Федеральное государственное автономное образовательное учреждение высшего

образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Отчет**

**По лабораторной работе №3**

По дисциплине «Базы данных»

Вариант 1300.

Выполнил: Чураков Александр Алексеевич, группа P3131

Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург

~ 2024 ~

# 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 основе 3NF (как минимум). Постройте схему на основе полученных отношений;
* опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе 3NF;
* 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.

# Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт Автоматически созданное описание

# Функциональные зависимости

Body: id 🡪 (skin\_color, finger\_amount, arm\_amount)

Head: id 🡪 (hair\_color, face\_type)

Leg: id 🡪 (leg\_amount)

Construction: id 🡪 (head\_id, body\_id, legs\_id)

Supporter: id 🡪 (name)

Theory: id 🡪 (name, description)

Preferred\_constrution: (construction\_id, supporter\_id) 🡪 ()

Preferred\_theory: id 🡪 (supporter\_id, theory\_id)

# Нормальные формы

**1NF**: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. Моя модель удовлетворяет 1NF, так как все атрибуты атомарны, и нет повторяющихся групп.

**2NF**: Отношение находится во 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от первичного ключа. Моя модель удовлетворяет 2NF, так как все неключевые атрибуты полностью функционально зависят от первичных ключей.

**3NF**: Отношение находится в 3NF, если оно находится во 2NF и не содержит транзитивных зависимостей. Моя модель удовлетворяет 3NF, так как все неключевые атрибуты зависят только от первичных ключей, и не содержат транзитивных зависимостей.

# BCNF

Отношение находится в BCNF, если для каждой функциональной зависимости X →Y, X является суперключом.

Моя модель удовлетворяет BCNF, так как для всех функциональных зависимостей X является суперключом.

# Денормализация

**Объединение связанных таблиц**: в некоторых случаях, объединение таблиц может уменьшить количество операций JOIN и ускорить обработку запросов. Например, можно соединить таблицы preferred\_theory и preferred\_construction, если необходимо часто получать информацию о предпочитаемой сторонником конструкции и теории

**Добавление избыточных атрибутов**: иногда добавление избыточных атрибутов может улучшить производительность запросов. Например, можно добавить атрибут supporters\_amount, который будет отображать количество поддерживающих конструкцию сторонников.

# Функция PL/sql

CREATE OR REPLACE FUNCTION *count\_supporters\_amount*(sup\_id int)  
 RETURNS int AS  
$$  
DECLARE  
 supporters\_amount int := null;  
BEGIN  
 select *count*(construction\_id)  
 into supporters\_amount  
 from preferred\_construction pc  
 where pc.supporter\_id = sup\_id  
 group by pc.supporter\_id;  
  
 if not FOUND then  
 raise exception 'supporter with such id not found';  
 end if;  
  
 return supporters\_amount;  
END;  
$$ LANGUAGE plpgsql;  
  
  
CREATE OR REPLACE FUNCTION *update\_construction*()  
 RETURNS trigger AS  
$$  
DECLARE  
 sup\_id\_to\_update int := null;  
 sup\_amount int := null;  
BEGIN  
  
  
 SELECT pc.supporter\_id  
 into sup\_id\_to\_update  
 from preferred\_construction pc  
 where pc.supporter\_id = NEW.supporter\_id;  
  
 sup\_amount := *count\_supporters\_amount*(sup\_id\_to\_update);  
  
 update construction  
 set supporters\_amount = sup\_amount  
 where construction.id = sup\_id\_to\_update;  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
  
CREATE OR REPLACE TRIGGER supporters\_amount\_trigger  
 AFTER INSERT OR UPDATE or delete  
 ON preferred\_construction  
 FOR EACH ROW  
EXECUTE function *update\_construction*();

# Вывод

При выполнении лабораторной работы я познакомился с понятием нормализации и денормализации. Научился определять функциональные зависимости модели. Познакомился с процедурным языком PL/pgSQL.