Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерных технологий

Базы данных

Отчёт по лабораторной работе №3

Вариант 55443

Выполнил:

*Чумаченко Д.О.*

*P3115*

Проверил:

*Мартин Райла*

Санкт-Петербург 2024

Оглавление

[Текст задания 3](#_Toc165228811)

[Модели 4](#_Toc165228812)

[Исходная 4](#_Toc165228813)

[Нормализованная 4](#_Toc165228814)

[Денормализованная 5](#_Toc165228815)

[Ответы на вопросы 6](#_Toc165228816)

[Дополнительное задание 7](#_Toc165228817)

[Новая диаграмма 7](#_Toc165228818)

[Скрипты, использованные для реализации 7](#_Toc165228819)

[new\_schema 7](#_Toc165228820)

[trigger&function 8](#_Toc165228821)

[clearing\_tables 9](#_Toc165228822)

[drop 9](#_Toc165228823)

[Выводы по работе 10](#_Toc165228824)

# Текст задания

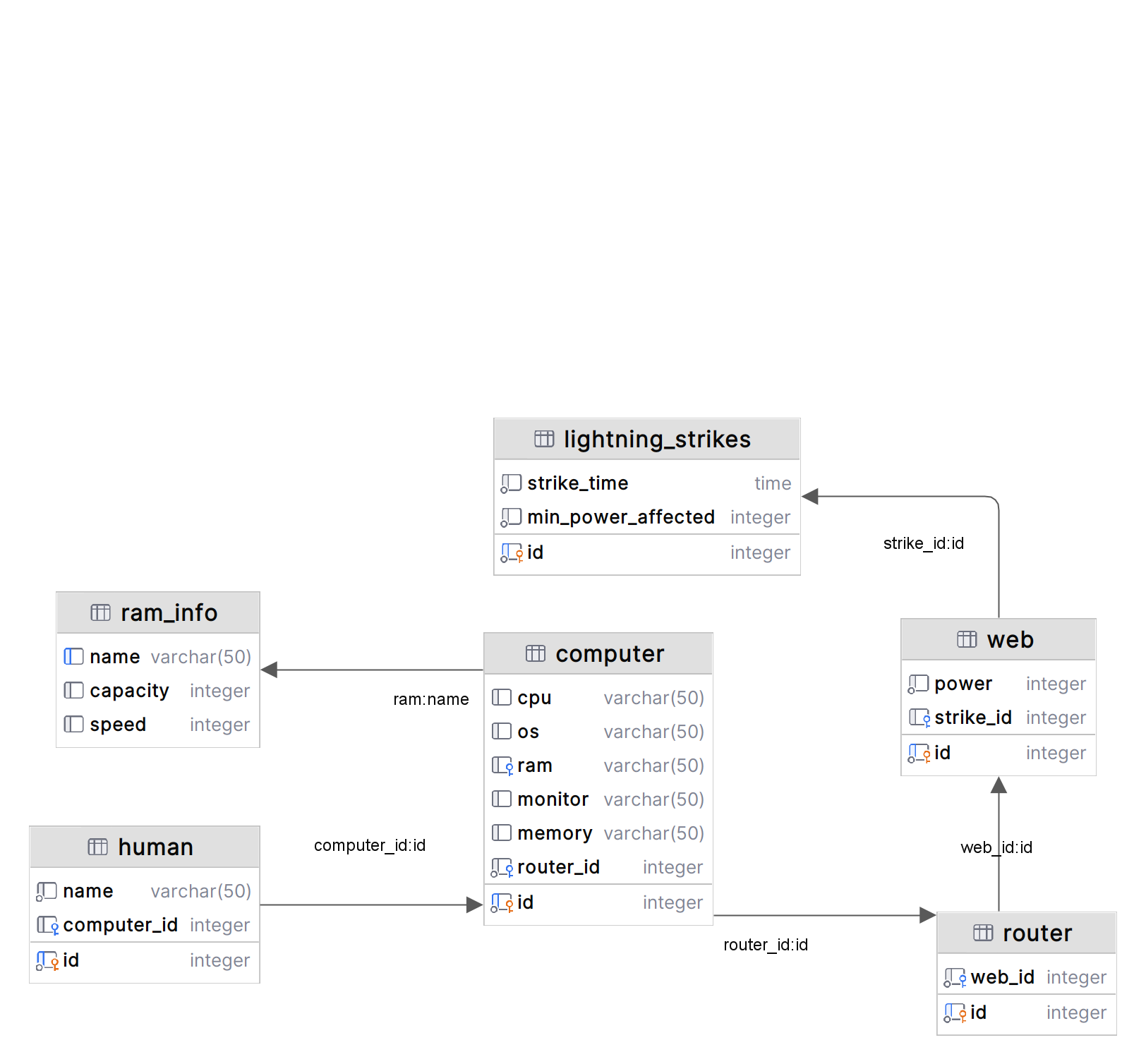
Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

* Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
* Приведите отношения в 3NF (как минимум). Постройте схему на 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.

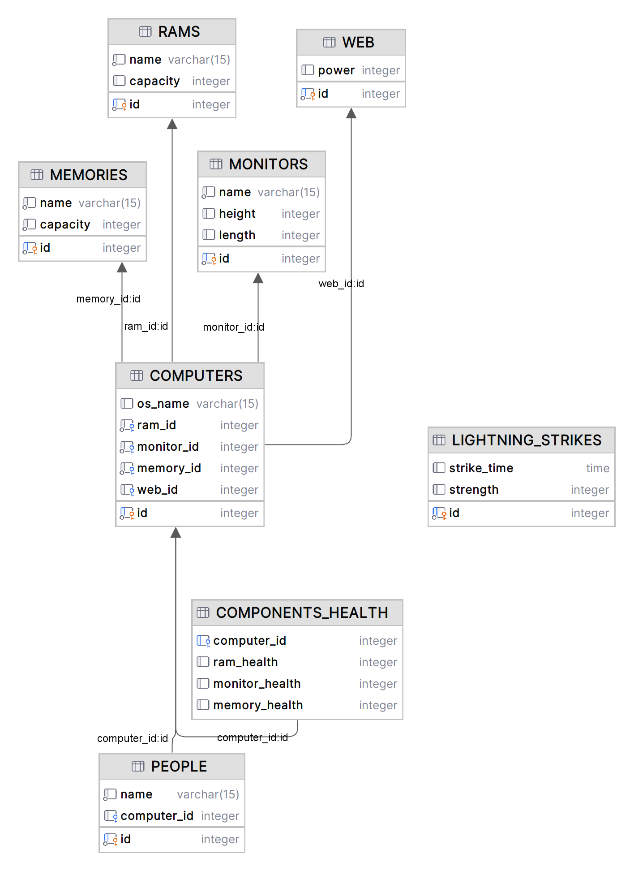
Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

# Модели

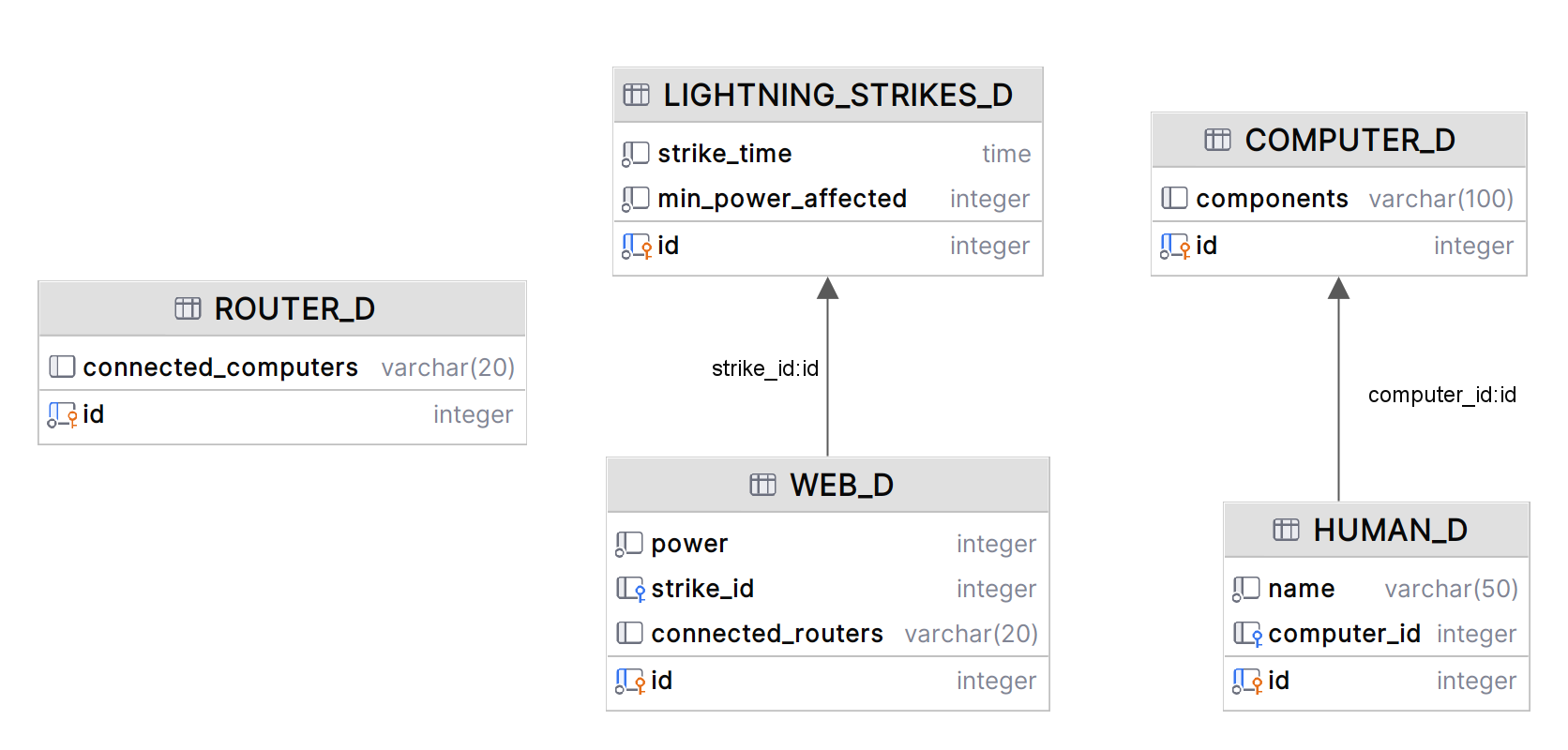
## Исходная



## Нормализованная



## Денормализованная



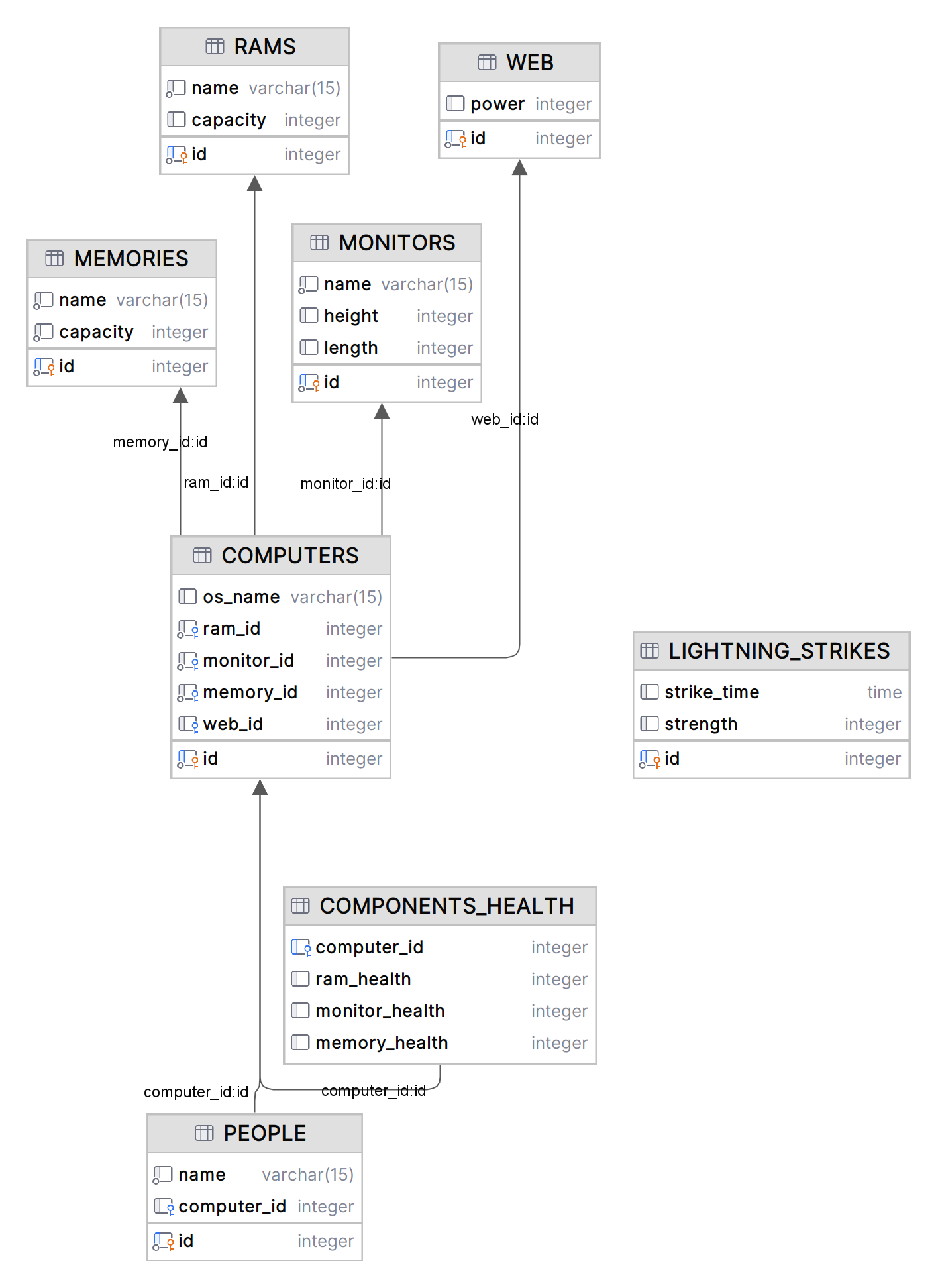
# Ответы на вопросы

Изначальная схема уже находилась в формате BCNF, единственное изменение – это отвязка молний от сети, так как связь устанавливается теперь не вручную, а через триггер.

1. Все атрибуты являются простыми, нет повторений строк в таблице.
2. Каждый не ключевой атрибут неприводимо зависит от первичного ключа.
3. Каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.
4. Каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

# Дополнительное задание

## Новая диаграмма



## Скрипты, использованные для реализации

### new\_schema

SET SEARCH\_PATH = "schema\_after\_dop";  
  
CREATE TABLE "LIGHTNING\_STRIKES"  
(  
 id SERIAL PRIMARY KEY,  
 strike\_time TIME DEFAULT *now*() + interval '3 hours',  
 strength INTEGER CHECK ( strength between 0 AND 100 ) DEFAULT 50  
);  
  
CREATE TABLE "WEB"  
(  
 id SERIAL PRIMARY KEY,  
 power INTEGER CHECK ( power BETWEEN 0 AND 1000 )  
);  
  
CREATE TABLE "RAMS"  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(15) NOT NULL,  
 capacity INTEGER CHECK ( capacity > 0 )  
);  
  
CREATE TABLE "MONITORS"  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(15) NOT NULL,  
 height INTEGER CHECK ( height > 0 ),  
 length INTEGER CHECK ( length > 0 )  
);  
  
CREATE TABLE "MEMORIES"  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(15) NOT NULL,  
 capacity INTEGER CHECK ( capacity > 0 ) NOT NULL  
);  
  
CREATE TABLE "COMPUTERS"  
(  
 id SERIAL PRIMARY KEY,  
 os\_name VARCHAR(15),  
 ram\_id INTEGER NOT NULL,  
 FOREIGN KEY (ram\_id) REFERENCES "RAMS" (id),  
 monitor\_id INTEGER NOT NULL,  
 FOREIGN KEY (monitor\_id) REFERENCES "MONITORS" (id),  
 memory\_id INTEGER NOT NULL,  
 FOREIGN KEY (memory\_id) REFERENCES "MEMORIES" (id),  
 web\_id INTEGER,  
 FOREIGN KEY (web\_id) REFERENCES "WEB" (id)  
);  
  
CREATE TABLE "PEOPLE"  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(15) NOT NULL,  
 computer\_id INTEGER,  
 FOREIGN KEY (computer\_id) REFERENCES "COMPUTERS" (id)  
);  
  
  
CREATE TABLE "COMPONENTS\_HEALTH" (  
 computer\_id INTEGER UNIQUE,  
 FOREIGN KEY (computer\_id) REFERENCES "COMPUTERS" (id),  
 ram\_health INTEGER CHECK ( ram\_health BETWEEN 0 AND 100 ) DEFAULT 100,  
 monitor\_health INTEGER CHECK ( monitor\_health BETWEEN 0 AND 100 ) DEFAULT 100,  
 memory\_health INTEGER CHECK ( memory\_health BETWEEN 0 AND 100 ) DEFAULT 100  
);

### trigger&function

SET SEARCH\_PATH = "schema\_after\_dop";  
  
CREATE OR REPLACE FUNCTION *damage\_computers*()  
 RETURNS TRIGGER AS  
$$  
DECLARE  
 affected\_web INTEGER = (SELECT (*count*(*\**) \* *random*() + 1)::INTEGER  
 FROM "WEB") ;  
 damage INTEGER = (new.strength \* *random*() \* (SELECT "WEB".power  
 FROM "WEB"  
 WHERE id = affected\_web)) / 1000;  
BEGIN  
 UPDATE "COMPONENTS\_HEALTH"  
 SET ram\_health = (  
 CASE  
 WHEN *random*() > 0.5  
 THEN (CASE WHEN ram\_health - damage > 0 THEN ram\_health - damage ELSE 0 END)  
 ELSE ram\_health END  
 ),  
 monitor\_health = (  
 CASE  
 WHEN *random*() > 0.5  
 THEN (CASE WHEN monitor\_health - damage > 0 THEN monitor\_health - damage ELSE 0 END)  
 ELSE monitor\_health END  
 ),  
 memory\_health = (  
 CASE  
 WHEN *random*() > 0.5  
 THEN (CASE WHEN memory\_health - damage > 0 THEN memory\_health - damage ELSE 0 END)  
 ELSE memory\_health END  
 )  
 WHERE *EXISTS*(SELECT *\** FROM "COMPUTERS"  
 WHERE "COMPUTERS".id = "COMPONENTS\_HEALTH".computer\_id  
 AND "COMPUTERS".web\_id = affected\_web);  
 RETURN NULL;  
end;  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER lightning\_strike  
 AFTER INSERT  
 ON "LIGHTNING\_STRIKES"  
 FOR EACH ROW  
EXECUTE FUNCTION *damage\_computers*();  
  
  
*-- some new triggers & functions*CREATE OR REPLACE FUNCTION *add\_components\_info*()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 INSERT INTO "COMPONENTS\_HEALTH" (computer\_id)  
 VALUES (new.id);  
 RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
  
CREATE OR REPLACE FUNCTION *remove\_components\_info*()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 DELETE FROM "COMPONENTS\_HEALTH" WHERE computer\_id = old.id;  
 RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
  
CREATE OR REPLACE TRIGGER new\_computer  
 AFTER INSERT  
 ON "COMPUTERS"  
 FOR EACH ROW  
EXECUTE FUNCTION *add\_components\_info*();  
  
  
CREATE OR REPLACE TRIGGER computer\_disappears  
 BEFORE DELETE  
 ON "COMPUTERS"  
 FOR EACH ROW  
EXECUTE FUNCTION *remove\_components\_info*();

### clearing\_tables

SET SEARCH\_PATH = "schema\_after\_dop";  
TRUNCATE TABLE "COMPUTERS", "LIGHTNING\_STRIKES", "MEMORIES", "MONITORS", "PEOPLE", "RAMS", "WEB", "COMPONENTS\_HEALTH";  
  
ALTER SEQUENCE "COMPUTERS\_id\_seq" RESTART;  
ALTER SEQUENCE "LIGHTNING\_STRIKES\_id\_seq" RESTART;  
ALTER SEQUENCE "MEMORIES\_id\_seq" RESTART;  
ALTER SEQUENCE "MONITORS\_id\_seq" RESTART;  
ALTER SEQUENCE "PEOPLE\_id\_seq" RESTART;  
ALTER SEQUENCE "RAMS\_id\_seq" RESTART;  
ALTER SEQUENCE "WEB\_id\_seq" RESTART;

### drop

SET SEARCH\_PATH = "schema\_after\_dop";  
DROP TABLE IF EXISTS "COMPUTERS", "LIGHTNING\_STRIKES", "MEMORIES", "MONITORS", "PEOPLE", "RAMS", "WEB", "COMPONENTS\_HEALTH" CASCADE;

# Выводы по работе

В ходе работы я познакомился с нормализацией и денормализацией схем, с триггерами и функциями, а также научился использовать их.