Отчёт по итоговому проекту по курсу СУБД

Оглавление

[Введение 3](#_Toc200048986)

[Структура проекта 3](#_Toc200048987)

[SQL-скрипты для тестового наполнения 9](#_Toc200048988)

[Настройка Docker, Prometheus и Grafana 11](#_Toc200048989)

[docker-compose.yml 11](#_Toc200048990)

[prometheus.yml 14](#_Toc200048991)

[Grafana dashboard 14](#_Toc200048992)

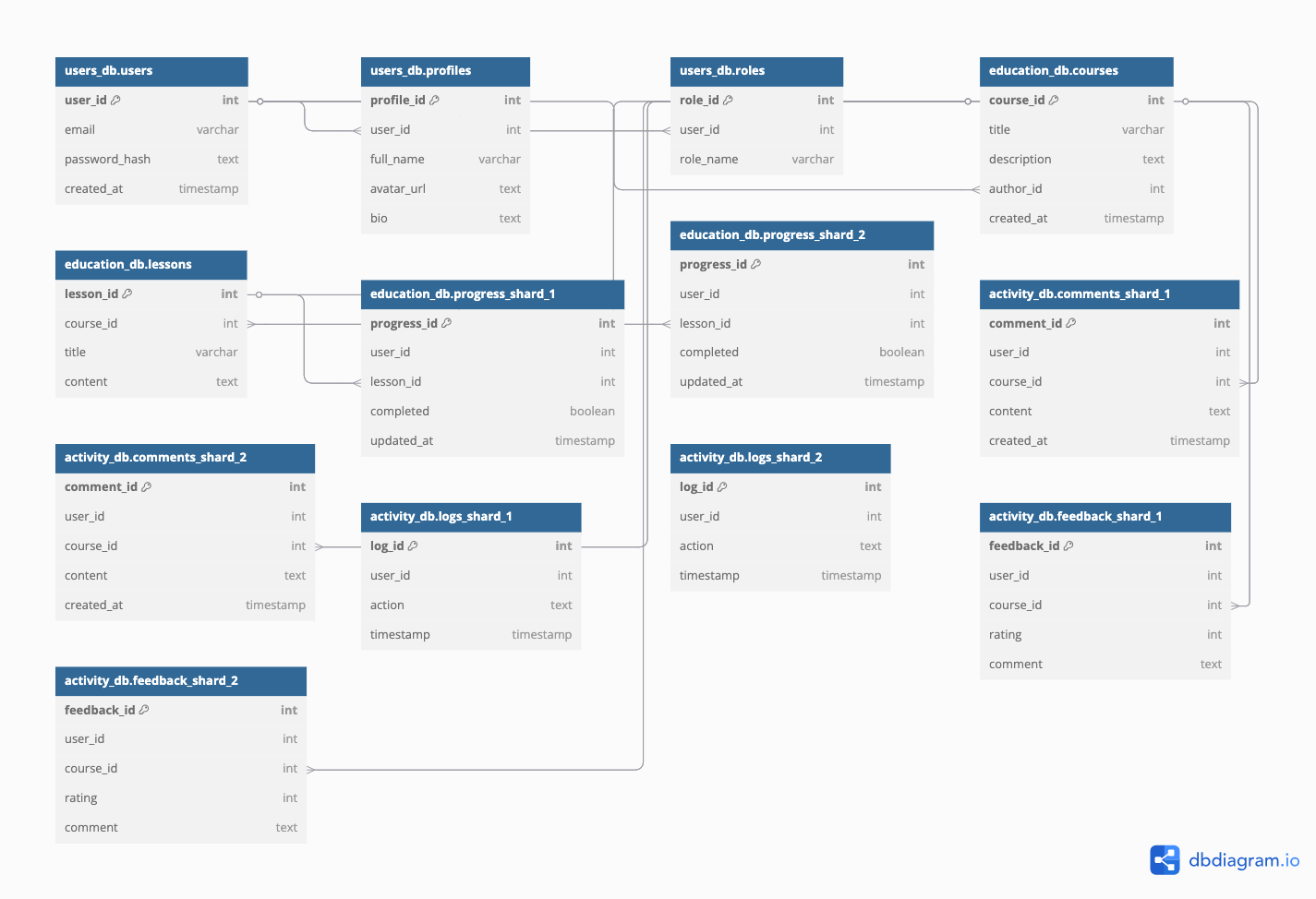
# 

# Введение

Тема: Платформа онлайн-курсов  
  
Цель: Создание масштабируемой и наблюдаемой БД-системы на PostgreSQL с шардированием и мониторингом.  
  
Компоненты проекта:  
- 3 PostgreSQL-базы: users\_db, education\_db, activity\_db  
- Prometheus, Grafana, Node Exporter, Postgres Exporter  
- Java-приложение с JDBC-доступом  
- Docker и docker-compose для автоматизированного развёртывания

# Структура проекта

На схеме ниже приведена ER-диаграмма проекта



Далее представлены **SQL-скрипты** по всем трём доменам, с учётом шардирования. Каждая таблица создаётся в отдельной схеме, соответствующей домену. Шардированные таблицы создаются в двух экземплярах с CHECK-ограничениями для имитации простого шардирования.

*-- users\_db: пользователи*

CREATE SCHEMA *IF* NOT EXISTS users\_db;

CREATE TABLE *users\_db*.users (

user\_id SERIAL *PRIMARY KEY*,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

created\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP

);

CREATE TABLE *users\_db*.profiles (

profile\_id SERIAL *PRIMARY KEY*,

user\_id INT *REFERENCES* users\_db.users(user\_id),

full\_name VARCHAR(255),

avatar\_url TEXT,

bio TEXT

);

CREATE TABLE *users\_db*.roles (

role\_id SERIAL *PRIMARY KEY*,

user\_id INT *REFERENCES* users\_db.users(user\_id),

role\_name VARCHAR(50) NOT NULL

);

*-- education\_db: учебный процесс*

CREATE SCHEMA *IF* NOT EXISTS education\_db;

CREATE TABLE *education\_db*.courses (

course\_id SERIAL *PRIMARY KEY*,

title VARCHAR(255) NOT NULL,

description TEXT,

author\_id INT *REFERENCES* users\_db.users(user\_id),

created\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP

);

CREATE TABLE *education\_db*.lessons (

lesson\_id SERIAL *PRIMARY KEY*,

course\_id INT *REFERENCES* education\_db.courses(course\_id),

title VARCHAR(255) NOT NULL,

content TEXT

);

CREATE TABLE *education\_db*.progress\_shard\_1 (

progress\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

lesson\_id INT *REFERENCES* education\_db.lessons(lesson\_id),

completed BOOLEAN *DEFAULT* FALSE,

updated\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 0)

);

CREATE TABLE *education\_db*.progress\_shard\_2 (

progress\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

lesson\_id INT *REFERENCES* education\_db.lessons(lesson\_id),

completed BOOLEAN *DEFAULT* FALSE,

updated\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 1)

);

*-- activity\_db: активность*

CREATE SCHEMA *IF* NOT EXISTS activity\_db;

CREATE TABLE *activity\_db*.comments\_shard\_1 (

comment\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

course\_id INT *REFERENCES* education\_db.courses(course\_id),

content TEXT NOT NULL,

created\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 0)

);

CREATE TABLE *activity\_db*.comments\_shard\_2 (

comment\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

course\_id INT *REFERENCES* education\_db.courses(course\_id),

content TEXT NOT NULL,

created\_at TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 1)

);

CREATE TABLE *activity\_db*.logs\_shard\_1 (

log\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

action TEXT NOT NULL,

timestamp TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 0)

);

CREATE TABLE *activity\_db*.logs\_shard\_2 (

log\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

action TEXT NOT NULL,

timestamp TIMESTAMP *DEFAULT* CURRENT\_TIMESTAMP,

*CHECK* (user\_id % 2 = 1)

);

CREATE TABLE *activity\_db*.feedback\_shard\_1 (

feedback\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

course\_id INT NOT NULL,

rating INT *CHECK* (rating BETWEEN 1 AND 5),

comment TEXT,

*CHECK* (course\_id % 2 = 0)

);

CREATE TABLE *activity\_db*.feedback\_shard\_2 (

feedback\_id SERIAL *PRIMARY KEY*,

user\_id INT NOT NULL,

course\_id INT NOT NULL,

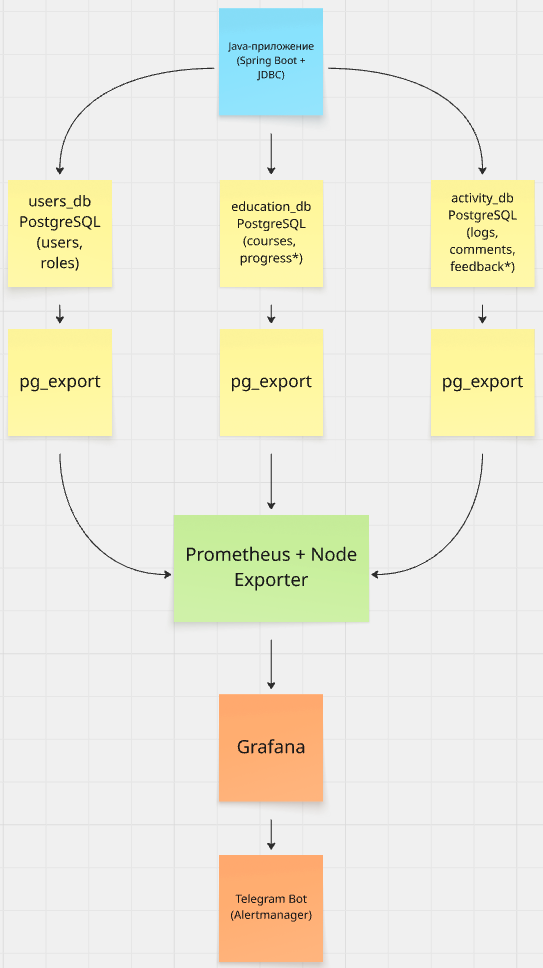
rating INT *CHECK* (rating BETWEEN 1 AND 5),

comment TEXT,

*CHECK* (course\_id % 2 = 1)

);

Архитектура проекта будет выглядеть следующим образом:



# SQL-скрипты для тестового наполнения

INSERT INTO users\_db.users (email, password\_hash) VALUES

('alice@example.com', 'hash1'),

('bob@example.com', 'hash2'),

('carol@example.com', 'hash3');

INSERT INTO users\_db.profiles (user\_id, full\_name, avatar\_url) VALUES

(1, 'Alice A.', 'https://...'),

(2, 'Bob B.', 'https://...');

INSERT INTO users\_db.roles (user\_id, role\_name) VALUES

(1, 'student'),

(2, 'instructor'),

(3, 'student');

INSERT INTO education\_db.courses (title, description, author\_id) VALUES

('Intro to Java', 'Basic Java course', 2),

('SQL for Beginners', 'Learn SQL', 2);

INSERT INTO education\_db.lessons (course\_id, title, content) VALUES

(1, 'Variables', '...'),

(1, 'Loops', '...'),

(2, 'SELECT queries', '...');

INSERT INTO education\_db.progress\_shard\_1 (user\_id, lesson\_id, completed) VALUES

(2, 1, true),

(2, 2, false);

INSERT INTO education\_db.progress\_shard\_2 (user\_id, lesson\_id, completed) VALUES

(1, 1, true),

(3, 3, true);

INSERT INTO activity\_db.comments\_shard\_1 (user\_id, course\_id, content) VALUES

(2, 1, 'Great lesson!');

INSERT INTO activity\_db.comments\_shard\_2 (user\_id, course\_id, content) VALUES

(1, 2, 'Thanks for the explanation');

INSERT INTO activity\_db.logs\_shard\_1 (user\_id, action) VALUES

(2, 'login');

INSERT INTO activity\_db.logs\_shard\_2 (user\_id, action) VALUES

(1, 'view\_lesson');

INSERT INTO activity\_db.feedback\_shard\_1 (user\_id, course\_id, rating, comment) VALUES

(2, 2, 5, 'Excellent');

INSERT INTO activity\_db.feedback\_shard\_2 (user\_id, course\_id, rating, comment) VALUES

(1, 1, 4, 'Very good');

# Настройка Docker, Prometheus и Grafana

## docker-compose.yml

version: '3.8'

services:

*# ====================*

*# PostgreSQL Instances*

*# ====================*

users\_db:

image: postgres:15

container\_name: users\_db

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: password

POSTGRES\_DB: users

ports:

- "5433:5432"

volumes:

- users\_data:/var/lib/postgresql/data

education\_db:

image: postgres:15

container\_name: education\_db

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: password

POSTGRES\_DB: education

ports:

- "5434:5432"

volumes:

- education\_data:/var/lib/postgresql/data

activity\_db:

image: postgres:15

container\_name: activity\_db

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: password

POSTGRES\_DB: activity

ports:

- "5435:5432"

volumes:

- activity\_data:/var/lib/postgresql/data

*# ====================*

*# Node Exporter*

*# ====================*

node\_exporter:

image: prom/node-exporter

container\_name: node\_exporter

ports:

- "9100:9100"

*# ====================*

*# Postgres Exporter*

*# ====================*

postgres\_exporter:

image: prometheuscommunity/postgres-exporter

container\_name: postgres\_exporter

environment:

DATA\_SOURCE\_NAME: "postgresql://postgres:password@users\_db:5432/users?sslmode=disable"

depends\_on:

- users\_db

ports:

- "9187:9187"

*# ====================*

*# Prometheus*

*# ====================*

prometheus:

image: prom/prometheus

container\_name: prometheus

volumes:

- ./prometheus.yml:/etc/prometheus/prometheus.yml

command:

- --config.file=/etc/prometheus/prometheus.yml

ports:

- "9090:9090"

*# ====================*

*# Grafana*

*# ====================*

grafana:

image: grafana/grafana

container\_name: grafana

ports:

- "3000:3000"

volumes:

- grafana\_data:/var/lib/grafana

volumes:

users\_data:

education\_data:

activity\_data:

grafana\_data:

## prometheus.yml

global:

scrape\_interval: 15s

scrape\_configs:

- job\_name: 'node\_exporter'

static\_configs:

- targets: ['node\_exporter:9100']

- job\_name: 'postgres\_exporter'

static\_configs:

- targets: ['postgres\_exporter:9187']

## Grafana dashboard

{

"dashboard": {

"id": null,

"title": "Business Dashboard - Online Courses",

"timezone": "browser",

"schemaVersion": 36,

"version": 1,

"refresh": "30s",

"panels": [

{

"type": "stat",

"title": "Total Registered Users",

"datasource": {

"type": "postgres",

"uid": "your\_postgres\_uid"

},

"targets": [

{

"format": "table",

"rawSql": "SELECT count(\*) FROM users\_db.users;",

"refId": "A"

}

],

"gridPos": {

"x": 0,

"y": 0,

"w": 6,

"h": 4

}

},

{

"type": "stat",

"title": "Completed Lessons",

"datasource": {

"type": "postgres",

"uid": "your\_postgres\_uid"

},

"targets": [

{

"format": "table",

"rawSql": "\n SELECT SUM(cnt) FROM (\n SELECT count(\*) as cnt FROM education\_db.progress\_shard\_1 WHERE completed = true\n UNION ALL\n SELECT count(\*) FROM education\_db.progress\_shard\_2 WHERE completed = true\n ) AS combined;\n ",

"refId": "B"

}

],

"gridPos": {

"x": 6,

"y": 0,

"w": 6,

"h": 4

}

},

{

"type": "stat",

"title": "Total Comments",

"datasource": {

"type": "postgres",

"uid": "your\_postgres\_uid"

},

"targets": [

{

"format": "table",

"rawSql": "\n SELECT SUM(cnt) FROM (\n SELECT count(\*) as cnt FROM activity\_db.comments\_shard\_1\n UNION ALL\n SELECT count(\*) FROM activity\_db.comments\_shard\_2\n ) AS combined;\n ",

"refId": "C"

}

],

"gridPos": {

"x": 0,

"y": 4,

"w": 6,

"h": 4

}

}

]

},

"overwrite": true

}

# JDBC-пример на Java