

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 №4

**Тема: «Основы аналитики данных»**

Выполнил: студент 3 курса, гр. ИБ-31вп

\_\_\_\_\_ Ворожцов А. Е.

Принял:

\_\_\_\_\_ Медведев С. А.

Санкт-Петербург, 2025 г.

# Содержание

<b>Отчет по настройке PostgreSQL и построению куба</b>	<b>2</b>
1. Порты и внешние подключения к серверам	2
2. «Рабочая» база db1: имя, пользователь, FDW-доступ	2
2.1. Новый пользователь в рабочей базе	2
3. «Аналитическая» база db2: имя, FDW, агрегат, куб и доступ к нему	3
3.1. Пользователь для импорта внешних таблиц из db1	3
3.2. Подключение внешних таблиц в аналитической базе (FDW к db1)	3
3.3. Агрегатная функция гармонического среднего	4
3.4. Скрипт создания куба	5
3.5. Пользователь аналитической БД для чтения куба	5
3.6. Внешнее подключение к кубу из рабочей БД	5
4. Изменения в Canteen Dishes, обновление и проверка куба	6

# Отчет по настройке PostgreSQL и построению куба

## 1. Порты и внешние подключения к серверам

Сервисы поднимаются через `docker-compose.yml`:

```
services:
  db1:
    container_name: postgres-db1
    environment:
      POSTGRES_DB: db1
      POSTGRES_USER: user
      POSTGRES_PASSWORD: user
    ports:
      - "5432:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U user -d db1"]

  db2:
    container_name: postgres-db2
    environment:
      POSTGRES_DB: db2
      POSTGRES_USER: user
      POSTGRES_PASSWORD: user
    ports:
      - "5433:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U user -d db2"]
```

- «Рабочая» БД: db1, внешний порт 5432.
- «Аналитическая» БД: db2, внешний порт 5433.
- Для проверок доступности используются команды вида `pg_isready -U user -d db1|db2`.

## 2. «Рабочая» база db1: имя, пользователь, FDW-доступ

Имя базы: db1 (см. POSTGRES\_DB выше).

### 2.1. Новый пользователь в рабочей базе

Пользователь только для чтения, через которого аналитическая БД обращается к данным db1:

```
-- db1/init/03-user.sql
CREATE USER fdw_reader_db1 WITH PASSWORD 'fdw_reader_db1';

GRANT CONNECT ON DATABASE db1 TO fdw_reader_db1;

GRANT USAGE ON SCHEMA public TO fdw_reader_db1;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO fdw_reader_db1;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO fdw_reader_db1;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT ON TABLES TO fdw_reader_db1;
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT USAGE, SELECT ON SEQUENCES TO fdw_reader_db1;
```

Итог: fdw\_reader\_db1 имеет только права чтения и автоматически получает доступ ко всем новым таблицам/последовательностям в public.

### 3. «Аналитическая» база db2: имя, FDW, агрегат, куб и доступ к нему

Имя базы: db2.

#### 3.1. Пользователь для импорта внешних таблиц из db1

```
-- db2/init/02-fdw-worker-db2.sql
CREATE USER fdw_worker_db2 WITH PASSWORD 'fdw_worker_db2';

GRANT CONNECT ON DATABASE db2 TO fdw_worker_db2;
GRANT USAGE, CREATE ON SCHEMA public TO fdw_worker_db2;
```

#### 3.2. Подключение внешних таблиц в аналитической базе (FDW к db1)

```
-- db2/init/03-fdw.sql
CREATE EXTENSION IF NOT EXISTS postgres_fdw;

CREATE SERVER cube_server_db1
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'db1', port '5432', dbname 'db1');

GRANT USAGE ON FOREIGN SERVER cube_server_db1 TO fdw_worker_db2;
GRANT USAGE ON FOREIGN SERVER cube_server_db1 TO CURRENT_USER;

CREATE USER MAPPING IF NOT EXISTS FOR fdw_worker_db2
SERVER cube_server_db1
OPTIONS (user 'fdw_reader_db1', password 'fdw_reader_db1');

CREATE USER MAPPING IF NOT EXISTS FOR CURRENT_USER
SERVER cube_server_db1
OPTIONS (user 'fdw_reader_db1', password 'fdw_reader_db1');
```

Импорт схемы из db1 в db2, кроме объекта куба (он создается локально):

```
# db2/init/04-import.sh (фрагмент)
psql -v ON_ERROR_STOP=1 -U fdw_worker_db2 -d db2 <<'SQL'
IMPORT FOREIGN SCHEMA public
EXCEPT (canteen_price_h_mean_cube)
FROM SERVER cube_server_db1
```

```
INTO public;  
SQL
```

### 3.3. Агрегатная функция гармонического среднего

```
-- db2/init/05-h-cube.sql  
CREATE OR REPLACE FUNCTION h_accum  
(  
    state numeric[],  
    value numeric  
)  
RETURNS numeric[] AS $$  
BEGIN  
    IF value IS NULL THEN  
        RETURN state;  
    END IF;  
  
    IF value = 0 THEN  
        RAISE EXCEPTION 'Harmonic mean is undefined for zero values';  
    END IF;  
  
    state[1] := state[1] + 1 / value;  
    state[2] := state[2] + 1;  
  
    RETURN state;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE FUNCTION h_final(state numeric[])  
RETURNS numeric AS $$  
BEGIN  
    IF state[2] = 0 OR state[1] = 0 THEN  
        RETURN NULL;  
    END IF;  
  
    RETURN state[2] / state[1];  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE AGGREGATE h_mean(numeric)  
(  
    SFUNC = h_accum,  
    STYPE = numeric[],  
    FINALFUNC = h_final,  
    INITCOND = '{0,0}'  
);
```

### 3.4. Скрипт создания куба

```
-- db2/init/06-h-mean-canteen-cube.sql
CREATE MATERIALIZED VIEW canteen_price_h_mean_cube AS
SELECT
    d.dishtype                                AS dishtype_id,
    CASE WHEN GROUPING(d.dishtype) = 0
        THEN MIN(dt.name)
    END                                        AS dishtype_name,
    d.cookid                                  AS cook_id,
    CASE WHEN GROUPING(d.cookid) = 0
        THEN MIN(p.name)
    END                                        AS cook_name,
    h_mean(d.price)                           AS harmonic_price_per_portion,
    CASE
        WHEN GROUPING(d.dishtype)=0 AND GROUPING(d.cookid)=0
        THEN string_agg(DISTINCT d.dishname, ', ' ORDER BY d.dishname)
    END                                        AS dish_names
FROM canteendishes d
LEFT JOIN dishtype dt ON dt.dishtype = d.dishtype
LEFT JOIN persons p ON p.personid = d.cookid
WHERE d.price IS NOT NULL
    AND d.price > 0
GROUP BY CUBE (d.dishtype, d.cookid);
```

### 3.5. Пользователь аналитической БД для чтения куба

```
-- db2/init/07-cube-reader.sql
CREATE USER fdw_reader_db2 WITH PASSWORD 'fdw_reader_db2';

GRANT CONNECT ON DATABASE db2 TO fdw_reader_db2;
GRANT USAGE ON SCHEMA public TO fdw_reader_db2;

GRANT SELECT ON TABLE canteen_price_h_mean_cube TO fdw_reader_db2;
```

### 3.6. Внешнее подключение к кубу из рабочей БД

```
-- db1/init/04-cube-fdw.sql
CREATE EXTENSION IF NOT EXISTS postgres_fdw;

CREATE SERVER IF NOT EXISTS cube_server_db2
    FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (host 'db2', port '5432', dbname 'db2');

CREATE USER MAPPING IF NOT EXISTS FOR CURRENT_USER
    SERVER cube_server_db2
    OPTIONS (user 'fdw_reader_db2', password 'fdw_reader_db2');
```

```
CREATE FOREIGN TABLE IF NOT EXISTS canteen_price_h_mean_cube (
  dishtype_id integer,
  dishtype_name character varying(100),
  cook_id integer,
  cook_name character varying(100),
  harmonic_price_per_portion numeric,
  dish_names text
) SERVER cube_server_db2
  OPTIONS (schema_name 'public', table_name 'canteen_price_h_mean_cube');
```

## 4. Изменения в Canteen Dishes, обновление и проверка куба

Ниже приведен полный текст консольного вывода `make run` по шагам, с указанием соответствующих SQL-запросов. Этот вывод включает изменения в Canteen Dishes, обновление материализованного куба и проверку чтения куба через FDW из db1.

Команда запуска:

```
make run
```

### Шаг 1. Обновление куба:

```
REFRESH MATERIALIZED VIEW canteen_price_h_mean_cube;
```

```
./scripts/harmonic_mean.sh
=== Обновляем куб ===
REFRESH MATERIALIZED VIEW
```

### Шаг 2. Вывод всего куба:

```
SELECT dishtype_id, dishtype_name, cook_id, cook_name,
       harmonic_price_per_portion, dish_names
FROM canteen_price_h_mean_cube
ORDER BY dishtype_id NULLS FIRST,
         cook_id NULLS FIRST,
         dishtype_name,
         cook_name;
```

```
=== Весь куб ===
dishtype_id | dishtype_name | cook_id | cook_name | harmonic_price_per_portion |
↪ dish_names
-----+-----+-----+-----+-----+-----
↪ -----+-----+-----+-----+-----+-----
      |      |      |      |      43.35357021173420494577 |
      |      |      |      |      25.96786962078554610581 |
      |      |      |      |      43.30251352664913095041 |
      |      |      |      |      53.31057199350141640576 |
      |      |      |      |      52.81709167163787121991 |
      |      |      |      |      48.45118154078247239683 |
1 | Напитки |      |      |      19.07164480322906155366 |
1 | Напитки |      |      |      15.88235294117647058814 | Мин.вода "Бон аква", Мин.вода
↪ "Шишкин лес", Сок "J7" в ассортименте
1 | Напитки |      |      |      23.3333333333333333217 | Компот из абрикосов, Компот
↪ из яблок
1 | Напитки |      |      |      25.00000000000000000000 | Компот из слив
2 | Первое блюдо |      |      |      48.95909324773657495199 |
2 | Первое блюдо |      |      |      48.97959183673469388155 | Суп гороховый, Суп из
↪ морепродуктов
```

2	Первое блюдо	19	Ломоносов Игорь Павлович	48.59154929577464788390	Борщ, Суп куриный, Щи
2	Первое блюдо	20	Линина Мария Семеновна	45.00000000000000000450	Суп овощной
2	Первое блюдо	21	Бабкина Надежда Григорьевна	54.99999999999999999450	Крем-суп из шампиньонов
3	Второе блюдо			83.55145249702878346883	
3	Второе блюдо	18	Сливкина Наталья Эдуардовна	82.89951923076923077601	Котлета жаренная, Стейк из
↪ семги					
3	Второе блюдо	19	Ломоносов Игорь Павлович	100.34262948207171317804	Бефстроганов, Рыба жаренная
3	Второе блюдо	20	Линина Мария Семеновна	65.18518518518518132	Бедро куриное, Сосиски отварные
3	Второе блюдо	21	Бабкина Надежда Григорьевна	110.60000000000000000686	Гуляш из говядины
4	Гарнир			45.94254492572856597086	
4	Гарнир	19	Ломоносов Игорь Павлович	38.85421994884910486016	Гречневая каша, Макароны
4	Гарнир	20	Линина Мария Семеновна	44.79999999999999999713	Смесь золотистого и дикого риса
4	Гарнир	21	Бабкина Надежда Григорьевна	52.81502905511184228649	Картофель по-русски,
↪ Картофельное пюре, Плов					
5	Десерт			65.11516247789005689893	
5	Десерт			71.14757978971344488353	Булочка с маком, Пирожное
↪ вишневое, Тирамиссу					
5	Десерт	18	Сливкина Наталья Эдуардовна	55.80035650623885918442	Пирожное "муравейник",
↪ Яблочный штрудель					
5	Десерт	19	Ломоносов Игорь Павлович	52.300000000000000000335	Булочка с творогом
5	Десерт	20	Линина Мария Семеновна	96.399999999999999996800	Творожная запеканка
6	Салат			47.53900839622835277742	
6	Салат	18	Сливкина Наталья Эдуардовна	44.43684200433390223619	Салат морковный, Салат
↪ оливье, Сельдь под шубой					
6	Салат	19	Ломоносов Игорь Павлович	60.400000000000000000435	Салат греческий
6	Салат	20	Линина Мария Семеновна	47.16954810097334395911	HM Demo Salad 1, HM Demo
↪ Salad 2, Салат "мостик", Салат крабовый					
6	Салат	21	Бабкина Надежда Григорьевна	48.900000000000000000900	Салат весенний

(34 rows)

### Шаг 3. Самый “неэффективный” повар:

```
SELECT cook_id, cook_name,
       harmonic_price_per_portion,
       (1.0 / harmonic_price_per_portion) AS portions_per_rub
FROM canteen_price_h_mean_cube
WHERE dishtype_id IS NULL
      AND cook_id IS NOT NULL
ORDER BY harmonic_price_per_portion DESC NULLS LAST
LIMIT 1;
```

=== Самый неэффективный повар ===

Чем больше harmonic\_price\_per\_portion, тем “неэффективнее” повар: за 1 рубль получается меньше порций (в сценарии равного ↪ бюджета на позицию).

cook_id	cook_name	harmonic_price_per_portion	portions_per_rub
19	Ломоносов Игорь Павлович	53.31057199350141640576	0.01875800545006158378

(1 row)

### Шаг 4. Демоданные для куба: очистка, вставка и повторная проверка:

```
-- Очистка демоданных
DELETE FROM canteendishes
WHERE dishid IN (10001,10002)
      OR dishname LIKE 'HM Demo%';

SELECT count(*) AS demo_rows_in_canteendishes
FROM canteendishes
WHERE dishid IN (10001,10002)
      OR dishname LIKE 'HM Demo%';
```

=== Демоданные для куба: очистка → проверка отсутствия → вставка → refresh → проверка наличия ===  
 – Удаляем старые демо-строки в db1 (dishid IN 10001,10002 | name LIKE 'HM Demo%')



```
DELETE 2
demo_rows_in_canteendishes
-----
0
(1 row)
```

*-- Состояние куба для (dishtype=6, cook=20) до вставки демоданных*  
**REFRESH MATERIALIZED VIEW** canteen\_price\_h\_mean\_cube;

```
SELECT dishtype_id, dishtype_name, cook_id, cook_name,
       harmonic_price_per_portion, dish_names
FROM canteen_price_h_mean_cube
WHERE dishtype_id = 6
      AND cook_id   = 20;
```

– Обновляем куб и показываем группу (dishtype=6, cook=20) – до вставки

```
REFRESH MATERIALIZED VIEW
dishtype_id | dishtype_name | cook_id |      cook_name      | harmonic_price_per_portion |      dish_names
-----+-----+-----+-----+-----+-----
6 | Салат        | 20 | Линина Мария Семеновна | 49.37975708502024291448 | Салат "мостик", Салат крабовый
(1 row)
```

*-- Вставка демоданных*

```
INSERT INTO canteendishes (dishid, dishname, dishtype, cookid, price)
VALUES
  (10001, 'HM Demo Salad 1', 6, 20, 35.5),
  (10002, 'HM Demo Salad 2', 6, 20, 62.0);
```

```
SELECT count(*) AS demo_rows_after_insert
FROM canteendishes
WHERE dishid IN (10001,10002)
      OR dishname LIKE 'HM Demo%';
```

– Вставляем демо-блюда в db1 под (dishtype=6, cook=20)

```
INSERT 0 2
demo_rows_after_insert
-----
2
(1 row)
```

*-- Состояние куба для (dishtype=6, cook=20) после вставки демоданных*  
**REFRESH MATERIALIZED VIEW** canteen\_price\_h\_mean\_cube;

```
SELECT dishtype_id, dishtype_name, cook_id, cook_name,
       harmonic_price_per_portion, dish_names
FROM canteen_price_h_mean_cube
WHERE dishtype_id = 6
      AND cook_id   = 20;
```

– Обновляем куб и показываем группу (dishtype=6, cook=20) – после вставки

```
REFRESH MATERIALIZED VIEW
dishtype_id | dishtype_name | cook_id |      cook_name      | harmonic_price_per_portion |      dish_names
-----+-----+-----+-----+-----+-----
6 | Салат        | 20 | Линина Мария Семеновна | 47.16954810097334395911 | HM Demo Salad 1, HM Demo Salad 2,
  ↳ Салат "мостик", Салат крабовый
(1 row)
```

**Шаг 5. Запрос куба через db1 (FDW → db2):**

```

SELECT dishtype_id, dishtype_name, cook_id, cook_name,
       harmonic_price_per_portion
FROM canteen_price_h_mean_cube
ORDER BY dishtype_id NULLS FIRST,
       cook_id      NULLS FIRST
LIMIT 10;

```

\n=== Запрос куба через db1 (FDW → db2, user=fdw\_reader\_db2) ===

dishtype_id	dishtype_name	cook_id	cook_name	harmonic_price_per_portion
				25.96786962078554610581
				43.35357021173420494577
		18	Сливкина Наталья Эдуардовна	43.30251352664913095041
		19	Ломоносов Игорь Павлович	53.31057199350141640576
		20	Линина Мария Семеновна	52.81709167163787121991
		21	Бабкина Надежда Григорьевна	48.45118154078247239683
1	Напитки			15.88235294117647058814
1	Напитки			19.07164480322906155366
1	Напитки	18	Сливкина Наталья Эдуардовна	23.333333333333333217
1	Напитки	21	Бабкина Надежда Григорьевна	25.000000000000000000

(10 rows)