Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №3

По дисциплине «Базы данных» (второй семестр)

Студент:
Дениченко Александр P3112
Практик:
Лисицина В.В

Санкт-Петербург
2023 г.

1 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 основеNF (как минимум). Постройте схему на основе полученных отношений;

3. опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основеNF;

4. преобразуйте отношения в BCNF.

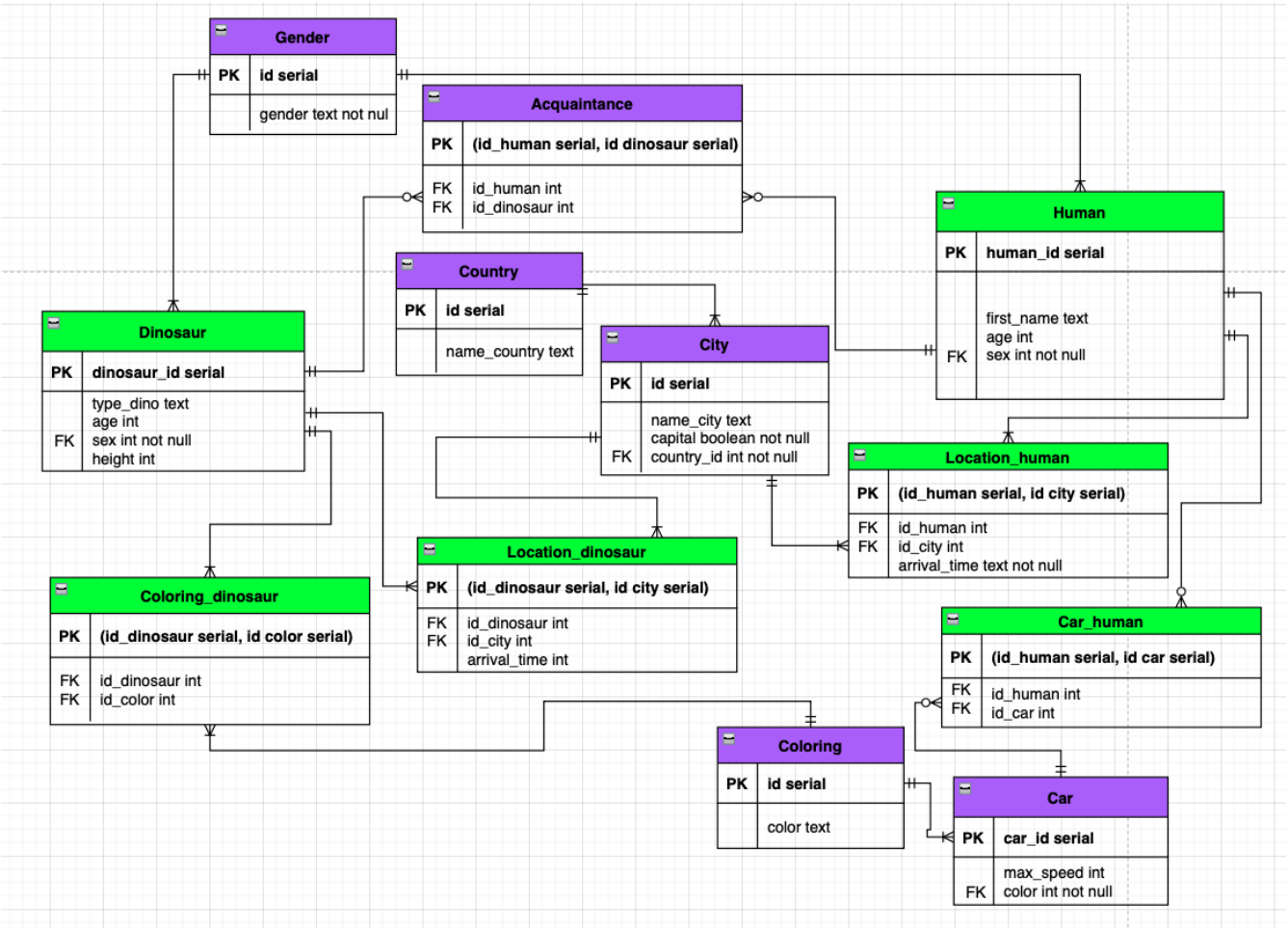
Докажите, что полученные отношения представлены в BCNF;

Если ваша схема находится уже в BCNF, докажите это.

Какие денормализации будут полезны для вашей схемы? Приведите подробное описание;

Придумайте функцию, связанную с вашей предметной областью, согласуйте ее с преподавателем и реализуйте на языке PL/pgSQL.

2 Исходная модель



Для каждой таблицы в представленной схеме можно определить следующие функциональные зависимости:

Таблица Coloring:
id -> color

Таблица Car:
id -> max speed
id -> color (FK)

Таблица Country:
id -> Name Country

Таблица City:
id -> name
id -> capital
id -> country (FK)

Таблица Gender:
id - gender

Таблица Dinosaur:
id -> type
id -> age
id -> sex (FK)
id -> height

Таблица Human:
id -> name
id -> age
id -> sex (FK)

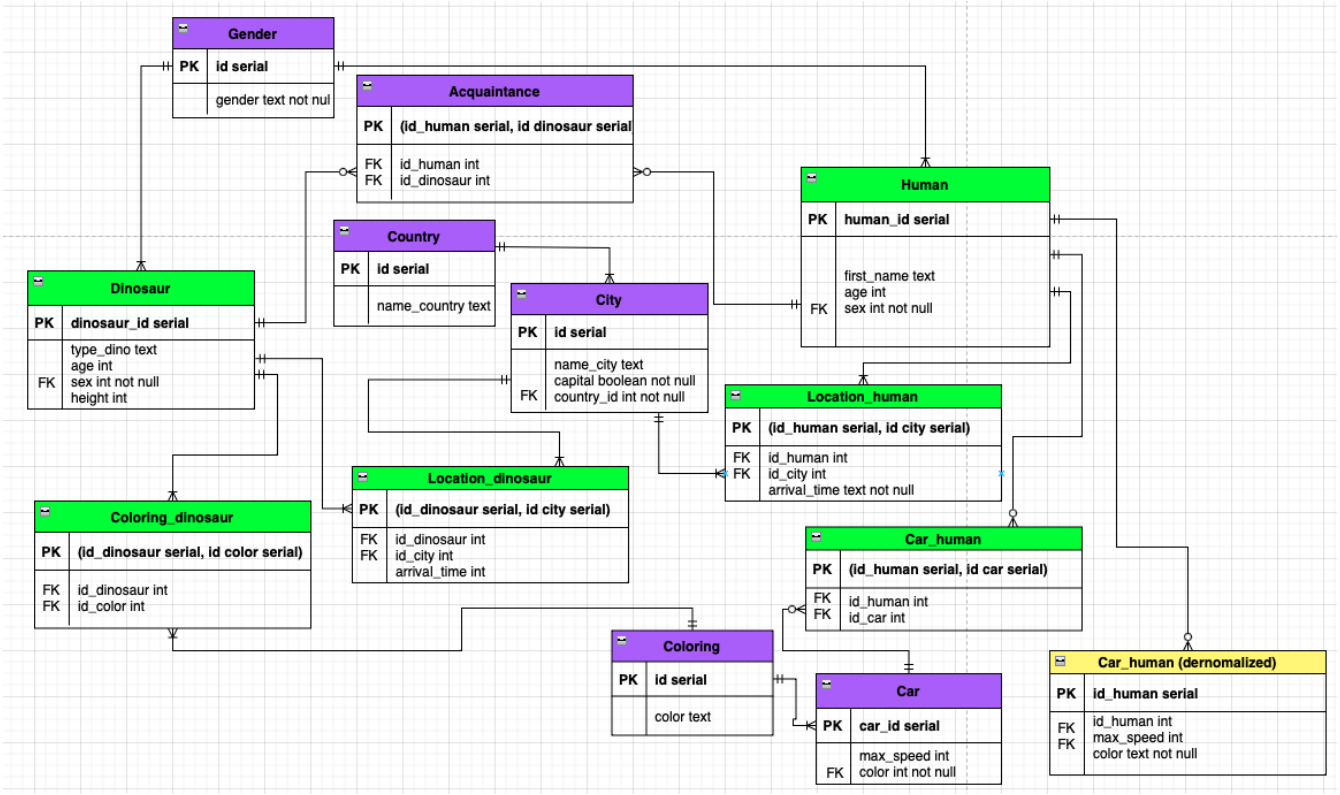
3 Нормализация

1НФ: Всё нормально, исправления не требуются. В отношениях базы данных нет групп из 2-х и более элементов.
2НФ: Всё нормально, исправления не требуются. В отношениях базы данных соблюдается полная функциональная зависимость. Нет частичных функциональных зависимостей. А также соблюдается 1НФ
3НФ: Всё нормально, исправления не требуются. В отношениях базы данных отсутствуют транзитивные функциональные зависимости. А также соблюдаются 1НФ и 2НФ.
НФБК: Всё нормально, исправления не требуются. В отношениях бд все детерминанты являются потенциальными первичными ключами. Т.е. нет ни одного детерминанта не являющегося ключом.

4 Полезная денормализация

В таблице Car_human можно добавить сразу данные о машине и её цвете. Возможна ситуация, когда нам нужно написать запрос, который будет оценивать какого цвета машины больше нравятся человеку и выбирать максимальную по скорости машину из имеющихся в собственности.

5 Функция на языке PL/pgSQL



Сделаем триггер для правильного дублирования данных с учётом полезной денормализации.
Новая таблица:

– Table with Car human dn
CREATE TABLE Car_human_dn
(
id_car SERIAL primary key,
id_human INTEGER REFERENCES Human (human_id),
max_speed integer,
color text
);

Функция:

```sql
CREATE OR REPLACE FUNCTION insert_Car_human_dn()
 RETURNS TRIGGER AS $$
    BEGIN
        IF EXISTS(SELECT 1 FROM Car_human_dn WHERE (Car_human_dn.id_human is null) AND Car_human_dn.id_car = NEW.id_car) THEN
            UPDATE Car_human_dn SET id_human = NEW.id_human
            WHERE (id_human is null) AND id_car = NEW.id_car;
        ELSE
        INSERT INTO Car_human_dn (id_human, max_speed, color)
            SELECT NEW.id_human, Car.max_speed, Coloring.color
            FROM Car_human
            INNER JOIN Car ON Car_human.id_car = Car.car_id
            INNER JOIN Coloring ON Car.color = Coloring.id
            WHERE (Car_human.id_human = NEW.id_human) AND (Car_human.id_car = NEW.id_car);
        END IF;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Car_human_insert_trigger AFTER INSERT ON Car_human
FOR EACH ROW EXECUTE FUNCTION insert_Car_human_dn();


CREATE OR REPLACE FUNCTION insert_Car()
 RETURNS TRIGGER AS $$
    BEGIN
        INSERT INTO Car_human_dn (max_speed, color)
            SELECT NEW.max_speed, Coloring.color
            FROM Car
            INNER JOIN Coloring ON Car.color = Coloring.id
            WHERE(NEW.car_id=Car.car_id);
    RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Car_insert_trigger AFTER INSERT ON Car
FOR EACH ROW EXECUTE FUNCTION insert_Car();

DROP TRIGGER IF EXISTS Car_human_insert_trigger ON Car_human;
DROP FUNCTION IF EXISTS insert_Car_human_dn();
DROP TRIGGER IF EXISTS Car_insert_trigger ON Car;
DROP FUNCTION IF EXISTS insert_Car();
```

6 Вывод

Мы рассмотрели тему нормализации базы данных, где ознакомились с уровнями нормализации. Выполнили полезную денормализацию и объяснили зачем она существует. Написали функцию для добавления новых значений в таблицу.

Список литературы

[1] Кириллов, В. В., Громов, Г. Ю. Введение в реляционные базы данных. Москва: BHV, 2009.