Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине

‘Базы данных’

Вариант №313121

*Выполнил:*

Студент группы P3131

Дворкин Борис Александрович

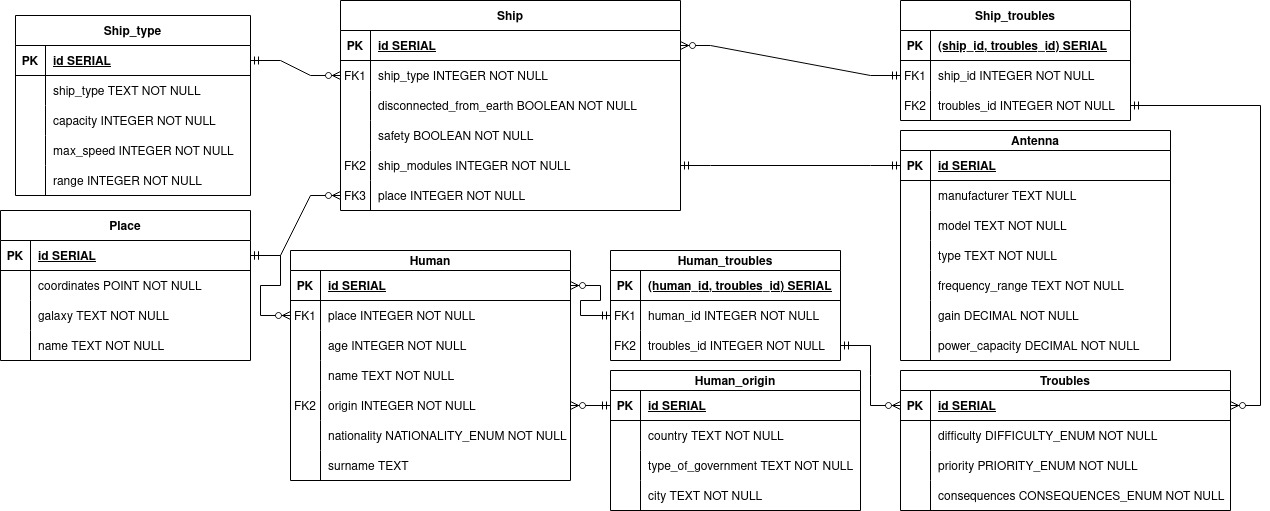
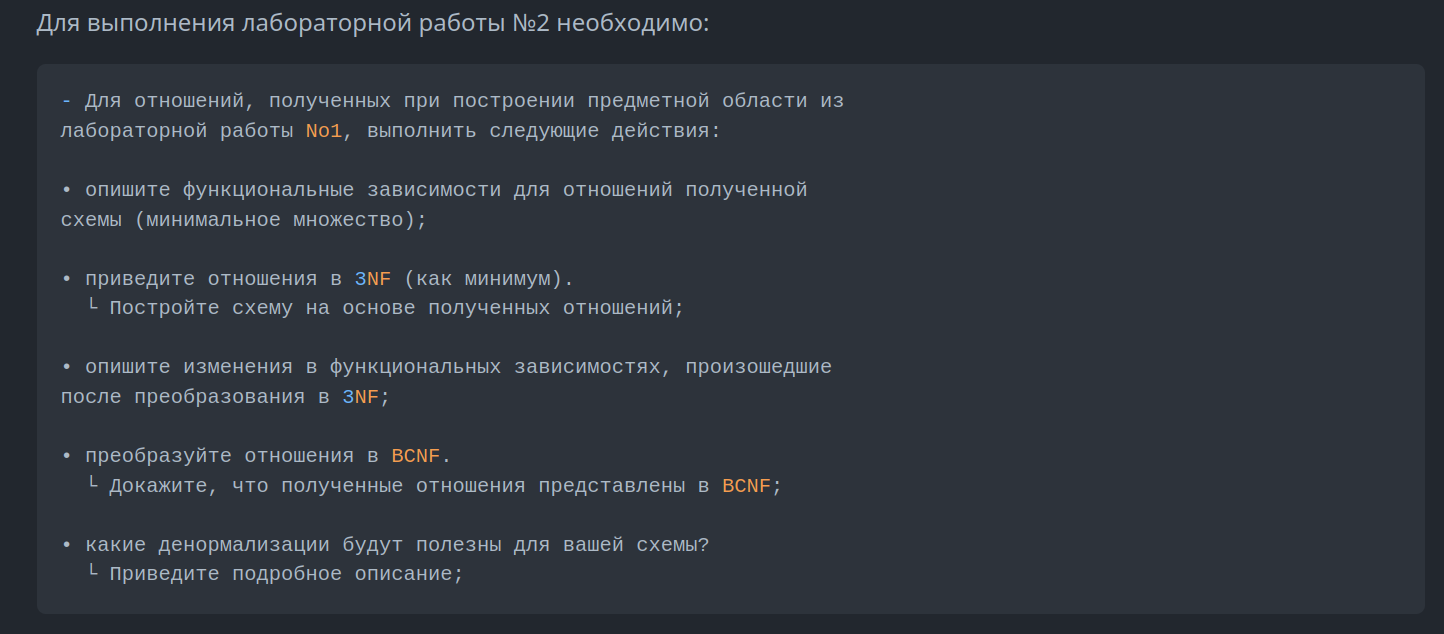
*Преподаватель:*

Наумова Надежда Александровна



Санкт-Петербург, 2023

1. **Текст задания**



1. **Функциональные зависимости**

|  |
| --- |
| antenna: id → (manufacturer, model, type, frequency\_range, gain, power\_capacity)    troubles: id → (difficulty, consequences, priority)    place: id → (coordinates, galaxy, name)    ship\_type: id → (ship\_type, ship\_capacity, max\_speed, range)    ship: id → (disconnected\_from\_earth, safety, ship\_modules, place, ship\_type)    human\_origin: id → (country, type\_of\_government, city)    human: id → (name, surname, age, nationality, origin, place)    human\_troubles: (human\_id, troubles\_id) → ()    ship\_troubles: (ship\_id, troubles\_id) → () |

**3. Нормальные формы**

|  |
| --- |
| 1NF: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. Моя модель удовлетворяет 1NF, так как все атрибуты атомарны, и нет повторяющихся групп.  2NF: Отношение находится во 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 удовлетворяет 3NF, так как все неключевые атрибуты зависят только от первичных ключей, и не содержат транзитивных зависимостей. |

**3. 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. |

**3. Денормализация**

|  |
| --- |
| **Объединение связанных таблиц**: В некоторых случаях, объединение таблиц может уменьшить количество операций JOIN и ускорить обработку запросов. Например, можно рассмотреть объединение таблиц human и human\_origin, если часто запрашиваются данные о человеке и его происхождении одновременно.  **Добавление избыточных атрибутов**: В некоторых случаях добавление избыточных атрибутов может улучшить производительность запросов. Например, если часто запрашивается количество кораблей определенного типа, можно добавить атрибут ship\_count в таблицу ship\_type. Это позволит избежать операций подсчета при каждом запросе, однако необходимо будет обновлять этот атрибут при добавлении или удалении кораблей. |

1. **Вывод**

При выполнении лабораторной работы я познакомился с понятием нормализации и денормализации. Научился определять функциональные зависимости модели, а также анализировать последнюю на соответствие различным нормальным формам. Изучил эффективные способы денормализации схемы базы данных и ситуации, в которых возможно их применение.

Доп.задание:

|  |
| --- |
| -- Доп: написать триггер, который будет выводить информацию об объекте,  -- который был добавлен/удалён в табличку ship\_type.  --    -- Удаляем существующие триггеры, если они существуют  DROP TRIGGER IF EXISTS ship\_type\_after\_insert ON ship\_type;  DROP TRIGGER IF EXISTS ship\_type\_before\_delete ON ship\_type;  DROP TRIGGER IF EXISTS ship\_type\_delete\_cascade\_trigger ON ship\_type;    -- Создаем функцию, которая вызывается при вставке записи в таблицу ship\_type  CREATE OR REPLACE FUNCTION ship\_type\_insert\_trigger() RETURNS TRIGGER AS $$  BEGIN  RAISE NOTICE 'INSERT: Ship Type with ID %, Type %, Capacity %, Max Speed %, and Range % has been added.', NEW.id, NEW.ship\_type, NEW.ship\_capacity, NEW.max\_speed, NEW.range;  RETURN NEW;  END;  $$ LANGUAGE plpgsql;    -- Создаем функцию, которая вызывается при удалении записи из таблицы ship\_type  CREATE OR REPLACE FUNCTION ship\_type\_delete\_trigger() RETURNS TRIGGER AS $$  BEGIN  RAISE NOTICE 'DELETE: Ship Type with ID %, Type %, Capacity %, Max Speed %, and Range % has been deleted.', OLD.id, OLD.ship\_type, OLD.ship\_capacity, OLD.max\_speed, OLD.range;  RETURN OLD;  END;  $$ LANGUAGE plpgsql;    -- Создаем триггер, который вызывает функцию ship\_type\_insert\_trigger при вставке записи в таблицу ship\_type  CREATE TRIGGER ship\_type\_after\_insert  AFTER INSERT ON ship\_type  FOR EACH ROW  EXECUTE FUNCTION ship\_type\_insert\_trigger();    -- Создаем триггер, который вызывает функцию ship\_type\_delete\_trigger при удалении записи из таблицы ship\_type  CREATE TRIGGER ship\_type\_before\_delete  BEFORE DELETE ON ship\_type  FOR EACH ROW  EXECUTE FUNCTION ship\_type\_delete\_trigger();    -- Создаем функцию, которая вызывается перед удалением записи из таблицы ship\_type  CREATE OR REPLACE FUNCTION ship\_type\_delete\_cascade() RETURNS TRIGGER AS $$  BEGIN  DELETE FROM ship\_troubles WHERE ship\_id IN (SELECT id FROM ship WHERE ship\_type = OLD.id);  DELETE FROM ship WHERE ship\_type = OLD.id;  RETURN OLD;  END;  $$ LANGUAGE plpgsql;    -- Создаем триггер, который вызывает функцию ship\_type\_delete\_cascade при удалении записи из таблицы ship\_type  CREATE TRIGGER ship\_type\_delete\_cascade\_trigger  BEFORE DELETE ON ship\_type  FOR EACH ROW  EXECUTE FUNCTION ship\_type\_delete\_cascade();    ---------------------------------------------------------------------------------------------    -- Давайте покажем функциональность написанных триггеров и процедурных функций    INSERT INTO ship\_type (ship\_type, ship\_capacity, max\_speed, range)  VALUES ('Cargo', 100, 400, 3000),  ('Passenger', 200, 600, 5000);    DELETE FROM ship\_type WHERE id = 1; |