Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

*Факультет программной инженерии и компьютерной техники*

**Лабораторная работа №3**

По дисциплине:

*“Базы данных”*

Выполнил:

Щукин Егор Вячеславович

Группа: Р3114

Преподаватель:

Кривоносов Егор Дмитриевич

Санкт-Петербург,

2024г

**Текст задания:**

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.

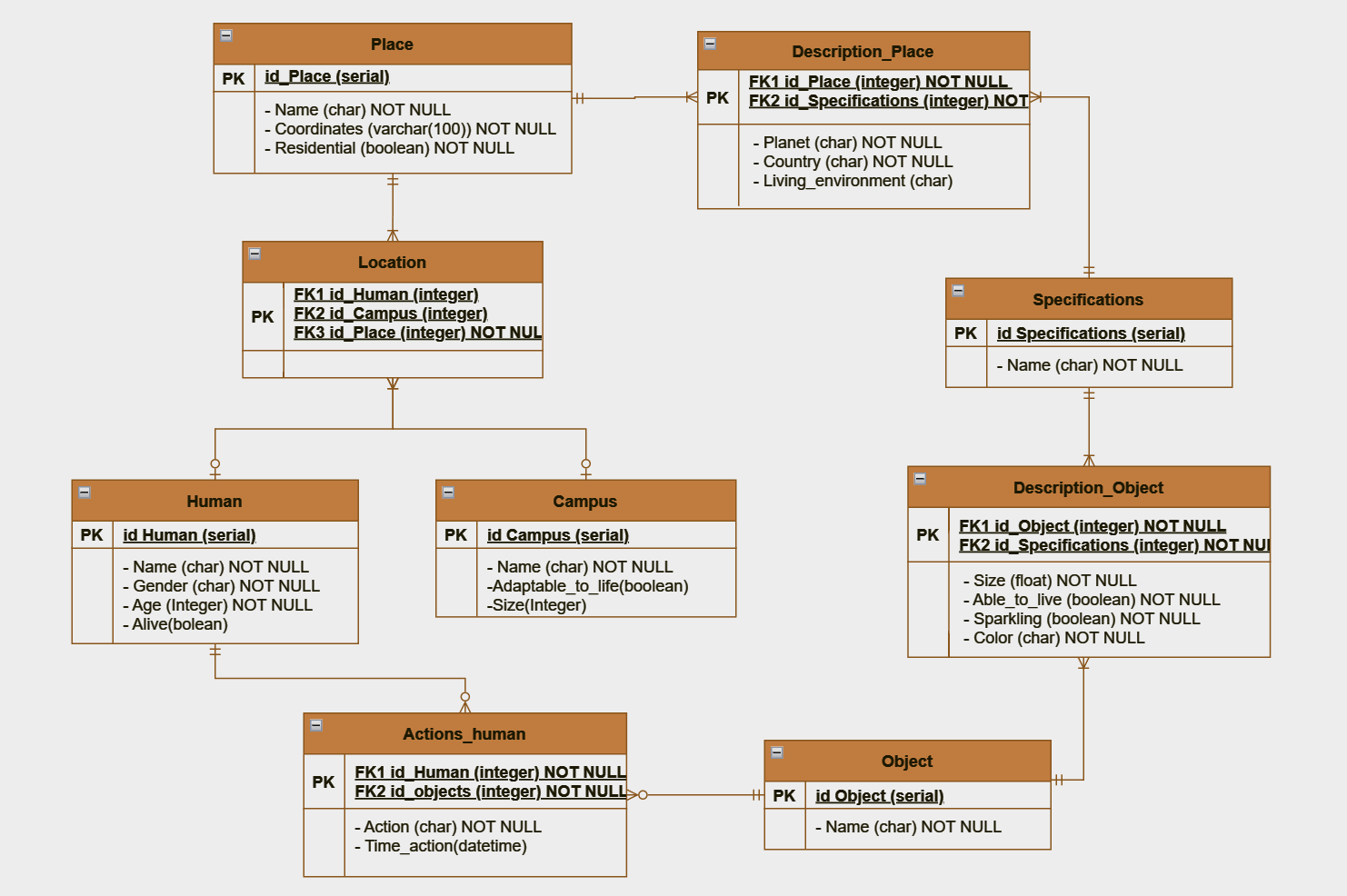
Отчёт по лабораторной работе должен содержать:

1. Текст задания.
2. Исходная, нормализованная и денормализованная модели.
3. Ответы на вопросы, представленные в задании.
4. Функция и триггер на языке PL/pgSQL
5. Выводы по работе.

Темы для подготовки к защите лабораторной работы:

1. Нормализация. Формы
2. Функциональные зависимости. Виды
3. Денормализация
4. Язык PL/pgSQL

Даталогическая модель:



SQL:

DROP TABLE IF EXISTS campus CASCADE;  
DROP TABLE IF EXISTS place CASCADE;  
DROP TABLE IF EXISTS human CASCADE;  
DROP TABLE IF EXISTS actions\_human CASCADE;  
DROP TABLE IF EXISTS object CASCADE;  
DROP TABLE IF EXISTS description\_object CASCADE;  
DROP TABLE IF EXISTS description\_place CASCADE;  
DROP TABLE IF EXISTS specification CASCADE;  
DROP TABLE IF EXISTS location CASCADE;  
  
CREATE TABLE place(  
id SERIAL PRIMARY KEY,  
Name TEXT NOT NULL,  
Coordinates VARCHAR(100) NOT NULL,  
Residential BOOLEAN NOT NULL  
);  
  
CREATE TABLE campus(  
id SERIAL PRIMARY KEY,  
Name VARCHAR(100) NOT NULL,  
Size FLOAT NOT NULL,  
Adaptable\_to\_life BOOLEAN  
);  
  
CREATE TABLE human(  
id SERIAL PRIMARY KEY,  
Name TEXT NOT NULL,  
Gender TEXT NOT NULL,  
Age INTEGER NOT NULL,  
Alive BOOLEAN  
);  
  
CREATE TABLE specification(  
id SERIAL PRIMARY KEY,  
Name TEXT NOT NULL  
);  
  
CREATE TABLE object(  
id SERIAL PRIMARY KEY,  
Name TEXT NOT NULL  
);  
  
CREATE TABLE description\_place(  
id\_Place BIGINT REFERENCES place (id) NOT NULL,  
id\_Specifications BIGINT REFERENCES specification (id) NOT NULL,  
Planet TEXT NOT NULL,  
Country TEXT NOT NULL,  
Living\_environment TEXT NOT NULL  
);  
  
CREATE TABLE actions\_human(  
id\_Object BIGINT REFERENCES object (id) NOT NULL,  
id\_Human BIGINT REFERENCES human(id) NOT NULL,  
Action TEXT NOT NULL,  
Time\_action TIME  
);  
  
CREATE TABLE description\_object(  
id\_Object BIGINT REFERENCES object (id) NOT NULL,  
id\_Specifications BIGINT REFERENCES specification (id) NOT NULL,  
Able\_to\_live BOOLEAN NOT NULL,  
Sparkling BOOLEAN NOT NULL,  
Color TEXT NOT NULL  
);  
  
  
  
CREATE TABLE Location(  
id\_Place BIGINT REFERENCES place (id) NOT NULL,  
id\_Campus BIGINT REFERENCES campus (id),  
id\_Human BIGINT REFERENCES human (id)  
);  
INSERT INTO place(Name, Coordinates, Residential) VALUES  
('Desrt', '43.121.123.230.212',FALSE);  
  
INSERT INTO campus(Name, Size, Adaptable\_to\_life) VALUES  
('Camp\_1',100.0,TRUE);  
  
  
INSERT INTO human(Name, Gender, Age, Alive) VALUES  
('He','Man',18, TRUE),  
('She','Woman',21, TRUE);  
  
INSERT INTO location(ID\_PLACE, ID\_CAMPUS, ID\_HUMAN) VALUES  
(1,1,1);  
  
INSERT INTO object(Name) VALUES  
('Dome'),  
('Bubble');  
  
INSERT INTO specification(Name) VALUES ('Big'), ('Little'), ('Small');  
  
INSERT INTO description\_place(id\_Place, id\_Specifications, planet, country, living\_environment) VALUES  
(1,1,'Earth','Egypt',FALSE);  
  
INSERT INTO description\_object(id\_Object, id\_Specifications, Able\_to\_live, Sparkling, Color) VALUES  
(1,3,TRUE,FALSE,'Blue'),  
(2,2,FALSE,TRUE,'Transparent');  
  
INSERT INTO actions\_human(ID\_OBJECT, ID\_HUMAN, ACTION, TIME\_ACTION) VALUES (1,1,'Узнавать','12:00');

1. **Определим Функциональные зависимости:**

Place: id -> Name, Coordinates, Residential;

Campus: id -> Name, Size, Adaptable\_to\_life;

Human: id -> Name, Gender, Age;

Specifications: id -> Name;

Object: id -> Name;

Description\_place: (id\_Place, id\_Specifications) -> Planet, Country, Living\_environment;

Actions\_human: (id\_Object, id\_Human) -> Action, Time\_action;

Description\_object: (id\_Object, id\_Specifications) -> Able\_to\_live, Sparkling, Color;

1. **Нормальные формы**

**Первая нормальная форма (1NF)** — это базовый уровень нормализации базы данных, который в основном касается структуры хранения данных в таблицах. Для того чтобы таблица базы данных соответствовала первой нормальной форме, она должна удовлетворять следующим условиям:

Атомарность значений: Каждое поле таблицы должно содержать атомарное значение, то есть значение, которое не может быть дальше разделено на более мелкие части с точки зрения базы данных.

Однозначность столбцов: Каждый столбец в таблице должен быть уникальным по названию и содержать данные только одного определённого типа.

Однозначное ключевое поле: Каждая строка в таблице должна быть уникально идентифицируема с помощью "ключа" (первичного ключа), который не содержит повторяющихся значений. Это обеспечивает возможность надёжного обращения к каждой записи в таблице.

Порядок данных не имеет значения: Порядок строк и столбцов не должен влиять на способ интерпретации данных. Это означает, что данные в таблице должны оставаться согласованными независимо от того, как строки или столбцы будут переставлены.

**Вывод:** Все мои таблицы соответствуют первой нормальной форме (1NF). Они содержат атомарные значения, каждая запись уникально идентифицируется, и порядок строк или столбцов не влияет на данные.

**Вторая нормальная форма (2NF) —** это следующий уровень нормализации в базах данных после первой нормальной формы (1NF). Основное требование 2NF заключается в устранении проблем, связанных с частичной функциональной зависимостью не ключевых атрибутов от первичного ключа. Все не ключевые атрибуты должны полностью зависеть от всего первичного ключа, а не только от его части.

**Вывод:** Все таблицы соответствуют требованиям второй нормальной формы (2NF), так как в каждой таблице все не ключевые атрибуты полностью зависят от всего первичного ключа и нет частичных зависимостей.

**Третья нормальная форма (3NF) —** это дополнительный уровень нормализации в базах данных, который следует за второй нормальной формой (2NF). Устранение транзитивных зависимостей: не ключевые атрибуты не должны зависеть от других не ключевых атрибутов, а только от первичных ключей. Транзитивная зависимость в базах данных происходит, когда один не ключевой атрибут зависит от другого не ключевого атрибута, который, в свою очередь, зависит от первичного ключа. Это может привести к избыточности данных и сложностям в обслуживании данных. Для приведения таблицы в 3NF необходимо разбить таблицу так, чтобы каждый не ключевой атрибут был зависим только от первичных ключей.

**Вывод:** Все мои таблицы соответствуют третьей нормальной форме (3NF). В них отсутствуют транзитивные зависимости, и все не ключевые атрибуты зависят только от первичных ключей.

1. **BCNF**

**BCNF** (Boyce-Codd Normal Form) — это усиленная версия третьей нормальной формы (3NF). Чтобы таблица соответствовала BCNF, она должна удовлетворять следующим критериям:

Быть в 3NF: Таблица должна уже соответствовать третьей нормальной форме.

Каждая зависимость определяется суперключом: для любой не ключевой функциональной зависимости, первичный ключ должен быть суперключом, т.е. в каждой нетривиальной функциональной зависимости вида X → Y, X должен быть суперключом. Суперключ — это набор одного или нескольких атрибутов, который может уникально идентифицировать строку в таблице. Первичный ключ является специальным случаем суперключа, который минимален, то есть не может содержать лишних атрибутов и все же сохранять свойство уникальности.

Моя таблица находится в BCNF. Составные атрибуты не составного ключа не должны зависеть от не ключевых атрибутов, что у меня выполнено.

1. **Денормализация**

***Объединение таблиц:*** Объединение таблиц может облегчить работу с Join тем самым ускорив работу запросов. В моей таблице, если будет часто запрашиваться действие человека и , например, лагерь, где он живет, то можно рассмотреть объединение human\_campus и Human.

***Добавление часто запрашиваемых атрибутов:*** Увеличение количества атрибутов может облегчить поиск. Например, если в моей таблице часто будут подсчитывать количество возможных действий у человека, тогда можно добавить атрибут actions\_count, который будет считать количество действий у человека и пересчитываться при добавлении или удалении действия.

1. **Триггер**

Триггер изменяет живой ли человек, если человека заселить в лагерь, который не приспособлен для жизни, то человек умирает и если статус лагеря обновлен на неприспособлен для жизни то человек тоже умирает

CREATE OR REPLACE FUNCTION *update\_adaptability*()  
RETURNS TRIGGER AS $$  
BEGIN  
 IF TG\_OP = 'INSERT' THEN

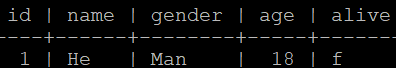
IF (SELECT Adaptable\_to\_life FROM campus WHERE id = NEW.id\_campus) = FALSE THEN   
 UPDATE human SET alive = FALSE WHERE id = NEW.id\_human;  
 END IF;  
 ELSEIF TG\_OP = 'UPDATE' THEN

IF NEW.Adaptable\_to\_life = FALSE THEN

UPDATE human SET alive = FALSE WHERE id IN (SELECT id\_human FROM location WHERE id\_campus = NEW.id);  
 END IF;  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER update\_adaptability\_trigger  
AFTER INSERT OR UPDATE OF Adaptable\_to\_life ON campus  
FOR EACH ROW  
EXECUTE FUNCTION *update\_adaptability*();

Примеры:

1) UPDATE campus set Adaptable\_to\_life=FALSE where id=1;



2) INSERT INTO campus(Name, Size, Adaptable\_to\_life) VALUES  
('Camp\_2',200.0,FALSE);

INSERT INTO location(ID\_PLACE, ID\_CAMPUS, ID\_HUMAN) VALUES  
(1,2,2);



**Вывод:** я изучил отличия нормальных форм, также понял виды триггеров, а также написал свой маленький триггер.