Прософт Системы

ПРОБЛЕМЫ EMBEDDED ИЛИ КАК МЫ ОТ SQLITE УШЛИ

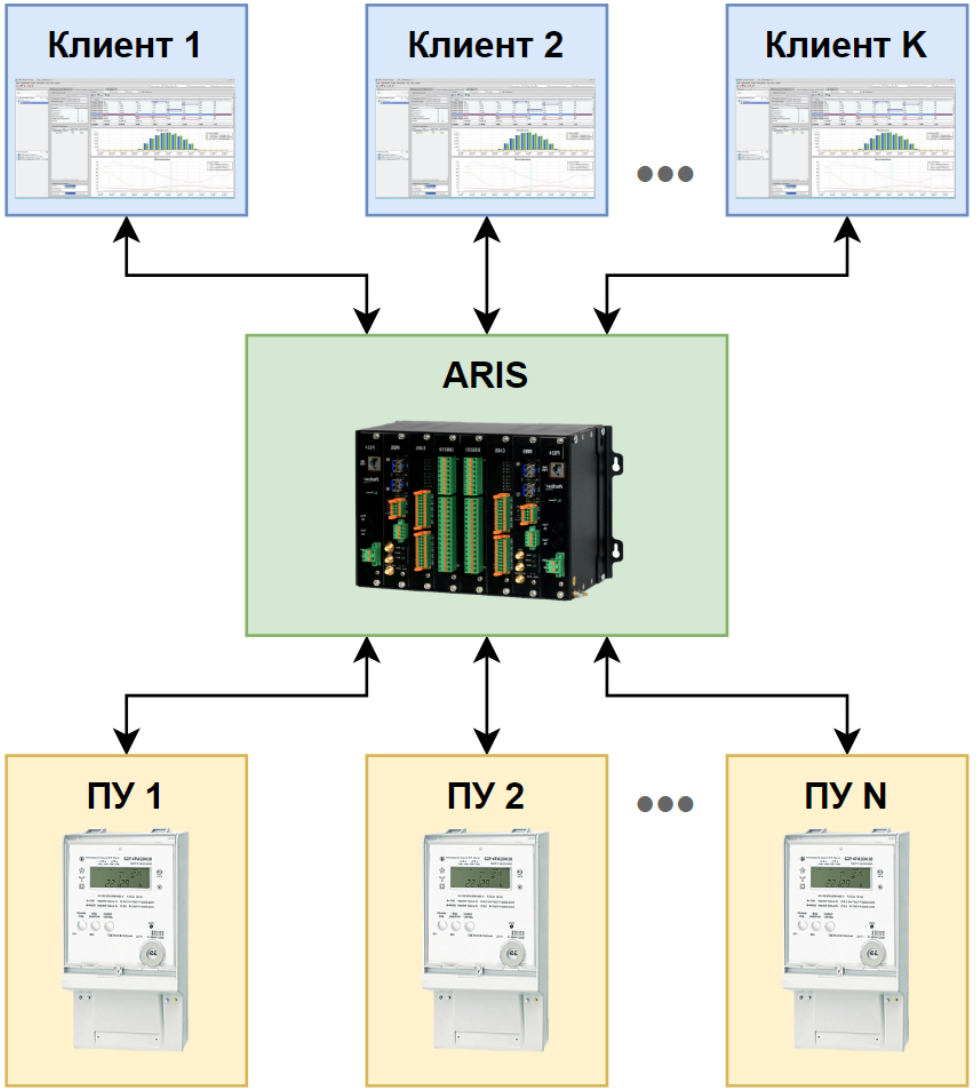
Беляев Михаил

БОЛЬШАЯ ЛИНЕЙКА УСТРОЙСТВ

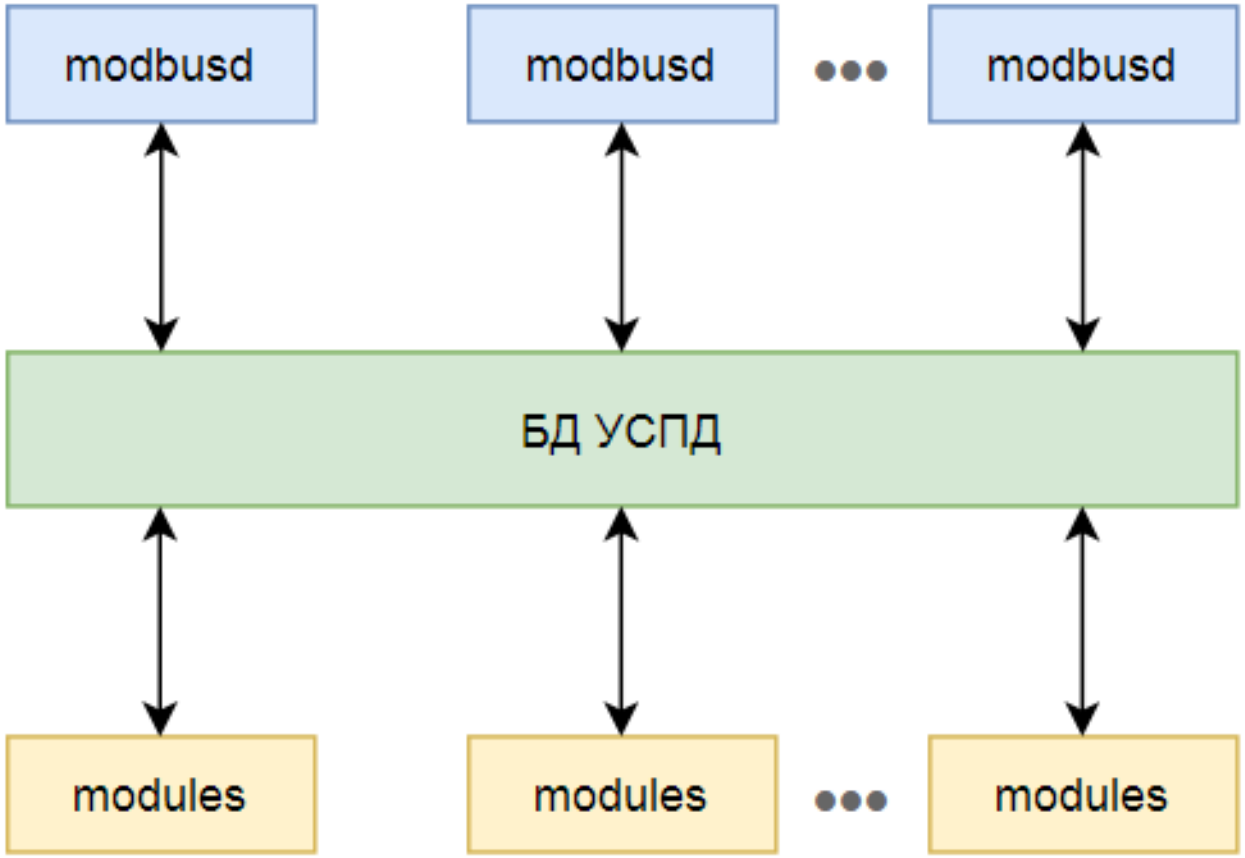
МЛАДШАЯ ЛИНЕЙКА УСТРОЙСТВ

Показатель		Значение
Процессор	Тип	AT91SAM9260
	Архитектура	ARM
	Разрядность	32
	Частота	198 МГц
ОЗУ		128 МБ
ПЗУ		236 МБ

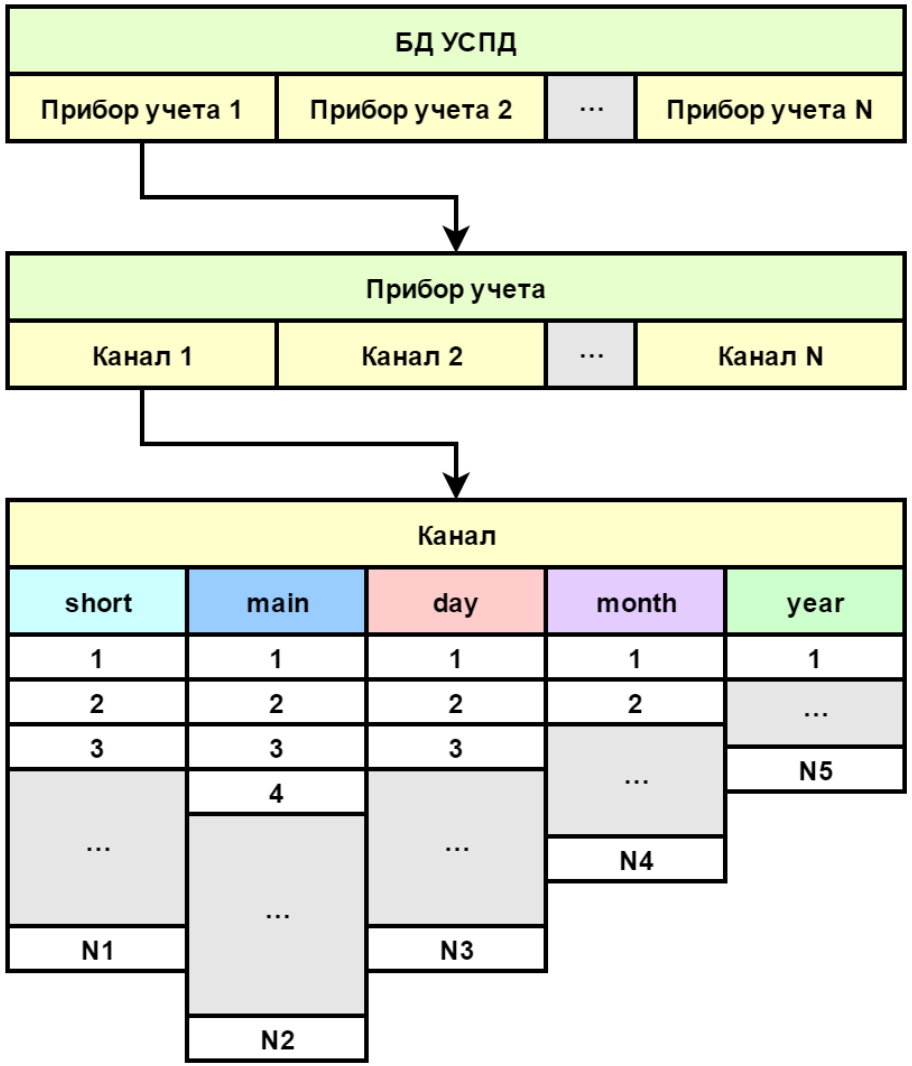
ФУНКЦИОНАЛЬНАЯ СХЕМА РАБОТЫ ARIS

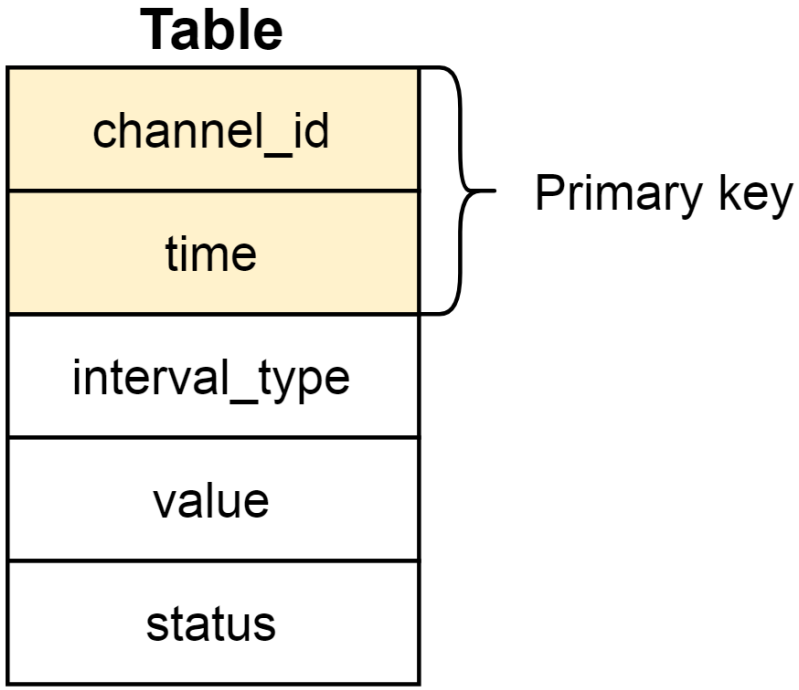


ФУНКЦИОНАЛЬНАЯ СХЕМА РАБОТЫ УСПД



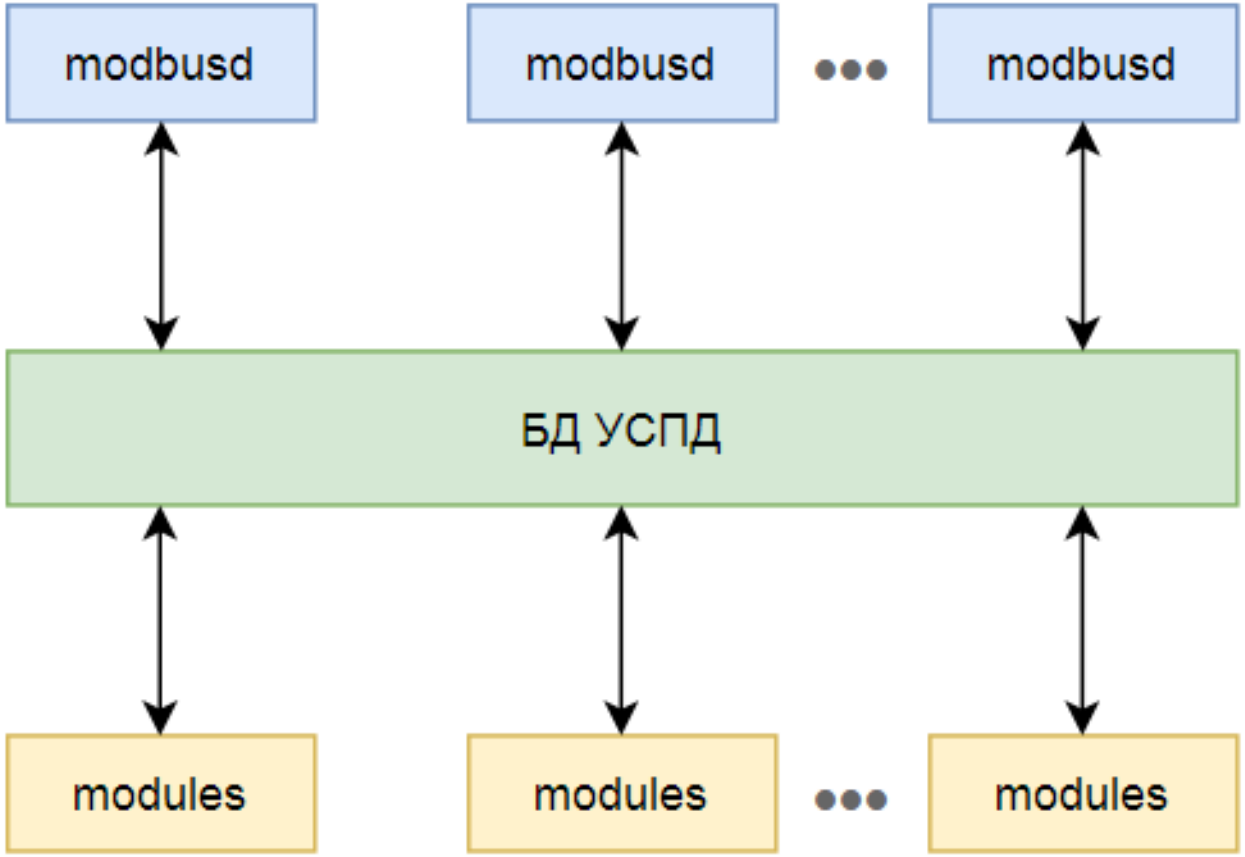
СТРУКТУРА БД УСПД



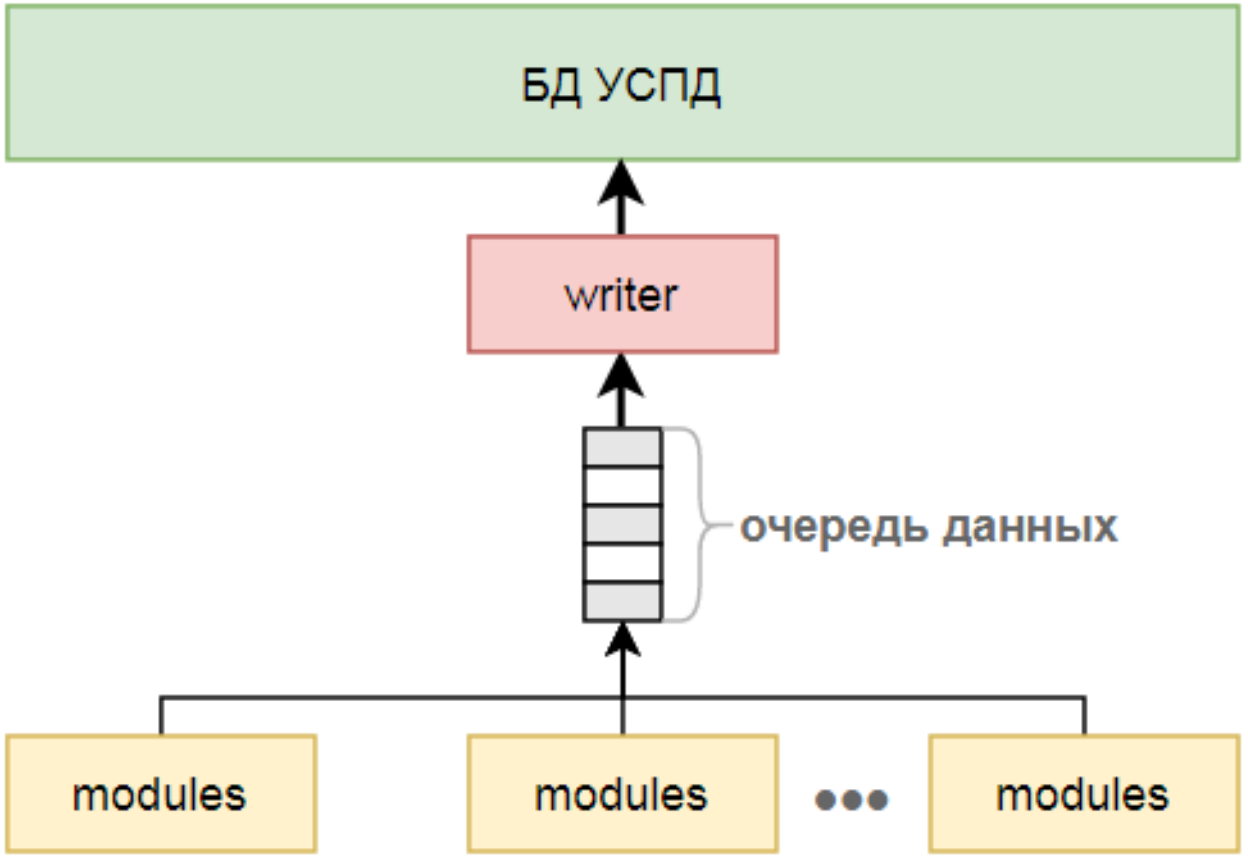


ЗАПИСЬ ДАННЫХ В БД SQLITE

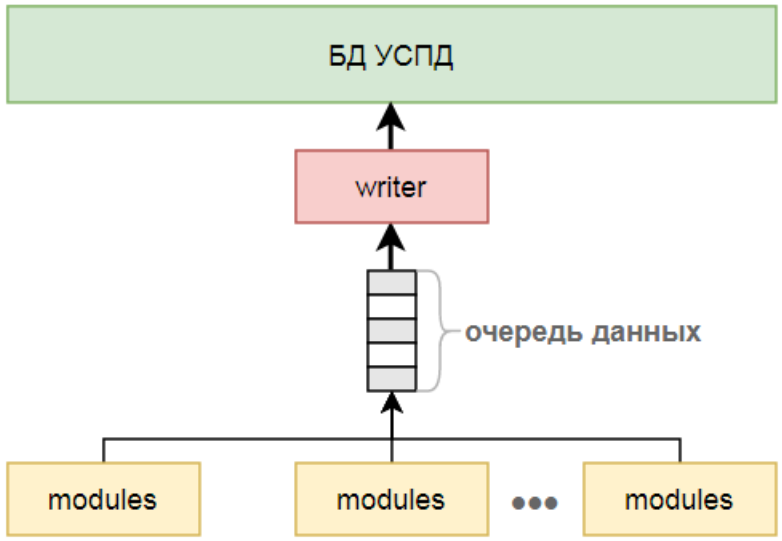
- Процессов modules много.
- Процессы modules пишут данные небольшими блоками.
- Процессы modbusd могут в любой момент затребовать интенсивное чтение большими блоками данных.
- Очень большая конкуренция между процессами за доступ к БД УСПД.

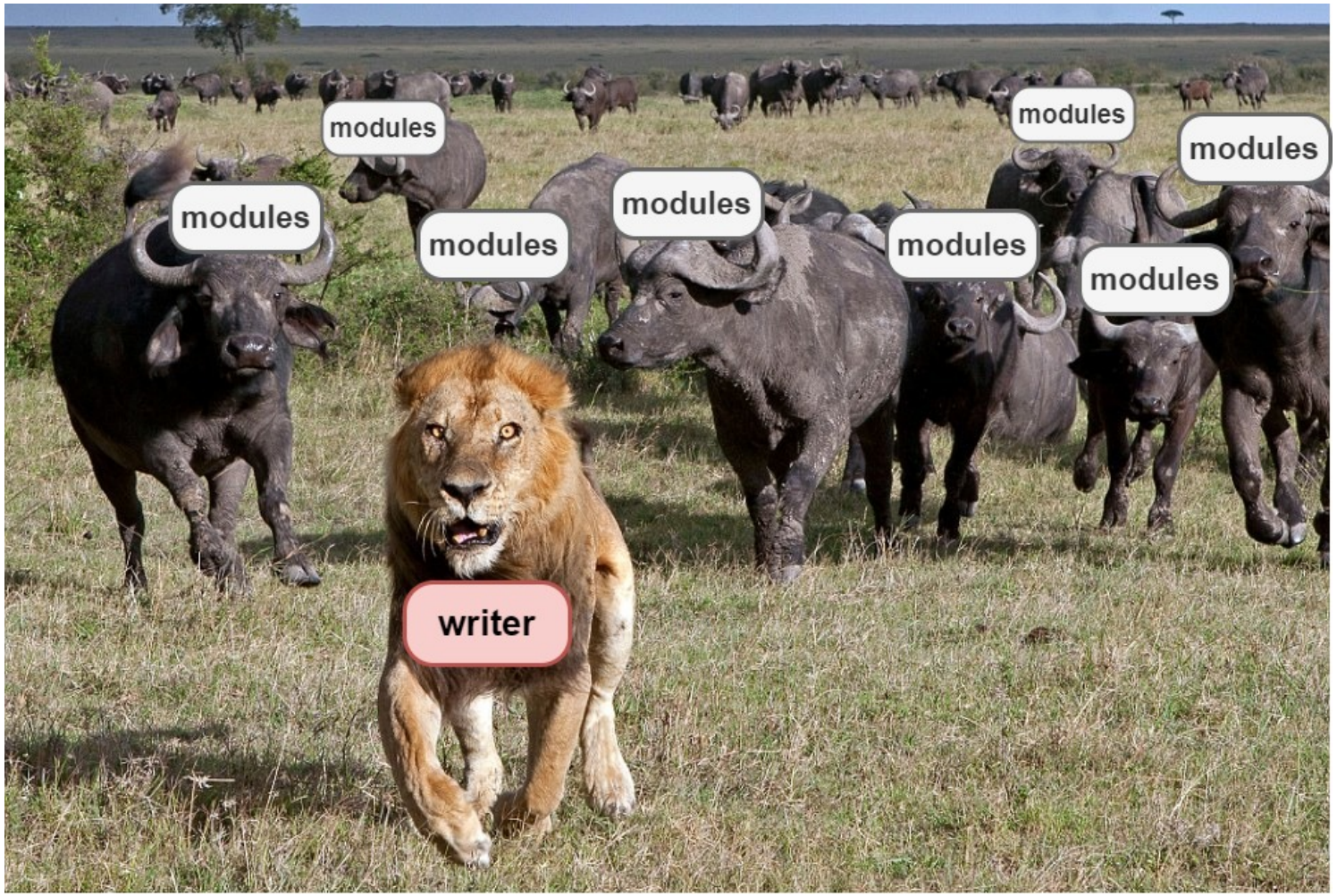


ПИШЕМ ДАННЫЕ БОЛЬШИМИ БЛОКАМИ

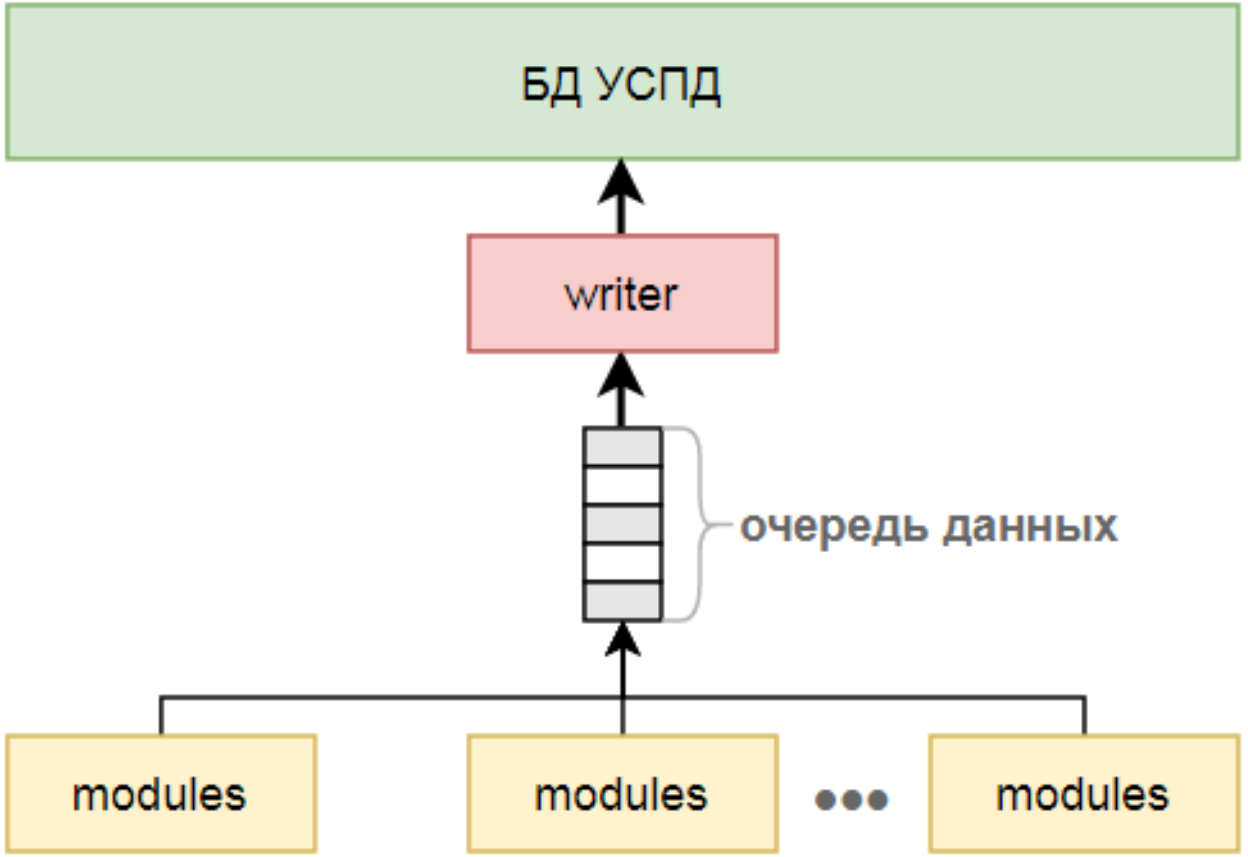


ПРОБЛЕМЫ ИСПОЛЬЗОВАНИЯ ЕДИНОЙ ОЧЕРЕДИ





ПИШЕМ ДАННЫЕ БОЛЬШИМИ БЛОКАМИ



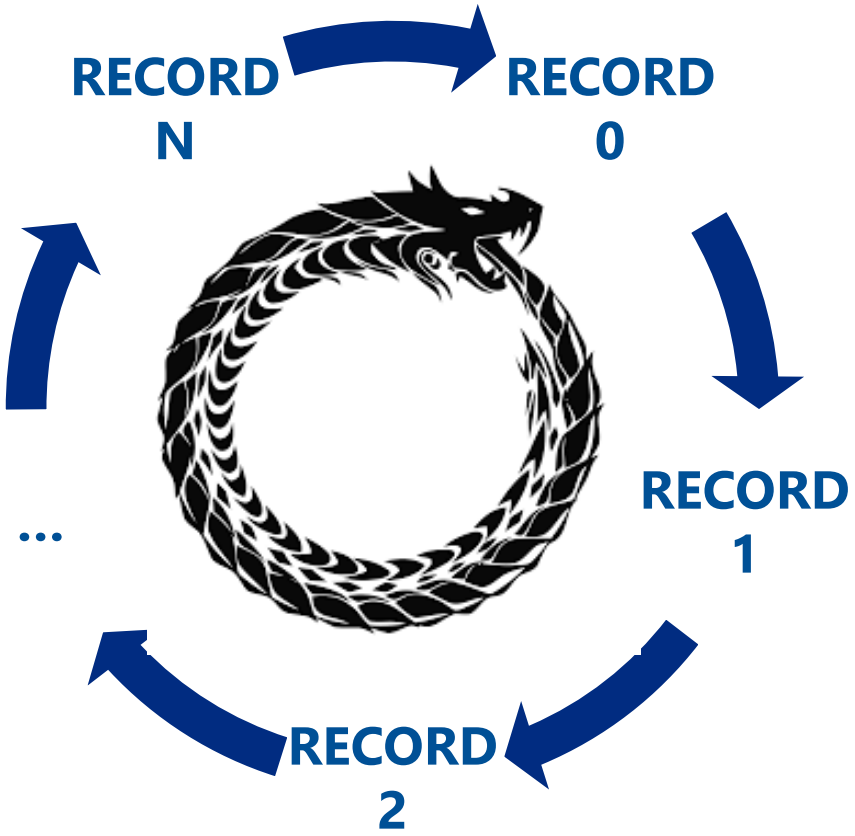
ПРОЧИЕ СЛОЖНОСТИ

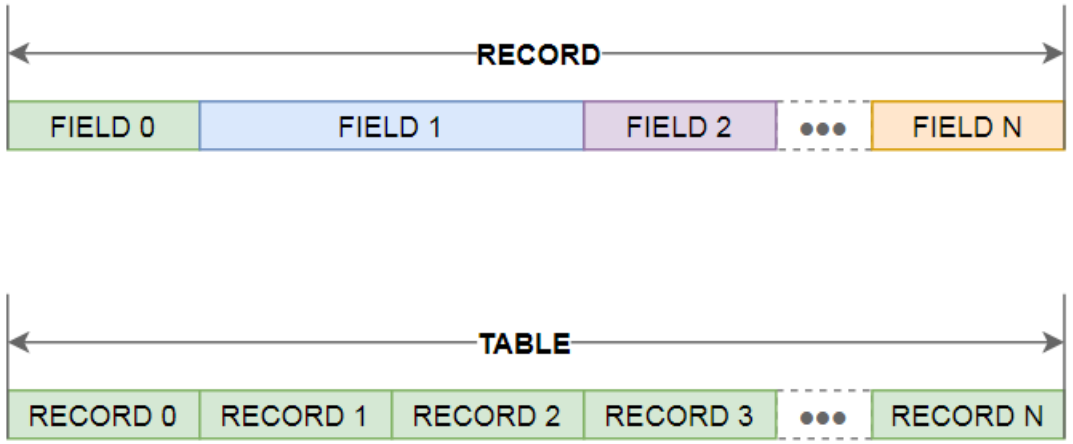
- Плохо прогнозируемый конечный размер БД.
- Высокая фрагментация данных.
- Требуется контролировать глубину хранения данных.
- Низкая скорость чтения/записи данных.
- Глобальная транзакция.

ТРЕБОВАНИЯ К ХРАНИЛИЩУ

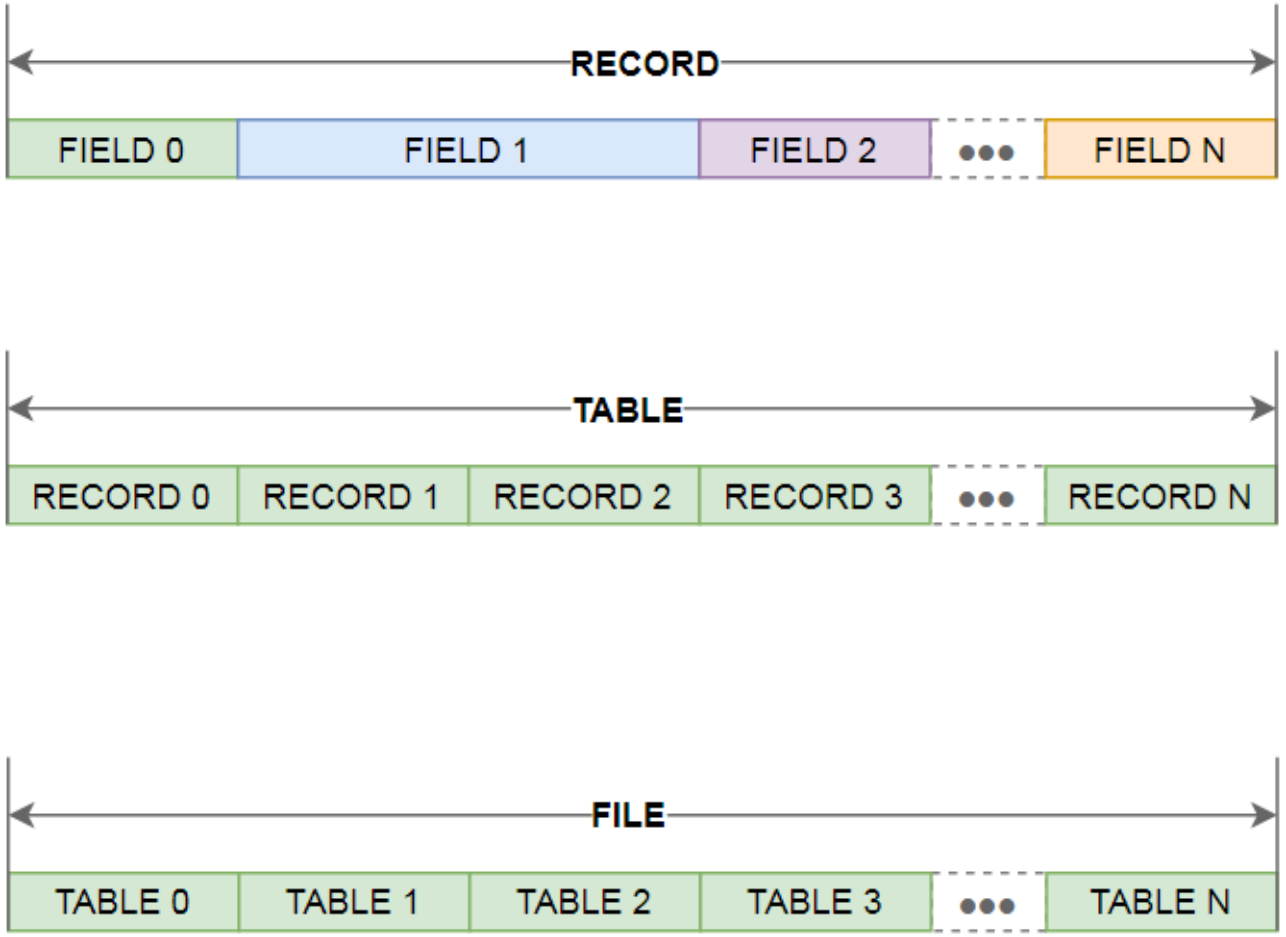
- Точное определение размера хранилища.
- Поддержка одновременного доступа к данным из нескольких процессов.
- Автоматический контроль глубины хранения данных.
- Блокировка на уровне отдельных таблиц.
- Контроль целостности данных (поддержка транзакций).
- ~~Можно грабить корованы.~~

АРХИТЕКТУРА ТАБЛИЦЫ

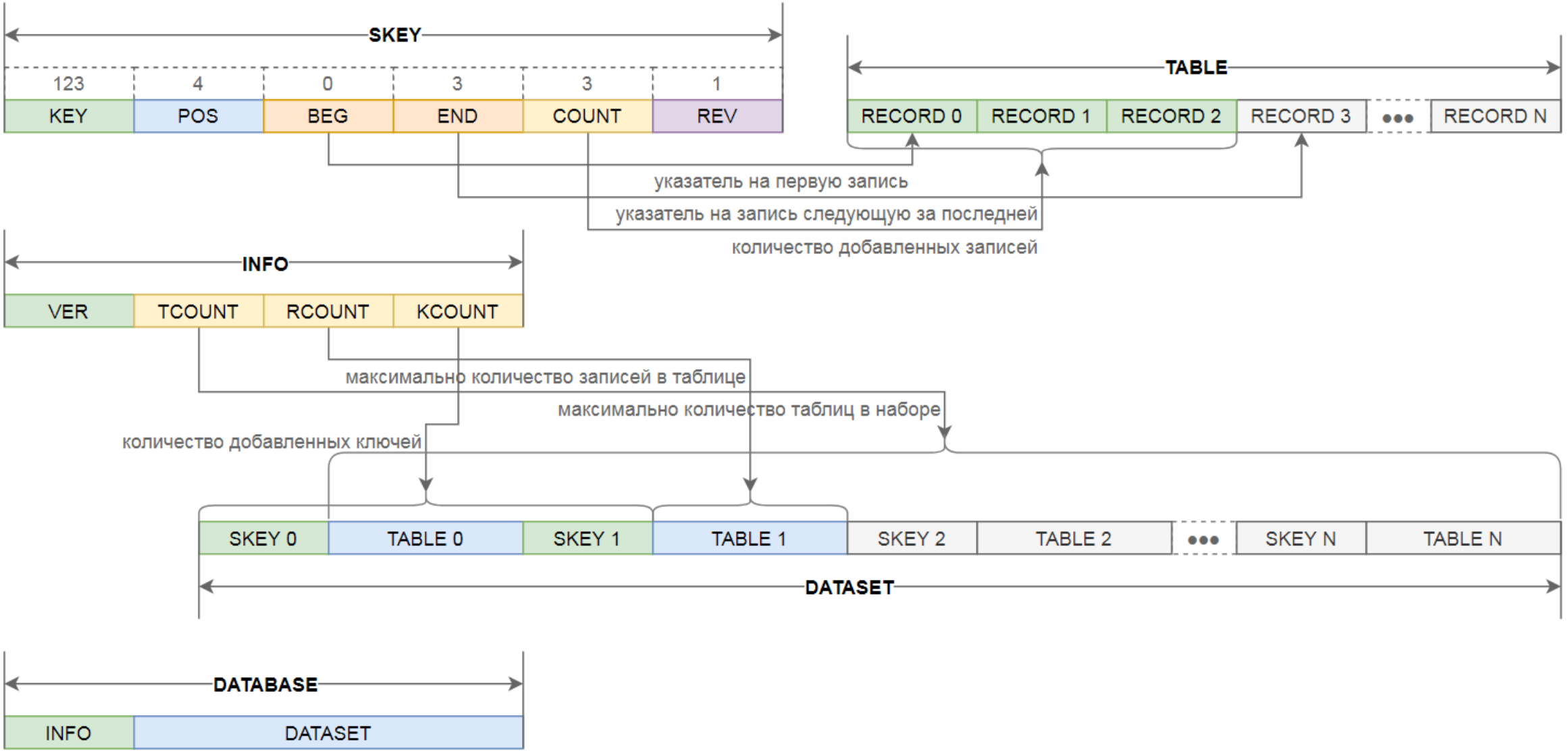




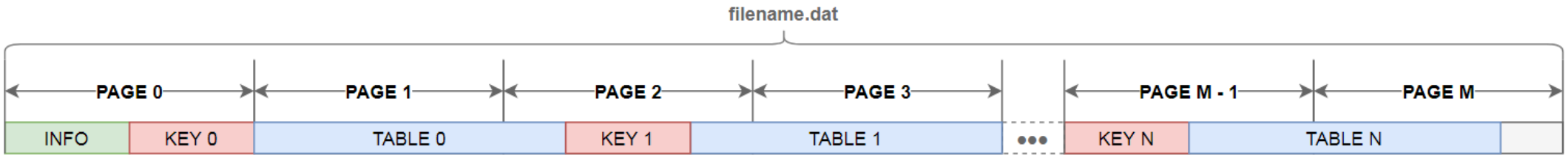
АРХИТЕКТУРА ТАБЛИЦЫ



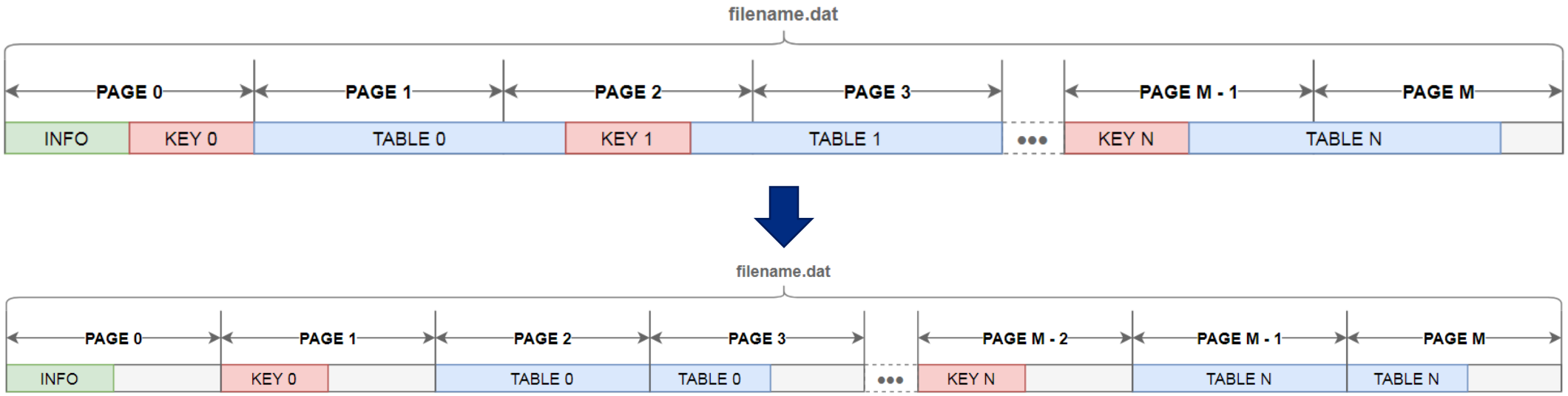
АРХИТЕКТУРА ФАЙЛА



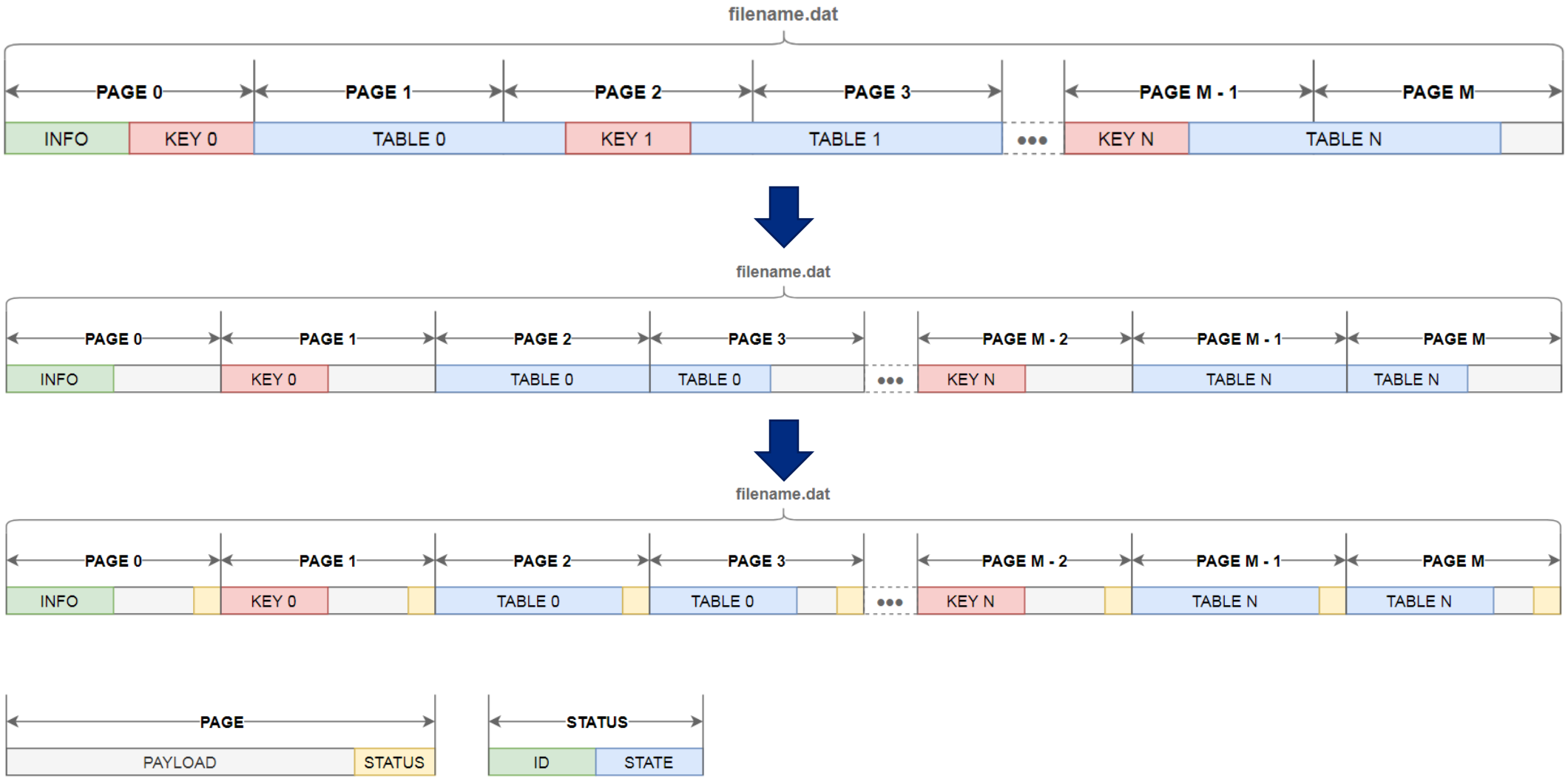
АРХИТЕКТУРА ФАЙЛА



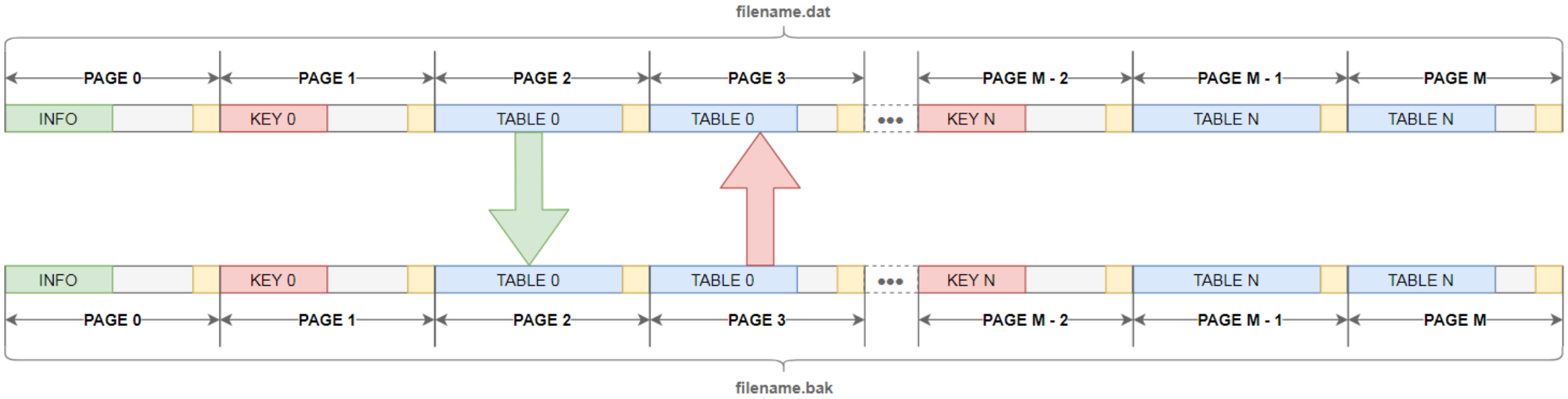
АРХИТЕКТУРА ФАЙЛА



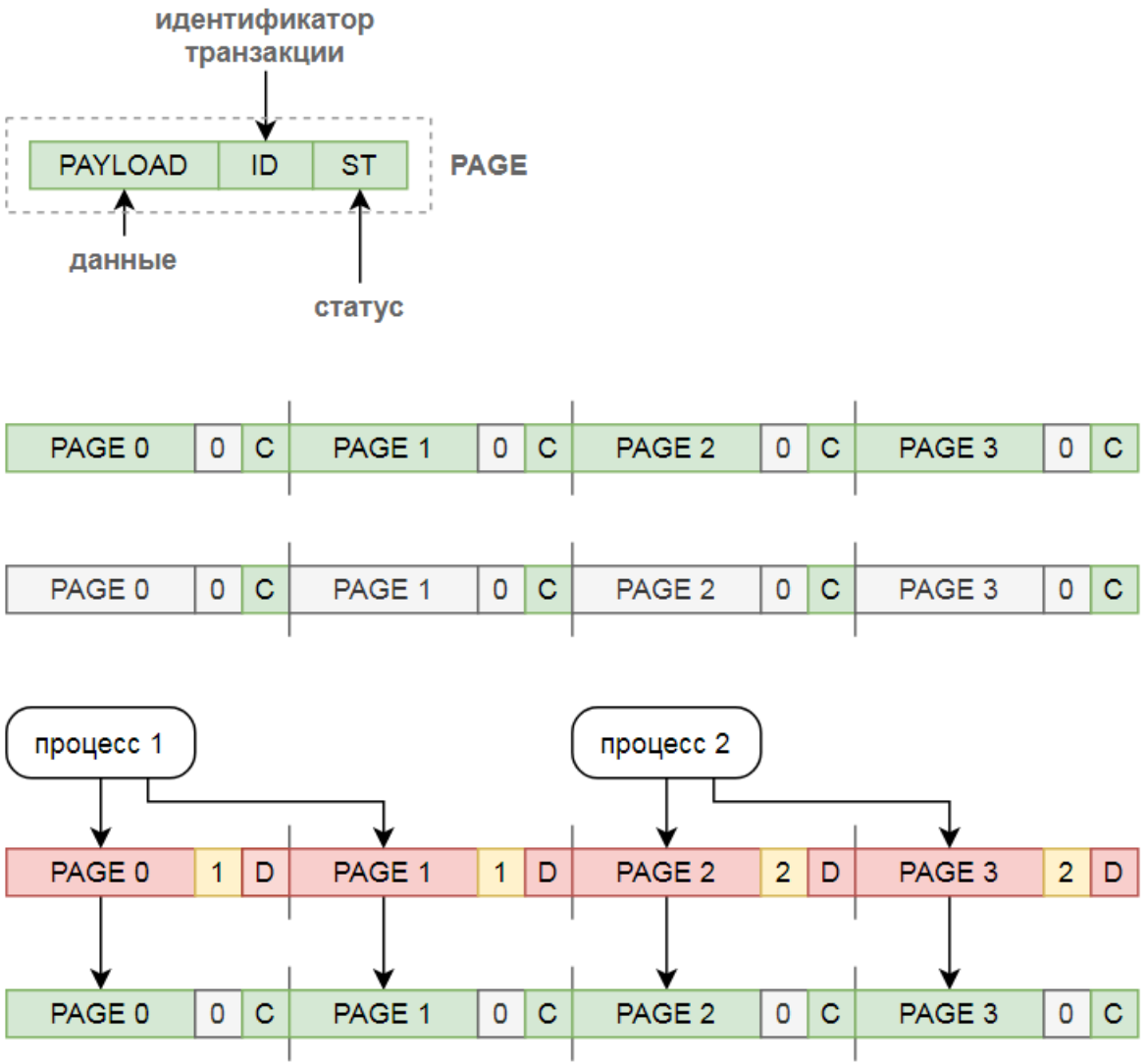
АРХИТЕКТУРА ФАЙЛА



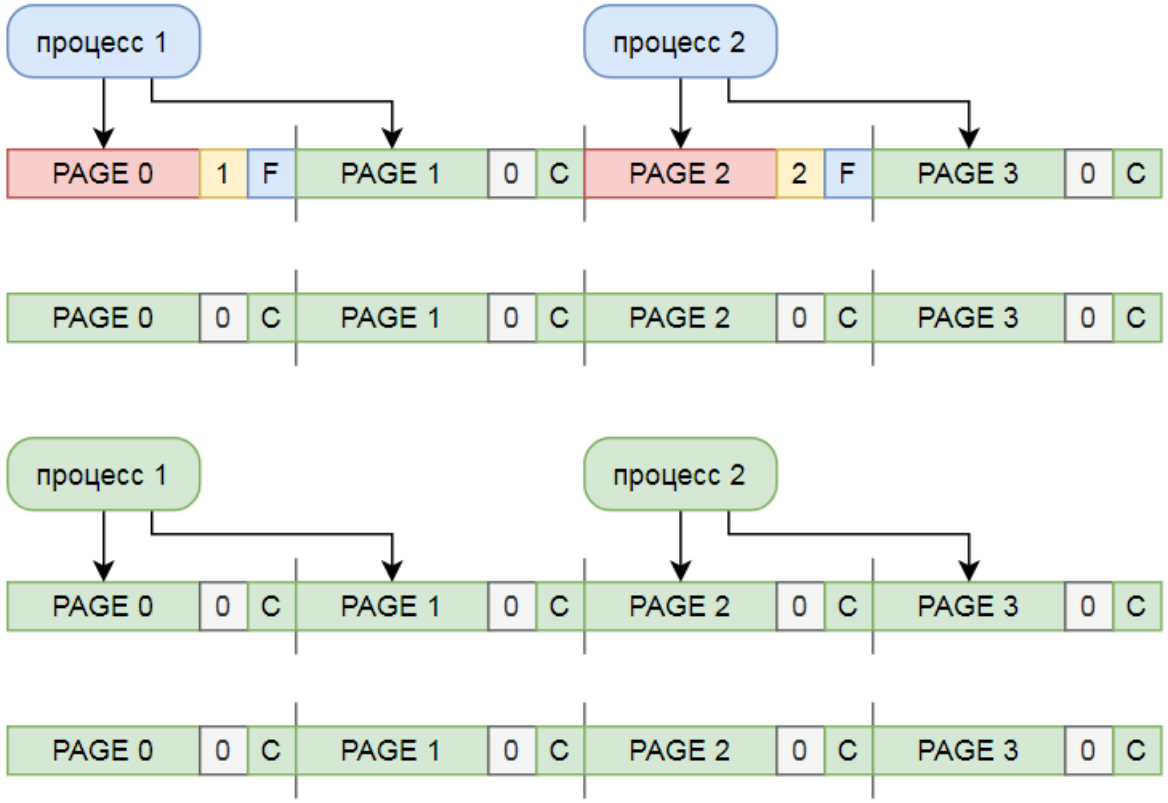
ФОРМИРОВАНИЕ ФАЙЛА ОТКАТА ТРАНЗАКЦИИ



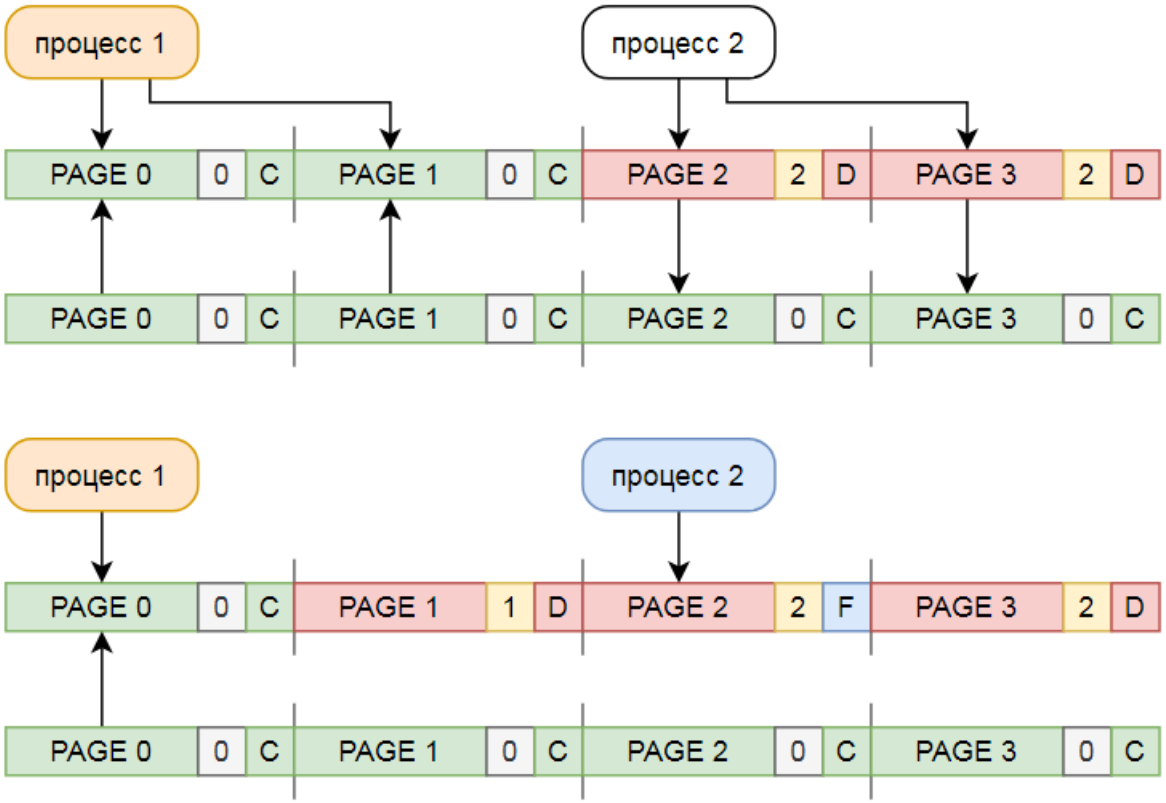
ДОСТУП К ДАННЫМ ИЗ РАЗНЫХ ПРОЦЕССОВ



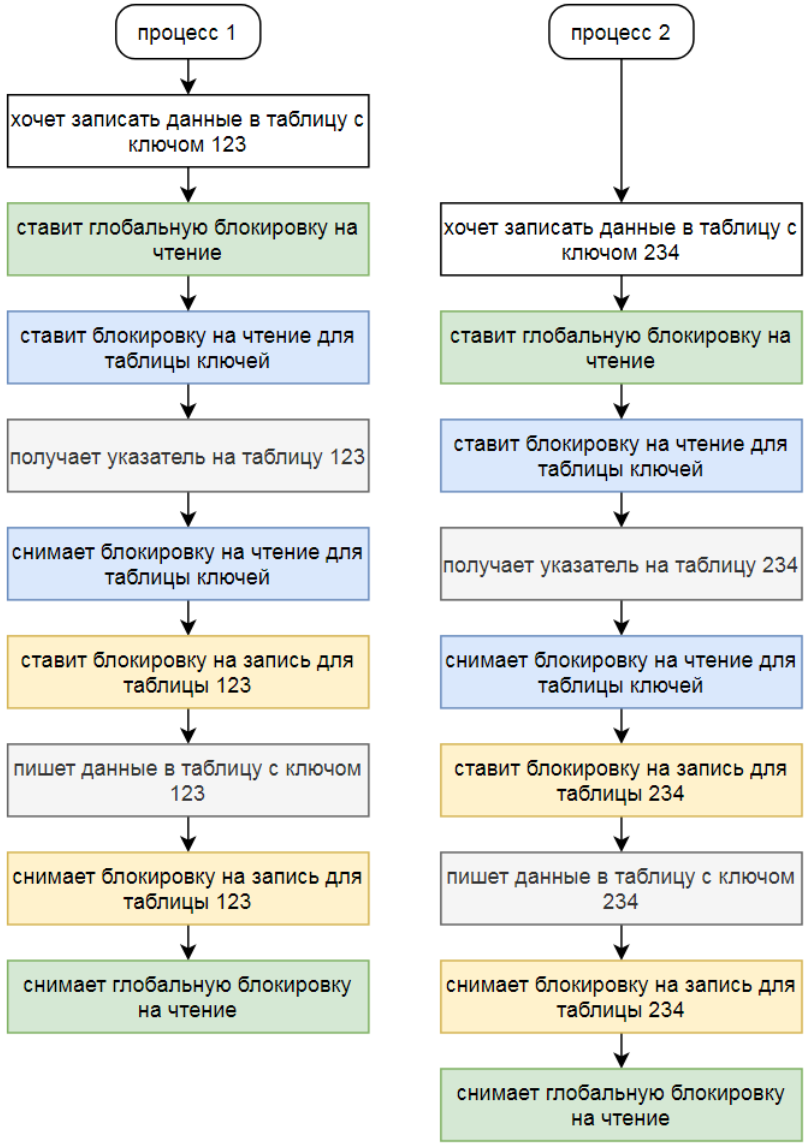
ДОСТУП К ДАННЫМ ИЗ РАЗНЫХ ПРОЦЕССОВ



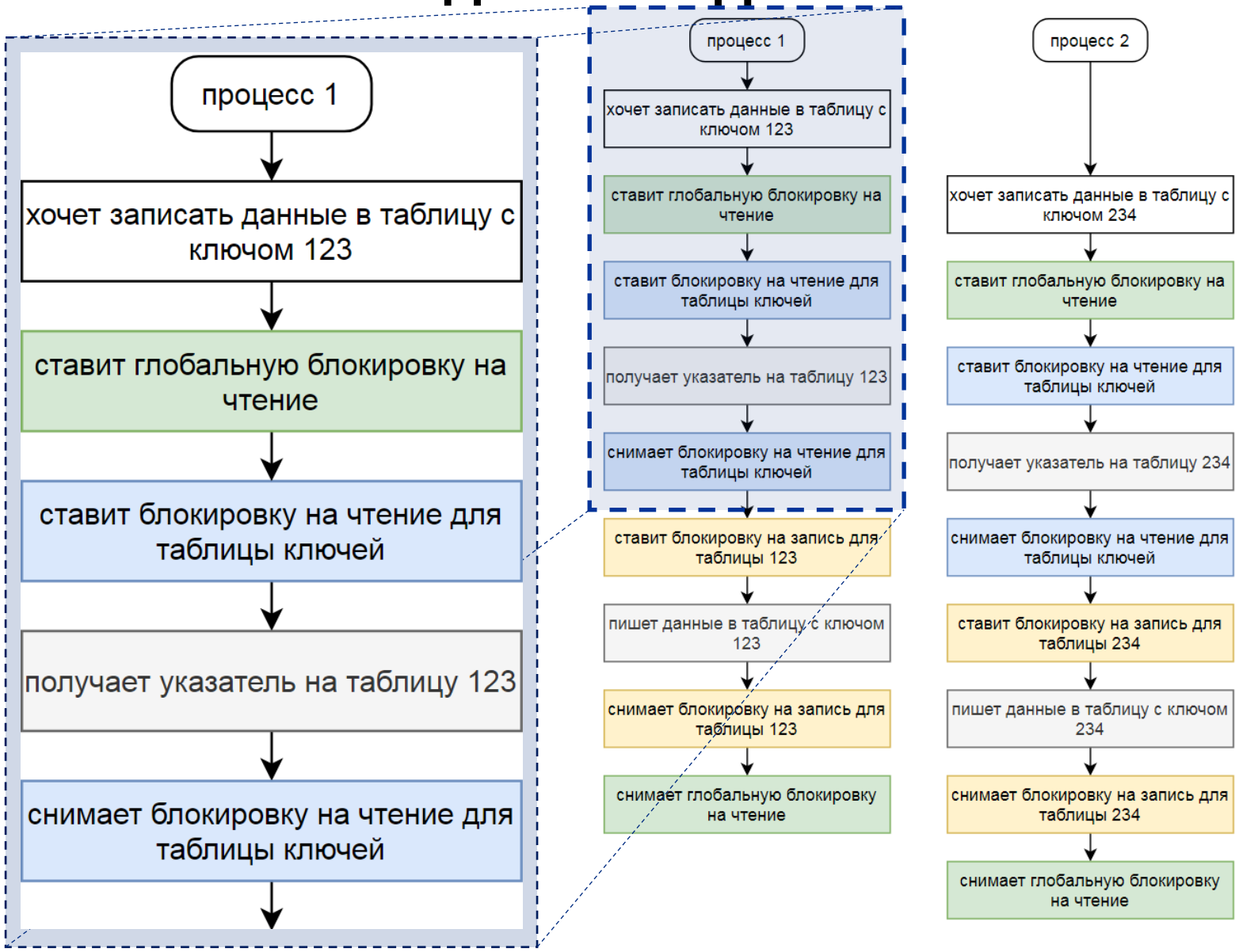
ОТКАТ И ФИКСАЦИЯ ТРАНЗАКЦИИ



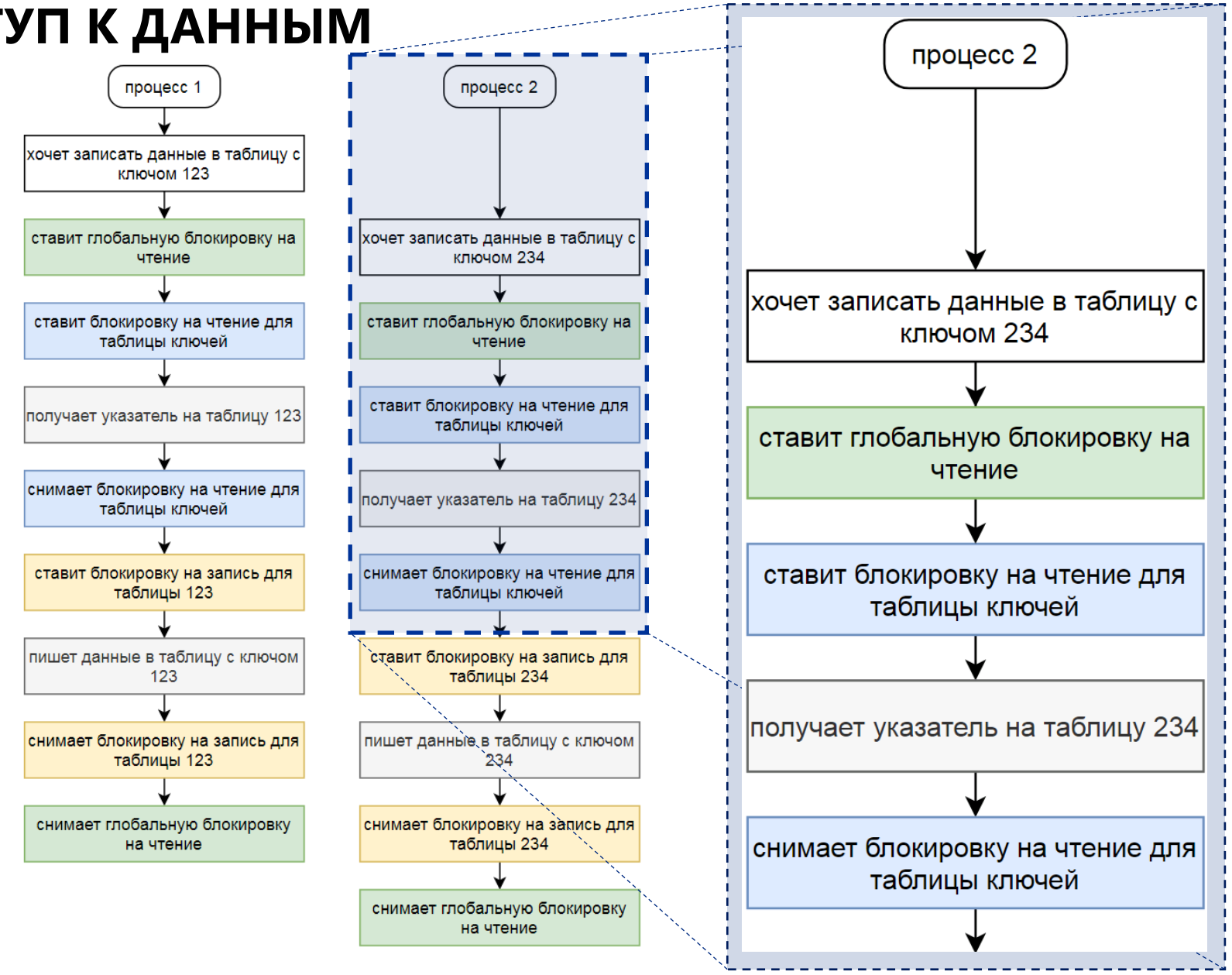
ПАРАЛЛЕЛЬНЫЙ ДОСТУП К ДАННЫМ



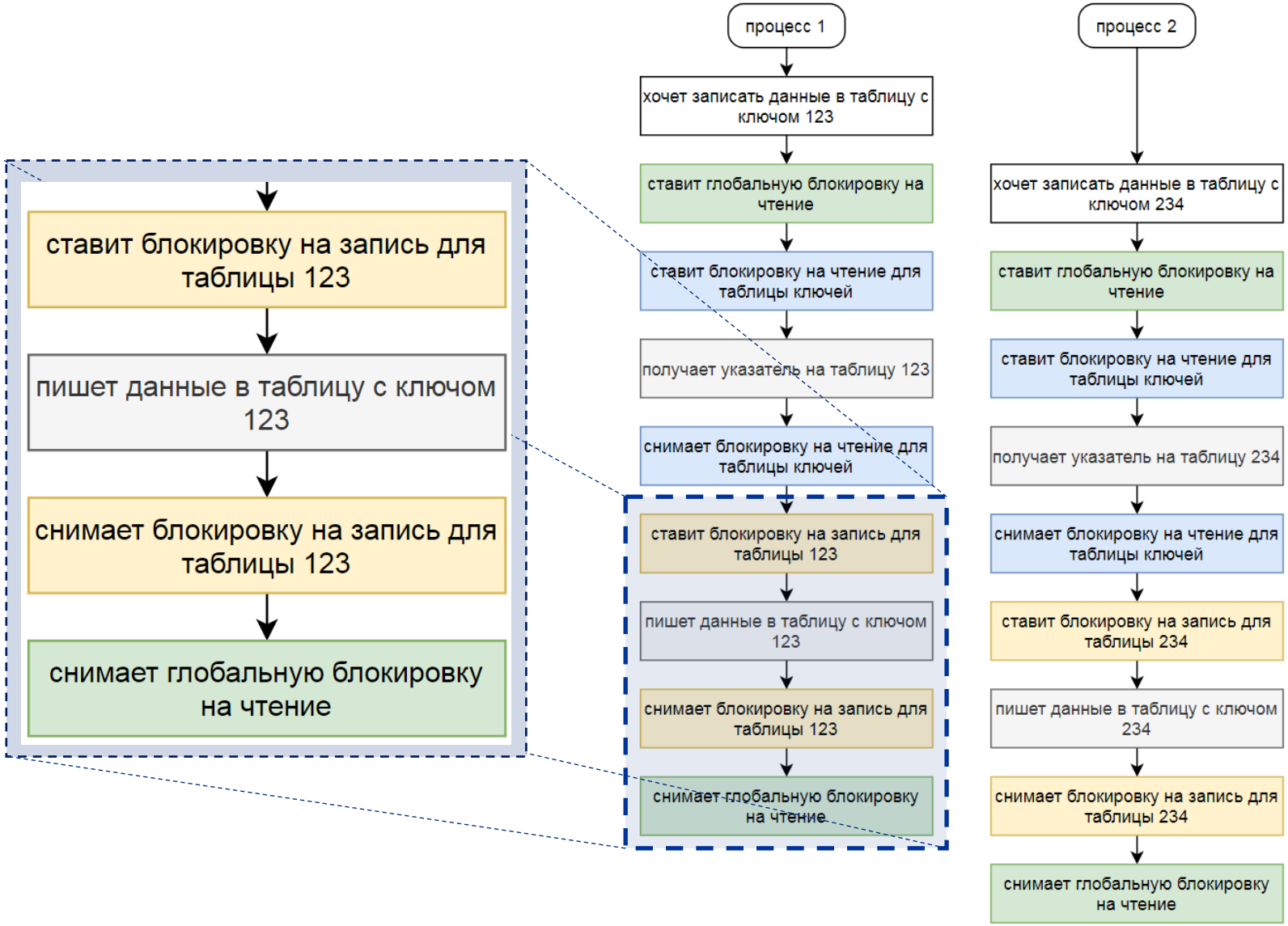
ПАРАЛЛЕЛЬНЫЙ ДОСТУП К ДАННЫМ



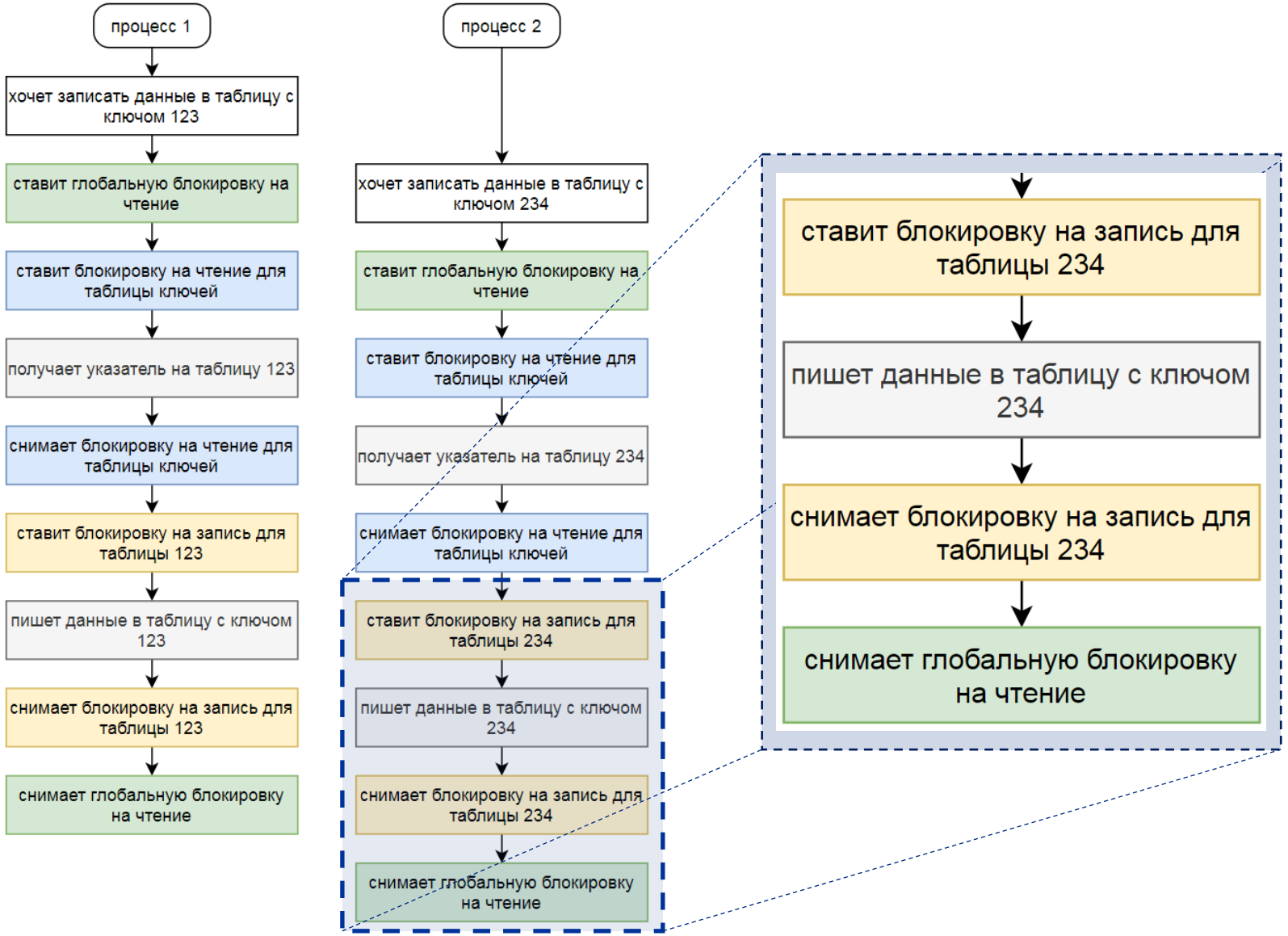
ПАРАЛЛЕЛЬНЫЙ ДОСТУП К ДАННЫМ



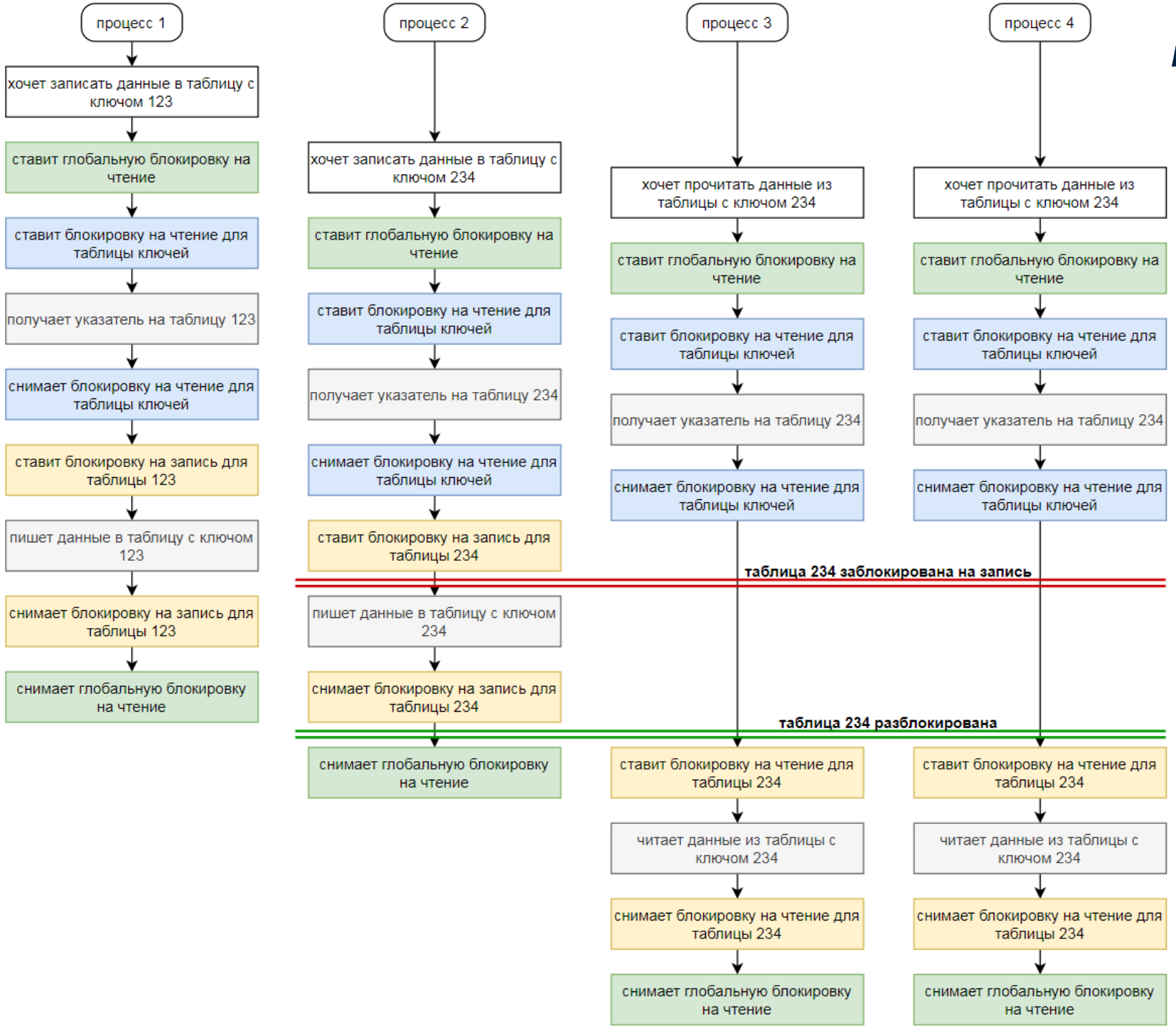
ПАРАЛЛЕЛЬНЫЙ ДОСТУП К ДАННЫМ



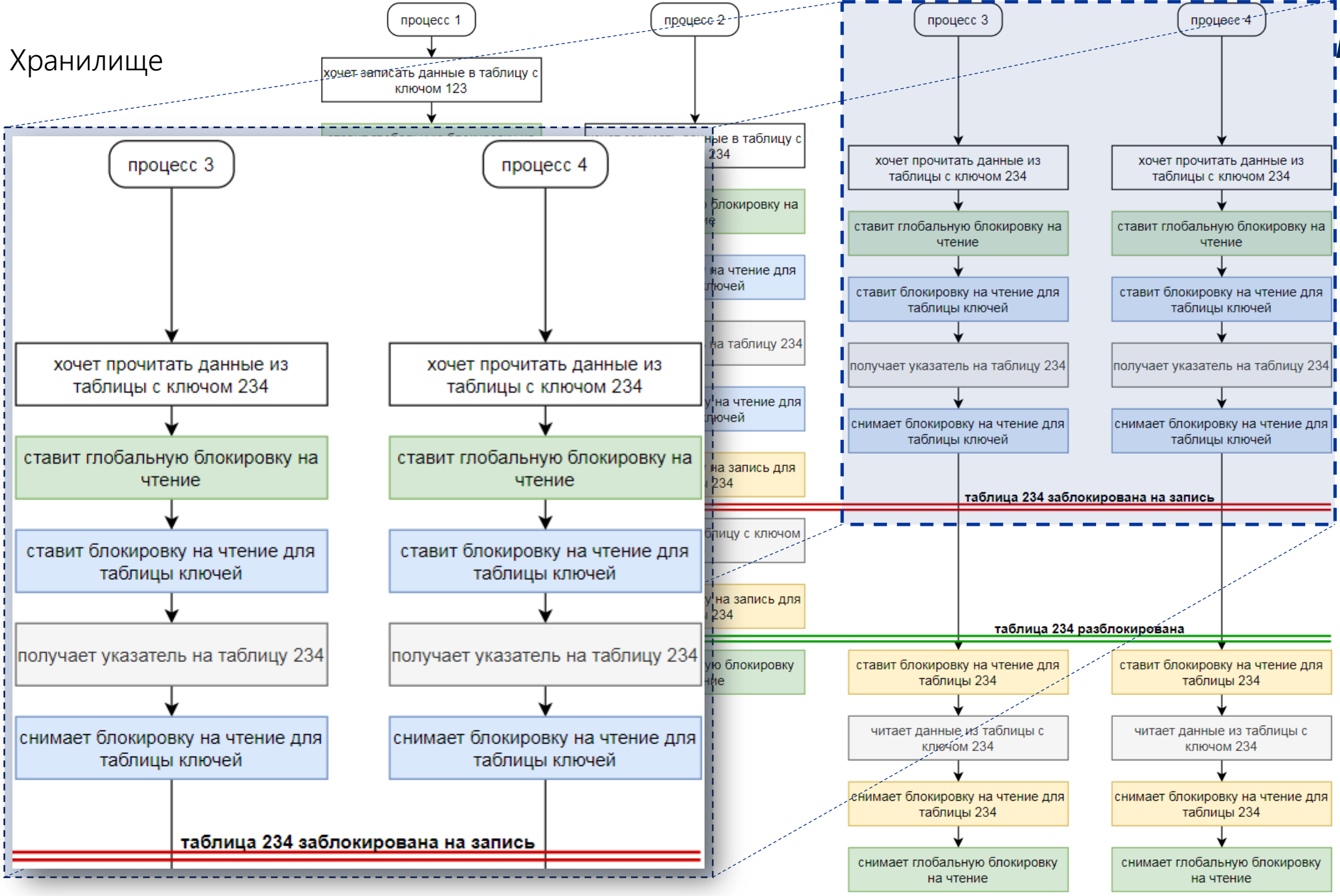
ПАРАЛЛЕЛЬНЫЙ ДОСТУП К ДАННЫМ



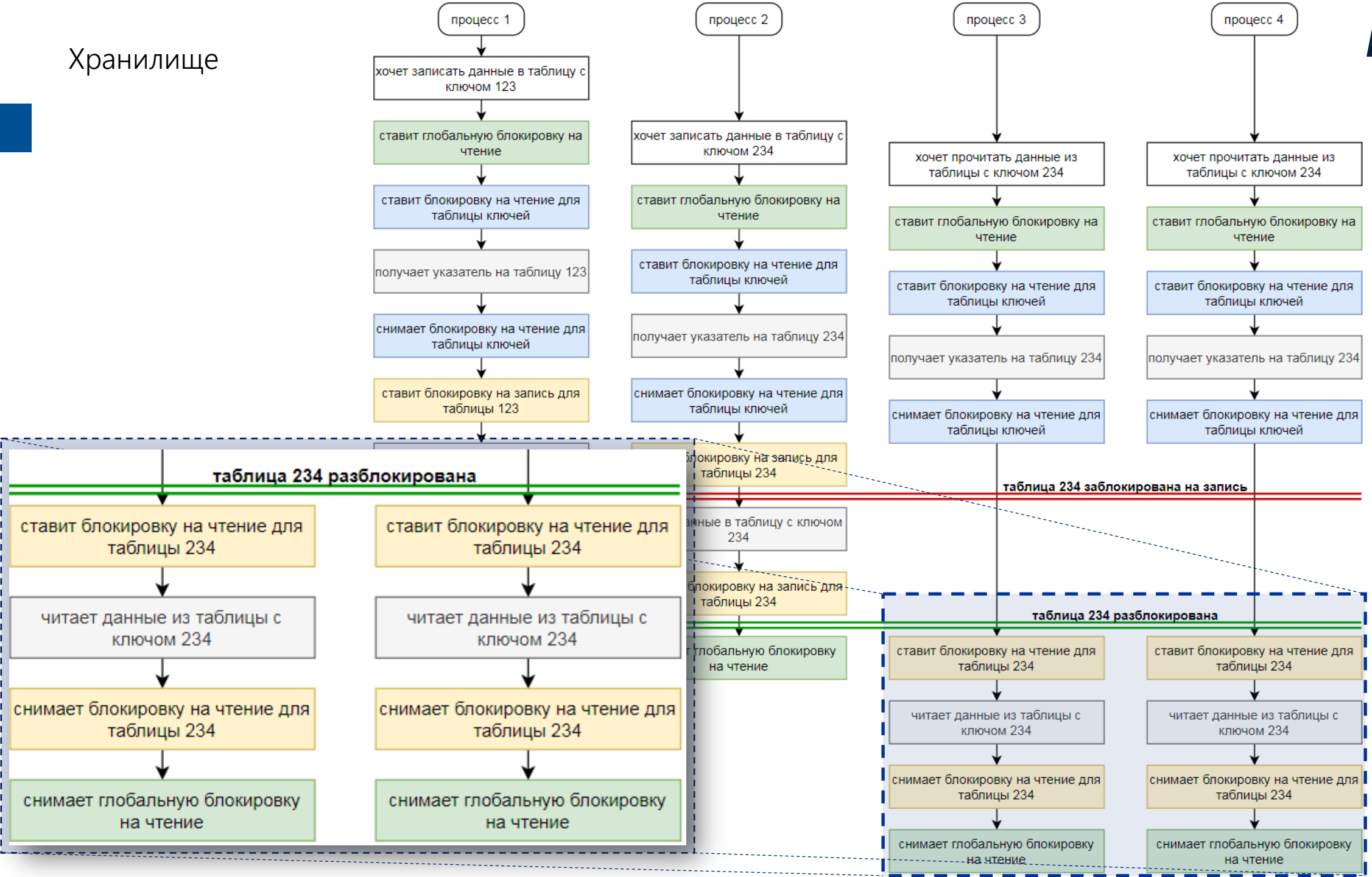
Хранилище


процесс 1
хочет записать данные в таблицу с ключом 123
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 123
снимает блокировку на чтение для таблицы ключей
ставит блокировку на запись для таблицы 123
пишет данные в таблицу с ключом 123
снимает блокировку на запись для таблицы 123
снимает глобальную блокировку на чтение
процесс 2
хочет записать данные в таблицу с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
ставит блокировку на запись для таблицы 234
пишет данные в таблицу с ключом 234
снимает блокировку на запись для таблицы 234
снимает глобальную блокировку на чтение
процесс 3
хочет прочитать данные из таблицы с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
ставит блокировку на чтение для таблицы 234
читает данные из таблицы с ключом 234
снимает блокировку на чтение для таблицы 234
снимает глобальную блокировку на чтение
процесс 4
хочет прочитать данные из таблицы с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
ставит блокировку на чтение для таблицы 234
читает данные из таблицы с ключом 234
снимает блокировку на чтение для таблицы 234
снимает глобальную блокировку на чтение
таблица 234 заблокирована на запись
таблица 234 разблокирована

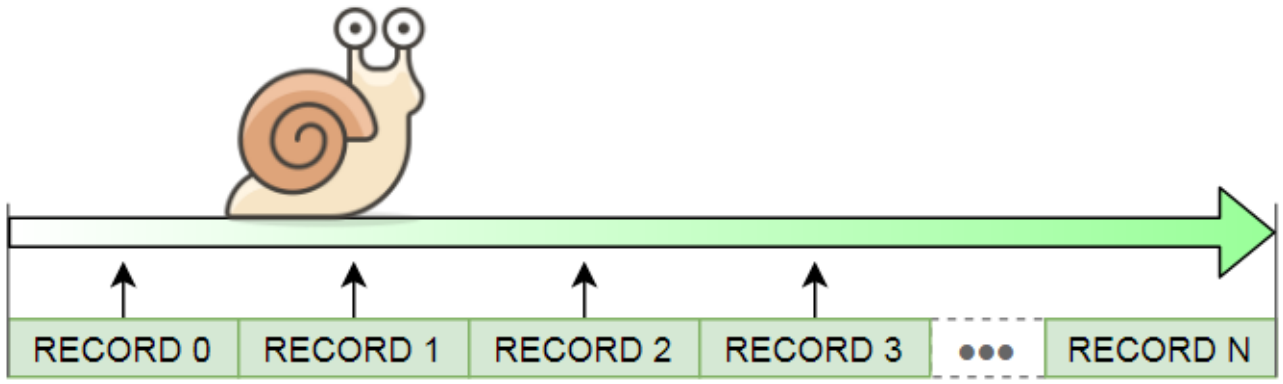
Хранилище


процесс 1
процесс 2
процесс 3
процесс 4
хочет записать данные в таблицу с ключом 123
хочет прочитать данные из таблицы с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
таблица 234 заблокирована на запись
таблица 234 разблокирована
ставит блокировку на чтение для таблицы 234
читает данные из таблицы с ключом 234
снимает блокировку на чтение для таблицы 234
снимает глобальную блокировку на чтение

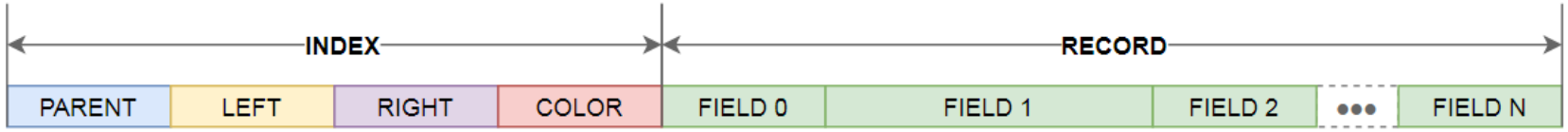
Хранилище

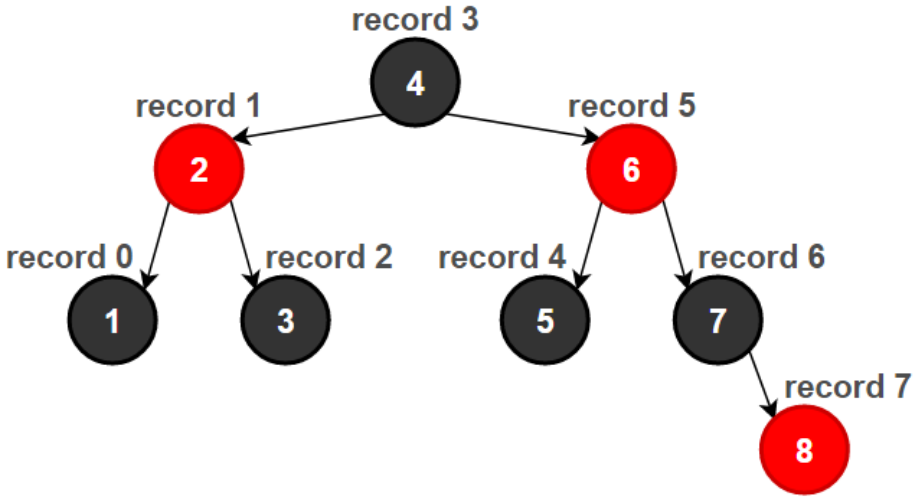

процесс 1
хочет записать данные в таблицу с ключом 123
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 123
снимает блокировку на чтение для таблицы ключей
ставит блокировку на запись для таблицы 123
процесс 2
хочет записать данные в таблицу с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
процесс 3
хочет прочитать данные из таблицы с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
процесс 4
хочет прочитать данные из таблицы с ключом 234
ставит глобальную блокировку на чтение
ставит блокировку на чтение для таблицы ключей
получает указатель на таблицу 234
снимает блокировку на чтение для таблицы ключей
таблица 234 разблокирована
ставит блокировку на чтение для таблицы 234
читает данные из таблицы с ключом 234
снимает блокировку на чтение для таблицы 234
снимает глобальную блокировку на чтение
таблица 234 заблокирована на запись
таблица 234 разблокирована
ставит блокировку на чтение для таблицы 234
читает данные из таблицы с ключом 234
снимает блокировку на чтение для таблицы 234
снимает глобальную блокировку на чтение

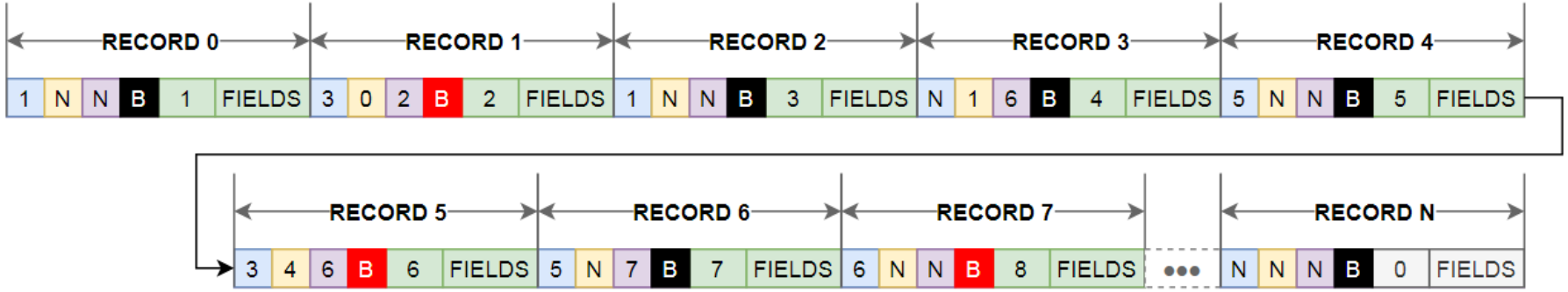
СЛУЧАЙНЫЕ ДАННЫЕ



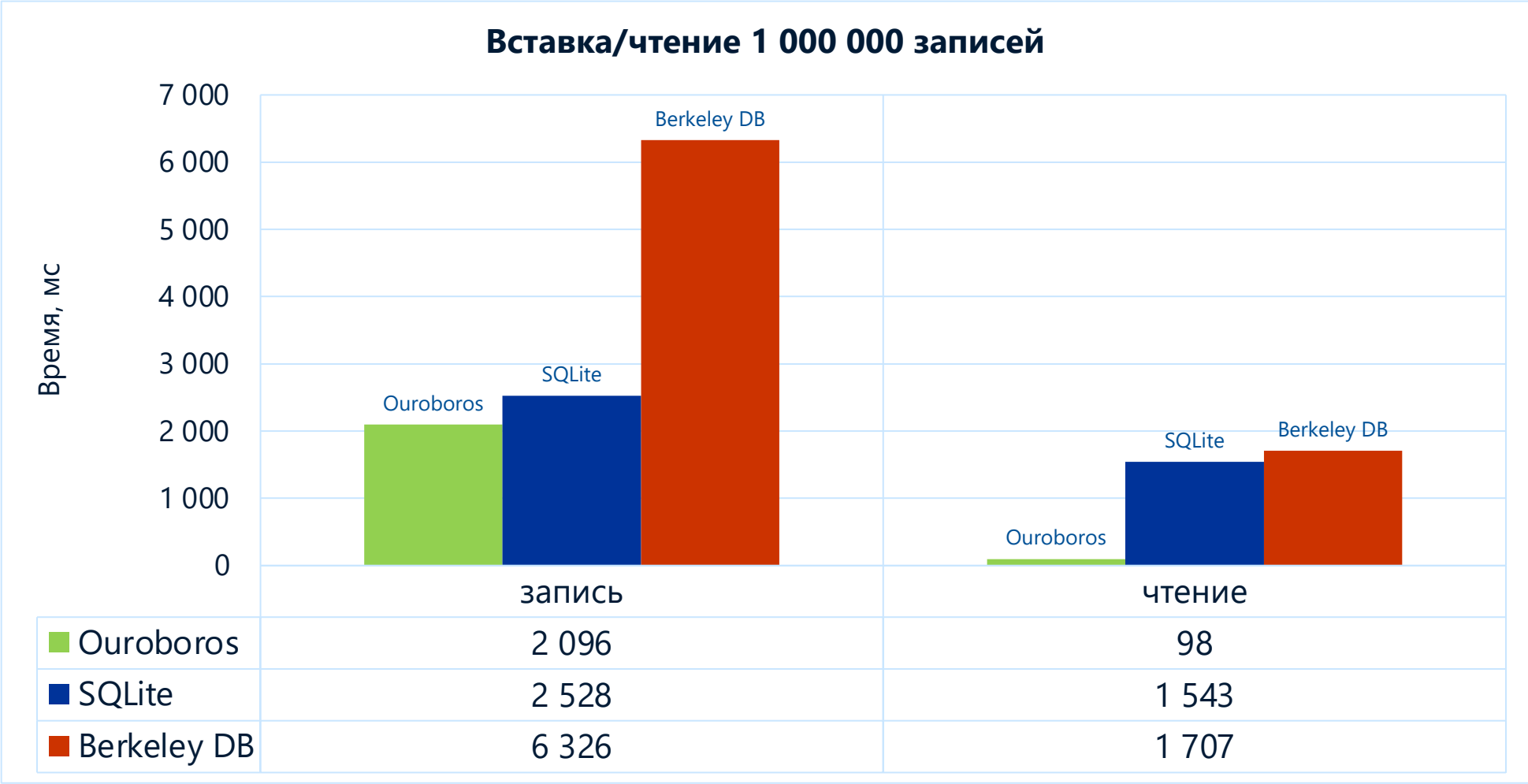
ИНДЕКСИРОВАННЫЕ ЗАПИСИ





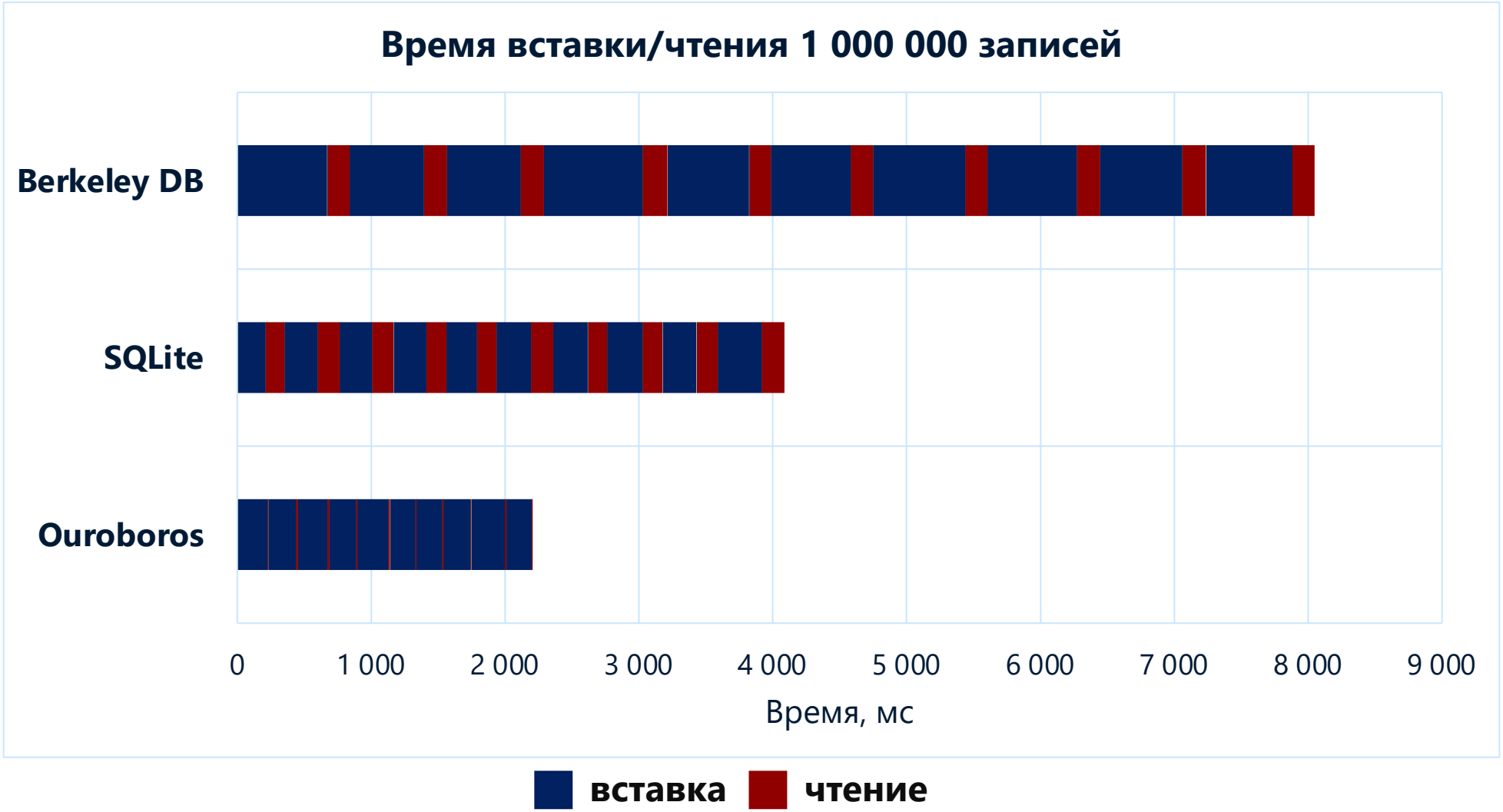


ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ

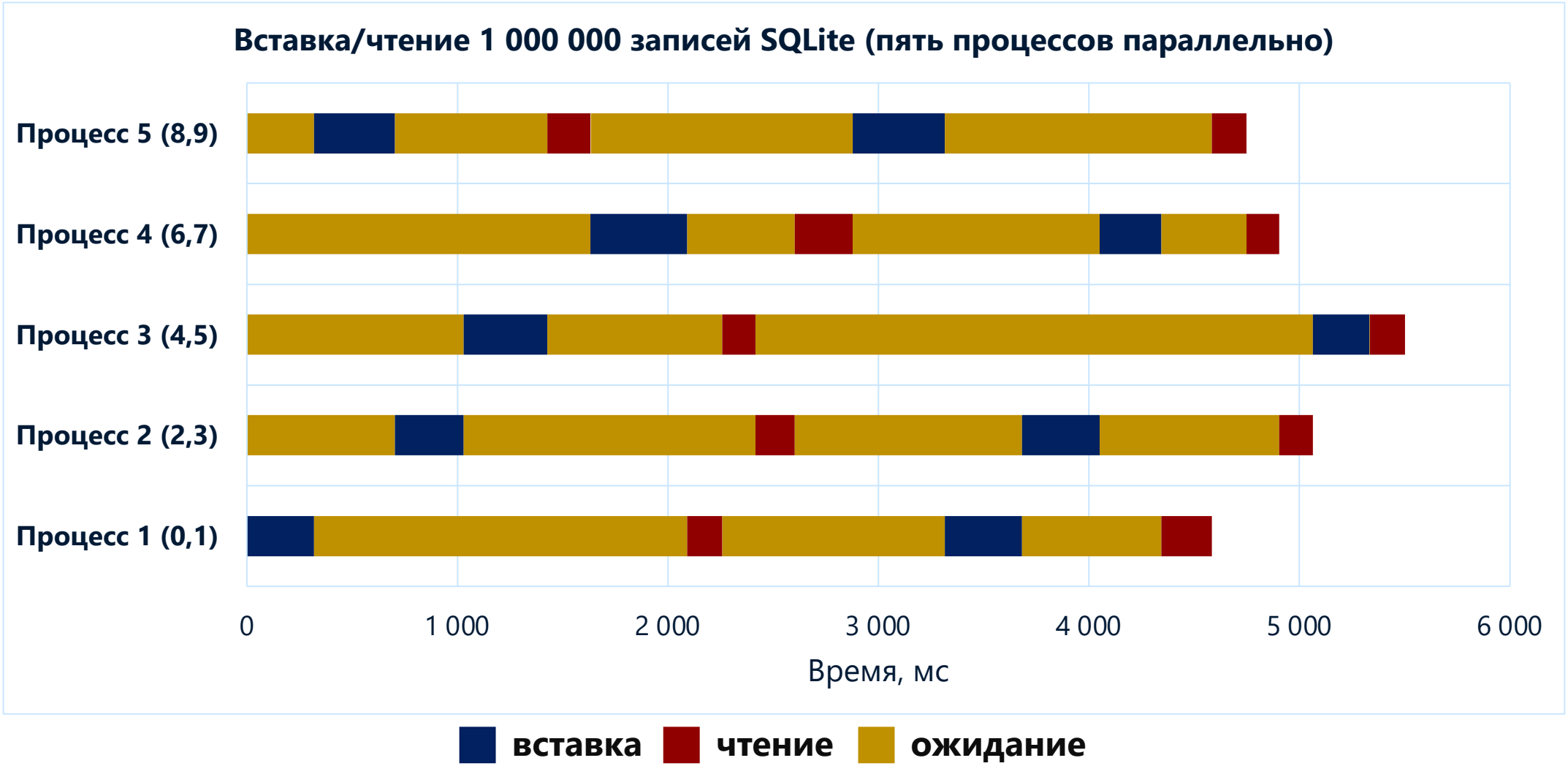


	запись	чтение
Ouroboros	2 096	98
SQLite	2 528	1 543
Berkeley DB	6 326	1 707

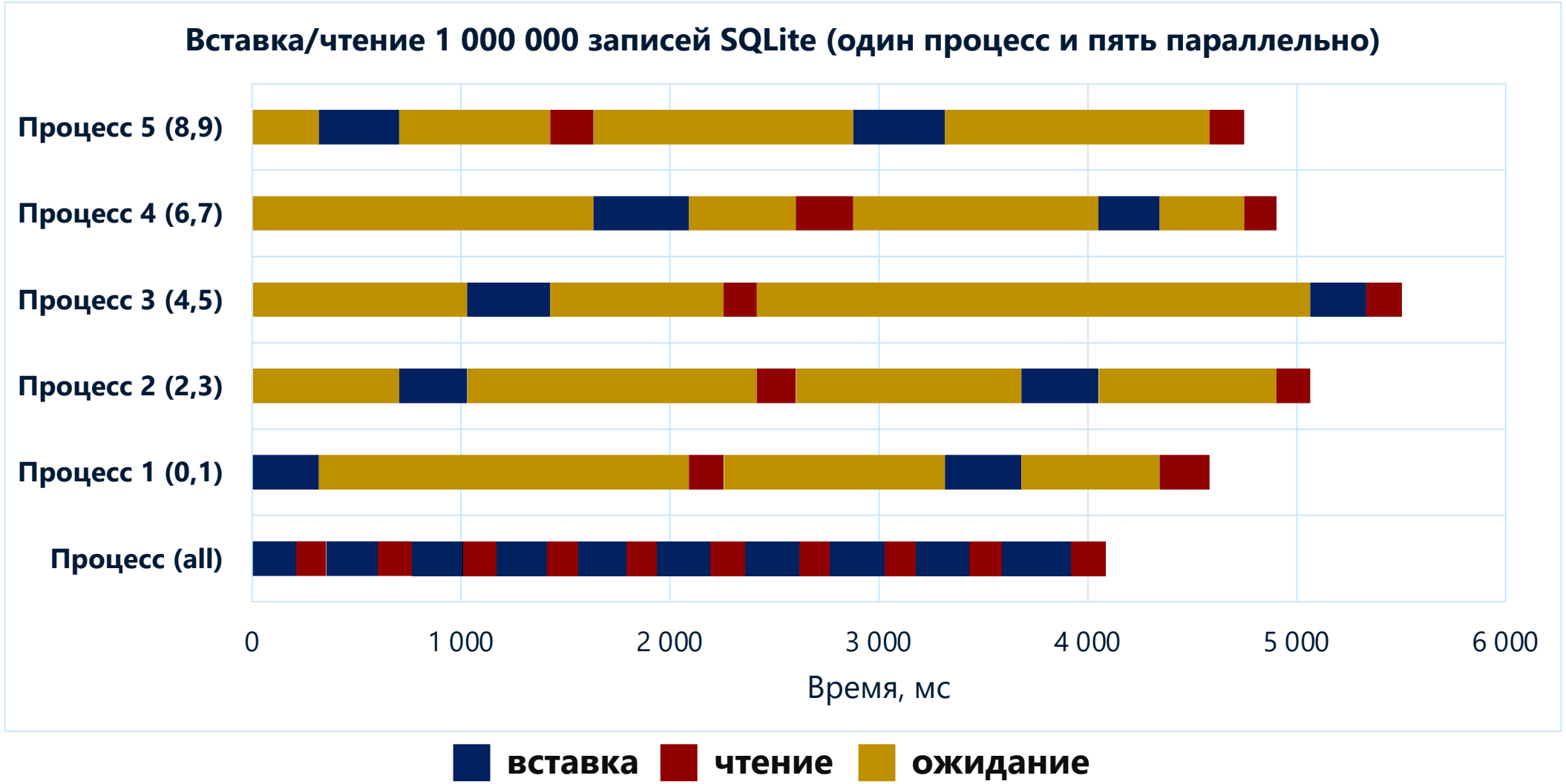
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ



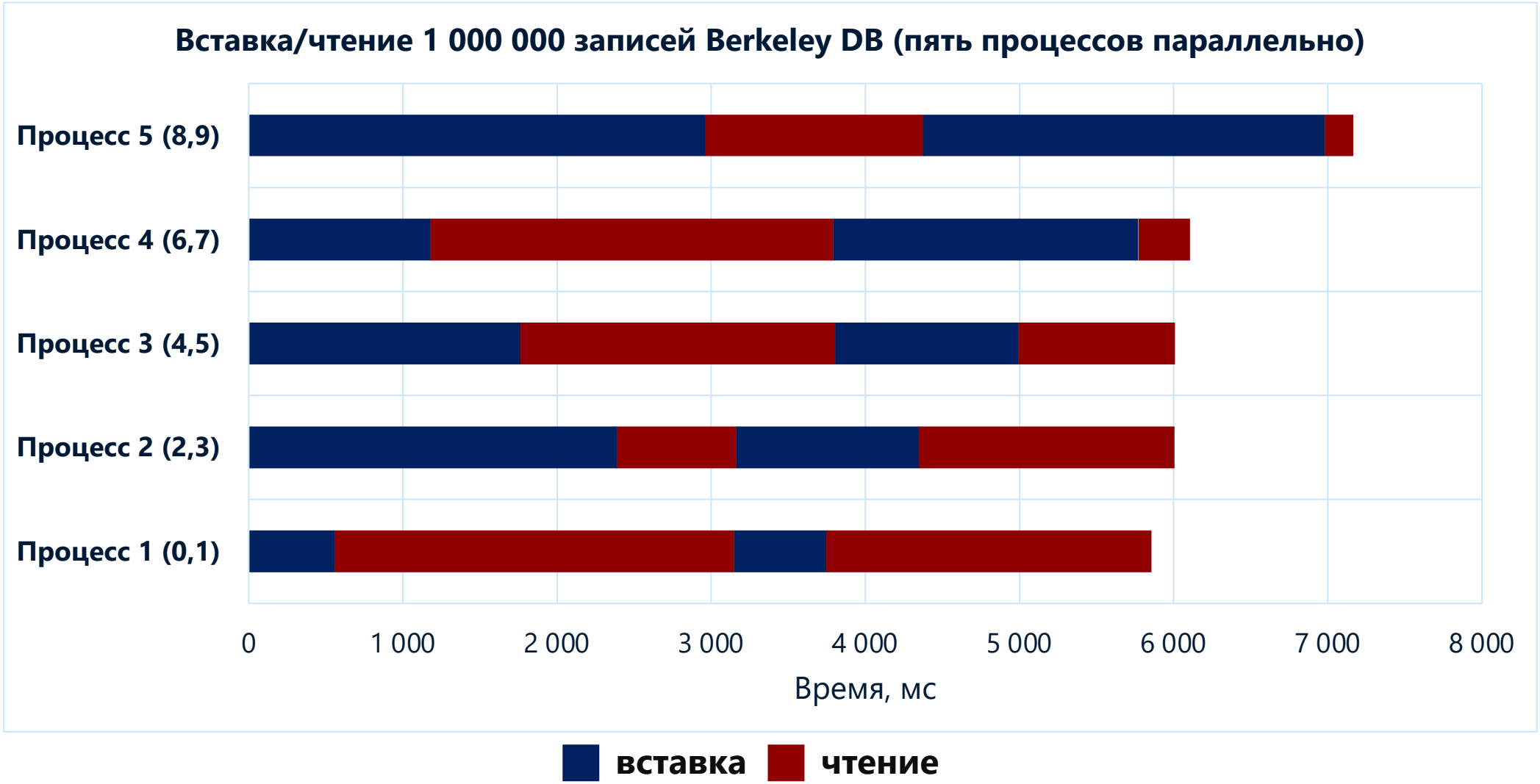
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ SQLITE



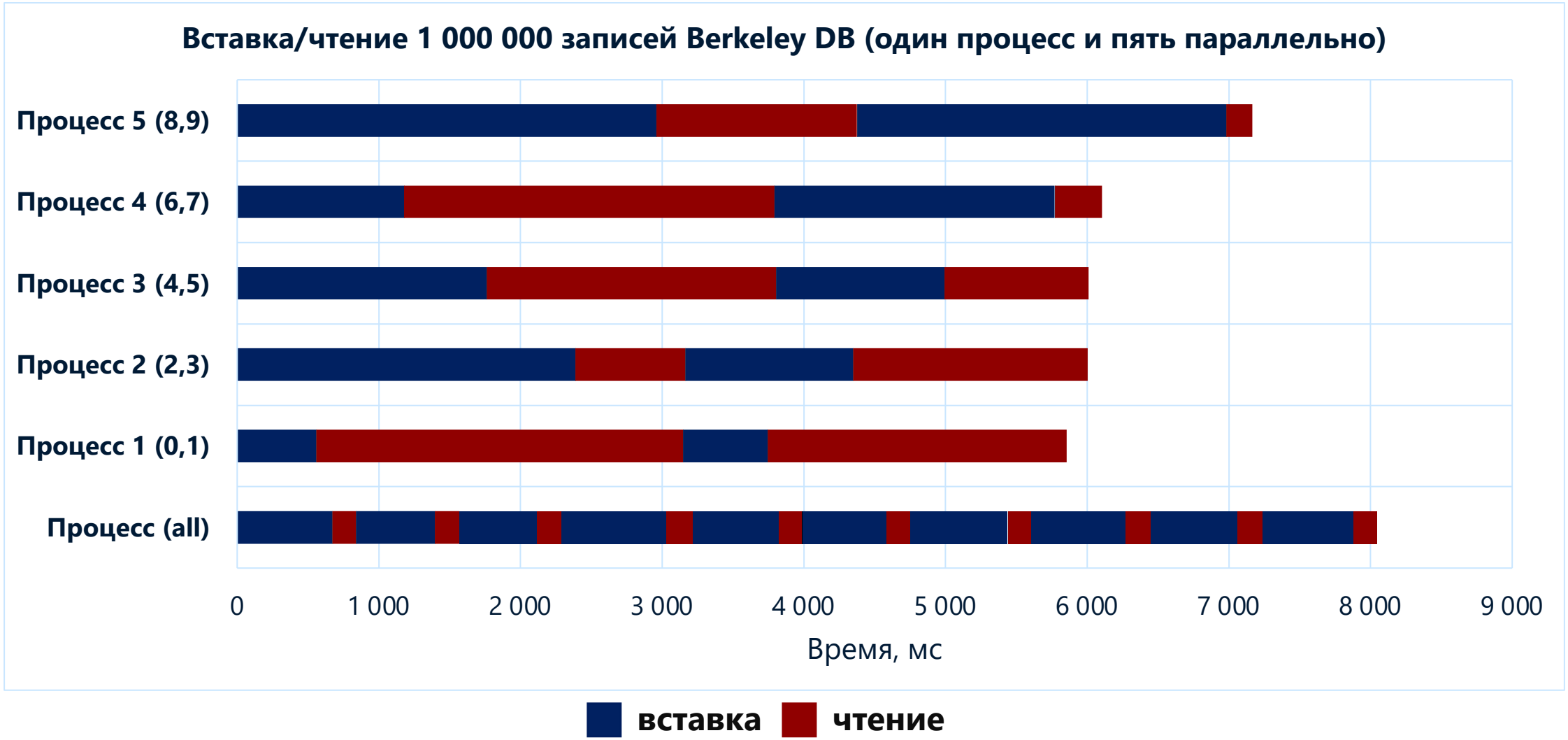
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ SQLITE



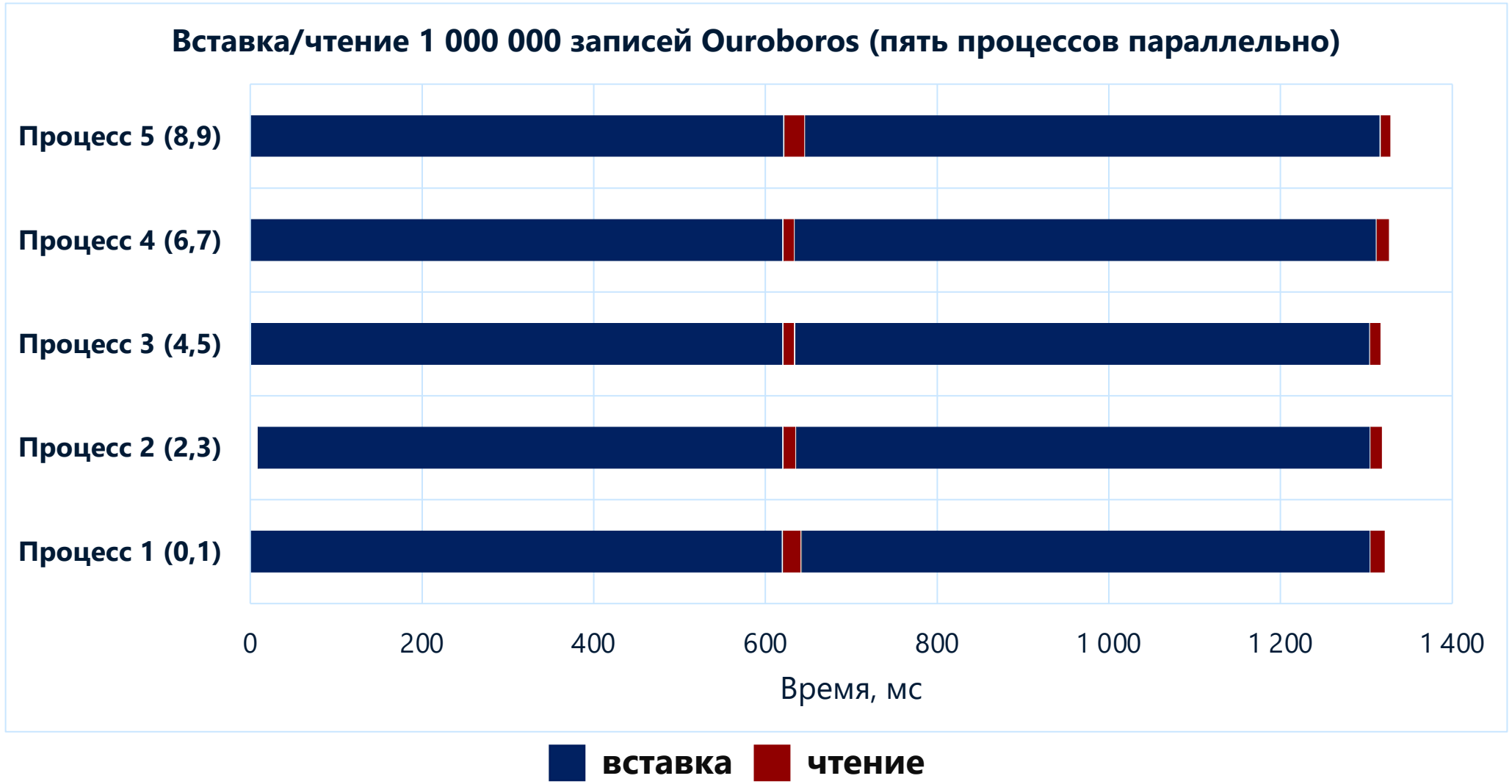
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ BERKELEY DB



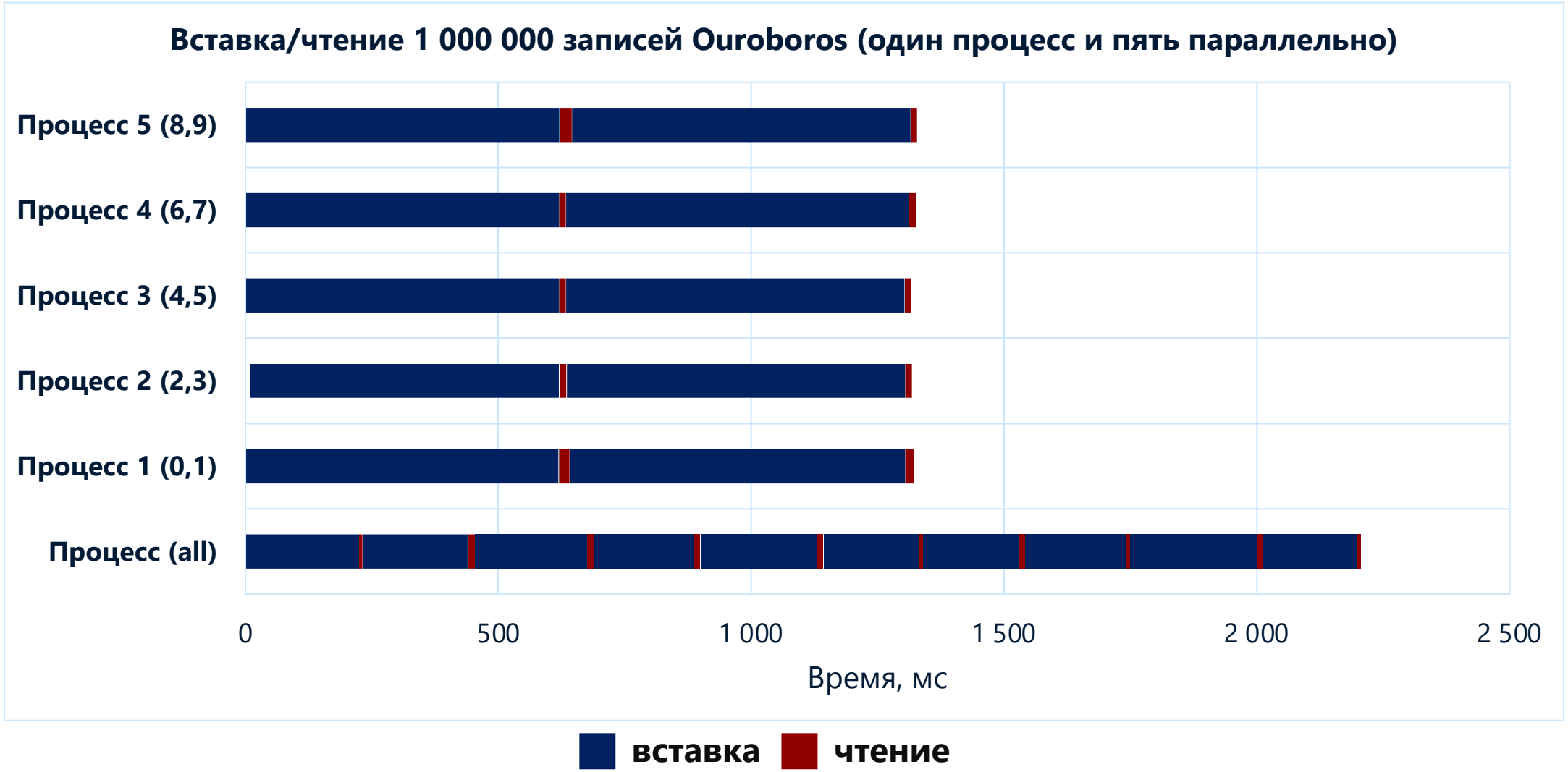
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ BERKELEY DB



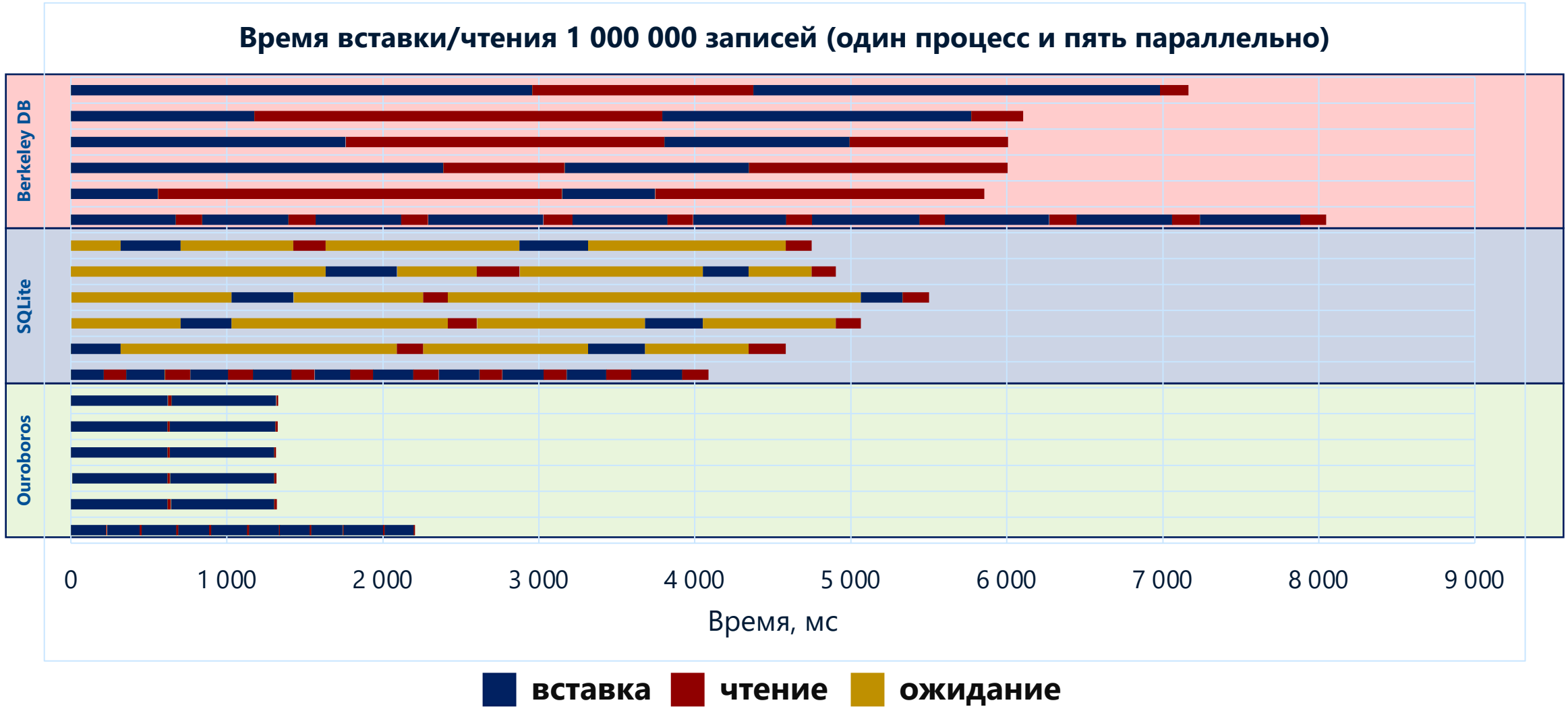
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ OUROBOROS



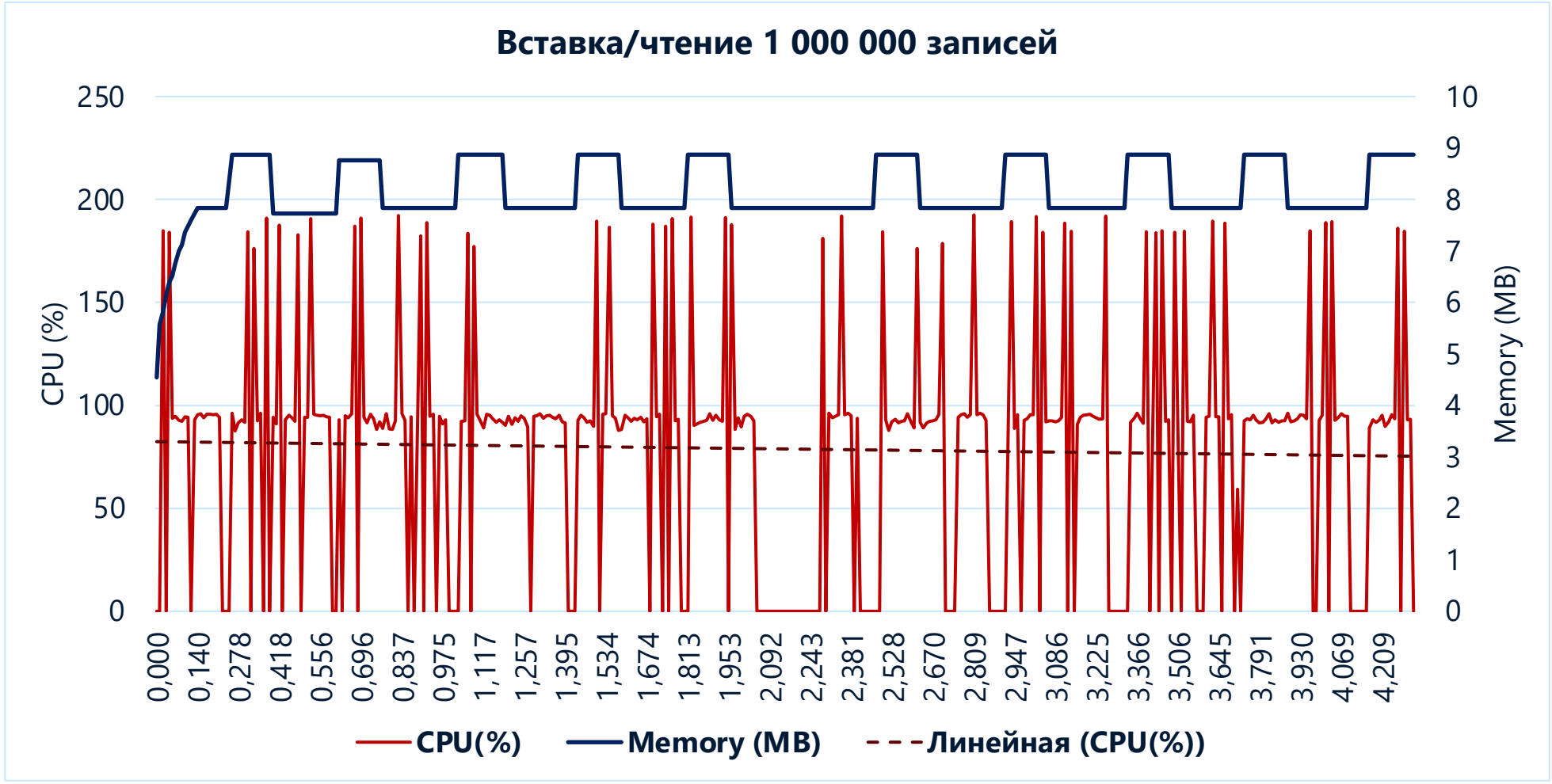
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ OUROBOROS



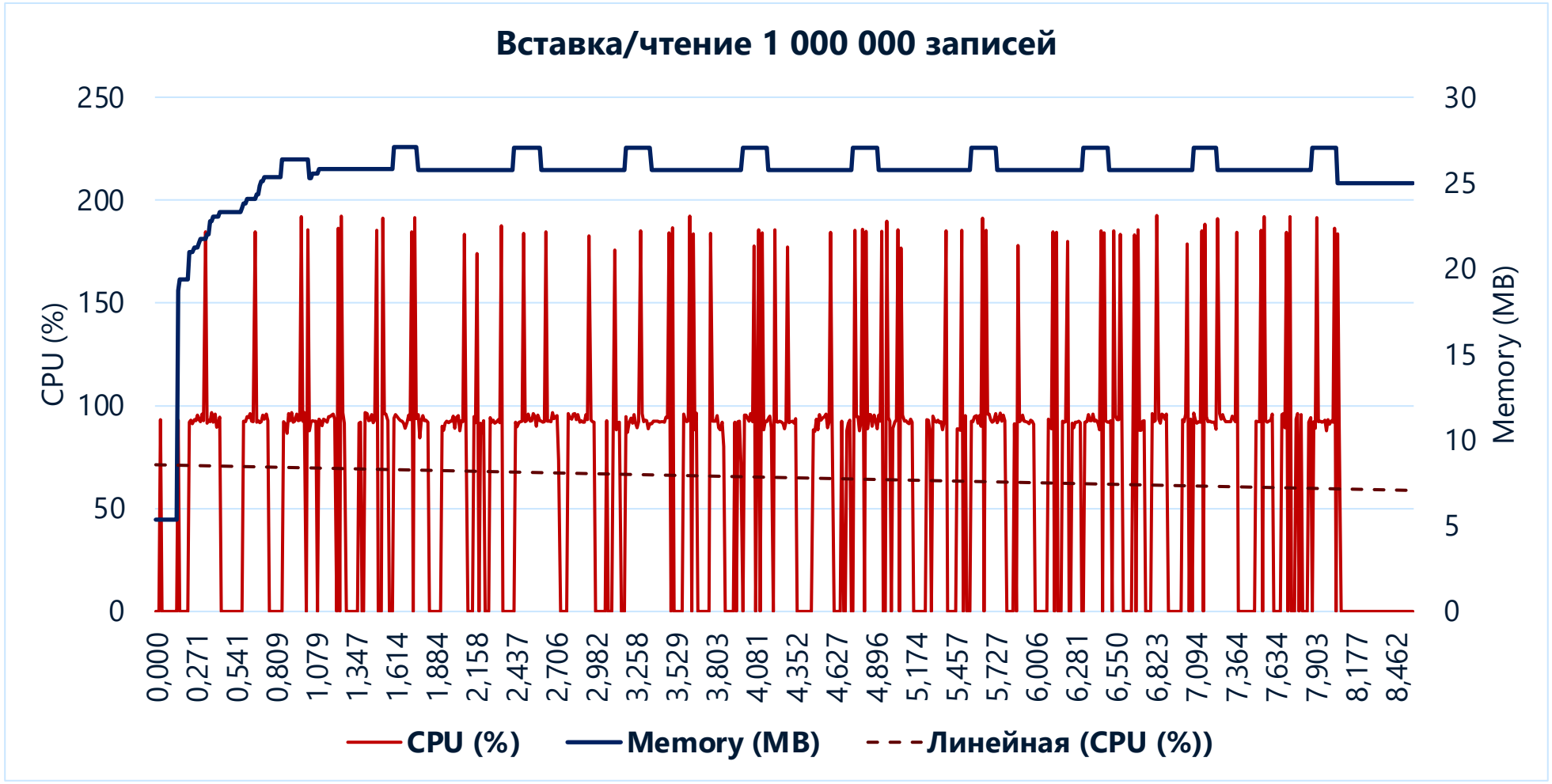
ВРЕМЯ ВСТАВКИ/ЧТЕНИЯ ЗАПИСЕЙ



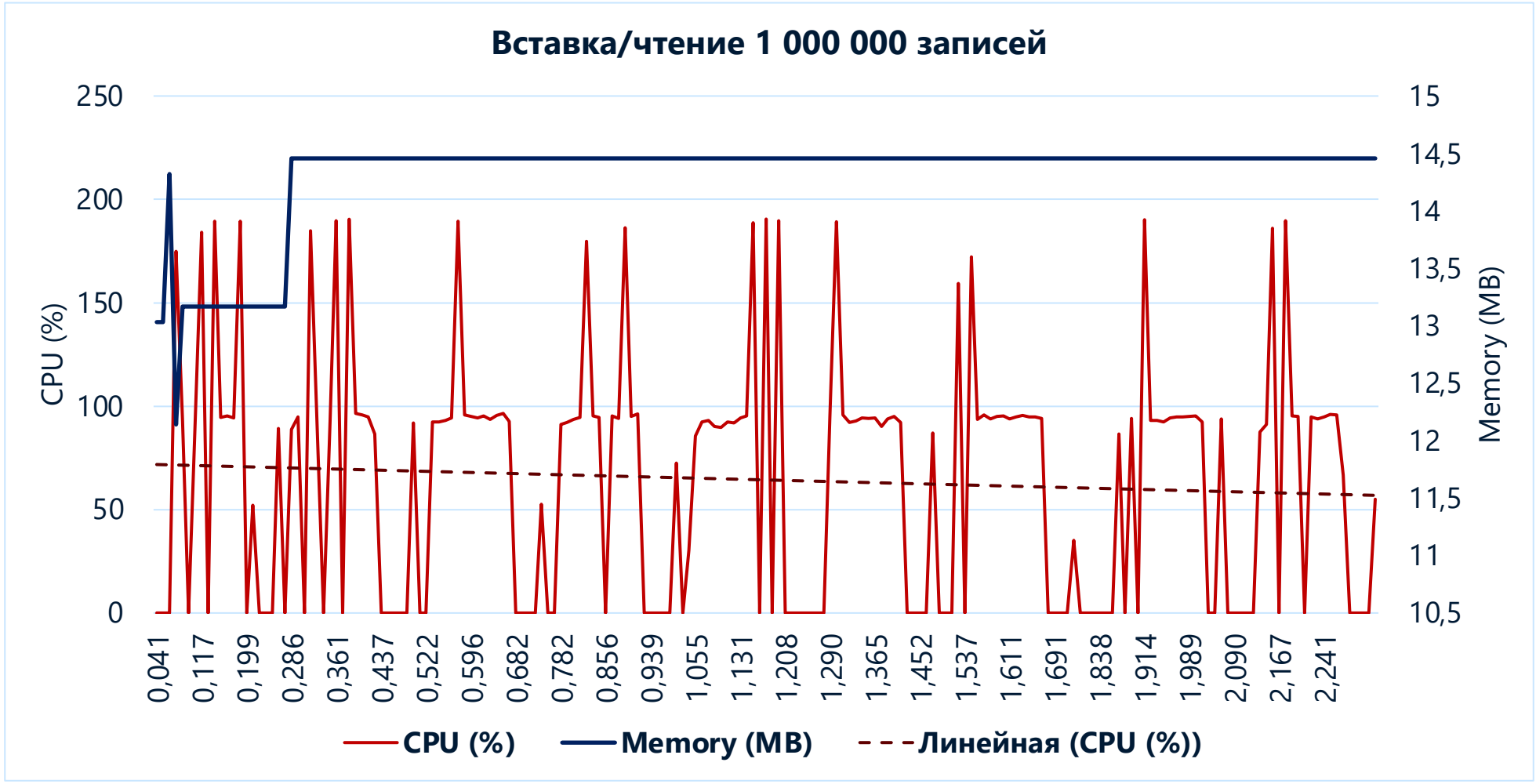
ПОТРЕБЛЕНИЕ ПРОЦЕССОРА И ПАМЯТИ SQLITE



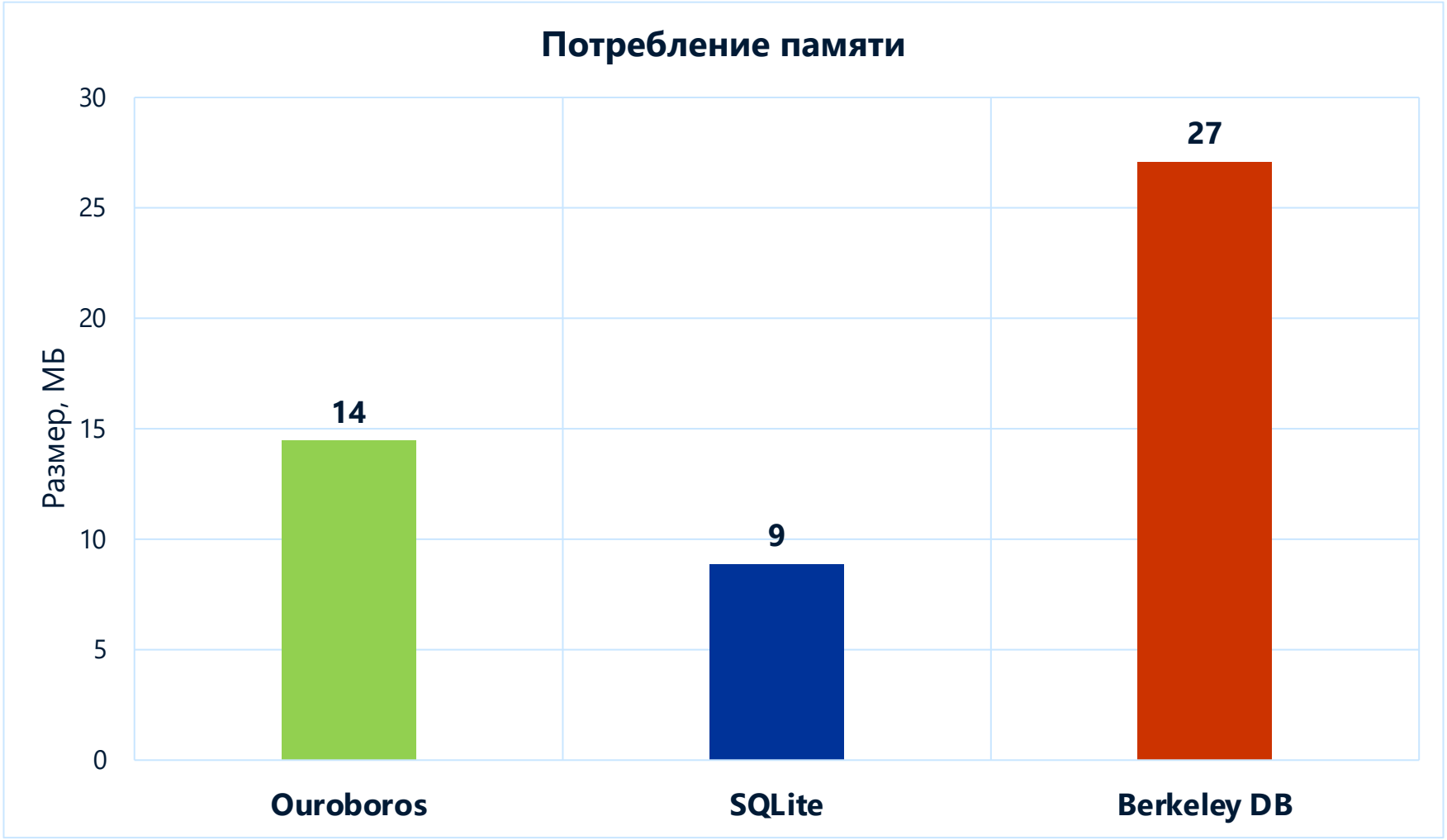
ПОТРЕБЛЕНИЕ ПРОЦЕССОРА И ПАМЯТИ BERKELEY DB



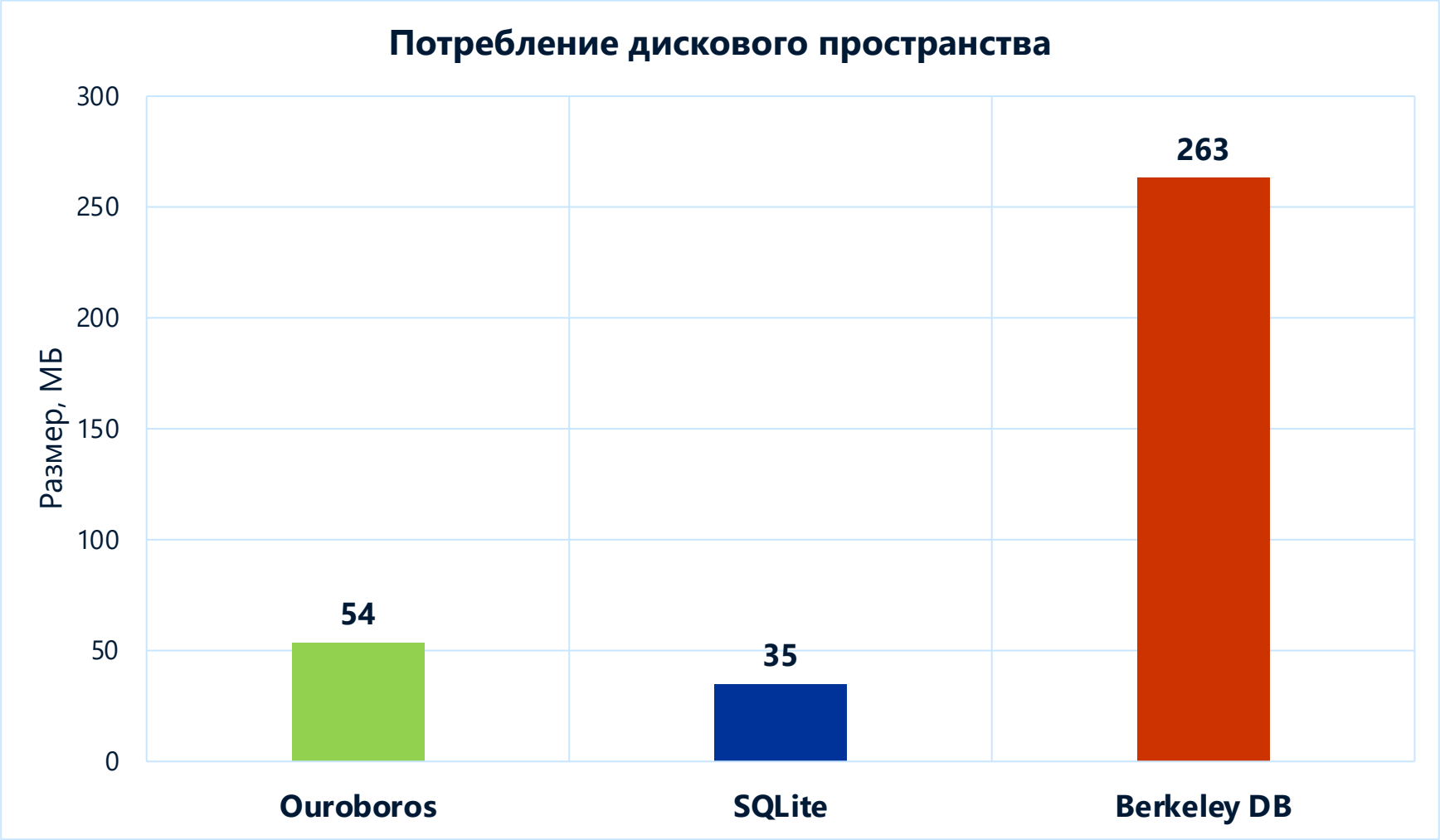
ПОТРЕБЛЕНИЕ ПРОЦЕССОРА И ПАМЯТИ OUROBOROS



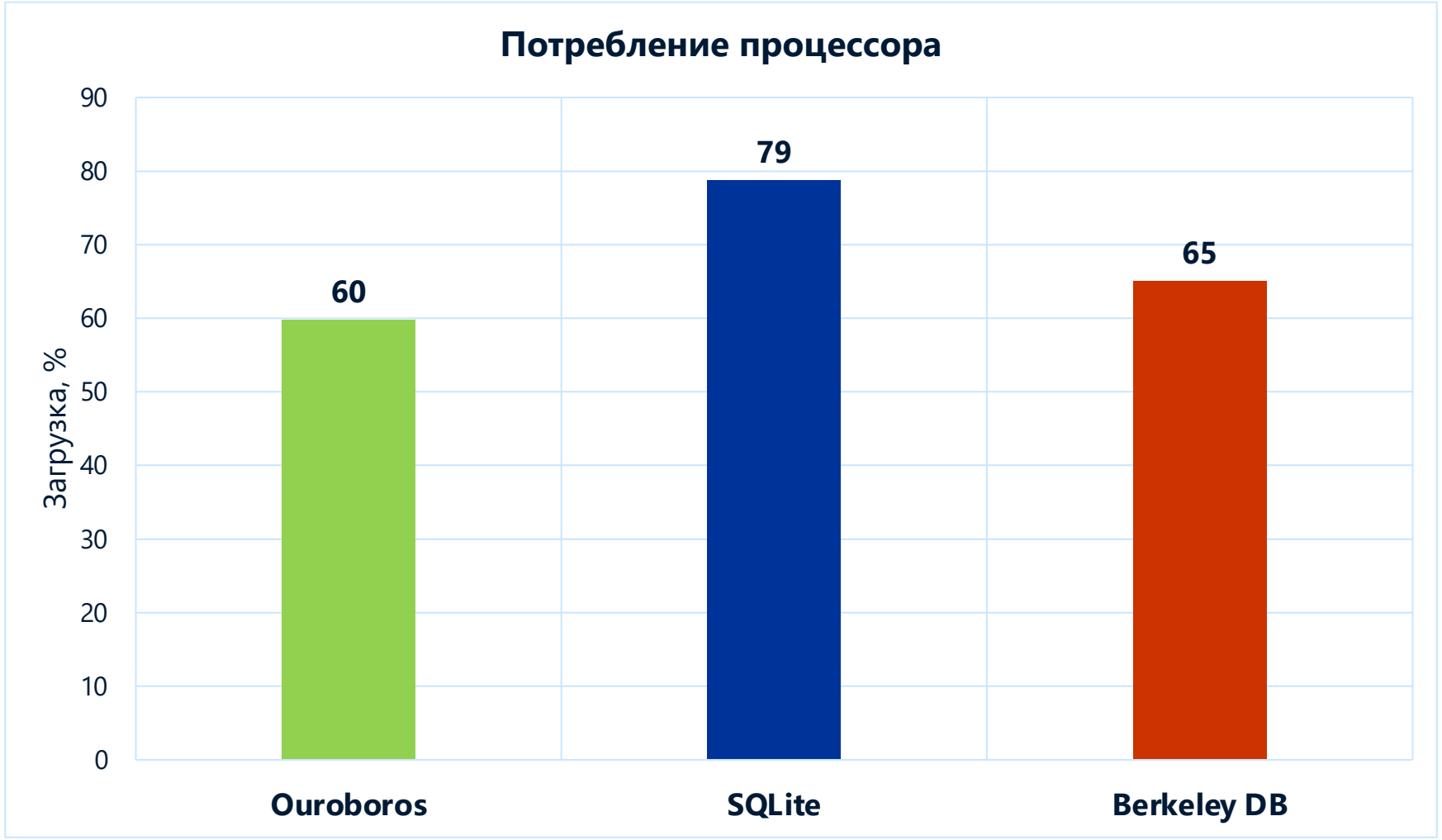
ПОТРЕБЛЕНИЕ РЕСУРСОВ



ПОТРЕБЛЕНИЕ РЕСУРСОВ



ПОТРЕБЛЕНИЕ РЕСУРСОВ



ПРИМЕР


```
#include "ouroboros/field_types.h"
#include "ouroboros/record.h"
#include "ouroboros/dataset.h"
#include "ouroboros/transaction.h"
#include "ouroboros/sharedinterface.h"

using namespace ouroboros;
typedef record3<FIELD_UINT32, FIELD_DOUBLE, FIELD_UINT16> record_type;
typedef data_set<simple_key, record_type, index_null, shared_interface> dataset_type;
typedef dataset_transaction<dataset_type> transaction_type;

int main(int argc, char *argv[])
{
    const std::string name = "db";
    const count_type tbl_count = 10;
    const count_type rec_count = 1000;

    dataset_type::key_type key = std::rand() % tbl_count;

    dataset_type dataset(name.c_str(), tbl_count, rec_count);
    {
        transaction_type transaction(dataset);
        if (!dataset.table_exists(key))
        {
           dataset.add_table(key);
        }
    }
    while (!is_terminated())
    {
        const count_type count = std::rand() % rec_count;
        dataset_type::record_list records(count);
        for (count_type i = 0; i < count; ++i)
```

ПРИМЕР


```
 1 #include "ouroboros/field_types.h"
 2 #include "ouroboros/record.h"
 3 #include "ouroboros/dataset.h"
 4 #include "ouroboros/transaction.h"
 5 #include "ouroboros/sharedinterface.h"
 6
 7 using namespace ouroboros;
 8 typedef record3<FIELD_UINT32, FIELD_DOUBLE, FIELD_UINT16> record_type;
 9 typedef data_set<simple_key, record_type, index_null, shared_interface> dataset_type;
10 typedef dataset_transaction<dataset_type> transaction_type;
11
12 int main(int argc, char *argv[])
13 {
14     const std::string name
15     const count_type tbl_c
16     const count_type rec_c
17
18     dataset_type::key_type
19
20     dataset_type dataset(n
21     {
22         transaction_type t
23         if (!dataset.table_exists(key))
24         {
25             dataset.add_table(key);
26         }
27     }
28     while (!is_terminated())
29     {
30         const count_type count = std::rand() % rec_count;
31         dataset_type::record_list records(count);
32         for (count_type i = 0; i < count; ++i)
```


```
1 #include "ouroboros/field_types.h"
2 #include "ouroboros/record.h"
3 #include "ouroboros/dataset.h"
4 #include "ouroboros/transaction.h"
5 #include "ouroboros/sharedinterface.h"
```

ПРИМЕР


```
 1 #include "ouroboros/field_types.h"
 2 #include "ouroboros/record.h"
 3 #include "ouroboros/dataset.h"
 4 #include "ouroboros/transaction.h"
 5 #include "ouroboros/sharedinterface.h"
 6
 7 using namespace ouroboros;
 8 typedef record3<FIELD_UINT32, FIELD_DOUBLE, FIELD_UINT16> record_type;
 9 typedef data_set<simple_key, record_type, index_null, shared_interface> dataset_type;
10 typedef dataset_transaction<dataset_type> transaction_type;
11
12 int main(int argc, char *argv[])
13 {
14     const std::string name = "db";
15     const count_type tbl_count = 10;
16     const count_type rec_count = 1000;
17
18     dataset_type::key_type key = std::rand() % tbl_count;
```


```
 7 using namespace ouroboros;
 8 typedef record3<FIELD_UINT32, FIELD_DOUBLE, FIELD_UINT16> record_type;
 9 typedef data_set<simple_key, record_type, index_null, shared_interface> dataset_type;
10 typedef dataset_transaction<dataset_type> transaction_type;
```


```
26         }
27     }
28     while (!is_terminated())
29     {
30         const count_type count = std::rand() % rec_count;
31         dataset_type::record_list records(count);
32         for (count_type i = 0; i < count; ++i)
```

ПРИМЕР

```
1  #include "ouroboros/field_types.h"
2  #include "ouroboros/record.h"
3  #include "ouroboros/dataset.h"
4  #include "ouroboros/transaction.h"
5  #include "ouroboros/sharedinterface.h"
```

```
20      dataset_type dataset(name.c_str(), tbl_count, rec_count);
21      {
22          transaction_type transaction(dataset);
23          if (!dataset.table_exists(key))
24          {
25              dataset.add_table(key);
26          }
27      }
```

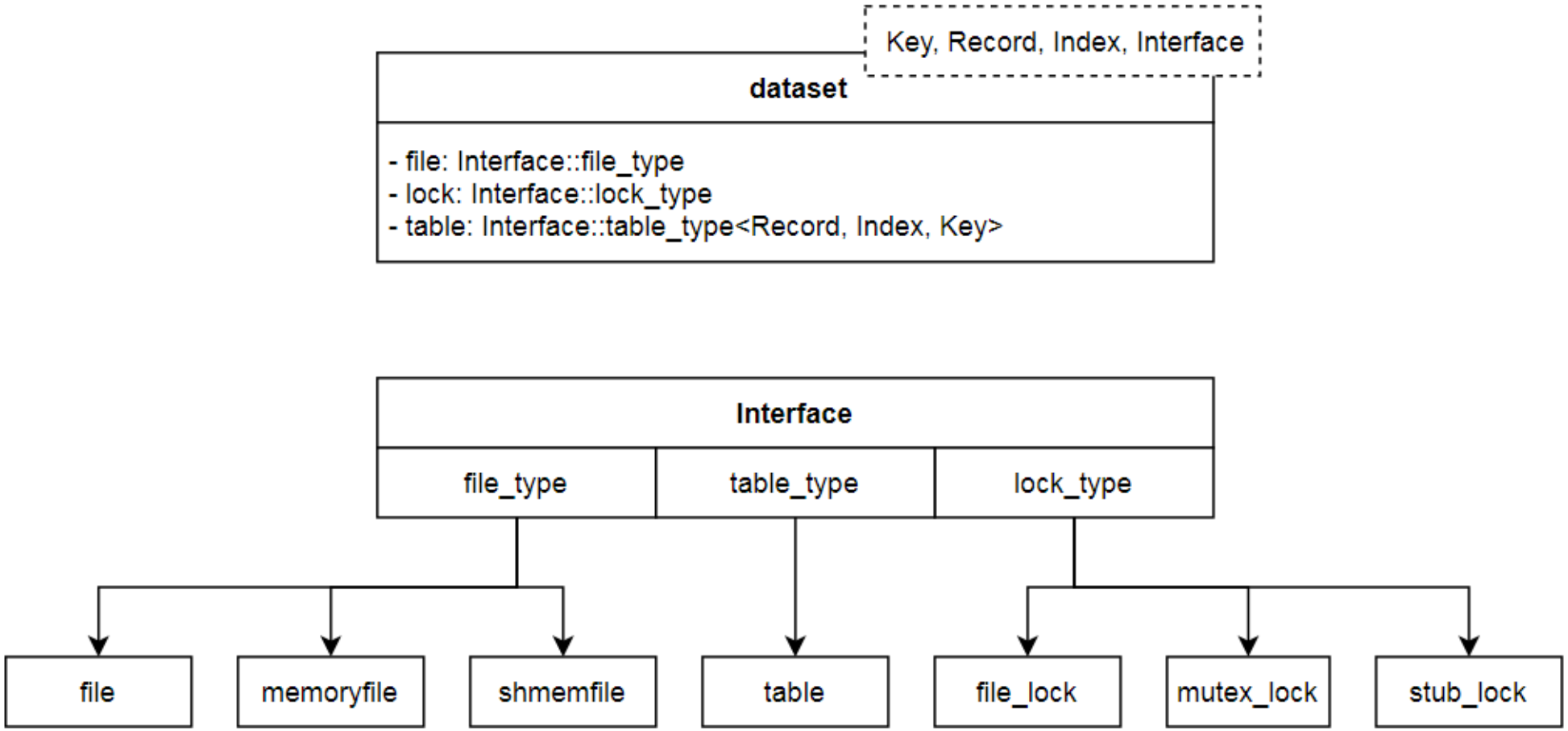
```
20      dataset_type dataset(name.c_str(), tbl_count, rec_count);
21      {
22          transaction_type transaction(dataset);
23          if (!dataset.table_exists(key))
24          {
25              dataset.add_table(key);
26          }
27      }
28      while (!is_terminated())
29      {
30          const count_type count = std::rand() % rec_count;
31          dataset_type::record_list records(count);
32          for (count_type i = 0; i < count; ++i)
```

ПРИМЕР


```
10 typedef dataset_transaction<dataset_type> transaction_type;
11
12 int main(int argc, char *argv[])
13 {
14     while (!is_terminated())
15     {
16         const count_type count = std::rand() % rec_count;
17         dataset_type::record_list records(count);
18         for (count_type i = 0; i < count; ++i)
19         {
20             records[i].field1(std::time(NULL));
21             records[i].field2(std::rand());
22             records[i].field3(std::rand());
23         }
24         dataset.session_wr(key)->add(records);
25     }
26     return 0;
27 }
```

```
31         dataset_type::record_list records(count);
32         for (count_type i = 0; i < count; ++i)
33         {
34             records[i].field1(std::time(NULL));
35             records[i].field2(std::rand());
36             records[i].field3(std::rand());
37         }
38         dataset.session_wr(key)->add(records);
```

АРХИТЕКТУРА



ГДЕ НАЙТИ

@belyaevms

https://github.com/belyaev-ms/ouroboros

ООО «Прософт-Системы»
Россия, г. Екатеринбург,
ул. Волгоградская, 194 а
Тел.: (343) 356-51-11

prosoftsystems.ru