ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет Информационных технологий и программной инженерии
Кафедра Систем обработки данных

ОТЧЕТ

по практическому занятию №5

Тема: «Репликация и шардирование»

Выполнил: студент 3 курса, гр. ИБ-31вп
_____ Ворожцов А. Е.

Принял:
_____ Медведев С. А.

Санкт-Петербург, 2025 г.

Содержание

Репликация PostgreSQL: первичная и резервная инстанции

Конфигурация_primary

Сервис `postgres-primary`

- Собирается из `primary/Dockerfile`, где в образ копируются `init-primary.sh` и `02-CANTEENDISHES.sql`.
- Переменные окружения (`docker-compose.yml`) задают базу `canteen` и пользователя `canteen`.
- Скрипт `primary/init-primary.sh` выполняется один раз при инициализации и отвечает за конфигурацию:

```
-- создание пользователя для репликации
CREATE ROLE repl_user WITH REPLICATION LOGIN PASSWORD 'repl_password';

-- настройка параметров через ALTER SYSTEM
ALTER SYSTEM SET listen_addresses = '*';
ALTER SYSTEM SET wal_level = 'replica';
ALTER SYSTEM SET max_wal_senders = 10;
ALTER SYSTEM SET max_replication_slots = 10;
ALTER SYSTEM SET wal_keep_size = '1GB';
ALTER SYSTEM SET archive_mode = 'on';
ALTER SYSTEM SET archive_command = 'test ! -f /var/lib/postgresql/archive/%f && cp %p /var/lib

-- правила pg_hba.conf (по умолчанию допускаем сети 172.16.0.0/12 и 192.168.0.0/16)
host replication repl_user <subnet> scram-sha-256
host all all <subnet> scram-sha-256
host all all 127.0.0.1/32 scram-sha-256
```

Сервис `postgres-replica`

- Собирается из `replica/Dockerfile`, где точкой входа служит `replica-entrypoint.sh`.
- Скрипт реплики выполняет следующую последовательность:
  1. Ждёт готовности primary (`pg_isready`).
  2. Полностью очищает `PGDATA` и делает `pg_basebackup` с ключами `-Fp -Xs -P -R -C -S replica_slot`.
  3. Запускает стандартный entrypoint с параметрами `listen_addresses=*` и `hot_standby=on`.
- Окружение (`docker-compose.yml`) передаёт координаты primary, логин/пароль `repl_user` и имя слота `replica_slot`.

Проверка `make run`

Вывод команды `make run` фиксирует состояние репликации и демонстрирует распространение изменений на строке `canteendishes.dishid=1`.

1. Состояние потоков на первичном сервере

Запрос:

```
SELECT pid, application_name, state, sync_state, write_lsn, flush_lsn, replay_lsn
FROM pg_stat_replication;
```

Вывод:

```
 pid | application_name |   state   | sync_state | write_lsn | flush_lsn | replay_lsn
-----+------------------+-----------+------------+-----------+-----------+-----------
--
 252 | walreceiver      | streaming | async      | 0/308C450 | 0/308C450 | 0/308C450
(1 row)
```

2. Статус реплики

Запрос:

```
SELECT pg_is_in_recovery();
```

Вывод:

```
 pg_is_in_recovery
-------------------
 t
(1 row)
```

3. Проверка согласованности данных

Запрос (для чтения):

```
SELECT price::text FROM canteendishes WHERE dishid=1;
```

Вывод до обновления:

```
Первичный сервер до обновления : 23.00
Сервер-реплика до обновления : 23.00
Значения совпадают до обновления.
```

Запрос (обновление на primary):

```
UPDATE canteendishes SET price = price + 1 WHERE dishid=1 RETURNING price::text;
```

Результат обновления:

```
Новое значение на первичном сервере: 24.00
Изменено строк: 1
```

Состояние после задержки 2 секунды (тот же SELECT):

```
Первичный сервер после обновления : 24.00
Сервер-реплика после обновления : 24.00
Репликация подтверждена для выбранной строки.
```

Таким образом, репликация активна, реплика находится в режиме `pg_is_in_recovery() = true`, а данные успешно распространяются на резервный сервер.

Шардирование на Citus

В каталоге `citus_cluster/` описан отдельный кластер на базе `citusdata/citus:13.2.0` (PostgreSQL 17) с пятью сервисами.

- `citus-coordinator` — узел-координатор (порт `5440`), куда отправляются все клиентские запросы на чтение/запись.
- `citus-worker1` и `citus-worker2` — рабочие примари. Каждый имеет streaming-реплику (`citus-worker1-replica`, `citus-worker2-replica`), которая синхронизируется через стандартный WAL.

Конфигурация рабочих серверов

`worker-primary/init-worker.sh` включает расширение Citus, создаёт роль `repl_user` и выполняет настройки:

```
ALTER SYSTEM SET listen_addresses = '*';
ALTER SYSTEM SET wal_level = 'replica';
ALTER SYSTEM SET max_wal_senders = 10;
ALTER SYSTEM SET max_replication_slots = 10;
ALTER SYSTEM SET wal_keep_size = '1GB';
ALTER SYSTEM SET archive_mode = 'on';
ALTER SYSTEM SET archive_command = 'test ! -f /var/lib/postgresql/archive/%f && cp %p /var/lib
ALTER SYSTEM SET shared_preload_libraries = 'citus';
```

`pg_hba.conf` автоматически получает записи `host replication repl_user … scram-sha-256` и `host all all … scram-sha-256` для указанных подсетей и `127.0.0.1/32`.

Настройка координатора

`coordinator/01-init-coordinator.sh` ожидает готовности рабочих узлов и выполняет:

```
CREATE EXTENSION IF NOT EXISTS citus;
SELECT master_add_node('citus-worker1', 5432) ...;
SELECT master_add_node('citus-worker2', 5432) ...;
```

Далее миграции идут в несколько шагов:

1. `data/01-create-personcanteenorder.sql` — создаёт таблицу `personcanteenorder(personid integer, dateorder date, dishid integer)`.
2. `coordinator/03-distribute-personcanteenorder.sql` — задаёт `citus.shard_count = 2`, `citus.shard_replication_factor = 1` и выполняет `create_distributed_table('personcanteenorder` (стандартное хэш-шардирование).
3. `coordinator/04-place-shards.sql` — перебирает созданные шарды и при необходимости вызывает `master_move_shard_placement`, чтобы первый оказался на `citus-worker1`, второй — на `citus-worker2`.
4. `data/02-insert-personcanteenorder.sql` — загружает исходный набор данных, после чего Citus сразу расставляет строки по воркерам.

Такой подход даёт «коробочное» поведение Citus: таблица распределяется по хэшу `personid`, а вспомогательный скрипт просто раскладывает два шарда по воркерам, не требуя дополнительных

столбцов или триггеров.

Как обращаться к кластеру

1. Собрать и поднять кластер: `docker compose -f citus_cluster/docker-compose.yml up -d --build` (последовательность запускать не требуется — compose создаст сеть и поднимет узлы в нужном порядке).
2. Все SQL-запросы направляйте в координатор, например `psql -h localhost -p 5440 -U postgres canteen`. Citus автоматически раскидает записи по шардам.
3. Проверить раскладку можно запросами:

```
SELECT shardid, nodename
FROM pg_dist_shard_placement
WHERE shardid IN (SELECT shardid FROM pg_dist_shard WHERE logicalrelid='personcanteenorde
```

4. Для просмотра распределения используйте системные представления Citus, например `SELECT * FROM pg_dist_shard_placement WHERE logicalrelid='personcanteenorder'::regclass;` — видно, какой shard обслуживает каждый воркер.

Клиентские операции записи выполняются только через координатор (`localhost:5440`). Рабочие реплики используются Postgres'ом для отказоустойчивости, но в Citus регистрируются только примари; при необходимости переключения придётся вручную обновить записи `master_add_node`.

Для автоматизированной проверки предусмотрен таргет `make citus` (см. `scripts/check-citus.sh`), который выполняет последовательность запросов:

1. Ждёт появления таблицы (`SELECT to_regclass('personcanteenorder');`).
2. Показывает расположение шардов:

```
SELECT sp.shardid, sp.nodename, sp.nodeport, s.shardminvalue, s.shardmaxvalue
FROM pg_dist_shard_placement sp
JOIN pg_dist_shard s USING (shardid)
WHERE s.logicalrelid='personcanteenorder'::regclass
ORDER BY sp.shardid, sp.nodename;
```

3. Вставляет две строки с уникальными `personid`:

```
INSERT INTO personcanteenorder(personid, dateorder, dishid)
VALUES (<ts>, CURRENT_DATE, 1);
INSERT INTO personcanteenorder(personid, dateorder, dishid)
VALUES (<ts+1>, CURRENT_DATE, 2);
```

4. Получает идентификаторы шардов через `SELECT get_shard_id_for_distribution_column('personcan…);` и определяет воркеры (`SELECT nodename FROM pg_dist_shard_placement WHERE shardid=…`).
5. Считывает данные:
    - На координаторе `SELECT personid, dateorder, dishid FROM personcanteenorder WHERE personid IN (...);`

- На соответствующих воркерах и их репликах `SELECT personid, dateorder, dishid FROM personcanteenorder_<shardid> WHERE personid=…;`

Фрагмент реального запуска:

```
$ make citus
./scripts/check-citus.sh
```

```
SELECT sp.shardid, sp.nodename, sp.nodeport, s.shardminvalue, s.shardmaxvalue
FROM pg_dist_shard_placement sp
JOIN pg_dist_shard s USING (shardid)
WHERE s.logicalrelid='personcanteenorder'::regclass
ORDER BY sp.shardid, sp.nodename;
```

```
 shardid |   nodename    | nodeport | shardminvalue | shardmaxvalue
---------+---------------+----------+---------------+---------------
  102008 | citus-worker1 |     5432 | -2147483648   | -1
  102009 | citus-worker2 |     5432 | 0             | 2147483647
(2 rows)
```

```
INSERT INTO personcanteenorder
  (personid, dateorder, dishid)
VALUES
  (1763326427, CURRENT_DATE, 1);

INSERT INTO personcanteenorder
  (personid, dateorder, dishid)
VALUES
  (1763326428, CURRENT_DATE, 2);
```

```
SELECT personid, dateorder, dishid
FROM personcanteenorder
WHERE personid IN (1763326427, 1763326428)
ORDER BY personid;
```

```
  personid  | dateorder  | dishid
------------+------------+--------
 1763326427 | 2025-11-16 |      1
 1763326428 | 2025-11-16 |      2
(2 rows)
```

```
SELECT personid, dateorder, dishid
FROM personcanteenorder_102008
WHERE personid = 1763326427;
```

```
  personid  | dateorder  | dishid
------------+------------+--------
 1763326427 | 2025-11-16 |      1
(1 row)
```

```sql
SELECT personid, dateorder, dishid
FROM personcanteenorder_102008
WHERE personid = 1763326427;
```

```
  personid  | dateorder  | dishid
------------+------------+--------
 1763326427 | 2025-11-16 |      1
(1 row)
```

```sql
SELECT personid, dateorder, dishid
FROM personcanteenorder_102009
WHERE personid = 1763326428;
```

```
  personid  | dateorder  | dishid
------------+------------+--------
 1763326428 | 2025-11-16 |      2
(1 row)
```

```sql
SELECT personid, dateorder, dishid
FROM personcanteenorder_102009
WHERE personid = 1763326428;
```

```
  personid  | dateorder  | dishid
------------+------------+--------
 1763326428 | 2025-11-16 |      2
(1 row)
```

То есть координатор и оба воркера (включая реплики) возвращают записи, каждая — из «своей» шардовой таблицы.