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
Валиев Руслан Новруз оглы
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.

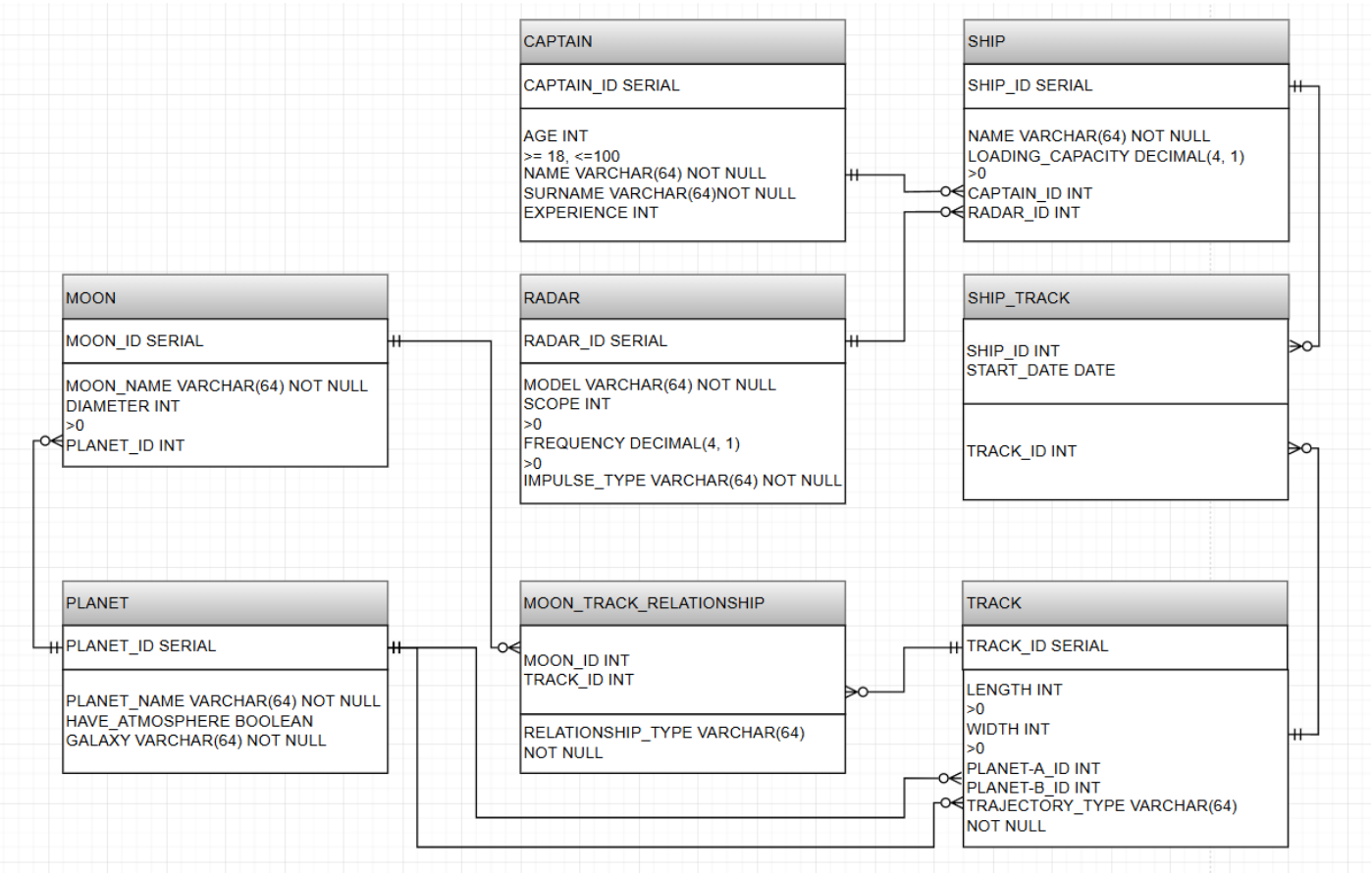
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

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

- **CAPTAIN**: CAPTAIN_ID → AGE, NAME, SURNAME, EXPERIENCE
- **RADAR**: RADAR_ID → MODEL, SCOPE, FREQUENCY, IMPULSE_TYPE
- **SHIP**: SHIP_ID → NAME, LOADING_CAPACITY, CAPTAIN_ID, RADAR_ID
- **SHIP_TRACK**: (SHIP_ID, START_DATE) → TRACK_ID
- **TRACK**: TRACK_ID → LENGTH, WIDTH, PLANET-A_ID, PLANET-B_ID, TRAJECTORY_TYPE
- **PLANET**: PLANET_ID → PLANET_NAME, HAVE_ATMOSPHERE, GALAXY
- **MOON_TRACK_RELATIONSHIP**: (MOON_ID, TRACK_ID) → RELATIONSHIP_TYPE
- **MOON:** MOON_ID → MOON_NAME, DIAMETER, PLANET_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 удовлетворяет 3NF так как во всех таблицах нет транзитивных зависимостей**.**

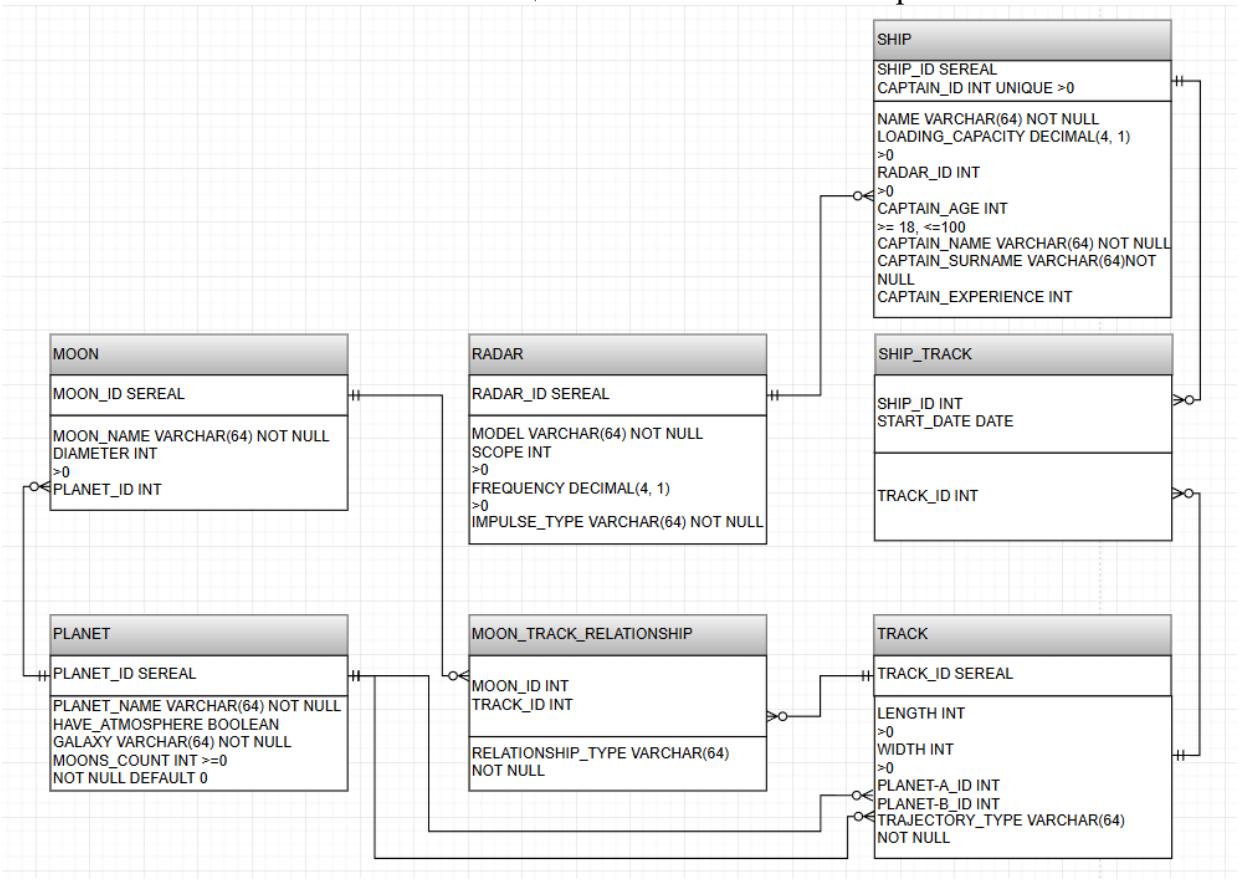
BCNF

- 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 **SHIP** и **CAPTAIN** так как вполне вероятно частое запрашивание капитанов по их кораблям (нарушает 2NF. Первичный ключ составной (SHIP_ID, CAPTAIN_ID), но атрибуты капитана зависят только от CAPTAIN_ID).

- **Добавление избыточных атрибутов:** в некоторых случаях можно улучшить производительность благодаря добавлению избыточных атрибутов. Например добавить атрибут **MOONS_COUNT** в **PLANET** для подсчета количества спутников для каждой планеты (нарушает 3NF так как автоматически обновляется триггерами на основе количества записей в таблице **MOON** и появляется транзитивная зависимость).



- **SHIP**: SHIP_ID → NAME, LOADING_CAPACITY, RADAR_ID
  CAPTAIN_ID → AGE, NAME, SURNAME, EXPERIENCE

- **PLANET:** PLANET_ID → PLANET_NAME, HAVE_ATMOSPHERE, GALAXY
  PLANET_ID → MOON_ID (через внешний ключ в MOON) → MOONS_COUNT

Триггер

В нашей денормализованной базе данных можно создать триггер, чтобы для каждой

планеты автоматически изменялось значение атрибута MOON_COUNT при добавлении\удалении\изменении значения атрибута PLANET_ID объектов таблицы MOON

```
-- ФУНКЦИЯ ДЛЯ ОБНОВЛЕНИЯ СЧЁТЧИКА ЛУН
CREATE OR REPLACE FUNCTION UPDATE_PLANET_MOON_COUNT()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
BEGIN
   IF (TG_OP = 'INSERT') THEN
      UPDATE PLANET
      SET MOON_COUNT = MOON_COUNT + 1
      WHERE PLANET_ID = NEW.PLANET_ID;

   ELSIF (TG_OP = 'DELETE') THEN
      UPDATE PLANET
      SET MOON_COUNT = MOON_COUNT - 1
      WHERE PLANET_ID = OLD.PLANET_ID;

   ELSIF (TG_OP = 'UPDATE' AND NEW.PLANET_ID IS DISTINCT FROM
OLD.PLANET_ID) THEN
      UPDATE PLANET
      SET MOON_COUNT = MOON_COUNT - 1
      WHERE PLANET_ID = OLD.PLANET_ID;

      UPDATE PLANET
      SET MOON_COUNT = MOON_COUNT + 1
      WHERE PLANET_ID = NEW.PLANET_ID;
   END IF;

   RETURN NULL;
END;
$$;

-- ТРИГГЕР ДЛЯ ОПЕРАЦИИ INSERT
CREATE TRIGGER MOON_INSERT_TRIGGER
AFTER INSERT ON MOON
FOR EACH ROW
EXECUTE FUNCTION UPDATE_PLANET_MOON_COUNT();

-- ТРИГГЕР ДЛЯ ОПЕРАЦИИ DELETE
CREATE TRIGGER MOON_DELETE_TRIGGER
AFTER DELETE ON MOON
FOR EACH ROW
EXECUTE FUNCTION UPDATE_PLANET_MOON_COUNT();

-- ТРИГГЕР ДЛЯ ОПЕРАЦИИ UPDATE
CREATE TRIGGER MOON_UPDATE_TRIGGER
AFTER UPDATE OF PLANET_ID ON MOON
FOR EACH ROW
WHEN (NEW.PLANET_ID IS DISTINCT FROM OLD.PLANET_ID)
EXECUTE FUNCTION UPDATE_PLANET_MOON_COUNT();
```

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.