Санкт-Петербургский Национальный Исследовательский

Университет Информационных Технологий, Механики и Оптики

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3

по дисциплине

“База Данных”

Выполнил

Чэнь Хаолинь : 407960

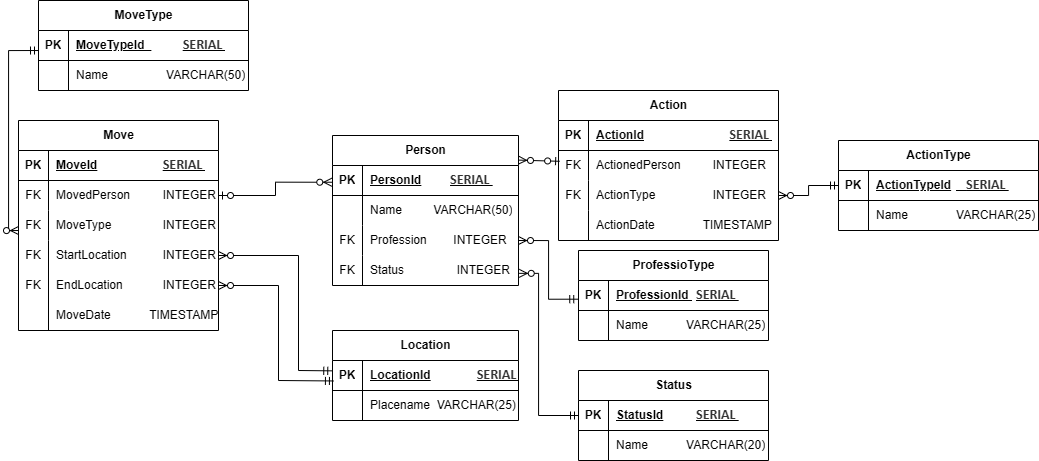
Студент группы P3116

**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.

**Даталогическая модель**

**Функциональные зависимости**

MoveType:

MoveTypeId → (Name)

Move:

MoveId → (MovedPerson, MoveType, StartLocation, EndLocation, MoveDate)

Person:

PersonId → (Name, Profession, Status)

Location:

LocationId → (Placename)

Action:

ActionId → (ActionedPerson, ActionType, ActionDate)

ActionType:

ActionTypeId → (Name)

Accession:

ProfessionalId → (Name)

Status:

StatusId → (Name)

**Нормальные формы**

**1NF**: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. Моя модель удовлетворяет 1NF, так как все атрибуты атомарны, и нет повторяющихся групп.

**2NF**: Отношение находится во 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 все неключевые атрибуты зависят только от первичных ключей, и не содержат транзитивных зависимостей.

**BCNF**

1. Каждая таблица имеет единственный потенциальный ключ  
   Все сущности используют SERIAL PK (MoveTypeId, MoveId, PersonId и т.д.), который является единственным идентификатором.
2. Для каждой функциональной зависимости X → Y:
   * X всегда является суперключом
   * Не существует зависимостей вида НеКлюч → Атрибут
3. Отсутствуют частичные и транзитивные зависимости

**Денормализация**

**Добавление избыточных атрибутов**: В некоторых случаях добавление избыточных атрибутов может улучшить производительность запросов. Например:

1.Если часто запрашивается количество Move, можно добавить атрибут TotalMoves в таблицу people.

2.Можно добавить дублирование Placename в таблице Move,если частые запросы на получение названий начальной и конечной локаций.

**Функция на языке PL/pgSQL**

При изменении статуса персонажа (например, с «Здоров» на «Лечится») в таблице действий автоматически создается запись «Изменение статуса» и фиксируется время, когда произошло изменение.

-- 1. Создание функции триггера

CREATE OR REPLACE FUNCTION log\_status\_change\_action()

RETURNS TRIGGER AS $$

DECLARE

    status\_change\_action\_id INTEGER;

BEGIN

    -- Получение ID типа действия 'Изменение статуса'

    SELECT ActionTypeId INTO status\_change\_action\_id

    FROM ActionType WHERE Name = 'Изменение статуса';

    -- Если статус изменен

    IF NEW.Status IS DISTINCT FROM OLD.Status THEN

        INSERT INTO Action (

            ActionedPerson,

            ActionType,

            ActionDate

        ) VALUES (

            NEW.PersonId,

            status\_change\_action\_id,

            NOW()

        );

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- 2. Создание триггера

CREATE TRIGGER person\_status\_change\_trigger

AFTER UPDATE OF Status ON Person

FOR EACH ROW

EXECUTE FUNCTION log\_status\_change\_action();

-- 3. Инициализация данных

INSERT INTO ActionType (Name)

VALUES ('Изменение статуса')

ON CONFLICT (Name) DO NOTHING;

-- Тип состояния инициализации

INSERT INTO Status (Name) VALUES ('Здоровый'), ('Лечение');

-- Создание тестового персонажа

INSERT INTO Person (Name, Status)

SELECT 'Флойд', StatusId FROM Status WHERE Name = 'Здоровый';

-- Обновление записи действия триггера статуса

UPDATE Person SET Status = (

    SELECT StatusId FROM Status WHERE Name = 'Лечение'

) WHERE Name = 'Флойд';

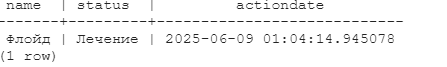
-- Проверьте результаты

SELECT p.Name, s.Name AS Status, a.ActionDate

FROM Person p

JOIN Status s ON s.StatusId = p.Status

JOIN Action a ON a.ActionedPerson = p.PersonId;

**Результат тестирования**

**Выводы по работе**

При выполнении лабораторной работы я познакомился с понятием нормализации и денормализации. Научился определять функциональные зависимости модели, а также анализировать последнюю на соответствие различным нормальным формам. Познакомился с процедурным языком PL/pgSQL. Изучил эффективные способы денормализации схемы базы данных и ситуации, в которых возможно их применение.