Министерство высшего образования и науки Российской Федерации

Национальный научно-исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

**«БАЗЫ ДАННЫХ».**

Вариант №696.

Выполнил:

Петров Вячеслав Маркович,

Студент группы P3108.

Преподаватель:

Афанасьев Дмитрий Борисович

Санкт-Петербург, 2024

Оглавление

[Текст задания 3](#_Toc164348110)

[Даталогическая модель (исходная) 4](#_Toc164348111)

[Функциональные зависимости (изначальные) 5](#_Toc164348112)

[Преобразование к 1НФ 5](#_Toc164348113)

[Преобразование к 2НФ 5](#_Toc164348114)

[Преобразование к 3НФ 6](#_Toc164348115)

[Преобразование к BCNF 6](#_Toc164348116)

[Денормализация 6](#_Toc164348117)

[Функциональные зависимости (после преобразований) 7](#_Toc164348118)

[Даталогическая модель (после преобразований) 8](#_Toc164348119)

[Триггер и функция 9](#_Toc164348120)

[Выводы по работе 10](#_Toc164348121)

# 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 основе NF (как минимум). Постройте схему на основе полученных отношений;
* опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе NF;
* преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF;
* 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)

Можно сравнить начальную (Рисунок 1) и полученную (Рисунок 2) модели.

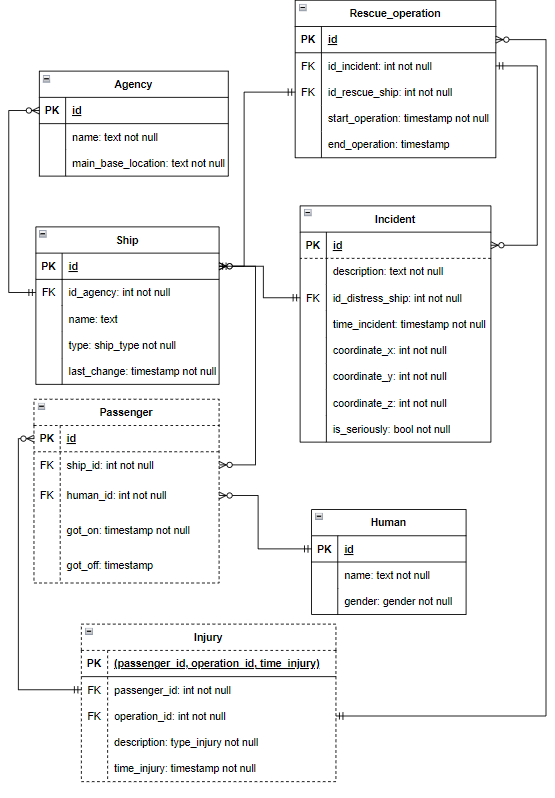


Рисунок 1 - Исходная модель

# Функциональные зависимости (изначальные)

Human:

id → (name, gender)

Agency:

id → (name, main\_base\_location)

Ship:

id → (id\_agency, name, type, last\_change)

Incident:

id → (description, id\_distress\_ship, time\_incident, coordinate\_x, coordinate\_y, coordinate\_z, is\_seriously)

Rescue\_operation:

id → (id\_incident, id\_rescue\_ship, start\_operation, end\_operation)

Passenger:

id → (ship\_id, human\_id, got\_on, got\_off)

Injury:

(passenger\_id, operation\_id, time\_injury) → (description)

# Преобразование к 1НФ

Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. Не потребовалось, условие “на пересечении каждой строки и столбца – 1 значение” и так выполнялось.

# Преобразование к 2НФ

Отношение находится во 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от первичного ключа. Не потребовалось, поскольку у всех первичных ключей нет подмножеств, а значит атрибуты всех отношений – в полной функциональной зависимости от соответствующих первичных ключей.

# Преобразование к 3НФ

В некоторых отношениях наблюдались транзитивные зависимости:

* agency: id -> name, name -> main\_base\_location, id -> main\_base\_location;

По сути, транзитивность здесь даёт то, что в этих отношениях как бы два первичных ключа, только один указан явно (id), а другой получается в силу того, что значения атрибута должно быть уникальными (name). Поэтому:

* в agency убираем id и делаем первичным ключом name.

# Преобразование к BCNF

Отношение находится в BCNF, если для каждой функциональной зависимости X →Y, X является потенциальным ключом. Моя модель удовлетворяет BCNF, так как для всех функциональных зависимостей X является потенциальным ключом.

# Денормализация

Объединение таблиц может ускорить обработку запросов. Например, можно рассмотреть объединение таблиц incident и rescue\_operation, если часто запрашиваются данные об инциденте и спасательной операции, прилагающейся к нему, одновременно.

Также можно добавить несколько избыточных атрибутов, что может улучшить производительность запросов. Например, если часто запрашивается количество погибших людей во время инцидента, можно добавить атрибут number\_of\_deaths в таблицу Incident. Это позволит избежать операций подсчета при каждом запросе, однако необходимо будет обновлять этот атрибут при добавлении Injury с атрибутом description = «не совместимые с жизнью».

# Функциональные зависимости (после преобразований)

Human:

id → (name, gender)

Agency:

name → (main\_base\_location)

Ship:

id → (id\_agency, name, type, last\_change)

Incident:

id → (description, id\_distress\_ship, time\_incident, coordinate\_x, coordinate\_y, coordinate\_z, is\_seriously, id\_rescue\_ship, start\_operation, end\_operation)

Passenger:

id → (ship\_id, human\_id, got\_on, got\_off)

Injury:

(passenger\_id, incident\_id, time\_injury) → (description)

# Даталогическая модель (после преобразований)

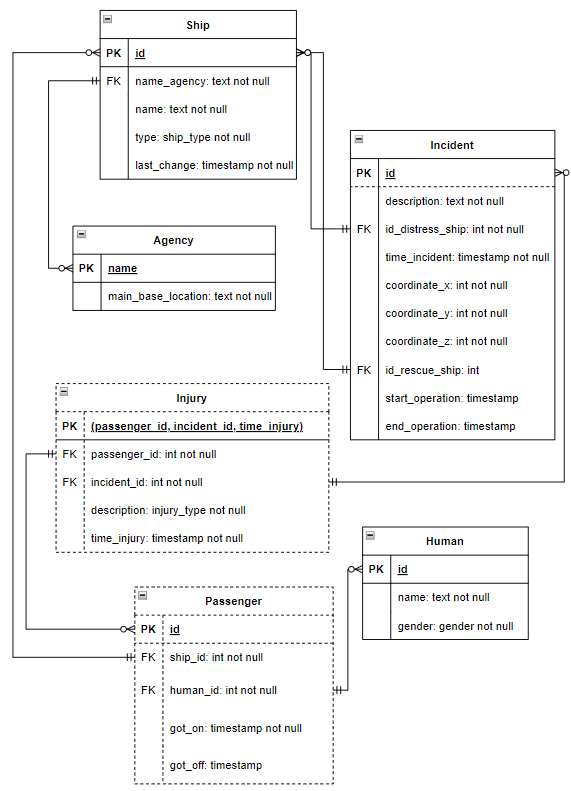
Можно сравнить начальную (Рисунок 1) и полученную (Рисунок 2) модели.

Рисунок 2 - получившаяся модель

# Триггер и функция

Триггер вызывает функцию, которая проверяет тип спасательного корабля, и если он «distress», то не добавляем запись, выкидывая исключение. Также, если ошибки нет, то обновляем поля спасательного и терпящего бедствие кораблей

CREATE OR REPLACE FUNCTION check\_and\_update\_type\_ship() RETURNS trigger AS

$$

declare

type\_of\_rescue\_ship ship\_type;

BEGIN

SELECT ship.type INTO type\_of\_rescue\_ship FROM ship WHERE ship.id = NEW.id\_rescue\_ship;

IF (type\_of\_rescue\_ship = 'distress') then

RAISE EXCEPTION 'Терпящий бедствие корабль не может прийти на помощь';

end if;

UPDATE ship

SET type = 'distress',

last\_change = now()

WHERE id = NEW.id\_distress\_ship;

UPDATE ship

SET type = 'rescue',

last\_change = now()

WHERE id = NEW.id\_rescue\_ship;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_and\_update\_type

BEFORE INSERT OR UPDATE

ON incident

EXECUTE FUNCTION check\_and\_update\_type\_ship();

# Выводы по работе

В ходе выполнения данной лабораторной работы я познакомился с различными нормальными формами и процессом приведения к ним, а также с процессом денормализации модели и написанием параметризированных функций на языке plpgsql.