МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине

‘Базы данных’

Вариант №408078

Выполнил:
Студент группы P3132
Юксель Хамза

Преподаватель:
Егошин Алексей
Васильевич

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2025

1. Текст задания

Лабораторная работа #3

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.

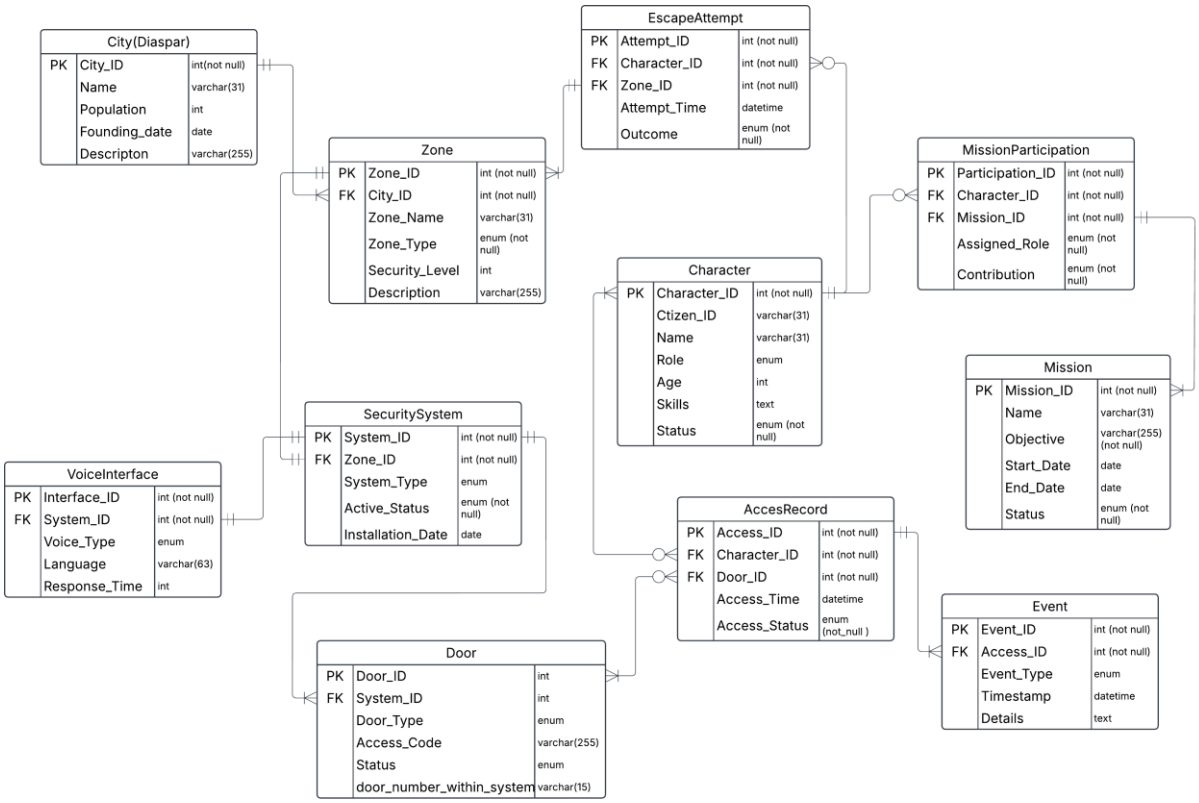
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

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



2. Функциональные зависимости

- **City:**
  - name -> population, founding_date, description
  - (and city_id -> all, name -> city_id)
- **Character:**
  - citizen_id -> name, role, age, skills, status
  - (and character_id -> all, citizen_id -> character_id)
- **Zone:**
  - {city_id, zone_name} -> zone_type, security_level, description
  - (and zone_id -> all, {city_id, name} -> zone_id)
- **Securitysystem:**
  - zone_id -> security_type, status, installation_date
  - (and system_id -> all, zone_id -> system_id)
- **Door:**
  - {System_ID, door_number_within_system} -> Door_Type, Access_Code, Status
  - (and Door_ID -> all, {System_ID, door_number_within_system} -> Door_ID)
- **AccessRecord:**
  - {Character_ID, Door_ID, Access_Time} -> Access_Status
  - (and Access_ID -> all, {Character_ID, Door_ID, Access_Time} -> Access_ID)
- **Event:**
  - {Access_ID, Event_Type, Timestamp} -> Details
  - (and Event_ID -> all, {Access_ID, Event_Type, Timestamp} -> Event_ID)
- **EscapeAttempt:**
  - {Character_ID, Zone_ID, Attempt_Time} -> Outcome
  - (and Attempt_ID -> all, {Character_ID, Zone_ID, Attempt_Time} -> Attempt_ID)

- **Mission:**
  - name -> objective, start_date, end_date, status
  - (and mission_id -> all, name -> mission_id)
- **MissionParticipation:**
  - {Character_ID, Mission_ID} -> Assigned_Role, Contribution
  - (and Participation_ID -> all, {Character_ID, Mission_ID} -> Participation_ID)
- **VoiceInterface:**
  - System_ID -> Voice_Type, Language, Response_Time
  - (and Interface_ID -> all, System_ID -> Interface_ID)

3. Нормальные формы

Условия 1NF:

- Каждая ячейка таблицы содержит единственное (атомарное) значение,
- Нет повторяющихся или многозначных столбцов,
- Строки таблицы могут быть отделены друг от друга.

1NF: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения.
Моя модель удовлетворяет 1NF, так как:

1. **Атомарность значений:** Каждый атрибут во всех таблицах предназначен для хранения одного значения соответствующего типа.
2. **Отсутствие повторяющихся групп**
3. **Уникальность строк:** Каждая таблица имеет первичный ключ (city_id, character_id, zone_id и т.д.), который уникально идентифицирует каждую строку, обеспечивая их различимость.

2NF: Отношение находится во 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от *каждого* первичного ключа (или, точнее, от каждого кандитатного ключа целиком).
Моя модель удовлетворяет 2NF, так как:

1. **Все таблицы находятся в 1NF**.
2. **Отсутствие частичных зависимостей:**
   - Для таблиц с **простыми (не составными) кандидатными ключами**, частичные зависимости невозможны по определению, так как ключ не имеет частей. Все неключевые атрибуты в этих таблицах зависят от всего кандидатного ключа.
     - Например, в City, population зависит от name (целиком), а не от части name.
     - В Character, role зависит от citizen_id (целиком).
   - Для таблиц с **составными кандидатными ключами** :
     - Zone: CK {city_id, name}. Неключевые атрибуты (zone_type, security_level, description) зависят от всей комбинации {city_id, name}, а не только от city_id или только от name. Например, security_level определяется конкретной зоной в конкретном городе.

3NF: Отношение находится в 3NF, если оно находится во 2NF и не содержит транзитивных зависимостей неключевых атрибутов от кандидатного ключа. То есть, ни один неключевой атрибут не зависит от другого неключевого атрибута.
Моя модель удовлетворяет 3NF, так как:

1. **Все таблицы находятся в 2NF**.
2. **Отсутствие транзитивных зависимостей:** В каждой таблице все неключевые атрибуты напрямую зависят от кандидатных ключей, а не через другие неключевые атрибуты.
   - **City**: population, founding_date, description напрямую зависят от name (или city_id). Нет такого, чтобы description зависело от population,

a population от name.
- **Character**: name, role, age, skills, status напрямую зависят от citizen_id (или character_id). Нет зависимости, например, skills от role, a role от citizen_id.

4. BCNF (Boyce-Codd Normal Form)

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.

- **City**:
  - Candidate Keys (superkeys): {city_id}, {name}.
  - All determinants of non-trivial FDs (city_id, name) are superkeys.
- **Character**:
  - Candidate Keys (superkeys): {character_id}, {citizen_id}.
  - All determinants of non-trivial FDs (character_id, citizen_id) are superkeys.

5. Денормализация

Объединение связанных таблиц: Таблицы SecuritySystem и VoiceInterface связаны отношением один-к-одному (каждая система имеет ровно один интерфейс, и каждый интерфейс принадлежит ровно одной системе). Часто при запросе информации о системе безопасности требуется также информация о ее голосовом интерфейсе.
Действие денормализации: Объединить атрибуты таблицы VoiceInterface с атрибутами таблицы SecuritySystem в одну новую таблицу SecuritySystem_Voice_Combined. Первичным ключом останется System_ID.

- **Преимущества:**
  - Устранение операции JOIN между SecuritySystem и VoiceInterface при запросе полной информации.
  - Некоторое упрощение схемы за счет уменьшения количества таблиц.
- **Недостатки:**
  - Таблица становится "шире", что может быть неэффективно, если часто запрашиваются только атрибуты системы безопасности *или* только атрибуты голосового интерфейса.
  - Если какие-то атрибуты голосового интерфейса могут быть NULL (например, если интерфейс необязателен, хотя по описанию он обязателен), это может привести к появлению NULL-значений в объединенной таблице.

Добавление избыточных атрибутов: В некоторых случаях добавление избыточных атрибутов может улучшить производительность запросов. Например, при просмотре журналов доступа (AccessRecord) отображение имени персонажа напрямую, без соединения с таблицей Character, может быть быстрее или удобнее, особенно если журналы обширны.
Недостатки: Избыточность (Character_Name хранится в Character и AccessRecord). Требуется обновление AccessRecord при изменении Character_Name.

We can add character name attribute to the access record so that we can display the access logs in the access records with the character name.

6. Функция на языке PL/pgSQL

Функция на языке PL/pgSQL для подсчета расстояние между двумя лабораториями.

При отклонении попытки доступа (AccesRecord.Access_Status = 'Denied'), автоматически создавать событие 'Security Alert' в таблице Event.

```
CREATE OR REPLACE FUNCTION log_denied_access_event()
RETURNS TRIGGER AS $$                    -- trigger türünde bir fonksiyon. Bu, sadece trigger tarafından çağrılabilir
BEGIN
    INSERT INTO Event (Access_ID, Event_Type, Timestamp, Details)
    VALUES (NEW.Access_ID, “SECURITY ALERT!!”, now(), “Access denied attempt logged for door” ||
NEW.Door_ID || “by character ID:” || NEW.Character_ID);

    RETURN NEW; -- insert triggerı için new döndürür.
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_log_denied_access
AFTER INSERT ON AccessRecord –access recorda kayıtttan sonra
FOR EACH ROW   -- eklenen her satır için
WHEN (NEW.Access_Status = 'Denied')  -- Sadece durum denied ise
EXECUTE FUNCTION log_denied_access_event() --yukarıdaki fonksiyonu çalıştır.



CREATE OR REPLACE FUNCTION check_installation_date()
RETURNS TRIGGER AS
$$
DECLARE
   city_founding_date DATE;
BEGIN
   -- City'nin founding_date bilgisini Zone üzerinden al.
   SELECT c.founding_date INTO city_founding_date
   FROM City c
   JOIN Zone z ON z.City_ID = c.City_ID
   WHERE z.Zone_ID = NEW.Zone_ID;

   -- Eğer şehir bulunamazsa veya tarih koşulu sağlanmazsa hata ver.
   IF city_founding_date IS NULL THEN
      RAISE EXCEPTION 'No city found for the given Zone_ID: %', NEW.Zone_ID;
   END IF;

   IF NEW.Installation_Date < city_founding_date THEN
      RAISE EXCEPTION 'Installation_Date (%) must be on or after city founding_date (%)',
                  NEW.Installation_Date, city_founding_date;
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_installation_date
```

```
BEFORE INSERT OR UPDATE ON SecuritySystem
FOR EACH ROW
EXECUTE FUNCTION check_installation_date();
```

7. Вывод

При выполнении лабораторной работы я познакомился с понятием нормализации и денормализации. Научился определять функциональные зависимости модели, а также анализировать последнюю на соответствие различным нормальным формам. Познакомился с процедурным языком PL/pgSQL. Изучил эффективные способы денормализации схемы базы данных и ситуации, в которых возможно их применение.