Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Базы данных

Лабораторная работа № 3

Вариант № 1309

Выполнил:

Сандов Кирилл Алекссевич

Группа:

P3113

Проверил:

преподаватель практики Горбунов Михаил Витальевич

Санкт-Петербург

2023

Оглавление

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). Постройте схему на основе полученных отношений;
- 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.

Исходная модель

Изначальная модель (Рисунок 1).

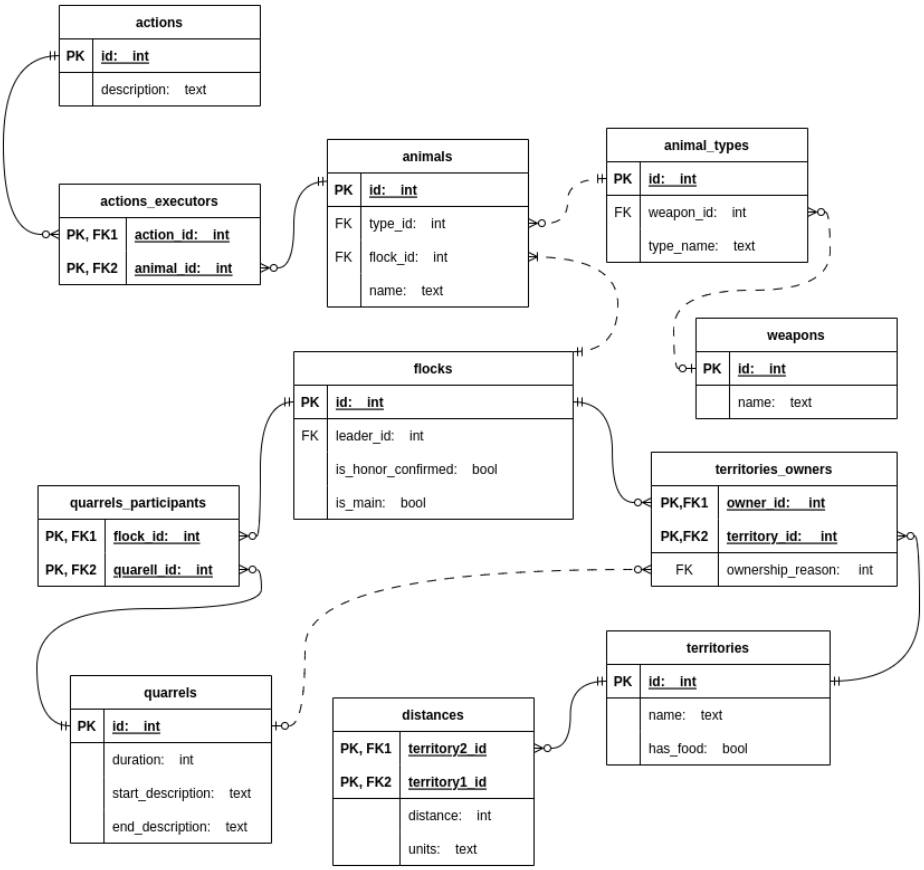

Рисунок 1

Нормализация модели

Перед началом нормализации выделим потенциальные естественные ключи у каждой таблицы, так как это понадобится на шагах нормализации к 2NF, 3NF и BCNF. Также опишем причины выбора таких ключей:

Таблица animals:

- (`type_id`, `flock_id`) - если учитывать, что в каждой стае животных зовут по-разному, то стая и имя определяют конкретного животного.

Таблица actions:

- (`description`) – каждое действие описывается по-разному.

Таблица animal_types:

- (`type_name`) – все виды животных имеют разные названия.

Таблица weapons:

- (`name`) – все оружия имеют разные названия.

Таблица flocks:

- (`leader_id`) – если учитывать, что один лидер может руководить только одной стаей, то каждый лидер определяет стаю.

Таблица territories:

- (`name`) – все территории имеют разные названия.

Таблица quarrels:

- (`duration`, `start_description`, `end_description`) – каждая перебранка чётко определяется своей продолжительностью, описанием начала и конца.

Сначала выделим основные функциональные зависимости модели:

Таблица animals:

- `id` → `flock_id`, `type_id`, `name`;
- `flock_id` → `type_id` (в каждой стае могут быть животные только одного вида, поэтому стая определяет вид животного).

Таблица animal_types:

- `id` → `weapon_id`, `type_name`;
- `type_name` → `weapon_id` (у каждого вида животного своё оружие).

Таблица flocks:

- `id` → `leader_id`, `is_honor_confirmed`, `is_main`;
- `leader_id` → `is_honor_confirmed`, `is_main` (у каждой стаи свой статус удовлетворения чести и основной роли в истории).

Таблица territories:

- `id` → `name`, `has_food`;
- `name` → `has_food` (у каждой территории свой статус наличия еды).

Таблица territories_owners:

- (`territory_id`, `owner_id`) → `ownership_reason`;

Таблица quarrels:

- `id` → `duration`, `start_description`, `end_description`;

Таблица distances:

- (`territory1_id`, `territory2_id`) → `distance`, `units` (две территории чётко определяют расстояние между ними и единицы этого расстояния).

1NF

Убедимся, что модель находится в 1NF. Этой действительно так, потому что у каждой таблицы есть ключ, и значение каждого атрибута может быть только одно.

2NF

Далее приведём модель к 2NF. Для этого нужно сделать такие декомпозиции, чтобы между ключом и каждым не ключевым атрибутом была полная функциональная

зависимость. Рассмотрев выписанные функциональные зависимости модели, можно найти нарушение этого правила:

- Таблица `animals` имеет составной естественный ключ (`flock_id`, `name`), однако нет полной функциональной зависимости между этим ключом и атрибутом `type_id`, так как есть функциональная зависимость `flock_id` → `type_id`.

  Решение: сделаем декомпозицию таблицы `animals` на две таблицы:
  `flock_types`(`flock_id`, `type_id`) – вид животного в каждой стае,
  `animals`(`id`, `flock_id`, `name`) – соответствие каждого животного и его стаи.

Остальные функциональные зависимости удовлетворяют правилу. Поэтому мы получили модель в 2NF.

3NF

Проверим то, что модель находится в 3NF. Для этого нужно убедиться, что отсутствуют транзитивные зависимости для любого атрибута A вида K → S → A, где K – суперключ, S – некоторое подмножество других атрибутов.

Просматривая функциональные зависимости можно обнаружить нарушения этого правила:

- Существует транзитивная зависимость: `id` → `flock_id` → `type_id`. Однако вспомним, что на шаге нормализации к 2NF мы сделали декомпозицию таблицы `animals` и вынесли атрибут `type_id` в другую таблицу. Соответственно, зависимость `flock_id` → `type_id` пропадает, и необходимости в декомпозиции нет.
- Есть множество нарушений, связанных с наличием суррогатных ключей, например, в таблице `flocks` есть транзитивная зависимость: `id` → `leader_id` → `is_honor_confirmed`. Эту проблему можно решить сделав декомпозицию `flocks` на:
  `flocks_leaders`(`id`, `leader_id`);
  `flocks`(`id`, `is_honor_confirmed`).

Однако композиции такого рода, выносящие суррогатный ключ, будут избыточными, поэтому будем игнорировать их.

Таким образом, модель приведена к 3NF.

BCNF

Модель будет находиться в BCNF, если для каждой функциональной зависимости её детерминант (левая часть) будет именно потенциальным ключом таблицы и ничем больше. Однако в данной модели (после декомпозиции на шаге приведения к 2NF) все функциональные зависимости соответствуют этому правилу. Значит, модель находится в BCNF.

Схема нормализованной модели

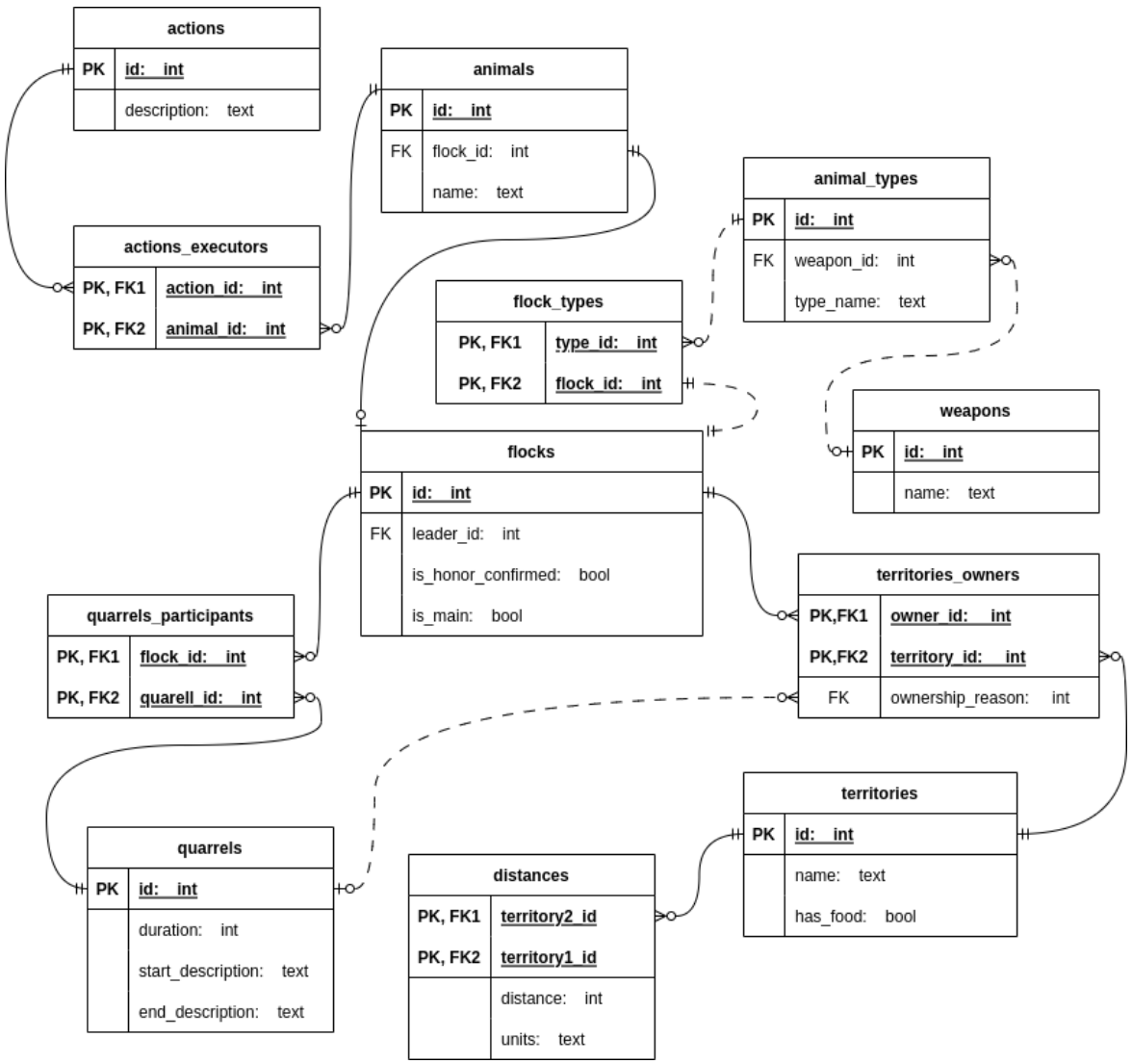

Рисунок 2

Денормализация модели

В процессе приведения модели к 2NF мы сделали декомпозицию таблицы `animals`, вынеся атрибут `type_id` в таблицу `flock_types`. Однако такая нормализация создаёт трудности для некоторых задач:

- Чтобы узнать вид определённого животного, нужно присоединить к `animals` таблицу `flock_types`;
- Чтобы узнать вид животных в стае нужно, нужно присоединить к `flocks` таблицу `flock_types`;
- Если потребуется удалить некоторый вид животного из таблицы `animal_types`, то придётся сначала удалить записи о нём в `flock_types`, затем каскадно удалить соответствие этим записям в `flocks`, а затем каскадно удалить соответствия в `animals`.

Всё это приводит к снижению производительности некоторых запросов. Поэтому можно соединить таблицы `animals` и `flock_types`, но при этом таблица не будет соответствовать 2NF.

Но есть и альтернативное решение, которое не будет являться денормализацией, так как после него модель останется в BCNF. Можно внести атрибут `type_id` в таблицу `flocks`. Он не будет никак функционально зависеть от других неключевых атрибутов, поэтому никаких нарушений правил нормализации до BCNF не будет. Полученая модель будет выглядеть так (Рисунок 3):

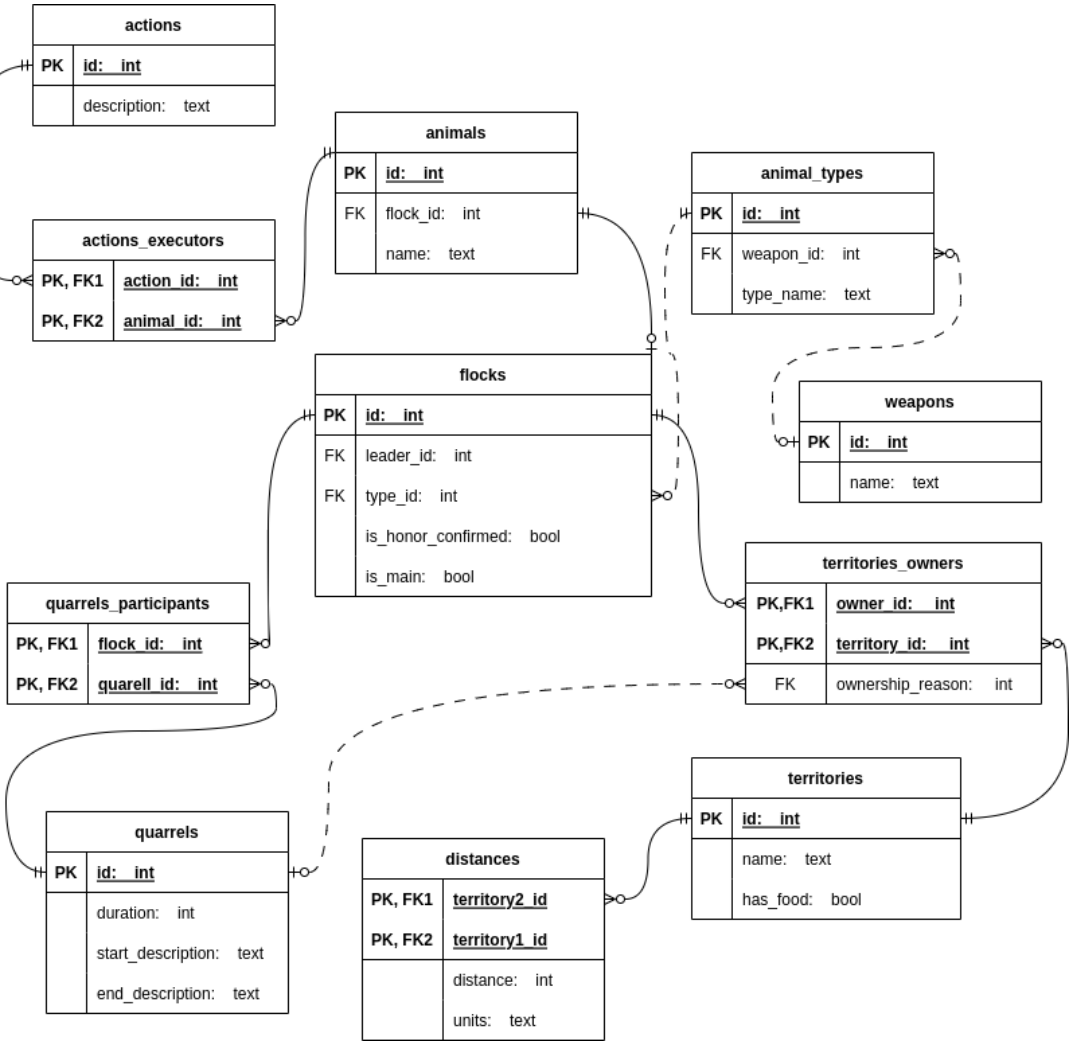
actions
PK id: int
description: text
animals
PK id: int
FK flock_id: int
name: text
animal_types
PK id: int
FK weapon_id: int
type_name: text
actions_executors
PK, FK1 action_id: int
PK, FK2 animal_id: int
flocks
PK id: int
FK leader_id: int
FK type_id: int
is_honor_confirmed: bool
is_main: bool
weapons
PK id: int
name: text
territories_owners
PK,FK1 owner_id: int
PK,FK2 territory_id: int
FK ownership_reason: int
quarrels_participants
PK, FK1 flock_id: int
PK, FK2 quarell_id: int
territories
PK id: int
name: text
has_food: bool
quarrels
PK id: int
duration: int
start_description: text
end_description: text
distances
PK, FK1 territory2_id
PK, FK2 territory1_id
distance: int
units: text

Рисунок 3

Функция

Описание функции

Функция `min_dist(integer a, integer b)` находит длину кратчайшего пути между территориями с id, равными a и b, в таблице `distances`. Если пути между этими территориями нет, то она возвращает NULL.

Для демонстрации работы этой функции в связке с триггером была создана таблица `min_distances`. В ней пользователь может добавить две территории, длину расстояния между которыми он хочет отслеживать. Далее для них в этой таблице автоматически генерируется значение поля `min_dist` на основе вышеприведённой функции.

Был создан триггер `distances_update`, срабатывающий при любом изменении таблицы `distances` (удаление, обновление, добавление строк). Он вызывает процедуру `update_min_distances`, которая в свою очередь заставляет все отслеживаемые минимальные расстояния в таблице `min_distances` быть пересчитанными.

Реализация

Исходный код реализации функций и триггера на языке PL/pgSQL можно увидеть в GitHub-репозитории по ссылке:

https://github.com/amphyxs/vt-labas/tree/main/sem-2/db/lab-3/min_dist.sql

Заключение

В результате выполнения данной лабораторной работы изучена теория по нормализации таблиц. Во-первых, рассмотрены нормальные формы отношений, каждая из которых позволяет исключать различные аномалии при вставке, обновлении, изменении данных. Нормализация была применена на практике: модель из прошлой лабораторной работы была приведена к нормальной форме Бойса-Кодда. Получены знания о денормализации таблиц для повышения эффективности операций работы с базой данных. Также был изучен язык PL/pgSQL, который позволяет реализовать функции для использования внутри БД. Была написана собственная функция на этом языке, а также триггер, который вызывал функцию при изменении некоторой таблицы.