Факультет программной инженерной и компьютерной техники

Базы Данных

Лабораторная работа №3

Вариант 10015

Выполнил: Алхимовици Арсений 408138

Принял: Бострикова Дарья Константиновна

P3110

Санкт-Петербург, 2024

Текст Задания

4. DML

Введите вариант: 10015

Описание предметной области, по которой должна быть построена доменная модель:

> Эти черточки характера мальчика не слишком тревожили Джизирака. От Неповторимого вполне можно было ожидать именно такого вот поведения, но в должный срок Олвин конечно же воспримет существующий в городе образ жизни. Ни один индивидуум, как бы эксцентричен, как бы талантлив он ни был, не сумел бы оказать возмущающего влияния на колоссальную инерцию общества, которое оставалось неизменным на протяжении более чем миллиарда лет. Джизирак не просто свято верил в эту стабильность. Ничего иного он и помыслить себе не мог.

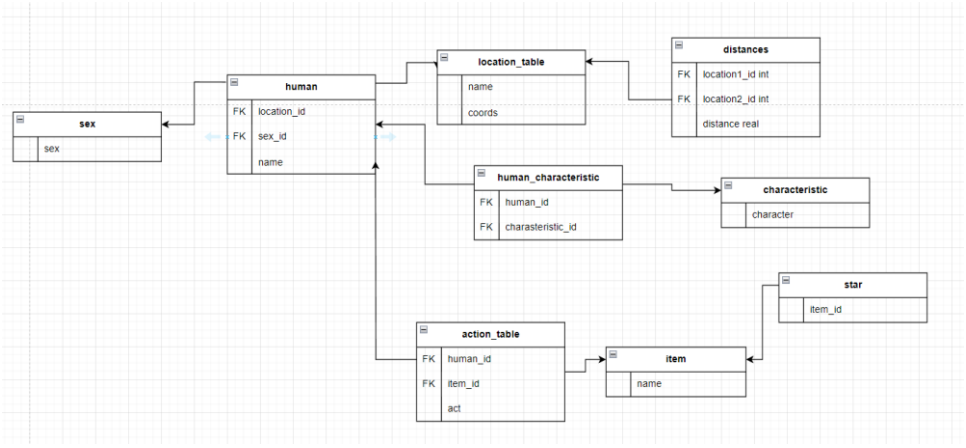
Лабораторная работа #2

Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

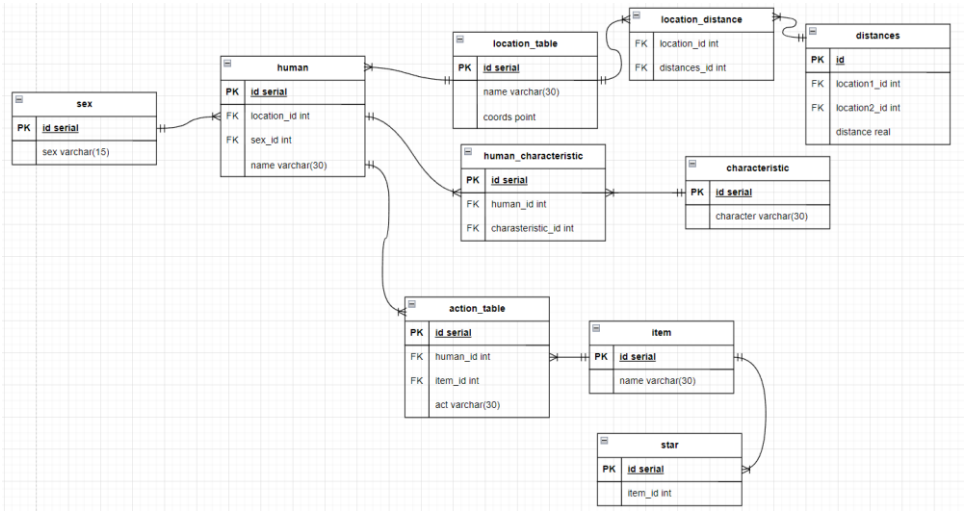
- Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
- Приведите отношения в 3NF (как минимум). Постройте схему на основеNF (как минимум).
- Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему 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.

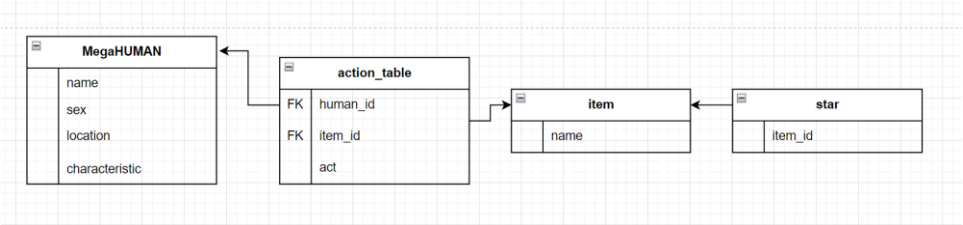
Нормализованная Инфологическая модель



Нормализованння Даталогическая модель



Денормализованная модель



Исходя из того, что кол-во JOIN-ов может быть еще больше, получение информации о человеке из отдельных нормализованных таблиц будет медленнее, чем то же действие из одной денормализованной таблицы, где будет хранится вообще вся информация о человеке, таким образом мы упростим обращение и обработку.

Функциональные зависимости

sex	id->	(sex)
location_table		(name)
human		(location_id, sex_id, name)
characteristic		(characteristic)
human_characteristic		(human_id, characteristic_id)
item		(name)
star		(item_id)
action_table		(human_id, item_id, act)

Нормальные формы

NF1: Отношения находятся в NF1, тк на пересечении каждой строки и столбца - одно значение
NF2: Отношения находятся в NF2, тк 1) отношение в NF1 и 2) атрибуты, не входящие в первичный ключ, в полной функциональной зависимости от первичного ключа отношения.
NF3: Отношения находятся в NF3, тк 1) отношения в NF2 и 2) все атрибуты, которые не входят в первичный ключ, не находятся в транзитивной функциональной зависимости от первичного ключа.
BCNF: Отношения находятся в BCNF, тк 1) отношения в NF3 и 2) все детерминанты это потенциальные ключи.

BCNF

Моя схема уже в BCNF, тк 1) отношения в NF3 и для любой функциональной зависимости X->Y, X – PRIMARY KEY.

Функция и триггер на языке PL/pgSQL

Триггер ждет изменения в таблице location_table и после совершения этих изменений вызывает функцию для каждой строки location_table. Функция в свою очередь считает расстояние между локациями, по итогу в таблицу distances приходят значения id локации 1, id локации 2 и расстояние между этими локациями.

```
--функция
CREATE OR REPLACE FUNCTION update_distances()
RETURNS TRIGGER AS $$
DECLARE
    id_1 int;
    id_2 int;
    location1_coords point;
    location2_coords point;
    x_1 real;
    x_2 real;
    y_1 real;
    y_2 real;
    distance real;
BEGIN
    FOR id_1 IN SELECT id FROM location_table LOOP
        FOR id_2 IN SELECT id FROM location_table WHERE id <> id_1 LOOP
            IF NOT EXISTS (
SELECT 1 FROM distances WHERE (distances.location1_id = id_1 AND
distances.location2_id = id_2)
OR (distances.location1_id = id_2 AND distances.location2_id = id_1)
            ) THEN
RAISE NOTICE 'подсчет расстояния между локациями % и %', id_1, id_2;
                SELECT coords INTO location1_coords FROM location_table
WHERE id = id_1;
                SELECT coords INTO location2_coords FROM location_table
WHERE id = id_2;
x_1 = location1_coords[0];
x_2 = location2_coords[0];
y_1 = location1_coords[1];
y_2 = location2_coords[1];
distance := SQRT((x_1-x_2)*(x_1-x_2)+(y_1-y_2)*(y_1-y_2));
```

```
            INSERT INTO distances (location1_id, location2_id, distance)
               VALUES (id_1, id_2, distance);
           END IF;
        END LOOP;
    END LOOP;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;


--тригер
CREATE OR REPLACE TRIGGER update_distances_trigger
AFTER INSERT OR UPDATE OR DELETE ON location_table
FOR EACH ROW EXECUTE FUNCTION update_distances();
```

Вывод:

В ходе данной лабораторной работы я научился работать с нормализацией, узнал для чего иногда нужно использовать денормализацию. Попрактиковался в создании таблиц. Научился делать функции и триггеры на языке PL/pgSQL.