НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Изображение выглядит как Шрифт, логотип, Графика, белый

Автоматически созданное описание

Базы Данных

**Лабораторная работа № 3**

Выполнил студент

Двоеглазова Наталья Николаевна 413751

Группа № P3123

Преподаватель:

Горбунов Михаил Витальевич

г. Санкт-Петербург

2024

**Задание**

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 3NF (как минимум). Постройте схему на основе NF;

4. Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;

5. Какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

6. Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

**1. Функциональные зависимости**

Location: id (name, size)

Object: id (name, real\_or\_not, location\_id)

Characteristic: id (name, object\_id)

Action: id (name, object\_id)

Influence: id (name, action\_id)

Person\_Condition: id (name, influence\_id)

Appearance: id (description)

Person: id (name, age, location\_id, condition\_id, appearance\_id)

**2. Нормализация**

1НФ:

Отношение находится в 1НФ, если все его атрибуты являются атомарными. Не должно быть повторений строк в таблице.

Моя модель удовлетворяет данным условиям, т.к. все атрибуты атомарные, и отсутствуют повторения строк в таблицах.

2НФ:

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК).

Моя модель находится в 1НФ и каждый атрибут зависит от Primary Key, в моем случае id.

* Модель удовлетворяет условиям 2НФ.

3НФ:

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.

У моей модели выполнены все условия принадлежности к 3НФ:

2НФ, нет транзитивных связей

**3. BCNF**

BCNF

Отношение будет находиться в BCNF при условии, что для каждой функциональной зависимости X Y – X будет являться суперключом.

В моей модели для всех функциональных зависимостей X (id) является суперключом.

**4. Денормализация**

Денормализация позволяет повысить производительность выполнения различных запросов путем объединения связных таблиц, добавление избыточных атрибутов.

**Объединение связанных таблиц:**

Объединение таблиц может уменьшить количество операций JOIN и ускорить обработку запросов.

Можно рассмотреть объединение таблиц Person и Appearance, если у человека будет часто запрашиваться данные о внешнем виде. Также можно добавить таблицы Object\_Location (id, object\_id, location\_id) и Action\_Location (id, action\_id, object\_location\_id). Таким образом можно получить данные о местонахождении объекта и локации действия.

**Добавление избыточных атрибутов:**

Например, если будет часто запрашиваться характеристика объекта, то можно будет добавить атрибут characteristic к таблице Object.

**5. Функция и триггер**

CREATE OR REPLACE FUNCTION person\_trigger()

RETURNS TRIGGER AS $person\_trigger$

BEGIN

IF new.id IS NULL OR

new.name IS NULL OR

new.location\_id IS NULL OR

new.appearance\_id is NULL OR

new.condition\_id IS NULL

THEN RAISE EXCEPTION 'Отсутствует значение одного из обязательных полей';

END IF;

IF new.age > '120'

THEN RAISE EXCEPTION 'Возраст не может быть больше 120 лет';

END IF;

IF new.location\_id NOT IN (SELECT (id) FROM Location)

THEN RAISE EXCEPTION '';

END IF;

IF new.condition\_id NOT IN (SELECT (id) FROM Person\_Condition)

THEN RAISE EXCEPTION '';

END IF;

IF new.appearance\_id NOT IN (SELECT (id) FROM Appearance)

THEN RAISE EXCEPTION '';

END IF;

RETURN new;

END;

$person\_trigger$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER Person\_trigger

BEFORE INSERT OR UPDATE ON Person

FOR EACH ROW EXECUTE FUNCTION person\_trigger();

Но этому триггеру не хватает логики, поскольку все указанные ограничения можно установить еще при создании таблицы, добавив пользовательские требования с помощью CHECK.

Поэтому, напишем триггер, который будет создавать резервные копии таблицы Person при удалении записи или изменении в ней данных.

CREATE TABLE Person\_backup (

id SERIAL PRIMARY KEY,

person\_id BIGINT NOT NULL,

name TEXT NOT NULL,

status TEXT NOT NULL

);

CREATE OR REPLACE FUNCTION backup\_update ()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Person\_backup(person\_id, name, status)

VALUES (OLD.id, OLD.name, 'updated');

RETURN OLD

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER Backup\_update

BEFORE UPDATE ON Person

FOR EACH ROW EXECUTE FUNCTION backup\_update();

UPDATE Person SET name='Илюха' WHERE name='Илья';

-----------------------------------------------------------------------

CREATE OR REPLACE FUNCTION backup\_delete ()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Person\_backup(person\_id, name, status)

VALUES (OLD.id, OLD.name, 'updated');

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER Backup\_delete

BEFORE DELETE ON Person

FOR EACH ROW EXECUTE FUNCTION backup\_delete();

INSERT INTO Person (name, age, location\_id, appearance\_id, condition\_id) VALUES ('Иркутянин', 35, 6, 1, 5);

DELETE FROM Person WHERE name='Иркутянин';

**6. Вывод**

В процессе выполнения работы я изучила такие понятия как: нормальные формы, BCNF, функциональные зависимости в БД, денормализация, функции и триггеры в PL/pgSQL.