МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Санкт-Петербургский национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Информационные системы и базы данных»

Вариант № 1677

Выполнил:

Студент группы P3116

Билошицкий Михаил Владимирович

Преподаватель:

Горбунов Михаил Витальевич

Санкт-Петербург, 2023

Содержание

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.

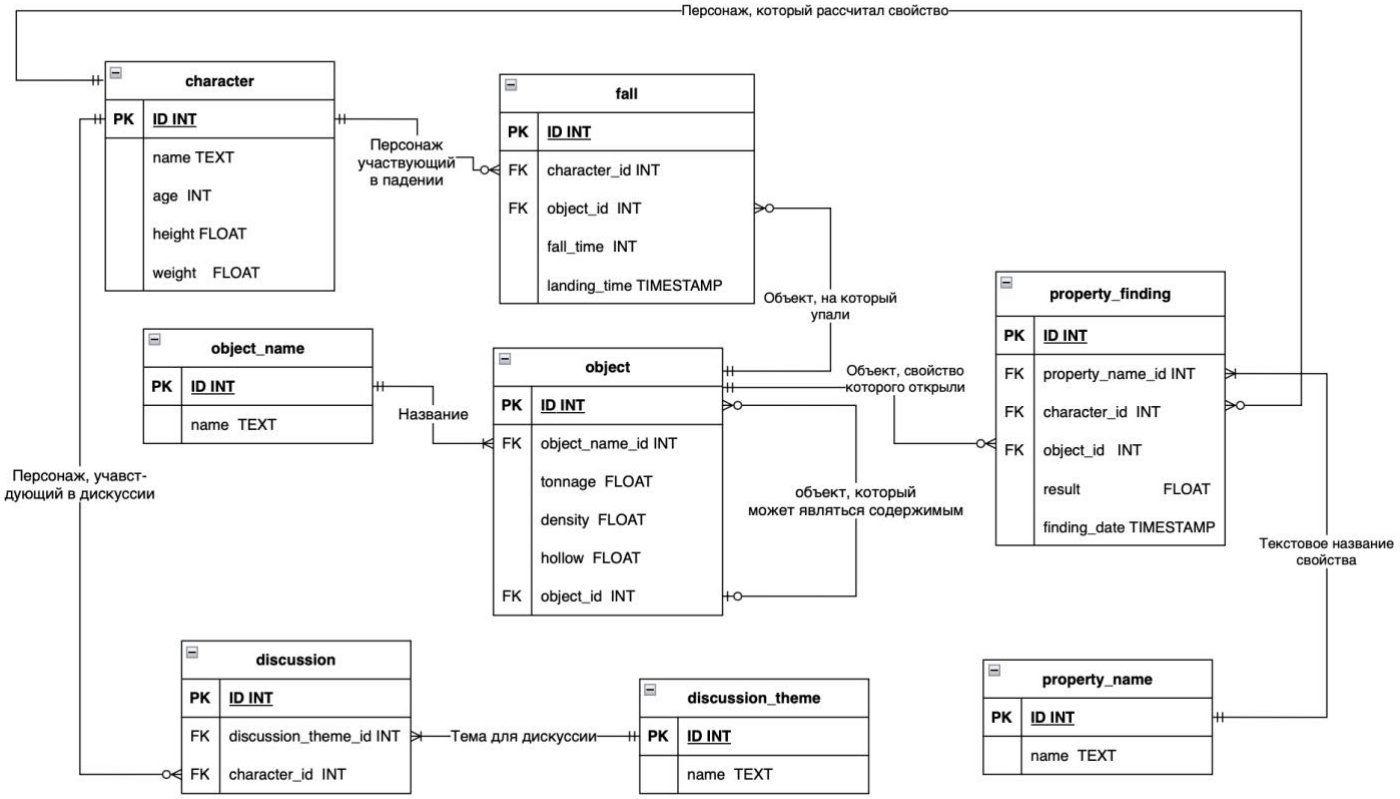
Отчёт по лабораторной работе должен содержать:

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

Исходная, нормализованная и денормализованная модели



Исходная модель

Функциональные зависимости:

```
character:
id -> name
id -> age
id -> height
id -> weight

fall (транзитивная зависимость):
id -> object_id
id -> character_id
id -> fall_time
object_id, character_id -> landing_time

object:
id -> object_name_id
id -> tonnage
id -> density
id -> hollow
id -> object_id

property_finding (транзитивная зависимость):
```

```
id -> property_name_id
id -> character_id
id -> object_id
id -> finding_date
property_name_id, character_id, object_id, finding_date -> result

discussion:
id -> discussion_theme_id
id -> character_id

discussion_theme:
id -> name

property_name:
id -> name

object_name:
id -> name
```

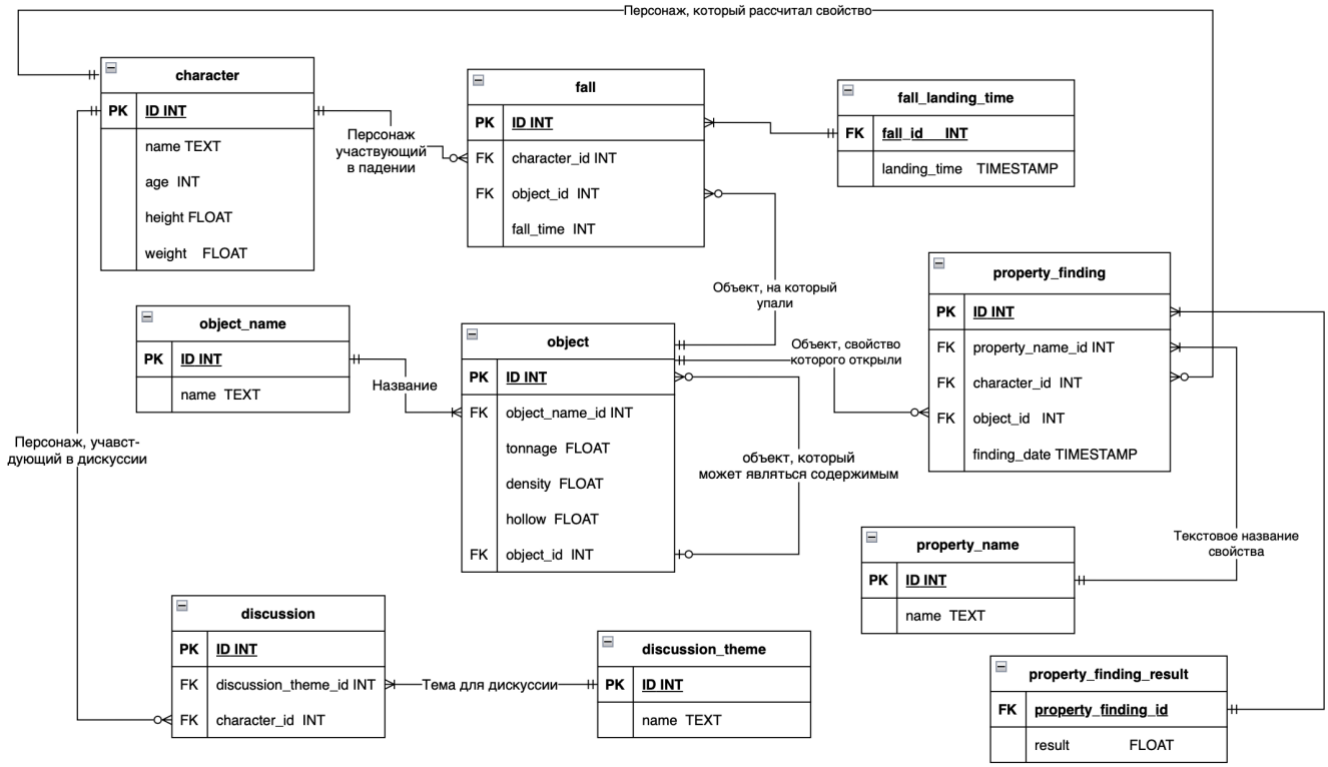
Ответы на вопросы

- Исходная схема соответствует 1НФ, поскольку на пересечении строк и столбцов не встречается нескольких значений.

- Исходная схема соответствует 2НФ, поскольку она соответствует 1НФ и атрибуты, не входящие в РК, находятся в полной функциональной зависимости от РК и нет частичных зависимостей от потенциальных ключей.
- Исходная схема не соответствует 3НФ, поскольку она соответствует 1НФ и 2 НФ, но не все атрибуты, не входящие в РК, не находятся в транзитивной зависимости от РК. Исправим схему и приведем ее к 3НФ:

    1. Разделим таблицу `fall` на таблицы `fall` и `fall_landing_time`, чтобы избежать транзитивной зависимости.
    2. Разделим таблицу `property_finding` на таблицы `property_finding` и `property_finding_result`, чтобы избежать транзитивной зависимости.

    В результате нормализации получим следующую схему с изменениями:



В результате нормализации получим следующие функциональные зависимости:

```
character:
id -> name
id -> age
id -> height
id -> weight

fall:
id -> object_id
id -> character_id
id -> fall_time

fall_landing_time:
fall_id -> landing_time

object:
id -> object_name_id
id -> tonnage
id -> density
id -> hollow
id -> object_id

property_finding:
id -> property_name_id
id -> character_id
id -> object_id
id -> finding_date

property_finding_result:
```

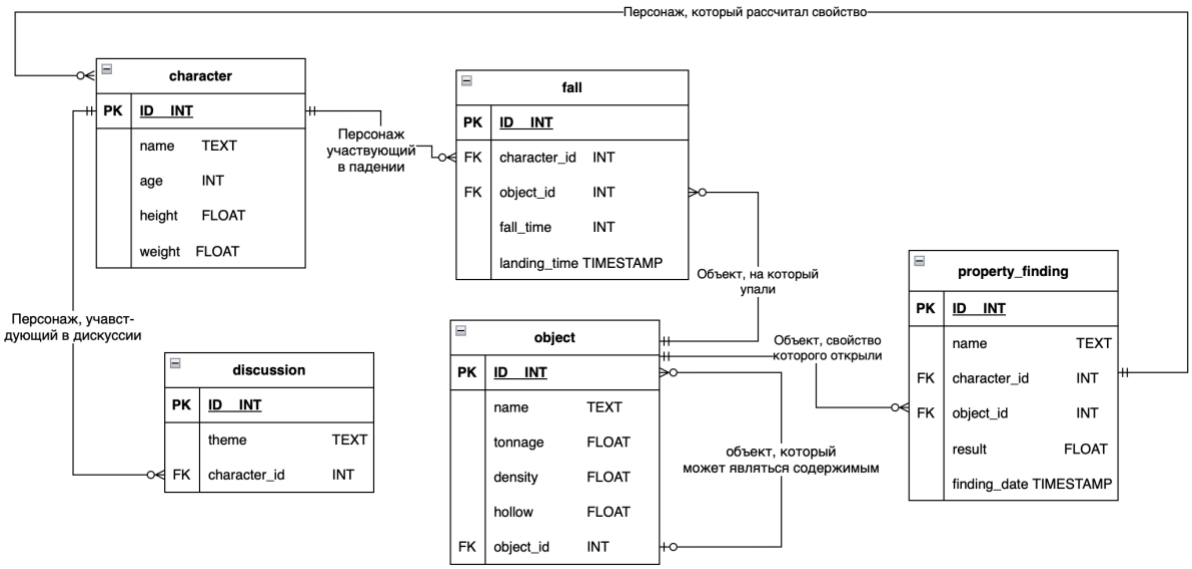
```
property_finding_id -> result

discussion:
id -> discussion_theme_id
id -> character_id

discussion_theme:
id -> name

property_name:
id -> name

object_name:
id -> name
```

- Нормализованная схема находится в BCNF, потому что выполняется условие BCNF – наличие 3НФ + во всех функциональных отношениях детерминанты вынесены как первичные ключи в отдельные таблицы. Столбцов, являющихся детерминантами за исключением первичного ключа, в таблицах не осталось.

  Полезные денормализации:

  1) Таблицы `discussion` и `discussion_theme` можно объединить во избежание постоянного объединения всех данных таблиц. Но минусом будет являться частая повторяемость данных.
  2) Таблицы `object` и `object_name` можно объединить во избежание постоянного объединения всех данных таблиц. Но минусом будет являться частая повторяемость данных.
  3) Таблицы `property_finding` и `property_name` можно объединить во избежание постоянного объединения всех данных таблиц. Но минусом будет являться частая повторяемость данных.
  4) Таблицы `fall` и `fall_landing_time` можно объединить во избежание постоянного объединения всех данных таблиц. Но минусом будет являться наличие транзитивной функциональной зависимости.
  5) Таблицы `property_finding` и `property_finding_result` можно объединить во избежание постоянного объединения всех данных таблиц. Но минусом будет являться наличие транзитивной функциональной зависимости.

  В результате денормализации получается следующая схема:



В результате денормализации получатся следующие отношения:

```
character:
id -> name
id -> age
id -> height
id -> weight

fall (транзитивная зависимость):
id -> object_id
id -> character_id
id -> fall_time
object_id, character_id -> landing_time

object:
id -> name
id -> tonnage
id -> density
id -> hollow
id -> object_id

property_finding (транзитивная зависимость):
id -> property_name_id
id -> character_id
id -> object_id
id -> finding_date
property_name_id, character_id, object_id, finding_date -> result

discussion:
id -> discussion_theme
id -> character_id
```

Функция и триггер

```
CREATE OR REPLACE FUNCTION property_finding_date_initialize() RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
    BEGIN
    if (new.finding_date IS NULL) THEN
        new.finding_date := current_timestamp AT TIME ZONE 'Europe/Moscow';
    end if;
    RETURN new;
END $$;

CREATE OR REPLACE TRIGGER property_finding_date_initialize_trigger
AFTER UPDATE OR INSERT ON property_finding FOR EACH ROW EXECUTE PROCEDURE
property_finding_date_initialize();
```

Мы создаем функцию, которая будет выполняться триггером на добавление или обновление нового свойства, где если дата нахождения свойства будет пустой, то триггер создаст ее.

Вывод

В рамках данной лабораторной работы были изучены нормальные формы. Проведен анализ модели на соответствие BCNF. Проведена денормализация с целью оптимизации выполнения запросов к таблице. Написана функция на языке PL/pgSQL.