МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И
КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

«Базы данных»

Выполнил:
Студент группы P3131
Родионов Максим Артемович
Преподаватель:
Вербовой Александр Александрович

Задание

Лабораторная работа #3

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

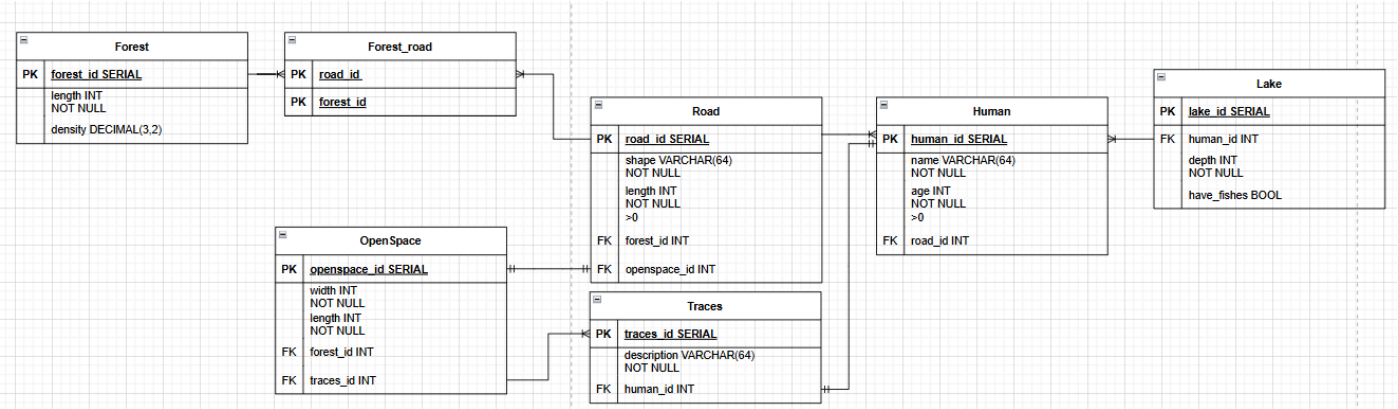
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

Отчёт по лабораторной работе должен содержать:

1. Текст задания.
2. Исходная, нормализованная и денормализованная модели.
3. Ответы на вопросы, представленные в задании.
4. Функция и триггер на языке PL/pgSQL
5. Выводы по работе.

Темы для подготовки к защите лабораторной работы:

1. Нормализация. Формы
2. Функциональные зависимости. Виды
3. Денормализация
4. Язык PL/pgSQL



Функциональные зависимости

- **forest**: (forest_id) -> (length, density)
- **human**: (human_id) -> (name,age)
- **road**: (road_id) -> (shape, length)
- **forest_road**: (road_id, forest_id) -> ()
- **traces**: (traces_id) -> (description,human_id)

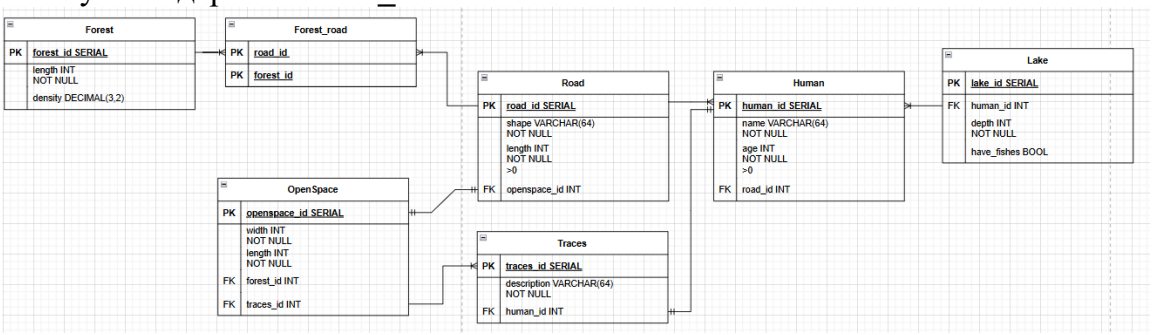
- **openspace**: (openspace_id) -> (width,length,forest_id,traces_id)
- **lake**: (lake_id) -> (depth,have_fishes,human_id)

Нормальные формы

- **1NF**: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения и отсутствуют повторяющиеся группы. Мои отношения удовлетворяет 1NF, так как все атрибуты атомарны, и нет повторяющихся групп.

- **2NF**: Отношение находится в 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от первичного ключа. Моя модель удовлетворяет 2NF, так как все неключевые атрибуты полностью функционально зависят от первичных ключей.

- **3NF**: Отношение находится в 3NF, если оно находится во 2NF и не содержит транзитивных зависимостей. Моя модель не удовлетворяет 3NF, так как у меня есть таблица **forest_road** для реализации связи многие-ко-многим, но таблица **road** уже содержит **forest_id**



- **forest**: (forest_id) -> (length, density)
- **human**: (human_id) -> (name,age)
- **road**: (road_id) -> (shape, length)

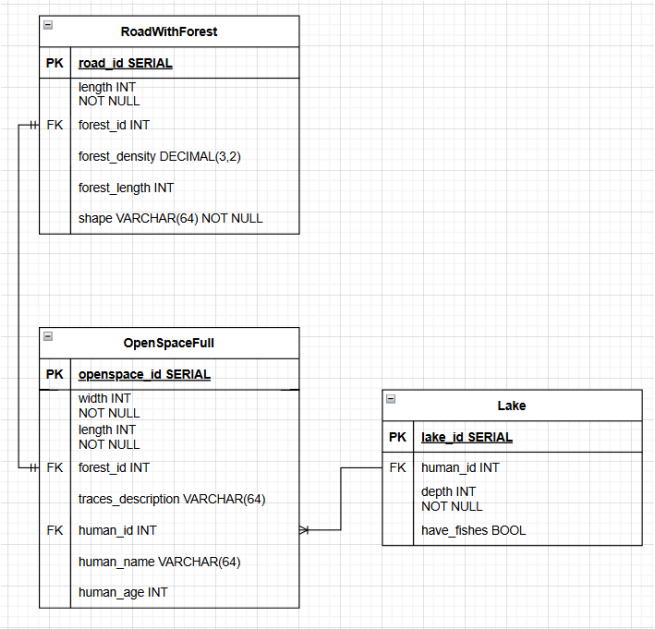
- **forest_road**: (road_id, forest_id) -> ()
- **traces**: (traces_id) -> (description,human_id)
- **openspace**: (openspace_id) -> (width,length,forest_id,traces_id)
- **lake**: (lake_id) -> (depth,have_fishes,human_id)

BCNF

- Отношение находится в BCNF, если для каждой функциональной зависимости X →Y, X является суперключом. Моя модель удовлетворяет BCNF, так как для всех функциональных зависимостей X является суперключом.

Денормализация

- **Объединение связанных таблиц**: В некоторых случаях, объединение таблиц может уменьшить количество операций JOIN, те уменьшить время обработки запросов. В моей схеме, можно рассмотреть объединение таблиц **road** и **forest**, для избежания джойнов (нарушает 3NF). Объединение таблиц **openspace**, **traces** и **human** для избежания джойнов (нарушает 3NF).



- **road_with_forest**: (road_id) -> (shape,length,forest_id,forest_length, forest_density)
  (forest_id) -> (forest_length, forest_density)

- **openspace_full**: (openspace_id) -> (width,length,forest_id,traces_description,human_id,human_name, human_age)
  (human_id) -> (human_name, human_age)

Триггер

Триггер при добавлении новых следов создаётся запись какие следы были добавлены, кем и когда. Для хранения логов используется отдельная таблица.

```
-- Триггер для создания логов следов
CREATE TABLE traces_log (
    log_id SERIAL PRIMARY KEY,
    traces_id INTEGER,
    description VARCHAR(64),
    human_id INTEGER,
    log_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION log_traces_insert()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO traces_log(traces_id, description, human_id)
    VALUES (NEW.traces_id, NEW.description, NEW.human_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_log_traces_insert
AFTER INSERT ON traces
FOR EACH ROW
EXECUTE FUNCTION log_traces_insert();
```

Заключение

Во время выполнения лабораторной работы я познакомился с процессами нормализации и денормализации. Научился анализировать схему и находить в ней узкие места. Узнал что такое триггер и потренировался писать свои реализации триггеров.