**Домашнее задание. SQL DDL.**

Создайте 3 таблицы: users и items, ratings. Таблица users содержит информацию о пользователях, совершающих заказы на маркетплейсе. Таблица items содержит информацию о товарах маркетплейса. Таблица ratings содержит информацию об отзывах на товары маркетплейса.

В таблице users создайте колонки user\_id, birth\_date, sex, age. В таблице orders создайте колонки item\_id, description, price, category. В таблице ratings создайте колонки item\_id, user\_id, review (отзыв пользователя на товар), rating (рейтинг товара, поставленный пользователем).

Обоснуйте выбор типов данных при создании таблиц. Подумайте, как связать таблицы. Добавьте дополнительные столбцы в таблицы, если сочтете это необходимым.

Добавьте несколько значений (~20) в каждую таблицу. Попробуйте сгенерировать значения для наполнения таблиц (а не прописывать их каждый раз вручную)

**Решение:**

Структура таблиц с указанием связей и типов данных:



**Структура таблиц в коде:**

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY NOT NULL,

birth\_day DATE NOT NULL,

sex varchar(15) NOT NULL,

age SMALLINT);

**Выбор типов данных в таблицы:**

User\_id – SERIAL, потому что обеспечивает autoincrement, первичный ключ, not null, потому что каждая строка в реляционной БД должна быть уникальной

Birth\_day – тип DATE, потому что дата рождения – это дата, не null, это нужно для того, чтобы можно было рассчитать поле age при необходимости

Sex – тип VARCHAR(15), потому что такое поле должно быть представлено строкой, это может быть строка типа F/M или male/female

Age – тип SMALLINT, причина – возраст – это число от 0 до преимущественно 100, вероятно, немного больше, что укладывается в рамки SMALLINT

CREATE TABLE orders (

item\_id SERIAL PRIMARY KEY NOT NULL,

description TEXT,

price DECIMAL not NULL,

category VARCHAR NOT NULL);

**Выбор типов данных в таблицы:**

item\_id - SERIAL, потому что обеспечивает autoincrement, первичный ключ, not null, потому что каждая строка в реляционной БД должна быть уникальной

description – тип TEXT, это связано с тем, что, как правило, описание товара – это большой текст, что с точки зрения маркетинга позволяет лучше объяснить клиенту почему он должен купить этот товар, может быть пустым, потому что бывают товары без описания

price – DECIMAL – вещественное число, зачастую продавцы могут указывать цены в виде 999.99$, например, для повышения привлекательности товара, не null.

Category – VARCHAR – категория – это очень важный элемент агрегации, является строкой, не null

CREATE TABLE ratings (

item\_id INTEGER NOT NULL REFERENCES orders,

user\_id INTEGER NOT NULL REFERENCES users,

review VARCHAR(255),

rating FLOAT);

**Выбор типов данных в таблицы:**

Item\_id – INTEGER – потому что важно сохранить возможность обращения к таблице orders, где item\_id представлен, как SERIAL, не null, указание на внешний ключ в таблицу orders

User\_id – INTEGER - потому что важно сохранить возможность обращения к таблице users, где item\_id представлен, как SERIAL, не null, указание на внешний ключ в таблицу users

Review – VARCHAR(255) – поле отзыва

**Генераторы для каждой из таблиц, создающие 20 строк:**

**Для таблицы orders:**

DO $$

DECLARE

r RECORD;

BEGIN

FOR r IN SELECT \* FROM generate\_series(1, 20)

LOOP

INSERT INTO orders (description, price, category)

VALUES (

md5(random()::text) || ' Description',

round((random() \* (1000 - 50 + 1) + 50)::numeric, 2),

(array['Электроника','Одежда','Книги','Спорт','Домашние товары'])[floor(random()\*5+1)]

);

END LOOP;

END $$;

**Для таблицы users:**

DO $$

DECLARE

r RECORD;

BEGIN

FOR r IN SELECT \* FROM generate\_series(1, 20)

LOOP

INSERT INTO users (birth\_day, sex, age)

VALUES (

(current\_date - (floor(random() \* 10000 + 7000))::int \* '1 day'::interval)::date,

(CASE WHEN random() < 0.5 THEN 'male' ELSE 'female' END),

floor(random() \* 31 + 20)::int

);

END LOOP;

END $$;

**Для таблицы ratings:**

DO $$

DECLARE

r RECORD;

BEGIN

FOR r IN SELECT \* FROM generate\_series(1, 20)

LOOP -- для каждого номера в списке делаем следующее:

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

VALUES (

floor(random() \* 20 + 1)::int,

floor(random() \* 20 + 1)::int,

'Review ' || md5(random()::text),

round((random() \* 5)::numeric, 2)

);

END LOOP;

END $$;

Какой тип связи/отношения между таблицами users и ratings? Между таблицами items, ratings?

**Ответ**: один ко многим. Это означает, что один пользователь может оставлять несколько отзывов на разные товары, а один товар может иметь несколько разных отзывов.

К какой/каким из созданных таблиц вы бы предложили создать индекс? Объясните, почему.

**Ответ:** Я бы предложил создать индексы следующим образом:

Таблица ratings: item\_id, user\_id – эта таблица можем содержать много строк, такой индекс ускорит поиск. Кроме того, значения могут фильтроваться и сортироваться по rating, поэтому я бы создал индекс и в данном случае

Таблица orders: price и category – могут проводиться селекты и сортировки

Таблица users: sex, birth\_day, age – могут проводиться селексы и сортировки

Есть две таблицы в базе, созданные следующим образом:

// Таблица1

// Таблица2

В таблицу ***последовательно (!!!!)*** добавляются следующие записи.“***Последовательно”*** означает, что сначала выполняется команда 1, затем команда 2 и т.д.

Отметьте те команды, которые смогут добавить записи в таблицу. Объясните, почему.

**Ответ:**

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

**Не выполнится**, т.к. количество значений не соответствует количеству столбцов. Имеется в виду, что в (..’E340BT) пропущена одинарная кавычка и в (..Lada Granta’) пропущена кавычка. Если исправить эти ковычки, то код всё равно не выполнится, потому что в таблице Car\_owner нет владельца с представленным ИНН.

1. INSERT INTO Car\_owner VALUES ('7984672834', 'Иван Петров')

**Выполнится.**

1. INSERT INTO Car\_ownerCar\_owner VALUES ('7984672834', 'Татьяна Иванова')

**Не выполнится**, т.к. запрос содержит ошибку- дважды указано название таблицы

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

**Не выполнится**, в запросе содержится ошибка в имени таблицы (прощенно \_, должно быть Car\_owner)

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

**Не выполнится**, проблема, аналогичная первому запросу – несоответствие значения количеству столбцов таблицы Car

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

**Не выполнится**. Значения запроса соответствуют количеству столбцов, но до этого не был добавлен владелец с ИНН '4752909757'.

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

**Выполнится**. До этого был добавлен владелец с ИНН '7984672834'. Количество значений соответствует количеству столбцов.

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

**Не выполнится**, т.к. значения (, '8971HP, 199, 'Kia Sportage') должны быть разделены, кроме того, присутствует ограничение CHECK(Power>50).

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

**Не выполнится**, т.к. ('E340BT, 77, 'Toyota RAV4') должны быть разделены. Кроме того, отсутствует значение пробега.

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

**Не выполнится**, т.к. сработает ограничение CHECK(Power>50).

Можно ли использовать данный скрипт для шардинга таблицы на 32 документа?  И генерацией из этой таблицы 16 тестовых таблиц примерно по 2 документа test.docs00, 01, 02, ..., 15.

INSERT INTO docs00

SELECT \* FROM documents WHERE (id%16)=0

...

INSERT INTO docs15

SELECT \* FROM documents WHERE (id%16)=15

**Ответ**: да, этот скрипт можно использовать для обозначенной цели, если перед его выполнением предусмотреть создание 16 таблиц docs00, docs01 … с соответствующей структурой таблиц. Каждая из 16 таблиц будет содержать документы, удовлетворяющие соответствующим условиям (id%16 = 0, …).