МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Отчёт по дисциплине «Теоретические основы баз данных»

Курсовая работа «Разработка и программирование базы данных о видах бабочек в Ленинградской области»

Студент: _____ Гусева Станислава Александровна

Преподаватель: _____ Попов Сергей Геннадевич

«____»_____ 20__ г.

Санкт-Петербург – 2024

Содержание

1 Аналитика

1.1 Описание предметной области

Бабочки играют важную роль в экосистемах, являясь показателями состояния окружающей среды. Многие виды бабочек подвергаются угрозам из-за потери мест обитания, загрязнения окружающей среды, климатических изменений и других видов человеческой деятельности. Изучение их видового разнообразия и распространения может помочь понять, какие виды нуждаются в защите.

Семейство в биологической классификации является высшим таксономическим рангом, находящимся между отрядом и родом. В контексте насекомых, семейство - это группа родственных видов, которые обладают сходными морфологическими и биологическими характеристиками. Семейство объединяет роды насекомых, которые имеют общих предков и демонстрируют определенные сходства в строении, поведении, биологии и экологии. Например, семейство может иметь сходные характеристики строения крыльев и тела, способы размножения и поведения.

Род в биологической классификации является таксономическим рангом, расположенным ниже семейства и выше вида. Род объединяет группу близких по родству организмов, которые имеют сходные морфологические, генетические и биологические характеристики. У бабочек роды делятся на основе ряда характеристик, включая морфологию, анатомию, биологию и генетические особенности. Эти характеристики помогают ученым классифицировать бабочек и определить их место в системе таксономии. Различные роды бабочек обычно имеют сходные черты и часто включают в себя несколько видов, которые могут иметь схожий образ жизни, экологию и поведение. Каждый род бабочек обычно характеризуется уникальными особенностями, такими как форма крыльев, цветовая гамма, рисунок на крыльях, строение тела и прочие анатомические особенности. Например, род Papilio включает в себя бабочек с крупными крыльями и яркими цветами, а род Nymphalis включает бабочек с красочными рисунками на крыльях. Каждый род бабочек обычно называется латинским именем, которое является уникальным для данного рода. Классификация бабочек на роды может изменяться в зависимости от новых исследований, а также от изменений в системе таксономии.

Вид в биологической классификации - это основная единица классификации живых организмов, которая объединяет особей с общими генетическими, морфологическими и экологическими характеристиками. Виды представляют собой группу особей, способных скрещиваться между собой и давать потомство, способное к размножению. Каждый вид обычно имеет уникальное научное название, которое состоит из двух частей: рода и видового эпитета. Например, у обыкновенного махаона научное название Papilio machaon, где Papilio - это род, а machaon - видовой эпитет. Виды бабочек могут варьироваться в своем образе жизни, питании, местообитаниях, особенностях рисунка на крыльях и других аспектах, что делает каждый вид уникальным и важным для биологического разнообразия.

В Ленинградской области насчитывается 79 видов бабочек, среди них есть такие семейства как: парусники, белянки, нимфалиды, голубянки и толстоголовки. Каждое семейство включает в себя несколько родов, а тот в свою очередь несколько видов.

Пример: семейство белянок включает в себя 9 родов (Зорька, Боярышница, Капустница, Брюквенница, Репница, Белянка горошковая, Желтушка, Крушинница/Лимонница; род Желтушка включает в себя 3 вида (Желтушка шафрановая (Colias croceus), Желтушка луговая (Colias hyale), Желтушка торфяниковая (Colias palaeno)), род Зорек и род Боярышниц включают в себя по 1 виду, названия которых соответствуют названию рода и тд.

Индивидуальные характеристики популяции:

1. Численность за последний год;
2. Продолжительность жизни;
3. Статус (внесен ли в Красную книгу России и, если внесен, то в какой раздел (Вероятно исчезнувшие, Находящиеся под угрозой исчезновения, Сокращающиеся в численности, Редкие, Неопределенные по статусу, Восстанавливаемые и восстанавливающиеся));
4. Семейство;
5. Род;
6. Вид;
7. Цвет и размер крыльев;
8. Орнамент крыльев;
9. Особенности лап и туловища;
10. Питание.

Для идентификации вида встреченной бабочки используется двухшаговая система. Сперва пользователь заполняет анкету в приложении, где отвечает на следующие вопросы:

1. цвет туловища;
2. цвет крыльев;
3. рисунок крыльев с множественным выбором вариантов ответа: без рисунка, черная полоса по краю, белая полоса по краю, черная полоса на уголке верхних крыльев, черные полосы сетчатого вида, крапинки, цветные круги, прожилки;
4. особенности рисунка крыльев с множественным выбором вариантов ответа: без особенностей, два круга, четыре круга, более четырех кругов, черные полосы имеют крапинки, поломы волнообразные;
5. поверхность (на чем сидит бабочка: цветы, земля, глина, дерево, компост, мох, трава/листья, другое);
6. примерный размер;
7. длина лап (длинные/короткие);
8. длина усиков (длинные/короткие);
9. цвет усиков.

Помимо этого пользователь прикладывает 1-4 фотографии бабочки. А также указывает координаты: широту и долготу, где была встречена бабочка, и дату (год).

Далее эксперт просматривает отправленные пользователями анкеты и заполняет еще три поля, определив следующие характеристики:

1. Семейство;
2. Род;
3. Вид.

Таким образом остаются только модерированные анкеты с полными и корректными данными.

Для сбора статистики по численности собираются данные о количестве бабочек каждого вида, зафиксированных в конкретный год.

Бабочки, количество которых от года к году существенно не меняется (+-5% от их числа в предыдущем году), но при этом их число в 2 и более раз меньше, чем среднее число бабочек остальных видов, считаются редким видом.

Бабочки, количество которых на протяжении последних 3х и более лет существенно уменьшается (на 15% и более от их числа в предыдущем году), считаются вымирающим видом.

1.2 Формулировка целей создания базы данных

1. Получение информации о текущей численности каждого вида;
2. Выявление наиболее редких видов бабочек;
3. Выявление вымирающих видов бабочек;
4. Фиксация ежегодного изменения популяции;

1.3 Сущности и атрибуты

Бабочка:
встреченные экземпляры видов.

- автор;
- вид;
- координаты;
- дата.

Пользователь:
человек, который присылает анкеты для бабочки.

- логин;
- пароль;
- имя (ник).

Фото:

- фотография.

Видео:

- видео.

Словарь видов:
содержит индивидуальные характеристики каждого вида.

- название;
- крылья;
- лапки;
- усики;
- туловище;
- питание;
- продолжительность жизни.

Крылья:
описание крыльев бабочек, присущих разным видам.

- цвет;
- орнамент;
- особенности орнамента;
- форма;
- размах.

Лапки:
описание лапок бабочек, присущих разным видам.

- длина;
- цвет.

Усики:
описание усиков бабочек, присущих разным видам.

- длина;
- цвет;
- особенности.

Туловище:
описание туловищ бабочек, присущих разным видам.

- цвет;
- орнамент.

Питание:
описание питаний бабочек, присущих разным видам.

- необходимые микроэлементы;
- любимые места.

Словарь Красной книги:
все, когда либо существовавшие, статусы всех бабочек из словаря видов

- статус;
- дата присвоения статуса;
- дата отмены статуса.

1.4 ER-диаграмма

1.4.1 Схема ER-диаграммы

На рис.1 приведена ER-диаграмма базы данных о видах бабочек в Ленинградской области.

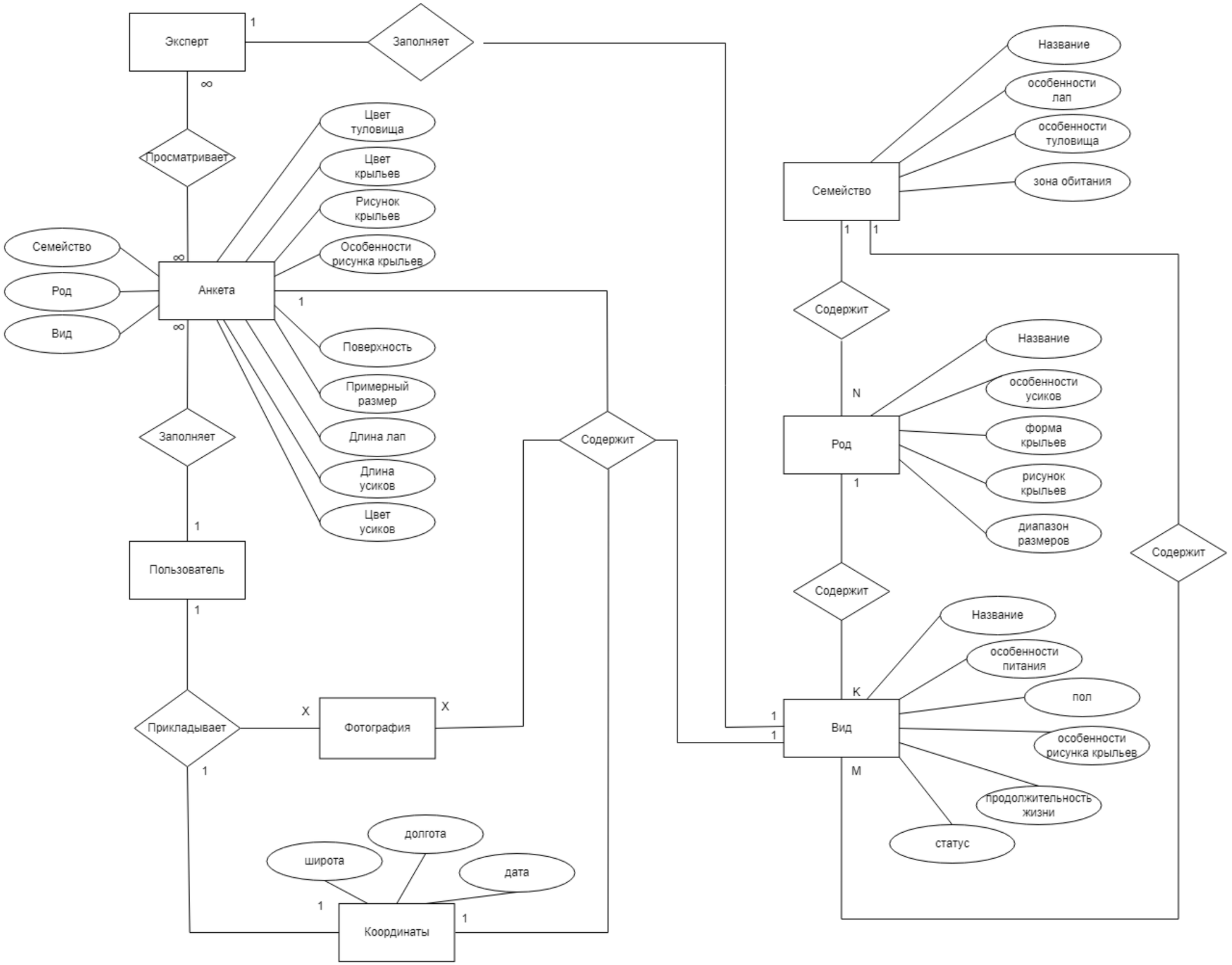


Рис. 1: ER-диаграмма

1.4.2 Описание ER-диаграммы

1. Просматривает:
    - Эксперт просматривает анкету.
2. Заполняет
    - Пользователь заполняет анкету;
    - Эксперт заполняет вид.
3. Прикладывает
    - Пользователь прикладывает фотографии;
    - Пользователь прикладывает координаты.
4. Содержит
    - Анкета содержит фотографии;
    - Анкета содержит координаты;
    - Анкета содержит вид;
    - Семейство содержит род;
    - Род содержит вид;
    - Семейство содержит вид.

1.5 Схема объектов

На рис.2 приведена схема объектов базы данных о видах бабочек в Ленинградской области.

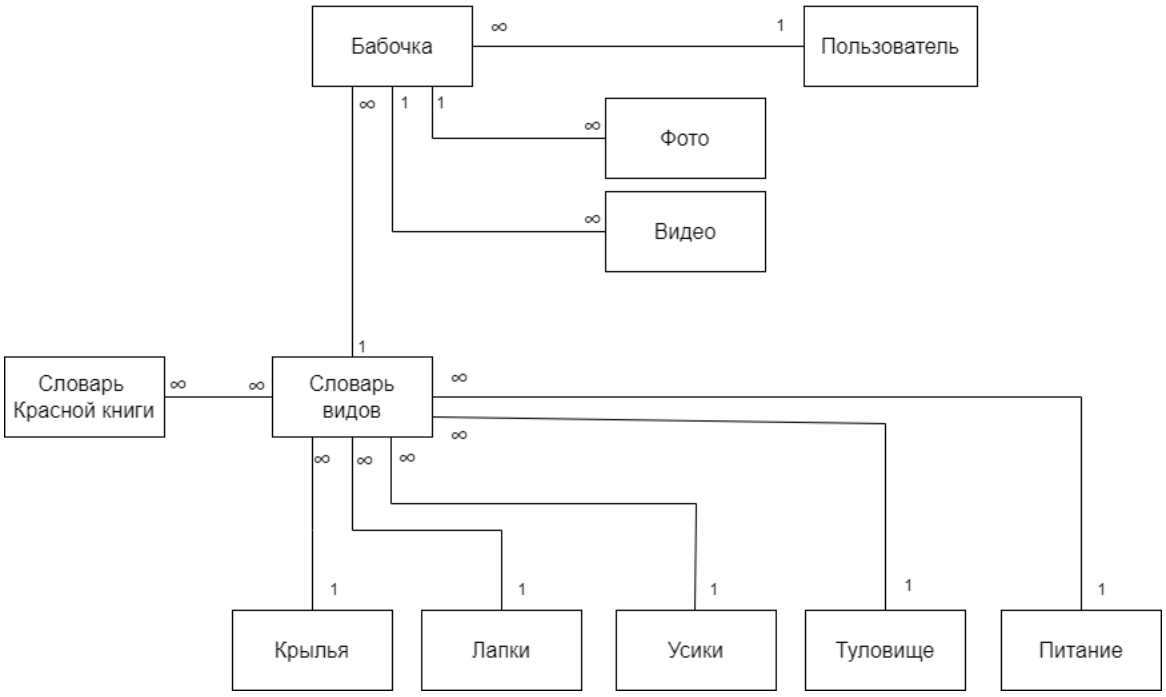


Рис. 2: Схема объектов

2 База данных

2.1 Таблицы базы данных

Код генерации таблиц представлен в приложении А.

2.1.1 Бабочка

- Координаты - varchar(20), запись координат длиной до 20 символов с точностью до 20м вида "55.7539°N 37.6208°E";
- Дата - varchar(10), запись даты длиной до 10 символов вида ”23.08.2024”, то есть месяц и год.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	автор	author	integer	FK	user_id
3	вид	species	integer	FK	Species_dictionary_id
4	координаты	coordinates	varchar(20)	-	-
5	дата	data	varchar(10)	-	-

Таблица 1: сущность бабочка (butterfly)

2.1.2 Пользователь

- Логин - varchar(20), длиной до 20 символов;
- Пароль - varchar(30), длиной до 30 символов;
- Псевдоним - varchar(20), длиной до 20 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	логин	login	varchar(20)	-	-
3	пароль	password	varchar(30)	-	-
4	псевдоним	nickname	varchar(20)	-	-

Таблица 2: сущность пользователь (user)

2.1.3 Фото

- фотография - BLOB, чтобы хранить бинарные данные.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	id_бабочки	butterfly_id	integer	FK	butterfly_id
3	фотография	photo	BLOB	-	-

Таблица 3: сущность фото (photo)

2.1.4 Видео

- видео - MEDIUMBLOB, чтобы хранить бинарные данные до 16 мегабайт.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	id_бабочки	butterfly_id	integer	FK	butterfly_id
3	видео	video	MEDIUMBLOB	-	-

Таблица 4: сущность видео (video)

2.1.5 Словарь видов

- Название - varchar(100), длиной до 100 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	название	name	varchar(100)	-	-
3	крылья	wings	integer	FK	wings_id
4	лапки	paws	integer	FK	paws_id
5	усики	tendrils	integer	FK	tendrils_id
6	туловище	body	integer	FK	body_id
7	питание	nutrition	integer	FK	nutrition_id
8	продолжительность_жизни	life_expectancy	integer	-	-

Таблица 5: сущность словарь видов (species dictionary)

2.1.6 Тип статуса

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	название	name	varchar(40)	-	-

Таблица 6: сущность тип статуса (status type)

2.1.7 Статус

- Статус - varchar(20), длиной до 20 символов;
- Дата присвоения статуса - varchar(10), запись даты длиной до 10 символов вида ”21.08.2024”;
- Дата отмены статуса - varchar(10), запись даты длиной до 10 символов вида ”30.10.2024”.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	тип	type	integer	FK	status_type_id
3	дата присвоения статуса	status_assignment_date	varchar(10)	-	-
4	дата отмены статуса	cancellation_of_status_date	varchar(10)	-	-
5	вид_id	species_id	integer	FK	species_dictionary_id

Таблица 7: сущность статус (stasus)

2.1.8 Крылья

- Цвет - varchar(10), длиной до 10 символов;
- Орнамент - varchar(100), длиной до 100 символов;
- Особенности орнамента - varchar(110), длиной до 110 символов;
- Форма - varchar(10), , длиной до 10 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	цвет	color	varchar(10)	-	-
3	орнамент	ornament	varchar(100)	-	-
4	особенности орнамента	ornament_features	varchar(110)	-	-
5	форма	shape	varchar(10)	-	-
5	размах	wingspan	integer	-	-

Таблица 8: сущность крылья (wings)

2.1.9 Лапки

- Цвет - varchar(10), длиной до 10 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	длина	length	integer	-	-
3	цвет	color	varchar(10)	-	-

Таблица 9: сущность лапки (paws)

2.1.10 Усики

- Цвет - varchar(10), длиной до 10 символов;
- Особенности - varchar(20), длиной до 20 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	длина	length	integer	-	-
3	цвет	color	varchar(10)	-	-
4	особенности	features	varchar(20)	-	-

Таблица 10: сущность усики (tendrils)

2.1.11 Туловище

- Цвет - varchar(10), длиной до 10 символов;
- Орнамент - varchar(20), длиной до 20 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	цвет	color	varchar(10)	-	-
3	орнамент	ornament	varchar(20)	-	-

Таблица 11: сущность туловище (body)

2.1.12 Питание

- Необходимые микроэлементы - varchar(30), длиной до 30 символов;
- Любимые места - varchar(20), длиной до 20 символов.

№	Название на русском	Название на английском	Тип	Тип ключа	Ссылка
1	id	id	integer	PK	-
2	необходимые_микроэлементы	essential_trace_elements	varchar(30)	-	-
3	любимые_места	favorite_places	varchar(20)	-	-

Таблица 12: сущность питание (nutrition)

2.2 Схема базы данных

На рис.3 и рис.4 приведены схемы базы данных о видах бабочек в Ленинградской области на русском и английском языках соотвественно.

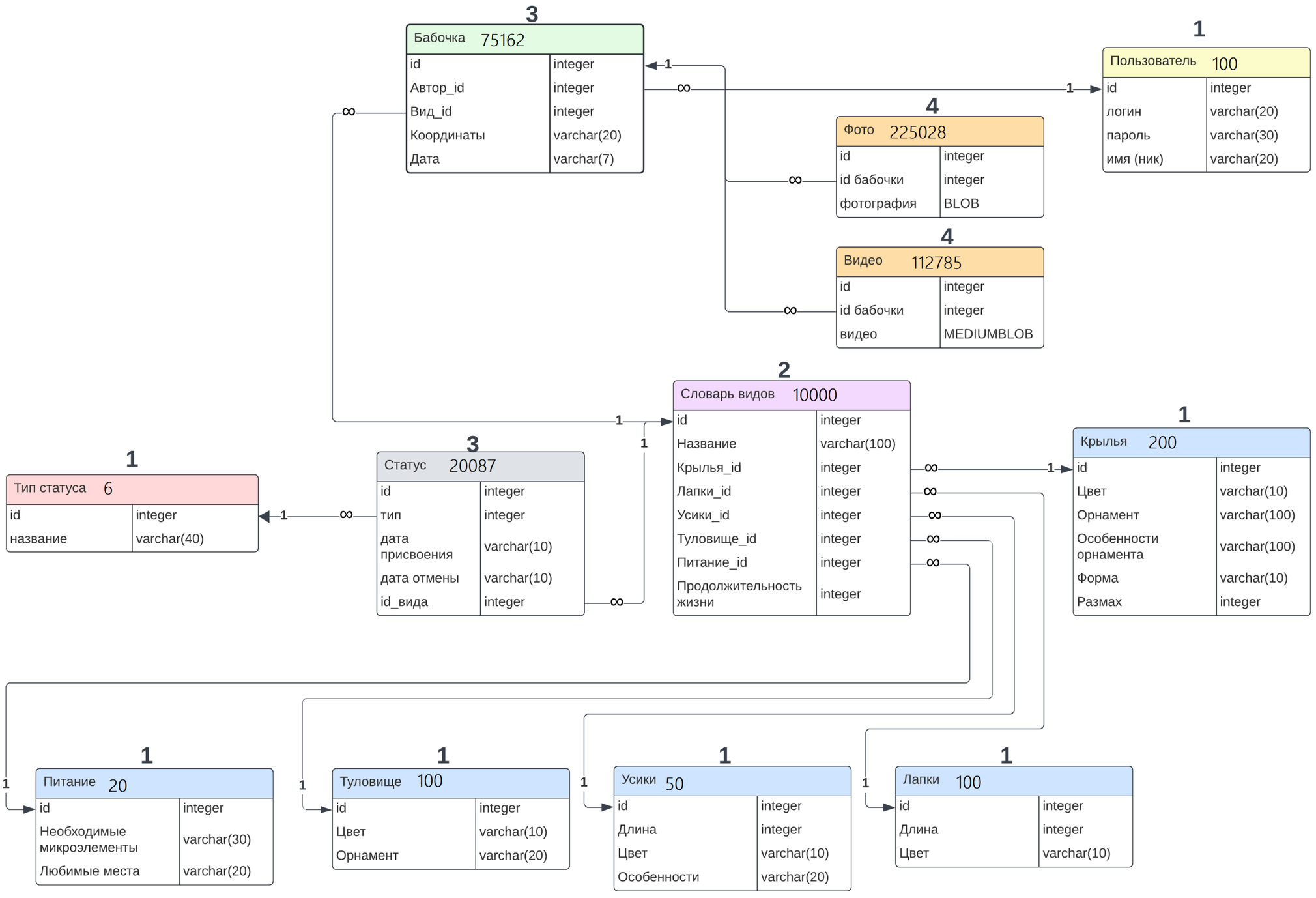


Рис. 3: Схема БД на русском языке

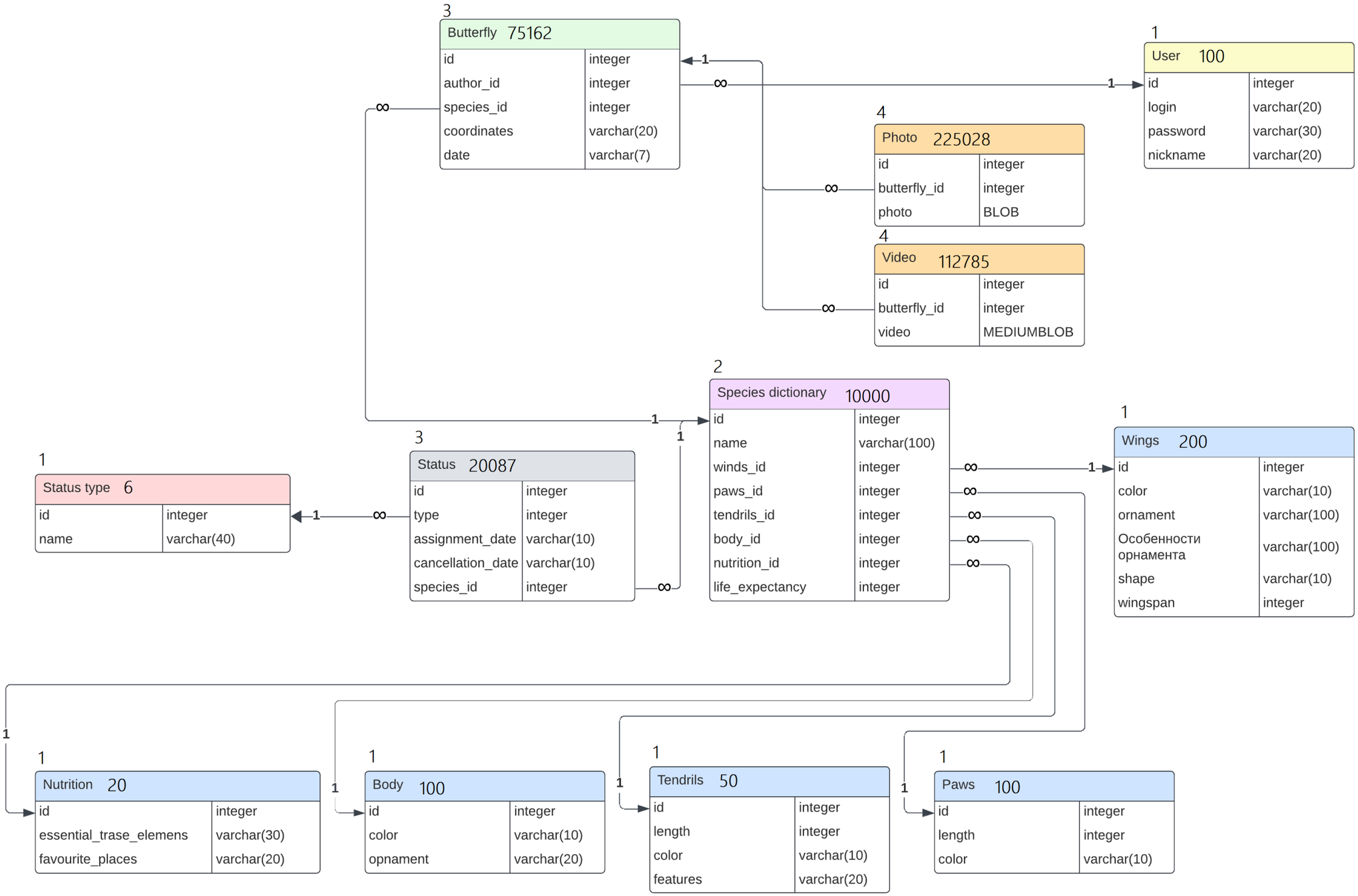


Рис. 4: Схема БД на английском языке

2.3 Заполнение базы данных

Заполнение базы данных происходит в 4 этапа:

1. Пользователь, тип статуса, питание, туловище, усики, лапки, крылья;
2. Словарь видов;
3. Статус, бабочка;
4. Фото, видео.

Для заполнения базы данных был написан скрипт на языке python, он представлен в Приложении В.

Полученное распределение данных:

- В таблице Пользователь сгенерировано 100 записей;
- В таблице Тип статуса сгенерировано 6 записей;
- В таблице Питание сгенерировано 20 записей;
- В таблице Туловище сгенерировано 100 записей;
- В таблице Усики сгенерировано 50 записей;
- В таблице Лапки сгенерировано 100 записей;
- В таблице Крылья сгенерировано 200 записей;
- В таблице Словарь видов сгенерировано 10000 записей;
- В таблице Статус сгенерировано 20087 записи: 1-3 статуса для каждой записи из Словарь видов;
- В таблице Бабочка сгенерировано 75162 записи: 5-10 записей для каждой записи из Словарь видов;
- В таблице Фото сгенерировано 225028 записей: 5-10 записей для каждой записи из Бабочка;
- В таблице Видео сгенерировано 112785 записей: 5-10 записей для каждой записи из Бабочка;

Итоговое количество записей: 443 638.
В таблице 13 приведено количество данных в каждой таблице.

№	Название таблицы	Количество
1	Пользователь	100
2	Тип статуса	6
3	Питание	20
4	Туловище	100
5	Усики	50
6	Лапки	100
7	Крылья	200
8	Словарь видов	10 000
9	Статус	20 087
10	Бабочка	75 162
11	Фото	225 028
12	Видео	112 785
	Итого	443 638

Таблица 13: Количество данных в таблицах

3 Запросы к базе данных

1.1 Найти пользователей, которые встречали бабочку вида A, и у которой статус B;

1.2 Найти пользователей, которые встречали бабочку вида A, и у которой статус B, не менее 3 раз;

2.1 Найти число встреченных бабочек вида A;

2.2 Найти число пользователей, которые повстречали бабочку вида A;

3.1 Для каждого типа статуса посчитать число видов;

3.2 Для каждого типа статуса посчитать число встреченных бабочек каждого вида;

4.1 Найти пользователя(ей) с максимальным числом наблюдений;

4.2 Найти пользователя(ей) с минимальным числом наблюдений;

5 Найти виды бабочек, которые встречались чаще чем A;

6 Найти число видов с одинаковым числом наблюдений;

7 Найти пользователей, которые ни разу не встречали бабочку типа A;

8 Для каждого типа статуса и пользователя посчитать число фото.

3.1 Запрос 1.1

Найти пользователей, которые встречали бабочку вида A, и у которой статус B.

- A = name 15
- B = likely extinct

Код запроса:

```
SELECT u.user_name, u.user_login, u.user_password
FROM user_ u
JOIN butterfly b ON u.id = b.author_id
JOIN species_book sb ON b.species_id = sb.id
JOIN status s ON sb.id = s.species_id
JOIN status_type st ON s.status_type = st.id
WHERE sb.species_name = 'name 15'
AND st.status_name = 'likely extinct';
```

Результат запроса:

```
user_name    | user_login |          user_password
-------------+------------+-------------------------------
MatthewvHmM  | Matthew    | d8sl5Bnv1YuN9IVsBEb6tk8M1SgO3
Jacob6Lj     | Jacob      | aSUQRlt5Bh0zNPDnC2i9nZJ
Conniep      | Connie     | Nq5nlLriZyi2z4ZKBs04M
Keith01      | Keith      | 6CLfB62MaVaa2GLtPprGW0UQ
Jeffrey3Us   | Jeffrey    | ruKWxcUCsiwvbWfOOpqAO
Dariuscc     | Darius     | 8hCsGZncfXSPrWKYNneraNlCjtKiK
SethBvh2     | Seth       | UNX6vLcK0wmPvk2QwlCo13Ynphfu
Sean8H44Y    | Sean       | 3Cuowm4xiEOjO7yAaU9Yi
RyanCo       | Ryan       | 41IAouu48Kfib7CI5CFpj8
(9 rows)
```

```
Total query runtime: 110 msec.
```

План запроса:

```
Nested Loop  (cost=1.16..17.85 rows=1 width=41)
   ->  Nested Loop  (cost=1.01..17.69 rows=1 width=4)
         Join Filter: (b.species_id = sb.id)
         ->  Nested Loop  (cost=0.72..17.00 rows=1 width=8)
               ->  Nested Loop  (cost=0.57..16.65 rows=2 width=12)
                     ->  Index Scan using idx_species_book_species_name on
                     ↪  species_book sb  (cost=0.29..8.30 rows=1 width=4)
                           Index Cond: ((species_name)::text = 'name 15'::text)
                     ->  Index Scan using idx_status_species_id on status s
                     ↪  (cost=0.29..8.32 rows=2 width=8)
                           Index Cond: (species_id = sb.id)
               ->  Index Scan using status_type_pkey on status_type st
               ↪  (cost=0.15..0.17 rows=1 width=4)
                     Index Cond: (id = s.status_type)
                     Filter: ((status_name)::text = 'likely extinct'::text)
         ->  Index Scan using idx_butterfly_species_id on butterfly b
         ↪  (cost=0.29..0.59 rows=8 width=8)
               Index Cond: (species_id = s.species_id)
   ->  Index Scan using user__pkey on user_ u  (cost=0.14..0.16 rows=1 width=45)
         Index Cond: (id = b.author_id)
```

Объяснение запроса:
В запросе выбираются имена, логины и пароли пользователей из таблицы user_. Данные объединяются из таблиц user_, butterfly, species_book, status и status_type. Фильтрация данных осуществляется для выбора только тех пользователей, которые зарегистрировали бабочку вида 'name 15' со статусом 'likely extinct'.

3.2 Запрос 1.2

Найти пользователей, которые встречали бабочку вида A, и у которой статус B, не менее 3 раз.

- A = name 15
- B = likely extinct

Код запроса:

```
SELECT u.user_name, u.user_login, u.user_password
FROM user_ u
JOIN butterfly b ON u.id = b.author_id
JOIN species_book sb ON b.species_id = sb.id
JOIN status s ON sb.id = s.species_id
JOIN status_type st ON s.status_type = st.id
WHERE sb.species_name = 'name 15'
  AND st.status_name = 'likely extinct'
GROUP BY u.id, u.user_name, u.user_login, u.user_password
HAVING COUNT(b.id) >= 3;
```

Результат запроса:

```
 user_name | user_login | user_password
-----------+------------+---------------
(0 rows)

Successfully run. Total query runtime: 67 msec.
```

План запроса:

```
1  GroupAggregate  (cost=17.86..17.88 rows=1 width=45)
2   Group Key: u.id
3   Filter: (count(b.id) >= 3)
4   ->  Sort  (cost=17.86..17.87 rows=1 width=49)
5         Sort Key: u.id
6         ->  Nested Loop  (cost=1.16..17.85 rows=1 width=49)
7               ->  Nested Loop  (cost=1.01..17.69 rows=1 width=8)
8                     Join Filter: (b.species_id = sb.id)
9                     ->  Nested Loop  (cost=0.72..17.00 rows=1 width=8)
10                          ->  Nested Loop  (cost=0.57..16.65 rows=2 width=12)
11                                ->  Index Scan using idx_species_book_species_name
                                  ↪  on species_book sb  (cost=0.29..8.30 rows=1
                                  ↪  width=4)
12                                      Index Cond: ((species_name)::text = 'name
                                        ↪  15'::text)
13                                ->  Index Scan using idx_status_species_id on status
                                  ↪  s  (cost=0.29..8.32 rows=2 width=8)
14                                      Index Cond: (species_id = sb.id)
15                          ->  Index Scan using status_type_pkey on status_type st
                            ↪  (cost=0.15..0.17 rows=1 width=4)
16                                Index Cond: (id = s.status_type)
17                                Filter: ((status_name)::text = 'likely
                                  ↪  extinct'::text)
18                    ->  Index Scan using idx_butterfly_species_id on butterfly b
                      ↪  (cost=0.29..0.59 rows=8 width=12)
19                          Index Cond: (species_id = s.species_id)
20              ->  Index Scan using user__pkey on user_ u  (cost=0.14..0.16 rows=1
                ↪  width=45)
21                    Index Cond: (id = b.author_id)
```

Объяснение запроса:
Выбираются имена, логины и пароли пользователей из таблицы user_. Он соединяет данные из нескольких таблиц: user_, butterfly, species_book, status и status_type. В запросе фильтруются данные, чтобы выбрать только тех пользователей, для которых выполняются условия A и B. После этого результаты группируются по пользователям, и применяется условие HAVING COUNT(b.id) >= 3, чтобы выбрать только тех пользователей, у которых количество записей о бабочках не менее 3.

3.3 Запрос 2.1

Найти число встреченных бабочек вида A.

- A = name 5567

Код запроса:

```
SELECT COUNT(*)
FROM butterfly b
JOIN species_book sb ON b.species_id = sb.id
WHERE sb.species_name = 'name 5567';
```

Результат запроса:

```
 count
-------
     9
(1 row)

Total query runtime: 52 msec.
```

План запроса:

```
1  Aggregate  (cost=12.84..12.85 rows=1 width=8)
2    ->  Nested Loop  (cost=0.58..12.82 rows=8 width=0)
3          ->  Index Scan using idx_species_book_species_name on species_book sb
           ↪  (cost=0.29..8.30 rows=1 width=4)
4                Index Cond: ((species_name)::text = 'name 5567'::text)
5          ->  Index Only Scan using idx_butterfly_species_id on butterfly b
           ↪  (cost=0.29..4.43 rows=8 width=4)
6                Index Cond: (species_id = sb.id)
```

Объяснение запроса:
Выбирается количество записей в таблице butterfly, соответствующих виду бабочки A, с помощью соединения таблицы butterfly с таблицей species_book по идентификатору вида. Фильтрация данных осуществляется по имени вида бабочки A.

3.4 Запрос 2.2

Найти число пользователей, которые повстречали бабочку вида A.

- A = name 5567

Код запроса:

```
SELECT COUNT(DISTINCT u.id) AS user_count
FROM user_ u
JOIN butterfly b ON u.id = b.author_id
JOIN species_book sb ON b.species_id = sb.id
WHERE sb.species_name = 'name 5567';
```

Результат запроса:

```
 user_count
------------
```

```
          8
(1 row)

Total query runtime: 62 msec.
```

План запроса:

```
1  Aggregate  (cost=18.26..18.27 rows=1 width=8)
2   ->  Sort  (cost=18.22..18.24 rows=8 width=4)
3         Sort Key: u.id
4         ->  Nested Loop  (cost=0.72..18.10 rows=8 width=4)
5               ->  Nested Loop  (cost=0.58..16.81 rows=8 width=4)
6                     ->  Index Scan using idx_species_book_species_name on
                      ↪  species_book sb  (cost=0.29..8.30 rows=1 width=4)
7                           Index Cond: ((species_name)::text = 'name 5567'::text)
8                     ->  Index Scan using idx_butterfly_species_id on butterfly b
                      ↪  (cost=0.29..8.43 rows=8 width=8)
9                           Index Cond: (species_id = sb.id)
10              ->  Index Only Scan using user__pkey on user_ u  (cost=0.14..0.16
                ↪  rows=1 width=4)
11                    Index Cond: (id = b.author_id)
```

Объяснение запроса:
Выбирается количество уникальных записей в таблице user_, которые встречали бабочку вида A. Для этого сначала таблица user_ соединяется с таблицей butterfly по идентификатору автора бабочки. Затем таблица butterfly соединяется с таблицей species_book по идентификатору вида бабочки. Фильтрация данных осуществляется по имени вида бабочки A, и подсчитывается количество уникальных пользователей, которые встречали данный вид бабочки.

3.5 Запрос 3.1

Для каждого типа статуса посчитать число видов.

Код запроса:

```
SELECT st.status_name, COUNT(DISTINCT sb.id) AS species_count
FROM status s
JOIN status_type st ON s.status_type = st.id
JOIN species_book sb ON s.species_id = sb.id
GROUP BY st.status_name;
```

Результат запроса:

```
                status_name                | species_count
-------------------------------------------+---------------
 declining in population and/or distribution |          2984
 endangered                                  |          3075
 likely extinct                              |          2935
 rare                                        |          3033
 recoverable and recovering                  |          2970
 undetermined status                         |          2973
(6 rows)
```

```
Total query runtime: 96 msec
```

План запроса:

```
GroupAggregate  (cost=2240.23..2392.88 rows=200 width=146)
  Group Key: st.status_name
  ->  Sort  (cost=2240.23..2290.45 rows=20087 width=142)
        Sort Key: st.status_name, sb.id
        ->  Hash Join  (cost=329.80..804.62 rows=20087 width=142)
              Hash Cond: (s.species_id = sb.id)
              ->  Hash Join  (cost=20.80..442.87 rows=20087 width=142)
                    Hash Cond: (s.status_type = st.id)
                    ->  Seq Scan on status s  (cost=0.00..368.87 rows=20087 width=8)
                    ->  Hash  (cost=14.80..14.80 rows=480 width=142)
                          ->  Seq Scan on status_type st  (cost=0.00..14.80 rows=480
                          ↪  width=142)
              ->  Hash  (cost=184.00..184.00 rows=10000 width=4)
                    ->  Seq Scan on species_book sb  (cost=0.00..184.00 rows=10000
                    ↪  width=4)
```

Объяснение запроса:
Для каждого типа статуса подсчитывается количество уникальных видов. Сначала таблица status соединяется с таблицей status_type по идентификатору типа статуса. Затем таблица status соединяется с таблицей species_book по идентификатору вида. После этого данные группируются по названию типа статуса, и для каждого типа подсчитывается количество уникальных видов.

На рис.5 представлена гистограмма для запроса 3.1.

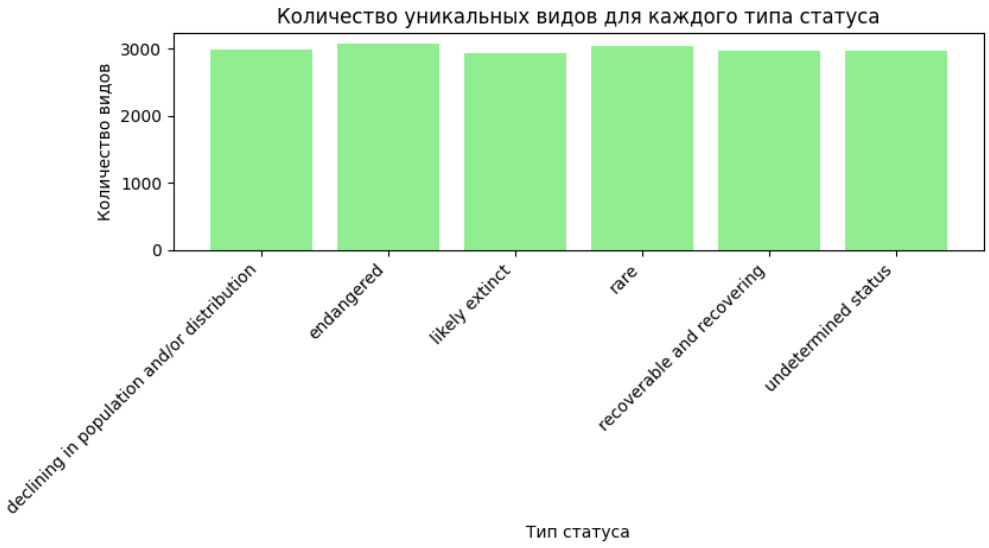


Рис. 5: Гистограмма для запроса 3.1

3.6 Запрос 3.2

Для каждого типа статуса посчитать число встреченных бабочек каждого вида.

Код запроса:

```
SELECT st.status_name, COUNT(b.id) AS butterfly_count
FROM status s
JOIN status_type st ON s.status_type = st.id
JOIN species_book sb ON s.species_id = sb.id
JOIN butterfly b ON sb.id = b.species_id
GROUP BY st.status_name
ORDER BY st.status_name;
```

Результат запроса:

```
                status_name                 | butterfly_count
--------------------------------------------+-----------------
 declining in population and/or distribution |           24863
 endangered                                 |           26092
 likely extinct                             |           24622
 rare                                       |           25440
 recoverable and recovering                 |           25333
 undetermined status                        |           24850
(6 rows)

Total query runtime: 107 msec.
```

План запроса:

```
1  Sort  (cost=5254.37..5254.87 rows=200 width=146)
2    Sort Key: st.status_name
3    ->  HashAggregate  (cost=5244.73..5246.73 rows=200 width=146)
4          Group Key: st.status_name
5          ->  Hash Join  (cost=1055.70..4489.84 rows=150978 width=142)
6                Hash Cond: (b.species_id = s.species_id)
7                ->  Seq Scan on butterfly b  (cost=0.00..1454.62 rows=75162 width=8)
8                ->  Hash  (cost=804.62..804.62 rows=20087 width=146)
9                      ->  Hash Join  (cost=329.80..804.62 rows=20087 width=146)
10                           Hash Cond: (s.species_id = sb.id)
11                           ->  Hash Join  (cost=20.80..442.87 rows=20087 width=142)
12                                 Hash Cond: (s.status_type = st.id)
13                                 ->  Seq Scan on status s  (cost=0.00..368.87
                                   ↪  rows=20087 width=8)
14                                 ->  Hash  (cost=14.80..14.80 rows=480 width=142)
15                                       ->  Seq Scan on status_type st
                                         ↪  (cost=0.00..14.80 rows=480 width=142)
16                           ->  Hash  (cost=184.00..184.00 rows=10000 width=4)
17                                 ->  Seq Scan on species_book sb  (cost=0.00..184.00
                                   ↪  rows=10000 width=4)
```

Объяснение запроса:
Для каждого типа статуса подсчитывается количество бабочек. Сначала таблицы status, status_type, species_book и butterfly объединяются с помощью операторов JOIN, чтобы получить необходимые данные. Затем результаты группируются по названию типа статуса, и для каждого типа подсчитывается количество уникальных бабочек.

На рис.6 представлена гистограмма для запроса 3.2.

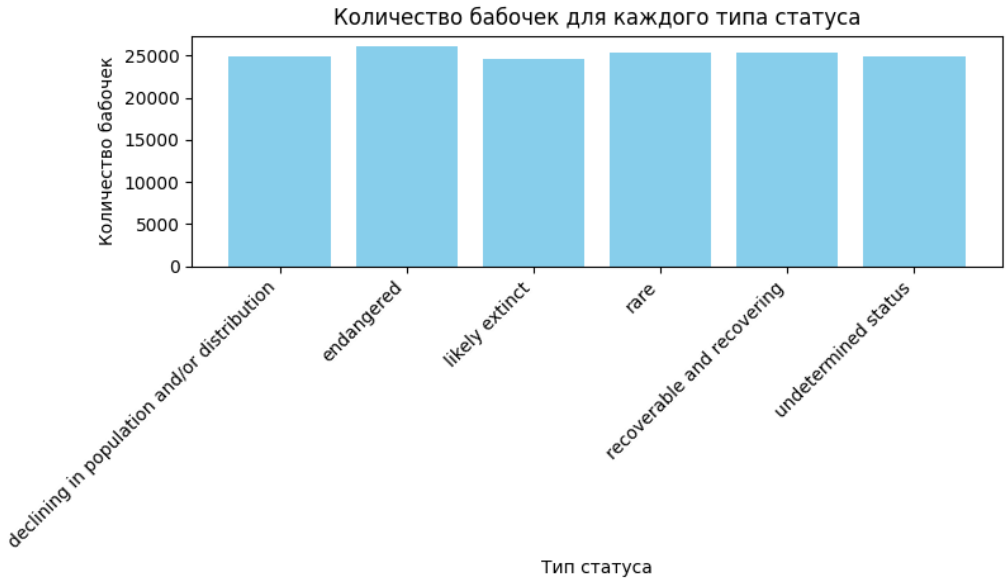


Рис. 6: Гистограмма для запроса 3.2

3.7 Запрос 4.1

Найти пользователя(ей) с максимальным числом наблюдений;

Код запроса:

```
SELECT u.user_name, COUNT(b.id) AS observation_count
FROM user_ u
JOIN butterfly b ON u.id = b.author_id
GROUP BY u.user_name
HAVING COUNT(b.id) = (
    SELECT COUNT(b.id) AS max_observation_count
    FROM butterfly b
    JOIN user_ u ON u.id = b.author_id
    GROUP BY u.id
    ORDER BY max_observation_count DESC
    LIMIT 1
);
```

Результат запроса:

```
   user_name     | observation_count
-----------------+-------------------
 ElizabethNrpc5 |               835
(1 row)

Total query runtime: 197 msec.
```

План запроса:

```
1  HashAggregate  (cost=4080.19..4081.44 rows=1 width=18)
2    Group Key: u.user_name
```

```
  Filter: (count(b.id) = $0)
  InitPlan 1 (returns $0)
    ->  Limit  (cost=2040.85..2040.85 rows=1 width=12)
          ->  Sort  (cost=2040.85..2041.10 rows=100 width=12)
                Sort Key: (count(b_1.id)) DESC
                ->  HashAggregate  (cost=2039.35..2040.35 rows=100 width=12)
                      Group Key: u_1.id
                      ->  Hash Join  (cost=3.25..1663.54 rows=75162 width=8)
                            Hash Cond: (b_1.author_id = u_1.id)
                            ->  Seq Scan on butterfly b_1  (cost=0.00..1454.62
                            ↪  rows=75162 width=8)
                            ->  Hash  (cost=2.00..2.00 rows=100 width=4)
                                  ->  Seq Scan on user_ u_1  (cost=0.00..2.00
                                  ↪  rows=100 width=4)
  ->  Hash Join  (cost=3.25..1663.54 rows=75162 width=14)
        Hash Cond: (b.author_id = u.id)
        ->  Seq Scan on butterfly b  (cost=0.00..1454.62 rows=75162 width=8)
        ->  Hash  (cost=2.00..2.00 rows=100 width=14)
              ->  Seq Scan on user_ u  (cost=0.00..2.00 rows=100 width=14)
```

Объяснение запроса:
Для каждого пользователя подсчитывается количество наблюдений. Сначала таблицы user_ и butterfly объединяются с помощью оператора JOIN по идентификатору пользователя. Затем результаты группируются по имени пользователя. После этого используется условие HAVING для фильтрации только тех пользователей, у которых количество наблюдений равно максимальному числу наблюдений среди всех пользователей. Максимальное количество наблюдений находится с помощью вложенного запроса, который сначала подсчитывает количество наблюдений для каждого пользователя, затем сортирует результаты по убыванию и ограничивает количество строк одной, чтобы получить максимальное количество наблюдений.

3.8 Запрос 4.2

Найти пользователя(ей) с минимальным числом наблюдений;

Код запроса:

```
SELECT u.user_name, COUNT(b.id) AS observation_count
FROM user_ u
LEFT JOIN butterfly b ON u.id = b.author_id
GROUP BY u.user_name
HAVING COUNT(b.id) = (
    SELECT COUNT(b.id) AS min_observation_count
    FROM butterfly b
    JOIN user_ u ON u.id = b.author_id
    GROUP BY u.id
    ORDER BY min_observation_count ASC
    LIMIT 1
);
```

Результат запроса:

```
 user_name  | observation_count
------------+-------------------
```

```
 RickybmCMk |                  687
(1 row)

Total query runtime: 154 msec.
```

План запроса:

```
HashAggregate  (cost=4080.19..4081.44 rows=1 width=18)
  Group Key: u.user_name
  Filter: (count(b.id) = $0)
  InitPlan 1 (returns $0)
    ->  Limit  (cost=2040.85..2040.85 rows=1 width=12)
          ->  Sort  (cost=2040.85..2041.10 rows=100 width=12)
                Sort Key: (count(b_1.id))
                ->  HashAggregate  (cost=2039.35..2040.35 rows=100 width=12)
                      Group Key: u_1.id
                      ->  Hash Join  (cost=3.25..1663.54 rows=75162 width=8)
                            Hash Cond: (b_1.author_id = u_1.id)
                            ->  Seq Scan on butterfly b_1  (cost=0.00..1454.62
                            ↪  rows=75162 width=8)
                            ->  Hash  (cost=2.00..2.00 rows=100 width=4)
                                  ->  Seq Scan on user_ u_1  (cost=0.00..2.00
                                  ↪  rows=100 width=4)
  ->  Hash Right Join  (cost=3.25..1663.54 rows=75162 width=14)
        Hash Cond: (b.author_id = u.id)
        ->  Seq Scan on butterfly b  (cost=0.00..1454.62 rows=75162 width=8)
        ->  Hash  (cost=2.00..2.00 rows=100 width=14)
              ->  Seq Scan on user_ u  (cost=0.00..2.00 rows=100 width=14)
```

Объяснение запроса:
Для каждого пользователя подсчитывается количество наблюдений. Сначала таблицы user_ и butterfly объединяются с помощью оператора JOIN по идентификатору пользователя. Затем результаты группируются по имени пользователя. После этого используется условие HAVING для фильтрации только тех пользователей, у которых количество наблюдений равно минимальному числу наблюдений среди всех пользователей. Минимальное количество наблюдений находится с помощью вложенного запроса, который сначала подсчитывает количество наблюдений для каждого пользователя, затем сортирует результаты по возрастанию и ограничивает количество строк одной, чтобы получить минимальное количество наблюдений.

3.9 Запрос 5

Найти виды бабочек, которые встречались чаще чем A.

- A = name 2

Код запроса:

```
SELECT sb.species_name, COUNT(*) AS observation_count
FROM species_book sb
JOIN butterfly b ON sb.id = b.species_id
GROUP BY sb.species_name
HAVING COUNT(*) > (
    SELECT COUNT(*)
```

```
    FROM species_book
    JOIN butterfly ON species_book.id = butterfly.species_id
    WHERE species_name = 'name 2'
);
```

Результат запроса:

```
 species_name | observation_count
--------------+-------------------
 name 4338    |                10
 name 3954    |                10
 name 1077    |                10
 name 8328    |                10
 name 7499    |                10
 name 9604    |                10
 name 2078    |                10
 name 8194    |                10
 name 421     |                10
 name 9601    |                10
 name 7732    |                10
 name 3779    |                10
 name 415     |                10
 name 3747    |                10
 name 2895    |                10
 name 9216    |                10
 name 764     |                10
 name 3604    |                10
 name 6615    |                10
 name 5207    |                10
 name 2841    |                10
 name 6685    |                10
 name 4687    |                10
 name 8496    |                10
 name 3775    |                10
 name 1905    |                10
 name 1745    |                10
--            --
```

```
(1666 row)

Total query runtime: 95 msec.
```

План запроса:

```
1  HashAggregate  (cost=2349.66..2474.66 rows=3333 width=17)
2    Group Key: sb.species_name
3    Filter: (count(*) > $1)
4    InitPlan 1 (returns $1)
5      ->  Aggregate  (cost=12.84..12.85 rows=1 width=8)
6            ->  Nested Loop  (cost=0.58..12.82 rows=8 width=0)
7                  ->  Index Scan using idx_species_book_species_name on species_book
                   ↪  (cost=0.29..8.30 rows=1 width=4)
```

```
                        Index Cond: ((species_name)::text = 'name 2'::text)
                  ->  Index Only Scan using idx_butterfly_species_id on butterfly
                  ↪  (cost=0.29..4.43 rows=8 width=4)
                        Index Cond: (species_id = species_book.id)
   ->  Hash Join  (cost=309.00..1961.01 rows=75162 width=9)
         Hash Cond: (b.species_id = sb.id)
         ->  Seq Scan on butterfly b  (cost=0.00..1454.62 rows=75162 width=4)
         ->  Hash  (cost=184.00..184.00 rows=10000 width=13)
               ->  Seq Scan on species_book sb  (cost=0.00..184.00 rows=10000
               ↪  width=13)
```

Объяснение запроса:
Этот запрос выбирает названия видов бабочек (species_name) из таблицы species_book и подсчитывает количество наблюдений для каждого вида, сгруппированных по названию вида. Затем используется условие HAVING, чтобы фильтровать только те виды бабочек, у которых количество наблюдений больше, чем количество наблюдений для бабочки с названием A.

3.10 Запрос 6

Найти число видов с одинаковым числом наблюдений.

Код запроса:

```
SELECT observation_count, COUNT(*) AS species_count
FROM (
    SELECT COUNT(*) AS observation_count
    FROM butterfly
    GROUP BY species_id
) AS observations_per_species
GROUP BY observation_count
ORDER BY observation_count;
```

Результат запроса:

```
observation_count | species_count
------------------+--------------
                5 |         1637
                6 |         1644
                7 |         1686
                8 |         1652
                9 |         1715
               10 |         1666
(6 rows)
Total query runtime: 122 msec.
```

План запроса:

```
 Sort  (cost=2089.07..2089.57 rows=200 width=16)
   Sort Key: (count(*))
   ->  HashAggregate  (cost=2079.43..2081.43 rows=200 width=16)
         Group Key: count(*)
         ->  HashAggregate  (cost=1830.43..1930.03 rows=9960 width=12)
```

```
6                 Group Key: butterfly.species_id
7                 ->  Seq Scan on butterfly  (cost=0.00..1454.62 rows=75162
                  ↪  width=4)
```

Объяснение запроса:
Этот запрос выполняет анализ числа видов бабочек с одинаковым числом наблюдений. Внутренний запрос SELECT COUNT(*) AS observation_count FROM butterfly GROUP BY species_id подсчитывает количество наблюдений для каждого вида путем группировки данных в таблице butterfly по species_id и подсчета числа строк в каждой группе. Внешний запрос SELECT observation_count, COUNT(*) AS species_count FROM (...) AS observations_per_species GROUP BY observation_count группирует результаты внутреннего запроса по числу наблюдений (observation_count) и подсчитывает количество видов (species_count), у которых одинаковое число наблюдений. Результаты сортируются по числу наблюдений.

На рис.7 представлена гистограмма для запроса 6.

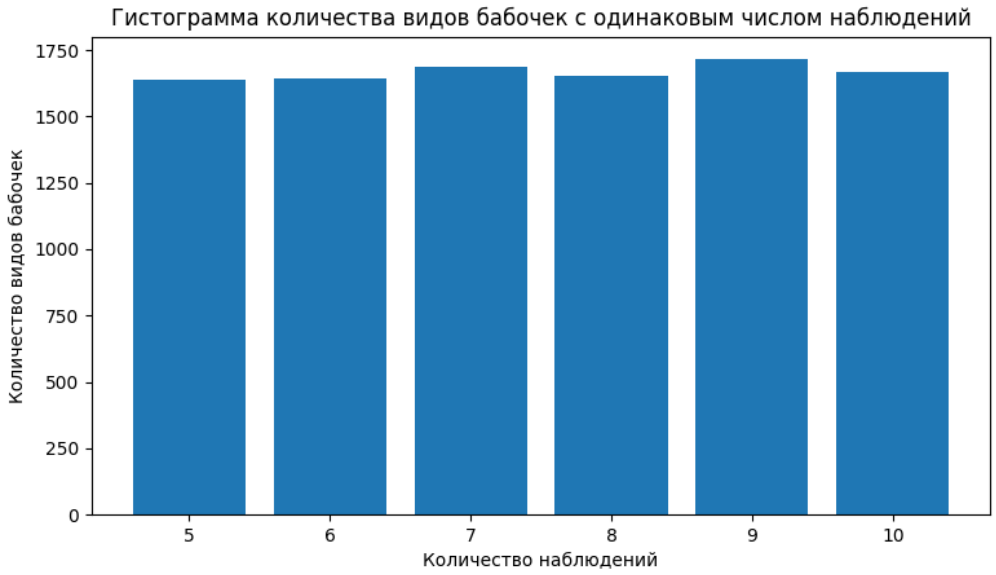


Рис. 7: Гистограмма для запроса 6

3.11 Запрос 7

Найти пользователей, которые ни разу не встречали бабочку типа A.

- A = name 1

Код запроса:

```
SELECT u.user_name
FROM user_ u
WHERE NOT EXISTS (
    SELECT 1
    FROM butterfly b
    JOIN species_book sb ON b.species_id = sb.id
    WHERE u.id = b.author_id AND sb.species_name = 'name 35674'
);
```

Результат запроса:

```
    user_name
-----------------
 EricGE
 Zachary10
 JasonZTAw
 JosephsUCaR
 JesusHRw
 Keith01
 AnthonylQZ
 EbonyJd
 Kristin1QC20
 RachelhjUG
 TomrsY
 Sean8H44Y
 StaceyN
 RickybmCMk
 Scottxt1
 RodneyZ
 JamesEON
 ElizabethNrpc5
 BrianaZXv
 DebraTalpz
 SethBvh2
 Jessicauj
 Bradleyceqh
 Heather3aC
 JamesYcv
 DanielleVXmbu
 TravisYduu
(100 rows)

Total query runtime: 65 msec.
```

План запроса:

```
 Hash Right Anti Join  (cost=3.83..20.10 rows=92 width=10)
  Hash Cond: (b.author_id = u.id)
  ->  Nested Loop  (cost=0.58..16.81 rows=8 width=4)
        ->  Index Scan using idx_species_book_species_name on species_book sb
        ↪  (cost=0.29..8.30 rows=1 width=4)
              Index Cond: ((species_name)::text = 'name 35674'::text)
        ->  Index Scan using idx_butterfly_species_id on butterfly b
        ↪  (cost=0.29..8.43 rows=8 width=8)
              Index Cond: (species_id = sb.id)
  ->  Hash  (cost=2.00..2.00 rows=100 width=14)
        ->  Seq Scan on user_ u  (cost=0.00..2.00 rows=100 width=14)
```

Объяснение запроса:
Для каждого пользователя проверяется с помощью подзапроса NOT EXISTS, есть ли у него наблюдения для бабочек типа "A". Подзапрос выполняет соединение таблицы butterfly с таблицей

species_book, чтобы найти наблюдения для бабочек типа "A среди которых пользователь является автором. Если подзапрос не возвращает ни одной строки (т.е. для данного пользователя не существует наблюдений бабочек типа "A"), то этот пользователь включается в результирующий набор данных.

3.12 Запрос 8

Для каждого типа статуса и пользователя посчитать число фото.

Код запроса:

```
SELECT
    st.status_name,
    u.user_name,
    (
        SELECT COUNT(p.id)
        FROM photo p
        JOIN butterfly b ON p.butterfly_id = b.id
        JOIN species_book sb ON b.species_id = sb.id
        JOIN status s ON sb.id = s.species_id AND b.author_id = u.id
        WHERE s.status_type = st.id
    ) AS photo_count
FROM
    status_type st
CROSS JOIN
    user_ u;
```

Результат запроса:

```
                status_name                |    user_name     | photo_count
-------------------------------------------+------------------+-------------
 likely extinct                            | JaimeP           |         671
 likely extinct                            | Jessicauj        |         764
 likely extinct                            | AndrewAD         |         766
 likely extinct                            | MelindaiK6       |         663
 likely extinct                            | MatthewpL48d     |         734
 likely extinct                            | Barbara00F       |         822
 likely extinct                            | Melanie8W        |         677
 likely extinct                            | Susan15gs        |         732
 likely extinct                            | RitaW            |         761
 likely extinct                            | JamesEON         |         801
 likely extinct                            | AnthonylQZ       |         775
 likely extinct                            | RodneyZ          |         711
 likely extinct                            | Michaeli1d       |         819
 likely extinct                            | MatthewvHmM      |         740
 likely extinct                            | TravisYduu       |         858
 likely extinct                            | ChristinaRZc     |         665
 likely extinct                            | Kristin1QC20     |         644
 likely extinct                            | ElizabethNrpc5   |         840
 likely extinct                            | AngelaNACX       |         720
 likely extinct                            | EricGE           |         721
 likely extinct                            | RebekahDeg       |         829
```

```
 likely extinct                                  | Shannon3qb8      |         611
 likely extinct                                  | DavidSNo         |         665
 likely extinct                                  | JonathanC        |         809
 likely extinct                                  | JasonZTAw        |         660
 likely extinct                                  | JosephsUCaR      |         764
 likely extinct                                  | JesusHRw         |         675
-- Далее  --

(600 rows)

Total query runtime: 5 secs 612 msec.
```

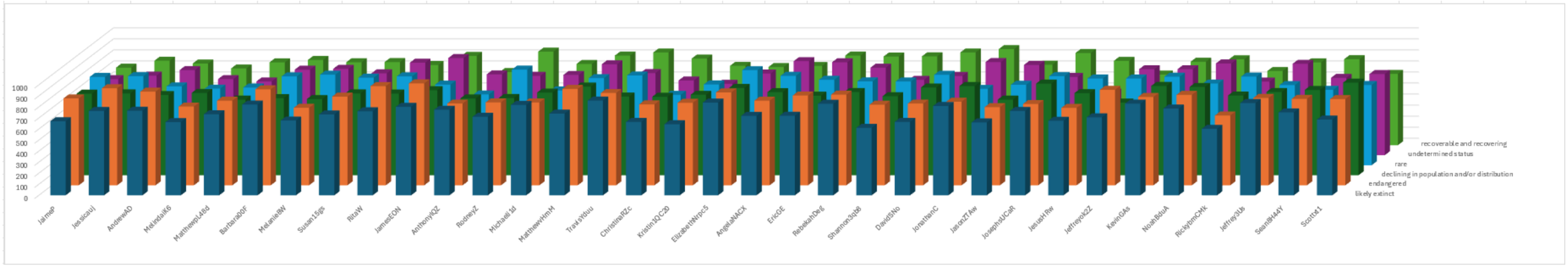
План запроса:

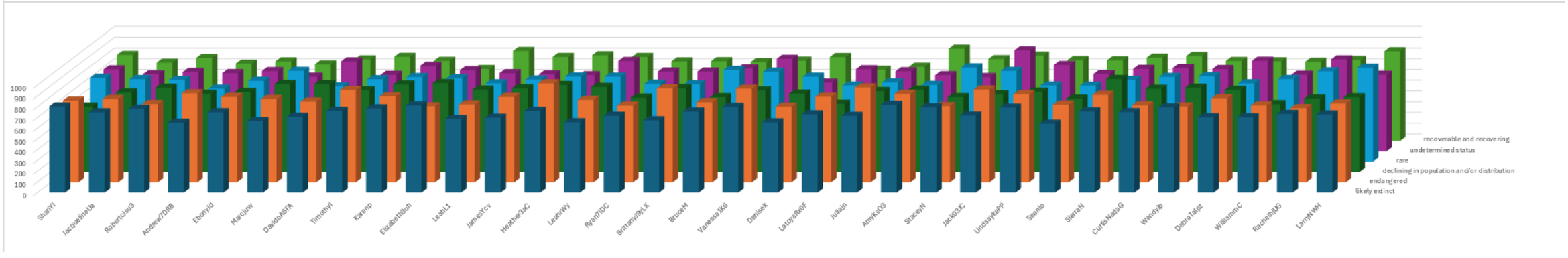
```
 Nested Loop  (cost=0.00..123494011.79 rows=48000 width=156)
  ->  Seq Scan on status_type st  (cost=0.00..14.80 rows=480 width=142)
  ->  Materialize  (cost=0.00..2.50 rows=100 width=14)
        ->  Seq Scan on user_ u  (cost=0.00..2.00 rows=100 width=14)
  SubPlan 1
    ->  Aggregate  (cost=2572.77..2572.78 rows=1 width=8)
          ->  Nested Loop  (cost=308.64..2570.88 rows=754 width=4)
                ->  Nested Loop  (cost=308.34..1145.82 rows=252 width=4)
                      Join Filter: (b.species_id = sb.id)
                      ->  Hash Join  (cost=308.06..1057.41 rows=252 width=12)
                            Hash Cond: (b.species_id = s.species_id)
                            ->  Bitmap Heap Scan on butterfly b  (cost=14.12..756.26
                            ↪  rows=752 width=8)
                                  Recheck Cond: (author_id = u.id)
                                  ->  Bitmap Index Scan on idx_butterfly_author_id
                                  ↪  (cost=0.00..13.93 rows=752 width=0)
                                  ↪  Index Cond: (author_id = u.id)
                            ->  Hash  (cost=252.08..252.08 rows=3348 width=4)
                                  ->  Bitmap Heap Scan on status s
                                  ↪  (cost=42.23..252.08 rows=3348 width=4)
                                        Recheck Cond: (status_type = st.id)
                                        ->  Bitmap Index Scan on
                                        ↪  idx_status_status_type
                                        ↪  (cost=0.00..41.40 rows=3348 width=0)
                                              Index Cond: (status_type = st.id)
                      ->  Index Only Scan using species_book_pkey on species_book sb
                      ↪  (cost=0.29..0.34 rows=1 width=4)
                            Index Cond: (id = s.species_id)
                ->  Index Scan using idx_photo_butterfly_id on photo p
                ↪  (cost=0.29..5.61 rows=4 width=8)
                      Index Cond: (butterfly_id = b.id)
```

Объяснение запроса:
Этот запрос выполняет выборка статусов и пользователей: Из таблицы status_type выбираются все возможные типы статусов, а из таблицы user_ выбираются все пользователи. CROSS JOIN используется для создания комбинации каждого типа статуса со всеми пользователями, что позволяет получить все возможные комбинации статусов и пользователей. Подзапрос для подсчета количества фотографий: Для каждой комбинации статуса и пользователя используется

подзапрос, который выполняет следующие действия: считает количество фотографий, сделанных пользователями с определенным статусом, присоединяет таблицу photo и находит количество фотографий, сделанных бабочками, соответствующими этому статусу и этому пользователю. Использует таблицы butterfly, species_book и status для нахождения бабочек, которые были сфотографированы пользователем с определенным статусом. Результатом запроса является список всех возможных комбинаций статусов и пользователей, включая количество фотографий, сделанных каждым пользователем с каждым статусом.

На рис.8 представлена гистограмма для запроса 8.





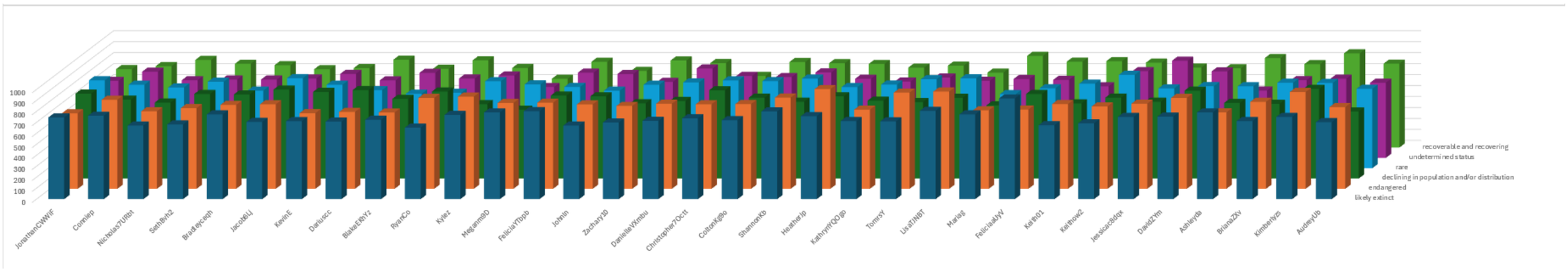


Рис. 8: Гистограмма для запроса 8

Заключение

Данная курсовая работа заключалась в создании базы данных для предметной области "Виды бабочек в Ленинградской области". Работа проведена в СУБД PosgreSQL 16.2.
Была построена ER-диаграмма и схема объектов, отращающая сущности предметной области и их отношения.

ER-диаграмма состоит из 8 сущностей и 10 компонент отношений. Схема объектов состоит из 11 объектов.

На основе созданных диаграмм были описаны таблицы базы данных на языке описания данных в СУБД PosgreSQL. Для каждой таблицы были расписаны атрибуты и связи с другими таблицами. Генерация данных была написана на языке python с использованием библиотеки psycopg2. Всего было сгенерировано 443 638 записей. Было реализовано 8 запросов на языке SQL, также приведело EXPLAIN (объяснение) каждого запроса.

Для ускорения запросов были использованы индексы.

В рамках данной курсовой работы были получены навыки анализа предметной области с точки зрения формального описания и анализа отношений сущностей.

Список литературы

[1] PostgreSQL: Докуметнация // PostgreSQL URL: https://www.postgresql.org/docs/ (дата обращения: 26.05.2024).

[2] OpenDota: Docs // OpenDota URL: https://docs.opendota.com/ (дата обращения: 26.05.2024).

[3] Основы технологий баз данных: учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. — 2-е изд. — М.: ДМК Пресс, 2020 — 582 с. – 208 с.

Приложение А Скрипт создания таблиц БД

```
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;

CREATE TABLE iF NOT EXISTS nutrition(
id SERIAL PRIMARY KEY,
essential_trace_elements VARCHAR(30) NOT NULL,
favorite_places VARCHAR(20) NOT NULL
);

CREATE TABLE iF NOT EXISTS body(
id SERIAL PRIMARY KEY,
color VARCHAR(10) NOT NULL,
ornament VARCHAR(20) NOT NULL
);

CREATE TABLE iF NOT EXISTS tendrils(
id SERIAL PRIMARY KEY,
len INT NOT NULL,
color VARCHAR(10) NOT NULL,
features VARCHAR(20) NOT NULL
);

CREATE TABLE iF NOT EXISTS paws(
id SERIAL PRIMARY KEY,
len INT NOT NULL,
color VARCHAR(10) NOT NULL
);

CREATE TABLE iF NOT EXISTS wings(
id SERIAL PRIMARY KEY,
color VARCHAR(10) NOT NULL,
ornament VARCHAR(100) NOT NULL,
ornament_features VARCHAR(110) NOT NULL,
shape VARCHAR(10) NOT NULL,
wingspan INT NOT NULL
);

CREATE TABLE iF NOT EXISTS species_book(
id SERIAL PRIMARY KEY,
species_name VARCHAR(100) NOT NULL,
wings_id INT NOT NULL,
paws_id INT NOT NULL,
tendrils_id INT NOT NULL,
body_id INT NOT NULL,
nutrition_id INT NOT NULL,
life_expectancy INT NOT NULL,

CONSTRAINT fk_wings_id FOREIGN KEY (wings_id) REFERENCES wings (id)
ON UPDATE CASCADE ON DELETE NO
ACTION,
CONSTRAINT fk_paws_id FOREIGN KEY (paws_id) REFERENCES paws (id)
```

```
52 ON UPDATE CASCADE ON DELETE NO
53 ACTION,
54 CONSTRAINT fk_tendrils_id FOREIGN KEY (tendrils_id) REFERENCES tendrils (id)
55  ON UPDATE CASCADE ON DELETE NO
56 ACTION,
57 CONSTRAINT fk_body_id FOREIGN KEY (body_id) REFERENCES body (id)
58 ON UPDATE CASCADE ON DELETE NO
59 ACTION,
60 CONSTRAINT fk_nutrition_id FOREIGN KEY (nutrition_id)
61 REFERENCES nutrition (id) ON UPDATE CASCADE ON DELETE NO
62 ACTION
63 );
64
65 CREATE TABLE status_type(
66 id SERIAL PRIMARY KEY,
67 status_name VARCHAR(60) NOT NULL
68 );
69
70 CREATE TABLE iF NOT EXISTS status(
71 id SERIAL PRIMARY KEY,
72 status_type INT NOT NULL,
73 assignment_date VARCHAR(10) NOT NULL,
74 cancellation_date VARCHAR(10) NOT NULL,
75 species_id INT NOT NULL,
76
77 CONSTRAINT fk_species_id FOREIGN KEY (species_id)
78 REFERENCES species_book (id) ON UPDATE CASCADE ON DELETE NO
79 ACTION,
80 CONSTRAINT fk_status_type FOREIGN KEY (status_type)
81 REFERENCES status_type (id) ON UPDATE CASCADE ON DELETE NO
82 ACTION
83 );
84
85
86 CREATE TABLE iF NOT EXISTS user_(
87 id SERIAL PRIMARY KEY,
88 user_login VARCHAR(20) NOT NULL,
89 user_password VARCHAR(30) NOT NULL,
90 user_name VARCHAR(20) NOT NULL
91 );
92
93 CREATE TABLE iF NOT EXISTS butterfly(
94 id SERIAL PRIMARY KEY,
95 author_id INT NOT NULL,
96 species_id INT NOT NULL,
97 coordinates VARCHAR(20) NOT NULL,
98 data VARCHAR(10) NOT NULL,
99
100 CONSTRAINT fk_species_id FOREIGN KEY (species_id) REFERENCES
101 species_book (id) ON UPDATE CASCADE ON DELETE NO
102 ACTION,
103 CONSTRAINT fk_author_id FOREIGN KEY (author_id) REFERENCES user_
104  (id) ON UPDATE CASCADE ON DELETE NO
```

```
105 ACTION
106 );
107
108 CREATE TABLE iF NOT EXISTS photo(
109 id SERIAL PRIMARY KEY,
110 butterfly_id INT NOT NULL,
111 photo BYTEA NOT NULL,
112
113 CONSTRAINT fk_butterfly_id FOREIGN KEY (butterfly_id)
114 REFERENCES butterfly (id) ON UPDATE CASCADE ON DELETE NO
115 ACTION
116 );
117
118 CREATE TABLE iF NOT EXISTS video(
119 id SERIAL PRIMARY KEY,
120 butterfly_id INT NOT NULL,
121 video BYTEA NOT NULL,
122
123 CONSTRAINT fk_butterfly_id FOREIGN KEY (butterfly_id)
124 REFERENCES butterfly (id) ON UPDATE CASCADE ON DELETE NO
125 ACTION
126 );
127
128 CREATE INDEX idx_species_book_species_name ON species_book (species_name);
129
130 CREATE INDEX idx_status_species_id ON status (species_id);
131
132 CREATE INDEX idx_butterfly_species_id ON butterfly (species_id);
133 CREATE INDEX idx_butterfly_author_id ON butterfly (author_id);
134
135 CREATE INDEX idx_photo_butterfly_id ON photo (butterfly_id);
136 CREATE INDEX idx_video_butterfly_id ON video (butterfly_id);
137
138
```

Приложение В Скрипт заполнения таблиц БД

```
import psycopg2
import random
import string
from datetime import datetime, timedelta
from faker import Faker

conn = psycopg2.connect(
dbname="postgres",
user="postgres",
password="...",
host="localhost",
port="5432"
)

cur = conn.cursor()

num_of_nutrition = 20
num_of_body = 100
num_of_tendrils = 50
num_of_paws = 100
num_of_wings = 200
num_of_users = 100
num_of_status_type = 6

num_of_species_book = 10000

num_of_statuses = (1, 3)
num_of_butterflies = (5, 10)
num_of_photoes = (1, 5)
num_of_videos = (1, 2)

# nutrition
data_to_insert =
    ("Железо", "Лес"),
    ("Кальций", "Луг"),
    ("Медь", "Река"),
    ("Цинк", "Гора"),
    ("Магний", "Озеро"),
    ("Калий", "Пустыня"),
    ("Селен", "Долина"),
    ("Фосфор", "Океан"),
    ("Натрий", "Поле"),
    ("Хром", "Каньон"),
    ("Марганец", "Пруд"),
    ("Кобальт", "Саванна"),
    ("Никель", "Тундра"),
    ("Бор", "Болото"),
    ("Фтор", "Плато"),
    ("Йод", "Ледник"),
    ("Ванадий", "Вулкан"),
    ("Молибден", "Равнина"),
```

```
    ("Литий", "Степь"),
    ("Кремний", "Холм")
]

sql_query = "INSERT INTO nutrition (essential_trace_elements,
 favorite_places) VALUES (%s, %s)"

cur.executemany(sql_query, data_to_insert)
print("nutrition finished")

colors = ["Red", "Blue", "Green", "Yellow", "Orange", "Purple", "Pink",
 "Brown", "Black", "White"]
ornaments = ["Stripes", "Polka Dots", "Checks", "Floral", "Plaid",
 "Geometric", "Abstract", "Tribal", "Animal Print", "Solid"]

# body
data_to_insert = []
existing_combinations = set()
while len(data_to_insert) < num_of_body:
color = random.choice(colors)
ornament = random.choice(ornaments)
combination = (color, ornament)

if combination not in existing_combinations:
data_to_insert.append(combination)
existing_combinations.add(combination)

sql_query = "INSERT INTO body (color, ornament) VALUES (%s, %s)"
cur.executemany(sql_query, data_to_insert)

print("body finished")

lengths = [5, 10, 15, 20, 25, 30]
colors = ["Red", "Brown", "Black", "White"]
features = ["Long", "Short", "White tips", "Straight", "Wavy",
 "Stripes", "Thick", "Thin", "Knotted", "Smooth"]

#tendrils
data_to_insert = []
existing_combinations = set()
while len(data_to_insert) < num_of_tendrils:
length = random.choice(lengths)
color = random.choice(colors)
feature = random.choice(features)
combination = (length, color, feature)

if combination not in existing_combinations:
data_to_insert.append(combination)
existing_combinations.add(combination)

sql_query = "INSERT INTO tendrils (len, color, features)
 VALUES (%s, %s, %s)"
cur.executemany(sql_query, data_to_insert)
```

```
105
106 print("tendrils finished")
107
108 lengths = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
109 colors = ["Red", "Blue", "Green", "Yellow", "Orange", "Purple",
110 "Pink", "Brown", "Black", "White"]
111
112 #paws
113 data_to_insert = []
114 existing_combinations = set()
115 while len(data_to_insert) < num_of_paws:
116 length = random.choice(lengths)
117 color = random.choice(colors)
118 combination = (length, color)
119
120 if combination not in existing_combinations:
121 data_to_insert.append(combination)
122 existing_combinations.add(combination)
123
124 sql_query = "INSERT INTO paws (len, color) VALUES (%s, %s)"
125 cur.executemany(sql_query, data_to_insert)
126
127 print("paws finished")
128
129 colors = ["Red", "Blue", "Green", "Yellow", "Orange", "Purple",
130 "Pink", "Brown", "Black", "White"]
131 ornaments = ["Stripes", "Polka Dots", "Checks", "Floral", "Plaid",
132  "Geometric", "Abstract", "Tribal", "Animal Print", "Solid"]
133 ornament_features = ["Shiny", "Matte", "Textured", "Smooth", "Reflective",
134  "Glittering", "Translucent", "Opaque", "Metallic", "Sparkling"]
135 shapes = ["Round", "Oval", "Triangular", "Diamond", "Spherical", "Irregular"]
136
137 data_to_insert = []
138 existing_combinations = set()
139 while len(data_to_insert) < num_of_wings:
140 color = random.choice(colors)
141 ornament = random.choice(ornaments)
142 ornament_feature = random.choice(ornament_features)
143 shape = random.choice(shapes)
144 wingspan = random.randint(1, 100)
145 combination = (color, ornament, ornament_feature, shape, wingspan)
146
147 if combination not in existing_combinations:
148 data_to_insert.append(combination)
149 existing_combinations.add(combination)
150
151 sql_query = "INSERT INTO wings (color, ornament, ornament_features, shape,
152  wingspan) VALUES (%s, %s, %s, %s, %s)"
153 cur.executemany(sql_query, data_to_insert)
154
155 print("wings finished")
156
157 #status type
```

```
statuses = [
"likely extinct",
"endangered",
"declining in population and/or distribution",
"rare",
"undetermined status",
"recoverable and recovering"
]

data_to_insert = [(status,) for status in statuses]

sql_query = "INSERT INTO status_type (status_name) VALUES (%s)"
cur.executemany(sql_query, data_to_insert)

print("status type finished")

def generate_random_string(length):
return ''.join(random.choices(string.ascii_letters + string.digits,
 k=length))

# user_
fake = Faker()
additional_random_names = [fake.first_name() for _ in range(num_of_users+1)]

data_to_insert = []
for i in range(num_of_users):
login = additional_random_names[i]
password = generate_random_string(random.randint(20, 30))
name = login + generate_random_string(random.randint(1, 5))
data_to_insert.append((login, password, name))

print("user finished")

data_to_insert = []
unique_combinations = set()

# species_book
while len(data_to_insert) < num_of_species_book:
wings_id = random.randint(1, 200)
paws_id = random.randint(1, 100)
tendrils_id = random.randint(1, 50)
body_id = random.randint(1, 100)
nutrition_id = random.randint(1, 20)
life_expectancy = random.randint(10, 150)
species_name = f"name {len(data_to_insert) + 1}"
combination = (wings_id, paws_id, tendrils_id, body_id, nutrition_id)

if combination not in unique_combinations:
unique_combinations.add(combination)
data_to_insert.append((species_name, wings_id, paws_id, tendrils_id,
 body_id, nutrition_id, life_expectancy))

sql_query = "INSERT INTO species_book (species_name, wings_id, paws_id,
```

```
 tendrils_id, body_id, nutrition_id, life_expectancy)\
VALUES (%s, %s, %s, %s, %s, %s, %s)"
cur.executemany(sql_query, data_to_insert)

print("species book finished")

def random_date(start_date, end_date):
delta = end_date - start_date
random_days = random.randint(0, delta.days)
return start_date + timedelta(days=random_days)

data_to_insert = []  # status

for i in range(1, num_of_species_book + 1):
num_of_statuses_for_species = random.randint(num_of_statuses[0],
 num_of_statuses[1])
species_id = i
for _ in range(num_of_statuses_for_species):
status_type = random.randint(1, 6)

start_date = datetime(1950, 1, 1)
end_date = datetime(2024, 12, 31)

assignment_date = random_date(start_date, end_date)
if random.randint(-1, 50) > 0:
cancellation_date = random_date(assignment_date, end_date)
cancellation_date_str = cancellation_date.strftime("%d-%m-%Y")
else:
cancellation_date_str = "-"

assignment_date_str = assignment_date.strftime("%d-%m-%Y")

data_to_insert.append((status_type, assignment_date_str,
 cancellation_date_str, species_id))

sql_query = "INSERT INTO status (status_type, assignment_date,
 cancellation_date, species_id) VALUES (%s, %s, %s, %s)"

cur.executemany(sql_query, data_to_insert)

print("status finished")

#butterfly
def generate_random_coordinates():
degrees_lat = random.uniform(-90, 90)
degrees_lon = random.uniform(-180, 180)

direction_lat = "N" if degrees_lat >= 0 else "S"
direction_lon = "E" if degrees_lon >= 0 else "W"

coordinates = "{:.4f}°{} {:.4f}°{}".format(abs(degrees_lat), direction_lat,
 abs(degrees_lon), direction_lon)

```

```
return coordinates

data_to_insert_butterfly = []
num_of_butterflies_in_tab = 0
for i in range(1, num_of_species_book + 1):
num_of_butterflies_for_species = random.randint(num_of_butterflies[0],
 num_of_butterflies[1])
species_id = i
for _ in range(num_of_butterflies_for_species):
author_id = random.randint(1, num_of_users)
coordinates = generate_random_coordinates()
data = random_date(datetime(2008, 1, 1), datetime(2024, 12, 31))
data = data.strftime("%d-%m-%Y")

data_to_insert_butterfly.append((author_id, species_id, coordinates, data))
num_of_butterflies_in_tab += 1

sql_query_butterfly = "INSERT INTO butterfly (author_id, species_id, coordinates,
 data) VALUES (%s, %s, %s, %s)"
cur.executemany(sql_query_butterfly, data_to_insert_butterfly)

print("butterfly finished")

#photo
data_to_insert = []
photo_num = 1
for i in range(1, num_of_butterflies_in_tab + 1):
num_of_photoes_for_butterfly = random.randint(num_of_photoes[0],
 num_of_photoes[1])
butterfly_id = i
for j in range(num_of_photoes_for_butterfly):
photo = f"photo {photo_num}"
data_to_insert.append((butterfly_id, photo))
photo_num+=1

sql_query_butterfly = "INSERT INTO photo (butterfly_id, photo)
VALUES (%s, %s)"

cur.executemany(sql_query_butterfly, data_to_insert)

print("photo finished")

#video
data_to_insert = []
video_num = 1
for i in range(1, num_of_butterflies_in_tab + 1):
num_of_videos_for_butterfly = random.randint(num_of_videos[0],
 num_of_videos[1])
butterfly_id = i
for j in range(num_of_videos_for_butterfly):
video = f"video {video_num}"
data_to_insert.append((butterfly_id, video))
video_num+=1
```

```
317
318 sql_query_butterfly = "INSERT INTO video (butterfly_id, video)
319 VALUES (%s, %s)"
320
321 cur.executemany(sql_query_butterfly, data_to_insert)
322
323 print("video finished")
324
325 conn.commit()
326 cur.close()
327 conn.close()
```