Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант № 1010

Лабораторная работа №3

По дисциплине

Базы Данных

Выполнил:

Захра дарабзадех  
студент группы P3130.

Преподаватель:

Перцев Тимофей Сергеевич

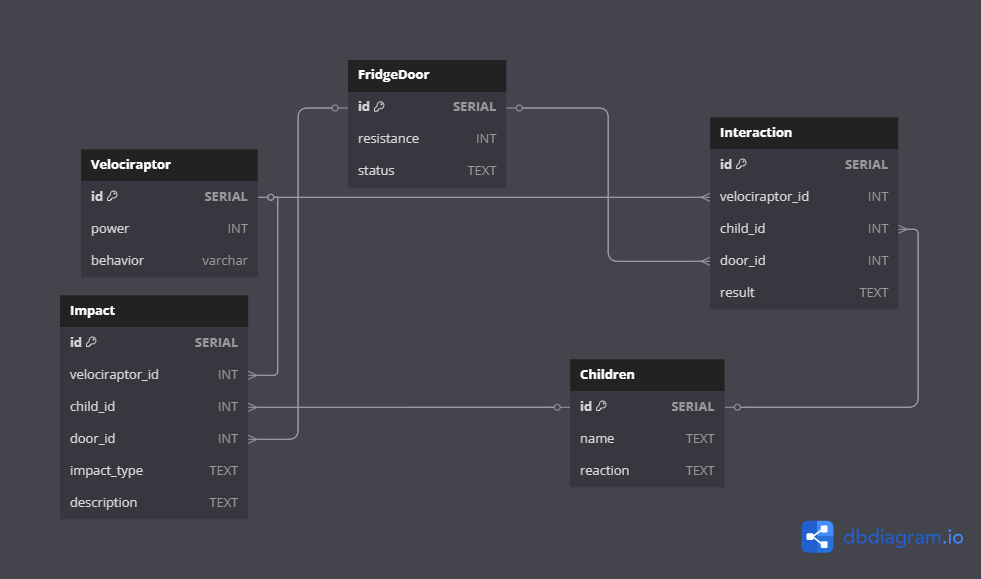
Санкт-Петербург 2025 г.

1. **Текст задания**

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.

Даталогическая модель (исходная):

1. **Функциональные зависимости**

Fridge door: id → (resistance , status)

Children: id → (name , reaction)

Velociraptor: id → (power , behavior)

Interaction: id → ( velociraptor\_id, child\_id, door\_id, result)

(velociraptor\_id, child\_id, door\_id) → result)

Impact: id → (velociraptor\_id, child\_id, door\_id, impact\_type, description)

(velociraptor\_id, child\_id, door\_id) → impact\_type, description)

1. **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

1. **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.

(Все наши FD либо имеют слева id (ключ), либо композицию (velociraptor\_id, child\_id, door\_id) — тоже ключ для альтернативной записи.)

1. **Денормализация**

**Объединение связанных таблиц:** в некоторых случаях объединение таблиц может сократить число JOIN и ускорить обработку запросов по типовым сценариям. Например, можно объединить таблицы Interaction и Children в одно отношение ChildInteraction(id, velociraptor\_id, door\_id, child\_id, child\_name, result), чтобы сразу получать имя ребёнка вместе с результатом взаимодействия без дополнительного соединения.  
**Добавление избыточных атрибутов:** иногда стоит ввести избыточные поля для ускорения агрегаций. Так, в таблицу FridgeDoor можно добавить столбец total\_impacts INT — хранить общее число записей в Impact, а в Velociraptor добавить last\_interaction\_result TEXT — для быстрого получения последнего результата взаимодействия. При вставке новых записей через триггер обновлять эти поля, чтобы избежать выполнения COUNT(\*) и ORDER BY в каждом запросе.

1. **Функция на языке PL/pgSQL**

**Сценарий:** При вставке новой записи в таблицу Impact, если impact\_type = 'hit', подсчитывается общее количество ударов по соответствующей двери. Когда количество ударов достигает или превышает значение сопротивления (resistance) этой двери, её статус меняется на 'broken'.

-- 1. Функция триггера

**CREATE** **OR** **REPLACE** **FUNCTION** update\_door\_status()

**RETURNS** **TRIGGER** **AS** $$

**DECLARE**

total\_hits INTEGER;

door\_resistance INTEGER;

**BEGIN**

IF **NEW**.impact\_type != 'hit' **THEN**

**RETURN** **NEW**;

**END** IF;

-- Получаем сопротивление двери

**SELECT** resistance **INTO** door\_resistance

**FROM** FridgeDoor

**WHERE** id = **NEW**.door\_id;

-- Считаем только действительные удары (hit)

**SELECT** **COUNT**(\*) **INTO** total\_hits

**FROM** Impact

**WHERE** door\_id = **NEW**.door\_id

**AND** impact\_type = 'hit'

**AND** id != **NEW**.id;

-- Проверяем условие поломки

IF (total\_hits + **1**) >= door\_resistance **THEN**

**UPDATE** FridgeDoor

**SET** status = 'broken'

**WHERE** id = **NEW**.door\_id;

RAISE NOTICE 'Дверь % сломана! Удары: %, Сопротивление: %',

**NEW**.door\_id, (total\_hits + **1**), door\_resistance;

**END** IF;

**RETURN** **NEW**;

**END**;

$$ **LANGUAGE** plpgsql;

-- 2. Создание триггера

**DROP** **TRIGGER** IF **EXISTS** check\_door\_status **ON** Impact;

**CREATE** **TRIGGER** check\_door\_status

**AFTER** **INSERT** **ON** Impact

**FOR** **EACH** **ROW**

**EXECUTE** **FUNCTION** update\_door\_status();

-- 3. Тестовые данные

**TRUNCATE** **TABLE** FridgeDoor, Impact **RESTART** **IDENTITY** **CASCADE**;

-- Дверь с сопротивлением 3

**INSERT** **INTO** FridgeDoor (resistance, status)

**VALUES** (**3**, 'intact');

-- Тест

**INSERT** **INTO** Impact (velociraptor\_id, child\_id, door\_id, impact\_type, description)

**VALUES**

(**1**, **1**, **1**, 'hit', 'Первый удар'),

(**1**, **2**, **1**, 'hit', 'Второй удар'),

(**1**, **3**, **1**, 'nothing', 'Третий удар'),

(**1**, **4**, **1**, 'hit', 'Четвертый удар'); -- Должен сломать дверь

-- Проверка результатов

**SELECT** \* **FROM** FridgeDoor; -- Статус должен быть 'broken'

**SELECT** \* **FROM** Impact;

Выводы :

Во время выполнения лабораторной работы я познакомился с процессами нормализации и денормализации. Научился анализировать схему и находить в ней узкие места. Узнал что такое триггер и потренировался писать свои реализации триггеров