Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №42 «Криптология и кибербезопасность»

Отчёт

О выполнении лабораторной работы №3 «Сложные запросы на выборку. Соединения» по курсу «Безопасность баз данных»

Студент: Лузганов К. А.
Группа: Б22-525

Москва – 2025

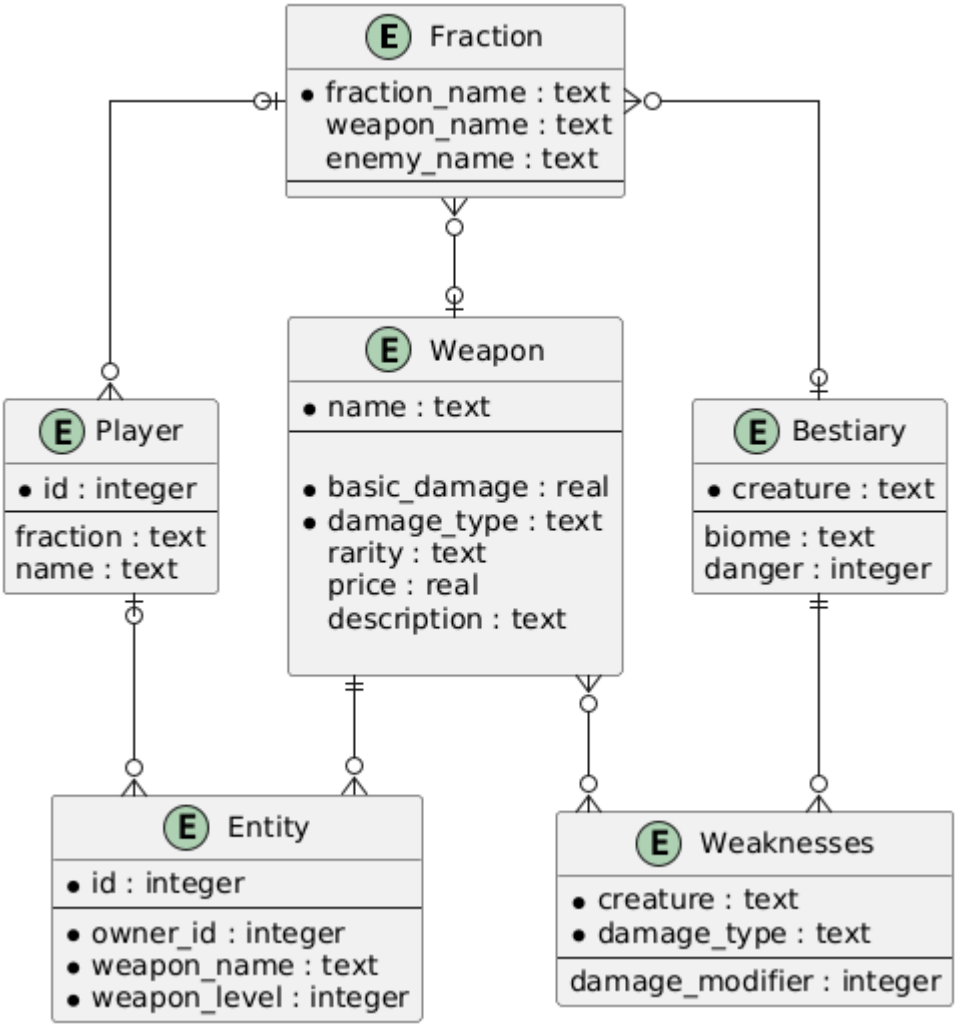

Рис. 1 – ERD-диаграмма разработанной базы данных

Ниже представлен список выполненных запросов SQL с описаниями их смысла и ожидаемых результатов, а также результатами их выполнения на хранящихся в БД данных.

---

1. Позволяет по оружию найти всех существ, против которых оно будет иметь преимущество

```
SELECT creature
FROM Weaknesses
WHERE damage_modifier > 100
AND damage_type = (
    SELECT damage_type
    FROM Weapon
    WHERE name = (
        SELECT weapon_name
        FROM Entity
        WHERE id = 1
    )
);
```

2. Ставит в соответствие идентификатору сущности оружия всех существ, уязвимых против него (ср. с первым запросом)

```
SELECT creature
FROM Entity
JOIN Weapon ON Entity.weapon_name = Weapon.name
JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
WHERE Entity.id = 1
  AND Weaknesses.damage_modifier > 100;
```

---

3. Ставит в соответствие идентификатору сущности оружия тип его урона

```
SELECT id, damage_type
FROM Entity, Weapon
WHERE Entity.weapon_name = Weapon.name
;
```

---

4. Снова по идентификатору сущности оружия находит всех существ, уязвимых против него (ср. с 1-м запросом)

```
WITH id_2_dtype AS (
    SELECT id, damage_type
    FROM Entity, Weapon
    WHERE Entity.weapon_name = Weapon.name
)
SELECT id, creature
FROM id_2_dtype, Weaknesses
WHERE id_2_dtype.damage_type = Weaknesses.damage_type
AND Weaknesses.damage_modifier > 100
;
```

---

5. Выводит соответствие: «игрок - оружие - тип урона - монстр – модификатор»

```
SELECT Player.id, Weapon.name, Weapon.damage_type, creature,
Weaknesses.damage_modifier
FROM Player
JOIN Entity ON Player.id = Entity.owner_id
JOIN Weapon ON Entity.weapon_name = Weapon.name
JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
WHERE Weaknesses.damage_modifier > 100
AND Player.id = 1
;
```

---

6. Определяет, имеет ли игрок преимущество против монстра или нет

```
WITH player_2_creature
AS (
    SELECT Player.id AS player_id, creature
    FROM Player
    JOIN Entity ON Player.id = Entity.owner_id
    JOIN Weapon ON Entity.weapon_name = Weapon.name
    JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
    WHERE Weaknesses.damage_modifier > 100
)
SELECT CASE
    WHEN EXISTS(
        SELECT 1 FROM player_2_creature WHERE player_id = 1 AND creature =
'проклятый дракон с палкой'
    ) THEN 1
    ELSE 0
END AS advantage
;
```

---

7. Вычисляет средний модификатор выбранного типа урона по всем существам, живущим в выбранном биоме

```
SELECT biome, damage_type, avg(damage_modifier)
FROM Bestiary
JOIN Weaknesses ON Weaknesses.creature = Bestiary.creature
GROUP BY biome, damage_type;
```

---

8. Вычисляет средний модификатор каждого типа урона по существам каждого биома

```
SELECT biome, damage_type, avg(damage_modifier)
FROM Bestiary
JOIN Weaknesses ON Weaknesses.creature = Bestiary.creature
GROUP BY biome, damage_type;
```

```
Вычисляет типы урона с минимальным и максимальным средним модификатором урона для
каждого биома

WITH avg_mods AS (
    SELECT
        Bestiary.biome,
        Weaknesses.damage_type,
        AVG(Weaknesses.damage_modifier) AS avg_modifier
    FROM Bestiary
    JOIN Weaknesses ON Bestiary.creature = Weaknesses.creature
    GROUP BY Bestiary.biome, Weaknesses.damage_type
),
```

```
ranked AS (
    SELECT *,
           RANK() OVER (PARTITION BY biome ORDER BY avg_modifier DESC) AS
max_rank,
           RANK() OVER (PARTITION BY biome ORDER BY avg_modifier ASC) AS min_rank
    FROM avg_mods
)
SELECT biome, damage_type, avg_modifier, 'max' AS type
FROM ranked
WHERE max_rank = 1

UNION ALL

SELECT biome, damage_type, avg_modifier, 'min' AS type
FROM ranked
WHERE min_rank = 1
ORDER BY biome, type;
```

Заключение

Как правило, при составлении запросов к базе данных требуется провести частичную денормализацию и использовать данные из двух и более таблиц одновременно. С этой целью в данной лабораторной работе были изучены механизмы денормализации данных в базе данных и объединении двух и более таблиц различными способами.

Приложение А. Ссылка на github–репозиторий проекта

https://github.com/luzganovka/Sqlite3_labs.git

Приложение Б. Литстинг использованных инструкций SQL

```sql
-- по оружию найти всех существ, против которых оно будет иметь преимущество
SELECT creature
FROM Weaknesses
WHERE damage_modifier > 100
AND damage_type = (
    SELECT damage_type
    FROM Weapon
    WHERE name = (
        SELECT weapon_name
        FROM Entity
        WHERE id = 1
    )
);


-- Ид оружия - все существа, уязвимые против него (ср. с первым запросом)
SELECT creature
FROM Entity
JOIN Weapon ON Entity.weapon_name = Weapon.name
JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
WHERE Entity.id = 1
  AND Weaknesses.damage_modifier > 100;


-- ИД оружия - тип урона
SELECT id, damage_type
FROM Entity, Weapon
WHERE Entity.weapon_name = Weapon.name
;


-- Ид оружия - все существа, уязвимые против него (ср. с 1-м запросом)
WITH id_2_dtype AS (
    SELECT id, damage_type
    FROM Entity, Weapon
    WHERE Entity.weapon_name = Weapon.name
)
SELECT id, creature
FROM id_2_dtype, Weaknesses
WHERE id_2_dtype.damage_type = Weaknesses.damage_type
AND Weaknesses.damage_modifier > 100
;



-- Игрок - оружие - тип урона - монстр - модификатор
```

```sql
SELECT Player.id, Weapon.name, Weapon.damage_type, creature,
Weaknesses.damage_modifier
FROM Player
JOIN Entity ON Player.id = Entity.owner_id
JOIN Weapon ON Entity.weapon_name = Weapon.name
JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
WHERE Weaknesses.damage_modifier > 100
AND Player.id = 1
;


-- Решить, имеет ли игрок преимущество против монстра или нет
WITH player_2_creature
AS (
    SELECT Player.id AS player_id, creature
    FROM Player
    JOIN Entity ON Player.id = Entity.owner_id
    JOIN Weapon ON Entity.weapon_name = Weapon.name
    JOIN Weaknesses ON Weapon.damage_type = Weaknesses.damage_type
    WHERE Weaknesses.damage_modifier > 100
)
SELECT CASE
    WHEN EXISTS(
        SELECT 1 FROM player_2_creature WHERE player_id = 1 AND creature =
'проклятый дракон с палкой'
    ) THEN 1
    ELSE 0
END AS advantage
;


-- Average damage modifier of all creatures living in certain biome against
certain damage_type
SELECT biome, damage_type, avg(damage_modifier)
FROM Bestiary
JOIN Weaknesses ON Weaknesses.creature = Bestiary.creature
GROUP BY biome, damage_type;


-- Average damage modifier of all creatures living in every biome against every
damage_type
SELECT biome, damage_type, avg(damage_modifier)
FROM Bestiary
JOIN Weaknesses ON Weaknesses.creature = Bestiary.creature
GROUP BY biome, damage_type;


-- Min and Max average damage modifier for every biom
WITH avg_mods AS (
    SELECT
        Bestiary.biome,
```

```
        Weaknesses.damage_type,
        AVG(Weaknesses.damage_modifier) AS avg_modifier
    FROM Bestiary
    JOIN Weaknesses ON Bestiary.creature = Weaknesses.creature
    GROUP BY Bestiary.biome, Weaknesses.damage_type
),
ranked AS (
    SELECT *,
           RANK() OVER (PARTITION BY biome ORDER BY avg_modifier DESC) AS
max_rank,
           RANK() OVER (PARTITION BY biome ORDER BY avg_modifier ASC) AS min_rank
    FROM avg_mods
)
SELECT biome, damage_type, avg_modifier, 'max' AS type
FROM ranked
WHERE max_rank = 1

UNION ALL

SELECT biome, damage_type, avg_modifier, 'min' AS type
FROM ranked
WHERE min_rank = 1
ORDER BY biome, type;
```