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

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №3

Выполнила:

Павличенко Софья Алексеевна, Р3115

Проверил:

Мартин Райла

Санкт-Петербург 2023г.

Оглавление

[Задание 3](#_Toc166514637)

[Исходная модель 4](#_Toc166514638)

[Функциональные зависимости 4](#_Toc166514639)

[Нормализация 5](#_Toc166514640)

[1НФ 5](#_Toc166514641)

[2НФ 5](#_Toc166514642)

[3НФ 7](#_Toc166514643)

[НФБК 9](#_Toc166514644)

[Нормализованная модель 10](#_Toc166514645)

[Денормализация 11](#_Toc166514646)

[Триггер 12](#_Toc166514647)

[Заключение 16](#_Toc166514648)

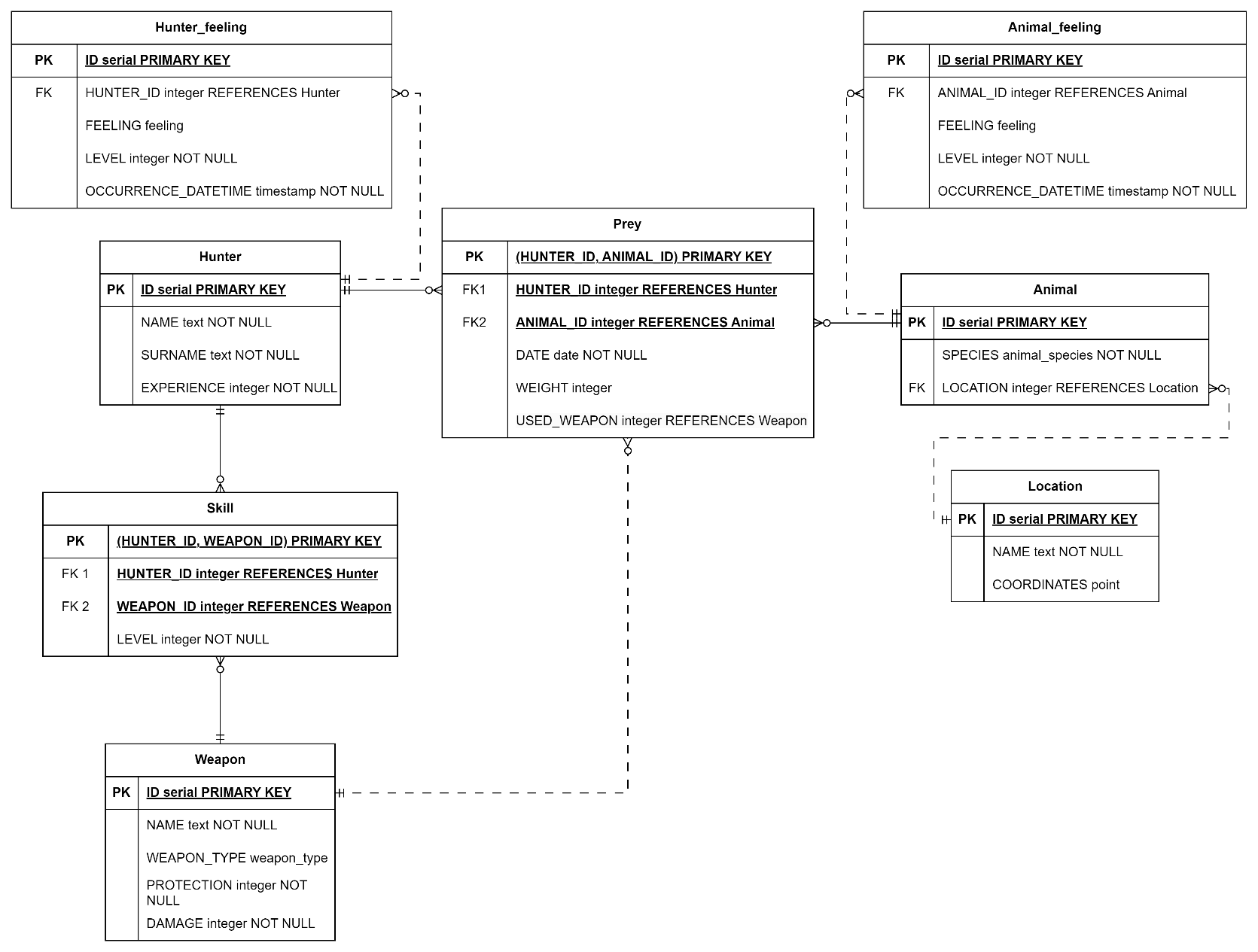
# Задание

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).
* 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



## Функциональные зависимости

Hunter: id -> name, surname, experience

Animal: id -> species, location

Weapon: id -> name, weapon\_type, protection, damage

Location: id -> name, coordinates

Skill: (hunter\_id, weapon\_id) -> level

Prey: (hunter\_id, animal\_id) -> date, weight, used\_weapon

Animal\_feeling: id -> animal\_id, feeling, level, occurrence\_datetime

Hunter\_feeling: id -> hunter\_id, feeling, level, occurrence\_datetime

# Нормализация

## 1НФ

**Отношение, на пересечении каждой строки и столбца — одно значение.**

Атрибуты, имеющие тип данных text, нарушают требования 1НФ. Для её достижения к атрибутам hunter.name, hunter.surname, weapon.name, location.name добавим ограничение, используя регулярное выражение:

CHECK (column\_name ~ '^\S+$')

Это гарантирует, что столбцы будут содержать только одно слово, не включая пробелов или других разделителей. Остальные столбца по типу данных позволяют записывать в себя так же лишь одно значение.

Таким образом, модель будет соответствовать требованиям первой нормальной формы.

## 2НФ

**1) Отношение в 1НФ и 2) атрибуты, не входящие в первичный ключ, в полной функциональной зависимости от первичного ключа отношения**

Проверим таблицы с составным первичным ключом для обнаружения частичных функциональных зависимостей:

1. Skill (hunter\_id, weapon\_id)

**level** функционально зависит от **hunter\_id** и **weapon\_id** одновременно, так как уровень владения не может быть просто связан с охотником или оружием по отдельности. Он связан с конкретной комбинацией охотника и оружия, поскольку навыки охотника с одним оружием могут отличаться от его навыков с другим оружием. Таким образом **level** находится в полной функциональной зависимости от первичного ключа таблицы skill.

1. Prey (hunter\_id, animal\_id)

**date** функционально зависит от **hunter\_id** и **animal\_id** одновременно, так как атрибут представляет дату, когда охотник поймал животное, то есть дата охоты является специфичной для каждой конкретной комбинации охотника и животного. Таким образом **date** находится в полной функциональной зависимости от первичного ключа таблицы prey.

**weight** функционально зависит от **hunter\_id** и **animal\_id** одновременно, но вес пойманного животного никак не зависит от охотника, который его поймал. То есть это частичная функциональная зависимость, есть меньшее подмножество атрибутов, от которых зависит атрибут **weight** – **animal\_id**. Чтобы исправить это, мы можем перенести атрибут **weight** в таблицу animal. Там он будет в полной функциональной зависимости от первичного ключа – **animal\_id**.

**used\_weapon** функционально зависит от **hunter\_id** и **animal\_id** одновременно, так как атрибут представляет оружие, которое охотник использовал для охоты на конкретное животное, то есть используемое оружие является специфичным для каждой конкретной комбинации охотника и животного, что означает, что выбор оружия зависит от контекста охоты. Таким образом **used\_weapon** находится в полной функциональной зависимости от первичного ключа таблицы prey.

Проверим остальные таблицы для обнаружения зависимостей от неключевых атрибутов:

1. Hunter (id)

**name** полностью функционально зависит от **id**, потому что это имя охотника, и каждый охотник имеет только одно имя.

**surname** полностью функционально зависит от **id**, потому что это фамилия охотника, и каждый охотник имеет только одну фамилию.

**experience** полностью функционально зависит от **id**, потому что это опыт охотника, и каждый охотник имеет свой уровень опыта.

1. Animal (id)

**species** полностью функционально зависит от **id**, потому что это вид животного, и каждый идентификатор животного связан только с одним видом.

**weight** полностью функционально зависит от **id**, потому что это вес животного, и каждый идентификатор животного связан только с одним значением веса.

**location** полностью функционально зависит от **id**, потому что это местонахождение животного, и каждый идентификатор животного связан только с одним местоположением.

1. Weapon (id)

**name** полностью функционально зависит от **id**, потому что это название оружия, и каждый идентификатор оружия связан только с одним названием.

**weapon\_type** полностью функционально зависит от **id**, потому что это тип оружия, и каждый идентификатор оружия связан только с одним типом.

**protection** полностью функционально зависит от **id**, потому что это показатель защиты оружия, и каждый идентификатор оружия связан только с одним значением защиты.

**damage** полностью функционально зависит от **id**, потому что это показатель урона оружия, и каждый идентификатор оружия связан только с одним значением урона.

1. Location (id)

**name** полностью функционально зависит от **id**, потому что это название местоположения, и каждый идентификатор местоположения связан только с одним названием.

**coordinates** полностью функционально зависит от **id**, потому что это координаты местоположения, и каждый идентификатор местоположения связан только с одним набором координат.

1. Hunter\_feeling (id)

**hunter\_id** полностью функционально зависит от **id**, поскольку каждое чувство охотника связано с конкретным охотником, идентифицированным **hunter\_id**.

**feeling** полностью функционально зависит от **id**, потому что это чувство, которое испытал охотник, и каждое чувство связано только с одним значением.

**level** полностью функционально зависит от **id**, потому что это уровень чувства охотника, и каждое чувство связано только с одним уровнем.

**occurrence\_datetime** полностью функционально зависит от **id**, потому что это дата и время, когда охотник испытал чувство, и каждое чувство связано только с одной датой и временем.

1. Animal\_feeling (id)

**animal\_id** полностью функционально зависит от **id**, поскольку каждое чувство животного связано с конкретным животным, идентифицированным **animal\_id**.

**feeling** полностью функционально зависит от **id**, потому что это чувство, которое испытало животное, и каждое чувство связано только с одним значением.

**level** полностью функционально зависит от **id**, потому что это уровень чувства животного, и каждое чувство связано только с одним уровнем.

**occurrence\_datetime** полностью функционально зависит от **id**, потому что это дата и время, когда животное испытало чувство, и каждое чувство связано только с одной датой и временем.

Полученная модель удовлетворяет 2НФ.

## 3НФ

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

Рассмотрим отношения между неключевыми атрибутами таблиц:

1. Hunter

Зависимости между неключевыми атрибутами отсутствуют:

**name** не зависит от **surname** и **experience**;

**surname** не зависит от **name** и **experience**;

**experience** не зависит от **name** и **surname**.

1. Animal

Зависимости между неключевыми атрибутами отсутствуют:

**species** не зависит от **weight** и **location**;

**weight** не зависит от **species** и **location**;

**location** не зависит от **species** и **weight**.

1. Weapon

Зависимости между неключевыми атрибутами отсутствуют:

**name** не зависит от **weapon\_type**, **protection** и **damage**;

**weapon\_type** не зависит от **name**, **protection** и **damage**;

**protection** не зависит от **name**, **weapon\_type** и **damage**;

**damage** не зависит от **name**, **weapon\_type** и **protection**.

1. Location

**coordinates** не зависит от **name**;

**name** зависит от **coordinates**: у мест с одними и теми же координатами должно быть одно и то же название. То есть существуеттранзитивная зависимость от первичного ключа: **id** -> **coordinates**, **coordinates** -> **name**. Мы можем исправить это, создав таблицу location\_name, состоящую из первичного ключа – **coordinates** и атрибута **name**, который будет полностью функционально зависеть от **coordinates**.

1. Skill

Зависимости между неключевыми атрибутами отсутствуют, так как неключевой атрибут всего один – **level**.

1. Prey

Зависимости между неключевыми атрибутами отсутствуют:

**date** не зависит от **used\_weapon**;

**used\_weapon** не зависит от date.

1. Hunter\_feeling

Зависимости между неключевыми атрибутами отсутствуют:

**feeling** не зависит от **level** и **occurrence\_datetime**;

**level** не зависит от **feeling** и **occurrence\_datetime**;

**occurrence\_datetime** не зависит от **feeling** и **level**.

1. Animal\_feeling

Зависимости между неключевыми атрибутами отсутствуют:

**feeling** не зависит от **level** и **occurrence\_datetime**;

**level** не зависит от **feeling** и **occurrence\_datetime**;

**occurrence\_datetime** не зависит от **feeling** и **level**.

Таким образом после исправления во всех таблицах отсутствуют транзитивные зависимости между неключевыми атрибутами. Модель удовлетворяет 3НФ.

## НФБК

**Отношение в НФБК, когда для любой нетривиальной функциональной зависимости X→Y, X является суперключом.**

Суперключ — это множество атрибутов, которое однозначно идентифицирует каждую запись в отношении.

Потенциальный ключ — это минимальное множество атрибутов, которое однозначно идентифицирует каждую запись в таблице.

Распишем функциональные зависимости от потенциальных ключей:

Hunter (**name**, **surname**):

(**name**, **surname**) -> **experience**

(**name**, **surname**) -> **id**

Animal (-)

Weapon (**name**, **protection**, **damage**):

(**name**, **protection**, **damage**) -> **weapon\_type**

(**name**, **protection**, **damage**) -> **id**

Location (**coordinates**):

(**coordinates**) -> **id**

Location\_name (-)

Prey (-)

Skill (-)

Hunter\_feeling (**hunter\_id**, **feeling**, **occurance\_datetime**):

(**hunter\_id**, **feeling**, **occurance\_datetime**) -> **level**

(**hunter\_id**, **feeling**, **occurance\_datetime**) -> **id**

Animal\_feeling (**animal\_id**, **feeling**, **occurance\_datetime**):

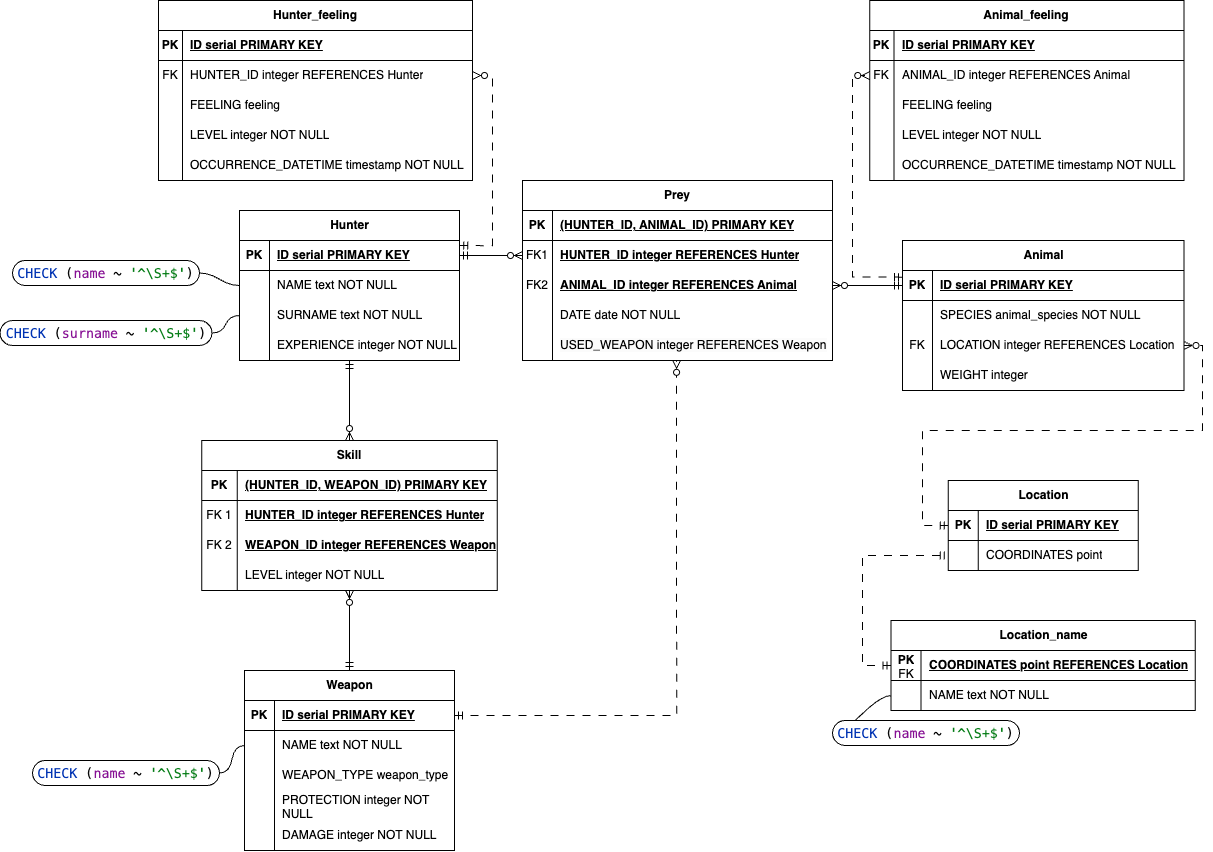
(**animal\_id**, **feeling**, **occurance\_datetime**) -> **level**

(**animal\_id**, **feeling**, **occurance\_datetime**) -> **id**

То есть каждый атрибут, не входящий в потенциальный ключ, функционально зависит от каждого потенциального ключа. Тогда каждый X функциональной зависимости X -> Y является суперключом, так как потенциальный ключ – минимальное множество суперключа.

Таким образом, модель удовлетворяет НФБК.

# Нормализованная модель



# Денормализация

**Добавление избыточных атрибутов:**

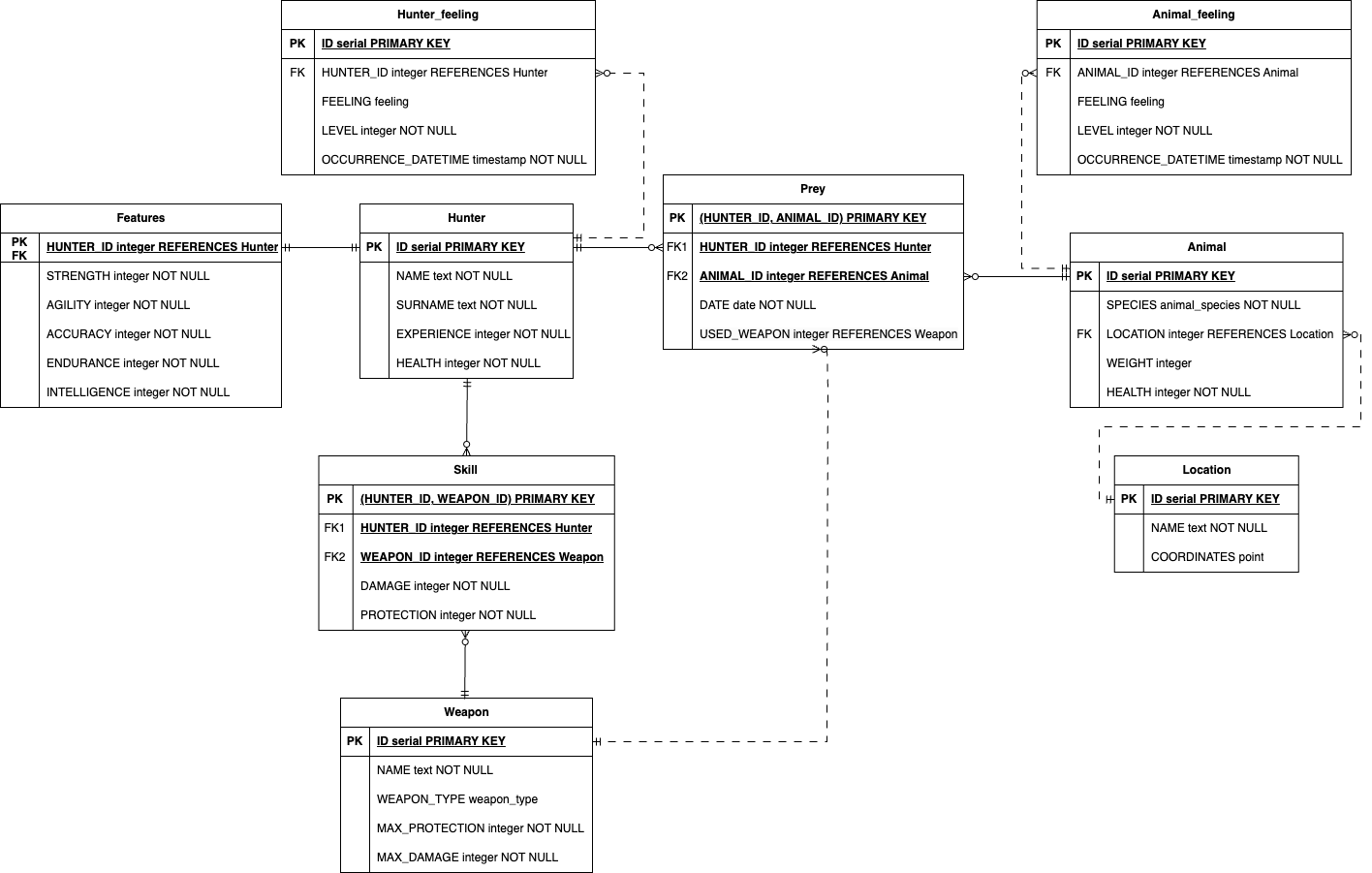
В некоторых случаях избыточность атрибутов может улучшить производительность запросов. Например, если часто запрашивается вес полученной добычи, то можно вернуть атрибут weight из таблицы Animal обратно в Prey, как это было в исходной модели. Это поможет избежать частого объединения таблиц для получения веса по id животного.

**Объединение связанных таблиц:**

Объединение связанных таблиц также поможет избежать частых JOIN. Например, можно объединить таблицы Hunter и Hunter\_feeling или Animal и Animal\_feeling, для быстрого доступа к ощущениям охотников или животных.

# Триггер

CREATE OR REPLACE FUNCTION *adjust\_weapon\_skills\_on\_hunter\_features*()  
 RETURNS TRIGGER AS $$  
 DECLARE type weapon\_type;  
 DECLARE hunter\_strength decimal;  
 DECLARE hunter\_agility decimal;  
 DECLARE hunter\_accuracy decimal;  
 DECLARE hunter\_endurance decimal;  
 DECLARE hunter\_intelligence decimal;  
  
BEGIN  
 SELECT weapon\_type INTO type FROM weapon WHERE weapon.id = NEW.**weapon\_id**;  
 SELECT strength / 100, agility / 100, accuracy / 100, endurance / 100, intelligence / 100  
 INTO hunter\_strength, hunter\_agility, hunter\_accuracy, hunter\_endurance, hunter\_intelligence  
 FROM features  
 JOIN hunter ON features.hunter\_id = hunter.id WHERE hunter.id = NEW.**hunter\_id**;  
 NEW.damage := (CASE  
 WHEN type = 'огнестрельное оружие' THEN  
 (0.4 \* hunter\_strength + 0.3 \* hunter\_accuracy + 0.3 \* hunter\_endurance) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'огнестрельное оружие')  
 WHEN type = 'лук и стрелы' THEN  
 (0.8 \* hunter\_accuracy + 0.2 \* hunter\_agility) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'лук и стрелы')  
 WHEN type = 'метательное оружие' THEN  
 (0.3 \* hunter\_agility + 0.7 \* hunter\_accuracy) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'метательное оружие')  
 WHEN type = 'ручное оружие' THEN  
 (0.5 \* hunter\_strength + 0.5 \* hunter\_agility) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'ручное оружие')  
 WHEN type = 'кинжалы и ножи' THEN  
 (0.5 \* hunter\_agility + 0.5 \* hunter\_accuracy) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'кинжалы и ножи')  
 WHEN type = 'дробящее оружие' THEN  
 (0.8 \* hunter\_strength + 0.2 \* hunter\_endurance) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'дробящее оружие')  
 WHEN type = 'копья' THEN  
 (0.6 \* hunter\_strength + 0.4 \* hunter\_agility) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'копья')  
 WHEN type = 'топоры и тесаки' THEN  
 (0.7 \* hunter\_strength + 0.3 \* hunter\_endurance) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'топоры и тесаки')  
 WHEN type = 'магическое оружие' THEN  
 (0.8 \* hunter\_intelligence + 0.3 \* hunter\_agility + 0.1 \* hunter\_accuracy) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'другое')  
 WHEN type = 'другое' THEN  
 (0.4 \* hunter\_strength + 0.3 \* hunter\_agility + 0.3 \* hunter\_accuracy) \*  
 (SELECT max\_damage FROM weapon WHERE weapon\_type = 'другое')  
 ELSE 0  
 END);  
 NEW.protection := (CASE  
 WHEN type = 'огнестрельное оружие' THEN  
 (0.4 \* hunter\_strength + 0.3 \* hunter\_accuracy + 0.3 \* hunter\_endurance) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'огнестрельное оружие')  
 WHEN type = 'лук и стрелы' THEN  
 (0.8 \* hunter\_accuracy + 0.2 \* hunter\_agility) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'лук и стрелы')  
 WHEN type = 'метательное оружие' THEN  
 (0.3 \* hunter\_agility + 0.7 \* hunter\_accuracy) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'метательное оружие')  
 WHEN type = 'ручное оружие' THEN  
 (0.5 \* hunter\_strength + 0.5 \* hunter\_agility) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'ручное оружие')  
 WHEN type = 'кинжалы и ножи' THEN  
 (0.5 \* hunter\_agility + 0.5 \* hunter\_accuracy) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'кинжалы и ножи')  
 WHEN type = 'дробящее оружие' THEN  
 (0.8 \* hunter\_strength + 0.2 \* hunter\_endurance) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'дробящее оружие')  
 WHEN type = 'копья' THEN  
 (0.6 \* hunter\_strength + 0.4 \* hunter\_agility) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'копья')  
 WHEN type = 'топоры и тесаки' THEN  
 (0.7 \* hunter\_strength + 0.3 \* hunter\_endurance) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'топоры и тесаки')  
 WHEN type = 'магическое оружие' THEN  
 (0.8 \* hunter\_intelligence + 0.3 \* hunter\_agility + 0.1 \* hunter\_accuracy) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'другое')  
 WHEN type = 'другое' THEN  
 (0.4 \* hunter\_strength + 0.3 \* hunter\_agility + 0.3 \* hunter\_accuracy) \*  
 (SELECT max\_protection FROM weapon WHERE weapon\_type = 'другое')  
 ELSE 0  
 END);  
 RETURN NEW;  
END  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER adjust\_weapon\_skills\_trigger  
 BEFORE INSERT OR UPDATE on skill  
 FOR EACH ROW  
 EXECUTE PROCEDURE *adjust\_weapon\_skills\_on\_hunter\_features*();  
  
  
CREATE OR REPLACE FUNCTION *hunter\_animal\_combat*()  
 RETURNS TRIGGER AS $$  
DECLARE  
 animal\_specie animal\_species;  
 weapon\_damage DECIMAL;  
 weapon\_protection DECIMAL;  
 hunter\_health INTEGER;  
 animal\_health INTEGER;  
 counter INTEGER := 0;  
BEGIN  
 SELECT damage, protection INTO weapon\_damage, weapon\_protection FROM skill  
 JOIN hunter ON NEW.**hunter\_id** = hunter.id;  
 SELECT health INTO hunter\_health FROM hunter  
 WHERE id = NEW.**hunter\_id**;  
 SELECT health, species INTO animal\_health, animal\_specie FROM animal  
 WHERE id = NEW.**animal\_id**;  
  
 WHILE counter < 3 AND animal\_health > 0 AND hunter\_health > 0 LOOP  
 IF *RANDOM*() <= weapon\_damage THEN animal\_health := animal\_health - 1;  
 RAISE NOTICE 'Охотник атаковал %!', animal\_specie;  
 END IF;  
 IF animal\_health > 0 AND *RANDOM*() >= weapon\_protection THEN hunter\_health := hunter\_health - 1;  
 RAISE NOTICE '% атаковал(-а) охотника!', animal\_specie;  
 END IF;  
 counter := counter + 1;  
 END LOOP;  
 UPDATE hunter SET health = hunter\_health WHERE id = NEW.hunter\_id;  
 UPDATE animal SET health = animal\_health WHERE id = NEW.animal\_id;  
 IF hunter\_health = 0 THEN  
 DELETE FROM features WHERE features.hunter\_id = NEW.hunter\_id;  
 DELETE FROM hunter WHERE id = NEW.hunter\_id;  
 RAISE NOTICE 'Охотник погиб.';  
 END IF;  
 IF animal\_health > 0 THEN RAISE WARNING 'Невозможно вставить запись о добыче, так как животное все еще живо.';  
 RETURN NULL;  
 ELSE RAISE NOTICE 'Животное погибло.';  
 RETURN NEW;  
 END IF;  
END  
$$ LANGUAGE plpgsql;  
  
  
  
CREATE TRIGGER combat\_outcome\_trigger  
 BEFORE INSERT OR UPDATE on prey  
 FOR EACH ROW  
EXECUTE PROCEDURE *hunter\_animal\_combat*();



# Заключение

В результате выполнения лабораторной работы я научилась создавать инфологическую и даталогическую модели, на их основе реализовывать базу данных с помощью SQL.

Если добавляется охотник с опытом больше н лет то он умеет пользоватться всем оружием

Если охотник убил с помощь weapon кого то то кровень владения повышается если он не максимум

Или н Пример есть ферма и на ферме растут деревья

Есть свинти

Надо написать когда расписываете первую и вторую нф

Свинья здесь заслуживает быть и зависит от ид фермы то зависит

Кока кола не должна быть фермы

Никакого отношения к ферме не имеет

Почему деномрализацяи полезна так же

Список причин

Как оно относится к ключу

Как упрощает жизнь

Триггер

Есть

апример проверка для использования вообще этого оружия

2няф

Везде писатьв

Таблица с соотсветствием навыка и количесвтмо урона типо

Нужно создать формулы

100 это в процентах нужно умножать на максимум

Например для типо лука

(0,2 от силы + 0,8 от меткости + 0,7 от ловкости) \* max()

Пусть max = 8

Округлять?

Тогда таблицц максимум хащиты и дамака

Только есть оружие только с определным набором навыков дамак от этого оружия был бы больше 50

Связь между навыком и его влиянием на оружие

Другая таблица

Рандом попадаие кол во атак

Мини бой

каждый детерминирующий атрибут (атрибут, который определяет другие атрибуты) является ключом.