Database systems 02170

Group 36: Veterinarian Clinic

Contributors:

- Ion Cararus (s213209)
- Dimitrios Salogiannis (s212471)
- Krzysztof Pac (s213033)
- Zacharias Dandoulis (s210729)
- Yahir Alejandro de Leon Dominguez (s210097)

28th March 2023

1. Statement of Requirements

The goal of this project is to understand how relation databases are designed, modelled and programmed. One of the first requirements was to pick so-called real life scenario where implementing the database can be easily justified.

In our project, we decided to choose a veterinary clinic. It is quite easy to understand the requirements that has to be fulfilled, when designing the database. Moreover, as for the relevance of this idea, we agreed that every veterinary clinic needs a proper way to store and manage the data of the patients. For example, it is important to have an easy access to the history of medical treatment and be able to updated it without spending much time on it. It is very time-effective and economical solution.

Every veterinary clinic has a **veterinarian**. Each doctor must have an unique identifier that is used to properly distinguish and identify them within database. Moreover, veterinarian must have a first and last name. Contact information are necessary, and therefore the table also preserves data about an email address and phone number. Thanks to that, each doctor can be contacted easily in the emergency case. Additionally, veterinarians has an appointment fee that can inform clients about the expected cost of the treatment.

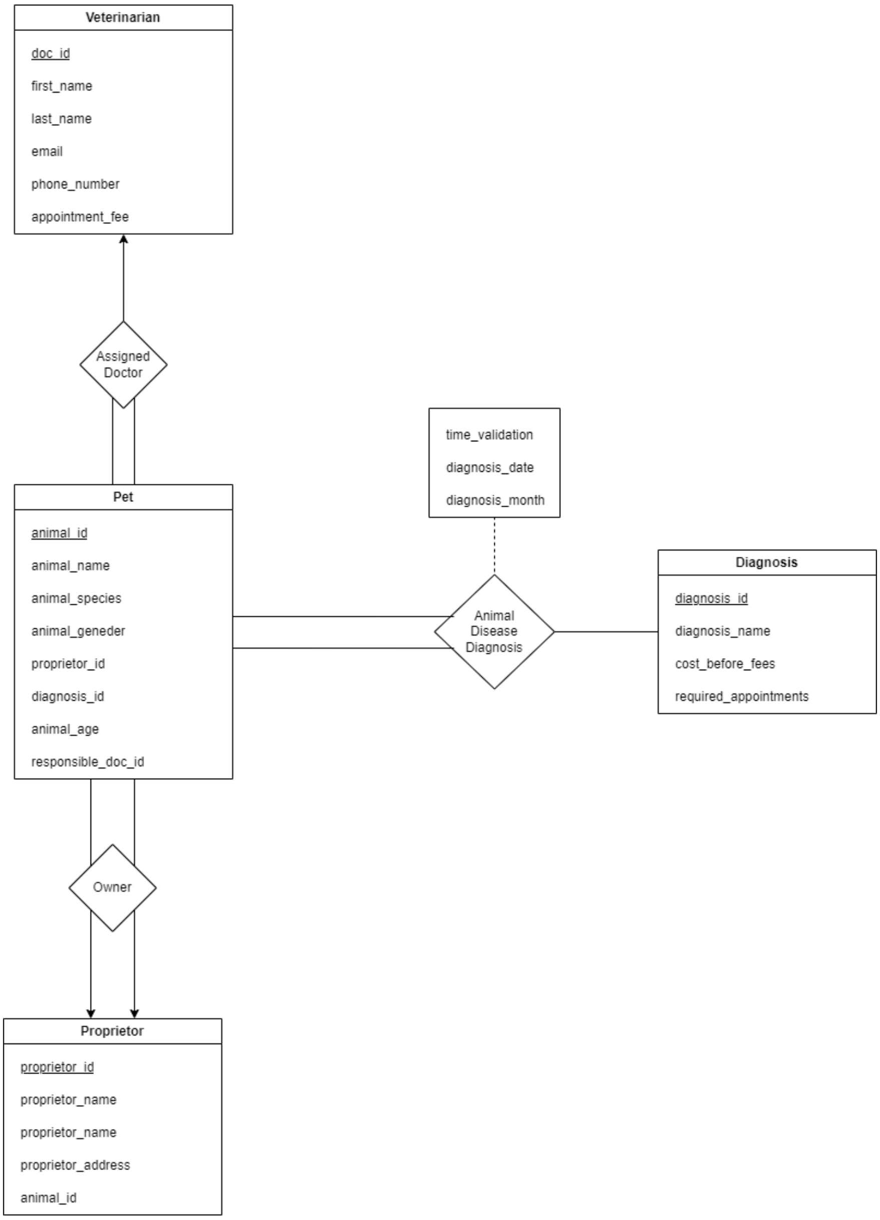
Patients of the clinic are **Pet**s. Each animal has an unique id, under which can be found in the database. In order to find out, who the patient is, clinic is storing data about the name, species, gender and age of the pet. Especially, the species, gender and name are important, because veterinarian can easily recognise the patient in the database and have the basic information provided, before the check-up. Moreover, each pet has a proprietor id, that leads to identify the owner. The last two attributes are a doctor id that has been assigned to the patient and the id of the last diagnosis.

As it was mentioned, each pet must have an owner. The **proprietor** can be also identified in the database with an unique id, assigned to them. Additionally, there rest information are the first and last name, that are also used to identify the owner. However, not from the database perspective, but human. In the end, owner has an address. There is also a simple **Ownership** relation between proprietor and animal, where the owners and its pets can be found as a pair of id's.

During the visit at veterinary clinic, each patient has a **diagnosis**. This table has been simplified. The diagnosis is a set of various illness/problems that were recognised in the clinic. Each diagnosis has an unique identifier, a name that says what is the problem, cost of the treatment (without doctors fee) and expected number of required appointments.

The last table (**Animal Disease Diagnosis**) preserves a relationship between pet and diagnosis. It stores the id of a patient and id of diagnosis. Thanks to that unique pair, we can have more than one diagnosis for each pet, saved in the database. Moreover, it makes sense to store the information about the date of medical check-up. Therefore, we decided that our database will also store the validation time and the date of the visit. Thanks to that, doctors can have an access to the whole treatment history chronologically preserved.

2. Conceptual Design



The above graph represents the conceptual model of our database. Starting from the veterinarian, we decided that doctors will be identified by doctor's id. This value is unique in the whole set and was chosen to be a primary key for that table. Thanks, to that it is possible to distinguish veterinarians and create relationships with the other entities. Other attributes are doctor's first and last name, email and phone number (to reach veterinarians when they are not present at the clinic). Lastly, we decided to add the information about appointment fee because veterinarians with greater experience and seniority tends be more proficient and therefore their expertise will cost more. The price that must be paid is a sum of medical treatment cost that is based on diagnosis (this will be covered later in that chapter) and veterinarian fee.

Each clinic has its patience. In our case those patients are animals. Once again, we decided to distinguish pets in the database by unique identifier (animal_id). Thanks, to that we can ensure that each creature is unrepeatable. This identifier was set to be a primary key. Moreover, relation has additionally two foreign keys that provide relationship with veterinarian and proprietor. The doc_id indicates which doctor is responsible for which animal. This relation is like in the Danish healthcare system where the citizen has assigned a first-contact doctor. Thanks to that, the clinic can ensure greater medical treatment, when veterinarian already has experience with its patient. Proprietor id is used to connect the pet with its owner. Between veterinarian and pet we have a partial participation, binary relation one-to-many. All pets registered in database must have assigned doctor, but not all veterinarians must have a patient. As for pet and proprietor the relationship is a total participation, many-to-one. It means that, owner can have more than one pet and when pet is required that both pet and owner are linked with each other (no pets without owner and no owner without a pet). Moreover, the relationship is represented in the ownership table, where there are stored id's of proprietor and pet. Both values create a primary key for that relation. Therefore, we preserved the unique key to identify each tuple. Later, this table can be modified by expanding it with new attributes, if needed. The remaining attributes in pet table are name, species, gender, age. Moreover, relation also stores the latest diagnosis.

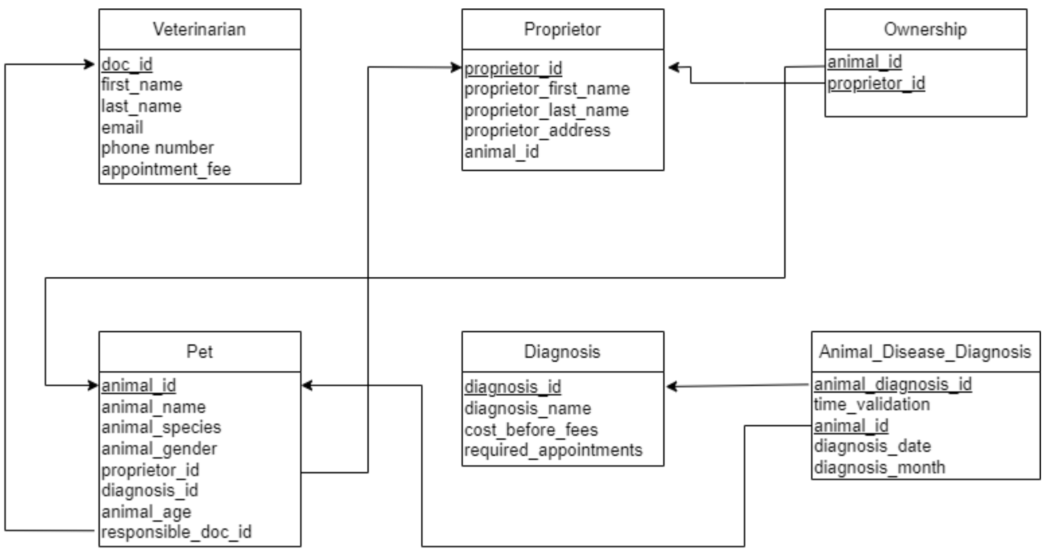
Next table is the aforementioned proprietor. The primary key is proprietor_id that ensures the uniqueness of the owner in the database system. Besides the id, owner must have first and last name and address.

In the veterinary system all patients are diagnosed. Each diagnosis is stored in the database with unique identifier. This id is set to be a primary key. Moreover, there are other information required to be able to say more about the problem. Therefore, other important attributes are diagnosis name, cost of treatment (without doctor's fee) and the number of required appointments that are needed to cure the patient completely.

Last table called "Animal Disease Diagnosis" represents the relationship between the pet and diagnosis. The primary key is a combination of diagnosis id from diagnosis table and animal id from pet table. This combination ensures that clinic can store the whole history of treatment for a specific animal, because diagnosis id will be different for each new diagnosis. Additionally, to save information about the time of the check-up, we decided to add the time and date of the visit at the clinic. The relationship between pet and diagnosis is partial participation, binary, many-to-one.

3. Logical Design

After gathering all information that were discussed above, we decided to build a logical design schema, which can be seen in the picture below.



The following relation schemas can be seen below:

Veterinarian(<u>doc_id</u>, first_name, last_name, email, phone_number, appointment_fee)

Proprietor(proprietor_id, proprietor_first_name, proprietor_last_name, proprietor_address, animal_id)

Pet(animal_id, animal_name, animal_species, animal_gender, proprietor_id, diagnsosis_id, animal_age, responsible_doc_id) **foreign key** (responsible_doc_id, proprietor_id) **references** (Veterinarian(doc_id), Proprietor(proprietor_id))

Ownership(animal_id, proprietor_id) **foreign key** (animal_id, proprietor_id) **references** (Pet(animal_id), Proprietor(proprietor_id)

Diagnosis(diagnosis_id, diagnosis_name, cost_before_fees, required_appointments)

Animal_Disease_Diagnosis(animal_diagnosis_id, time_validation, animal_id, diagnosis_date, diagnosis_month) **foreign key** (animal_diagnosis_id, animal_id) **references** (diagnosis(diagnosis_id), pet(animal_id))

4. Implementation

- ### Creating the database

```
DROP DATABASE IF EXISTS VetClinic;
CREATE DATABASE VetClinic;
USE VetClinic;

DROP TABLE IF EXISTS Veterinarian;
DROP TABLE IF EXISTS proprietor;
DROP TABLE IF EXISTS Ownership;
DROP TABLE IF EXISTS pet;
DROP TABLE IF EXISTS diagnosis;
DROP TABLE IF EXISTS animal_disease_diagnosis;
```

- ### Creating the tables

```
CREATE TABLE Veterinarian (
    doc_id BIGINT UNIQUE AUTO_INCREMENT,
    first_name VARCHAR(40),
    last_name VARCHAR(40),
    email VARCHAR(40),
    phone_number INT,
    appointment_fee INT,
    PRIMARY KEY(doc_id)
);


SELECT
    CONCAT(Veterinarian.First_Name,' ', Veterinarian.Last_Name)
    AS FullName
from Veterinarian;

CREATE TABLE Proprietor (
    proprietor_id INT UNIQUE AUTO_INCREMENT,
    proprietor_first_name VARCHAR(40),
    proprietor_last_name VARCHAR(40),
    proprietor_address VARCHAR(40),
    PRIMARY KEY(proprietor_id)
);

SELECT
    CONCAT(proprietor.proprietor_First_Name,' ', proprietor.proprietor_Last_Name)
    AS FullName
from proprietor;
```

```sql
CREATE TABLE Pet (
    animal_id BIGINT UNIQUE AUTO_INCREMENT NOT NULL,
    animal_name VARCHAR(40),
    animal_species VARCHAR(40),
    animal_gender ENUM ('male', 'female'),
    proprietor_id INT,
    diagnosis_id INT,
    animal_age INT,
    responsible_doc_id BIGINT,
    PRIMARY KEY(animal_id),
    FOREIGN KEY (responsible_doc_id) REFERENCES Veterinarian(doc_id),
    FOREIGN KEY (proprietor_id) REFERENCES Proprietor(proprietor_id)
);


CREATE TABLE Ownership (
animal_id BIGINT,
proprietor_id INT,
PRIMARY KEY (animal_id, proprietor_id),
FOREIGN KEY (proprietor_id) REFERENCES Proprietor (proprietor_id),
FOREIGN KEY (animal_id) REFERENCES Pet (animal_id)
);

CREATE TABLE Diagnosis (
    diagnosis_id BIGINT,
    diagnosis_name VARCHAR (40),
    Cost_before_fees DECIMAL(10,2),
    Required_appointments INT,
    PRIMARY KEY(diagnosis_id)
);

CREATE TABLE Animal_disease_diagnosis (
    animal_diagnosis_id BIGINT,
    time_validation TIME,
    animal_id BIGINT,
    diagnosis_date DATE,
    diagnosis_month int,
    PRIMARY KEY(animal_diagnosis_id, animal_id),
    FOREIGN KEY (animal_id) REFERENCES pet (animal_id),
    FOREIGN KEY (animal_diagnosis_id) REFERENCES Diagnosis (diagnosis_id)
);
```

- **Views**

```sql
CREATE VIEW doctor_info AS
    SELECT doc_id, CONCAT(Veterinarian.First_Name,' ', Veterinarian.Last_Name) AS FullName, phone_number
FROM Veterinarian;
```

5. Database Instance.

Below is how we populate the tables Ownership and Animal_disease_diagnosis

```sql
INSERT INTO Ownership
SELECT pet.animal_id, proprietor.proprietor_id
from pet
inner join proprietor
on pet.proprietor_id = proprietor.proprietor_id
order by animal_id;
```

PET

animal_id	animal_name	animal_species	animal_gender	proprietor...	diagnosis_id	animal_age	responsible_doc...
1	Bella	Dog	female	1	6	4	2
2	Luna	Rabbit	female	19	5	10	3
3	Ringo	Dog	male	1	7	6	8
4	Charlie	Cat	male	13	5	6	1
5	Lucy	Mouse	female	9	5	8	5
6	Seven	Dog	female	17	1	7	5
7	Yellow	Parrot	female	4	6	10	7
8	Zoe	Guinea pig	female	6	3	1	6
9	Milly	Horse	female	2	6	6	9
10	Lily	Dog	female	12	1	3	2
11	Ozzy	Snake	male	16	2	4	9

DIAGNOSIS

diagnosis_id	diagnosis_name	Cost_before_fe...	Required_appointme...
1	Strangles	270.00	15
2	Enzootic bovine leucosis	80.00	12
3	Rabies due to rabies virus	75.00	3
4	Influenza A in pigs	165.00	7
5	Avian influenza	325.00	13
6	Screw-worm fly	290.00	12
7	Botulism in poultry	145.00	14
8	Johne disease	405.00	8
NULL	NULL	NULL	NULL

PROPRIETOR

proprietor...	proprietor_first_na...	proprietor_last_na...	proprietor_address
1	Crichton	Jessen	96571 Kingsford Drive
2	Koren	Oakland	68 Thompson Court
3	Zachary	Peachman	2 Magdeline Court
4	Andrew	Glaubermann	86 Florence Avenue
5	Mallorie	Culcheth	2248 Derek Plaza
6	Brandy	Vallentine	6381 Ruskin Circle St.
7	Jesse	Pinkman	127 Nebraska St.
8	Pam	Beesly	524 Scrantoncity St.
9	Michael	Scott	4381 Di Loreto Terrace
10	Hugo	Stiglitz	9 Inglorious St.
11	Byrle	Glenton	22 Steensland Plaza
12	Alvaro	Vaya	Plaza de Pepe Mena 2
13	Phoebe	Buffay	Central Perk 178
14	Britta	Perry	Amazon St 91
15	Annie	Edison	Doodle St 4
16	Annie	Kim	Dan Harmon St. 1
17	Marshall	Kane	4th September St 51
18	Shirley	Bennett	Magnitude St. 6
19	Jim	Morrison	37 The Doors St.
20	Hermione	Granger	Kings Cross St 9
21	Harry	Potter	Hogwarts St 47

6. SQL Data Queries (more in the code)

- What are the top 3 most common diagnosed diseases?

The first query retrieves the top three most commonly diagnosed diseases by counting the number of animals with a specific diagnosis and grouping them by diagnosis name. It joins two tables, diagnosis and animal_disease_diagnosis, to get the required information.

```
SELECT d.diagnosis_id, d.diagnosis_name, COUNT(ad.animal_diagnosis_id) AS Frequency
FROM diagnosis d LEFT JOIN animal_disease_diagnosis ad
ON d.diagnosis_id = ad.animal_diagnosis_id
GROUP BY d.diagnosis_id, d.diagnosis_name
ORDER BY Frequency DESC
LIMIT 3
;
```

diagnosis_id	diagnosis_name	Frequency
1	Strangles	4
3	Rabies due to rabies virus	4
2	Enzootic bovine leucosis	3

- What is the total cost of treating every of the diseases with Dr. Alexander Müller?

The second SQL query calculates the total cost of treating each disease with a specific veterinarian, in this case Dr. Alexander Müller. It calculates the total fees by multiplying the number of appointments required for each diagnosis with the veterinarian's appointment fee and adds it to the cost before fees.

```
SELECT d.*,
d.Required_appointments * v.Appointment_fee AS Total_Fees,
((d.Required_appointments * v.Appointment_fee) + Cost_before_fees) AS Total_Cost
FROM diagnosis d JOIN veterinarian v
WHERE v.first_name LIKE '%Alexander%' AND v.last_name LIKE '%Mûller%'
;
```

diagnosis_id	diagnosis_name	Cost_before_fe...	Required_appointme...	Total_Fees	Total_Cost
1	Strangles	270.00	15	525	795.00
2	Enzootic bovine leucosis	80.00	12	420	500.00
3	Rabies due to rabies virus	75.00	3	105	180.00
4	Influenza A in pigs	165.00	7	245	410.00
5	Avian influenza	325.00	13	455	780.00
6	Screw-worm fly	290.00	12	420	710.00
7	Botulism in poultry	145.00	14	490	635.00
8	Johne disease	405.00	8	280	685.00

- What is the month when more animals get sick (are diagnosed)?
-

The third query determines the month when more animals are diagnosed. It groups the diagnoses by month and counts the number of diagnoses per month. Then it sorts the results by the number of diagnoses in descending order and limits the output to the first row, which is the month with the highest number of diagnoses.

```
SELECT d.diagnosis_id, d.diagnosis_name, COUNT(p.diagnosis_id) AS No_of_animals, ROUND(avg(p.animal_age),1) AS Average_age
FROM pet p LEFT JOIN diagnosis AS d
ON p.diagnosis_id = d.diagnosis_id
WHERE d.diagnosis_name LIKE '%Avian influenza%'
GROUP BY d.diagnosis_id, d.diagnosis_name
;
```

diagnosis_id	diagnosis_name	No_of_animals	Average_age
5	Avian influenza	8	7.9

7. SQL Programming

- Function – takes as input the diagnosis_id and a VAT Rate and returns the final price to be paid

```
DELIMITER //
CREATE FUNCTION diagnosis_final_price(input_diagnosis_id BIGINT, VAT DECIMAL(5,2)) RETURNS DECIMAL(10,2)
BEGIN
    DECLARE v_Final_price DECIMAL(10,2);

    SELECT (Cost_before_fees + (Cost_before_fees) * VAT) * Required_appointments INTO v_Final_price
    FROM Diagnosis
    WHERE diagnosis_id = input_diagnosis_id;
--
    RETURN v_Final_price;
END//
DELIMITER ;
```

```
-- let's try if it works, for input_diagnosis_id = 1 and VAT = 25%
SELECT diagnosis_final_price (1, 0.25);
```

40:71

Result Grid | Filter Rows: Search | Export:

diagnosis_final_price (1, 0....
5062.50

- Procedure – find the number of pets belonging to a single customer.

```
-- With this procedure, we are able to find the number of pets belonging to a single customer
DELIMITER &&
CREATE PROCEDURE Number_Of_Pets (IN vproprietor INT, OUT vpet BIGINT)
BEGIN
  SELECT COUNT(animal_id) INTO vpet
  FROM pet
  WHERE proprietor_id = vproprietor;
END &&
DELIMITER ;

-- Let's check if it works properly. We'll try for animal_id = 1
CALL Number_Of_Pets(1, @result);
SELECT @result AS number_of_pets;
```

Result

number_of_pets
2

- **Trigger**

8. SQL Table Modifications

- **Update statement : update doctor's email**

```
-- UPDATE Doctor's email given first name and last name
UPDATE Veterinarian SET email = "glenden.rance@gmail.com"
WHERE first_name = "Glenden" AND last_name = "Rance";

select * from veterinarian
```

doc_id	first_name	last_name	email	phone_number	appointment_f...
1	Tammy	McKenna	tmckenna0@epa.gov	60258164	25
2	Claybourne	Cooney	ccooney1@quantcast.com	55596988	30
3	Nannette	Iskov	niskov2@toplist.cz	18215063	20
4	Dinah	Oppy	doppy3@deliciousdays.com	81678073	40
5	Deloria	Cockle	dcockle4@mapquest.com	63083333	35
6	Laura	Maasze	lmaasze5@jugem.jp	55862596	28
7	Bernardina	Wyper	bwyper6@stumbleupon.com	93863442	32
8	Glenden	Rance	glenden.rance@gmail.com	18456913	27
9	Sofia	Petrovic	sofia.petrovic@fakemail.com	331762106	39
10	Miroslav	Horvat	miroslav.horvat@fakemail.com	396238244	34
11	Luka	Novak	luka.novak@fakemail.com	93534921	39
12	Anja	Jankovic	anja.jankovic@fakemail.com	31264527	39
13	Alexander	Müller	alexander.müller@fakemail.c...	61460242	35
14	Ingrid	Andersen	ingrid.andersen@fakemail.com	533105305	38
15	Stefan	Weber	stefan.weber@fakemail.com	12531434	33
16	Ana	Popescu	ana.popescu@fakemail.com	30481589	37
17	Lars	Berg	lars.berg@fakemail.com	91864283	33
18	Tatiana	Ivanova	tatiana.ivanova@fakemail.com	89369532	27
19	Marco	Rossi	marco.rossi@fakemail.com	37449861	33
20	Magdalena	Nowak	magdalena.nowak@fakemail...	11732413	40
NULL	NULL	NULL	NULL	NULL	NULL

- **TRIGGER with error handling**

```sql
-- TRIGGER
/* DO NOT PERMIT TO DELETE ENTRIES THAT ARE NOT OLDER THAN 2 YEARS */

DROP TRIGGER IF EXISTS delete_old_diagnoses;

DELIMITER //

CREATE TRIGGER delete_old_diagnoses
BEFORE DELETE ON Animal_disease_diagnosis
FOR EACH ROW
BEGIN
  IF OLD.diagnosis_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 2 YEAR) THEN
      signal sqlstate 'HY000'
      SET mysql_errno = 6666,
      message_text = 'You can delete only entries older than 2 year';
  END IF;
END//

DELIMITER ;

-- Test the trigger
delete from animal_disease_diagnosis
where animal_id = 16;
```

100% 61:93

Action Output

	Time	Action	Response
43	16:39:43	USE VETCLINIC2	0 row(s) affected
44	16:39:43	delete from animal_disease_diagnosis where animal_id = 16	Error Code: 6666. You can delete only entries older than 2 year

- **DELETE STATEMENT**

```sql
-- DELETE STATEMENT
delete from animal_disease_diagnosis
where animal_id = 9;

select * from animal_disease_diagnosis
```

Action Output

	Time	Action	Response
45	16:46:28	select * from animal_disease_diagnosis LIMIT 0, 1000	20 row(s) returned
46	16:46:41	delete from animal_disease_diagnosis where animal_id = 9	1 row(s) affected
47	16:47:10	select * from animal_disease_diagnosis LIMIT 0, 1000	19 row(s) returned