Санкт-Петербургский Национальный Исследовательский Университет ИТМО
Факультет Программной Инженерии и Компьютерной Техники

Вариант №492541
Лабораторная работа №3
По дисциплине
Базы Данных

Выполнил студент группы P3117:
Изаак Герман Константинович

Преподаватель:
Чупанов Аликылыч Алибекович

Санкт-Петербург 2025 г.

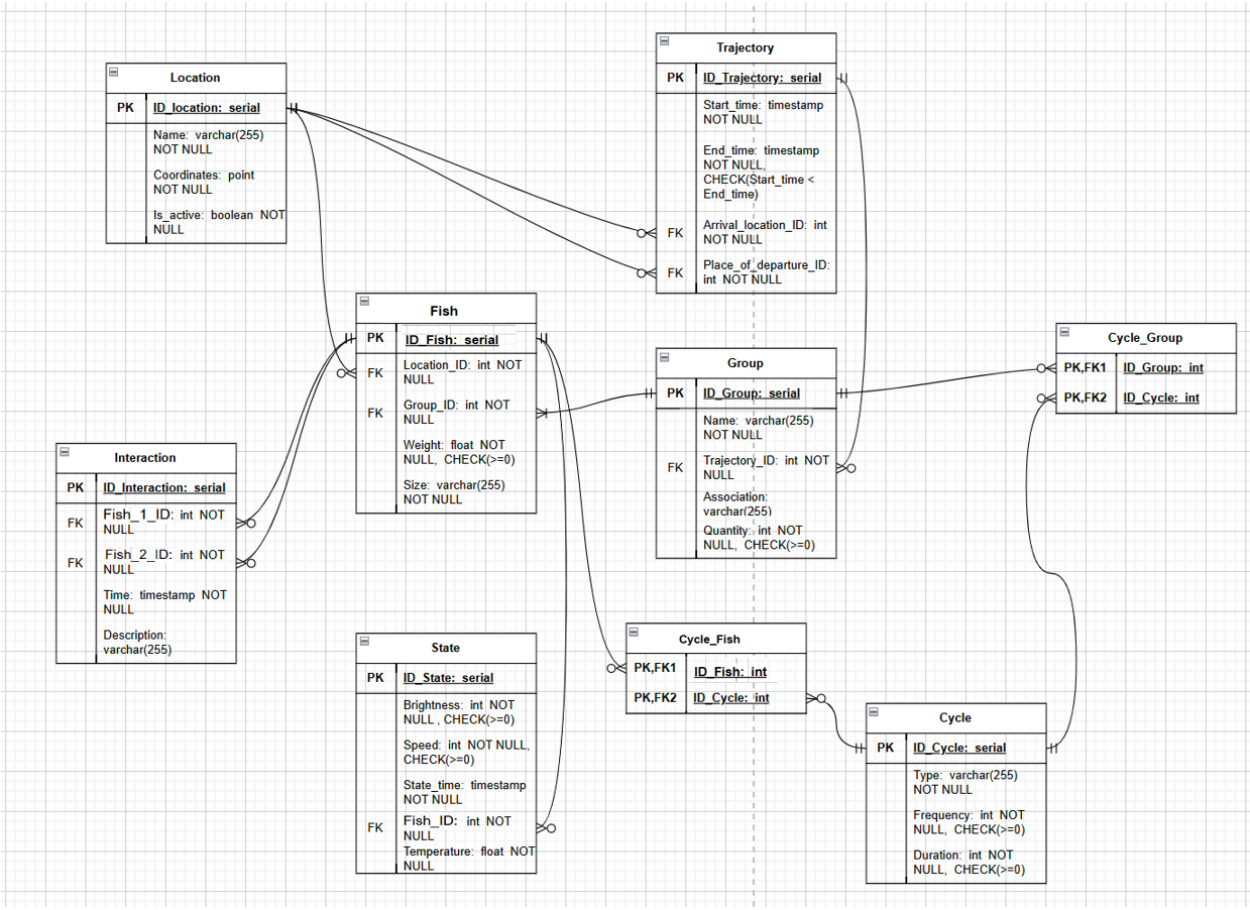
1. Текст задания

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.

2. Функциональные зависимости



Location: id_location → (name, coordinates, is_active)

Trajectory: id_trajectory → (start_time, end_time, arrival_location_id, place_of_departure_id)

Groups: id_group → (name, association, quantity, trajectory_id)

Fish: id_fish → (weight, size, location_id, group_id)

Interaction: id_interaction → (time, description, fish_1_id, fish_2_id)

State: id_state → (brightness, speed, state_time, temperature, fish_id)
Cycle: id_cycle → (type, frequency, duration)
Cycle_Group: (id_cycle, id_group) → ()
Cycle_Fish: (id_fish, id_cycle) → ()

3. Нормальные формы

1NF: Убедимся, что модель находится в 1NF. Это действительно так, потому что все поля являются атомарными и нет повторяющихся наборов столбцов.

2NF: Модель удовлетворяет 1NF и атрибуты, не входящие в первичный ключ, находятся в полной функциональной зависимости от первичного ключа. Единственные отношения с составными ключами — cycle_group(id_cycle, id_group) и cycle_fish(id_fish, id_cycle). В них нет других столбцов, поэтому частичных зависимостей не существует. Во всех остальных таблицах PK одинарный → частичные зависимости невозможны. Значит, модель находится в 2NF.

3NF: Модель удовлетворяет 3NF, так как неключевые столбцы никак не определяют друг друга внутри одной таблицы. Например: в groups атрибуты name, association, quantity, trajectory_id независимы друг от друга и зависят только от id_group. Аналогично для fish, state, trajectory, cycle и др. Следовательно транзитивных зависимостей нет.

4. BCNF

Все существующие функциональные зависимости детерминированы ключами, значит требования BCNF выполняются без дополнительной декомпозиции.

5. Денормализация

- **Координаты локации прямо в таблице fish**
  Где? Добавляем столбец coord POINT в fish.
  Как? ALTER TABLE fish ADD COLUMN coord POINT; + триггер, который при вставке/изменении location_id копирует location.coordinates в fish.coord.
  Зачем? Запросы «покажи всех рыб в радиусе R» больше не требуют JOIN с location.

- **Названия пунктов прямо в trajectory**

  Где? Добавляем столбцы departure_name VARCHAR(255) и arrival_name VARCHAR(255) в trajectory.

  Как? ALTER TABLE trajectory ADD COLUMN … + триггер BEFORE INSERT/UPDATE, который по place_of_departure_id и arrival_location_id подтягивает location.name и заносит копии.

  Зачем? Список маршрутов приходит без двух JOIN location, запросы становятся короче и работают быстрее.

- **Численность группы в самой таблице groups**

  Где? Поле quantity уже есть, нужно поддерживать его актуальным.

  Как? Триггер AFTER INSERT/DELETE ON fish увеличивает/уменьшает groups.quantity.

  Зачем? Отчёт «топ-10 самых больших групп» выполняется мгновенно — без подсчёта каждую секунду.

6. Триггер

Таблица **groups** уже содержит поле **quantity** (число рыб). Чтобы это значение всегда оставалось актуальным и не требовало отдельного UPDATE со стороны приложения, создадим триггер на таблицу **fish**, который автоматически корректирует groups.quantity при INSERT, DELETE и смене group_id у рыбы.

```
CREATE OR REPLACE FUNCTION trg_update_group_quantity()
    RETURNS TRIGGER AS
$$
BEGIN
    -- Added a new fish
    IF TG_OP = 'INSERT' THEN
        UPDATE groups
        SET    quantity = quantity + 1
        WHERE  id_group = NEW.group_id;

    -- The fish was removed
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE groups
        SET    quantity = quantity - 1
        WHERE  id_group = OLD.group_id;

    -- Changed the fish's group
    ELSIF TG_OP = 'UPDATE' THEN
        IF NEW.group_id <> OLD.group_id THEN
            UPDATE groups
            SET    quantity = quantity - 1
            WHERE  id_group = OLD.group_id;
```

```
            UPDATE groups
            SET    quantity = quantity + 1
            WHERE  id_group = NEW.group_id;
        END IF;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;


DROP TRIGGER IF EXISTS trg_fish_group_quantity ON fish;

CREATE TRIGGER trg_fish_group_quantity
    AFTER INSERT OR DELETE OR UPDATE OF group_id
    ON fish
    FOR EACH ROW
EXECUTE FUNCTION trg_update_group_quantity();
```

7. Вывод по работе

В ходе выполнения работы было установлено, что разработанная модель соответствует 3НФ и НФ Бойса–Кодда. Предложены денормализации для оптимизации частых запросов, включая кэширование координат и численности групп. Также реализован триггер на языке PL/pgSQL, автоматически поддерживающий актуальное количество рыб в каждой группе.