Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Лабораторная работа №3

Вариант 4456

Выполнил:

Хабиров Тимур Рустемович

Группа P3132

Преподаватели:

Гиря Максим Дмитриевич

Санкт-Петербург 2025

Содержание

Текст задания

Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

1. Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);

2. Приведите отношения в 3NF (как минимум). Постройте схему на основеNF (как минимум).

3. Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основеNF;

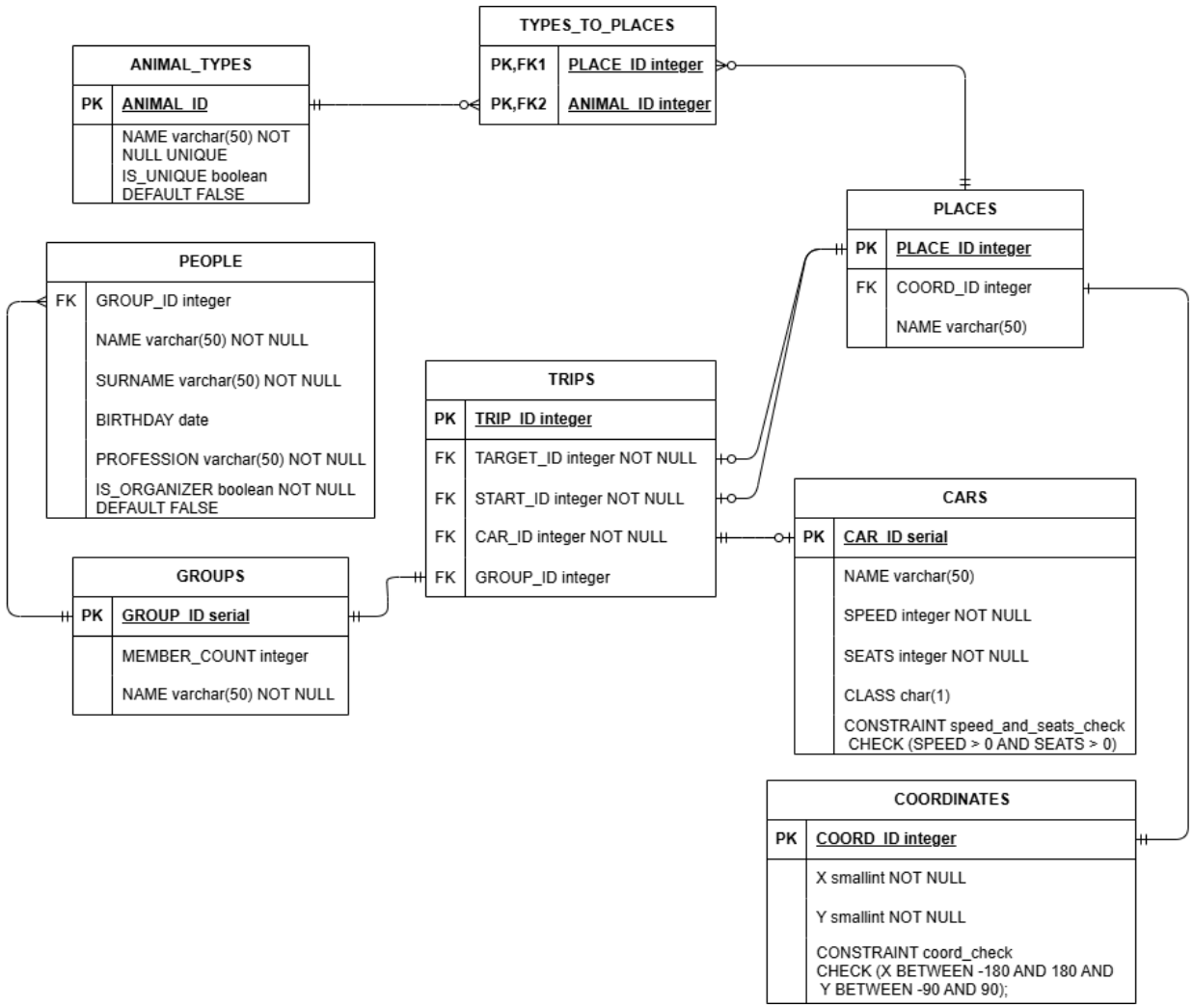
4. Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;

5. Какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

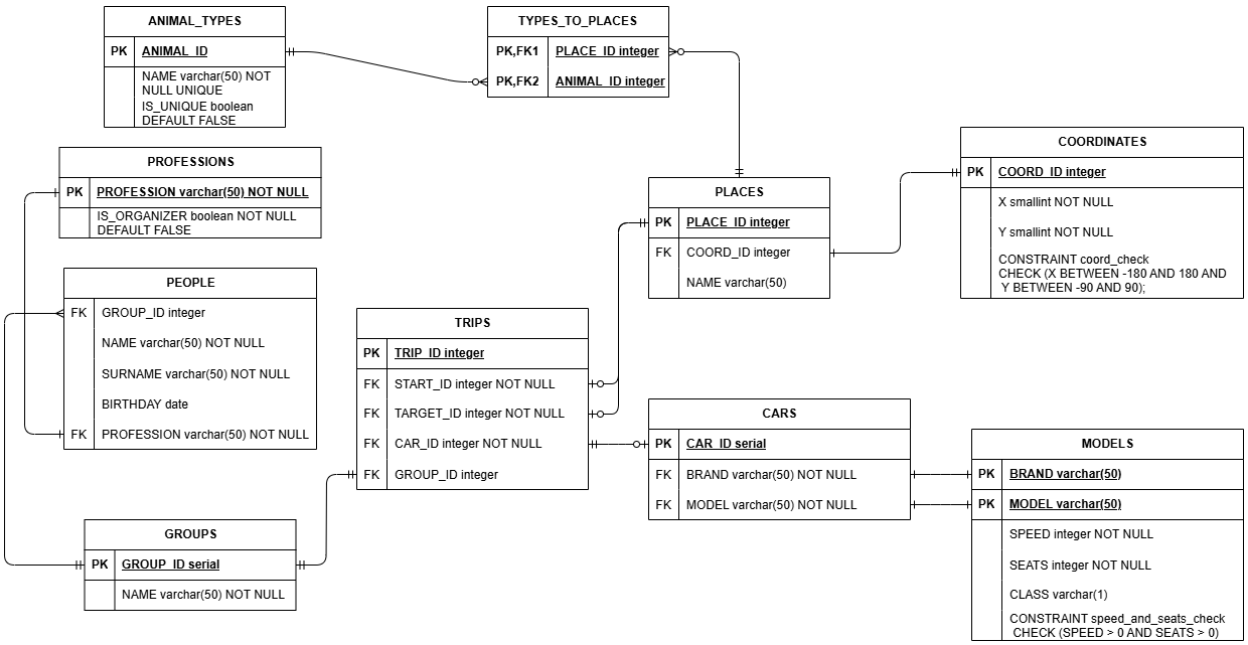
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

Исходная и нормализованная модели

Исходная модель:



Приведенная модель:



Ответы на вопросы, представленные в задании

Функциональные зависимости исходной схемы:

PEOPLE:	PROFESSION → IS_ORGANIZER
GROUPS:	GROUP_ID → (NAME, MEMBER_COUNT)
COORDINATES:	COORD_ID → (X, Y)
CARS:	CAR_ID → (NAME, SPEED, SEATS, CLASS)
	NAME → (SPEED, SEATS, CLASS)
ANIMAL_TYPES:	ANIMAL_ID → (NAME, IS_UNIQUE)
PLACES:	PLACE_ID → (COORD_ID, NAME)
TYPES_TO_PLACES:	(PLACE_ID, ANIMAL_ID) → ()
TRIPS:	TRIP_ID → (TARGET_ID, START_ID, CAR_ID, GROUP_ID)

Приведение к нормальным формам:

1NF: отношение, на пересечение каждой строки и столбца которого находится ровно одно значение. В моей схеме есть логическая ошибка в отношении CARS (атрибут NAME несет в себе несколько значений). Его следует разделить на два: BRAND и MODEL

Соответственно изменились функциональные связи в данной таблице:

CARS:	CAR_ID → (BRAND, MODEL, SPEED, SEATS, CLASS)
	(BRAND, MODEL) → (SPEED, SEATS, CLASS)

2NF: отношение находится в 1NF и все атрибуты, не являющиеся PK, находятся в полной функциональной зависимости от PK. Моя схема удовлетворяет данному условию, поскольку нет таких атрибутов, которые находились бы в зависимости от подмножества PK

3NF: отношение находится в 1NF и в 2NF и все атрибуты, не являющиеся PK, не находятся в транзитивной зависимости от PK. В моей схеме существует транзитивная зависимость в таблице CARS. Ее можно убрать создав новую таблицу MODELS с атрибутами PK(BRAND, MODEL) и SPEED, SEATS, CLASS

BCNF: отношение находится в BCNF, когда детерминанты всех ее функциональных зависимостей являются потенциальными ключами. Рассмотрим отношение PEOPLE. Оно имеет два потенциальных ключа: {NAME, SURNAME, BIRTHDAY} и {NAME, SURNAME, PROFESSION}, однако оно же имеет функциональную зависимость PROFESSION → IS_ORGANIZER, детерминант которого не является потенциальным ключом. Исправить это можно создав таблицу PROFESSIONS с атрибутами PROFESSION и IS_ORGANIZER

Денормализации:

В моей модели можно объединить таблицы PLACES и COORDINATES (убрать атрибут COORD_ID и добавить X, Y), что облегчит структуру модели, возможно, стоит объединить таблицы TRIPS и CARS в случае частых запросов к этим таблицам. Также можно добавить в таблицу GROUP атрибут count, считающий количество людей в группе, обновляющийся при каждом добавлении или удалении из таблицы PEOPLE

Функция и триггер на языке PL/pgSQL

Функция – вычисление времени путешествия

Реализация:

```
/*
Функция вычисляет время путешествия (сначала подсчитывается длина маршрута как дуга на шаре, затем данное расстояние длеится на скорость)

trip - id в таблице TRIPS
*/
CREATE FUNCTION count_trip_time(trip integer, OUT result real)
    AS $$
        DECLARE

            -- x - долгота
            -- y - широта

            x1 smallint;
            x2 smallint;
            y1 smallint;
            y2 smallint;

            rady1 float;
            rady2 float;

            radx1 float;
            radx2 float;

            sp integer;
        BEGIN
            IF trip IS NULL THEN
                RAISE EXCEPTION 'trip не может быть NULL!';
```

```
END IF;

SELECT X, Y INTO x1, y1
        FROM COORDINATES
        JOIN PLACES
                ON PLACES.COORD_ID = COORDINATES.COORD_ID
        JOIN TRIPS
                ON PLACES.PLACE_ID = TRIPS.START_ID
                        AND TRIP_ID = trip;
IF NOT FOUND THEN
        RAISE EXCEPTION 'нет trip с таким id';
END IF;

SELECT X, Y INTO x2, y2
        FROM COORDINATES
        JOIN PLACES
                ON PLACES.COORD_ID = COORDINATES.COORD_ID
        JOIN TRIPS
                ON PLACES.PLACE_ID = TRIPS.TARGET_ID
                        AND TRIP_ID = trip;

SELECT SPEED INTO sp
        FROM MODELS
        NATURAL JOIN CARS
        JOIN TRIPS
                ON TRIPS.CAR_ID = CARS.CAR_ID
                        AND TRIP_ID = trip;

IF sp IS NULL THEN
        RAISE EXCEPTION 'car не может быть NULL';
END IF;
```

```
                IF x1 = x2 AND y1 = y2 THEN
                        result = 0;
                ELSE
                        radx1 = pi() * x1 / 180;
                        radx2 = pi() * x2 / 180;
                        rady1 = pi() * y1 / 180;
                        rady2 = pi() * y2 / 180;

                        result = 111.2 * ACOS(sin(rady1) * sin(rady2) + cos(rady1) *
cos(rady2) * cos(radx1 - radx2));
                        result = result / sp;
                END IF;

        END;
    $$ LANGUAGE plpgsql;
```

Триггер – динамический подсчет количества участников в группе

Реализация:

```
/*
Функция-триггер обновляет значение поля member_count при DELETE, INSERT или
UPDATE
*/

CREATE FUNCTION update_member_count()
    RETURNS TRIGGER
    AS $$
    BEGIN
        IF (TG_OP = 'DELETE') THEN
            UPDATE GROUPS SET member_count = member_count - 1
                WHERE GROUP_ID = OLD.GROUP_ID;
        ELSIF (TG_OP = 'INSERT') THEN
            UPDATE GROUPS SET member_count = member_count + 1
                WHERE GROUP_ID = NEW.GROUP_ID;
```

```
        ELSIF (TG_OP = 'UPDATE') THEN
                UPDATE GROUPS SET member_count = member_count - 1
                        WHERE GROUP_ID = OLD.GROUP_ID;
                UPDATE GROUPS SET member_count = member_count + 1
                        WHERE GROUP_ID = NEW.GROUP_ID;
        END IF;
        RETURN NULL;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER member_counter
AFTER INSERT OR UPDATE OR DELETE ON PEOPLE
    FOR EACH ROW EXECUTE FUNCTION update_member_count();
```

Вывод

В ходе выполнения лабораторной работы я познакомился с различными нормальными формами и способами приведения к ним, ознакомился со структурой PL/pgSQL, научился создавать функции и создавать триггеры