-- 1 Задание

-- Создание таблицы users

CREATE TABLE users (

user\_id INT PRIMARY KEY, -- Тип INT, был выбор с SERIAL, но подумал для рандомной генерации его будет не очень удобно использовать

birth\_date DATE, -- Дата без времени

sex VARCHAR(10), -- Male | Female

age INT -- Возраст взял интовым значением

);

-- Создание таблицы items

CREATE TABLE items (

item\_id INT PRIMARY KEY, -- INT тип у ключа, по аналогии с users думал про SERIAL

description VARCHAR(255), -- описание товара, тип varchar

price NUMERIC(10, 2), -- цена товара до копеек, хотя в БД принято хранить целое значение цены и при выборке делить на 100

category VARCHAR(50) -- наименование категории товара

);

-- Создание таблицы ratings

CREATE TABLE ratings (

rating\_id INT PRIMARY KEY, -- INT ключ

item\_id INT REFERENCES items(item\_id), -- Внешний ключ, тип INT

user\_id INT REFERENCES users(user\_id), -- Внешний ключ, тип INT

review TEXT, -- Отзыв о товаре TEXT, не стал использовать varchar, потому что отзыв бывает и более 255 символов

rating INT -- INT рейтинг, оценка

);

-- Генерируем 20 случайных пользователей

INSERT INTO users (user\_id, birth\_date, sex, age)

SELECT

generate\_series AS user\_id,

now() - (random() \* (interval '30 years')) AS birth\_date,

CASE WHEN random() < 0.5 THEN 'Male' ELSE 'Female' END AS sex,

floor(random() \* 50 + 18)::int AS age

FROM generate\_series(1, 20);

-- Генерируем 20 случайных товаров

INSERT INTO items (item\_id, description, price, category)

SELECT

generate\_series AS item\_id,

md5(random()::text) || 'Item' AS description,

random() \* 1000 AS price,

unnest(array['Electronics', 'Clothing', 'Home Appliances']) AS category

FROM generate\_series(1, 20);

-- Генерируем 20 случайных отзывов

INSERT INTO ratings (rating\_id, item\_id, user\_id, review, rating)

SELECT

generate\_series AS rating\_id,

floor(random() \* 20 + 1) AS item\_id,

floor(random() \* 20 + 1) AS user\_id,

'Random review text ' || generate\_series AS review,

floor(random() \* 5 + 1) AS rating

FROM generate\_series(1, 20);

/\* В контексте генерации данных я столкнулся с неизвестной проблемой,

поначалу данные сгенерироваться не смогли, потом ничего не трогая и запустив скрипт ещё раз - всё получилось \*/

-- 2 Задание

/\*

Тип связи между таблицами users и ratings является одним ко многим (one-to-many).

Это означает, что каждый пользователь может оставить несколько отзывов,

но каждый отзыв принадлежит только одному пользователю.

Таким образом, таблица users связана с таблицей ratings с использованием внешнего ключа,

где user\_id в таблице ratings ссылается на user\_id в таблице users.

Тип связи между таблицами items и ratings также является одним ко многим (one-to-many).

Каждый товар может иметь несколько отзывов, но каждый отзыв принадлежит только одному товару.

Таким образом, таблица items связана с таблицей ratings с использованием внешнего ключа,

где item\_id в таблице ratings ссылается на item\_id в таблице items.

\*/

-- 3 Задание

/\*

Рекомендации по созданию индексов:

Для таблицы users рекомендуется создать индекс на user\_id,

так как это первичный ключ и часто используется для быстрого поиска конкретных пользователей.

Для таблицы items рекомендуется создать индекс на item\_id,

так как это первичный ключ и часто используется для быстрого поиска конкретных товаров.

Для таблицы ratings, индексы можно создать на столбцах item\_id и user\_id,

так как они используются для связывания отзывов с товарами и пользователями.

Эти индексы помогут ускорить поиск отзывов по конкретным товарам или пользователям.

Важно помнить, что индексы стоит создавать аккуратно,

дабы это в противном случае не замедлило операции поиска нужных данных, вставки и замены значений

\*/

-- 4 Задание

--1

--INSERT INTO Car VALUES ('7984672834', 'E340BT', '77', 'Lada Granta', 'Kрасный', 87, 2017, 35);

/\*

Запрос не выполнится,

поскольку в таблице car\_owner нет соответствующей записи владельца машины с таким идентификатором 7984672834

\*/

--2

--INSERT INTO Car\_owner VALUES ('7984672834', 'Иван Петров');

/\* Запрос отработает и добавит рекорд данных в таблицу car\_owner\*/

--3

--INSERT INTO Car\_owner VALUES ('7984672834', 'Татьяна Иванова');

/\* Запрос не отработает, поскольку в таблице уже есть запись с тем же INN владельца машины \*/

--4

--INSERT INTO Car Owner VALUES ('4752909757', 'Mван Петров')

/\* Запрос составлен некорректно и не отработает, таблица car\_owner существует в базе,

в отличие от car owner, к которой мы хотим применить DML операции \*/

--5

--INSERT INTO Car VALUES ('6239572784', 'E340BT, 77, 'Volkswagen Polo', 'Синий', 105, 2018, 40)

/\* Запрос не отработает, поскольку данные, которые мы хотим добавить составлены неправильно. \*/

--6

--INSERT INTO Car VALUES ('4752909757', 'A822EY', 99, 'Skoda Rapid', 'черный', 125, 2021, 35);

/\*

Запрос не выполнится,

поскольку в таблице car\_owner нет соответствующей записи владельца машины с таким идентификатором 4752909757

\*/

--7

--INSERT INTO Car VALUES ('7984672834', 'A822EY', 99, 'Hyundai Solaris', 'черный', 123, 2019, 20);

/\* Запрос отработает и добавит записи в таблицу \*/

--8

--INSERT INTO Car VALUES ('74478679847', '8971HP, 199, 'Kia Sportage', 'бeлый', 18, 2017, 35);

/\* Запрос составлен некорректно, данные в INSERT перемешались и не могут быть добавлены в таблицу \*/

--9

--INSERT INTO Car VALUES ('7984672834', 'E340BT, 77, 'Toyota RAV4', 'Серебристо-серый, 146, 2019);

/\* Запрос не отрботает по 2-м причинам

1. Если нет ошибок во вставке данных, то данные в таблице уже есть по этой машине, поэтому он вернёт ошибку exist

2. Если ошибка во вставке данных намеренная, он вернёт ошибку, потому что не может вставить такие данные в таблице \*/

--10

--INSERT INTO Car VALUES ('7984672834', 'H454EE', 98, 'Skoda Rapid', 'чeрный', 45, 2021, 0);

/\* Запрос не отработает, поскольку выдаст ошибку check constraint, Такая запись с таким INN уже есть в таблице \*/

-- 5 Задание

/\*

Да, данный скрипт можно использовать для шардинга таблицы documents на 32 фрагмента и создания 16 тестовых таблиц,

например, docs00, docs01, и так далее до docs15

Данный скрипт разбивает данные из таблицы documents на 16 фрагментов,

используя остаток от деления на 16 (подразумевая, что id - это уникальный идентификатор документа).

Немного допишем код:

-- Создание 16 тестовых таблиц

CREATE TABLE docs00 (LIKE documents);

CREATE TABLE docs01 (LIKE documents);

-- ...

-- Разделение данных и вставка в тестовые таблицы

INSERT INTO docs00 SELECT \* FROM documents WHERE (id % 16) = 0;

INSERT INTO docs01 SELECT \* FROM documents WHERE (id % 16) = 1;

-- ...

\*/