МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РФ

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

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине
БАЗЫ ДАННЫХ

Выполнила:
Студент группы P3132
Чмурова Мария
Владиславовна
Проверила:
Харитонова
Анастасия Евгеньевна

г. Санкт-Петербург
2023 год

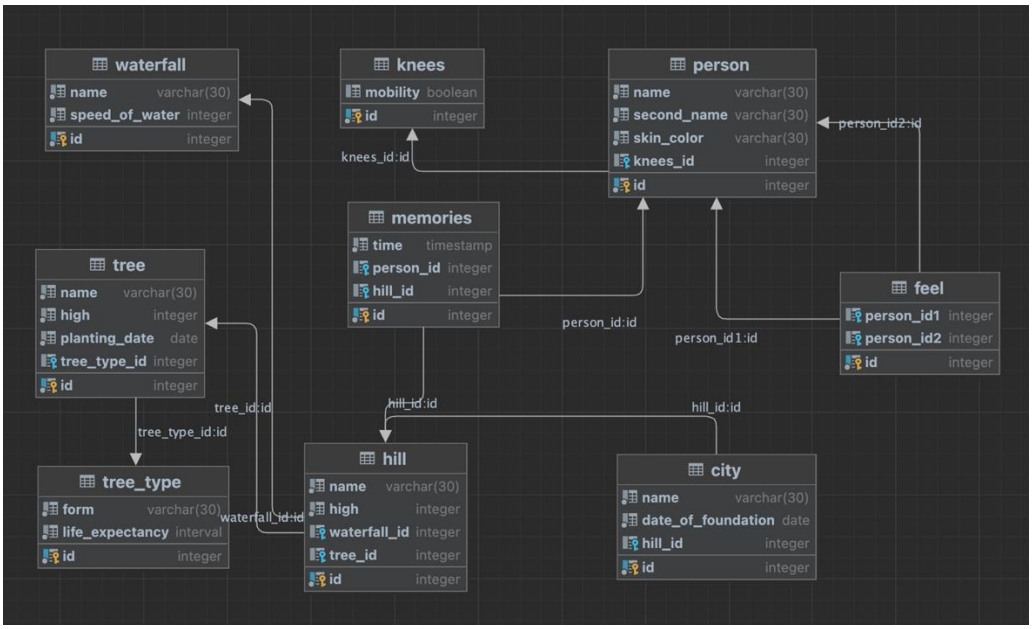
Задание

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

- Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
- Приведите отношения в 3NF (как минимум). Постройте схему на основе NF (как минимум).
- Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе NF;
- Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;
- 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 схема:



Ход работы

Функциональные зависимости:

person: id -> (name, second_name, skin_color, knees_id)

city: id -> (name, date_of_foundation, hill_id)

hll: id -> (name, high, waterfall_id, tree_id)

waterfall: id -> (name, speed_of_water)

tree: id -> (name, high, planting_dat, tree_type_id)

memories: id -> (time, person_id, hill_id)

feel: id -> (perdon_id1, person_id2)

knees: id -> (mobility)

tree_type: id -> (form, life_expectancy)

Приведение к 1NF:

Первая нормальная форма определяет, что каждая ячейка таблицы должна содержать только одно значение, а не список значений или структурированные данные. Каждая колонка в таблице должна иметь уникальное имя, а каждая строка должна иметь уникальный идентификатор, так называемый первичный ключ.

Модель находится в 1NF, так как не имеет повторяющихся столбцов и в каждой ячейке находится только 1 значение.

Приведение к 2NF

Вторая нормальная форма применяется, когда таблица имеет составной первичный ключ и некоторые поля зависят только от части ключа, а не от всего ключа целиком. В этом случае поля, зависящие от полного составного ключа, выделяются в отдельную таблицу, связанную с исходной таблицей через внешний ключ. Таким образом, достигается устранение избыточности данных.

В базе данных не имеется составных первичных ключей, и она уже приведена к форме 1NF, поэтому она находится в 2NF.

Приведение к 3NF

Применяется, когда таблица находится во второй нормальной форме, и при этом нет транзитивных функциональных зависимостей между не ключевыми полями

Модель находится в 3NF, потому что она уже находится в 2NF, а также все не ключевые атрибуты зависят только от первичных ключей.

Преобразование отношения в BCNF

Так как в базе данных нет составных ключей, то она автоматически соответствует условиям BCNF.

Полезные для схемы денормализации

Таблица "tree" содержит информацию о типе дерева, которая хранится в отдельной таблице "tree_type". Чтобы избежать JOIN для объединения таблиц ‘tree” и “tree_type”, можно денормализовать данные и добавить поля из таблицы "tree_type" в таблицу "tree". Таким образом, избегается необходимость объединения таблиц для получения полной информации о дереве и его типе

Соединение таблицы "city" и "hill" может быть денормализовано путем добавления информации о холмах в таблицу "city". Это может быть полезно, если требуется частый доступ к данным о холмах для конкретных городов, и такие запросы будут выполняться более эффективно без необходимости объединения таблиц

Логирующий триггер:

--Создание таблицы для хранения логов изменений в таблице "person"

```
CREATE TABLE person_changes_log (
    time DATE,
    type VARCHAR(200) NOT NULL,
    primary_key INT
);
```

--Создание функции для обработки триггера

```
CREATE OR REPLACE FUNCTION log_person_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO person_changes_log (time, type, primary_key)
        VALUES (current_date, 'INSERT', NEW.id);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO person_changes_log (time, type, primary_key)
        VALUES (current_date, 'UPDATE', NEW.id);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

--Создание триггера для таблицы "person"

```
CREATE TRIGGER person_changes_trigger
AFTER INSERT OR UPDATE ON person
FOR EACH ROW
EXECUTE FUNCTION log_person_changes();
```

Вывод

В ходе данной лабораторной работы я познакомилась с триггерами в PostgrreSQL и узнала про три нормальные формы, которыми должны обладать таблицы