НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Программной инженерии и компьютерной техники

Базы данных

Лабораторная работа № 3

Выполнил студент:

Лабор Тимофей Владимирович

Группа № P3125

Преподаватель:

Бострикова Дарья Константиновна

г. Санкт-Петербург

2024

Оглавление

Вариант:

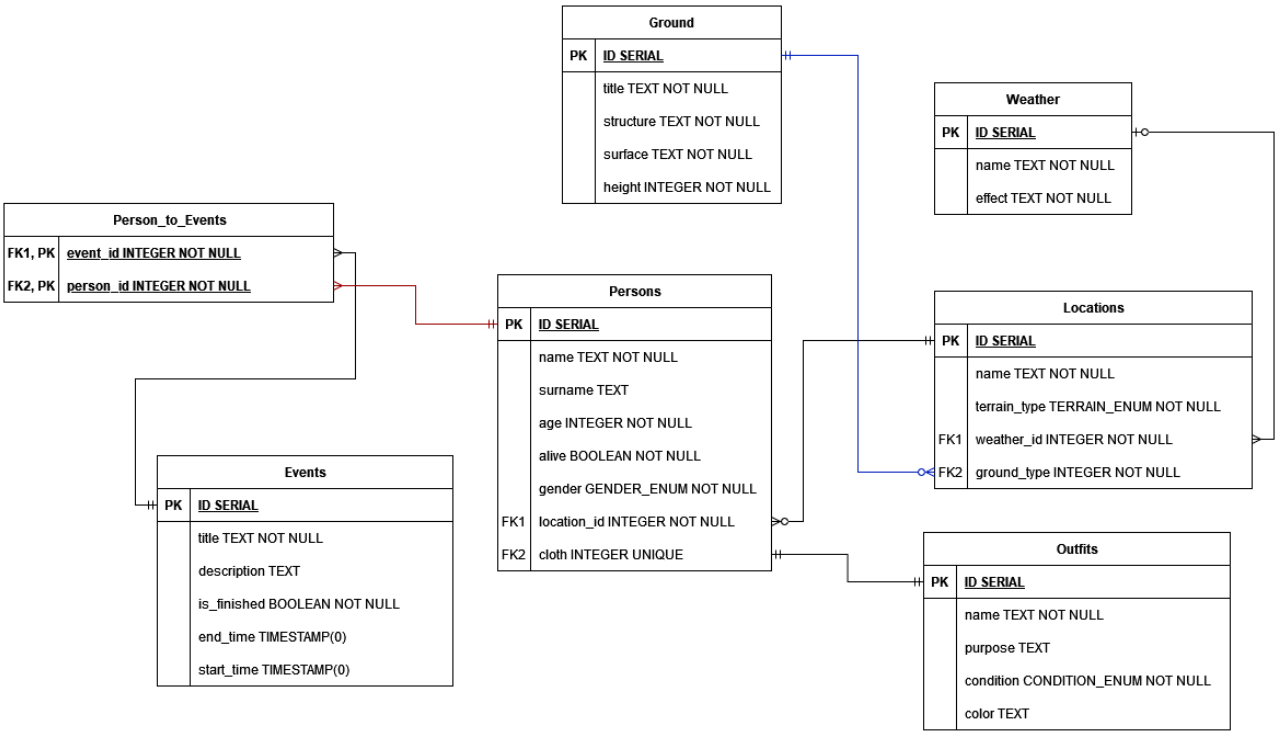
2530

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.



Отчет:

1. Функциональные зависимости

Locations: id → (name, terrain_type, weather_id, ground_type)
- многозначная, нетривиальная

Outfits: id → (name, purpose, condition, color)
- многозначная, нетривиальная

Weather: id → (name, effect)
- многозначная, нетривиальная

Ground: id → (title, structure, surface, height)
- многозначная, нетривиальная

Events: id → (title, description, is_finished, end_time, start_time)
- многозначная, нетривиальная

Persons: id → (name, surname, age, alive, gender, location_id, cloth)
- многозначная, нетривиальная

Persons_to_Events: (event_id, person_id) → ()
- нетривиальная

2. Нормальные формы

1 Нормальная форма (1НФ):

- на пересечении строки и столбца содержится только одно значение;
- наличие первичного ключа.

Модель уже соответствует условиям 1НФ.

Изменения не требуются

2 Нормальная форма (2НФ):

- отношения находятся в 1НФ;
- каждый атрибут, не входящий в первичный ключ, полностью функционально зависит от первичного ключа.

Чтобы привести к 2НФ необходимо убрать частичные зависимости от первичного ключа (т.е. в случае наличия составного первичного ключа, все атрибуты полностью зависят от ключа, а не только от его части).

В каждой таблице атрибуты, не включенные в первичный ключ, зависят только от него. Следовательно схема удовлетворяет условиям 2НФ.

- Locations: id → (name, terrain_type, weather_id, ground_type)
- Outfits: id → (name, purpose, condition, color)
- Weather: id → (name, effect)
- Ground: id → (title, structure, surface, height)
- Events: id → (title, description, is_finished, end_time, start_time)

- Persons: id → (name, surname, age, alive, gender, location_id, cloth)
- Persons_to_Events: (event_id, person_id) → ()

Изменения не требуются

3 Нормальная форма (3НФ):

- отношения находятся в 2НФ;
- нет атрибутов, не входящих в первичный ключ, которые находятся в транзитивной зависимости от первичного ключа.

Для того, чтобы доказать, что отношения находятся в 3НФ необходимо перебрать все возможные транзитивные зависимости.

Locations:

- id → name
- id → terrain_type
- id → weather_id
- id → ground_type

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты напрямую зависят от первичного ключа и только от него.

Outfits:

- id → name
- id → purpose
- id → condition
- id → color

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты напрямую зависят от первичного ключа и только от него.

Weather:

- id → name
- id → effect

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты напрямую зависят от первичного ключа и только от него.

Ground:

- id → title
- id → structure
- id → surface
- id → height

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты напрямую зависят от первичного ключа и только от него.

Events:

- id → title
- id → description

- id → is_finished
- id → start_time
- id → end_time

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты нпрямую зависят от первичного ключа и только от него.

Persons:

- id → name
- id → surname
- id → age
- id → gender
- id → alive
- id → cloth
- id → location_id

Атрибуты не зависят друг от друга, следовательно нет случая транзитивной зависимости, все атрибуты напрямую зависят от первичного ключа и только от него.

Нет ни одного отношения с транзитивными зависимостями, значит база данных находится в 3НФ.

Изменения не требуются

3. BCNF

Нормальная форма Бойса-Кодда:

- отношения находятся в 3НФ;
- каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

A1 → A2 , A2 → может быть первичным ключом, A1 — обязательно ключ. Ситуация, когда отношение будет находиться в 3НФ, но не в Бойса-Кодда, возникает, например, при условии, что отношение имеет два или более потенциальных ключа, которые являются составными, и между отдельными атрибутами таких ключей существует функциональная зависимость. Поскольку описанные зависимости не являются транзитивной, то такая ситуация под определение 3НФ не подпадает. На практике такие отношения встречаются достаточно редко, для всех прочих отношений 3НФ и Нормальная форма Бойса-Кодда эквивалентны.

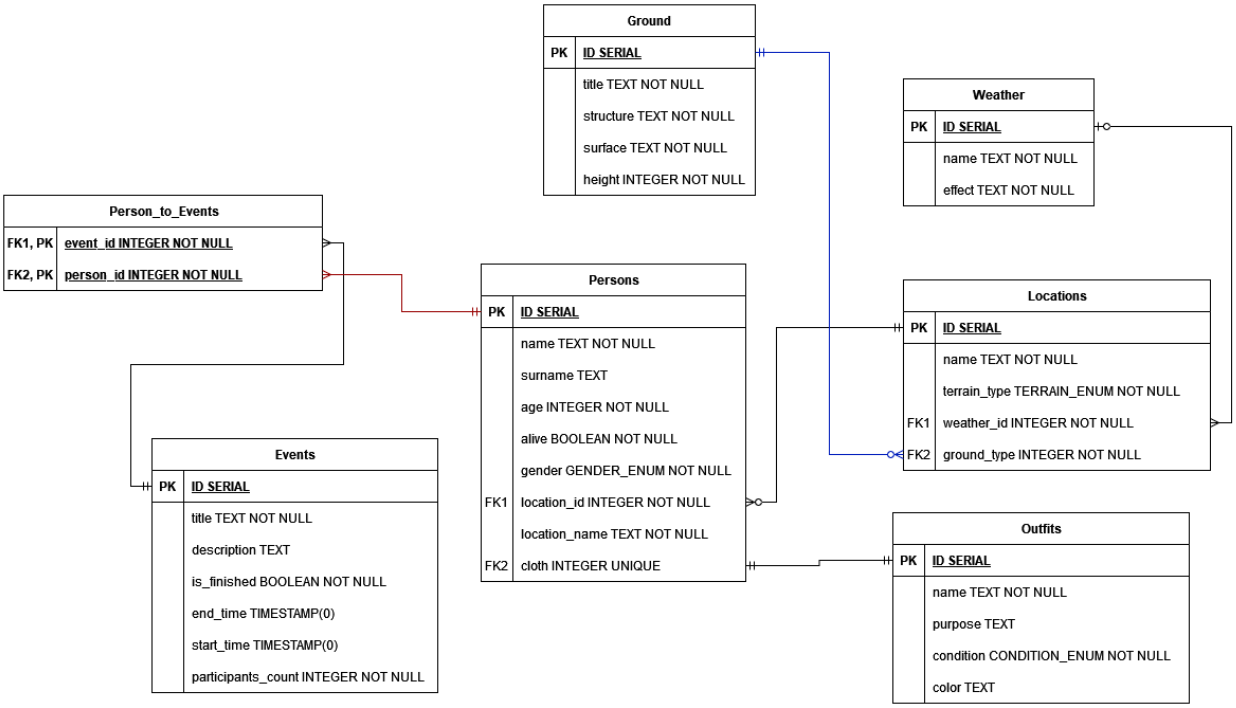
Каждый детерминант каждой таблицы является потенциальным ключем.

Изменения не требуются

4. Денормализация

Объединение связанных таблиц: в некоторых случаях, объединение таблиц может уменьшить количество операций JOIN и ускорить обработку запросов. Например, можно рассмотреть частичное объединение таблиц *persons* и *locations*, добавив в первую название локации, если оно часто запрашиваются.

Добавление избыточных атрибутов: в некоторых случаях добавление избыточных атрибутов может улучшить производительность запросов. Например, если часто запрашивается статистика по событиям, можно добавить атрибут *participants_count* в таблицу *events*. Это позволит избежать операций подсчета при каждом запросе, однако необходимо будет обновлять этот атрибут при добавлении или удалении записей в *persons_to_events*.



5. Функция на языке PL/pgSQL

Функция на языке PL/pgSQL.

```
-- триггер и связанная с ним функция, которая выводит для нового человека в таблице persons количество человек в его локации

-- Удаляем существующий триггер
DROP TRIGGER IF EXISTS persons_count_in_location_trigger on persons;


-- Создаем функцию, которая вызывается при вставке записи в таблицу spaceship
CREATE OR REPLACE FUNCTION persons_count_in_location() RETURNS TRIGGER AS $$
DECLARE
    persons_count int;
BEGIN
```

```
    SELECT COUNT(id) INTO persons_count FROM persons WHERE location_id =
NEW.location_id;

    RAISE NOTICE 'New Persons % % added in location % with % persons in it',
        NEW.name, NEW.surname, (SELECT name FROM locations WHERE id =
NEW.location_id), persons_count;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- Создаем триггер, который вызывает функцию persons_count_in_location при вставке
записи в таблицу persons
CREATE TRIGGER persons_count_in_location_trigger
    AFTER INSERT ON persons
    FOR EACH ROW
    EXECUTE FUNCTION persons_count_in_location();


-- Демонстрация функциональности
INSERT INTO
    persons(name, surname, age, gender, alive, location_id, cloth)
VALUES
    ('Vasek', 'Popov', 'male', True, 2, 3),
    ('Evsey', 'Petrov', 'male', True, 4, 2);

-- удаление демонстрационных данных
DELETE FROM persons WHERE name in ('Vasek', 'Evsey');
```

Вывод:

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