Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант №31182

Лабораторная работа №3

По дисциплине

«Информационные системы и базы данных»

Выполнил студент группы P3118:

Богданова Мария Михайловна

Преподаватель:

          Инячина Диана Александровна

Санкт-Петербург

2023 г.

**Задание:**

Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:  
1.Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);  
2.Приведите отношения в 3NF (как минимум). Постройте схему на основе NF (как минимум). 3.Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3 NF (как минимум). Постройте схему на основеNF;  
4.Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это  
6. 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.

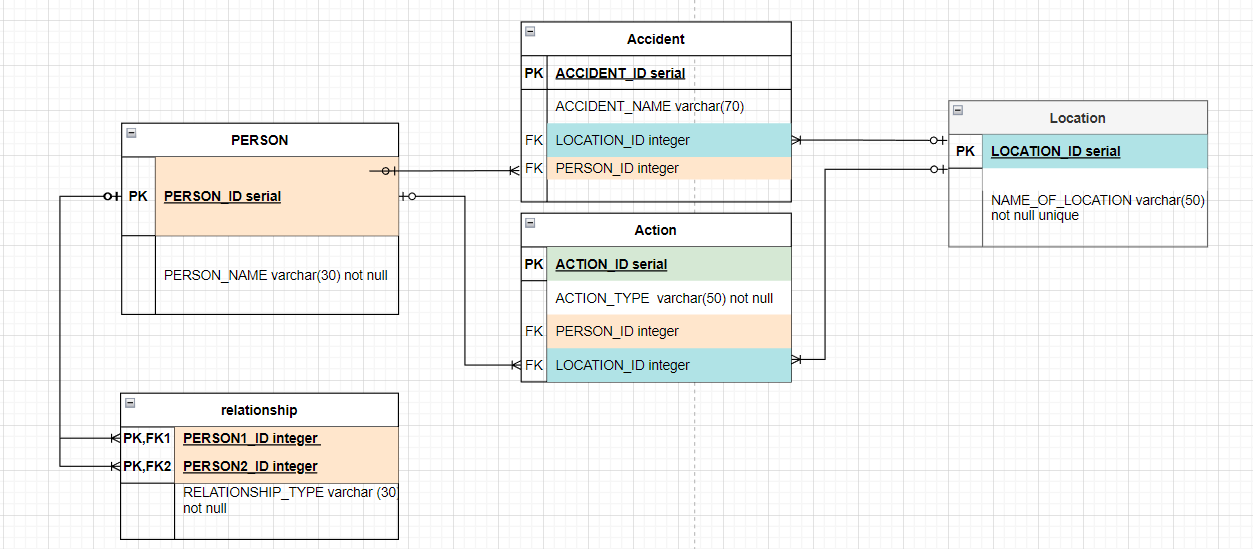
**Предметная область:**

Диаспар почти не видел Олвина в последующие несколько недель, хотя всего лишь какая-то горстка людей заметила его отсутствие. Джизирак, обнаружив, что его ученик, вместо того чтобы бродить в районе границ города, все свое время проводит в Зале Совета, испытал некоторое облегчение, ибо полагал, что уж там-то с Олвином никакой беды не приключится. Эристон и Итания раз-другой навестили его комнату, убедились, что сын отсутствует, и не придали этому значения. Что же касается Алистры, то она оказалась более настойчивой.

**Описание предметной области:**

Люди могут выполнять действия. Действия могут иметь локацию. С людьми может что-то произойти (инцидент). Инцидент может произойти в какой-то локации. Люди могут иметь отношения между собой. В локации может ничего не происходить.

**Исходная даталогическая модель:**



1. **Функциональные зависимости:**

Person:

person\_id -> person\_name (каждый person\_id однозначно определяет person\_name)

Locations:

location\_id -> location\_name (каждый location\_id однозначно определяет location\_name)

Accident:

accident\_id -> accident\_name (каждый accident\_id однозначно определяет accident\_name)

location\_id -> accident\_id (каждый location\_id может быть связан с несколькими accident\_id)

person\_id -> accident\_id (каждый person\_id может быть связан с несколькими accident\_id)

Action:

action\_id -> action\_type (каждый action\_id однозначно определяет action\_type)

person\_id -> action\_id (каждый person\_id может быть связан с несколькими action\_id)

location\_id -> action\_id (каждый location\_id может быть связан с несколькими action\_id)

Relationship:

relationship\_id -> relationship\_type (каждый relationship\_id однозначно определяет relationship\_type)

person1\_id -> relationship\_id (каждый person1\_id может быть связан с несколькими relationship\_id)

person2\_id -> relationship\_id (каждый person2\_id может быть связан с несколькими relationship\_id)

1. **Первая нормальная форма.**   
   Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. Почти все атрибуты в таблицах моей схемы являются атомарными и каждая ячейка содержит только одно значение. Однако, есть нарушение первой нормальной формы в отношении "Action", поскольку атрибут "person\_id" не является атомарными и может содержать несколько значений. Для того, чтобы привести схему в 1NF нам необходимо разделить:  
   1) отношение “Accident” на:  
    - Accident с атрибутами accident\_id (PK), accident\_name и location\_id (FK), предполагая, что один инцидент может произойти в одном месте  
   - Accident\_People с атрибутами accident\_id (FK) и people\_id(FK) из соображений, что один случай может произойти с несколькими людьми в один момент времени в одном месте.

2) отношение "Action" на:

- Action с атрибутами action\_id (PK), action\_type и location\_id (FK) из соображений, что одно действие может производиться только в одном месте  
 - Action\_people с атрибутами action\_id (FK) и people\_id (FK), рассматривая тот факт, что действие может производиться несколькими людьми совместно или одновременно.  
Остальные отношения можно оставить без изменений.

**Теперь схема находится в первой нормальной форме, поскольку ни одна из строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто**.

1. **Вторая нормальная форма.** Условие: таблица должна быть в первой нормальной форме и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.  
   В данной схеме таблицы все столбцы не ключевых атрибутов полностью зависят от первичного ключа соответствующей таблицы. Так, в таблице "Accident" атрибут "accident\_name" зависит от всего первичного ключа "accident\_id" и не зависит от атрибута "location\_id". Аналогично, в таблице "Action" атрибут "action\_type" зависит только от первичного ключа "action\_id" и не зависит от "location\_id". Таким образом, таблица находится во второй нормальной форме. Таблица "Accident\_people" содержит атрибуты "accident\_id" и "person\_id", которые оба являются внешними ключами, ссылаясь на другие таблицы. **Так как каждый атрибут полностью зависит от первичного ключа "accident\_id", таблица находится в 2НФ.**
2. **Третья нормальная форма.** Условие: схема удовлетворяет 2НФ и ни одно из ее неключевых полей не зависит функционально от любого другого неключевого поля. Неключевые атрибуты не должны зависеть друг от друга. В данной схеме все столбцы в каждой таблице зависят только от первичного ключа или от других столбцов, а неключевые атрибуты не зависят друг от друга.  
   В таблицах Person, Action, Location, Accident все столбцы являются атомарными, в них нет повторяющихся групп данных, что соответствует первой нормальной форме. Она также соответствует также и второй нормальной форме, потому что при составном первичном ключе, каждый неключевой атрибут зависит от всего ПК, а не от его части. В таблицах Action\_people и accident\_people столбцы action\_id и person\_id, accident\_id и person\_id соответственно являются ключами, и между ними нет транзитивной зависимости. **То есть, неключевые атрибуты не имеют транзитивных зависимостей от других неключевых атрибутов, и таблица находится в третьей нормальной форме.**

2) Схема:  
  
  
Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

3) Нормальная форма Бойса-Кодда:

Условие: любая функциональная зависимость между его полями сводится к полной функциональной зависимости от возможного ключа (каждый детерминант (атрибут, от которого полностью функционально зависит другой атрибут) - потенциальный ключ). Данная схема находится в третьей нормальной форме, однако для того, чтобы привести схему в БКНФ, нужно добавить составные первичные ключи для отношений action\_people и accident\_person. В таблицах Action, Accident, Location, Person и Relationship ни один из вторичных ключей не может стать исходным для других таблиц, в оставшихся таблицах же нет вторичных ключей вообще. Таблица находится в нормальной форме Бойса-Кодда (BCNF), т. к. все неключевые атрибуты функционально зависят только от ключа, то есть нет неключевых атрибутов, которые функционально зависят от других неключевых атрибутов.   
Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

4) Денормализация. В представленной схеме базы данных можно рассмотреть несколько потенциальных денормализаций:

1. Добавление атрибутов "person\_name" и "location\_name" в таблицы "Accident\_people" и "Action\_people", чтобы избежать лишних соединений (JOIN) с таблицами "Person" и "Locations" при выполнении запросов, связанных с этими связующими таблицами. Это может ускорить процесс получения информации о людях, связанных со случаями и действиями.  
2. Добавление атрибутов "person\_name" и "location\_name" в таблицы "Accident" и "Action", чтобы избежать необходимости объединения с таблицами "Person" и "Locations" при выполнении сложных запросов. Это может быть полезным, если часто требуется запращивать данные по инцидентам и действиям, включая информацию о связанных людях и местоположениях. Денормализация таблицы "Action": если требуется часто получать информацию о типе действия и локации, а также именах людей, связанных с действием, можно добавить дополнительные столбцы в таблицу "Action", содержащие дублирующую информацию из таблиц "Locations" и "Person". Это также может помочь упростить запросы.  
3. Добавление атрибута "person\_name" в таблицу "Relationship" для каждого из людей (person1\_name и person2\_name), чтобы избежать необходимости объединения с таблицей "Person" при поиске отношений между людьми. Это может быть полезным, если требуется быстрый доступ к информации об отношениях между людьми.  
  
5) Триггер и связанная с ним функция:

//Функция для обновления person\_name в таблицах accident\_people и action\_people.  
CREATE OR REPLACE FUNCTION update\_people\_names()   
RETURNS TRIGGER AS $$

BEGIN

IF NEW.person\_name <> OLD.person\_name THEN

UPDATE Accident\_people

SET person\_name = NEW.person\_name  
 WHERE person\_id = NEW.person\_id;

UPDATE Action\_people  
 SET person\_name = NEW.person\_name  
 WHERE person\_id = NEW.person\_id;

END IF;

RETURN NEW;

ENG;

$$ LANGUAGE plpsql;

//триггер на изменения в таблице person

CREATE TRIGGER update\_people\_name\_trigger  
AFTER UPDATE ON person  
FOR EACH ROW

EXECUTE FUNCTION update\_people\_names();  
  
  
  
psql -h pg -d studs

\i запуск файла

\d просмотр таблиц

Скрипт:  
postgreSQL

CREATE TABLE Person (

person\_id SERIAL PRIMARY KEY,

person\_name varchar(50) NOT NULL

);

CREATE TABLE Locations(

location\_id SERIAL PRIMARY KEY,

location\_name varchar(80) NOT NULL UNIQUE

);

REPLACE TABLE Accident(

accident\_id SERIAL PRIMARY KEY,

accident\_name varchar (50),

location\_id int REFERENCES Locations(location\_id),

);

CREATE TABLE Accident\_people(

accident\_id int REFERENCES Accident(accident\_id),

person\_id int REFERENCES Person(person\_id)

);

REPLACE TABLE Action (

action\_id SERIAL PRIMARY KEY,

action\_type varchar(70) NOT NULL,

location\_id int REFERENCES Locations(location\_id)

);

CREATE TABLE Action\_people(

action\_id int REFERENCES Action(action\_id),

person\_id int REFERENCES Person(person\_id)

);

CREATE TABLE Relationship(

relationship\_id SERIAL PRIMARY KEY,

relationship\_type varchar(30) NOT NULL,

person1\_id int REFERENCES Person(person\_id),

person2\_id int REFERENCES Person(person\_id),

check (person1\_id != person2\_id)

);  
CREATE OR REPLACE FUNCTION update\_people\_names()   
RETURNS TRIGGER AS $$

BEGIN

IF NEW.person\_name <> OLD.person\_name THEN

UPDATE Accident\_people

SET person\_name = NEW.person\_name  
 WHERE person\_id = NEW.person\_id;

UPDATE Action\_people  
 SET person\_name = NEW.person\_name  
 WHERE person\_id = NEW.person\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER update\_people\_name\_trigger  
AFTER UPDATE ON person  
FOR EACH ROW

EXECUTE FUNCTION update\_people\_names();  
  
Написать транзакцию:  
BEGIN;

UPDATE Person SET person\_name = 'Новое\_имя\_чел' WHERE person\_id = 1;  
  
--SAVEPOINT savepoint1;

UPDATE Action\_people SET action\_id = 2 WHERE person\_id = 1;

--ROLLBACK TO savepoint1;

COMMIT;