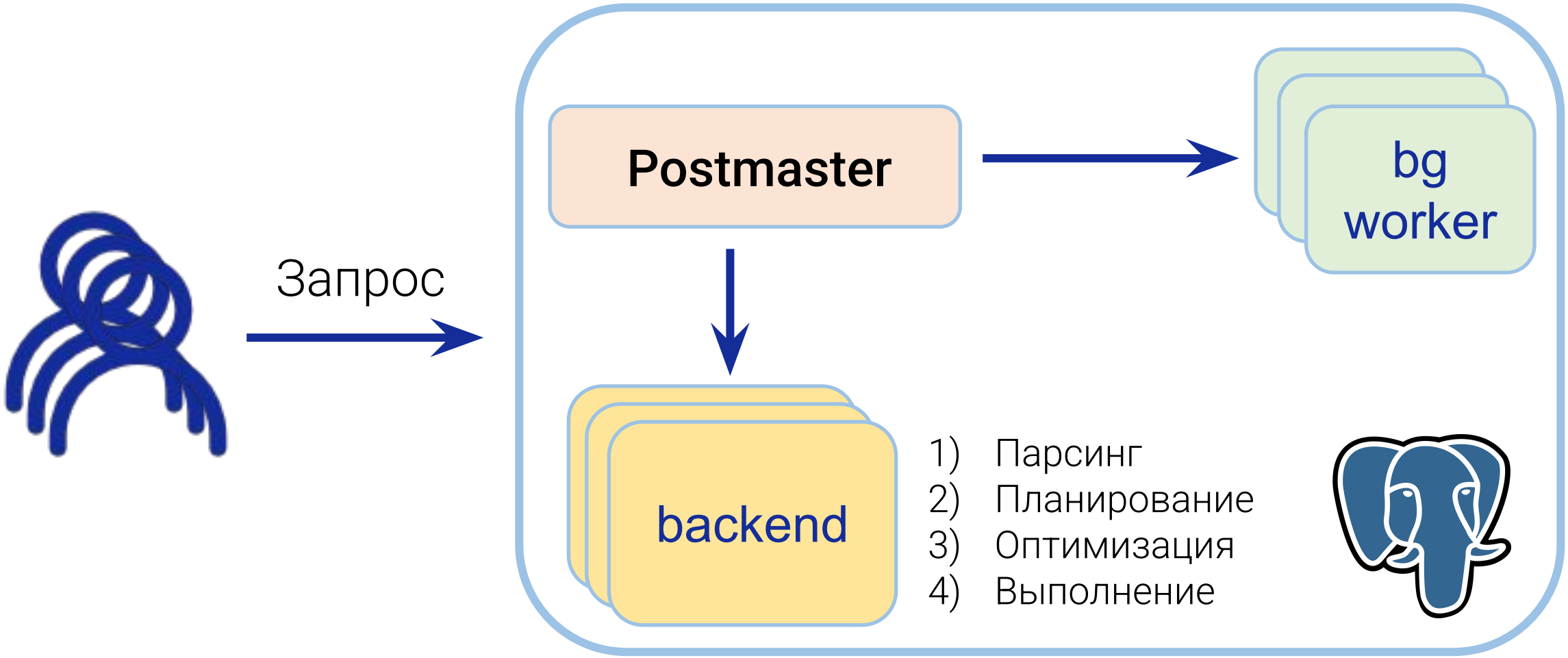
Процессная модель Postgres

PostgresPro



Процессная модель Postgres

PostgresPro



Кэш в PostgreSQL

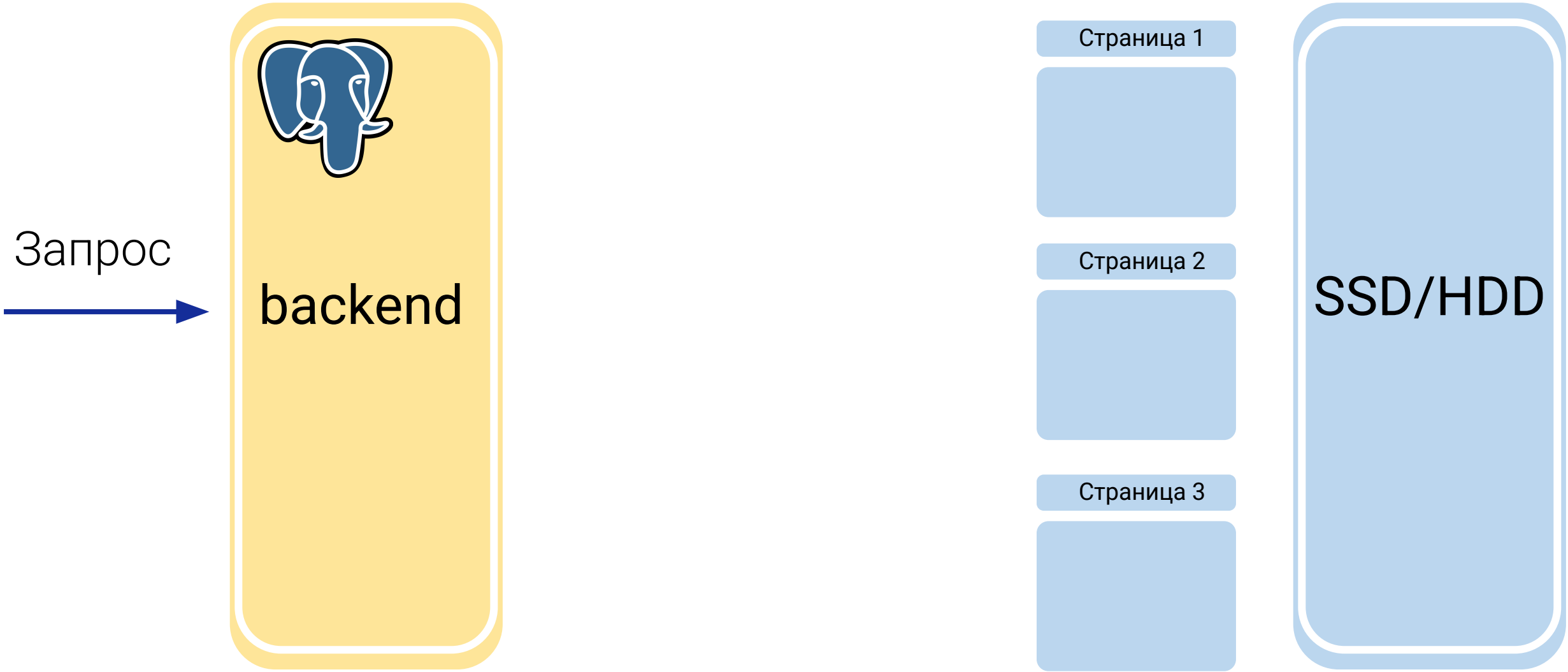
PostgresPro

backend

SSD/HDD

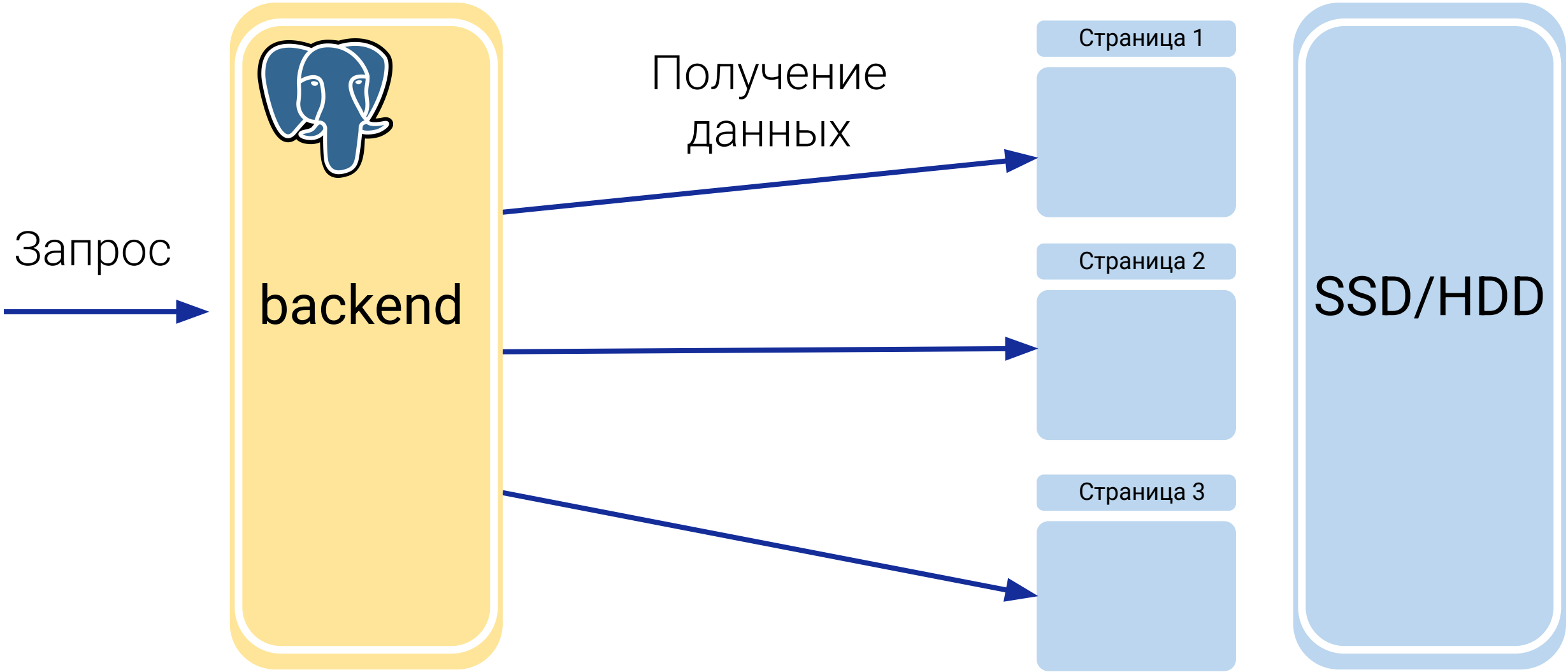
Кэш в PostgreSQL

PostgresPro



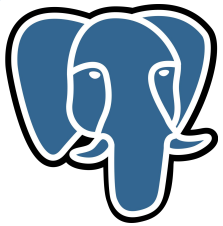
Кэш в PostgreSQL

PostgresPro

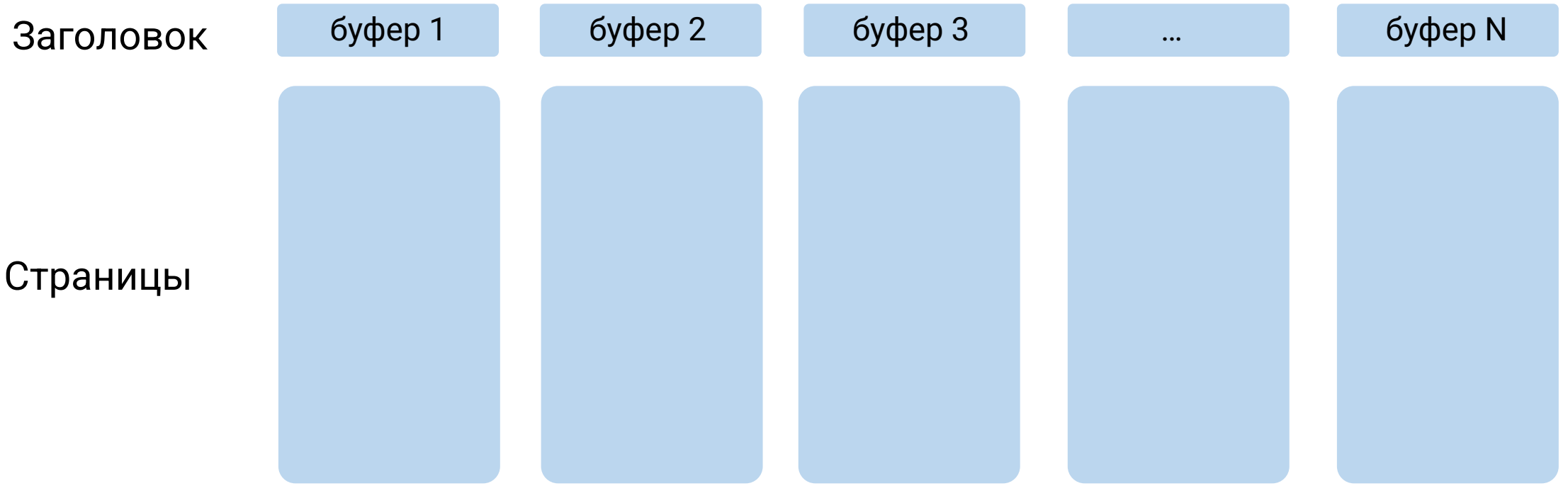


Буферный кэш

PostgresPro

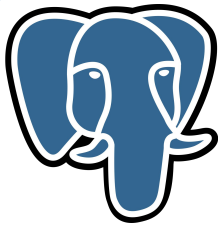


Postgres



Буферный кэш

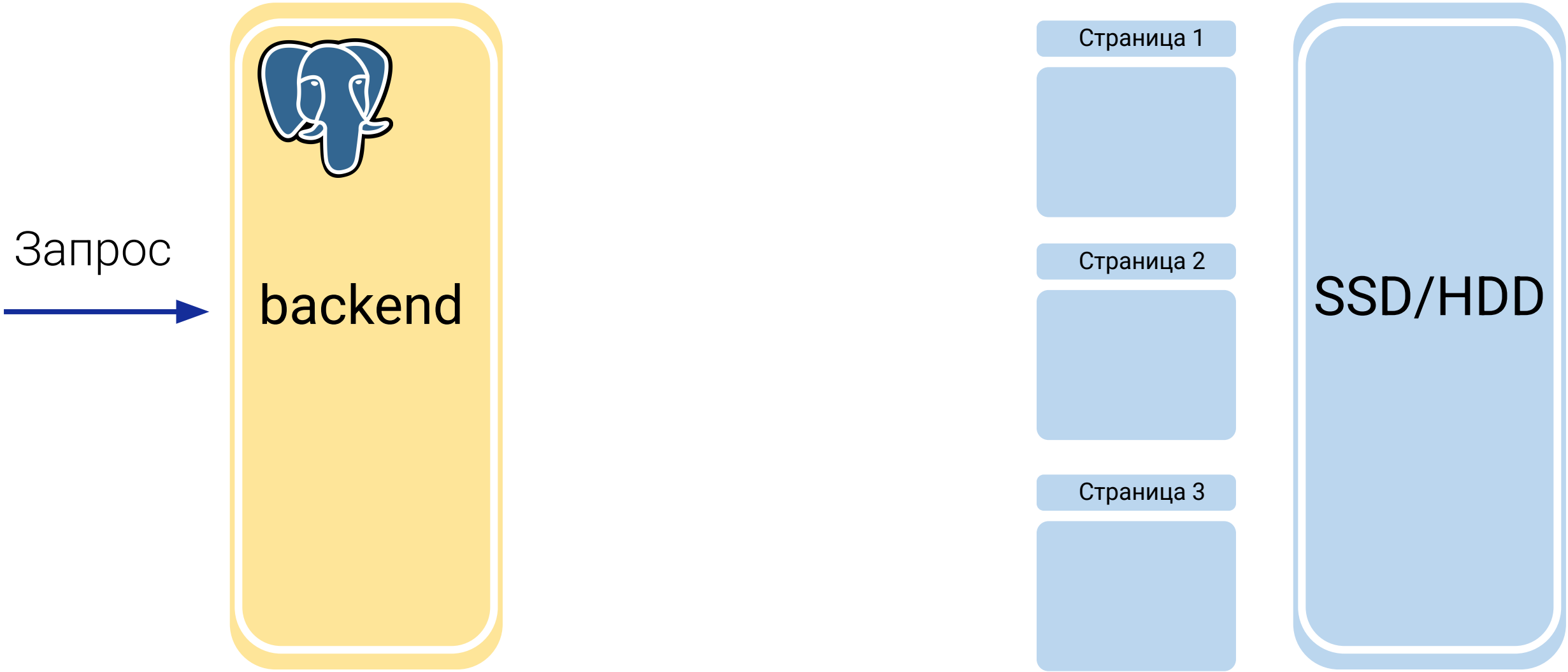
PostgresPro



Postgres

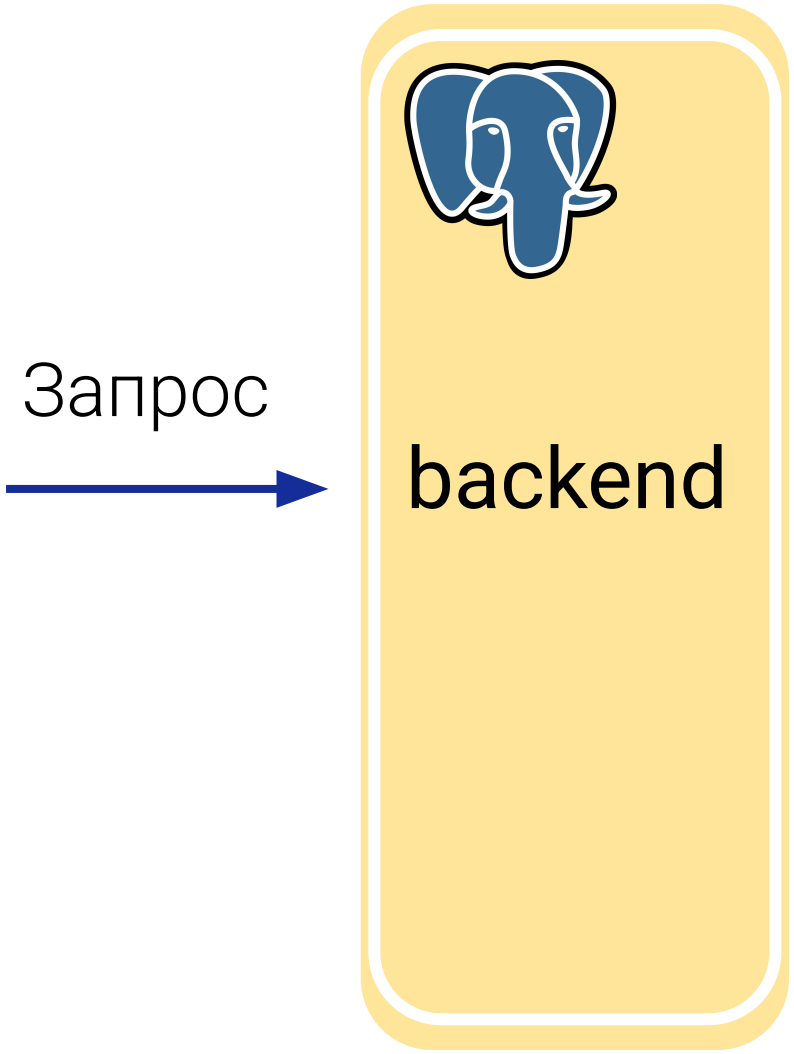
Кэш в PostgreSQL

PostgresPro



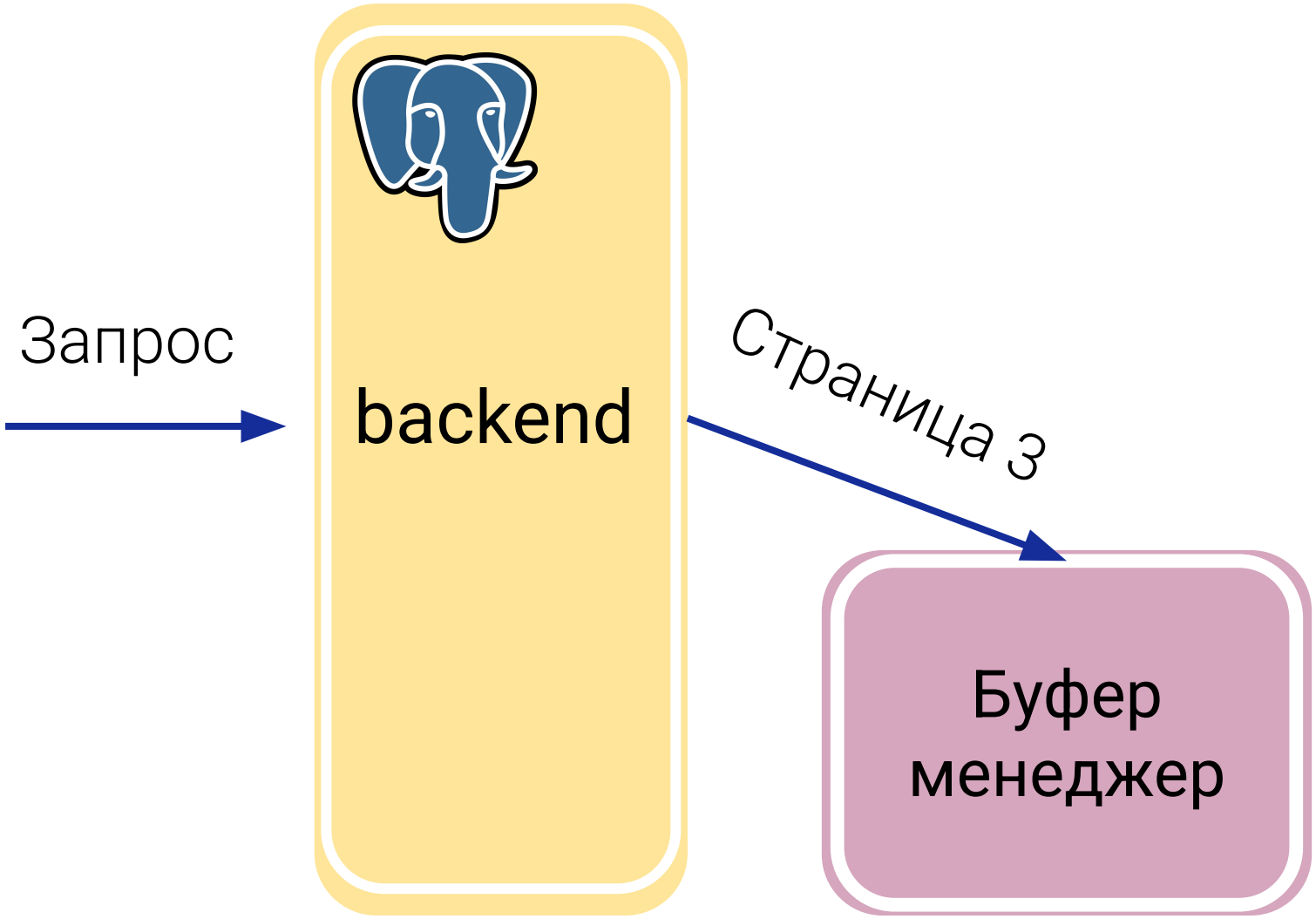
Кэш в PostgreSQL

PostgresPro



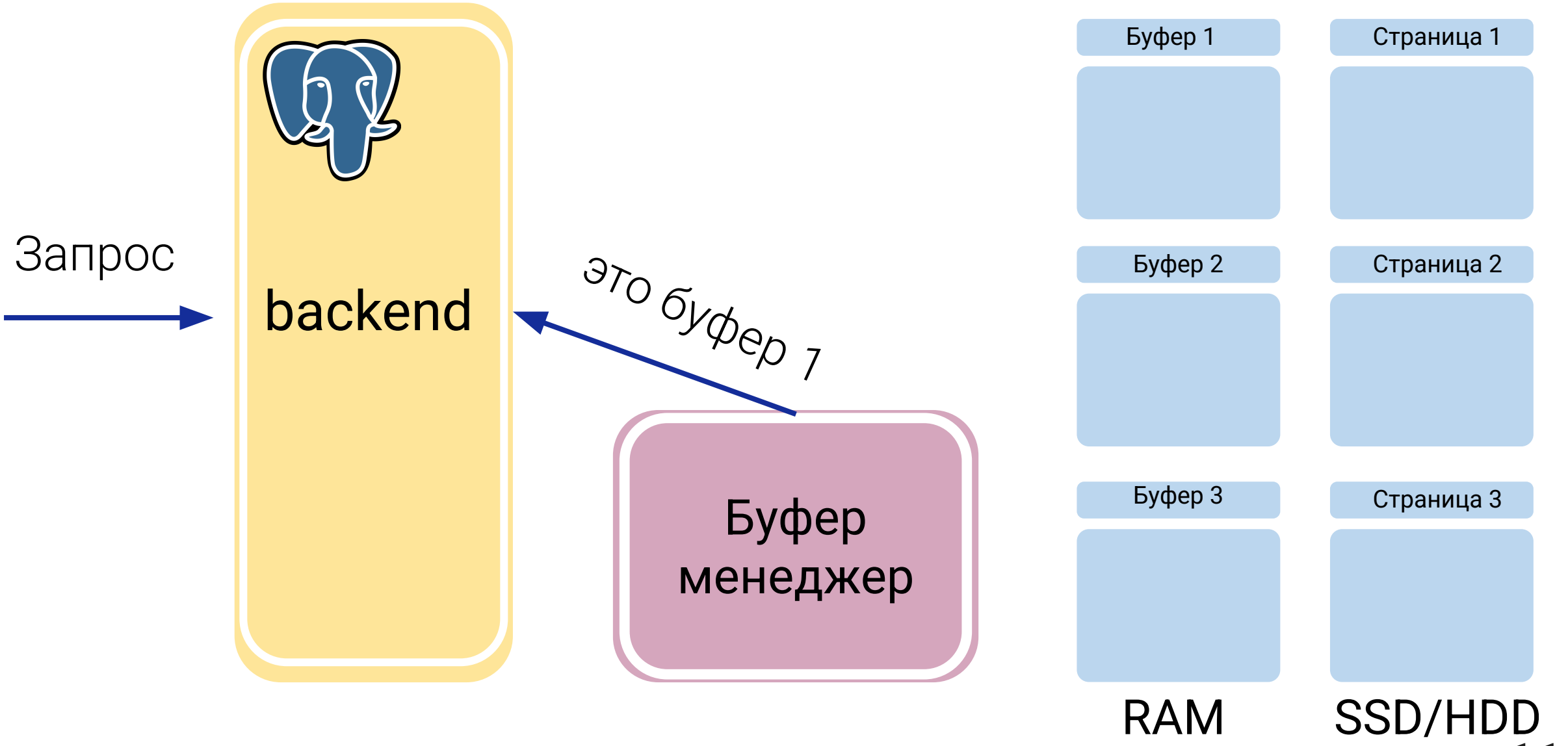
Кэш в PostgreSQL

PostgresPro



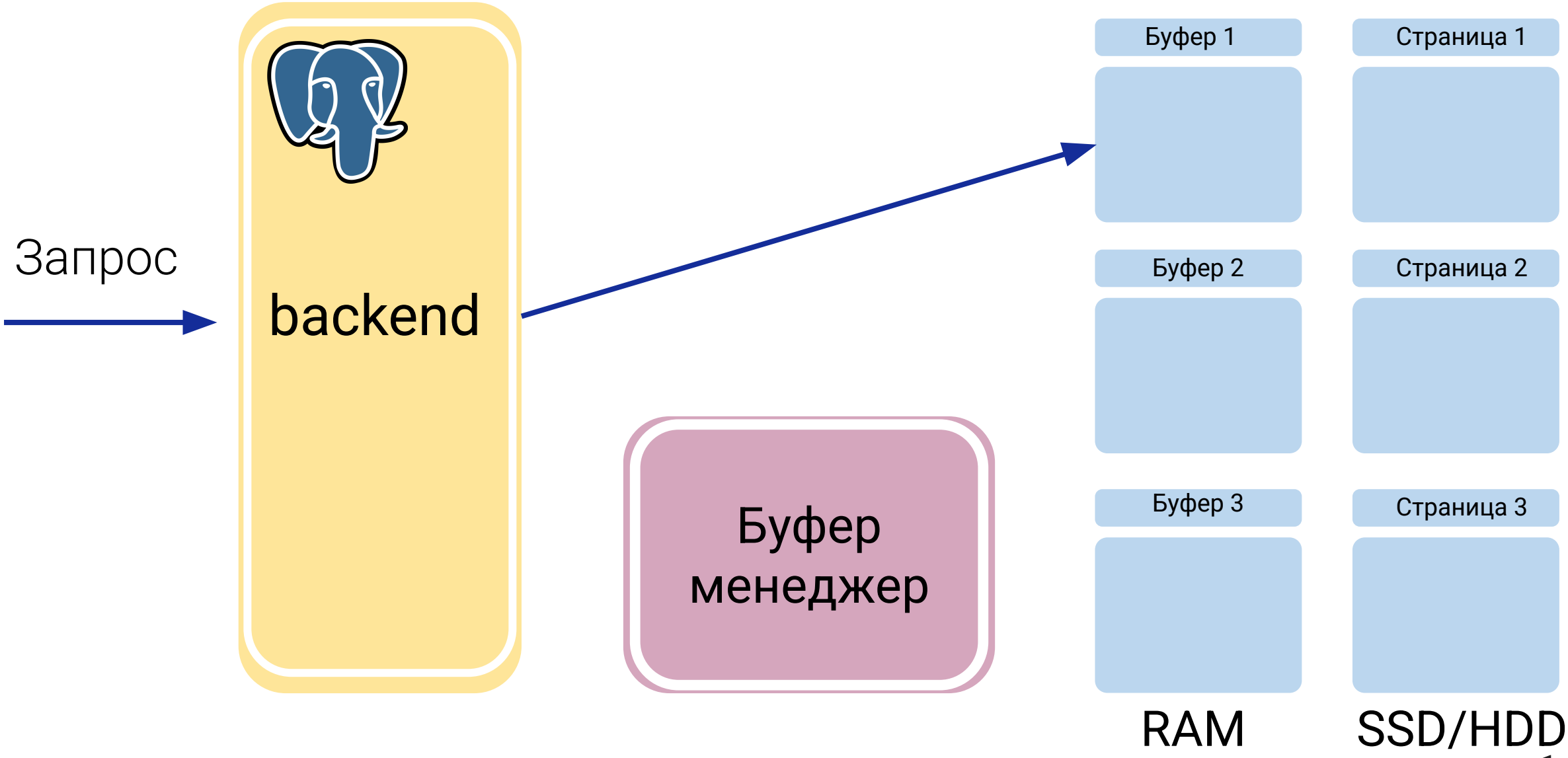
Кэш в PostgreSQL

PostgresPro



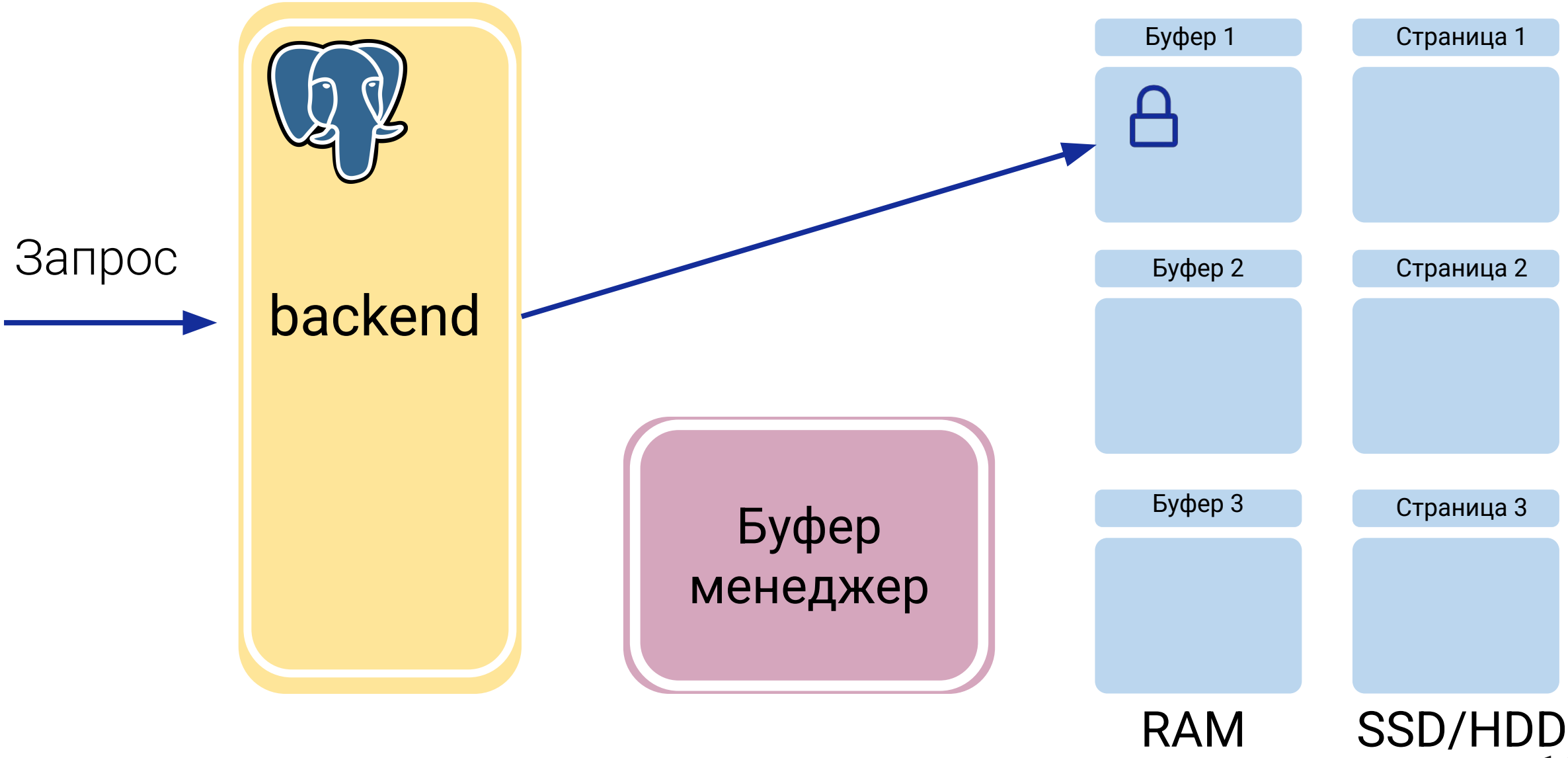
Кэш в PostgreSQL

PostgresPro



Кэш в PostgreSQL

PostgresPro



Маленький shared_buffers

```
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 2
maximum number of tries: 1
duration: 300 s
number of transactions actually processed: 504
number of failed transactions: 0 (0.000%)
latency average = 6019.989 ms
initial connection time = 18.265 ms
tps = 1.661133 (without initial connection time)
```

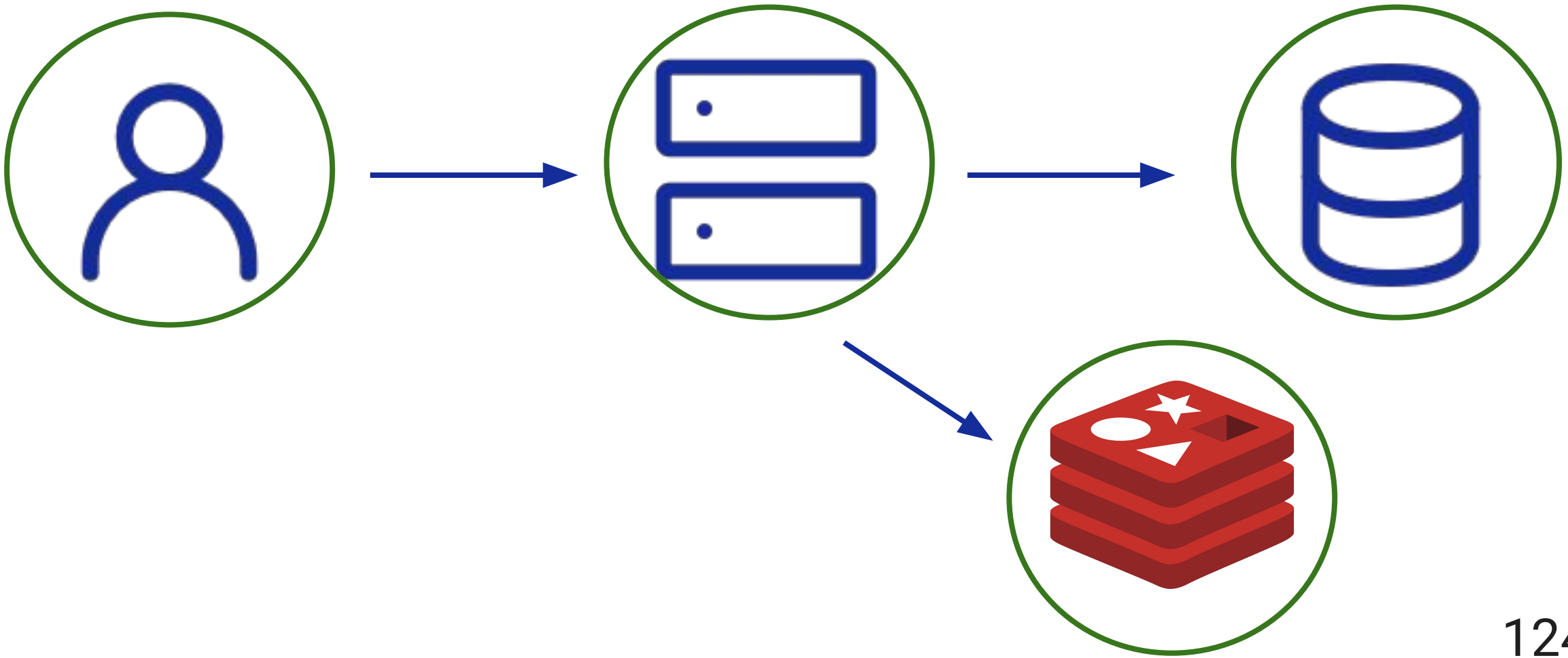
Большой shared_buffers

```
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 2
maximum number of tries: 1
duration: 300 s
number of transactions actually processed: 1085
number of failed transactions: 0 (0.000%)
latency average = 2780.035 ms
initial connection time = 19.782 ms
tps = 3.597078 (without initial connection time)
```

Общая картина

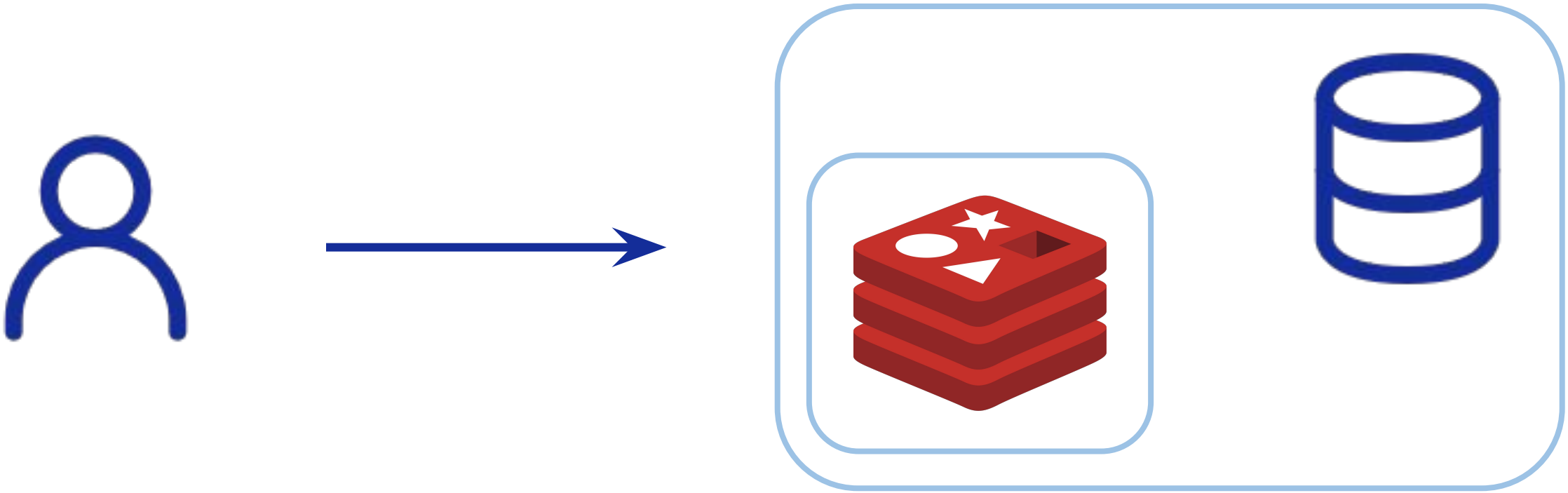
Обработка запроса пользователя

PostgresPro



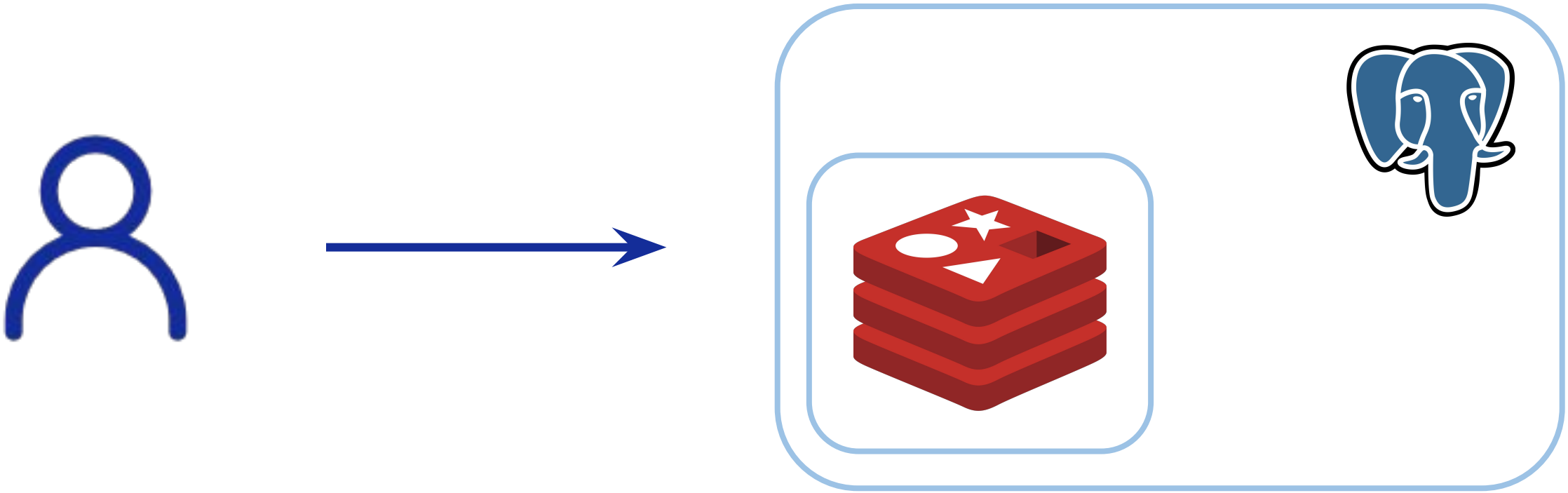
Почему нельзя так?

PostgresPro



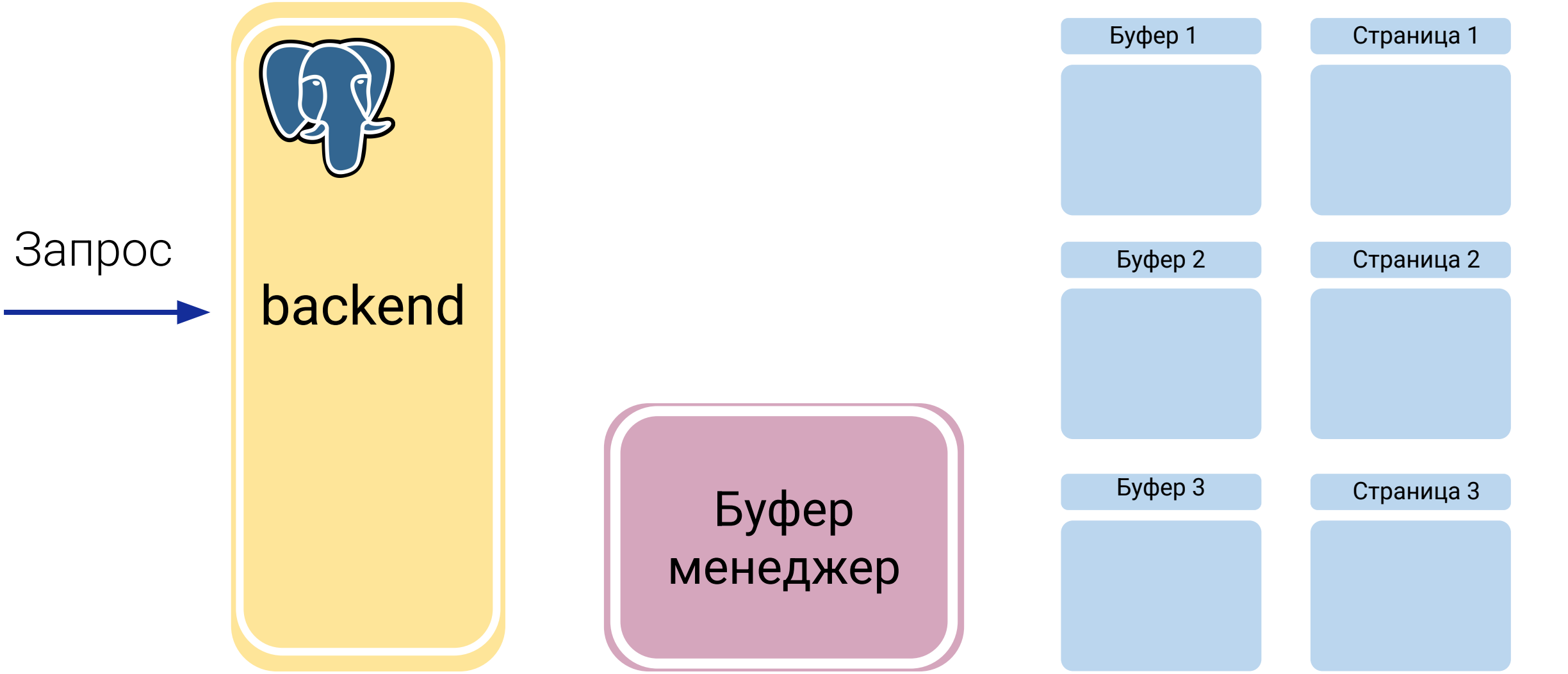
Почему нельзя так?

PostgresPro



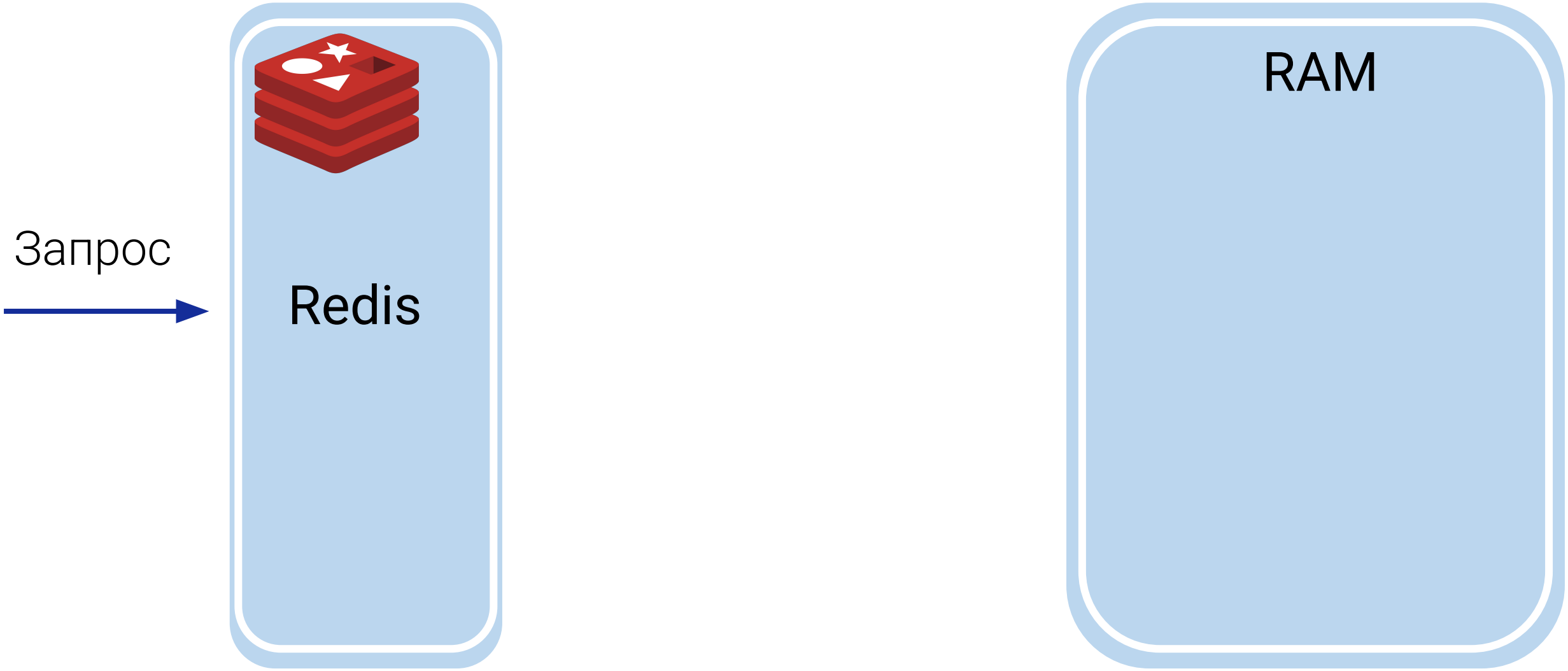
Кэш в PostgreSQL

PostgresPro



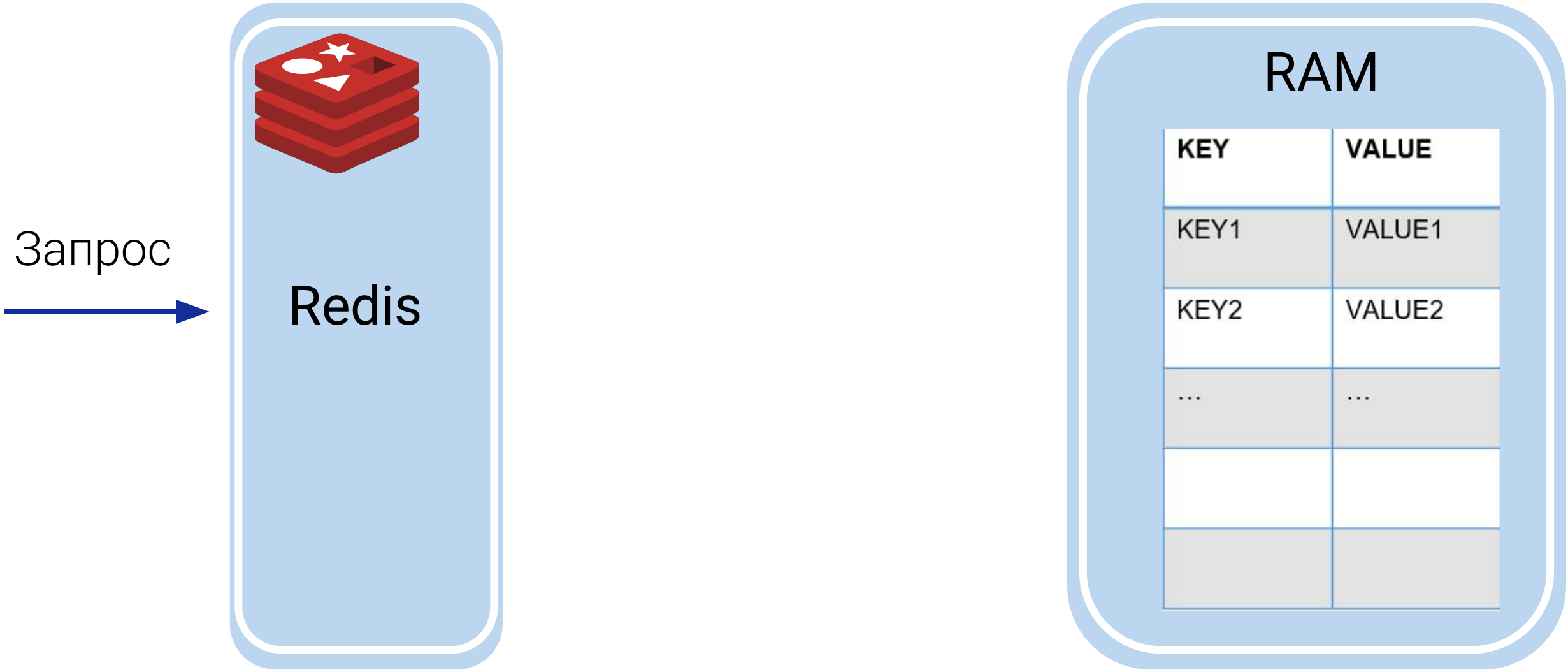
Кэш в Redis

PostgresPro



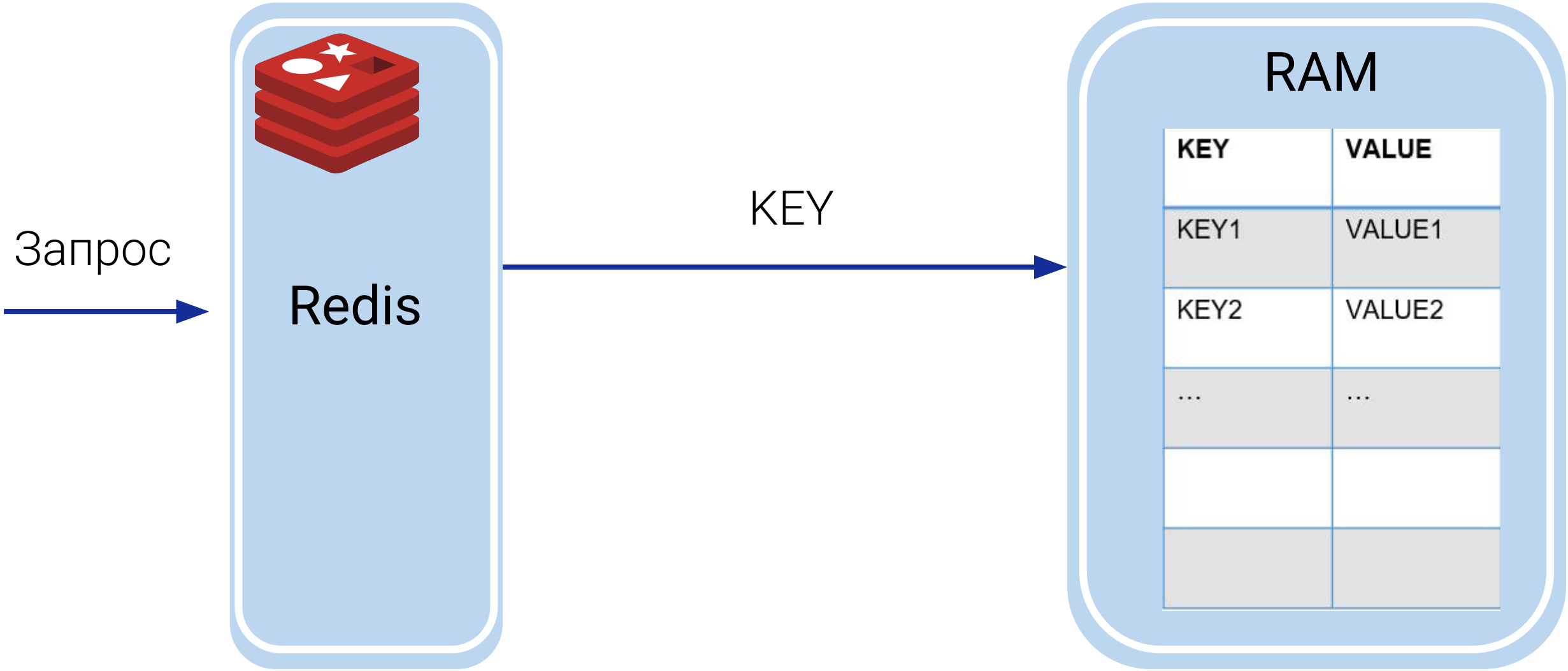
Кэш в Redis

PostgresPro



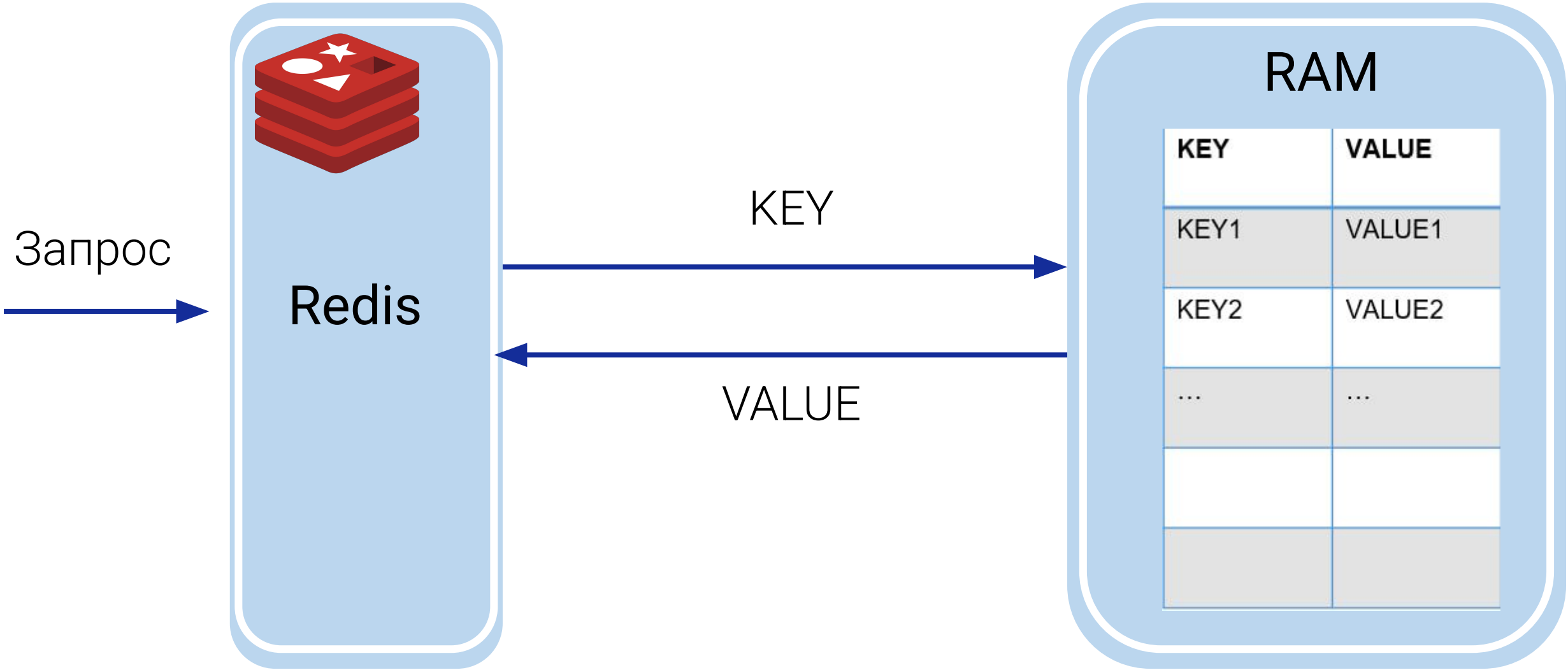
Кэш в Redis

PostgresPro



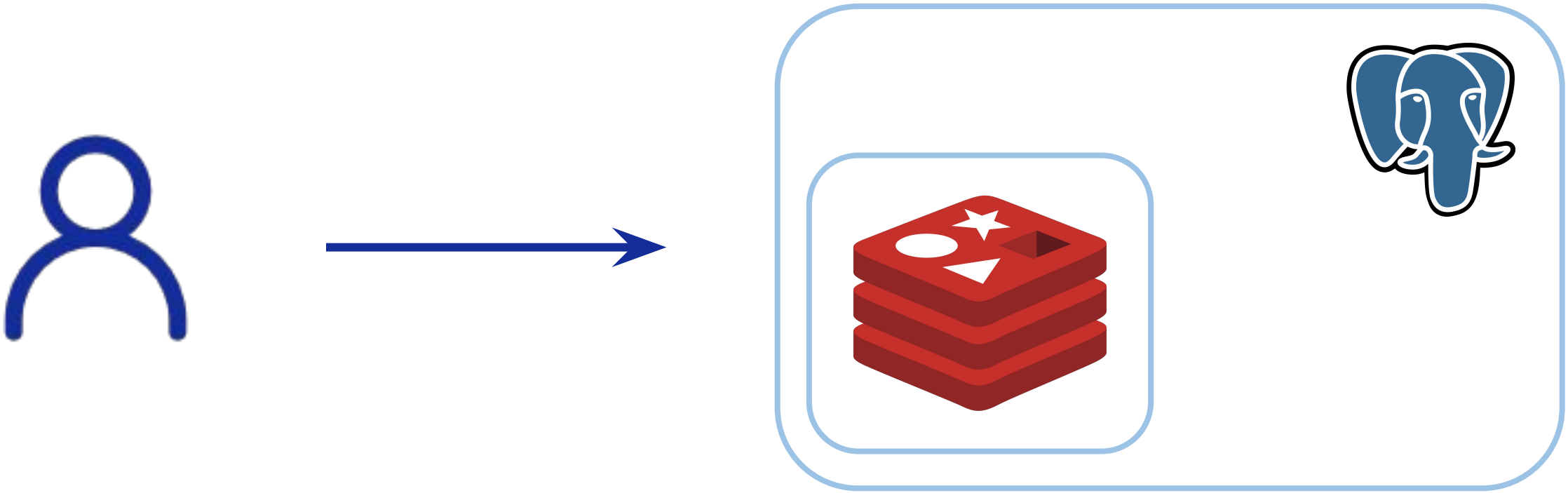
Кэш в Redis

PostgresPro



Почему нельзя так?

PostgresPro

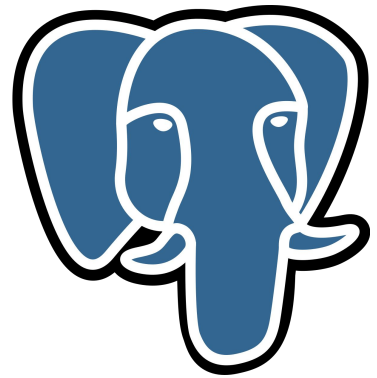


PostgresPro: KVik

Как это сделать?

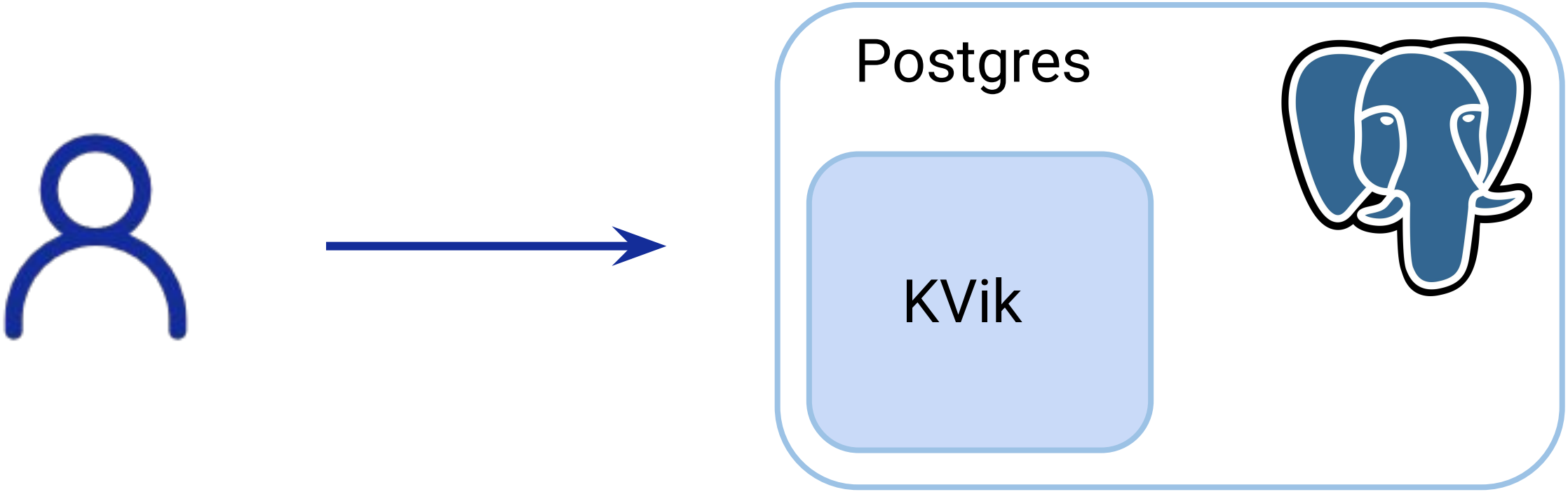
PostgresPro

Postgres



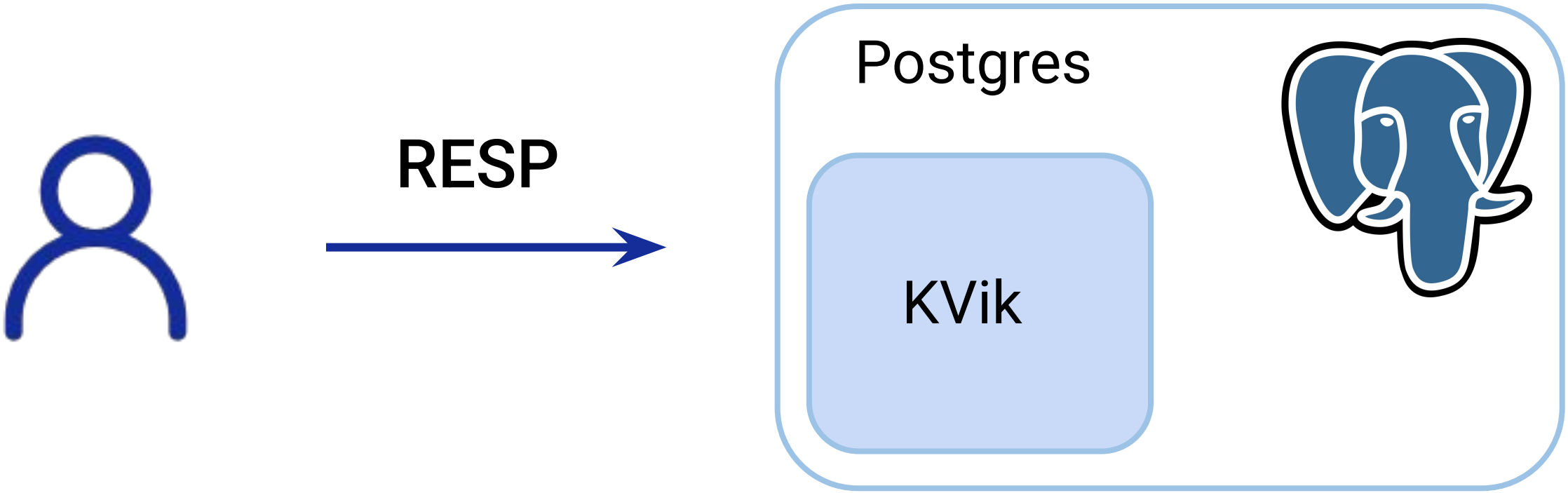
Как это сделать?

PostgresPro



Как это сделать?

PostgresPro



RESP

PostgresPro

"+OK\r\n" - Strings

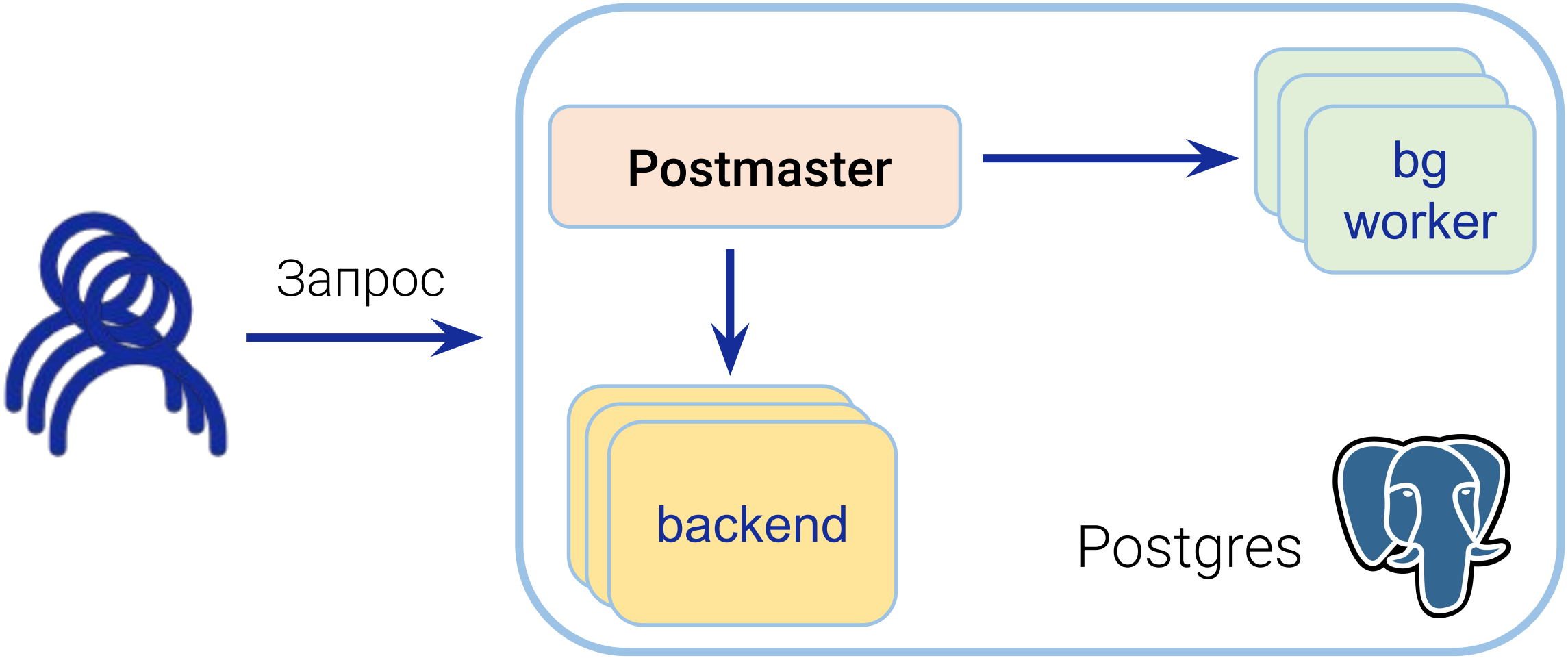
"-Error message\r\n" - errors

"$6\r\nfoobar\r\n" - Bulk Strings

"*2\r\n$3\r\nfoo\r\n$3\r\nbar\r\n" - arrays

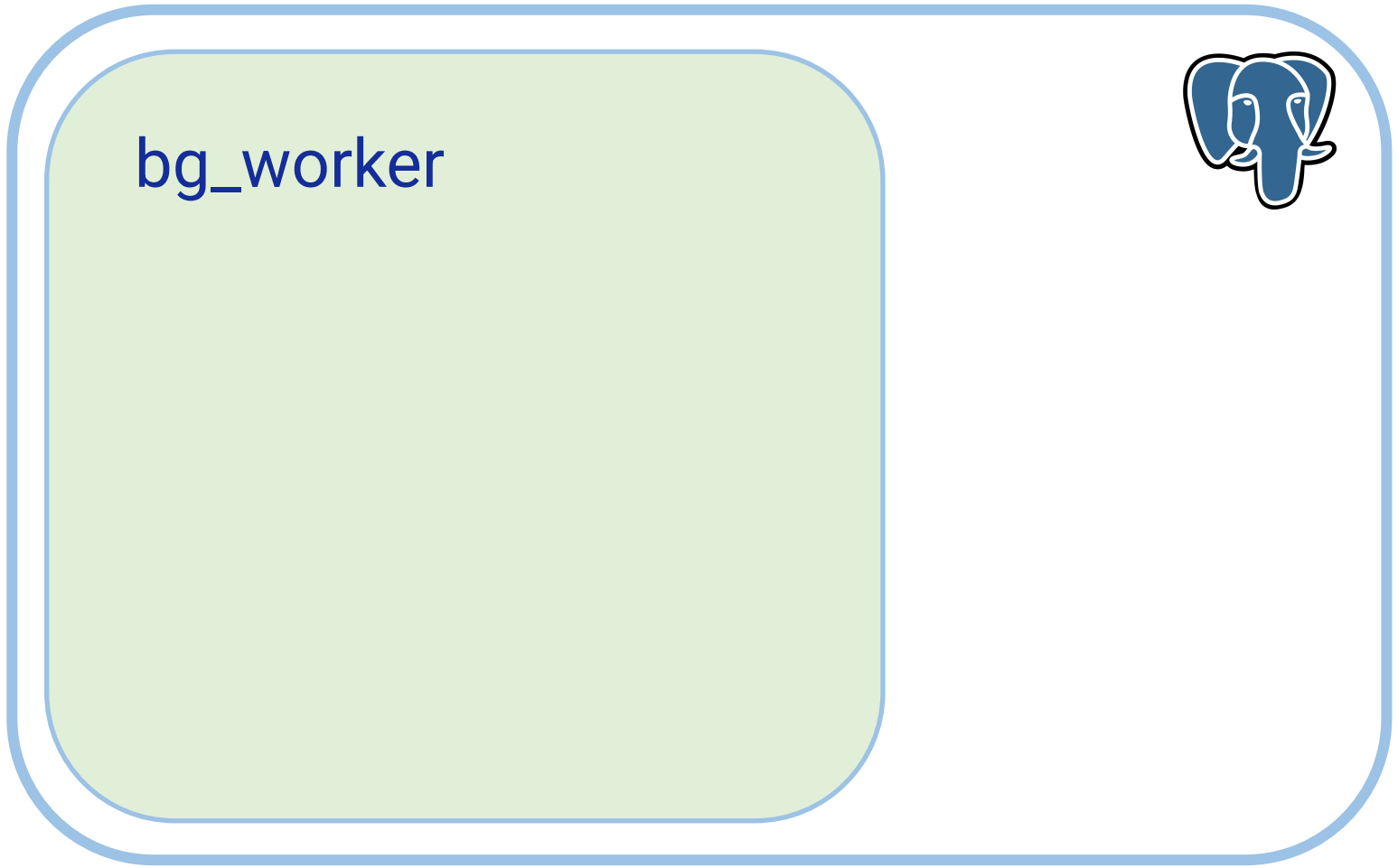
":0\r\n" - Integers

KVik



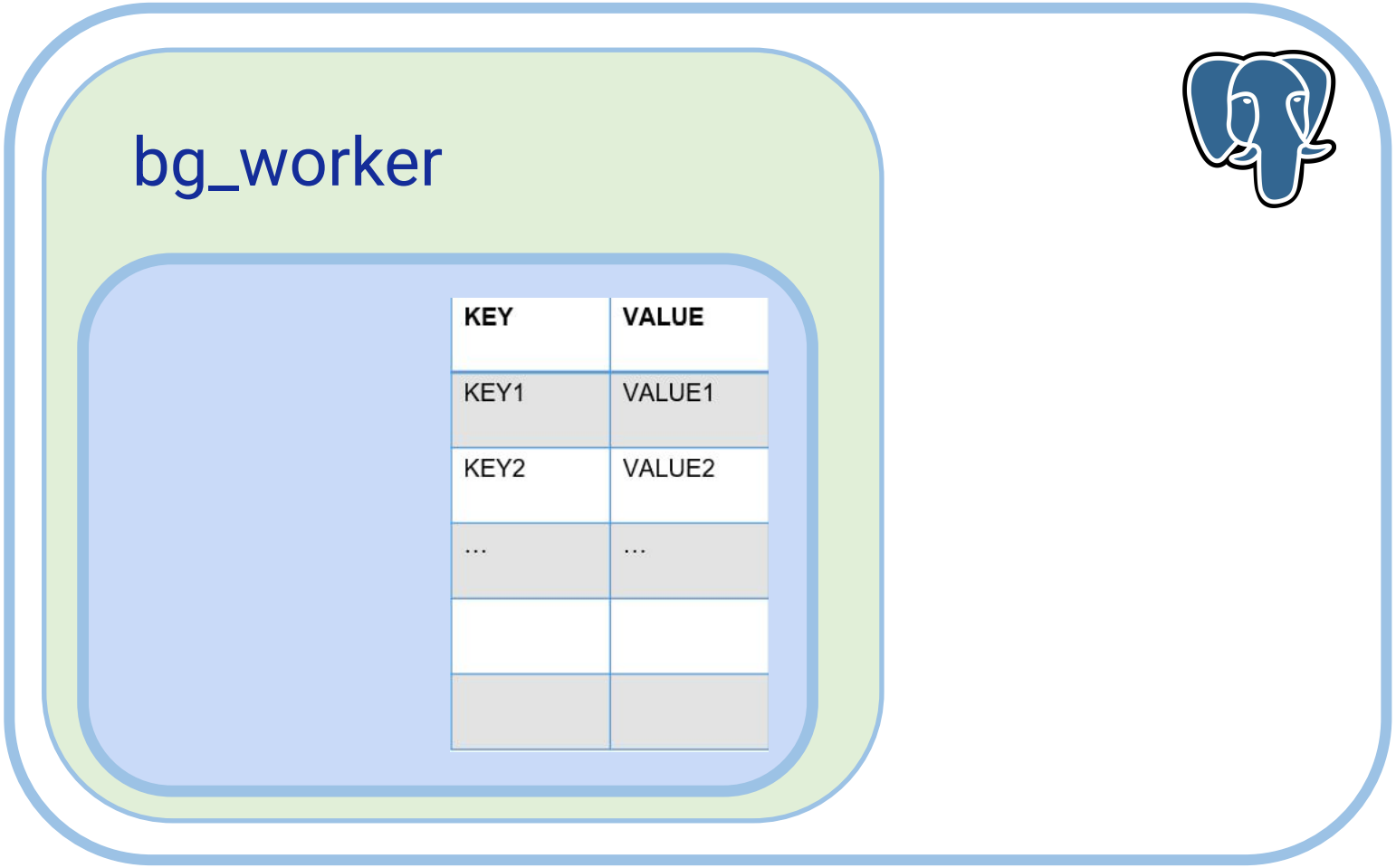
KVik

PostgresPro



KVik

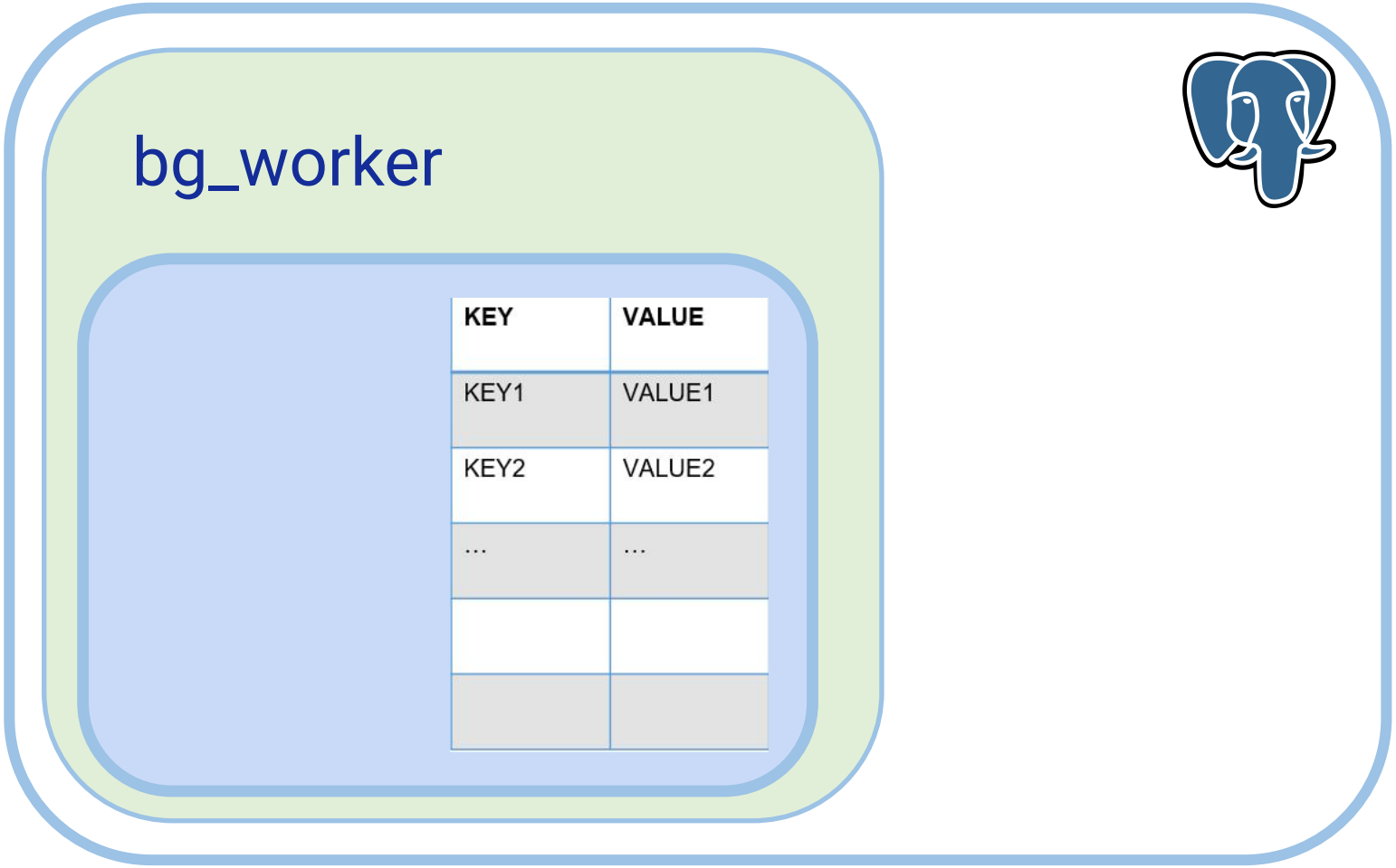
PostgresPro



KVik

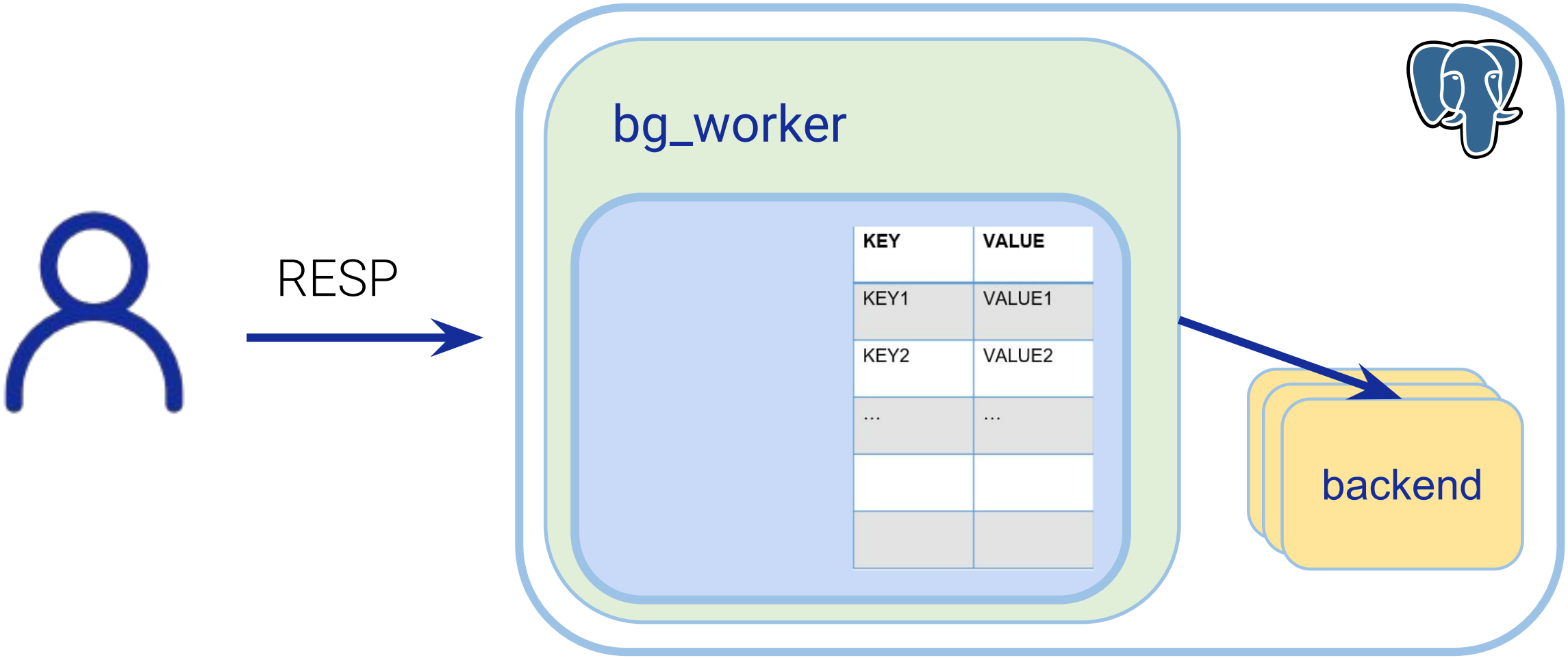
PostgresPro

RESP



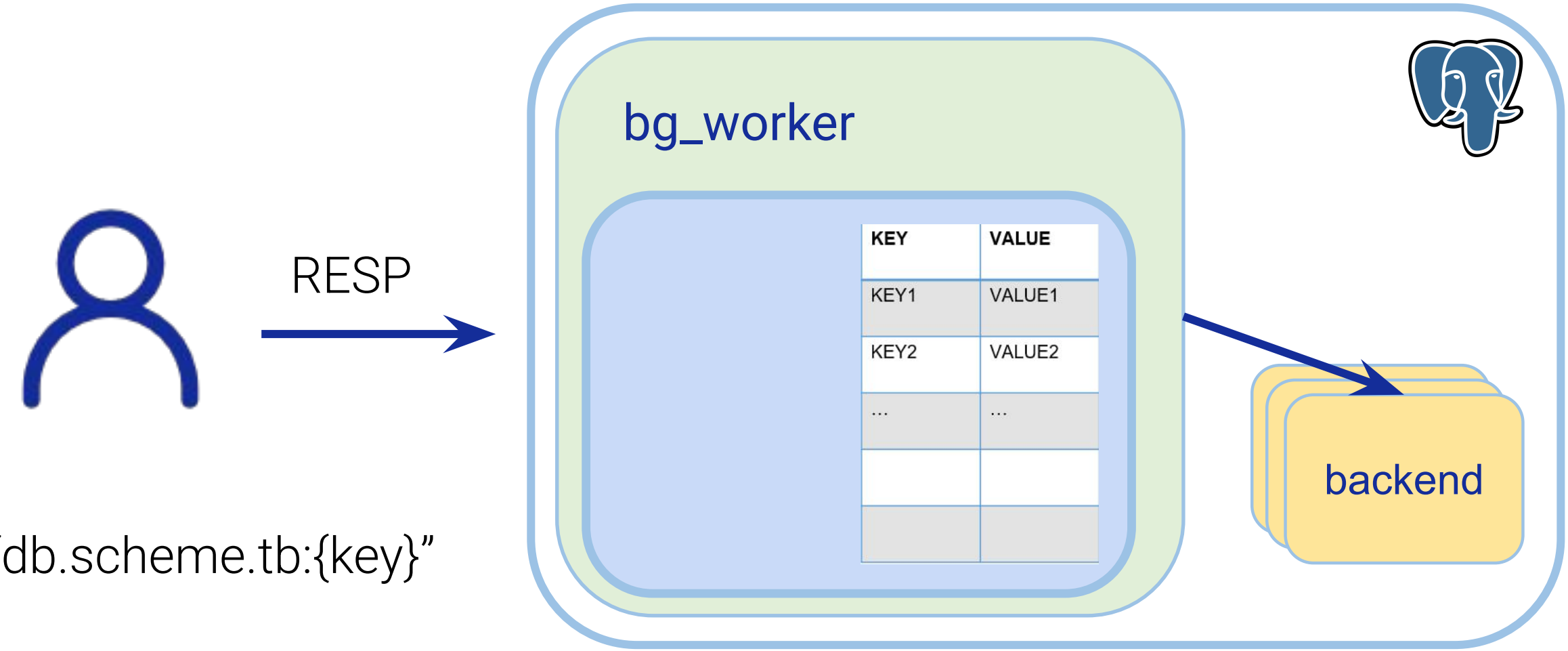
KVik

PostgresPro



KVik

PostgresPro

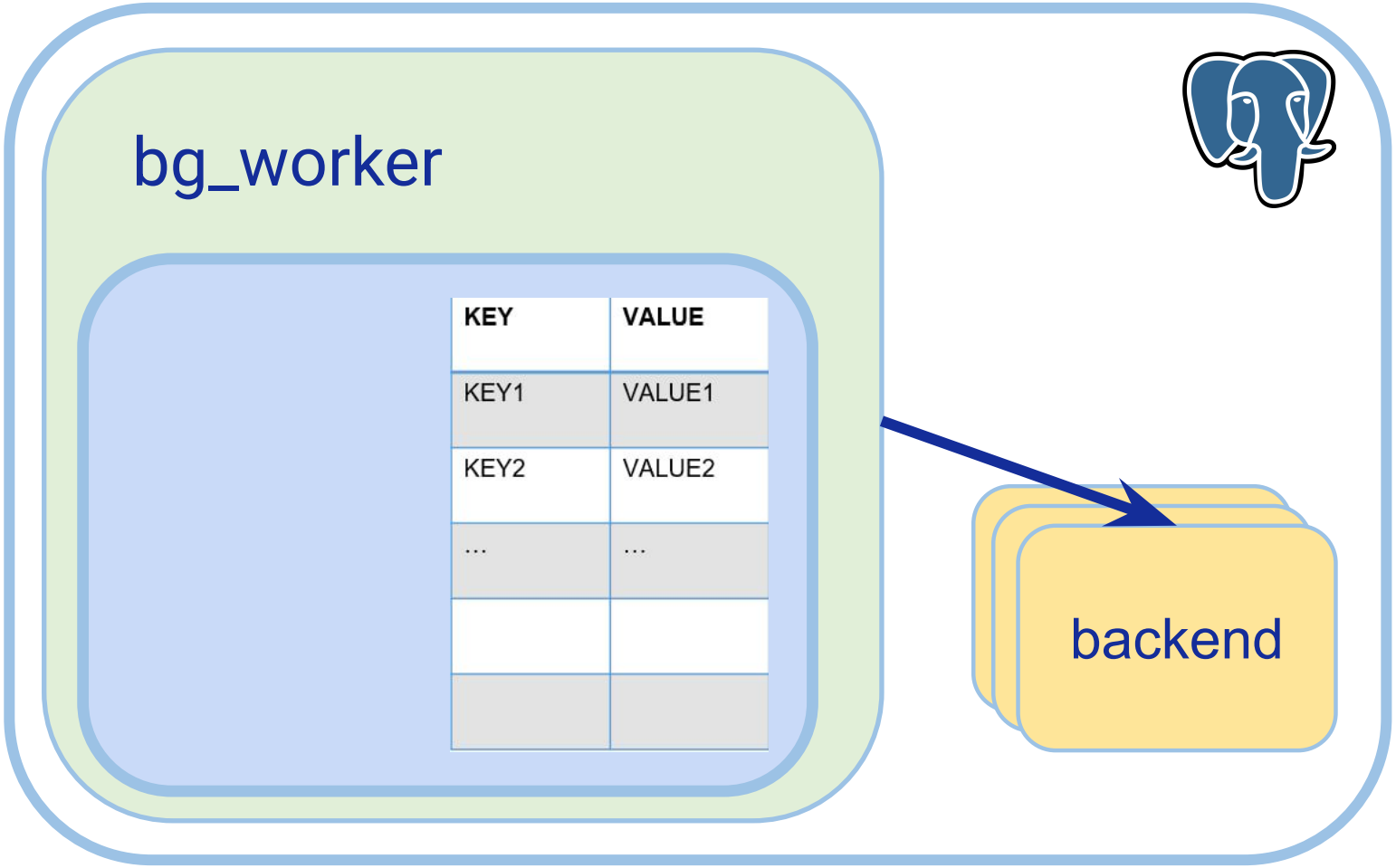


KVik

PostgresPro

RESP

"db.scheme.tb:{key}"

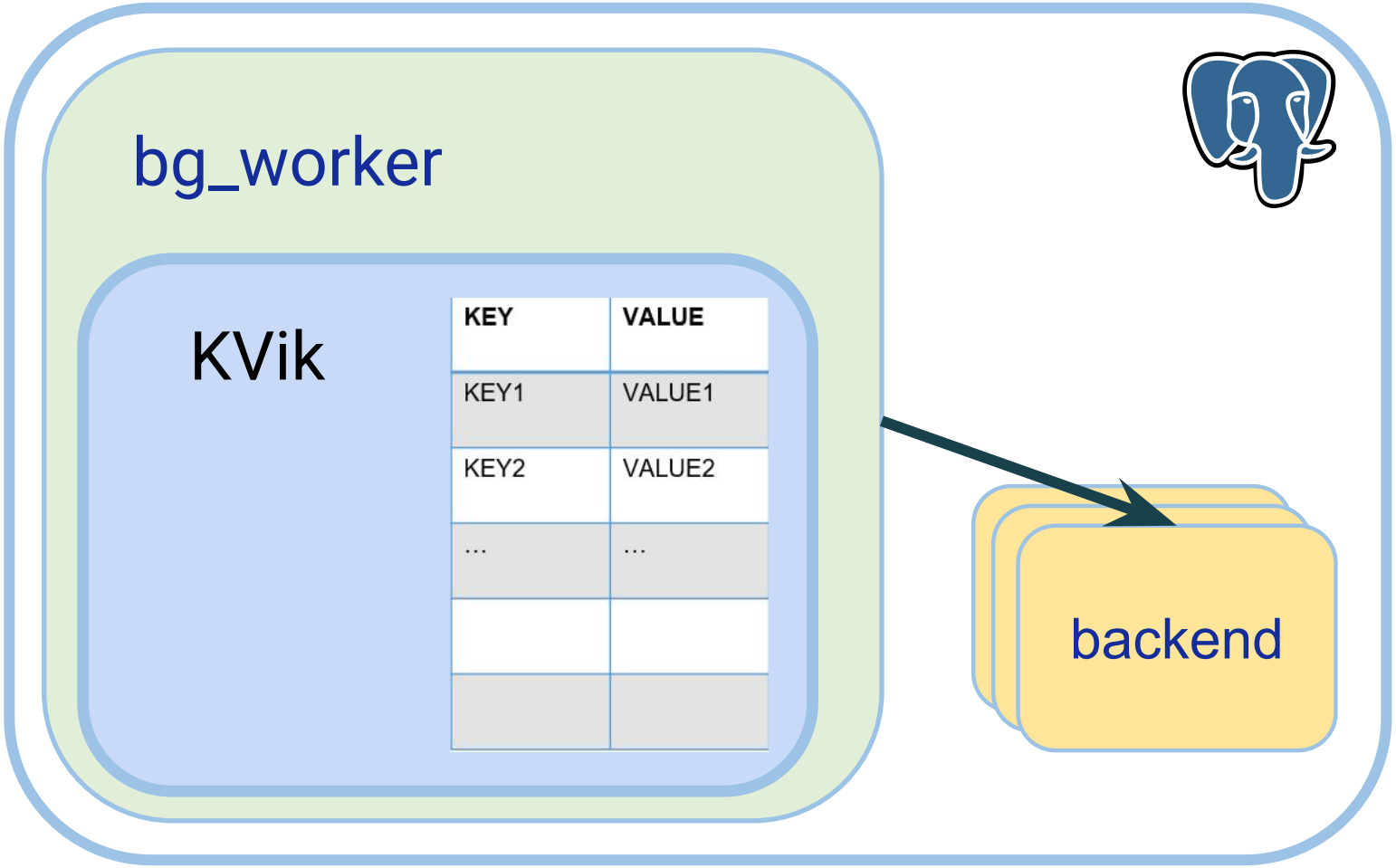


KVik

PostgresPro

RESP

"db.scheme.tb:{key}"



Документация KVik

PostgresPro

Плюсы и минусы KVik

PostgresPro

+ Высокая производительность	— Работает только с Postgres
+ Инвалидация из коробки	— Только читающий профиль нагрузки
+ Простая конфигурация и эксплуатация	

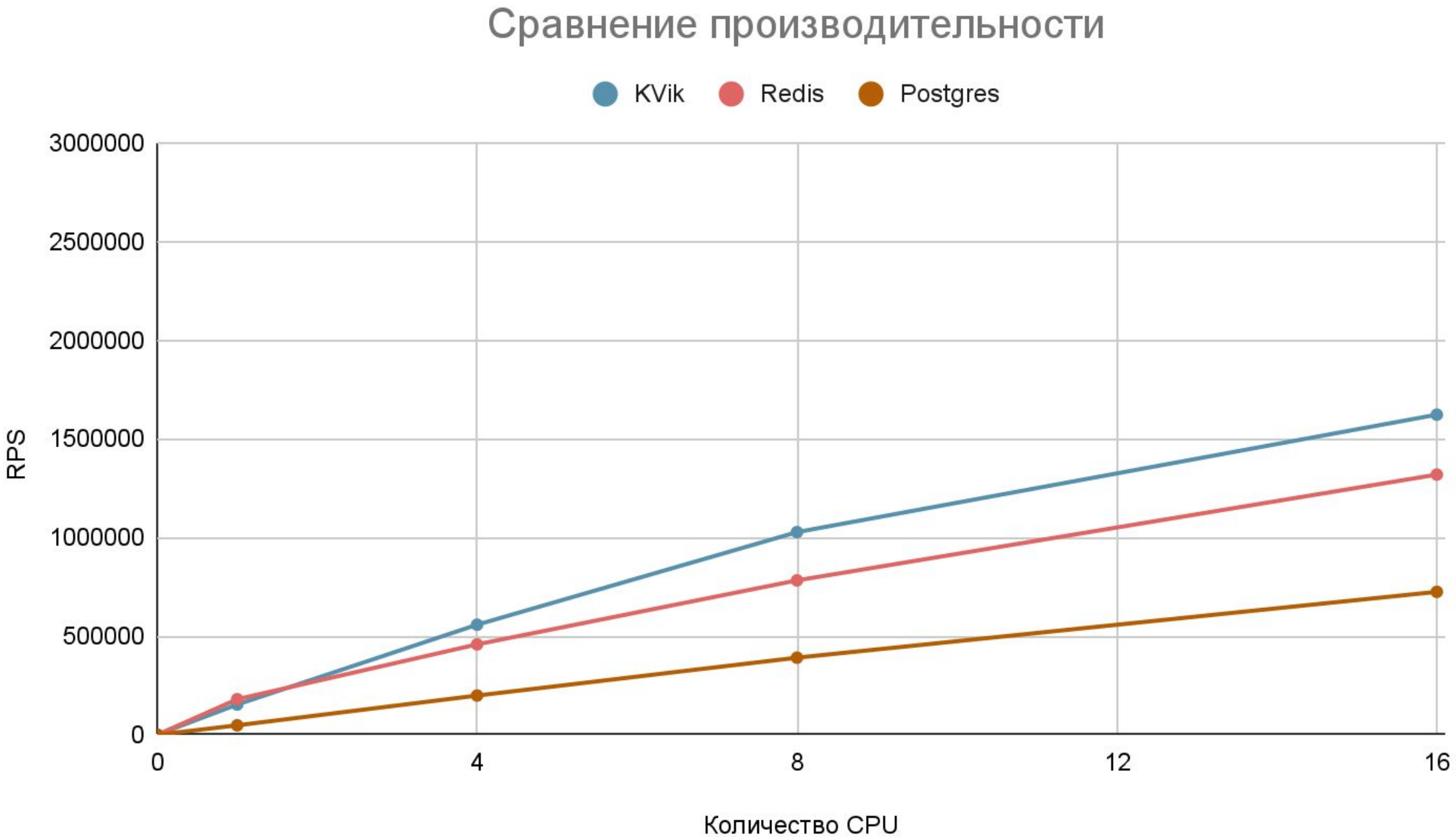
```python
# Функция для получения погодных данных из PostgreSQL
def get_weather(city):
    cached_data = r.get(f"weather:{city}") # Сначала проверим, есть ли данные в Redis
    if cached_data:
        return cached_data.decode('utf-8')
    else: # Если в Redis нет , то выполняем запрос к базе данных
        try:
            conn = psycopg2.connect(dbname="db",user="user", password="pass", host="host", port="port")
            cursor = conn.cursor()
            query = sql.SQL("""SELECT description, temperature FROM weather
                    WHERE city = %s ORDER BY created_at DESC LIMIT 1""")
            cursor.execute(query, (city,))
            data = cursor.fetchone()
            if data:
                weather_description, temperature = data
                result = f'Погода в {city}: {weather_description}, Температура: {temperature}°C'
                r.setex(f'weather:{city}', 600, result) # Сохраняем данные в Redis на 10 минут
                return result
            else:
                return 'Данные о погоде для этого города нигде не найдены'
        except:
            return 'Ошибка получения данных с сервера'
```

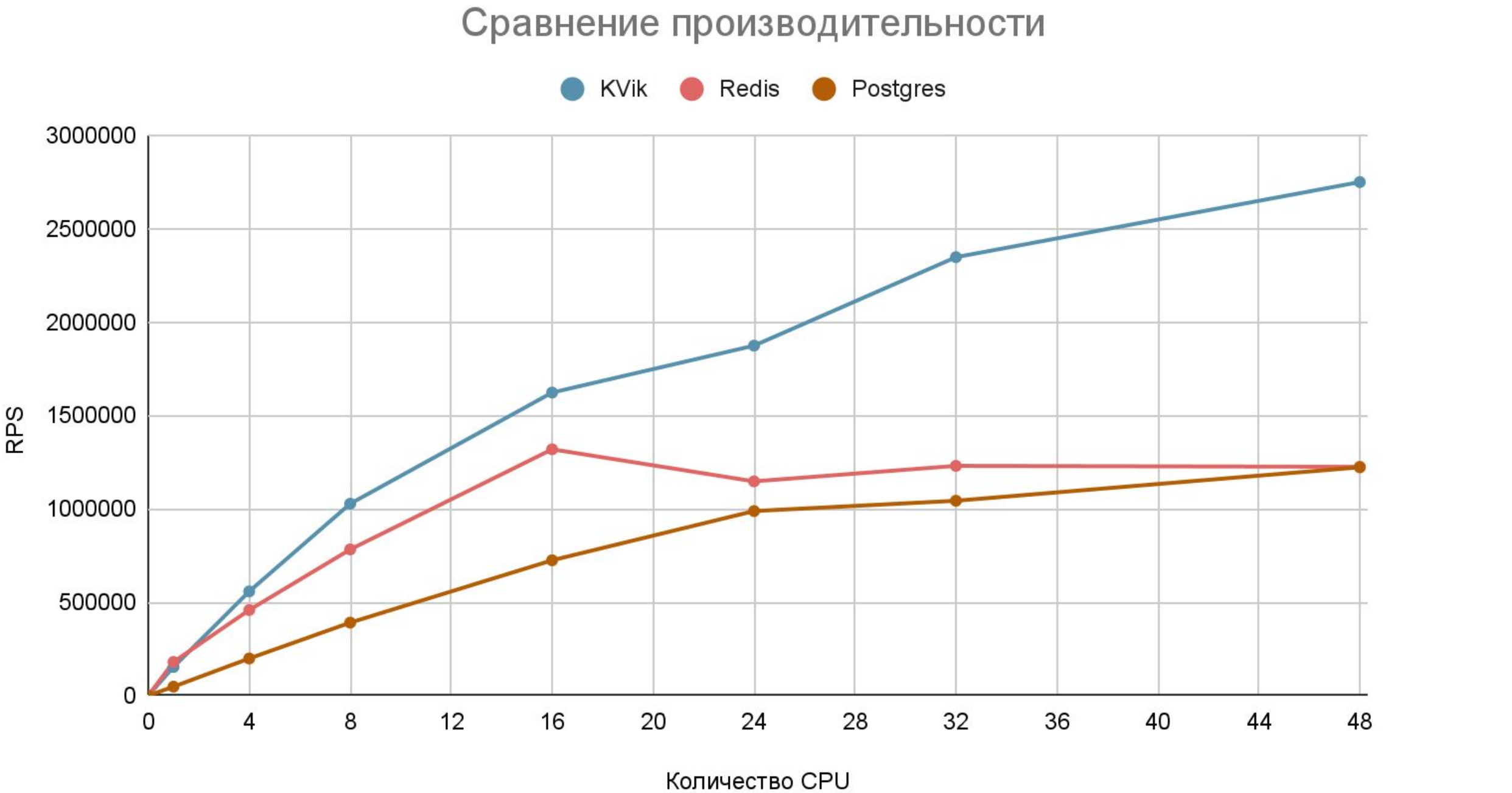
```python
# Функция для получения погодных данных из PostgreSQL
def get_weather(city):
    # Если в кеше данных нет, то KVik сам выполнит запрос к БД
    cached_data = r.get(f" 'CRUD:mydb.public.weather:{city}")
    if cached_data:
        return cached_data.decode('utf-8')
    else:
        return 'Данные о погоде для этого города нигде не найдены'
```

Тестирование

PostgresPro

- Postgres, Redis, KVik
- Всегда попадаем в кэш
- Размер одной записи 500 байт
- Случайные данные
- redis-benchmark (GET) и pgbench(SELECT)
- CPU от 1 до 48.


Сравнение производительности
KVik
Redis
Postgres
3000000
2500000
2000000
1500000
1000000
500000
0
RPS
0
4
8
12
16
Количество CPU


Сравнение производительности
KVik
Redis
Postgres
RPS
3000000
2500000
2000000
1500000
1000000
500000
0
0
4
8
12
16
20
24
28
32
36
40
44
48
Количество CPU

Когда стоит использовать KVik

PostgresPro

- Вам нужен эффективный кэш запросов
- Нет дополнительных ресурсов на сервер/лицензию

Когда не стоит использовать KVik

PostgresPro

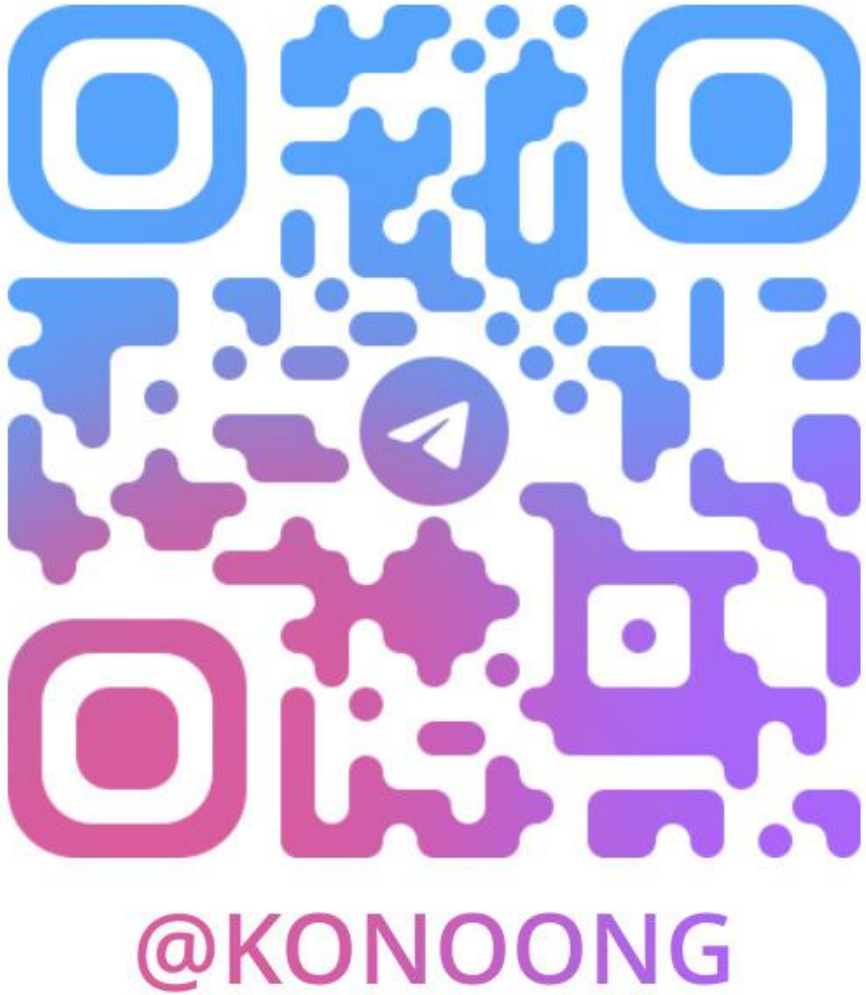
- Вам нужен не только кэш запросов
- Нужная поддержка сложных типов данных

Способ кэширования	+	−	Примеры
На Клиенте	• Нужна работа оффлайн • Одинаковые запросы у клиента	• Нужно хранить много данных • Разные запросы у клиента	• Кэш браузера • Кэш Мобильных приложений
На сервера	• Данные специфичны для инстанса приложения • Много клиентов	• Нужна сложная логика кэширования	• Маршрутизатор • Фреймворки пример: caffeine
На стороннем сервере	• Нужна сложная логика кэширования • Нужен согласованный кэш	• У вас простое приложение • Необхоима консистентность с СУБД	• Redis, • memcache • gernet

Способ кэширования	+	−	Примеры
в СУБД	• Всегда)	• Нужен кэш запросов	Буф. кэш PosgreSQL
KVik	• Нужен эффективный, простой в использовании кэш запросов с встроенной инвалидацией	• Нужен функционал шире, чем кэш запросов	KVik

PostgresPro

Ответим на ваши вопросы!



Ахмедов Влад