

Содержание:

<b>Условия задачи</b>	<b>3</b>
<b>Разработка структуры базы данных (серверная часть)</b>	<b>5</b>
Графическая схема базы данных	5
Таблицы и их связи	6
Создание таблиц	11
Ограничения целостности	16
<b>Написание SQL запросов к спроектированной базе данных согласно заданию.</b>	<b>17</b>
Запросы	17
<b>Реализация триггеров и хранимых процедур (PL / SQL).</b>	<b>23</b>
Триггеры	23
Процедуры	25
<b>Разработка приложения клиента (формы ввода, редактирования и поиска данных по запросам)</b>	<b>27</b>

## Условия задачи

Для выбранного проекта студент разрабатывает структуру базы данных и реализует приложение в архитектуре клиент-сервер, выполняющее операции внесения данных в базу данных, редактирование данных и запросы, указанные в проекте. Клиентская часть реализуется на языке программирования высокого уровня. В описании проекта дана обобщенная пользовательская спецификация приложения. Спецификация не предполагает оптимального определения структур данных, но задает полный перечень хранимой в базе данных информации и выполняемых программой функций. Таблицы должны заполняться адекватными значениями, примерно по 7 объектов в каждой.

Разработка проекта предполагает выполнение следующих этапов:

1. Разработка структуры базы данных (серверная часть)
  - 1.1 Проектирование инфологической модели задачи. Определение сущностей, атрибутов сущностей, идентифицирующих атрибутов, связей между сущностями. При проектировании должны учитываться требования гибкости структур для выполнения перечисленных функций и не избыточного хранения данных.
  - 1.2 Проектирование схемы базы данных: описание схем таблиц, типов (доменов) атрибутов, определение ограничений целостности. Написание SQL скриптов по созданию таблиц БД.
  - 1.3 Создание и заполнение разработанной БД на стороне сервера.
2. Написание SQL запросов к спроектированной базе данных согласно заданию.
3. Реализация триггеров и хранимых процедур (PL / SQL).
4. Разработка приложения клиента (формы ввода, редактирования и поиска данных по запросам)

### **Проект: “Информационная система зоопарка”.**

Служащих зоопарка можно подразделить на несколько категорий: ветеринары, уборщики, дрессировщики, строители-ремонтники, работники администрации. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. За каждым животным ухаживает определенный круг служащих, причем только ветеринарам, уборщикам и дрессировщикам разрешен доступ в клетки к животным. В зоопарке обитают животные различных климатических зон, поэтому часть животных на зиму необходимо переводить в отапливаемые помещения. Животных можно подразделить на хищников и травоядных. При расселении животных по клеткам необходимо учитывать не только потребности

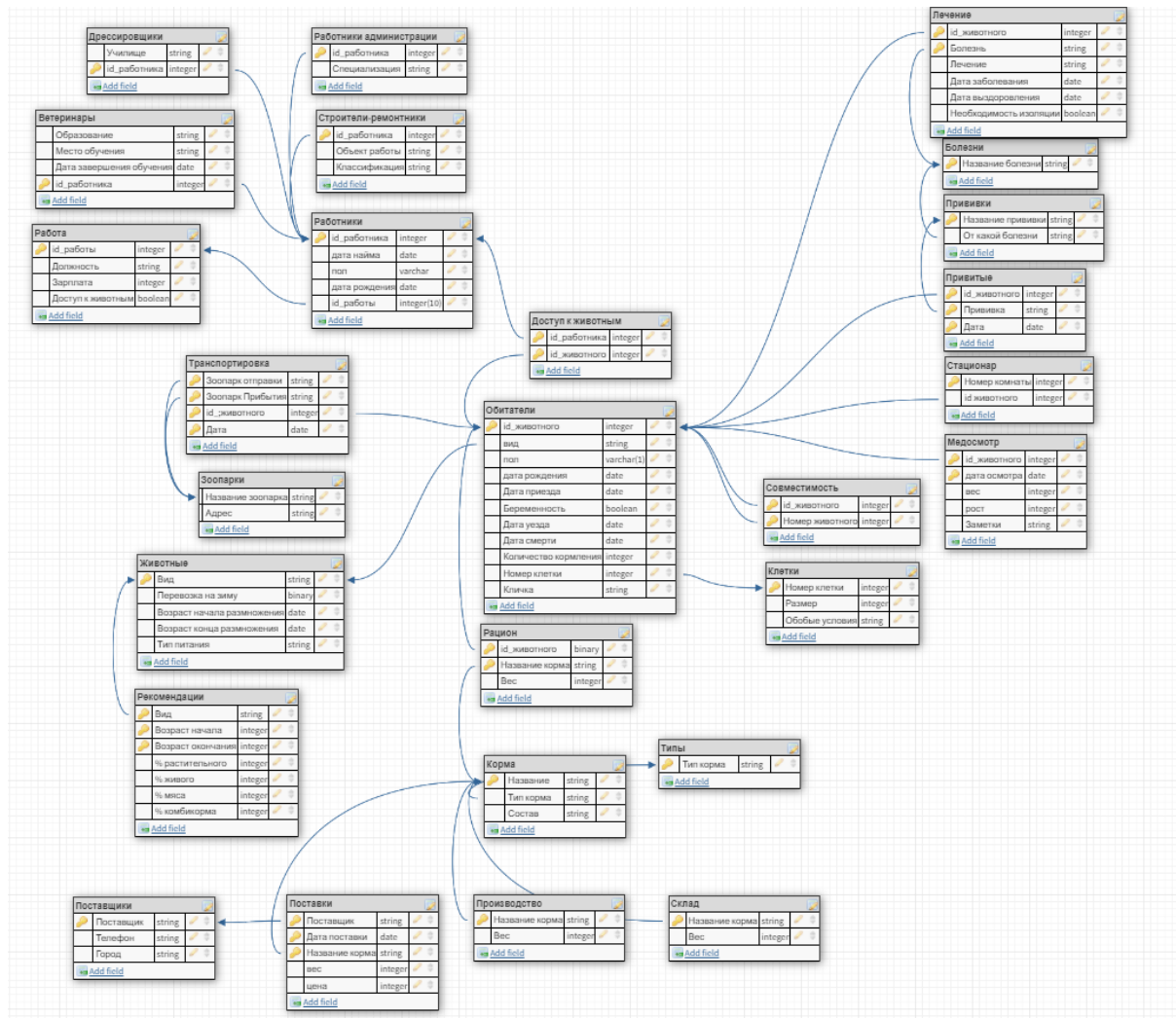
данного вида, но и их совместимость с животными в соседних клетках (нельзя рядом селить, например, волков и их добычу - различных копытных).

Для кормления животных необходимы различные типы кормов: растительный, живой, мясо и различные комбикорма. Растительный корм это фрукты и овощи, зерно и сено. Живой корм - мыши, птицы, корм для рыб. Для каждого вида животных рассчитывается свой рацион, который в свою очередь варьируется в зависимости от возраста, физического состояния животного и сезона. Таким образом у каждого животного в зоопарке имеется меню на каждый день, в котором указывается количество и время кормлений в день, количество и вид пищи (обезьянам необходимы фрукты и овощи, мелким хищникам - хорькам, ласкам, совам, некоторым кошачьим, змеям - надо давать мышей). У зоопарка имеются поставщики кормов для животных. Каждый поставщик специализируется на каких-то конкретных видах кормов. Часть кормов зоопарк может производить сам: запастись сеном, разводить мышей и т.д. Ветеринары должны проводить медосмотры, следить за весом, ростом, развитием животного, ставить своевременно прививки и заносить все эти данные в карточку, которая заводится на каждую особь при ее появлении в зоопарке. Больным животным назначается лечение и при необходимости их можно изолировать в стационаре.

При определенных условиях (наличие пары особей, подходящих по возрасту, физическому состоянию) можно ожидать появления потомства. Потомство от данной пары животных при достижении ими положенного возраста можно либо оставить в зоопарке, создав для них подходящие условия содержания, либо обменяться с другими зоопарками или просто раздать в другие зоопарки - по решению администрации.

# Разработка структуры базы данных (серверная часть)

## Графическая схема базы данных



<https://dbdesigner.page.link/nnii6vjGpsESzaVi7>

## Таблицы и их связи

### Работники

Работники			
	id_работника	integer	
	дата найма	date	
	пол	varchar	
	дата рождения	date	
	id_работы	integer(10)	

Таблица “Работники” содержит все личные данные о работниках, такие как пол, дата найма, дата рождения и уникальный номер. В таблице также хранится уникальный номер работы, который связан с таблицей “Работа”.

Работа			
	id_работы	integer	
	Должность	string	
	Зарплата	integer	
	Доступ к животным	boolean	

Таблица “Работа” хранит информацию о всех работах, а именно о должности и зарплате. Также в ней хранится информация возможности доступа к животным.

Доступ к животным			
	id_работника	integer	
	id_животного	integer	

Таблица “Доступ к животным” связывает уникальный номер работника и животного к которому он имеет доступ. Поля связаны с таблицами “Работники” и “Обитатели”.

Таблицы “Ветеринары”, “Дрессировщики”, “Строители-ремонтники”, “Работники администрации” хранит информацию о определенных категориях работников и содержит в себе уникальные-атрибуты характеристики. Каждая из таблиц


Ветеринары			
	Образование	string	
	Место обучения	string	
	Дата завершения обучения	date	
	id_работника	integer	

содержит в себе уникальный номер работника из таблицы “Работники”.


Для ветеринаров хранится информация о их образовании, месте обучения и дате его завершения.

Дрессировщики		
	Училище	string
	id_работника	integer

Для дрессировщиков - информация об училище.

Строители-ремонтники		
	id_работника	integer
	Объект работы	string
	Классификация	string

Для строителей-ремонтников - классификация и объект на котором они работают.

Работники администрации		
	id_работника	integer
	Специализация	string

Для работников администрации - специализация.

## Расселение

Обитатели		
	id_животного	integer
	вид	string
	пол	varchar(1)
	дата рождения	date
	Дата приезда	date
	Беременность	boolean
	Дата уезда	date
	Дата смерти	date
	Количество кормления	integer
	Номер клетки	integer
	Кличка	string

В зоопарке обитают животные, основная информация для которых хранится в таблице “Обитатели”, где содержится уникальный номер животного, кличка, пол, количество кормления, дата рождения, дата приезда, дата уезда и дата смерти, если такие имеются. Также хранится информация о беременности и вид связанный с таблицей “Животные”. Животные в зоопарке живут в клетках, поэтому в таблице “Обитатели” имеется поле хранящее в себе номер клетки и связанное с таблицей “Клетки”.


Клетки		
	Номер клетки	integer
	Размер	integer
	Обоные условия	string

Таблица “Клетки” содержит в себе номер клетки, её размер и информацию об особых условиях.

Совместимость		
	id_животного	integer
	Номер животного	integer

При расселении животных нужно учитывать совместимость, для этого в таблице “Совместимость” хранится пара животных связанных с таблицей “Обитатели” уникальными номерами.


Животные		
	Вид	string
	Перевозка на зиму	binary
	Возраст начала размножения	date
	Возраст конца размножения	date
	Тип питания	string



Таблица “Животные” содержит в себе информацию о различных видах, нужде в теплом помещении зимой, период размножения и тип питания.

## Лечение


Медосмотр		
	id_животного	integer
	дата осмотра	date
	вес	integer
	рост	integer
	Заметки	string

Ветеринары должны проводить медосмотры и записывать информацию о животном из таблицы “Обитатели”, дате

осмотра, весе и росте в таблицу “Медосмотр”, также имеется возможность оставлять заметки.

Лечение		
	id_животного	integer
	Болезнь	string
	Лечение	string
	Дата заболевания	date
	Дата выздоровления	date
	Необходимость изоляции	boolean

Если животное заболело необходимо внести данные в таблицу “Лечение”, где для животного указывается болезнь из таблицы “Болезни” в которой

Болезни		
	Название болезни	string

содержатся названия болезней, лечение, дата заболевания и дата выздоровления,

необходимость в изоляции.

Стационар		
	Номер комнаты	integer
	id животного	integer




Изолировать животное можно в стационаре, при изоляции данные вносятся в таблицу “Стационар”.

Данная таблица содержит номер комнаты и номер животного из таблицы “Обитатели”.

Прививки		
	Название прививки	string
	От какой болезни	string

При необходимости животных можно привить, информация о прививках содержится в таблице “Прививки”, в ней содержится название прививки и название болезни от которой производится прививка,

это поля связано с полем “название болезни” из таблицы “Болезни”.

Привитые		
	id_животного	integer
	Прививка	string
	Дата	date

В таблицы “Привитые” содержится информация о прививки животного из таблицы “Обитатели” и дате прививки. Поле “Прививка” связано с таблицей “Прививки”.

## Транспортировка

Транспортировка		
	Зоопарк отправки	string
	Зоопарк Прибытия	string
	id_животного	integer
	Дата	date

Животными можно обмениваться и таблица

“Транспортировка” содержит всю необходимую информацию, а именно: зоопарк отправки и зоопарк прибытия,

Зоопарки	
Название зоопарка	string
Адрес	string

оба поля связаны с таблицей “Зоопарки” в которой хранится название зоопарка и его адрес, животное из таблицы “Обитатели”, которое было перевезено, и дата транспортировки.

## Кормление

Рекомендации		
🔑	Вид	string
🔑	Возраст начала	integer
🔑	Возраст окончания	integer
	% растительного	integer
	% живого	integer
	% мяса	integer
	% комбикорма	integer

Для каждого вида определенного возраста существуют рекомендации по кормлению, хранящиеся в таблице “Рекомендации”. Поле “вид” связано с одноименным полем из “Животные”, так же имеются поля возраста начала и окончания кормления для данной рекомендации, а также % от каждого типа корма.

Корма		
🔑	Название	string
	Тип корма	string
	Состав	string

Таблица “Корма” содержит в себе тип корма, название и

Типы		
🔑	Тип корма	string

состав.

тип корма связан с таблицей типы

Рацион		
🔑	id_животного	binary
🔑	Название корма	string
	Вес	integer

Для каждого животного определен свой рацион в таблице “Рацион” хранящий id животного из таблицы “Обитатели”, название корма из “Корма” и количество корма необходимое животному в один прием пищи.

## Производство и поставка корма

Производство		
🔑	Название корма	string
	Вес	integer

Часть кормов производит зоопарк, а часть предоставляют. Таблица “Производство” хранит в себе название корма из “Корма” и вес производимый в год.


Поставщики		
🔑	Поставщик	string
	Телефон	string
	Город	string

Таблица “Поставщики” содержит в себе информацию о поставщике, его телефон и город

Поставки		
🔑	Поставщик	string
🔑	Дата поставки	date
🔑	Название корма	string
	вес	integer
	цена	integer

Все поставки хранятся в таблице “Поставки”, для каждой поставки хранится поставщик, дата поставки, название корма, поставленный вес и цена которую за эту поставку заплатили. Поле “Поставщик” связано с таблицей “Поставщики”.



Склад		
	Название корма	string
	Вес	integer

Корм отправляется на склад, а данные заносятся в таблицу "Склад", где указывается название корма и вес.

## Создание таблиц

```
CREATE TABLE Работа(  
id_работы INT PRIMARY KEY,  
Должность VARCHAR2(100) NOT NULL,  
Зарплата INT NOT NULL CHECK(Зарплата > 0),  
Доступ CHAR(1) NOT NULL CHECK(Доступ in ('N', 'Y'))  
);
```

```
CREATE TABLE Работники(  
id_работника INT PRIMARY KEY,  
дата_найма DATE NOT NULL,  
пол CHAR(1) NOT NULL CHECK(пол in ('M', 'W')),  
дата_рождения DATE NOT NULL,  
id_работы INT NOT NULL REFERENCES Работа(id_работы)  
);
```

```
CREATE TABLE Дрессировщики(  
id_работника INT PRIMARY KEY REFERENCES Работники(id_работника),  
училище VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Администрация(  
id_работника INT PRIMARY KEY REFERENCES Работники(id_работника),  
специализация VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Строители(  
id_работника INT PRIMARY KEY REFERENCES Работники(id_работника),  
объект VARCHAR(100),  
классификация VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Ветеринары(  
id_работника INT PRIMARY KEY REFERENCES Работники(id_работника),  
место_обучения VARCHAR(100) NOT NULL,  
завер_обуч DATE NOT NULL,  
образование VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Животные(  
вид VARCHAR(100) PRIMARY KEY,  
перевозка CHAR(1) NOT NULL CHECK(перевозка in ('Y', 'N')),
```

```
нач_размножения INT NOT NULL CHECK(нач_размножения > 0),
кон_размножения INT NOT NULL CHECK(кон_размножения > 0),
тип_питания VARCHAR(100) NOT NULL
);
CREATE TABLE Рекомендации(
индекс INT PRIMARY KEY,
вид VARCHAR(100) NOT NULL REFERENCES Животные(вид),
возраст_начала INT NOT NULL,
возраст_конца INT NOT NULL,
растительного INT NOT NULL CHECK ( растительного >= 0 AND растительного
<= 100 ),
живого INT NOT NULL CHECK (живого >= 0 AND живого <= 100 ),
мяса INT NOT NULL CHECK (мяса >= 0 AND мяса <= 100),
комбикорма INT NOT NULL CHECK (комбикорма >= 0 AND комбикорма <= 100 ),
CONSTRAINT key UNIQUE (вид, возраст_начала, возраст_конца)
);
```

```
CREATE TABLE Клетки(
номер_клетки INT PRIMARY KEY,
размер INT NOT NULL CHECK (размер > 0),
особые_условия VARCHAR(100)
);
```

```
CREATE TABLE Обитатели(
id_животного INT PRIMARY KEY,
вид VARCHAR(100) NOT NULL REFERENCES Животные(вид),
пол CHAR(1) NOT NULL CHECK ( пол in ('М','Ж')),
дата_рождения DATE NOT NULL,
дата_приезда DATE,
беременность CHAR(1) CHECK ( беременность in ('Y','N')),
дата_уезда DATE,
дата_смерти DATE,
кормления INT NOT NULL CHECK (кормления > 0),
номер_клетки INT REFERENCES Клетки(номер_клетки)
);
```

```
CREATE TABLE Совместимость(
индекс INT PRIMARY KEY,
id_животного INT REFERENCES Обитатели(id_животного),
номер_животного INT REFERENCES Обитатели(id_животного)
);
```

```
CREATE TABLE Медосмотр(
индекс INT PRIMARY KEY,
```

```
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),
дата_осмотра DATE NOT NULL,
вес INT NOT NULL,
рост INT NOT NULL,
заметки VARCHAR(100),
CONSTRAINT id_date UNIQUE(id_животного, дата_осмотра)
);
```

```
CREATE TABLE Стационар(
номер_комнаты INT PRIMARY KEY,
id_животного INT REFERENCES Обитатели(id_животного)
);
```

```
CREATE TABLE Болезни(
болезнь VARCHAR(100) PRIMARY KEY
);
```

```
CREATE TABLE Прививки(
прививка VARCHAR(100) PRIMARY KEY,
болезнь VARCHAR(100) NOT NULL REFERENCES Болезни(болезнь)
);
```

```
CREATE TABLE Привитые(
индекс INT PRIMARY KEY,
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),
прививка VARCHAR(100) NOT NULL REFERENCES Прививки(прививка),
дата DATE NOT NULL,
CONSTRAINT прививки UNIQUE(id_животного, прививка, дата)
);
```

```
CREATE TABLE Лечение(
индекс INT PRIMARY KEY,
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),
болезнь VARCHAR(100) NOT NULL REFERENCES Болезни(болезнь),
лечение VARCHAR(100) NOT NULL,
дата_начала DATE NOT NULL,
дата_конца DATE,
изоляция CHAR(1) DEFAULT('N') CHECK(изоляция in ('Y', 'N'))
);
```

```
CREATE TABLE Зоопарки(
название VARCHAR(100) PRIMARY KEY,
адрес VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Транспортировка(  
индекс INT PRIMARY KEY,  
з_отправки VARCHAR(100) NOT NULL,  
з_прибытия VARCHAR(100) NOT NULL,  
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),  
дата DATE NOT NULL,  
CONSTRAINT транспортировка UNIQUE(з_отправки, з_прибытия, id_животного,  
дата)  
);
```

```
CREATE TABLE Типы(  
тип_корма VARCHAR(100) PRIMARY KEY  
);
```

```
CREATE TABLE Корма(  
название VARCHAR(100) PRIMARY KEY,  
тип_корма VARCHAR(100) NOT NULL REFERENCES Типы(тип_корма),  
состав VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Рацион(  
индекс INT PRIMARY KEY,  
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),  
название VARCHAR(100) NOT NULL REFERENCES Корма(название),  
вес INT NOT NULL,  
CONSTRAINT рацион UNIQUE(id_животного, название)  
);
```

```
CREATE TABLE Поставщики(  
поставщик VARCHAR(100) PRIMARY KEY,  
телефон VARCHAR(100) NOT NULL,  
город VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Поставки(  
индекс INT PRIMARY KEY,  
поставщик VARCHAR(100) NOT NULL REFERENCES Поставщики(поставщик),  
дата_поставки DATE NOT NULL,  
название_корма VARCHAR(100) NOT NULL REFERENCES Корма(название),  
вес INT NOT NULL CHECK( вес > 0),  
цена INT NOT NULL CHECK(цена > 0),  
CONSTRAINT поставки UNIQUE(поставщик, дата_поставки, название_корма)  
);
```

```
CREATE TABLE Производство(  
название_корма VARCHAR(100) PRIMARY KEY REFERENCES  
Корма(название),  
вес INT NOT NULL  
);
```

```
CREATE TABLE Склад(  
название_корма VARCHAR(100) PRIMARY KEY REFERENCES  
Корма(название),  
вес INT NOT NULL  
);
```

```
CREATE TABLE Доступ(  
индекс INT PRIMARY KEY,  
id_работника INT NOT NULL REFERENCES Работники(id_работника),  
id_животного INT NOT NULL REFERENCES Обитатели(id_животного),  
CONSTRAINT доступ UNIQUE(id_работника, id_животного)  
);  
ALTER TABLE Работники ADD имя VARCHAR(100) NOT NULL
```

```
ALTER TABLE Работники ADD фамилия VARCHAR(100) NOT NULL
```

```
ALTER TABLE Обитатели DROP COLUMN пол;
```

```
ALTER TABLE Обитатели ADD пол CHAR(1) CHECK (пол in ('W','M'));  
ALTER TABLE Обитатели ADD кличка VARCHAR(100) NOT NULL
```

## Ограничения целостности

- для работника данные в поле id\_работы из таблицы "Работник" связанного с таблицей "Работа" должны совпадать с занесением этого работника в одну из таблиц "Дрессировщики", "Администрация", "Ветеринары", "Строители"
- id\_работника не должен повторяться в таблицах "Дрессировщики", "Администрация", "Ветеринары", "Строители"
- Животные живущие в одной клетки, должны быть совместимы, то есть пара животных должна присутствовать в таблице "Совместимость" // сделать для этого триггер

## Написание SQL запросов к спроектированной базе данных согласно заданию.

### Запросы

**1. Получить список и общее число служащих зоопарка, либо служащих данной категории полностью, по продолжительности работы в зоопарке, по половому признаку, возрасту, размеру заработной платы.**

1.1     SELECT имя, фамилия, дата\_найма, пол, дата\_рождения  
          COUNT (\*) OVER() AS Число\_служащих  
          FROM Работники  
          ORDER BY дата\_найма, пол, дата\_рождения DESC  
          WHERE (дата\_рождения < '06.06.2000') AND (пол in 'W') AND (id\_работы  
                  = 3)

1.2     SELECT имя, фамилия, дата\_найма, пол, дата\_рождения, Зарплата  
          FROM Работники  
          JOIN Работа ON Работники.id\_работы = Работа.id\_работы  
          ORDER BY Зарплата

1.3     SELECT имя, фамилия  
          FROM Работники  
          JOIN Администрация ON Работники.id\_работника =  
          Администрация.id\_работника

**3. Получить список и общее число служащих зоопарков, имеющих доступ к указанному виду животных либо к конкретной особи.**

```
SELECT имя, фамилия, вид
FROM Работники
JOIN Доступ ON Работники.id_работника = Доступ.id_работника
JOIN Обитатели ON Доступ.id_животного = Обитатели.id_животного
WHERE (вид = 'Панда китайская') AND (кликка = 'Пушок')
```

**5. Получить перечень и общее число нуждающихся в теплом помещении на зиму, полностью животных только указанного вида или указанного возраста.**

```
SELECT кличка, Обитатели.вид,
COUNT (*) OVER() AS общее_число
```



```

FROM Обитатели
JOIN Животные ON Обитатели.вид = Животные.вид
WHERE (перевозка in 'Y') AND (Животные.вид in 'Панда китайская') AND
(дата_уезда is NULL) AND (дата_смерти is NULL) AND (дата_рождения <
SYSDATE - interval '8' month)

```

**6. Получить перечень и общее число животных, которым поставлена указанная прививка, либо переболевших некоторой болезнью, по длительности пребывания в зоопарке, половому признаку, возрасту.**

```

6.1  SELECT кличка, ПОЛ, ДАТА_РОЖДЕНИЯ,
COUNT (*) OVER() AS общее_число
FROM Обитатели
WHERE (id_животного in( SELECT id_животного
FROM Привитые
WHERE (прививка = ?) AND (дата_уезда is NULL) AND
(дата_смерти is NULL)))

```

```

6.2  SELECT кличка, ПОЛ, ДАТА_РОЖДЕНИЯ,
COUNT (*) OVER() AS общее_число
FROM Обитатели
WHERE (id_животного in( SELECT id_животного
FROM ЛЕЧЕНИЕ
WHERE (болезнь = ?) AND (дата_уезда is NULL) AND
(дата_смерти is NULL)))

```

**4. Получить перечень и общее число всех животных в зоопарке либо животных указанного вида, живущих в указанной клетке, по половому признаку, возрасту, весу, росту.**

```

SELECT кличка, дата_рождения, вес, рост, пол,
COUNT (*) OVER() AS общее_число
FROM Обитатели
JOIN (SELECT id_животного, вес, рост
      FROM (SELECT id_животного, дата_осмотра, вес, рост,
ROW_NUMBER() over (partition by id_животного order by
дата_осмотра desc) as num
      FROM медосмотр)
WHERE num = 1)med on med.id_животного = Обитатели.id_животного
WHERE номер_клетки = 1
ORDER BY пол, вес, рост, дата_рождения

```

**7. Получить перечень всех животных, совместимых с указанным видом или тех, которые нуждаются в теплом помещении.**

```
SELECT кличка
FROM Обитатели
JOIN Животные ON Обитатели.вид = Животные.вид
WHERE (id_животного in (SELECT Совместимость.id_животного
                        FROM Совместимость
                        JOIN Обитатели ON Обитатели.id_животного =
                        Совместимость.номер_животного
                        WHERE вид in 'Выдра'))AND (перевозка in 'Y')
```

**8. Получить перечень и общее число поставщиков кормов полностью, либо поставляющих только определенный корм, поставивших в указанный период, по количеству поставляемого корма, цене, датам поставок.**

8.1

```
SELECT поставщик,
COUNT(*) over () as общее_число
FROM Поставщики
WHERE поставщик in(SELECT поставщик
                    FROM Поставки
                    WHERE (название_корма = 'K1') AND (дата_поставки BETWEEN
                    '22.04.2022' AND SYSDATE) AND (цена > 1000))
```

8.2

```
SELECT поставщик, kg as "ВСЕГО ПОСТАВИЛ", manny as
"ЦЕНА", ДАТА_ПОСТАВКИ,
COUNT(*) over () as общее_число
FROM(SELECT поставщик, вес, название_корма,
дата_поставки,
ROW_NUMBER() over (partition by поставщик, название_корма
order by дата_поставки desc) as num,
SUM(вес) over(partition by поставщик, название_корма) as
kg,
SUM(цена) over (partition by поставщик, название_корма)as
manny
FROM поставки)
WHERE (num = 1) AND (название_корма = ?)
```

8.3

```
SELECT поставщик, kg as "ВСЕГО ПОСТАВИЛ", manny as
"ЦЕНА", ДАТА_ПОСТАВКИ,
COUNT(*) over () as общее_число
```

```

FROM(SELECT поставщик, вес, название_корма,
дата_поставки,
ROW_NUMBER() over (partition by поставщик order by
дата_поставки desc) as num,
SUM(вес) over(partition by поставщик) as kg,
SUM(цена) over (partition by поставщик) as manny
FROM поставки)
WHERE num = 1

```

**9. Получить перечень и объем кормов, производимых зоопарком полностью, либо только тех кормов, в поставках которых зоопарк не нуждается (обеспечивает себя сам).**

9.1 SELECT название\_корма, вес,  
COUNT(\*) over () as общее\_число  
FROM Производство

9.2 SELECT название, вес  
FROM(SELECT название, кормления \* вес \* 30 as в\_месяц, вес  
FROM Рацион  
JOIN Обитатели on Обитатели.id\_животного = Рацион.id\_животного  
JOIN Производство on Рацион.название =  
Производство.название\_корма)  
WHERE в\_месяц <= вес

**10.Получить перечень и общее число животных полностью, либо указанного вида, которым необходим определенный тип кормов, возрасте или круглый год.**

```

SELECT кличка, Обитатели.вид, Рекомендации.вид, возраст_начала,
возраст_конца,
COUNT(*) over () as общее_число
FROM Обитатели, Рекомендации
WHERE (Обитатели.вид =
Рекомендации.вид)AND((SYSDATE-дата_рождения)/365>=
возраст_начала)AND((SYSDATE-дата_рождения)/365<=
возраст_конца)AND(Обитатели.вид = 'Выдра')AND(дата_смерти is
NULL)AND(дата_уезда is NULL)AND ((растительного = 100)OR(живого =
100)OR(мяса = 100)OR(комбикорма = 100))

```

**11.Получить полную информацию (рост, вес, прививки, болезни, дата поступления в зоопарк или дата рождения, возраст) о всех животных, или о животных только данного вида, о конкретном животном, об особи, живущей в указанной клетке.**

```
11.1  SELECT кличка, дата_рождения, вес, рост, пол, дата_приезда,  
      ROUND((SYSDATE - дата_рождения)/365, 2)  
      FROM Обитатели  
      JOIN (SELECT id_животного, вес, рост  
            FROM (SELECT id_животного, дата_осмотра, вес, рост,  
                  ROW_NUMBER() over (partition by id_животного order by  
                  дата_осмотра desc) as num  
            FROM медосмотр)  
            WHERE num = 1)med on med.id_животного =  
            Обитатели.id_животного  
      WHERE (номер_клетки = 1)AND(вид = 'Панда китайская')AND(кличка =  
      'Пушок')
```

```
11.2  SELECT кличка, прививка  
      FROM Обитатели  
      JOIN Привитые On Привитые.id_животного = Обитатели.id_животного  
      WHERE (дата < SYSDATE)AND(номер_клетки = 1)AND(вид = 'Панда  
      китайская')AND(кличка = 'Пушок')
```

```
11.3  SELECT кличка, болезнь  
      FROM Обитатели  
      JOIN Лечение On Лечение.id_животного = Обитатели.id_животного  
      WHERE (номер_клетки = 1)AND(вид = 'Панда китайская')AND(кличка =  
      'Пушок')
```

**12.Получить перечень животных, от которых можно ожидать потомство в перспективе, в указанный период.**

```
SELECT кличка  
FROM Обитатели  
JOIN Животные ON Животные.вид = Обитатели.вид  
WHERE (пол in 'F') AND ((SYSDATE - дата_рождения)/365 >=  
нач_размножения) AND ((SYSDATE - дата_рождения)/365 <=  
кон_размножения) AND (дата_смерти is NULL) AND (дата_уезда is NULL)
```

**13.Получить перечень и общее число зоопарков, с которыми был произведен обмен животными в целом или животными только указанного вида.**

```
SELECT название,  
COUNT(*) over () as общее_число  
FROM Зоопарки  
WHERE (название in (  
    SELECT з_отправки  
    FROM Транспортировка  
    JOIN Обитатели ON Обитатели.id_животного =  
    Транспортировка.id_животного  
WHERE (вид in 'Кролик')) OR  
(название in (  
    SELECT з_прибытия  
    FROM Транспортировка  
    JOIN Обитатели ON Обитатели.id_животного =  
    Транспортировка.id_животного  
WHERE (вид in 'Кролик')))
```

## Реализация триггеров и хранимых процедур (PL / SQL).

### Триггеры

**Триггер "Animal\_moving" срабатывает после вставки данных в таблицу ТРАНСПОРТИРОВКА, если внесено, что животное транспортировали из нашего зоопарка, то автоматически данному животному указывается дата отъезда.**

```
create or replace trigger "Animal_moving"
after insert ON "ТРАНСПОРТИРОВКА"
for each row
BEGIN
if (:NEW.З_ОТПРАВКИ in ('В мире животных')) then
UPDATE Обитатели SET дата_езда = :NEW.ДАТА
WHERE (Обитатели.id_животного = :New.ID_ЖИВОТНОГО);
end if;
END;
```

**Триггер срабатывающий перед изменением или вставкой данных в таблицу ОБИТАТЕЛИ и проверяющий, что животное поселится в клетку с совместимыми животными, если же его хотят поселить в клетку с животными с которыми данное не уживется, то данный триггер не дает изменить номер клетки.**

```
create or replace trigger "Moving_into_cells"
before insert OR Update ON "ОБИТАТЕЛИ"
for each row
BEGIN
IF ((:NEW.НОМЕР_КЛЕТКИ not NULL) AND ((Select id_животного
FROM Обитатели
WHERE номер_клетки = :NEW.НОМЕР_КЛЕТКИ) not in
(Select id_животного
FROM Совместимость
WHERE номер_животного = :NEW.id_животного) ))THEN
:NEW.НОМЕР_КЛЕТКИ := :OLD.НОМЕР_КЛЕТКИ
END IF;
END;
```

**Триггер срабатывающий после внесения данных в таблицу СОВМЕСТИМОСТЬ и вносит новые зеркальные данные.**

```
create or replace trigger "SWAP"
after insert ON "СОВМЕСТИМОСТЬ"
```

```

for each row
BEGIN
  INSERT INTO СОВМЕСТИМОСТЬ VALUES(:NEW.индекс + 1,
:NEW.номер_животного, :NEW.id_животного);
END;

```

**Триггер срабатывает после внесения изменений в таблицу**

**СОВМЕСТИМОСТЬ, которые автоматически изменяет зеркальные данные.**

```

create or replace trigger "CHANGE"
after update ON "СОВМЕСТИМОСТЬ"
for each row
BEGIN
  UPDATE СОВМЕСТИМОСТЬ SET id_животного =
:NEW.номер_животного, номер_животного = :NEW.id_животного
  WHERE ((id_животного = :OLD.номер_животного)AND(номер_животного
= :OLD.id_животного));
END;

```

**Триггер срабатывающий после внесения данных в таблицу ПОСТАВКИ и автоматически добавляющий поставленное количество корма на склад.**

```

create or replace trigger "Food_supply"
after insert ON "ПОСТАВКИ"
for each row
BEGIN
  IF (:NEW.НАЗВАНИЕ_КОРМА not in (Select название_корма
FROM склад)) THEN
    INSERT INTO СКЛАД VALUES (:NEW.название_корма, :NEW.вес)
  ELSEIF (:NEW.НАЗВАНИЕ_КОРМА in (Select название_корма
FROM склад)) THEN
    UPDATE СКЛАД SET (вес = вес + :NEW.вес)
    WHERE (:NEW.название_корма = название_корма)
  END IF;
END;

```

**Триггер срабатывающий при удалении данных из таблицы “Работники” и удаляющий все зависимые данные из других таблиц**

```

CREATE OR REPLACE TRIGGER "Removal"
BEFORE DELETE ON "РАБОТНИКИ"
FOR EACH ROW
BEGIN
  IF (:OLD.id_работы in ('4')) THEN
    DELETE FROM АДМИНИСТРАЦИЯ WHERE
(Администрация.id_работника = :OLD.id_работника);

```

```

END IF;
IF (:OLD.id_работы = 5) THEN
DELETE FROM ДРЕССИРОВЩИКИ WHERE (id_работника =
:OLD.id_работника);
END IF;
IF (:OLD.id_работы in (6, 7)) THEN
DELETE FROM ВЕТЕРИНАРЫ WHERE (id_работника =
:OLD.id_работника);
END IF;
IF (:OLD.id_работы in (1, 2)) THEN
DELETE FROM СТРОИТЕЛИ WHERE (id_работника =
:OLD.id_работника);
END IF;
DELETE FROM ДОСТУП WHERE (id_работника = :OLD.id_работника);
END;

```

## Процедуры

**Процедура получающая на вход данные о работнике и добавляющая его в необходимые таблицы**

```

CREATE OR REPLACE PROCEDURE "Addworker" (gender IN CHAR(1),
birth IN DATE, work IN NUMBER, name IN VARCHAR2(100), surname IN
VARCHAR2(100), first IN VARCHAR2(100), second IN VARCHAR2(100), dt
IN DATE)
AS
BEGIN
id:= (SELECT MAX(id_работника)
FROM Работники) + 1;
INSERT INTO РАБОТНИКИ VALUES(id, SYSDATE, gender, birth, work,
name, surname);
if (work = 4) THEN
INSERT INTO АДМИНИСТРАЦИЯ VALUES(id, first);
END IF;
IF ((work = 6)OR(work =7)) THEN
INSERT INTO ВЕТЕРИНАРЫ VALUES(id, first, dt, second);
END IF;
IF (work = 5) THEN
INSERT INTO ДРЕССИРОВЩИКИ VALUES(id, first);
END IF;
IF ((work = 1)OR(work = 2)) THEN
INSERT INTO СТРОИТЕЛИ VALUES(id, first, second);

```



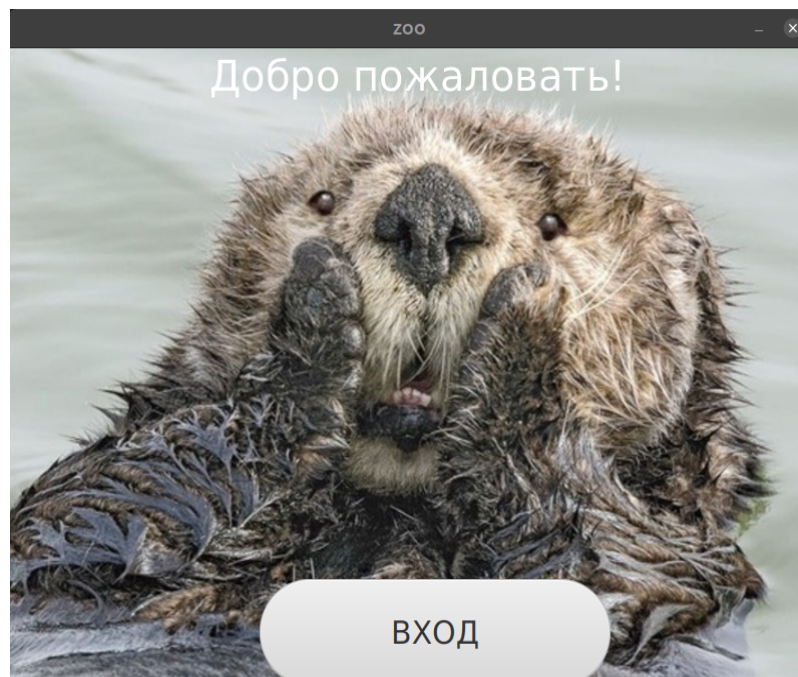
```
END IF;  
END;
```

## Разработка приложения клиента (формы ввода, редактирования и поиска данных по запросам)

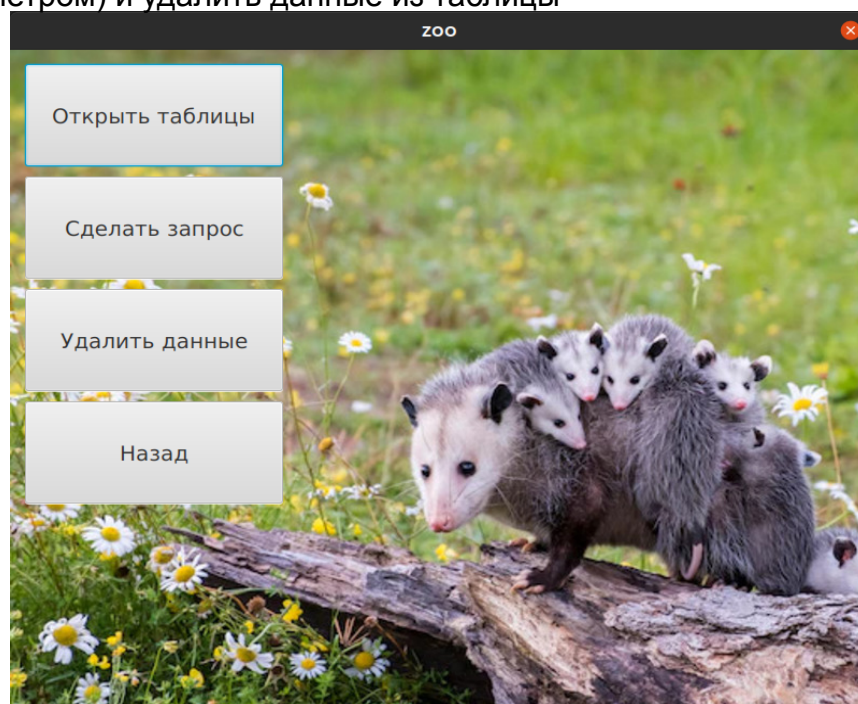
Реализовано клиентское приложение для работников зоопарка на языке Java с использованием базы данных ORACLE. Приложение позволяет просматривать таблицы, делать запросы с параметром и без, удалять данные из таблиц.

Приложение имеет следующий вид:

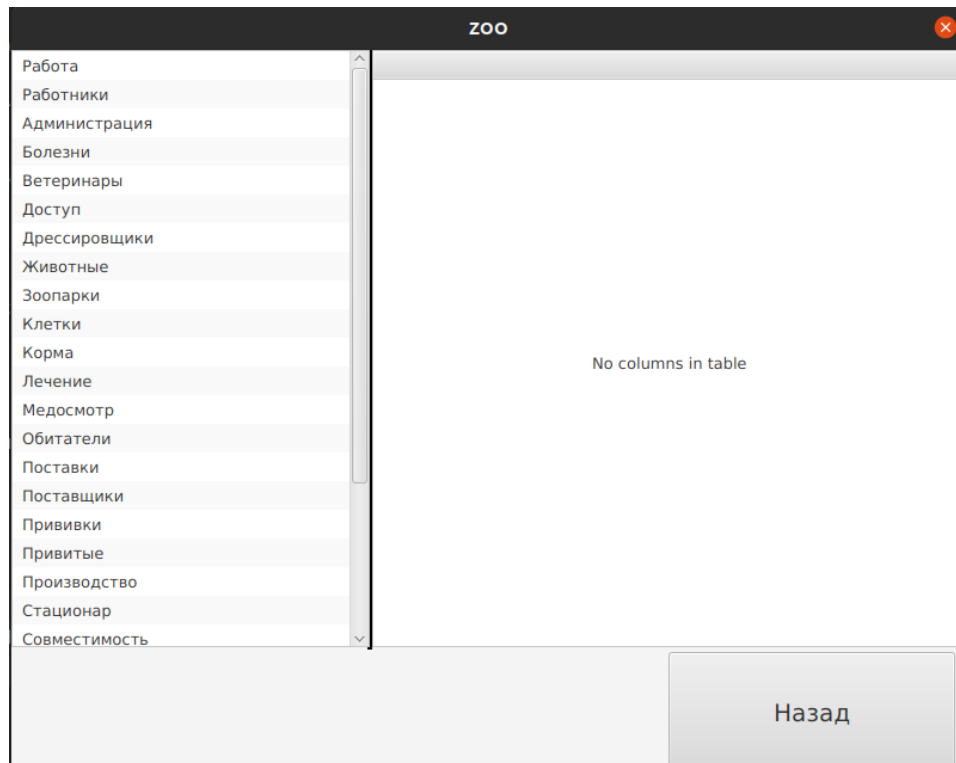
1. Приветствие, в рамках данного окна вы можете войти в систему.



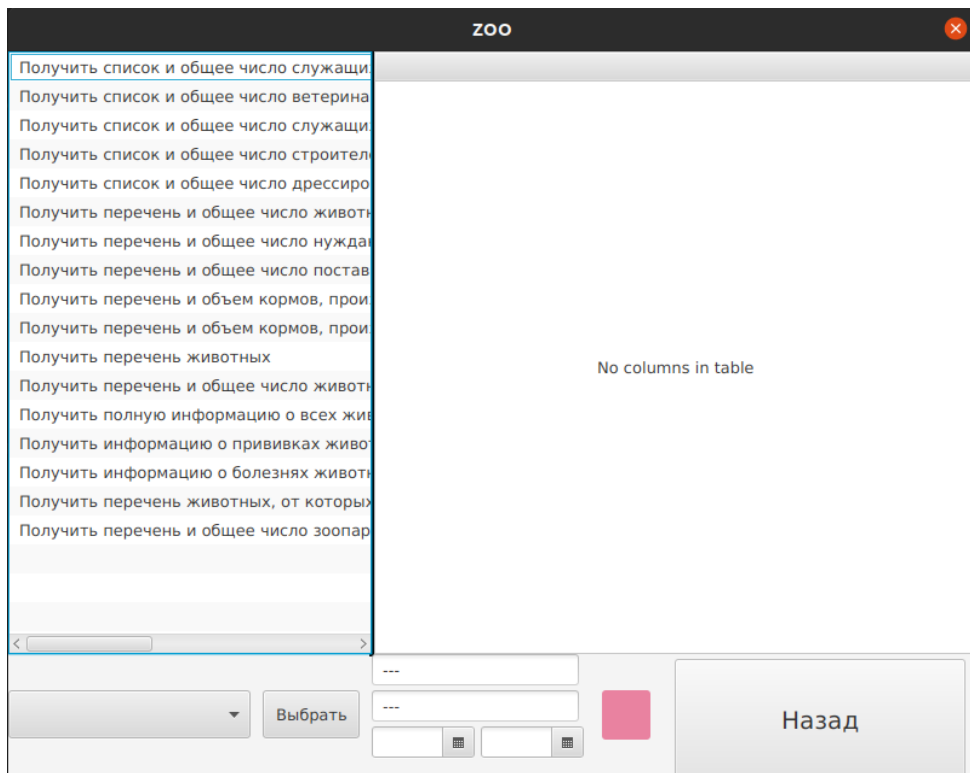
2. После осуществления входа, на выбор предлагают три действия: Посмотреть таблицы, сделать запросы(в том числе запросы с параметром) и удалить данные из таблицы



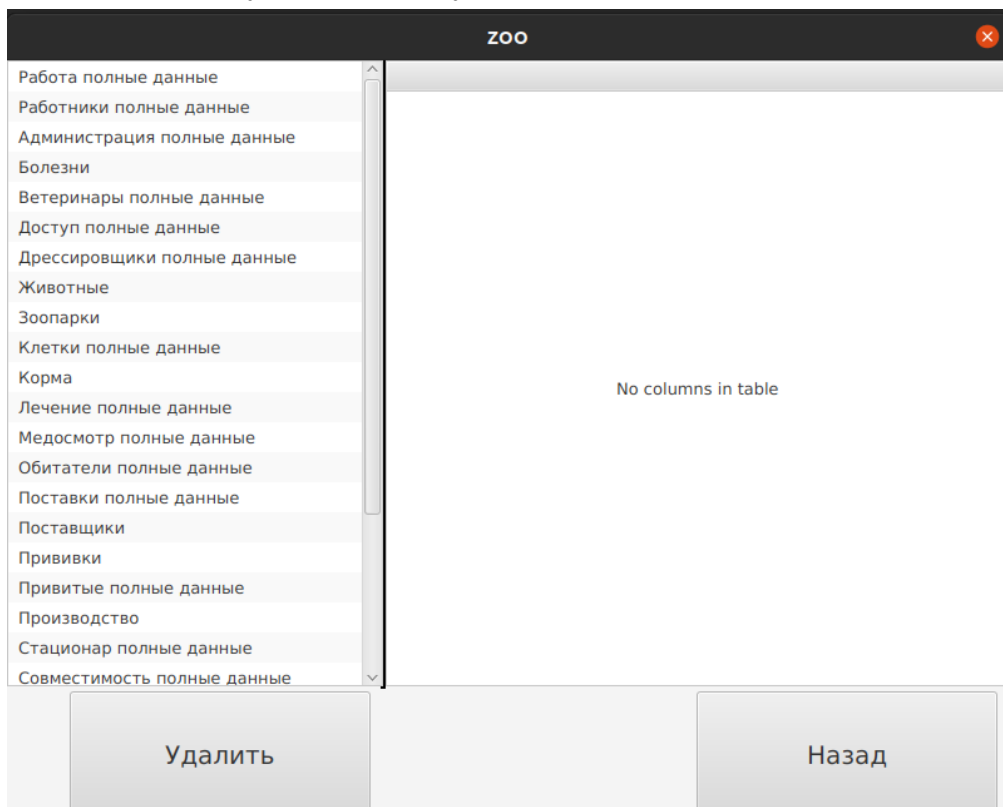
3. Здесь вы можете посмотреть все таблицы



4. Здесь вы можете посмотреть запросы, у некоторых запросов можно выбрать параметры, которые появляются в выпадающем списке, после применения параметров, для удобства, вам покажется что и куда записывать. Для выполнения запроса с параметром нужно нажать розовую кнопку



5. Здесь вы можете удалить данные из таблиц, для этого выбираете таблицу, выбираете строчку и нажимаете удалить.



Вывод: В ходе проделанной работы в рамках курса “Базы данных” было реализовано клиентское приложение на языке Java с использованием спроектированной базы данных зоопарка, создана графическая схема этой базы данных с указанием связей на основе технического задания, реализованы запросы для взаимодействия с базой данных, а также триггеры для осуществления контроля работы базы данных.