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.

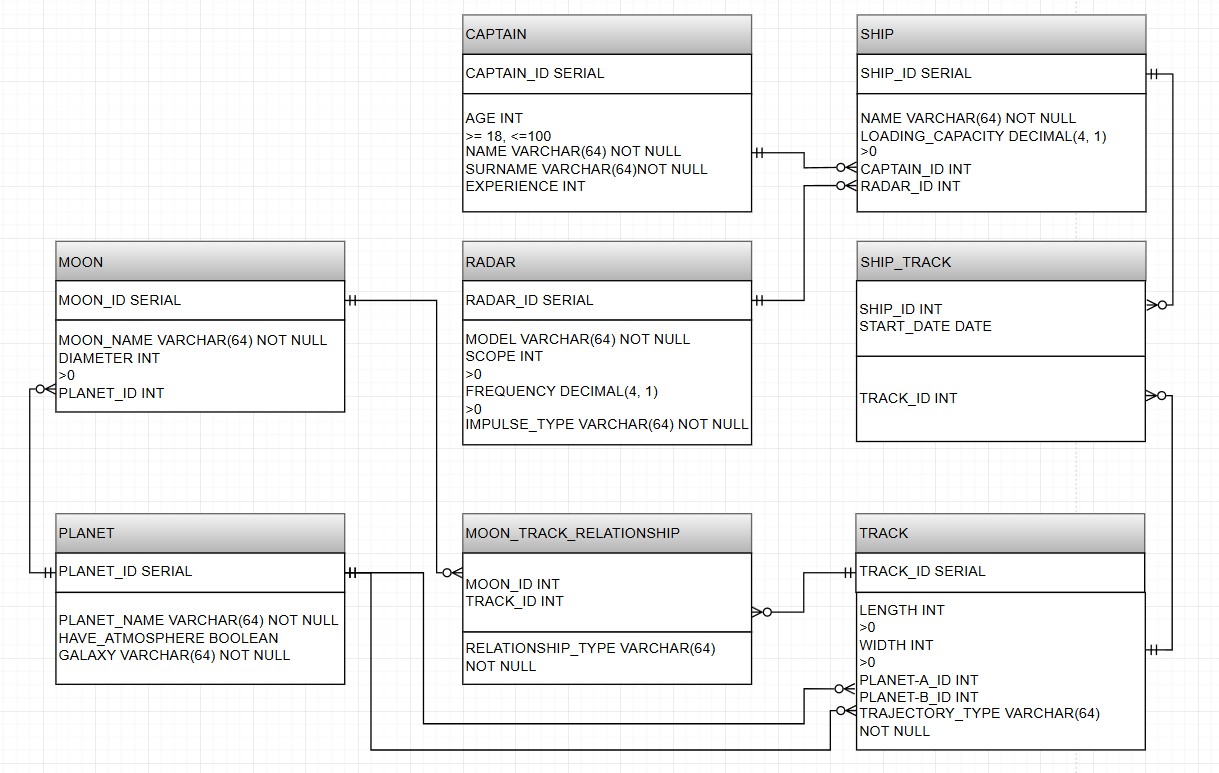
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

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

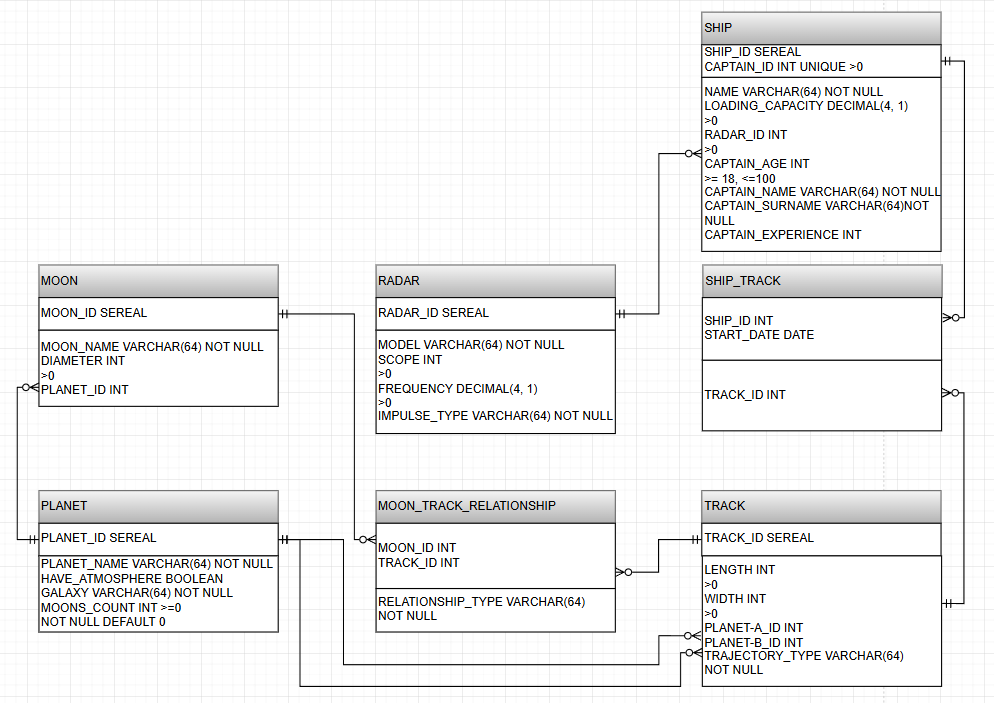
* + Отношение находится в BCNF, если для каждой функциональной зависимости X →Y, X является суперключом. Моя модель удовлетворяет BCNF, так как для всех функциональных зависимостей X является суперключом.

# 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.