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 P3107  
 Чусовлянов Максим Сергеевич

***Преподаватель:***Байрамова Хумай Бахруз Кызы

# 

# **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.

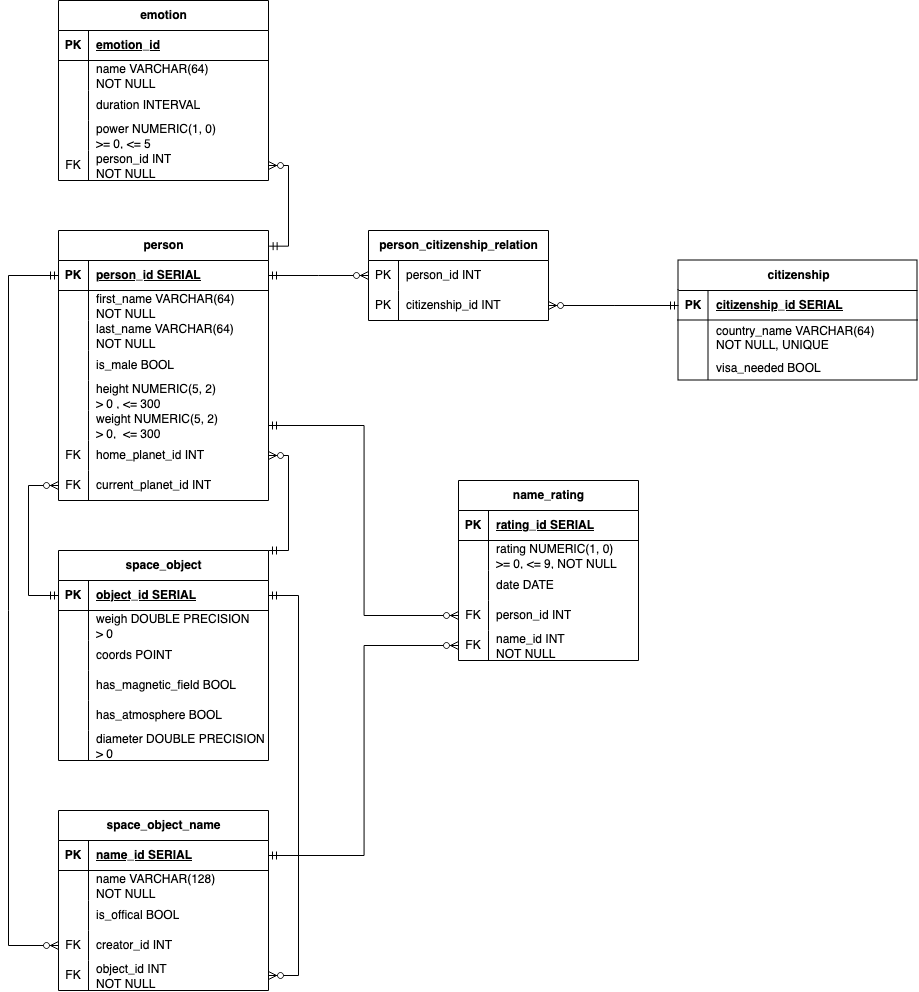
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

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**

* **emotion**: (emotion\_id) -> (name, duration, power, person\_id)
* **person**: (person\_id) -> (first\_name, last\_name, is\_male, height, weight, home\_planet\_id, current\_planet\_id)
* **citizenship**: (citizenship\_id) -> (country\_name, visa\_needed)

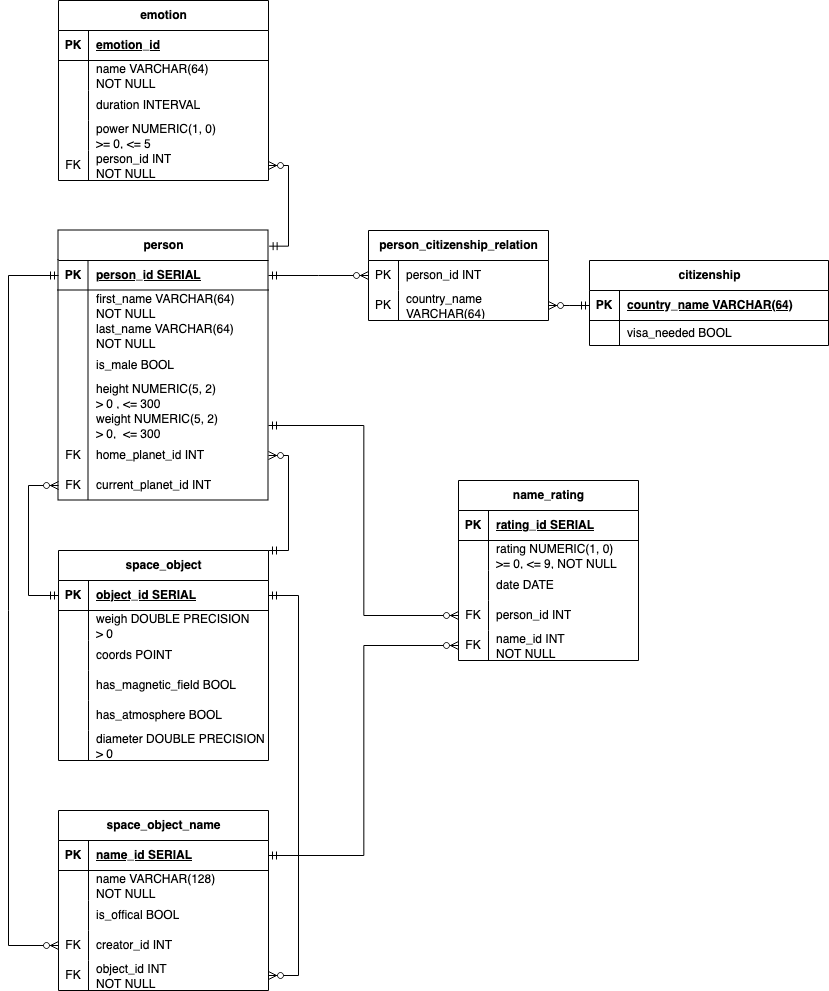
(country\_name) -> (visa\_needed)

* **person\_citizenship\_relation**: (person\_id, citizenship\_id) -> ()
* **space\_object**: (object\_id) -> (weight, coords, has\_magnetic\_field, has\_atmosphere, diameter)
* **space\_object\_name**: (name\_id) -> (name, is\_official, creator\_id, object\_id)
* **name\_rating**: (rating\_id) -> (rating, date, person\_id, name\_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

**visa\_needed** зависит от **country\_name** в таблице **citizenship.**

****

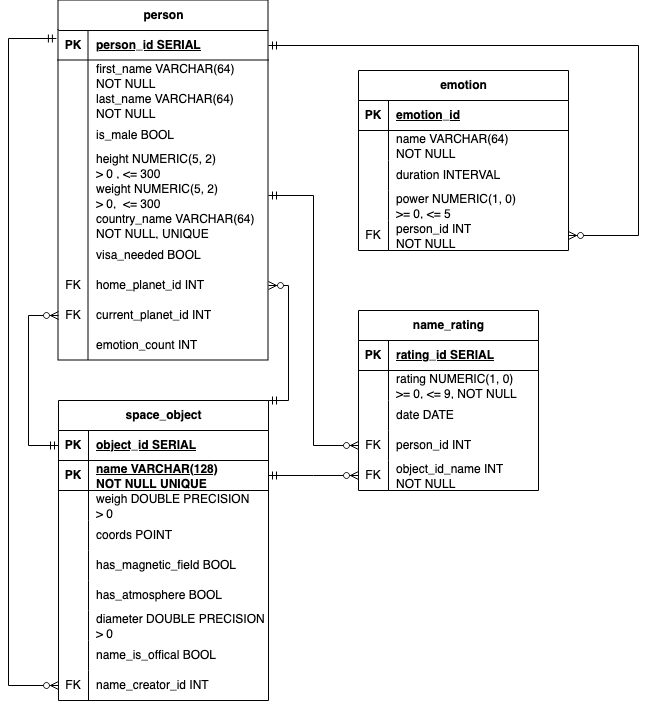
* **emotion**: (emotion\_id) -> (name, duration, power, person\_id)
* **person**: (person\_id) -> (first\_name, last\_name, is\_male, height, weight, home\_planet\_id, current\_planet\_id)
* **citizenship**: (country\_name) -> (visa\_needed)
* **person\_citizenship\_relation**: (person\_id, country\_name) -> ()
* **space\_object**: (object\_id) -> (weight, coords, has\_magnetic\_field, has\_atmosphere, diameter)
* **space\_object\_name**: (name\_id) -> (name, is\_official, creator\_id, object\_id)
* **name\_rating**: (rating\_id) -> (rating, date, person\_id, name\_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 **space\_object\_name** и **space\_object** в случае, если часто запрашиваются космические объекты по именам (нарушает 2NF). Объединение **person** с таблицей **citizenship** для избежания джойнов при выборке данных о гражданстве человека (нарушает 2NF).
* **Добавление избыточных атрибутов:** В некоторых случаях можно улучшить производительность благодаря добавлению избыточных атрибутов. Например добавить атрибут **emotion\_count** в **person** для подсчета количества чувств которые человек чувствует сейчас (нарушает 2NF).



* **person**: (person\_id) -> (first\_name, last\_name, is\_male, height, weight, home\_planet\_id, current\_planet\_id, country\_name, visa\_needed, emotion\_count)

(country\_name) -> (visa\_needed)

* **space\_object**: (object\_id) -> (weight, coords, has\_magnetic\_field, has\_atmosphere, diameter)

(name) -> (name\_is\_official, name\_creator\_id)

* **name\_rating**: (rating\_id) -> (rating, date, person\_id, object\_id\_name)

**Триггер**

Триггер при добавлении новой оценки имени обновляется рейтинг самых активых оценщиков имен. Если человек попадает в топ-10, то у него добавляется эмоции “Счастье” на 1 день. Если после добавление записей человек пропадает из топа, то у него добавляется эмоция “Тильт” на неделю. Для хранения топа используется отдельная табличка.

-- Триггер для обновления топа оценщиков и добавления эмоций

CREATE TABLE top\_name\_raters (

person\_id INT PRIMARY KEY,

rating\_count INT NOT NULL

);

CREATE OR REPLACE FUNCTION update\_top\_name\_raters()

RETURNS TRIGGER AS $$

DECLARE

current\_count INT;

new\_top\_count INT;

person\_id\_to\_delete INT;

BEGIN

-- Увеличиваем счетчик оценок для человека

SELECT COUNT(\*) INTO current\_count FROM name\_rating WHERE person\_id = NEW.person\_id;

-- Если человек уже в топе, обновляем его счетчик

IF EXISTS (SELECT 1 FROM top\_name\_raters WHERE person\_id = NEW.person\_id) THEN

UPDATE top\_name\_raters SET rating\_count = current\_count WHERE person\_id = NEW.person\_id;

ELSE

-- Вставляем новый счетчик, если человека нет в топе

INSERT INTO top\_name\_raters (person\_id, rating\_count) VALUES (NEW.person\_id, current\_count);

INSERT INTO emotion (name, duration, power, person\_id) VALUES ('Счастье', INTERVAL '1 day', 5, NEW.person\_id);

RAISE NOTICE 'Человек с id=% добавлен в топ-10, он счастлив 1 день', NEW.person\_id;

END IF;

-- Получаем айди человека, который вылетает из топа

SELECT person\_id FROM top\_name\_raters ORDER BY (rating\_count, person\_id) DESC OFFSET 11 LIMIT 1 INTO person\_id\_to\_delete;

-- Если человек существует, то удаляем его

IF person\_id\_to\_delete IS NOT NULL THEN

RAISE NOTICE 'Человек с id=% удален из топа, он впал в тильт на 7 дней', person\_id\_to\_delete;

RAISE NOTICE ' ';

DELETE FROM top\_name\_raters WHERE person\_id = person\_id\_to\_delete;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_update\_top\_name\_raters

AFTER INSERT ON name\_rating

FOR EACH ROW

EXECUTE FUNCTION update\_top\_name\_raters(); **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.