Национальный исследовательский ядерный университет
«МИФИ» (Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

ОТЧЁТ

Лабораторная работа №1-4:
«Переход на PostgreSQL»

Группа Б21-525
Студент Р.Т. Мясников
Преподаватель М.А. Куприяшин

Москва 2024

Оглавление

1. Настройка PostgreSQL

Запуск PostgreSQL был выполнен в Docker контейнере с использованием Docker Compose:

```
name: db_lab4

services:
  postgres:
    image: postgres
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_DB=db_lab4
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    volumes:
      - ./scripts:/docker-entrypoint-initdb.d
```

Подключение к СУБД было произведено с помощью клиента `psql`:

```
psql -h localhost -p 5432 -U postgres db_lab4
```

2. Создание таблиц

Для создания таблиц в PostgreSQL были использованы следующие запросы:

```
CREATE TABLE "kanjis" (
  "kanji" varchar PRIMARY KEY,
  "meaning" text,
  "key" varchar,
  "readings" text,
  "strokes" integer
);

CREATE TABLE "consist_kanjis" (
  "kanji" varchar NOT NULL,
  "consist_kanji" varchar NOT NULL
);

CREATE TABLE "words" (
  "word" text PRIMARY KEY,
  "reading" text,
  "meaning" text
);

CREATE TABLE "words_kanjis" (
  "word" varchar NOT NULL,
  "kanji" varchar NOT NULL
);

CREATE TABLE "users_lists_of_kanjis" (
  "username" text NOT NULL,
  "list_id" int NOT NULL
);

CREATE TABLE "lists_of_kanjis" (
  "id" integer PRIMARY KEY,
  "username" text,
  "table_name" text NOT NULL,
  "description" text
);

CREATE TABLE "lists_of_words" (
  "id" integer PRIMARY KEY,
  "username" text,
  "table_name" text NOT NULL,
  "description" text
);

CREATE TABLE "lists_words" (
  "list_id" int NOT NULL,
  "word" text NOT NULL
);

CREATE TABLE "lists_kanjis" (
  "list_id" int NOT NULL,
  "kanji" varchar NOT NULL
```

```
);

CREATE TABLE "users_lists_of_words" (
  "username" text NOT NULL,
  "list_id" int NOT NULL
);

ALTER TABLE "words_kanjis" ADD FOREIGN KEY ("kanji") REFERENCES "
   kanjis" ("kanji");

ALTER TABLE "words_kanjis" ADD FOREIGN KEY ("word") REFERENCES "
   words" ("word");

ALTER TABLE "consist_kanjis" ADD FOREIGN KEY ("kanji") REFERENCES "
   kanjis" ("kanji");

ALTER TABLE "consist_kanjis" ADD FOREIGN KEY ("consist_kanji")
   REFERENCES "kanjis" ("kanji");

ALTER TABLE "users_lists_of_kanjis" ADD FOREIGN KEY ("list_id")
   REFERENCES "lists_of_kanjis" ("id");

ALTER TABLE "lists_kanjis" ADD FOREIGN KEY ("kanji") REFERENCES "
   kanjis" ("kanji");

ALTER TABLE "lists_kanjis" ADD FOREIGN KEY ("list_id") REFERENCES "
   lists_of_kanjis" ("id");

ALTER TABLE "lists_words" ADD FOREIGN KEY ("list_id") REFERENCES "
   lists_of_words" ("id");

ALTER TABLE "lists_words" ADD FOREIGN KEY ("word") REFERENCES "
   words" ("word");

ALTER TABLE "users_lists_of_words" ADD FOREIGN KEY ("list_id")
   REFERENCES "lists_of_words" ("id");
```

Различия SQLite3 и PostreSQL:

- все идентификаторы были заключены в ковычки, так как они могут конфликтовать с зарезервированными словоми;
- в PostgreSQL отличается определение FOREIGN KEY: в данном случае они были определены с помощью запроса AFTER TABLE;
- в SQLite3 вместо CHAR использовались TEXT.

3. Выполнение запросов SELECT

1. Объединение двух списков слов

При выполнении этого запроса в `SQLite3` и `PostgerSQL` отличается порядок вывода строк. Причина заключается в том, что во втором после выполнения `UNION` порядок строк в результирующем наборе не гарантируется, если не используется дополнительный оператор ORDER BY, в отличие от `SQLite3`.

SQL запрос

```
SELECT word
FROM lists_words
WHERE list_id = 1
UNION
    SELECT word
    FROM lists_words
    WHERE list_id = 2;
```

Полученный результат

SQLite3	PostgreSQL
人	金曜日
先生	水曜日
土曜日	先生
女	男の子
女の子	日曜日
子供	月曜日
学生	男
日曜日	男女
日本人	火曜日
月曜日	学生
木曜日	父
水曜日	女の子
火曜日	女
父	子供
男	日本人
男の子	土曜日
男女	木曜日
金曜日	人

Результаты остальных запросов полностью совпали.

4. Дополнительные запросы

Вывод времени в днях, которое понадобится для изучения всех иероглифов из списка, предполагая, что на изучение иероглифа требуется (10 + 2 * (количество черт)) минут, а человек занимателся по 30 минут в день.

SQL запрос

```
SELECT lists_kanjis.list_id, (10+2*SUM(kanjis.strokes)-1)/30+1 AS
   days
FROM lists_kanjis
JOIN kanjis ON kanjis.kanji = lists_kanjis.kanji
GROUP BY lists_kanjis.list_id
ORDER BY lists_kanjis.list_id;
```

Полученный результат

list_id	days
1	3
2	6
3	2
4	4
5	2

5. Заключение

В ходе данной работы была запущена СУБД PostgreSQL внутри Docker контейнера с использованием Docker Compose. Далее были переписаны запросы с SQLite на PostgreSQL и рассмотрены различия между их результатами.

6. Приложение

Репозиторий: GitHub
SQL запросы: Scripts