Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №3**

по дисциплине «Базы данных»

**Вариант: 3505**

Выполнил:

студент 1 курса

Батманов Даниил Евгеньевич

Группа: Р3107

Приняла:

Бострикова Дарья Константиновна

Отчёт принят «\_\_»\_\_\_\_\_2022 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2023

**Оглавление**

[Задание 3](#_Toc129812876)

[Основные этапы работы 5](#_Toc129812877)

[Заключение 10](#_Toc129812878)

# 

# 

# 

# 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 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;

Если ваша схема находится уже в BCNF, докажите это.

Какие денормализации будут полезны для вашей схемы? Приведите подробное описание;

Придумайте функцию, связанную с вашей предметной областью, согласуйте ее с преподавателем и реализуйте на языке PL/pgSQL.

# Даталогическая модель предметной области:

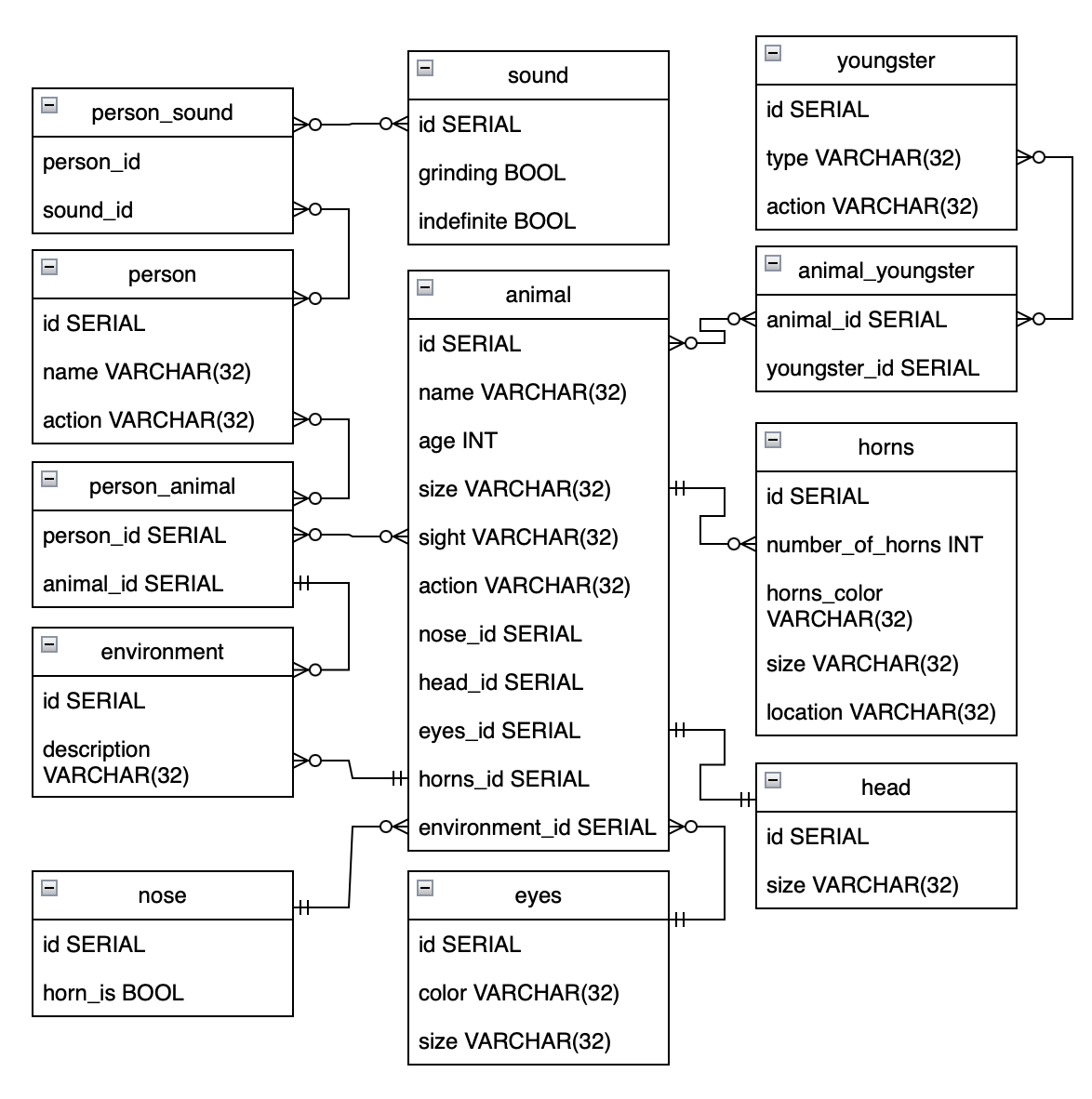


Рисунок 1

# Основные этапы работы

**Функциональные зависимости:**

* person: id –> (name, action, environment\_id)
* sound: id –> (grinding, indefinite)
* animal: id –> (name, age, size, sight, action, nose\_id, head\_id, eyes\_id, horns\_id, environment\_id)
* environment: id –> (description)
* head: id –> (size)
* eyes: id –> (color, size)
* horns: id –> (number\_of\_horns, horns\_colour, size, location)
* nose: id –> (horn\_is)
* youngster: id –> (type, action)
* person\_sound: (person\_id, sound\_id) –> ()
* animal\_youngster: (animal\_id, youngster\_id) –> ()
* person\_animal: (person\_id, animal\_id) –> ()

**Нормальные формы:**

* **1NF:** Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. В моём случае все атрибуты атомарны, нет повторяющихся групп, потому моя модель удовлетворяет 1NF
* **2NF:** Отношение находится во 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от первичных ключей. В моём случае все неключевые аттрибуты полностью функционально зависят от первичных ключей, потому моя модель удовлетворяет 2NF
* **3NF:** Отношение находится в 3NF, если оно находится во 2NF и не содержит транзитивных зависимостей (если у нас есть два атрибута А и В, и В зависит от С, который зависит от А, то мы должны разбить таблицу на две или более таблицы). В моём случае все неключевые атрибуты зависят только от первичных ключей, не содержат транзитивных зависимостей, потому моя модель удовлетворяет 3NF

**BCNF:**

1. Отношение находится в BCNF, если для каждой функциональной зависимости X –> Y, X является суперключом (ключевые атрибуты не должны зависеть от ключевых). В моём случае для всех функциональных зависимостей X является суперключом, потому моя модель удовлетворяет BCNF

**Денормализация:**

* В таблицу animal добавить столбцы eyes\_color, eyes\_size, horns\_number\_of, horns\_color, horns\_size, horns\_location, nose\_horn\_is, head\_size, и скопировать в них соответствующие данные из таблиц eyes, horns, head и nose соответственно. Это избавит от необходимости использования соединений при запросе информации о животных и их особенностях.

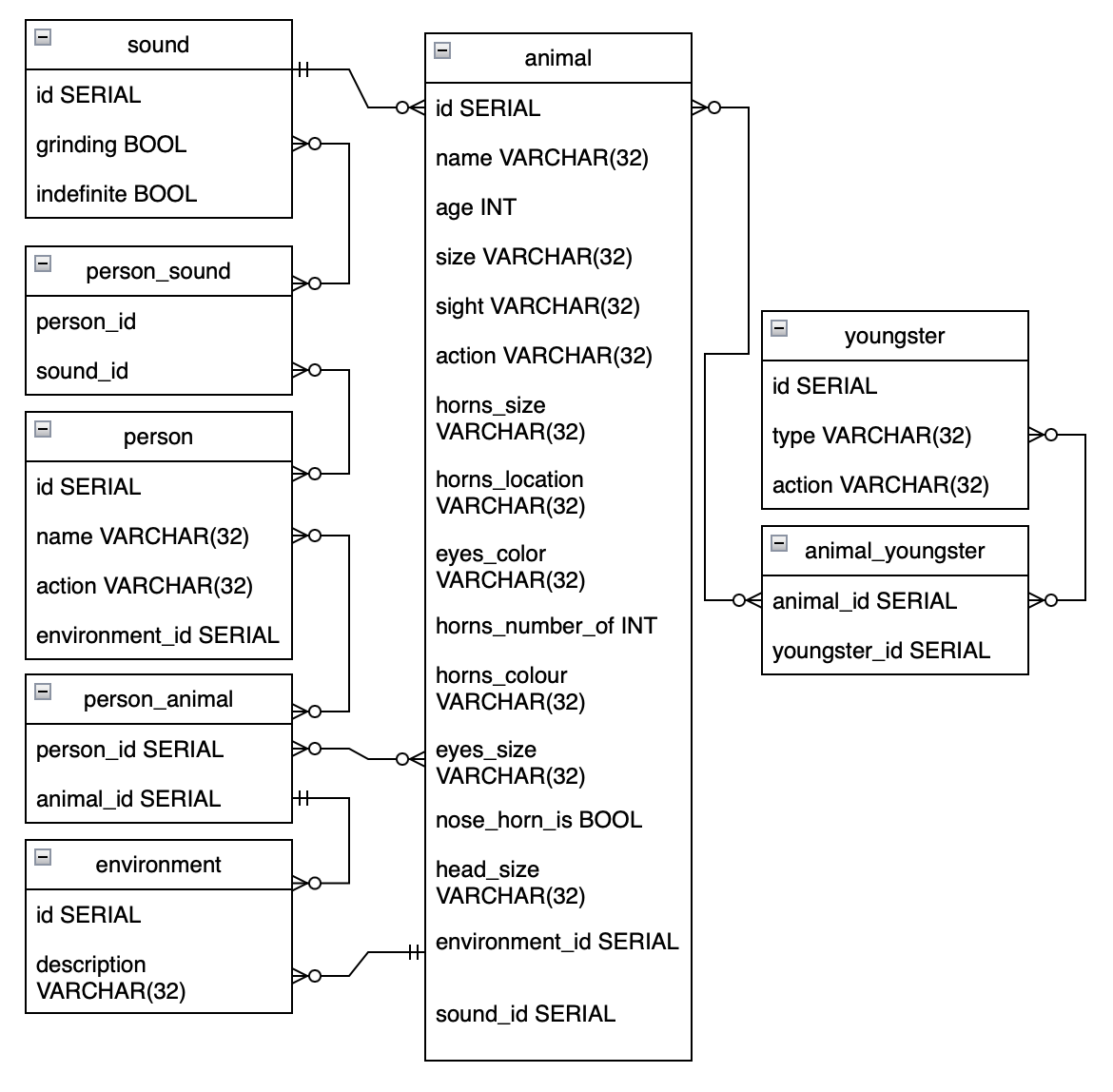


Рисунок 2

**Триггер: если возраст животного больше 50, тогда оно издаёт скрежет и если человек слышит звук скрежета, тогда начинает убегать от животного.**

CREATE OR REPLACE FUNCTION animal\_grinding\_trigger() RETURNS TRIGGER AS $$

BEGIN

IF (NEW.age > 50) THEN

UPDATE sound SET grinding = TRUE WHERE id = NEW.sound\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER animal\_grinding

AFTER INSERT OR UPDATE ON animal

FOR EACH ROW

EXECUTE FUNCTION animal\_grinding\_trigger();

CREATE OR REPLACE FUNCTION person\_running\_away\_trigger() RETURNS TRIGGER AS $$

BEGIN

IF (NEW.grinding = TRUE) THEN

UPDATE person SET action = 'running away'

WHERE id = NEW.person\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER person\_running\_away

AFTER INSERT OR UPDATE ON sound

FOR EACH ROW

EXECUTE FUNCTION person\_running\_away\_trigger();

# Заключение

В ходе выполнения данной лабораторной работы мне удалось разобраться с нормализацей БД, а также триггерами.