**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет ПИиКТ**

**Дисциплина: Базы данных**

**Лабораторная работа №3**

**Нормализации и триггер**

**Вариант 312118**

Выполнил: Михайлов Петр Сергеевич

Группа: Р3111

Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург 2025г.

Содержание

[Текст задания 3](#_Toc198643384)

[Выполнение нормализации 4](#_Toc198643385)

[Исходная модель 4](#_Toc198643386)

[Функциональные зависимости 4](#_Toc198643387)

[Анализ зависимостей 4](#_Toc198643388)

[1NF 4](#_Toc198643389)

[2NF 5](#_Toc198643390)

[3NF 5](#_Toc198643391)

[BCNF 5](#_Toc198643392)

[Денормализация 6](#_Toc198643393)

[Триггер и связанная функция на языке PL/pgSQL 7](#_Toc198643394)

[Выводы по работе 9](#_Toc198643395)

# Текст задания

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:

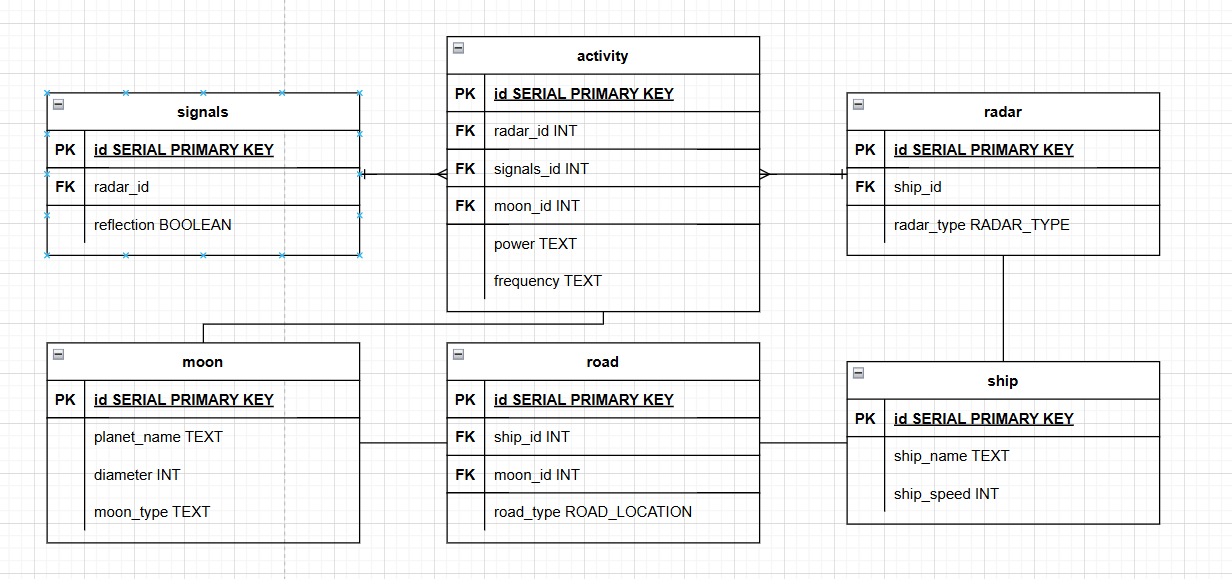
1. Текст задания.
2. Исходная, нормализованная и денормализованная модели.
3. Ответы на вопросы, представленные в задании.
4. Функция и триггер на языке PL/pgSQL
5. Выводы по работе.

Темы для подготовки к защите лабораторной работы:

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

# Выполнение нормализации

Исходная модель



*рис. 1. Исходная даталогическая модель*

Функциональные зависимости

ship: id → (ship\_name, ship\_speed),

road: id → (ship\_id, moon\_id, road\_type),

moon: id → (planet\_name, diameter, moon\_type),

radar: id → (ship\_id, radar\_type),

activity: id → (radar\_id, signals\_id, moon\_id, power, frequency),

signals: id → (radar\_id, reflection).

Анализ зависимостей

1NF

* 1. Отношение, на пересечении каждой строки и столбца — одно значение.

- Таблица удовлетворяет условия 1NF.

2NF

1. Таблица в 1NF;
2. Атрибуты, не входящие в первичный ключ, в полной функциональной зависимости от первичного ключа отношения.

- Таблица удовлетворяет 2NF (я не вижу способов выделить новые сущности, чтобы отразить полную функциональную зависимость).

3NF

1. Таблица в 1NF и 2NF;
2. Все атрибуты, которые не входят в первичный ключ, не находятся в транзитивной функциональной зависимости от первичного ключа.

- Таблица не содержит транзитивных связей, а значит, она удовлетворяет 3NF.

BCNF

1. Таблица 3NF;
2. Каждый неключевой атрибут должен зависеть только от всего ключа таблицы, а не от части или других полей.

- Во всех таблицах (кроме m2m) первичный ключ состоит из одного элемента, что автоматически приводит 3NF в BCNF.

# Денормализация

Полезность денормализации зависит от того, какие запросы на чтение данных чаще всего будут поступать в модель. Следующие денормализации могут быть полезны для уменьшения количества запросов:

1. Денормализация частоиспользуемых атрибутов
   * radar 🡨 ship\_name
   * signals 🡨 radar\_type
   * road 🡨 planet\_name, moon\_type
   * activity 🡨 ship\_name
2. Объединение таблиц для уменьшения соединений
   * radar\_signals (объединение таблиц radar и signals)
   * ship\_with\_radars (объединение ship и информации о его радарах)
   * moon\_details (объединение moon и statistics о путях к ней)
3. Дублирование внешних ключей для прямого доступа
   * activity 🡨 ship\_id (чтобы избежать соединения через radar)
   * signals 🡨 moon\_id (если часто нужна связь сигналов с конкретными лунами)
4. Материализованные представления
   * radar\_activity\_view (представление всей активности радаров с информацией о кораблях)
   * ship\_route\_view (представление всех маршрутов кораблей с деталями о лунах)
5. Агрегирующие поля
   * ship 🡨 radar\_count (количество радаров на корабле)
   * moon 🡨 road\_count (количество маршрутов к луне)
   * radar 🡨 signals\_count (количество сигналов с радара)

Выбор конкретных денормализаций должен основываться на анализе наиболее частых запросов к базе данных и потенциальных узких мест производительности.

# Триггер и связанная функция на языке PL/pgSQL

Триггер, который будет автоматически регистрировать все случаи обнаружения отраженного сигнала от луны. Это позволит вести журнал обнаружений для дальнейшего анализа и мониторинга активности.

Содержание файла trigger.sql:

-- Создание таблицы для журнала обнаружений

CREATE TABLE reflection\_log (

id SERIAL PRIMARY KEY,

radar\_id INT NOT NULL,

ship\_id INT NOT NULL,

moon\_id INT,

detected\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

signal\_power TEXT,

signal\_frequency TEXT,

description TEXT

);

-- Создание функции триггера

CREATE OR REPLACE FUNCTION log\_reflection\_detection()

RETURNS TRIGGER AS $$

DECLARE

v\_ship\_id INT;

v\_radar\_type TEXT;

v\_moon\_id INT;

v\_power TEXT;

v\_frequency TEXT;

v\_description TEXT;

BEGIN

IF NEW.reflection = TRUE THEN

SELECT r.ship\_id, r.radar\_type::TEXT INTO v\_ship\_id, v\_radar\_type

FROM radar r

WHERE r.id = NEW.radar\_id;

SELECT a.moon\_id, a.power, a.frequency INTO v\_moon\_id, v\_power, v\_frequency

FROM activity a

WHERE a.signals\_id = NEW.id

LIMIT 1;

v\_description := 'Радар типа "' || v\_radar\_type ||

'" обнаружил отраженный сигнал';

IF v\_moon\_id IS NOT NULL THEN

v\_description := v\_description || ' от луны с ID ' || v\_moon\_id;

END IF;

INSERT INTO reflection\_log (radar\_id, ship\_id, moon\_id, signal\_power,

signal\_frequency, description)

VALUES (NEW.radar\_id, v\_ship\_id, v\_moon\_id, v\_power, v\_frequency, v\_description);

RAISE NOTICE 'Обнаружен отраженный сигнал от радара %', NEW.radar\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Создание триггера

CREATE TRIGGER reflection\_detection\_trigger

AFTER INSERT OR UPDATE ON signals

FOR EACH ROW

EXECUTE FUNCTION log\_reflection\_detection();

# Выводы по работе

В процессе выполнения лабораторной работы я познакомился с разновидностями нормальных форм при создании баз данных, научился определять их и приводить к ним, денормализировать таблицы, если это необходимо. Изучил приемы создания триггеров и функций на языке PL/pgSQL.