

НАСТРОЙКА РЕПЛИКАЦИИ	2
ЧТЕНИЕ СО СЛЕЙВОВ	Ошибка! Закладка не определена.
ПОТЕРЯ МАСТЕРА ПРИ АСИНХРОННОЙ РЕПЛИКАЦИИ ...	Ошибка! Закладка не определена.
ПОТЕРЯ МАСТЕРА ПРИ СИНХРОННОЙ РЕПЛИКАЦИИ.....	Ошибка! Закладка не определена.
ВЫВОД	Ошибка! Закладка не определена.

НАСТРОЙКА ШАРДИРОВАНИЯ

В качестве БД – **Postgres 15.2**

Инструмент для шардирования - **Citus 11.2**

Будем делать master-ноду и несколько worker-нод для шардирования для этого используем менеджер **citusdata/membership-manager:0.3.0**

Документация: <https://github.com/citusdata/docker>

Ниже представлен фрагмент docker-compose с конфигурацией.

```
services:
  pg-0:
    image: citusdata/citus
    labels: ["com.citusdata.role=Master"]
    environment:
      &AUTH
      POSTGRES_DB: "chat-db"
      POSTGRES_USER: "someuser"
      POSTGRES_PASSWORD: "p@ssw0rd"
      PGUSER: "someuser"
      PGPASSWORD: "p@ssw0rd"
      POSTGRES_HOST_AUTH_METHOD: "trust"
      CITUS_HOST: "pg-0"
    ports:
      - "5432:5432"
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U someuser -d chat-db" ]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 10s

  pg-worker:
    image: citusdata/citus
    labels: ["com.citusdata.role=Worker"]
    environment: *AUTH
    command: "/wait-for-manager.sh"
    volumes:
      - healthcheck-volume:/healthcheck
    depends_on:
      pg-0:
        condition: service_healthy
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U someuser -d chat-db" ]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 10s

  shard-manager:
    image: citusdata/membership-manager:0.3.0
    volumes:
      - "${DOCKER_SOCKET:-/var/run/docker.sock}:/var/run/docker.sock"
      - healthcheck-volume:/healthcheck
    depends_on:
      pg-0:
```

```

        condition: service_healthy
environment: *AUTH

migrations-chat:
  build:
    context: ./chat
    dockerfile: ./migrate.Dockerfile
  environment:
    - POSTGRES_HOST=pg-0
    - POSTGRES_PORT=5432
    - POSTGRES_USER=someuser
    - POSTGRES_PASSWORD=p@ssw0rD
    - POSTGRES_DB=chat-db
  depends_on:
    pg-0:
      condition: service_healthy

chat:
  build:
    context: ./chat
    dockerfile: ./Dockerfile
  ports:
    - "5060:5060"
  environment:
    - POSTGRES_HOST=pg-0
    - POSTGRES_PORT=5432
    - POSTGRES_USER=someuser
    - POSTGRES_PASSWORD=p@ssw0rD
    - POSTGRES_DB=chat-db
    - LOG_LEVEL=debug
    - ADDR=5060
    - JWT_KEY=kek
  depends_on:
    migrations-chat:
      condition: service_completed_successfully

volumes:
  healthcheck-volume:

```

Запускать с несколькими нодами будем так:

`docker-compose -f docker-compose.chat.yaml up --scale pg-worker=5`

Проверим, что все 5 нод подключились

```
SELECT * FROM master_get_active_worker_nodes();
```

Output master_get_active_worker_nodes	
node_name	node_port
1 highload-architect-pg-worker-3	5432
2 highload-architect-pg-worker-1	5432
3 highload-architect-pg-worker-2	5432
4 highload-architect-pg-worker-4	5432
5 highload-architect-pg-worker-5	5432

СОЗДАНИЕ ТАБЛИЦЫ

Создадим таблицу для чатов.

```
CREATE TABLE IF NOT EXISTS chats
(
    id          UUID NOT NULL,
    sender_id   UUID                                NOT NULL,
    receiver_id UUID                                NOT NULL,
    text        text NOT NULL,
    created     TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT now(),
    PRIMARY KEY (id, sender_id)
);

CREATE INDEX IF NOT EXISTS chats_sender_receiver_created_idx ON chats
(sender_id, receiver_id, created DESC);
```

Основной запрос на чтение

```
SELECT
    c.id,
    c.sender_id,
    c.receiver_id,
    c.text,
    c.created
FROM chats c
WHERE c.sender_id=$1 AND c.receiver_id=$2
ORDER BY c.created DESC;
```

Поэтому создаем индекс chats_sender_receiver_created_idx

Специально делаем **PRIMARY KEY** (id, sender_id) что бы потом шардироваться по sender_id.

ШАРДИРУЕМ ТАБЛИЦУ

Добавляем шардирование

```
SELECT create_distributed_table('chats', 'sender_id');
```

Ключ шардирования – `sender_id`, так как запросы идут для получения сообщений в определенном чате (`sender_id`, `receiver_id`). Но потенциально могут понадобиться запросы для получения всех чатов для Юзера(`sender_id`). Что бы не ходить по разным шардам, будем хранить все чаты Юзера на одном шарде.

Заполнение данных:

```
INSERT INTO chats (id, sender_id, receiver_id, text)
SELECT gen_random_uuid(), gen_random_uuid(), gen_random_uuid(),
gen_random_uuid()
FROM generate_series(1, 1000000);
```

Ребалансировка шардов

```
SELECT rebalance_table_shards();
```

Проверим, что все работает

```
EXPLAIN ANALYZE SELECT
  c.id,
  c.sender_id,
  c.receiver_id,
  c.text,
  c.created
FROM chats c
WHERE c.sender_id='f67aeab1-a30b-47ee-ad9e-99bcb3261149' AND
      c.receiver_id='7d6cfb57-3c48-4689-aae0-12669ea8648a'
ORDER BY c.created DESC;
```

```
Custom Scan (Citus Adaptive)      (cost=0.00..0.00 rows=0 width=0) (actual
time=3.448..3.475 rows=1 loops=1)
```

```
Task Count: 1
```

```
Tuple data received from nodes: 92 bytes
```

```
Tasks Shown: All
```

```
-> Task
```

```
    Tuple data received from node: 92 bytes
```

```
    Node: host=highload-architect-pg-worker-1 port=5432 dbname=chat-db
```

-> Index Scan using chats_sender_receiver_created_idx_102008 on chats_102008 c (cost=0.41..8.43 rows=1 width=93) (actual time=0.044..0.079 rows=1 loops=1)

Index Cond: ((sender_id = 'f67aeab1-a30b-47ee-ad9e-99bcb3261149'::uuid) AND (receiver_id = '7d6cfb57-3c48-4689-aae0-12669ea8648a'::uuid))

Planning Time: 0.275 ms

Execution Time: 0.170 ms

Planning Time: 0.241 ms

Execution Time: 3.563 ms

`EXPLAIN ANALYZE SELECT count(*) FROM chats;`

Aggregate (cost=250.00..250.02 rows=1 width=8) (actual time=6812.831..6812.874 rows=1 loops=1)

-> Custom Scan (Citrus Adaptive) (cost=0.00..0.00 rows=100000 width=8) (actual time=6812.069..6812.441 rows=32 loops=1)

Task Count: 32

Tuple data received from nodes: 256 bytes

Tasks Shown: One of 32

-> Task

Tuple data received from node: 8 bytes

Node: host=highload-architect-pg-worker-2 port=5432 dbname=chat-db

-> Aggregate (cost=864.21..864.22 rows=1 width=8) (actual time=6630.886..6631.305 rows=1 loops=1)

-> Seq Scan on chats_102014 chats (cost=0.00..786.77 rows=30977 width=0) (actual time=0.038..3235.543 rows=30977 loops=1)

Planning Time: 1.354 ms

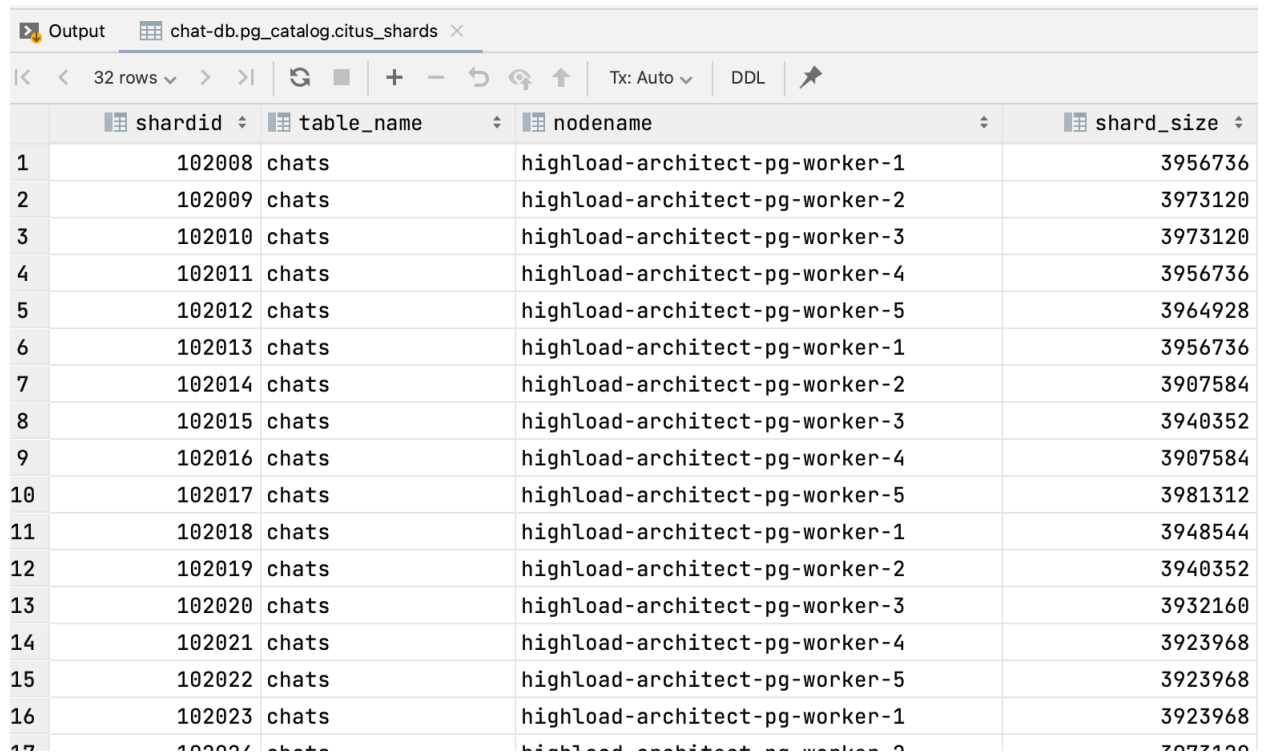
Execution Time: 6636.647 ms

Planning Time: 0.468 ms

Execution Time: 6812.955 ms

ИТОГ

```
SELECT cs.shardid, cs.table_name, cs.nodename, cs.shard_size
FROM citus_shards cs;
```



The screenshot shows a database query output window with the title 'chat-db.pg_catalog.citus_shards'. The output displays 32 rows of data. The columns are 'shardid', 'table_name', 'nodename', and 'shard_size'. The data shows that the 'chats' table is distributed across 5 nodes, with each node having 8 shards. The shard sizes vary slightly between nodes.

	shardid	table_name	nodename	shard_size
1	102008	chats	highload-architect-pg-worker-1	3956736
2	102009	chats	highload-architect-pg-worker-2	3973120
3	102010	chats	highload-architect-pg-worker-3	3973120
4	102011	chats	highload-architect-pg-worker-4	3956736
5	102012	chats	highload-architect-pg-worker-5	3964928
6	102013	chats	highload-architect-pg-worker-1	3956736
7	102014	chats	highload-architect-pg-worker-2	3907584
8	102015	chats	highload-architect-pg-worker-3	3940352
9	102016	chats	highload-architect-pg-worker-4	3907584
10	102017	chats	highload-architect-pg-worker-5	3981312
11	102018	chats	highload-architect-pg-worker-1	3948544
12	102019	chats	highload-architect-pg-worker-2	3940352
13	102020	chats	highload-architect-pg-worker-3	3932160
14	102021	chats	highload-architect-pg-worker-4	3923968
15	102022	chats	highload-architect-pg-worker-5	3923968
16	102023	chats	highload-architect-pg-worker-1	3923968
17	102024	chats	highload-architect-pg-worker-2	3923120

Итого мы сделали 5 распределенных нод, на которых созданы 32 части таблицы.

Это дефолтное поведение для `SELECT create_distributed_table(...);`

citus.shard_count (integer)

Sets the shard count for hash-partitioned tables and defaults to **32**. This value is used by the `create_distributed_table` UDF when creating hash-partitioned tables. This parameter can be set at run-time and is effective on the coordinator.

citus.shard_max_size (integer)

Sets the maximum size to which a shard will grow before it gets split and defaults to 1GB. When the source file's size (which is used for staging) for one shard exceeds this configuration value, the database ensures that a new shard gets created. This parameter can be set at run-time and is effective on the coordinator.

Решардинг работает из коробки

```
SELECT rebalance_table_shards();
```