Санкт-петербургский государственный университет
Кафедра информационно-аналитических систем

Помыткина Татьяна Борисовна

# Язык SQL

Дисциплина [003659] «Базы данных»
по направлению 09.03.04 Программная инженерия

Рекомендовано к печати Учебно-методической комиссией
по УГСН 09.00.00 «Информатика и вычислительная техника»

Санкт-Петербург, 2024

# Содержание

## Введение

Методическое пособие предназначено для студентов 3 курса СПбГУ направления 09.03.04 Программная инженерия и сопровождает практические занятия по учебному курсу Базы данных.

Задания, представленные в пособии, направлены на получение обучающимися практических навыков программирования на языке SQL, и в первую очередь – навыков построения SQL-запросов разной степени сложности.

Выполнение этих заданий предполагается проводить в компьютерных классах в присутствии преподавателя, с возможностью обсуждать результаты выполнения заданий в индивидуальном и групповом форматах.

Пособие состоит из двух основных частей:

1. Общая часть, задания которой относятся к практике на применение знаний и умений.

   Во время выполнения этих заданий обучающиеся применяют теоретические положения в условиях решения учебных задач и выполняют упражнения по образцам, при этом им доступны вспомогательные материалы в виде презентаций, сопровождающих курс.

2. Самостоятельное задание, которое представляет собой практику по формированию умений и навыков.

   При выполнении этого задания обучающиеся творчески используют сформированные на предыдущем этапе умения и навыки. Самостоятельное задание каждый обучающийся реализует на своей индивидуальной базе данных, разработка которой является частью этого задания.

По окончании курса обучающиеся будут уметь описывать структуры данных в терминах модели «Сущность-связь», нормализовать эту модель, создавать объекты базы данных, описывать ограничения целостности для данных, писать запросы на языке SQL, создавать представления, триггеры, хранимые процедуры и функции.

## Подготовка к выполнению заданий

Для работы на практике обучающийся может выбрать любую из свободно распространяемых реляционных систем управления базами данных. В качестве среды разработчика допустимо использовать не только клиентское приложение выбранной СУБД, но и любую универсальную программу, предоставляющую сервисы соединения с сервером СУБД и функциональность для отладки SQL-скриптов.

> Замечание:
> Выбор, установка и настройка программного обеспечения не представляют особой трудности для студентов третьего курса направления Программная инженерия, которые с первого семестра обучаются решать задачи комплексно и в условиях, приближенных к реальным. Кроме того, они всегда имеют возможность получить на практических занятиях консультацию преподавателя. Так что эта задача является для студентов полезным дополнительным упражнением в практическом курсе по работе с базами данных.

Для лучшего освоения материала студентам рекомендуется поработать более чем с одной СУБД. Например, задания общей части выполнять в среде MS SQL Server, доступной в компьютерных классах, а самостоятельное задание – в среде той СУБД, которую обучающийся самостоятельно выберет и установит на личном компьютере.

После установки программного обеспечения на личном компьютере рекомендуется произвести проверку готовности его к работе. Для этого можно выполнить следующие шаги:

- Создать тестовую базу данных:
  ```
  CREATE DATABASE test;
  ```
- Создать в базе таблицу:
  ```
  CREATE TABLE Person(Name VARCHAR, Phone VARCHAR);
  ```
- Добавить в таблицу данные:
  ```
  INSERT INTO Person VALUES ('Иванов', '+79123456789');
  INSERT INTO Person VALUES ('Петров', '+7900000000');
  ```
- Выполнить выборку данных:
  ```
  SELECT * FROM Person;
  ```

Если все операторы отработали безошибочно и результатом выполнения SELECT оказались введенные ранее данные о персонах, значит установка и настройка СУБД прошли успешно и можно приступать к выполнению заданий.

## Общая часть

### *1. Создание учебной базы данных*

Создайте учебную базу данных с помощью скрипта, приведенного в Приложении. Для этого:

- Из среды разработчика соединитесь с сервером базы данных, скопируйте в рабочее окно скрипт из приложения. Этот скрипт написан для СУБД MS SQL Server, для сохранения его работоспособности в другой реляционной СУБД могут понадобиться небольшие корректировки кода в части использования типов данных, форматов констант и стандартных функций.

- Выполните скрипт и убедитесь, что он отработал без ошибок, и что новая база данных успешно создана и заполнена тестовыми данными.

- Если в среде разработчика имеется такая возможность, сгенерируйте er-диаграмму по созданной уже базе данных – для дополнительного контроля успешности создания правильной схемы.

Учебная база данных предназначена для работы со следующей предметной областью:

> *Компания обеспечивает для жителей микрорайона услуги по выгулу собак, стрижке и кормлению собак и котов. Структура хранения данных должна быть такова, чтобы можно было работать со следующей информацией:*
>
> - *Списки владельцев (ФИО, адрес, телефон, питомец).*
> - *Списки котов (кличка, возраст, порода) и собак (кличка, порода, время выгула).*
> - *Списки исполнителей (ФИО, адрес, телефон, специализация).*
> - *Заказы на неделю (владелец, питомец, исполнитель, дата, отметка о выполнении, оценка работы).*
> - *Пакет заказов на день.*
> - *Отчеты для владельцев.*
> - *Рейтинг исполнителей.*

### *2. Доработка скрипта*

Дополните структуру базы данных двумя новыми сущностями: *прививка* и *вид прививки*: (“Мы должны знать даты постановки питомцам прививок и иметь сканы документов, подтверждающих факты вакцинации”). Доработайте скрипт создания базы данных, добавив необходимые операторы создания таблиц и правил целостности, операторы добавления тестовых данных в новые таблицы.

Для этого напишите небольшой скрипт, включающий:

- операторы создания новых таблиц и ключей
- операторы добавления тестовых записей (по 2-5 для каждой таблицы)
- операторы удаления таблиц (в правильном порядке)

Выполните этот скрипт на учебной базе. Убедитесь, что он отработал безошибочно. Если имеется такая возможность, сгенерируйте er-диаграмму по измененной базе данных.

### *3. Простые запросы к одной таблице*

Напишите следующие запросы к таблице питомцев учебной базы данных:

1. Данные на питомца по кличке Партизан.
2. Клички и породы всех питомцев с сортировкой по возрасту.
3. Питомцы, имеющие хоть какое-нибудь описание.
4. Средний возраст пуделей.
5. Количество владельцев.
6. Сколько имеется питомцев каждой породы.
7. Сколько имеется питомцев каждой породы (если только один - не показывать эту породу).

Сами придумайте к этой таблице запросы с использованием перечисленных ниже синтаксических элементов:

8. запрос с BETWEEN
9. запрос с LIKE
10. запрос с IN(...)

### *4. Запросы на соединение таблиц*

Напишите следующие запросы к таблицам учебной базы данных (попробуйте использовать в разных запросах оба синтаксиса соединения таблиц - с JOIN и без):

1. Данные на Партизана (включая вид животного).
2. Список всех собак с кличками, породой и возрастом.
3. Средний возраст кошек.
3. Время и исполнители невыполненных заказов.
4. Список хозяев собак (имя, фамилия, телефон).
5. Все виды питомцев и клички представителей этих видов (внешнее соединение).

6. Сколько имеется котов, собак и т.д. в возрасте 1 год, 2 года, и т.д.

7. Фамилии сотрудников, выполнивших более трех заказов.

8. Придумайте какой-нибудь осмысленный запрос про прививки, в котором задействованы не менее четырех таблиц базы данных. Не забудьте добавить текстовую формулировку.

### *7. Запросы сложной структуры*

Напишите следующие запросы к таблицам демонстрационной базы данных:

1. Все оценки по заказам, исполнителями которых являлись студенты.
   (используйте IN (SELECT...))
2. Фамилии исполнителей, не получивших еще ни одного заказа.
   Последовательность отладки:
   - id исполнителей, у которых нет заказов
     (используйте NOT IN (SELECT...))
   - фамилии этих исполнителей
     (присоедините еще одну таблицу)
3. Список заказов (вид услуги, время, фамилия исполнителя, кличка питомца, фамилия владельца).
   (используйте псевдонимы)
4. Общий список всех комментариев, имеющихся в базе.
   (“Хватит захламлять базу, посмотрите, что вы пишите в комментариях!”).
   (используйте UNION)
5. Перепишите предыдущий запрос в каком-либо ином синтаксисе, без EXISTS.

### *8. Модификация данных*

1. Напишите оператор, добавляющий пометку “(s)” в начало комментария по каждому заказу, исполнитель которого – студент. Выполните этот оператор.

2. Напишите оператор, удаляющий все заказы по combing-у, которые еще не выполнены (“Мы приостанавливаем оказание этой услугу из-за не продления лицензии”).

3. Напишите оператор, добавляющий в таблицу PERSON новое физическое лицо с сохранением последовательной нумерации записей (используйте вложенный select с “max(…) + 1”).

   Добавьте одну новую запись (например, себя).

   Создайте в базе новую таблицу для хранения данных о документах физ.лиц (вид и номер документа). При создании связи от нее к таблице PERSON укажите свойства каскадности редактирования и удаления.

   Добавьте в нее пару документов для только что созданного нового физ.лица.

Убедитесь, что каскад работает:

- Измените Person_ID последнего созданного физ.лица и проверьте, остались ли его документы за ним.

- Удалите это физ.лицо и посмотрите, удалились ли автоматически все его документы.

Подумайте, каким еще ключам следовало бы добавить каскадные свойства, чтобы модель сохранила работоспособность. Ведь если этого не сделать, попытка удаления любой персоны, зарегистрированной уже как сотрудник или владелец, приведет к ошибке.

### *9. Создание представлений*

1. Создайте представление “Собаки” со следующими атрибутами: кличка, порода, возраст, фамилия и имя хозяина. Используя это представление, получите список пуделей: кличка, фамилия хозяина.

2. Создайте представление “Рейтинг сотрудников”: фамилия, имя, количество выполненных заказов, средний балл (по оценке). Используя это представление, выведите рейтинг с сортировкой по убыванию балла.

3. Создайте представление “Заказчики”: фамилия, имя, телефон, адрес. Используя это представление, напишите оператор, после выполнения которого у всех заказчиков без адреса в это поле добавится вопросительный знак.

   Замечание: Если вы работаете в СУБД, где невозможно редактировать представления с JOIN, попробуйте изменить запрос – перепишите его с IN. Если и так не работает, сделайте вместо редактирования выборку, как в предыдущих пунктах.

## Задание для самостоятельной работы

В процессе выполнения этого задания предполагается применение и развитие следующих навыков: проектирование модели сущность-связь по текстовой формулировке требований к информационной системе; создание и сопровождение базы данных; проектирование и реализация отдельных элементов серверной и клиентской частей информационной системы, связанных с обработкой данных.

1. **Спроектируйте базу данных.**

   Во вспомогательных материалах к практическим занятиям предложен большой перечень описаний предметных областей. Рассмотрите одну из как набор требований к созданию небольшой информационной системы.

   Нарисуйте ER-диаграмму для базы данных - центра этой информационной системы (можно пользоваться любыми техническими средствами). Диаграмма должна состоять примерно из 7-8 таблиц, связанных между собой.

2. **Напишите скрипт создания и заполнения базы данных.**

   Помимо включения контроля целостности по ссылкам (PK и FK, в том числе с опцией cascade), не забудьте добавить элементы контроля целостности по сущностям (NOT NULL, UNIQUE, CHECK).

   Добавьте операторы создания индексов – по полям и комбинациям полей, которые в перспективе могут оказаться особо востребованными в операциях поиска, в фильтрах и т.д.

   Не забудьте написать операторы заполнения базы тестовыми данными. Эти данные должны быть более-менее осмысленными, где-то от 3 до 50 записей в каждой таблице.

   Включите в скрипт операторы удаления всех таблиц (операторы drop, в правильном порядке).

   При желании можете поэкспериментировать с такими вещами, как авто-генерация значений уникальных полей (в том числе с использованием секвенций и триггеров), ссылка таблицы самой на себя и проч.

   Отладьте скрипт в среде разработчика той реляционной СУБД, которую вы выбрали для реализации проекта.

3. **Опишите функциональность проектируемой информационной системы и реализуйте некоторые элементы серверной и клиентской частей.**

При описании не замахивайтесь на полноту и точность отражения предметной области, главное – продемонстрировать понимание сути разделения функциональности между серверной и клиентской частями системы.

При реализации также не требуется создавать законченную версию информационной системы, главное – потренироваться в использовании различных возможностей SQL (а заодно и языка программирования выбранной СУБД). Однако же используйте при реализации все следующие элементы SQL:

- фильтры, сортировки, функции агрегирования;
- соединение двух и более таблиц;
- внешнее соединение таблиц (left или right join);
- группирование и фильтрация групп (group by и having);
- подзапросы двух-трех разных видов (select…(select…)… );
- соединение двух или более запросов (union) с сортировкой;
- изменение и удаление данных с подзапросами.

Подробнее:

3.1. Разработайте и опишите бизнес-логику серверной части информационной системы – выделите операции по обработке данных, которые имеет смысл хранить в самой базе данных. Перечислите также основные элементы функционала клиентской части. Объем – по семь-восемь элементов каждого вида, относительно разнообразных.

3.2. Реализуйте часть бизнес-логики путем создания хранимых процедур\функций, триггеров и представлений, которые могут пригодиться либо для вызова из клиентской части, либо внутри серверной.

В результате должно быть не менее трех реализованных процедур\функций, трех триггеров, трех представлений. И где-то должны появиться операторы, вызывающие выполнение перечисленных объектов (хотя бы в комментариях). Рекомендую поэкспериментировать в процедурах с конструкцией cursor.

3.3. Создайте несколько запросов к данным, которые могли бы быть использованы при реализации пользовательского интерфейса и отчетов.

Таких запросов должно быть не менее дюжины, и они не должны быть совсем уж примитивными или однообразными. Проверьте, какие элементы и синтаксические конструкты языка SQL (упомянутые в начале п.3). не случилось использовать при реализации бизнес-логики и реализуйте их здесь, пусть даже запросы получатся относительно искусственными. В запросах можно обращаться не только к таблицам, но и к созданными ранее представлениям.

4. **Создайте отчет по выполненной работе.**

Отчет должен содержать требования к информационной системе, диаграмму сущность-связь, краткое описание функциональности системы, перечень и описание приложенных скриптов:

- Скрипт создания и тестового заполнения таблиц базы данных.
- Скрипт создания процедур, функций, триггеров и представлений.
- Скрипт клиентских запросов.
- Скрипт удаления всех объектов базы данных в продуманном порядке.

Все скрипты должны быть отлаженными и содержать краткие комментарии.

## Приложение: Скрипт создания учебной базы данных

```
CREATE DATABASE pet_db;
GO
USE pet_db;
-----------------------------------------------------------------
-- Создание таблиц и PK
-----------------------------------------------------------------
CREATE TABLE Person(
    Person_ID         INTEGER      NOT NULL,
    Last_Name           VARCHAR(20)  NOT NULL,
    First_Name          VARCHAR(20),
    Phone         VARCHAR(15)  NOT NULL,
    Address       VARCHAR(50)  NOT NULL,
CONSTRAINT Person_PK PRIMARY KEY (Person_ID)
)
;
CREATE TABLE Owner(
    Owner_ID                INTEGER         NOT NULL,
    Description               VARCHAR(50),
    Person_ID               INTEGER         NOT NULL,
CONSTRAINT Owner_PK PRIMARY KEY (Owner_ID)
)
;
CREATE TABLE Employee(
    Employee_ID           INTEGER         NOT NULL,
    Spec              VARCHAR(15),
    Person_ID             INTEGER         NOT NULL,
CONSTRAINT Employee_PK PRIMARY KEY (Employee_ID)
)
;
CREATE TABLE Pet_Type(
    Pet_Type_ID           INTEGER      NOT NULL,
    Name              VARCHAR(15)  NOT NULL,
CONSTRAINT Pet_Type_PK PRIMARY KEY (Pet_Type_ID)
)
;
CREATE TABLE Pet(
    Pet_ID                  INTEGER      NOT NULL,
    Nick                VARCHAR(15)  NOT NULL,
    Breed                   VARCHAR(20),
    Age                     INTEGER,
    Description             VARCHAR(50),
    Pet_Type_ID             INTEGER         NOT NULL,
    Owner_ID                INTEGER         NOT NULL,
CONSTRAINT Pet_PK PRIMARY KEY (Pet_ID)
)
;
CREATE TABLE Service(
    Service_ID              INTEGER         NOT NULL,
    Name                VARCHAR(15)  NOT NULL,
CONSTRAINT Service_PK PRIMARY KEY (Service_ID)
)
;
CREATE TABLE Employee_Service(
    Employee_ID                   INTEGER         NOT NULL,
    Service_ID                    INTEGER         NOT NULL,
    Is_Basic                        INTEGER
)
;
CREATE TABLE Order1(
    Order_ID                      INTEGER         NOT NULL,
    Owner_ID                      INTEGER         NOT NULL,
    Service_ID                    INTEGER         NOT NULL,
```

```
    Pet_ID                          INTEGER         NOT NULL,
    Employee_ID                     INTEGER         NOT NULL,
    Time_Order                      DATETIME      DEFAULT GETDATE()    NOT NULL,
    Is_Done                    INTEGER       DEFAULT 0           NOT NULL,
    Mark                       INTEGER,
    Comments                        VARCHAR(50),
CONSTRAINT Order_Is_Done CHECK (Is_Done in (0,1)),
CONSTRAINT Order_PK PRIMARY KEY (Order_ID)
)
;

-------------------------------------------------------------
-- Создание FK
-------------------------------------------------------------
ALTER TABLE Owner ADD CONSTRAINT FK_Owner_Person
    FOREIGN KEY (Person_ID)
    REFERENCES Person(Person_ID)
;
ALTER TABLE Employee ADD CONSTRAINT FK_Employee_Person
    FOREIGN KEY (Person_ID)
    REFERENCES Person(Person_ID)
;
ALTER TABLE Pet ADD CONSTRAINT FK_Pet_0wner
    FOREIGN KEY (Owner_ID)
    REFERENCES Owner(Owner_ID)
;
ALTER TABLE Pet ADD CONSTRAINT FK_Pet_Pet_Type
    FOREIGN KEY (Pet_Type_ID)
    REFERENCES Pet_Type(Pet_Type_ID)
;
ALTER TABLE Employee_Service ADD CONSTRAINT FK_Empl_Serv_Employee
    FOREIGN KEY (Employee_ID)
    REFERENCES Employee(Employee_ID)
;
ALTER TABLE Employee_Service ADD CONSTRAINT FK_Empl_Serv_Service
    FOREIGN KEY (Service_ID)
    REFERENCES Service(Service_ID)
;
ALTER TABLE Order1 ADD CONSTRAINT FK_Order_Employee
    FOREIGN KEY (Employee_ID)
    REFERENCES Employee(Employee_ID)
;
ALTER TABLE Order1 ADD CONSTRAINT FK_Order_0wner
    FOREIGN KEY (Owner_ID)
    REFERENCES Owner(Owner_ID)
;
ALTER TABLE Order1 ADD CONSTRAINT FK_Order_Pet
    FOREIGN KEY (Pet_ID)
    REFERENCES Pet(Pet_ID)
;
ALTER TABLE Order1 ADD CONSTRAINT FK_Order_Service
    FOREIGN KEY (Service_ID)
    REFERENCES Service(Service_ID)
;

-------------------------------------------------------------
-- Заполнение таблиц тестовыми данными
-------------------------------------------------------------

INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (1,
'Иванов', 'Иван', '+79123456789', 'Средний пр. ВО, 34-2');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (2,
'Петров', 'Петр', '+79234567890', 'Садовая ул., 17\2-23');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (3,
'Васильев', 'Василий', '+7345678901', 'Невский пр., 9-11');
```

```
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (4,
'Орлов', 'Олег', '+7456789012', '5 линия ВО, 45-8');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (5,
'Галкина', 'Галина', '+7567890123', '10 линия ВО, 35-26');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (6,
'Соколов', 'С.', '+7678901234', 'Средний пр. ВО, 27-1');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (7,
'Воробьев', 'Вова', '123-45-67', 'Университетская наб., 17');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (8,
'Иванов', 'Ваня', '+7789012345', 'Малый пр. ВО, 33-2');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (9,
'Соколова', 'Света', '234-56-78', '');
INSERT INTO Person(Person_ID, Last_Name, First_Name, Phone, Address) VALUES (10,
'Зотов', 'Миша', '111-56-22', '');

INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (1, 'хороший парень',
2);
INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (2, '', 3);
INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (3, '', 5);
INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (4, 'из Академии
Художеств', 6);
INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (5, '', 8);
INSERT INTO Owner(Owner_ID, Description, Person_ID) VALUES (6, 'странный', 9);

INSERT INTO Employee(Employee_ID, Spec, Person_ID) VALUES (1, 'шеф', 1);
INSERT INTO Employee(Employee_ID, Spec, Person_ID) VALUES (2, 'грумер', 4);
INSERT INTO Employee(Employee_ID, Spec, Person_ID) VALUES (3, 'студент',7);
INSERT INTO Employee(Employee_ID, Spec, Person_ID) VALUES (4, 'студент', 10);

INSERT INTO Pet_Type(Pet_Type_ID, NAME) VALUES (1, 'СОБАКА');
INSERT INTO Pet_Type(Pet_Type_ID, NAME) VALUES (2, 'КОТ');
INSERT INTO Pet_Type(Pet_Type_ID, NAME) VALUES (3, 'КОРОВА');
INSERT INTO Pet_Type(Pet_Type_ID, NAME) VALUES (4, 'РЫБКА');

INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (1, 'Бобик', 'unknown', 3, NULL, 1, 1);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (2, 'Муся', NULL, 12, NULL, 2, 1);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (3, 'Каток', NULL, 2, 'буйный', 2, 1);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (4, 'Апельсин', 'poodle', 5, NULL,1, 2);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (5, 'Партизан', 'Siamese', 5, 'очень большой', 2, 2);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (6, 'Даниэль', 'спаниель', 14, NULL, 1, 3);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (7, 'Модель', NULL, 5, NULL, 3, 4);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (8, 'Маркиз', 'пудель', 1, NULL, 1, 5);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (9, 'Зомби', 'unknown', 7, 'дикий', 2, 6);
INSERT INTO Pet(Pet_ID, Nick, Breed, Age, Description, Pet_Type_ID, Owner_ID)
VALUES (10, 'Лас', 'сиамец', 7, '', 2, 6);

INSERT INTO Service(Service_ID, NAME) VALUES (1, 'Выгул');
INSERT INTO Service(Service_ID, NAME) VALUES (2, 'Груминг');
INSERT INTO Service(Service_ID, NAME) VALUES (3, 'Дойка');

INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (1, 1, 0);
INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (1, 2, 0);
INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (1, 3, 1);
INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (2, 1, 0);
INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (2, 2, 1);
INSERT INTO Employee_Service(Employee_ID, Service_ID, Is_Basic) VALUES (3, 1, 1);
```

```
-- установка формата даты:
set dateformat ymd;

INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (1, 1, 1, 1, 3, '2023-09-11 08:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (2, 1, 2, 2, 2, '2023-09-11 09:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (3, 1, 2, 3, 2, '2023-09-11 09:00', 1, 4, 'Этот кот просто дикий!');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (4, 1, 1, 1, 3, '2023-09-11 00:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (5, 1, 1, 1, 3, '2023-09-16 11:00', 1, 3, 'опаздал');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (6, 1, 1, 1, 3, '2023-09-17 17:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (7, 1, 2, 2, 2, '2023-09-17 18:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (8, 2, 1, 5, 3, '2023-09-17 18:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (9, 2, 1, 4, 3, '2023-09-17 10:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (10, 2, 1, 5, 3, '2023-09-18 17:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (11, 2, 1, 4, 3, '2023-09-18 18:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (12, 2, 1, 5, 3, '2023-09-18 12:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (13, 2, 1, 4, 3, '2023-09-18 14:00', 1, 4, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (14, 3, 1, 6, 3, '2023-09-19 10:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (15, 3, 2, 6, 2, '2023-09-19 18:00', 0, 0, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (16, 3, 1, 6, 3, '2023-09-20 10:00', 0, 0, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (17, 3, 1, 6, 3, '2023-09-20 11:00', 0, 0, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (18, 3, 1, 6, 3, '2023-09-22 10:00', 0, 0, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (19, 3, 1, 6, 3, '2023-09-23 10:00', 0, 0, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark, Comments)
       VALUES (20, 4, 3, 7, 1, '2023-09-30 11:00', 1, 5, '');
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark)
       VALUES (21, 4, 3, 7, 1, '2023-10-01 11:00', 0, 0);
```

```
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark)
       VALUES (22, 4, 3, 7, 1, '2023-10-02 11:00', 0, 0);
INSERT   INTO   Order1(Order_ID,   Owner_ID,   Service_ID,   Pet_ID,   Employee_ID,
Time_Order, Is_Done, Mark)
       VALUES (23, 5, 2, 8, 2, '2023-10-03 16:00', 0, 0);

-----------------------------------------------------------------
-- Удаление таблиц
-----------------------------------------------------------------

/*
DROP TABLE Order1;
DROP TABLE Employee_Service;
DROP TABLE Service;
DROP TABLE Pet;
DROP TABLE Pet_Type;
DROP TABLE Employee;
DROP TABLE Owner;
DROP TABLE Person;
*/
```