ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

Факультет программной инженерии и компьютерной техники

отчет по лабораторной работе №1

Вариант 87664

Выполнила: Шмидт А. А, Группа Р3115

Преподаватель:

Райла Мартин

Санкт-Петербург

2023

Оглавление

[Задание 3](#_Toc165307511)

[Выполнение задания 4](#_Toc165307512)

# Задание

Изображение выглядит как текст, чек, алгебра

Автоматически созданное описание

# Выполнение заданияИзображение выглядит как текст, снимок экрана, диаграмма, План Автоматически созданное описание

1. Функциональные зависимости

Tomb -> id\_floor, id\_ceiling, name

Ceiling -> id\_material, name

Floor -> id\_material, name

Tomb\_material -> id\_tomb, id\_material

Material -> name

1. Анализ зависимостей

1НФ:

Отношение на пересечении каждой строки и столбца – одно значение.

Представленная модель удовлетворяет 1НФ

2НФ:

1. Отношение в 1НФ
2. Атрибуты, не входящие в первичный ключ, в полной функциональной зависимости от первичного ключа отношения

Представленная модель удовлетворяет 2НФ

3НФ:

1. Отношение в 2НФ
2. Все атрибуты, которые не входят в первичный ключ, не находятся в транзитивной функциональной зависимости от первичного ключа

Представленная модель удовлетворяет 3НФ

НФБК:

Детерминанты всех функциональных зависимостей являются потенциальными ключами

Представленная модель удовлетворяет НФБК

1. Денормализация

В случае, если данных в таблице «Tomb» слишком мало, может быть применено объединение таблиц «ceiling» и «floor» с таблицей «tomb», для сокращения количества бесполезных таблиц для таких размеров.

1. Функция и триггер

Обновляет здоровье человека и ловушек, проверяет на рабочее состояние ловушку/этаж/комнату/гробницу

Дополнение БД, создание триггера и функции:

CREATE TABLE trap (  
 id serial PRIMARY KEY,  
 health int NOT NULL,  
 condition boolean  
);

CREATE TABLE room (  
 id serial PRIMARY KEY,  
 id\_floor int NOT NULL REFERENCES tombs\_floor(id),  
 condition boolean  
);  
  
CREATE TABLE room\_trap (  
 id\_room int NOT NULL REFERENCES room(id),  
 id\_trap int NOT NULL REFERENCES trap(id),  
 PRIMARY KEY (id\_room, id\_trap)  
);  
  
CREATE TABLE tombs\_floor (  
 id serial PRIMARY KEY,  
 id\_tomb int NOT NULL REFERENCES tomb(id),  
 condition boolean  
);

create table people(id serial primary key,  
 health int not null,  
 id\_tombs\_floor int references tombs\_floor(id),   
 id\_tombs\_room int references room(id));  
alter table people add id\_trap int references trap(id);

alter table people add condition boolean

CREATE OR REPLACE FUNCTION *update\_health\_and\_condition*()  
RETURNS TRIGGER AS $$  
BEGIN  
 -- Получаем состояние ловушки, с которой столкнулся человек  
 IF TG\_TABLE\_NAME = 'people' THEN  
 -- Получаем состояние ловушки, с которой столкнулся человек  
 IF *EXISTS* (SELECT 1 FROM trap WHERE id = NEW.id\_trap AND condition = TRUE) THEN  
 -- Уменьшаем здоровье человека на 1  
 NEW.health := NEW.health - 1;  
  
 -- Если здоровье человека стало 0 или меньше, деактивируем его  
 IF NEW.health <= 0 THEN  
 NEW.condition := FALSE;  
 END IF;  
 END IF;  
 ELSEIF TG\_TABLE\_NAME = 'trap' THEN  
 -- Уменьшаем здоровье ловушки на 1  
 NEW.health := NEW.health - 1;  
  
 -- Если здоровье ловушки стало 0, деактивируем ее  
 IF NEW.health <= 0 THEN  
 NEW.condition := FALSE;  
 END IF;  
 END IF;  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
-- Создаем триггер для таблицы people  
CREATE TRIGGER update\_people\_health\_trigger  
BEFORE UPDATE ON people  
FOR EACH ROW  
EXECUTE FUNCTION *update\_health\_and\_condition*();  
  
-- Создаем триггер для таблицы trap  
CREATE TRIGGER update\_trap\_health\_trigger  
BEFORE UPDATE ON trap  
FOR EACH ROW  
EXECUTE FUNCTION *update\_health\_and\_condition*();

CREATE OR REPLACE FUNCTION *update\_condition*()  
RETURNS TRIGGER AS $$  
DECLARE  
 room\_count INT;  
 trap\_count INT;  
BEGIN  
 -- Получаем количество активных комнат на этаже  
 SELECT *COUNT*(\*) INTO room\_count FROM room WHERE id\_floor = NEW.id AND condition = TRUE;  
  
 -- Если количество активных комнат на этаже стало 0, деактивируем этаж  
 IF room\_count = 0 THEN  
 UPDATE tombs\_floor SET condition = FALSE WHERE id = NEW.id;  
 ELSE  
 -- Получаем количество активных ловушек в комнате  
 SELECT *COUNT*(\*) INTO trap\_count  
 FROM room\_trap rt  
 INNER JOIN trap t ON rt.id\_trap = t.id  
 WHERE rt.id\_room = NEW.id AND t.condition = TRUE;  
  
 -- Если количество активных ловушек в комнате стало 0, деактивируем комнату  
 IF trap\_count = 0 THEN  
 UPDATE room SET condition = FALSE WHERE id = NEW.id;  
 END IF;  
 END IF;  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
  
-- Создаем триггер для таблицы tombs\_floor  
CREATE TRIGGER update\_tombs\_floor\_condition\_trigger  
AFTER INSERT ON tombs\_floor  
FOR EACH ROW  
EXECUTE FUNCTION *update\_condition*();  
  
-- Создаем триггер для таблицы room  
CREATE TRIGGER update\_room\_condition\_trigger  
AFTER INSERT ON room  
FOR EACH ROW  
EXECUTE FUNCTION *update\_condition*();

CREATE OR REPLACE FUNCTION *update\_people\_trap\_id*()  
RETURNS TRIGGER AS $$  
BEGIN  
 -- Проверяем, если здоровье ловушки стало равным 0  
 IF NEW.health = 0 THEN  
 -- Увеличиваем id\_trap на 1 для всех записей в таблице people  
 UPDATE people SET id\_trap = id\_trap + 1;  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER update\_people\_trap\_id\_trigger  
AFTER UPDATE OF health ON trap  
FOR EACH ROW  
EXECUTE FUNCTION *update\_people\_trap\_id*();