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: P3111
Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург 2025г.

Содержание

Текст задания 3
Выполнение нормализации 4
Исходная модель 4
Функциональные зависимости 4
Анализ зависимостей 4
1NF 4
2NF 5
3NF 5
BCNF 5
Денормализация 6
Триггер и связанная функция на языке PL/pgSQL 7
Выводы по работе 9

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

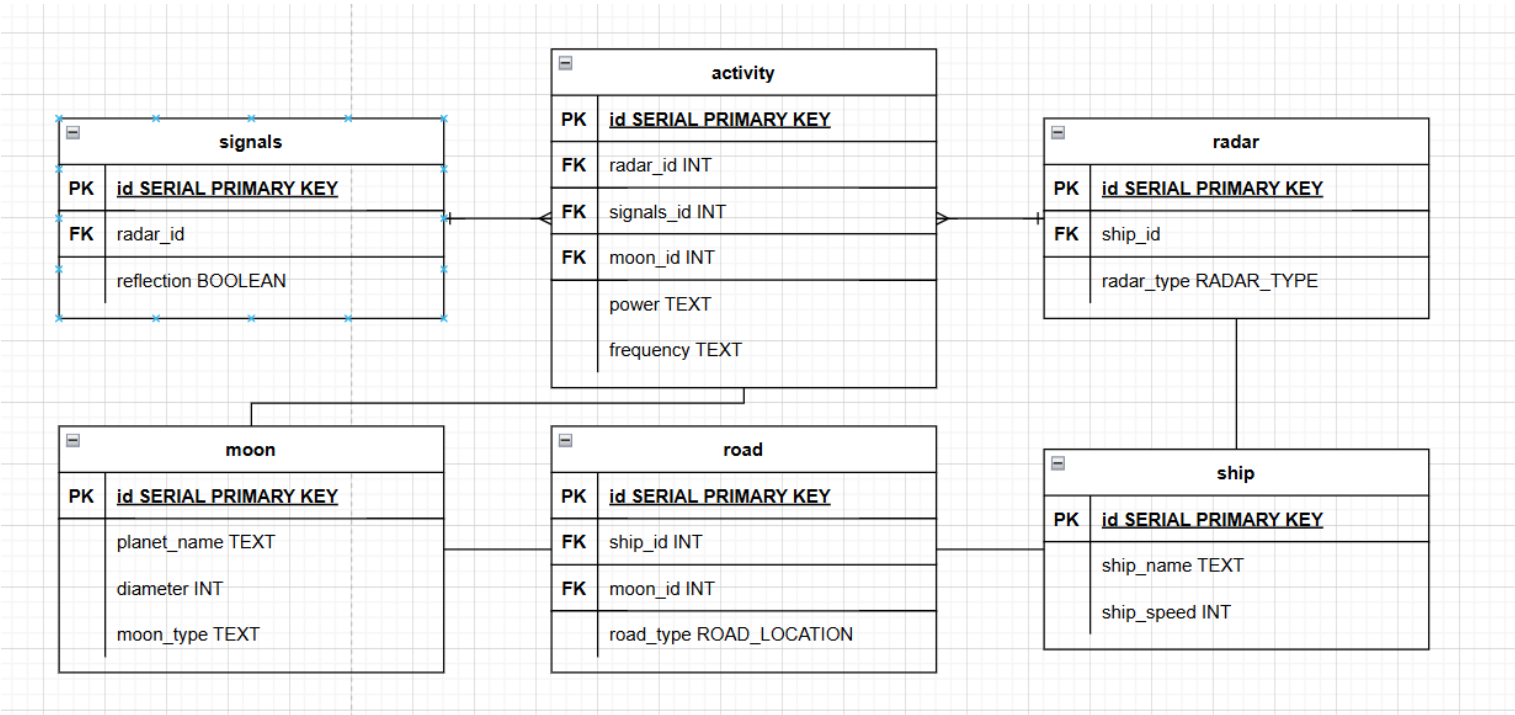

рис. 1. Исходная даталогическая модель

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.