Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине

‘Базы данных’

*Выполнил:*

Студент группы P3115

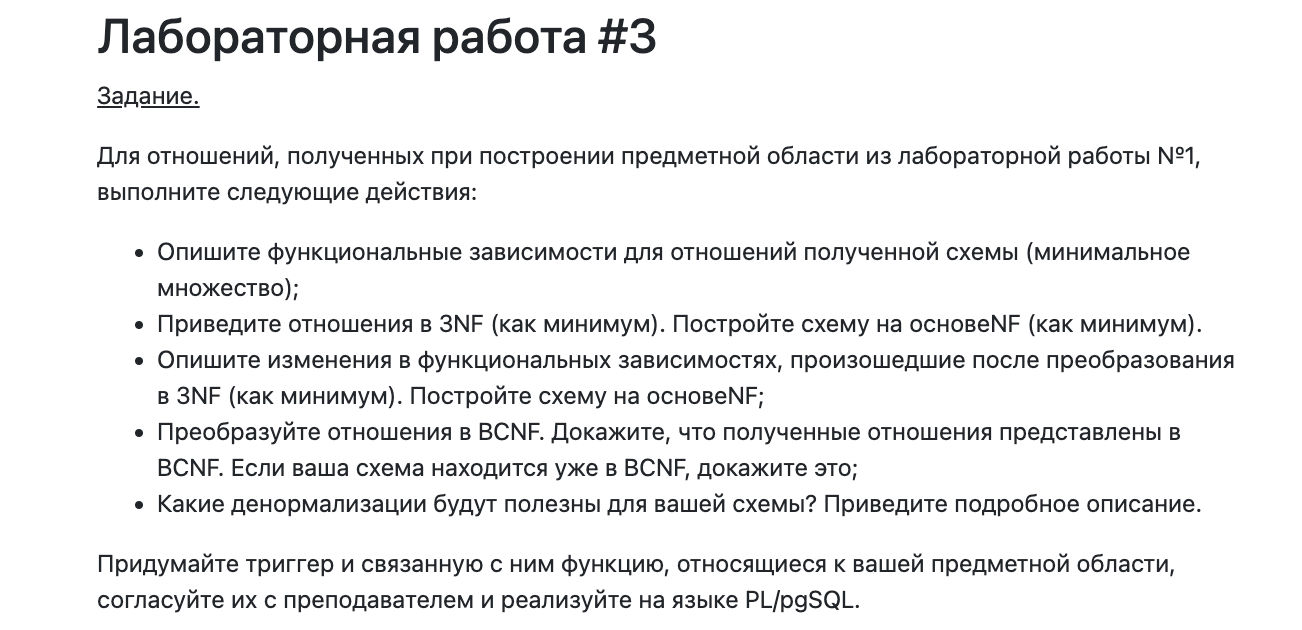
Жук Иван Александрович

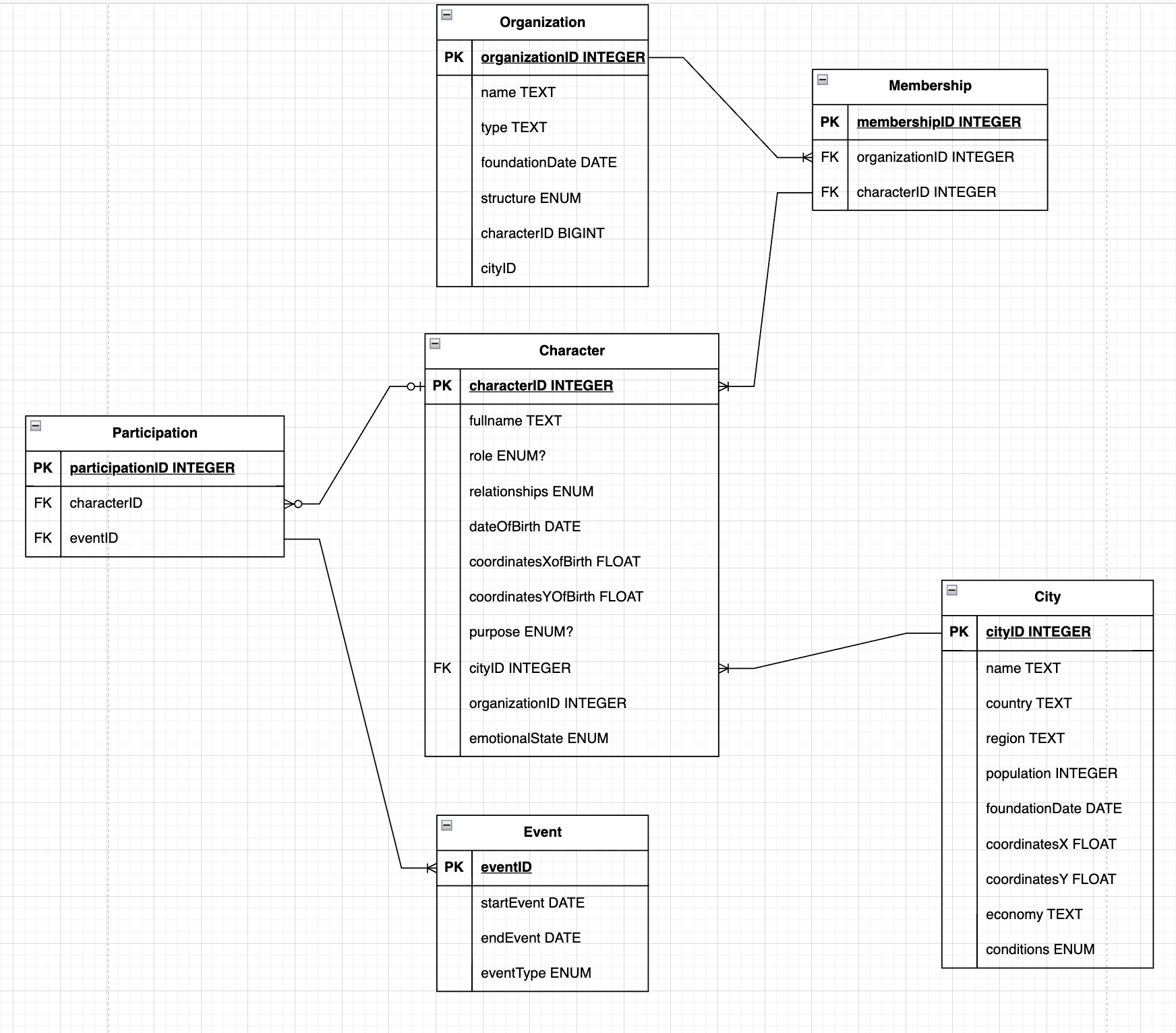
*Преподаватель:*

Мартин Райла



Санкт-Петербург, 2023

1. **Текст задания  
   **



1. **Функциональные зависимости**

|  |
| --- |
| city: id → (cityName, country, region, population, foundationDate, coordinatesX, coordinatesY, economy, cityState)    person: id → (fullname, role, dateOfBirth, dateOfDeath, coordinatesXofBirth, coordinatesYofBirth, cityID, emotionalState)    organization: id → (organizationName, type, foundationDate, structure, cityID, numberOfPeople)    event: id → (startEvent, endEvent, eventType)    membership: id → (organizationID, personID)    participation: id → (personID, eventID) |

**3. 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. |

**3. BCNF**

|  |
| --- |
| Реляционная структура данных соответствует условиям BCNF, если в каждом случае функциональной зависимости X → Y, X выступает в роли суперключа. Моя разработанная модель отвечает требованиям BCNF, потому что во всех случаях функциональной зависимости X представляет собой суперключ. |

**3. Денормализация**

|  |
| --- |
| Для повышения производительности можно рассмотреть следующие денормализации:  Денормализация города в person и organization: добавление атрибутов cityName, country, region непосредственно в таблицы person и organization для уменьшения количества JOIN операций при получении информации о городе.  Создание агрегированных таблиц: создание специальных таблиц для хранения предрасчитанных значений, например, количество участников на каждого человека и т.д. |

1. **Вывод**

В ходе выполнения лабораторной работы я освоил концепции нормализации и денормализации. Я научился определению функциональных зависимостей в модели и анализу соответствия модели различным нормальным формам. Я также изучил эффективные методы денормализации структуры базы данных и сценарии, где их применение может быть целесообразно.

Доп.задание:

|  |
| --- |
| --Доп: написать и утвердить триггер -- Удаление существующего триггера и функции, если они есть  **DROP** **TRIGGER** **IF** **EXISTS** participation\_trigger **ON** participation;  **DROP** **FUNCTION** **IF** **EXISTS** check\_participation();  -- Создание функции триггера  **CREATE** **FUNCTION** check\_participation() **RETURNS** trigger **AS** $$  **DECLARE**  participant\_city\_id INTEGER;  event\_city\_id INTEGER;  event\_type event\_type;  organization\_id INTEGER;  city\_participants INTEGER;  participant\_dob DATE;  **BEGIN**  -- Получение cityID участника  **SELECT** cityID, dateOfBirth **INTO** participant\_city\_id, participant\_dob  **FROM** person  **WHERE** personID = NEW.personID;  -- Получение cityID события  **SELECT** city.cityID **INTO** event\_city\_id  **FROM** event  **JOIN** person **ON** person.cityID = participant\_city\_id  **WHERE** event.eventID = NEW.eventID;  -- Проверка, что участник и событие находятся в одном городе  **IF** participant\_city\_id **IS** **DISTINCT** **FROM** event\_city\_id **THEN**  **RAISE** **EXCEPTION** 'Participant and event are in different cities';  **END** **IF**;  -- Получение типа события  **SELECT** eventType **INTO** event\_type  **FROM** event  **WHERE** eventID = NEW.eventID;  -- Обновление эмоционального состояния участника на основе типа события  **IF** event\_type = 'celebration' **THEN**  **UPDATE** person **SET** emotionalState = 'happy' **WHERE** personID = NEW.personID;  **ELSIF** event\_type = 'war' **THEN**  **UPDATE** person **SET** emotionalState = 'angry' **WHERE** personID = NEW.personID;  **ELSIF** event\_type = 'memorial\_service' **THEN**  **UPDATE** person **SET** emotionalState = 'sad' **WHERE** personID = NEW.personID;  **ELSE**  **UPDATE** person **SET** emotionalState = 'content' **WHERE** personID = NEW.personID;  **END** **IF**;  -- Получение organizationID участника  **SELECT** organizationID **INTO** organization\_id  **FROM** membership  **WHERE** personID = NEW.personID  **LIMIT** 1;  -- Увеличение счетчика участников в организации  **IF** organization\_id **IS** **NOT** **NULL** **THEN**  **UPDATE** organization  **SET** numberOfPeople = numberOfPeople + 1  **WHERE** organizationID = organization\_id;  **END** **IF**;  -- Обновление статистики города  **INSERT** **INTO** city\_stats (cityID, totalParticipants)  **VALUES** (participant\_city\_id, 1)  **ON** **CONFLICT** (cityID) **DO** **UPDATE**  **SET** totalParticipants = city\_stats.totalParticipants + 1;  -- Получение текущего количества участников в городе  **SELECT** totalParticipants **INTO** city\_participants  **FROM** city\_stats  **WHERE** cityID = participant\_city\_id;  -- Проверка и обновление состояния города на основе количества участников  **IF** city\_participants > 1000 **THEN**  **UPDATE** city  **SET** cityState = 'vibrant'  **WHERE** cityID = participant\_city\_id;  **ELSIF** city\_participants > 500 **THEN**  **UPDATE** city  **SET** cityState = 'developing'  **WHERE** cityID = participant\_city\_id;  **END** **IF**;  -- Добавление нового события в таблицу event при достижении определенного порога участников  **IF** city\_participants = 1000 **THEN**  **INSERT** **INTO** event (startEvent, eventType)  **VALUES** (CURRENT\_DATE, 'celebration');  **END** **IF**;  -- Проверка и изменение структуры организации, если количество участников превышает порог  **IF** organization\_id **IS** **NOT** **NULL** **THEN**  **UPDATE** organization  **SET** structure = 'matrix'  **WHERE** organizationID = organization\_id **AND** numberOfPeople > 1000;  **END** **IF**;  -- Проверка даты рождения участника и добавление бонусного события  **IF** participant\_dob **IS** **NOT** **NULL** **AND** participant\_dob = CURRENT\_DATE **THEN**  **INSERT** **INTO** event (startEvent, eventType)  **VALUES** (CURRENT\_DATE, 'birthday\_celebration');  **END** **IF**;  **RETURN** NEW;  **END**;  $$ **LANGUAGE** plpgsql;  -- Создание триггера, который вызывает функцию при вставке в таблицу participation  **CREATE** **TRIGGER** participation\_trigger  **BEFORE** **INSERT** **ON** participation  **FOR** **EACH** **ROW**  **EXECUTE** **FUNCTION** check\_participation(); |