UNIVERSITÀ DEGLI STUDI DI PADOVA

Università di Padova

Dipartimento di Matematica

Relazione per il progetto di Basi di Dati

Alberto Adami-1031350

Indice

1 Abstract

Si vuole realizzare una base di dati per modellare la gestione di uno studio veterinario; in particolare si vogliono gestire le visite effettuate nello studio veterinario e quindi i veterinari, i clienti e gli animali coinvolti in esse.
Le operazioni tipiche sono la registrazione di nuovi animali e la prenotazione di nuove visite per gli animali già registrati presso lo studio.
Lo studio è stato chiamato Zoo Planet.

2 Analisi dei requisiti

Il progetto modella alcune classi coinvolte nella gestione delle visite svolte o che sono in programma di essere svolte in uno studio veterinario.
È stata realizzata un' interfaccia web attraverso la quale i clienti possono interagire con la base di dati.
L'entità principale che si vuole gestire sono le Visite.
Di ogni visita interessano la data e ora in cui è stata effettuata(o sarà effettuata), l'animale che è stato visitato, il tipo di visita(controllo, vaccino, ecc..) e i veterinari che hanno effettuato la visita.
Il costo della visita e il numero dei veterinari necessari sono determinati dal tipo della visita effettuata.
Di un cliente invece interessano il nome, cognome, data di nascita, sesso, città di residenza, telefono, e-mail e gli animali che sono stati registrati presso lo studio.
Ogni Cliente ha inoltre un account associato caratterizzato da username e password necessario per accedere al sito web.
Di ogni veterinario invece interessano il nome, cognome, data di nascita, sesso, città di residenza, telefono, data di assunzione e lo stipendio mensile.
Inoltre di ogni veterinario interessano le visite nelle quali sono stati coinvolti e i tipi di visita che possono eseguire(controllo, interventi, ecc..).
Degli animali interessano nome, sesso, data di nascita(anche presunta), il peso, la razza, il proprietario, il tipo dell'animale(gatto,

cane, ecc..) ed il colore del pelo e degli occhi. Inoltre di ogni animale interessano anche le visite effettuate in precedenza o in programma. Ad ogni animale è associata una scheda clinica della quale interessano il numero di scheda che la identifica, la data di creazione, se l'animale è stato sverminato, vaccinato e sterilizzato ed eventuali annotazioni sull'animale.

3 Progettazione Concettuale

3.1 Descrizione delle classi

Persone: modella una generica persona all'interno dello studio.

- Nome: string
- Cognome: string
- Data di nascita: date
- Città di residenza: string
- Sesso: char
- Numero di telefono: string

Sono state definite le seguenti sottoclassi con attributi propri:

-**Clienti**: modella un cliente del nostro studio veterinario.

- E-mail: string

-**Veterinari**: modella un veterinario che lavora presso lo studio.

- Stipendio mensile: double
- Data di assunzione: date

Account: modella un account posseduto da un cliente dello studio.

- Username: string
- Password: string

Animali: modella gli animali che sono stati registrati presso lo studio.

- Nome: string
- Sesso: string
- Data di nascita: date
- Razza: string
- Tipo dell'animale: string
- Peso: double
- Colore del pelo: string
- Colore degli occhi:string

Schede Cliniche: modella le schede cliniche degli animali registrati presso lo studio.

- Numero di scheda: integer

- Sverminato: boolean
- Vaccinato: bolean
- Sterilizzato: boolean
- Annotazioni: string

Tipo Visita: questa classe rappresenta i vari tipi di visita che lo studio fornisce per gli animali.

- Nome: string
- Prezzo: double
- Veterinari necessari: integer

Visite: modella una visita che è stata effettuata oppure è in programma di essere effettuata presso lo studio.

- Data visita: date
- Orario visita: time

3.2 Descrizione delle associazioni

Clienti - Account: possiede(1:1)

Ogni Cliente possiede un account e un account è posseduto da un unico Cliente. L'associazione è totale verso entrambi i lati.

Animali - SchedeCliniche: ha(1:1)

Ogni animale ha una scheda clinica associata e viceversa ogni scheda clinica riguarda un singolo animale. L'associazione è totale verso entrambi i lati.

Clienti - Animali: è padrone(1:M)

Ogni Cliente può essere padrone di più animali, un animale invece ha un unico padrone.
La associazione è parziale da Cliente ad Animali in quanto un Cliente può non aver registrato ancora alcun animale presso lo studio.
Da animali a clienti l'associazione è invece totale in quanto ogni animale ha sempre un padrone.

Visite - Animali: Visitato(M:1)

Ogni animale può esser stato sottoposto a più visite, invece in una visita può esser stato visitato un solo animale.
L'associazione è parziale da Animali verso Visite in quanto può esser stato registrato un animale presso lo studio veterinario, ma l'animale può non aver ancora fatto alcuna visita.
Invece l'associazione è totale nell'altro verso, in una visita è stato visitato esattamente un animale.

Visite - Veterinari: VisiteVeterinari(M:N)

Ogni veterinario può aver fatto più visite e una visita può esser stata fatta da più veterinari.
L'associazione è parziale da Veterinari a Visite in quanto può accadere che un Veterinario sia stato appena assunto e che quindi non abbia ancora fatto alcuna visita.
Dall'altro verso invece l'associazione è totale in quanto una visita è stata fatta da almeno un veterinario.

Visite - TipoVisite: Ha tipo(M:1)

Ogni visita può avere un unico TipoVisita, ma viceversa di un TipoVisita possono essere state fatte più visite.
L' associazione è totale da Visite a TipoVisite in quanto ogni Visita deve avere necessariamente un TipoVisita e parziale dall'altro lato in quanto possono esistere dei tipi visita dei quali non sono state fatte ancora visite.

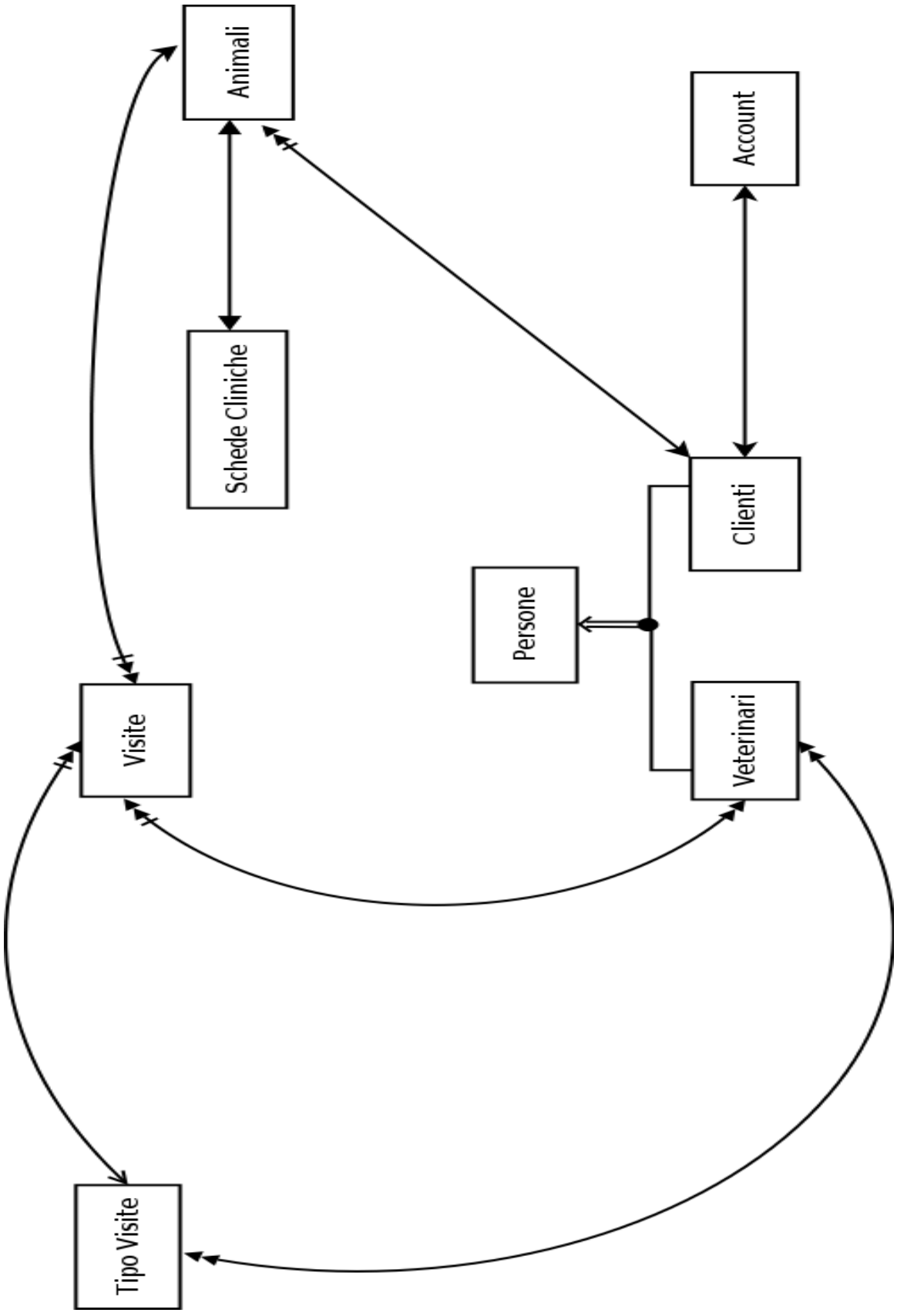
Veterinari - TipoVisite: Specializzazione(M:N)

Ogni veterinario può essere specializzato in più tipi di visita e ci possono essere più veterinari specializzati nello stesso tipo visita.
L'associazione è totale verso entrambi i lati.

3.3 Descrizione delle gerarchie

Le sottoclassi di Persone(Clienti e Veterinari) hanno il vincolo di copertura e disgiunzione in quanto un cliente non può essere anche un veterinario e lo stesso vale per il contrario.

3.4 Schema concettuale in forma grafica



4 Progettazione Logica

4.1 Descrizione della gerarchia di persone

È stato scelto di implementare la gerarchia di persone tramite il partizionamento verticale, in quanto le sottoclassi hanno attributi propri.
La gerarchia completa è quindi composta di tre classi:
Persone: è stata creata una chiave primaria Idpersona apposita per identificare una persona.

- Idpersona <<PK>>
- Nome
- Cognome
- Sesso
- Data di nascita
- Città di residenza
- Telefono

Clienti: Idcliente è una chiave esterna che riferisce la chiave primaria di Persone(Idpersona) ed ha anche il ruolo chiave primaria.

- Idcliente <<PK>><<FK(Persone)>>
- E-mail

Veterinari: Idveterinario è una chiave esterna che riferisce la chiave primaria di Persone(Idpersona) è fa anche da chiave primaria.

- Idveterinario <<PK>><<FK(Persone)>>
- Stipendio
- Data assunzione

4.2 Descrizione dello schema relazionale

Per le tabelle che non avevano una chiave primaria tra i loro attributi, è stata aggiunta una chiave primaria sintetica Id di tipo AUTO_INCREMENT in modo tale che la chiave primaria venga assegnata automaticamente.
Le tabelle Specializzazioni e VisiteVeterinari servono per rappresentare le relazioni M:N rispettivamente tra Veterinari-TipoVisite e Veterinari-Visite.
Persone(Idpersona, Nome, Cognome , Sesso ,Data nascita, Città, Tel)

- PK(Idpersona)

Clienti(Idcliente,e-mail)

- PK(Idcliente)

- Idcliente FK Persone(Idpersona)

Veterinari(Idveterinario, Stipendio, Data assunzione)
- PK(Idveterinario)
- Idveterinario FK Persone(Idpersona)

Account(Idcliente, Username, Password)
- PK Username
- Idcliente FK Clienti(Idcliente)

Animali(Idanimale, Nome, Sesso, Peso, Razza, Tipoanimale, Coloreocchi, Colorepelo, Datanascita, Padrone)
- PK Idanimale
- Padrone FK Clienti(Idcliente)

SchedeCliniche(Numeroscheda, Idanimale, Sterilizzato, Vaccinato, Sverminato, Annotazioni)
- PK Numeroscheda
- Idanimale FK Animali(Idanimale)

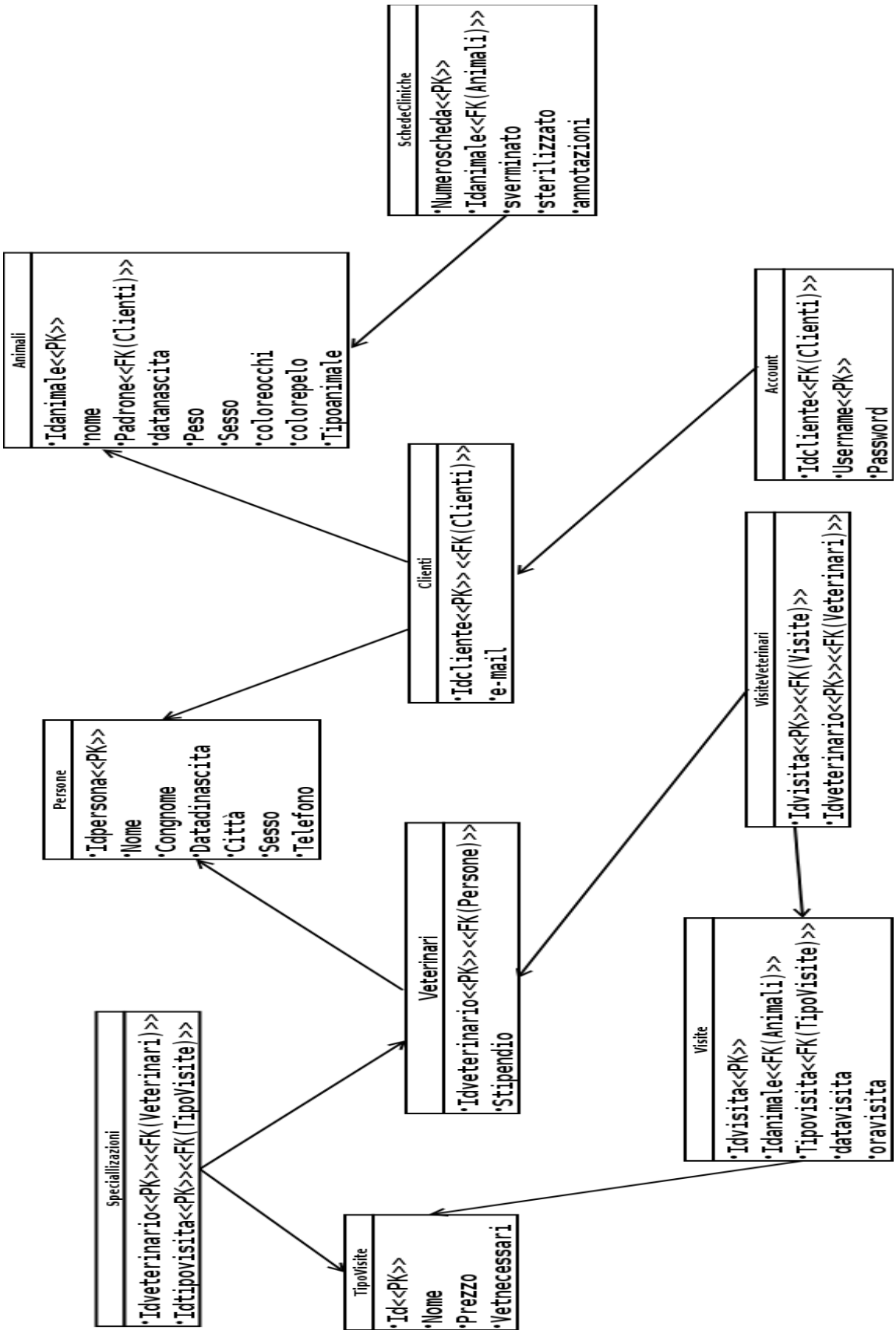
TipoVisite(Id, Nome, Costo,VetNecessari)
- PK Id

Visite(Idvisita, Idanimale, Tipovisita, Orario, Data)
- PK Idvisita
- Idanimale FK Animali(Idanimale)
- Tipovisita FK TipoVisite(Id)

VisiteVeterinari(Idvisita,Idveterinario)
- PK(Idvisita,Idveterinario)
- Idvisita FK Visite(Idvisita)
- Idveterinario FK Veterinari(Idveterinario)

Specializzazioni(Idveterinario, Idtipovisita)
- PK(Idveterinario,Idtipovisita)
- Idveterinario FK Veterinari(Idveterinario)
- Idtipovisita FK TipoVisite(Id)

4.3 Schema relazionale in forma grafica



5 Implementazione della base di dati con il DDL

Nell'implementazione della base di dati con il DDL di MySql è stata creata una tabella aggiuntiva errori che viene usata dai Trigger passivi per generare errori su vincoli di integrità non direttamente esprimibili tramite in DDL.

```
/*
Abilito gli eventi.
*/
SET GLOBAL event_scheduler = ON;
/*
Elimino le tabelle, qualora esistessero già.
*/
DROP TABLE IF EXISTS SchedeCliniche;
DROP TABLE IF EXISTS VisiteVeterinari;
DROP TABLE IF EXISTS Visite;
DROP TABLE IF EXISTS Animali;
DROP TABLE IF EXISTS Specializzazioni;
DROP TABLE IF EXISTS TipoVisite;
DROP TABLE IF EXISTS Veterinari;
DROP TABLE IF EXISTS Account;
DROP TABLE IF EXISTS Clienti;
DROP TABLE IF EXISTS Persone;
DROP TABLE IF EXISTS Errori;
/*
Elimino le viste se esistono.
*/
DROP VIEW IF EXISTS clienticosto;
DROP VIEW IF EXISTS numvisite;
/*
Elimino i trigger e gli eventise esistono.
*/
DROP EVENT IF EXISTS Aggiornascheda;
DROP TRIGGER IF EXISTS CreateScheda;
DROP TRIGGER IF EXISTS Aggiornastipendio;
DROP TRIGGER IF EXISTS Newveterinario;
DROP TRIGGER IF EXISTS Assegnavet;
DROP TRIGGER IF EXISTS Visitacorretta;
DROP TRIGGER IF EXISTS Insertanimale;
DROP TRIGGER IF EXISTS Updateanimale;
/*
Elimino le funzioni se esistono.
*/
DROP FUNCTION IF EXISTS VerificaDisponibile;
DROP FUNCTION IF EXISTS VetDisponibili;
DROP FUNCTION IF EXISTS Haspecializzazione;
DROP FUNCTION IF EXISTS Nuovesverm;
DROP FUNCTION IF EXISTS Nuovevacc;
DROP FUNCTION IF EXISTS Nuovester;
```

```
/*
Creo la tabella delle Persone.
*/
CREATE TABLE Persone (
   Idpersona INTEGER PRIMARY KEY AUTO_INCREMENT,
   Nome VARCHAR(15) NOT NULL,
   Cognome VARCHAR(15) NOT NULL,
   DataNasc DATE NOT NULL,
   Citta VARCHAR(20) NOT NULL,
   Sesso CHAR(1) NOT NULL CHECK (Sesso IN ('M' , 'F')),
   Telefono VARCHAR(10) NOT NULL UNIQUE
)  ENGINE=InnoDB;
/*
Creo la tabella dei Clienti.
*/
CREATE TABLE Clienti (
   Idcliente INTEGER PRIMARY KEY,
   email VARCHAR(32) NOT NULL UNIQUE,
   FOREIGN KEY (Idcliente)
      REFERENCES Persone (IdPersona)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
/*
Creo la tabella degli Account.
*/
CREATE TABLE Account (
   Idcliente INTEGER NOT NULL UNIQUE,
   Username VARCHAR(20) PRIMARY KEY,
   Password CHAR(8) NOT NULL,
   FOREIGN KEY (Idcliente)
      REFERENCES Clienti (Idcliente)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
/*
Creo la tabella dei Veterinari.
*/
CREATE TABLE Veterinari (
   Idveterinario INTEGER PRIMARY KEY,
   Stipendio DOUBLE NOT NULL CHECK (Stipendio > 0),
   Dataassunzione DATE NOT NULL,
   FOREIGN KEY (Idveterinario)
      REFERENCES Persone (IdPersona)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
/*
Creo la tabella degli Animali.
*/
CREATE TABLE Animali (
   Idanimale INTEGER PRIMARY KEY AUTO_INCREMENT,
   Padrone INTEGER NOT NULL,
   Nome VARCHAR(15) NOT NULL,
```

```
  Peso DOUBLE NOT NULL CHECK (Peso > 0),
   Coloreocchi VARCHAR(20) NOT NULL,
   Colorepelo VARCHAR(20) NOT NULL,
   Razza VARCHAR(20) NOT NULL,
   Tipoanimale VARCHAR(30) NOT NULL,
         Datanasc DATE NOT NULL,
   Sesso CHAR(1) NOT NULL CHECK (Sesso IN ('M' , 'F')),
   FOREIGN KEY (Padrone)
      REFERENCES Clienti (Idcliente)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
/*
Creo la tabella delle Schede Cliniche.
*/
CREATE TABLE SchedeCliniche (
   Idscheda INTEGER PRIMARY KEY AUTO_INCREMENT,
   Idanimale INTEGER NOT NULL UNIQUE,
   Datacreazione DATE NOT NULL,
   Sverminato BOOLEAN DEFAULT False,
   Vaccinato BOOLEAN DEFAULT False,
   Sterilizzato BOOLEAN DEFAULT FALSE,
   Annotazioni VARCHAR(255) NOT NULL DEFAULT '',
   FOREIGN KEY (Idanimale)
      REFERENCES Animali (Idanimale)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
/*
Creo la tabella TipoVisita.
*/
CREATE TABLE TipoVisite (
   Id INTEGER AUTO_INCREMENT PRIMARY KEY,
   Nome VARCHAR(25) UNIQUE NOT NULL,
   Costo DOUBLE NOT NULL CHECK (Costo > 0),
         Vetnecessari INTEGER NOT NULL CHECK(Vetnecessari>0)
)  ENGINE=InnoDB;
/*
Creo la tabella delle visite.
*/
CREATE TABLE Visite (
   Idvisita INTEGER PRIMARY KEY AUTO_INCREMENT,
   Tipovisita INTEGER NOT NULL,
   Datavisita DATE NOT NULL,
   Oravisita TIME NOT NULL,
   Animale INTEGER NOT NULL,
   FOREIGN KEY (Tipovisita)
      REFERENCES TipoVisite (Id)
      ON DELETE CASCADE,
   FOREIGN KEY (Animale)
      REFERENCES Animali (idAnimale)
      ON DELETE CASCADE
)  ENGINE=InnoDB;
```

```
/*
Creo la tabella VisiteVeterinari.
*/
CREATE TABLE VisiteVeterinari (
   Idvisita INTEGER,
   Idveterinario INTEGER,
   PRIMARY KEY (Idveterinario ,Idvisita),
   FOREIGN KEY (Idveterinario) REFERENCES Veterinari (Idveterinario)
      ON DELETE CASCADE,
   FOREIGN KEY (Idvisita)
      REFERENCES Visite (Idvisita)
      ON DELETE CASCADE
)ENGINE=InnoDB;
/*
Creo la tabella degli errori.
*/
CREATE TABLE Errori (
   Numerrore INTEGER PRIMARY KEY,
   Descrizione VARCHAR(50)
)  ENGINE=InnoDB;
/*
Creo le viste.
*/
/*
Creo la vista Numvisite
*/
CREATE VIEW Numvisite AS
        SELECT v.Idveterinario,p.Nome,p.Cognome,COUNT(*) AS Num
        FROM Persone p JOIN Veterinari v ON(p.Idpersona=v.Idveterinario) NATURAL JOIN
VisiteVeterinari vv NATURAL JOIN Visite vis
        WHERE vis.Datavisita<CURDATE()
        GROUP BY v.Idveterinario,p.Nome,p.Cognome
UNION
        SELECT v.Idveterinario,p.Nome,p.Cognome,0 AS Num
        FROM Persone p JOIN Veterinari v ON(p.Idpersona=v.Idveterinario)
        WHERE v.Idveterinario NOT IN(SELECT DISTINCT vv.Idveterinario
                                            FROM VisiteVeterinari vv)
        OR v.Idveterinario NOT IN(SELECT DISTINCT vv.Idveterinario
                                     FROM Visite v NATURAL JOIN VisiteVeterinari vv
                                    WHERE v.Datavisita<CURDATE());
/*
Creo la vista ClientiCosto
*/
CREATE VIEW ClientiCosto AS
        SELECT c.Idcliente,p.Nome,p.Cognome,c.email,SUM(tv.costo) AS CostoTot
        FROM (TipoVisite tv JOIN Visite v ON(tv.Id=v.Tipovisita) JOIN Animali a
ON(v.animale=a.Idanimale)
                JOIN Clienti c ON(a.Padrone=c.Idcliente) JOIN Persone p
ON(p.Idpersona=c.Idcliente))
        WHERE v.Datavisita<CURDATE()
        GROUP BY c.Idcliente
UNION
```

```
SELECT DISTINCT c.Idcliente,p.Nome,p.Cognome,c.email,0 AS CostoTot
        FROM Clienti c JOIN Persone p ON(c.Idcliente=p.Idpersona)
        WHERE c.Idcliente NOT IN(SELECT DISTINCT a.Padrone
                                                FROM Animali a)
                OR NOT EXISTS(SELECT *
                                FROM Animali a
                                WHERE a.Padrone=c.Idcliente AND
                                a.Idanimale IN(SELECT v.Animale
                                                FROM Visite v));
/*
Imposto il delimiter per trigger, eventi e funzioni.
*/
DELIMITER $
/*
Creo le funzioni.
*/
/*
Creo la funzione VerificaDisponibile.
*/
CREATE FUNCTION VerificaDisponibile(Vet INTEGER,datav DATE,orav TIME) RETURNS BOOLEAN
BEGIN
        DECLARE num INTEGER;
        SELECT COUNT(*) INTO num
        FROM Veterinari v
        WHERE v.Idveterinario=Vet AND (Idveterinario NOT IN (SELECT vv.Idveterinario

FROM VisiteVeterinari vv)
                OR v.Idveterinario NOT IN(SELECT vv.Idveterinario
                                        FROM VisiteVeterinari vv NATURAL JOIN Visite vi
                                        WHERE vi.Datavisita=datav   ANDvi.Oravisita=orav));
        IF(num!=0)
                THEN
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
        END IF;
END$
/*
Creo la funzione VetDisponibili.
*/
CREATE FUNCTION VetDisponibili(Tipov INTEGER,datav DATE,timev TIME) RETURNS INTEGER
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Veterinari v
        WHERE v.Idveterinario NOT IN(SELECT vv.Idveterinario
                                                        FROM VisiteVeterinari vv)
                OR v.Idveterinario NOT IN(SELECT vv1.Idveterinario
                                                        FROM VisiteVeterinari vv1
NATURAL JOIN Visite vi
                                                        WHERE NOT(vi.Datavisita=datav
AND vi.Oravisita=timev));
        RETURN conta;
END$
```

```
/*
Creo la funzione Haspecializzazione.
*/
CREATE FUNCTION Haspecializzazione(Vet INTEGER,Idtipo INTEGER) RETURNS BOOLEAN
BEGIN
	DECLARE conta INTEGER;
	SELECT COUNT(*) INTO conta
	FROM Specializzazioni s
	WHERE s.Idveterinario=Vet AND s.Idtipovisita=Idtipo;
	IF conta>0
		THEN
		RETURN TRUE;
	ELSE
		RETURN FALSE;
	END IF;
END$
/*
Creo la funzione Nuovesverm.
*/
CREATE FUNCTION Nuovesverm() RETURNS BOOLEAN
BEGIN
	DECLARE conta INTEGER;
	SELECT COUNT(*) INTO conta
	FROM Visite v
	WHERE v.Datavisita<CURDATE() AND v.Tipovisita=3
		AND v.Animale IN(SELECT s.Idanimale
					FROM SchedeCliniche s
					WHERE s.Sverminato=FALSE);
	IF(conta!=0)
		THEN
			RETURN TRUE;
		ELSE
			RETURN FALSE;
	END IF;
END$
/*
Creo la funzione Nuovevacc.
*/
CREATE FUNCTION Nuovevacc() RETURNS BOOLEAN
BEGIN
	DECLARE conta INTEGER;
	SELECT COUNT(*) INTO conta
	FROM Visite v
	WHERE v.Datavisita<CURDATE() AND v.Tipovisita=1
		AND v.Animale IN(SELECT s.Idanimale
					FROM SchedeCliniche s
					WHERE s.Vaccinato=FALSE);
	IF(conta!=0)
		THEN
			RETURN TRUE;
		ELSE
			RETURN FALSE;
	END IF;
END$
```

```
/*
Creo la funzione Nuovester.
*/
CREATE FUNCTION Nuovester() RETURNS BOOLEAN
BEGIN
	DECLARE conta INTEGER;
	SELECT COUNT(*) INTO conta
	FROM Visite v
	WHERE v.Datavisita<CURDATE() AND v.Tipovisita=2
		AND v.Animale IN(SELECT s.Idanimale
					FROM SchedeCliniche s
					WHERE s.Sterilizzato=FALSE);
	IF(conta!=0)
		THEN
			RETURN TRUE;
		ELSE
			RETURN FALSE;
	END IF;
END$
/*
Creo i trigger e gli eventi.
*/
/*
Creo il trigger CreateScheda.
*/
CREATE TRIGGER CreateScheda
AFTER INSERT ON Animali
FOR EACH ROW
BEGIN
	INSERT INTO SchedeCliniche(Idanimale,
Datacreazione,Sverminato,Vaccinato,Sterilizzato,Annotazioni)
	VALUES(NEW.Idanimale,CURDATE(),FALSE,FALSE,FALSE,"");
END$
/*
Creo il trigger Aggiornastipendio.
*/
CREATE TRIGGER Aggiornastipendio
BEFORE UPDATE ON Veterinari
FOR EACH ROW
BEGIN
	IF(NEW.Stipendio<=0)
		THEN
			INSERT INTO Errori(Numerrore) VALUES(2);
	END IF;
END$
/*
Creo il trigger Newveterinario.
*/
CREATE TRIGGER Newveterinario
BEFORE INSERT ON Veterinari
FOR EACH ROW
BEGIN
	IF(NEW.Stipendio<=0)
```

```
THEN
                        INSERT INTO Errori(Numerrore) VALUES (2);
        END IF;
END$
/*
Creo il trigger Assegnavet.
*/
CREATE TRIGGER Assegnavet
AFTER INSERT ON Visite
FOR EACH ROW
BEGIN
        DECLARE Nec INTEGER;
        SELECT tv.Vetnecessari INTO Nec
        FROM Tipovisite tv
        WHERE tv.Id=NEW.Tipovisita;
        IF (VetDisponibili(NEW.Tipovisita,NEW.Datavisita,NEW.Oravisita))>=Nec
        THEN
        IF Nec=1
        THEN
                INSERT INTO VisiteVeterinari
                SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
                FROM Numvisite nv
                WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
                                AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
                ORDER BY nv.Num ASC
                LIMIT 1;
        ELSEIF Nec=2
        THEN
                INSERT INTO VisiteVeterinari
                SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
                FROM Numvisite nv
                WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
                                AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
                ORDER BY nv.Num ASC
                LIMIT 2;
        ELSEIF Nec=3
        THEN
        INSERT INTO VisiteVeterinari
        SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
        FROM Numvisite nv
        WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
                                AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
        ORDER BY nv.Num ASC
        LIMIT 3;
        END IF;
        ELSE
                DELETE FROM Visite
                WHERE Idvisita=NEW.Idvisita;
        END IF;
END$
/*
Creo il triggetr Visitacorretta.
*/
```

```
CREATE TRIGGER Visitacorretta
BEFORE INSERT ON Visite
FOR EACH ROW
BEGIN
	DECLARE conta INTEGER;
	DECLARE sver INTEGER;
	DECLARE vacc INTEGER;
	DECLARE ster INTEGER;
	SELECT COUNT(*) INTO sver
	FROM SchedeCliniche s
	WHERE s.Idanimale=NEW.Animale AND s.Sverminato=TRUE;
	SELECT COUNT(*) INTO vacc
	FROM SchedeCliniche s
	WHERE s.Idanimale=NEW.Animale AND s.Vaccinato=TRUE;
	SELECT COUNT(*) INTO ster
	FROM SchedeCliniche s
	WHERE s.Idanimale=NEW.Animale AND s.Sterilizzato=TRUE;
	SELECT COUNT(*) INTO conta
	FROM Visite v
	WHERE v.Animale=NEW.Animale AND v.Tipovisita=NEW.Tipovisita;
	IF((NEW.Tipovisita=1) AND(vacc>0 OR conta>0))
		OR (NEW.Tipovisita=2 AND (ster>0 OR conta>0))
		OR ((NEW.Tipovisita=3 AND (sver>0 OR conta>0)))
	THEN
		INSERT INTO Errori(Numerrore) VALUES(5);
	END IF;
END$
/*
Creo il trigger Insertanimale.
*/
CREATE TRIGGER Insertanimale
BEFORE INSERT ON Animali
FOR EACH ROW
BEGIN
	IF(NEW.Peso<=0)
		THEN
			INSERT INTO Errori(Numerrori) VALUES(4);
		END IF;
END$
/*
Creo il trigger Updateanimale.
*/
CREATE TRIGGER Updateanimale
BEFORE UPDATE ON Animali
FOR EACH ROW
BEGIN
	IF(NEW.Peso<=0)
		THEN
			INSERT INTO Errori(Numerrori) VALUES(4);
		END IF;
END$
```

```
/*
Creo l'evento Aggiornascheda.
*/
CREATE EVENT Aggiornascheda
ON SCHEDULE  EVERY 1 SECOND
DO
BEGIN
	IF Nuovesverm()
	THEN
		UPDATE SchedeCliniche
		SET Sverminato=TRUE
		WHERE Idanimale IN(SELECT v.Animale
					FROM Visite v
					 WHERE v.Datavisita<CURDATE() AND v.Tipovisita=3)
						AND Sverminato=FALSE;
	END IF;
	IF Nuovester()
	THEN
		UPDATE SchedeCliniche
		SET Sterilizzato=TRUE
		WHERE Idanimale IN(SELECT v.Animale
					FROM Visite v
					WHERE v.Datavisita<CURDATE() AND v.Tipovisita=2)
					AND Sterilizzato=FALSE;
	END IF;
	IF Nuovevacc()
	THEN
		UPDATE SchedeCliniche
		SET Vaccinato=TRUE
		WHERE Idanimale IN(SELECT v.Animale
					FROM Visite v
					WHERE v.Datavisita<CURDATE() AND v.Tipovisita=1)
						AND Vaccinato=FALSE;
	END IF;
END$
/*
Reimposto il delimiter
*/
DELIMITER ;
```

Nota: nel trigger Assegnavet è stato necessario distinguere i vari casi poiché la versione mysql presente in laboratorio non permette di scrivere un istruzione come LIMIT Nec.

6 Query, Trigger, Viste, Eventi e Procedure

6.1 Query e Viste

Seguono le query più significative sulla base di dati, alcune di esse fanno uso di view.

1. Vogliamo trovare i veterinari che hanno effettuato un numero di visite superiore alla media.

```
CREATE VIEW Numvisite AS
        SELECT v.Idveterinario,p.Nome,p.Cognome,COUNT(vv.Idvisita) AS Num
        FROM Persone p JOIN Veterinari v ON(p.Idpersona=v.Idveterinario) NATURAL JOIN
VisiteVeterinari vv NATURAL JOIN Visite vis
        WHERE vis.Datavisita<CURDATE()
        GROUP BY v.Idveterinario,p.Nome,p.Cognome
UNION
        SELECT v.Idveterinario,p.Nome,p.Cognome,0 AS Num
        FROM Persone p JOIN Veterinari v ON(p.Idpersona=v.Idveterinario)
        WHERE v.Idveterinario NOT IN(SELECT DISTINCT vv.Idveterinario
                                        FROM VisiteVeterinari vv)
        OR v.Idveterinario NOT IN(SELECT DISTINCT vv.Idveterinario
                                    FROM Visite v NATURAL JOIN VisiteVeterinari vv
                                    WHERE v.Datavisita<CURDATE());
SELECT *
FROM Numvisite n
WHERE n.Num>(SELECT AVG(n1.Num)
                    FROM Numvisite n1)
ORDER BY n.Num DESC;
```

Output:

Idveterinario	Nome	Cognome	Num
2	Mario	Rossi	5
48	Beatrice	Girti	5
60	Lucia	Vetri	5
34	Luigi	Marchesini	5
3	Beatrice	Bianchi	4
49	Marco	Montagner	4
58	Morris	Tessariol	4
31	Giorgio	Sereni	4
52	Anna	Bersi	4
41	Tiziana	Forte	4
59	Antonio	Mazzobel	4
39	Lucia	Berini	3
56	Tiziana	Antico	3
47	Luigi	Rossani	3

14 rows in set (0,01 sec)

2. Vogliamo trovare i clienti che hanno speso il massimo per i loro animali.

```
CREATE VIEW ClientiCosto AS
        SELECT c.Idcliente,p.Nome,p.Cognome,c.email,SUM(tv.costo) AS CostoTot
        FROM (TipoVisite tv JOIN Visite v ON(tv.Id=v.Tipovisita) JOIN Animali a
ON(v.animale=a.Idanimale)
                JOIN Clienti c ON(a.Padrone=c.Idcliente) JOIN Persone p
ON(p.Idpersona=c.Idcliente))
        WHERE v.Datavisita<CURDATE()
        GROUP BY c.Idcliente
UNION
        SELECT DISTINCT c.Idcliente,p.Nome,p.Cognome,c.email,0 AS CostoTot
        FROM Clienti c JOIN Persone p ON(c.Idcliente=p.Idpersona)
        WHERE c.Idcliente NOT IN(SELECT DISTINCT a.Padrone
                                                FROM Animali a)
                OR NOT EXISTS(SELECT *
                                FROM Animali a
                                WHERE a.Padrone=c.Idcliente AND
                                        a.Idanimale IN(SELECT v.Animale
                                                        FROM Visite v));
SELECT *
FROM ClientiCosto cc
WHERE cc.CostoTot=(SELECT MAX(cc1.CostoTot)
                        FROM ClientiCosto cc1);
```

Output:

```
+-----------+-------+-----------+---------------------+-----------+
| Idcliente | Nome  | Cognome   | email               | CostoTot  |
+-----------+-------+-----------+---------------------+-----------+
|         4 | Marco | Marchetti | marchetti@live.com  |      1445 |
+-----------+-------+-----------+---------------------+-----------+
1 row in set (0,01 sec)
```

3. Vogliamo trovare i veterinari della città di Padova che nell'anno corrente hanno effettuato almeno due visite nel giorno del loro compleanno.

```
SELECT v.Idveterinario,p.Nome,p.Cognome,COUNT(*) AS Numvisite
FROM Persone p JOIN Veterinari v ON(p.Idpersona=v.Idveterinario) NATURAL
JOIN VisiteVeterinari vv NATURAL JOIN Visite vi
WHERE p.Citta='Padova' AND (MONTH(p.Datanasc)=MONTH(vi.Datavisita)) AND
(DAY(P.Datanasc)=DAY(vi.Datavisita)) AND
(YEAR(vi.Datavisita)=YEAR(CURDATE())) AND vi.Datavisita<CURDATE()
GROUP BY v.Idveterinario,p.Nome,p.Cognome
HAVING COUNT(*)>=2;
```

Output:

```
+---------------+-------+---------+-----+
| Idveterinario | Nome  | Cognome | num |
+---------------+-------+---------+-----+
|             2 | Mario | Rossi   |   2 |
+---------------+-------+---------+-----+
1 row in set (0,00 sec)
```

4. Vogliamo trovare le cinque razze di animali più diffuse che hanno già effettuato almeno due visite.

```
SELECT a.Razza, COUNT(*) AS Numanimali
FROM Animali a
WHERE a.Idanimale IN(SELECT a1.Idanimale
                     FROM Animali a1 JOIN Visite v ON(a1.Idanimale=v.Animale)
                     WHERE v.Datavisita<CURDATE()
                     GROUP BY a1.Idanimale
                     HAVING COUNT(*)>=2)
GROUP BY a.Razza
ORDER BY COUNT(*) DESC
LIMIT 5;
```

Output:

```
+-------------------+------------+
| Razza             | Numanimali |
+-------------------+------------+
| Meticcio          |          3 |
| Terranova         |          1 |
| Persiano          |          1 |
| British Shortair  |          1 |
| Sacro di birmania |          1 |
+-------------------+------------+
5 rows in set (0,00 sec)
```

5. Vogliamo i clienti che possiedono animali che hanno effettuato solo visite con un costo superiore ai 20 euro.

```
SELECT c.Idcliente,p.Nome,p.Cognome,c.email
FROM Clienti c JOIN Persone p ON(c.Idcliente=p.Idpersona)
WHERE NOT EXISTS(SELECT *
                 FROM TipoVisite tv JOIN Visite v ON(tv.Id=v.Tipovisita)
                 JOIN Animali a ON(a.Idanimale=v.Animale)
                 WHERE a.Padrone=c.Idcliente AND tv.Costo<=20 AND
v.Datavisita<CURDATE()  AND c.Idcliente IN(SELECT a.Padrone
                                           FROM Animali a);
```

Output:

```
+-----------+-------+---------+--------------------+
| Idcliente | Nome  | Cognome | email              |
+-----------+-------+---------+--------------------+
|        30 | Luisa | Verdi   | verdiluisa@live.com |
+-----------+-------+---------+--------------------+
1 row in set (0,00 sec)
```

6. Vogliamo trovare i clienti che possiedono solo gatti o solo cani. La terza condizione della query è necessaria, altrimenti verranno considerati anche i Clienti che non hanno ancora registrato alcun animale presso lo studio.

```
SELECT p.Idpersona,p.Nome,p.Cognome,c.email
FROM Persone p JOIN Clienti c ON(p.Idpersona=c.Idcliente)
WHERE c.Idcliente IN(SELECT DISTINCT(a.Padrone)
                     FROM Animali a)
      AND(c.Idcliente NOT IN(SELECT DISTINCT a.Padrone
                             FROM Animali a
                             WHERE a.Tipoanimale<>'Gatto')
      OR c.Idcliente NOT IN(SELECT DISTINCT a.Padrone
                            FROM Animali a
                            WHERE a.Tipoanimale<>'Cane'));
```

Output:

```
+-----------+---------+-----------+--------------------+
| Idpersona | Nome    | Cognome   | email              |
+-----------+---------+-----------+--------------------+
|         1 | Giorgio | Rossi     | giorgio@gmail.com  |
|         4 | Marco   | Marchetti | marchetti@live.com |
|         6 | Mario   | Berton    | berton1@libero.it  |
|        30 | Luisa   | Verdi     | verdiluisa@live.com |
+-----------+---------+-----------+--------------------+
4 rows in set (0,00 sec)
```

6.1 Trigger ed Eventi

1. Trigger che dopo l'inserimento di un nuovo animale nel database, ne crea la corrispondente scheda clinica.

```
DROP TRIGGER IF EXISTS CreateScheda;
DELIMITER $
CREATE TRIGGER CreateScheda
AFTER INSERT ON Animali
FOR EACH ROW
BEGIN
        INSERT INTO SchedeCliniche(Idanimale,
Datacreazione,Sverminato,Vaccinato,Sterilizzato,Annotazioni)
        VALUES(NEW.Idanimale,CURDATE(),FALSE,FALSE,FALSE,"");
END$
DELIMITER ;
```

2. Trigger che genera un errore se si tenta di modificare lo stipendio di un veterinario con un valore negativo o zero.

```
DROP TRIGGER IF EXISTS Aggiornastipendio;
DELIMITER $
CREATE TRIGGER Aggiornastipendio
BEFORE UPDATE ON Veterinari
FOR EACH ROW
BEGIN
        IF(NEW.Stipendio<=0)
                THEN
                        INSERT INTO Errori(Numerrore) VALUES(2);
        END IF;
END$
DELIMITER ;
```

3. Trigger analogo al precedente, ma in questo caso l'errore viene generato dall'inserimento di un nuovo veterinario.

```
DROP TRIGGER IF EXISTS Newveterinario;
DELIMITER $
CREATE TRIGGER Newveterinario
BEFORE INSERT ON Veterinari
FOR EACH ROW
BEGIN
        IF(NEW.Stipendio<=0)
                THEN
                        INSERT INTO Errori(Numerrore) VALUES (2);
        END IF;
END$
DELIMITER ;
```

4. Trigger che dopo l'inserimento di una nuova visita nel database, assegna i veterinari, e se non ce ne sono un numero adeguato di disponibili, la visita viene cancellata.

```
DROP TRIGGER IF EXISTS Assegnavet;
DELIMITER $
CREATE TRIGGER Assegnavet
AFTER INSERT ON Visite
FOR EACH ROW
BEGIN
	DECLARE Nec INTEGER;
	SELECT tv.Vetnecessari INTO Nec
	FROM Tipovisite tv
	WHERE tv.Id=NEW.Tipovisita;
	IF (VetDisponibili(NEW.Tipovisita,NEW.Datavisita,NEW.Oravisita))>=Nec
	THEN
		IF Nec=1
		THEN
		INSERT INTO VisiteVeterinari
		SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
		FROM Numvisite nv
		WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
			AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
		ORDER BY nv.Num ASC
		LIMIT 1;
		ELSEIF Nec=2
		THEN
		INSERT INTO VisiteVeterinari
		SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
		FROM Numvisite nv
		WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
			AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
		ORDER BY nv.Num ASC
		LIMIT 2;
		ELSEIF Nec=3
		THEN
		INSERT INTO VisiteVeterinari
		SELECT NEW.Idvisita AS Idvisita, nv.Idveterinario AS Idveterinario
		FROM Numvisite nv
		WHERE VerificaDisponibile(nv.Idveterinario,NEW.Datavisita,NEW.Oravisita)
			AND Haspecializzazione(nv.Idveterinario,NEW.Tipovisita)
		ORDER BY nv.Num ASC
		LIMIT 3;
		END IF;
	ELSE
		DELETE FROM Visite
		WHERE Idvisita=NEW.Idvisita;
	END IF;
END$
DELIMITER ;
```

5. Trigger che genera un errore se si prova a inserire una nuova visita di vaccinazione, sverminazione, sterilizzazione se l'animale ne ha già effettuata una.

```
DROP TRIGGER IF EXISTS Visitacorretta;
DELIMITER $
CREATE TRIGGER Visitacorretta
BEFORE INSERT ON Visite
FOR EACH ROW
BEGIN
        DECLARE conta INTEGER;
        DECLARE sver INTEGER;
        DECLARE vacc INTEGER;
        DECLARE ster INTEGER;
        SELECT COUNT(*) INTO sver
        FROM SchedeCliniche s
        WHERE s.Idanimale=NEW.Animale AND s.Sverminato=TRUE;
        SELECT COUNT(*) INTO vacc
        FROM SchedeCliniche s
        WHERE s.Idanimale=NEW.Animale AND s.Vaccinato=TRUE;
        SELECT COUNT(*) INTO ster
        FROM SchedeCliniche s
        WHERE s.Idanimale=NEW.Animale AND s.Sterilizzato=TRUE;
        SELECT COUNT(*) INTO conta
        FROM Visite v
        WHERE v.Animale=NEW.Animale AND v.Tipovisita=NEW.Tipovisita;
        IF((NEW.Tipovisita=1) AND(vacc>0 OR conta>0))
                OR (NEW.Tipovisita=2 AND (ster>0 OR conta>0))
                OR ((NEW.Tipovisita=3 AND (sver>0 OR conta>0)))
        THEN
                INSERT INTO Errori(Numerrore) VALUES(5);
        END IF;
END$
DELIMITER ;
```

6. Trigger che non permette di inserire un animale con un peso minore o uguale a zero.

```
DROP TRIGGER IF EXISTS Insertanimale;
DELIMITER $
CREATE TRIGGER Insertanimale
BEFORE INSERT ON Animali
FOR EACH ROW
BEGIN
        IF(NEW.Peso<=0)
                THEN
                        INSERT INTO Errori(Numerrori) VALUES(4);
                END IF;
END$
```

7. Trigger analogo al precedente, ma in questo caso l'evento catturato è l'inserimento.

```
DROP TRIGGER IF EXISTS Updateanimale;
DELIMITER $
CREATE TRIGGER Updateanimale
BEFORE UPDATE ON Animali
FOR EACH ROW
BEGIN
        IF(NEW.Peso<=0)
                THEN
                        INSERT INTO Errori(Numerrori) VALUES(4);
                END IF;
END$
DELIMITER ;
```

8. Evento che una volta al giorno aggiorna le informazioni delle schede cliniche degli animali.

```
DROP EVENT IF EXISTS Aggiornascheda;
DELIMITER $
CREATE EVENT Aggiornascheda
ON SCHEDULE  EVERY 1 DAY
DO
BEGIN
        IF Nuovesverm()
        THEN
                UPDATE SchedeCliniche
                SET Sverminato=TRUE
                WHERE Idanimale IN(SELECT v.Animale
                                        FROM Visite v
                                        WHERE v.Datavisita<CURDATE() AND v.Tipovisita=3);
        END IF;
        IF Nuovester()
        THEN
                UPDATE SchedeCliniche
                SET Sterilizzato=TRUE
                WHERE Idanimale IN(SELECT v.Animale
                                        FROM Visite v
                                       WHERE v.Datavisita<CURDATE() AND v.Tipovisita=2);
        END IF;
        IF Nuovevacc()
        THEN
                UPDATE SchedeCliniche
                SET Vaccinato=TRUE
                WHERE Idanimale IN(SELECT v.Animale
                                        FROM Visite v
                                       WHERE v.Datavisita<CURDATE() AND v.Tipovisita=1);
        END IF;
END$
```

6.3 Funzioni

Le seguenti funzioni sono utilizzate da alcuni trigger ed eventi nei punti precedenti.

1. Funzione che verifica se un certo veterinario è disponibile in una certa data e ora.

```
DROP FUNCTION IF EXISTS VerificaDisponibile;
DELIMITER $
CREATE FUNCTION VerificaDisponibile(Vet INTEGER,datav DATE,orav TIME) RETURNS
BOOLEAN
BEGIN
        DECLARE num INTEGER;
        SELECT COUNT(*) INTO num
        FROM Veterinari v
        WHERE v.Idveterinario=Vet AND
        (Idveterinario NOT IN (SELECT vv.Idveterinario
                                FROM VisiteVeterinari vv)
                OR v.Idveterinario NOT IN(SELECT vv.Idveterinario
                                            FROM VisiteVeterinari vv     NATURAL JOIN Visite vi
                                            WHERE vi.Datavisita=datav AND vi.Oravisita=orav));
        IF(num!=0)
                THEN
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
        END IF;
END$
DELIMITER ;
```

2. Funzione che ritorna il numero di veterinari disponibili per un certo tipo di visita in un certo giorno e orario.

```
DROP FUNCTION IF EXISTS VetDisponibili;
DELIMITER $
CREATE FUNCTION VetDisponibili(Tipov INTEGER,datav DATE,timev TIME) RETURNS INTEGER
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Veterinari v
        WHERE v.Idveterinario NOT IN(SELECT vv.Idveterinario
                                        FROM VisiteVeterinari vv)
         OR v.Idveterinario NOT IN(SELECT vv1.Idveterinario
                                    FROM VisiteVeterinari vv1 NATURAL JOIN Visite vi
                                    WHERE NOT(vi.Datavisita=datav AND vi.Oravisita=timev));
        RETURN conta;
END$
DELIMITER ;
```

3. Funzione che verifica se un veterinario può eseguire un certo tipo di visita.

```
DROP FUNCTION IF EXISTS Haspecializzazione;
DELIMITER $
CREATE FUNCTION Haspecializzazione(Vet INTEGER,Idtipo INTEGER) RETURNS
BOOLEAN
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Specializzazioni s
        WHERE s.Idveterinario=Vet AND s.Idtipovisita=Idtipo;
        IF conta>0
                THEN
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END$
DELIMITER ;
```

4. Funzione che verifica se ci sono nuove visite di sverminazione da registrare.

```
DROP FUNCTION IF EXISTS Nuovesverm;
DELIMITER $
CREATE FUNCTION Nuovesverm() RETURNS BOOLEAN
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Visite v
        WHERE v.Datavisita<CURDATE() AND v.Tipovisita=3
                AND v.Animale IN(SELECT s.Idanimale
                                        FROM SchedeCliniche s
                                        WHERE s.Sverminato=FALSE);
        IF(conta!=0)
                THEN
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
        END IF;
END$
DELIMITER ;
```

5. Funzione che verifica se ci sono nuove visite di vaccinazione da registrare.

```
DROP FUNCTION IF EXISTS Nuovevacc;
DELIMITER $
CREATE FUNCTION Nuovevacc() RETURNS BOOLEAN
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Visite v
        WHERE v.Datavisita<CURDATE() AND v.Tipovisita=1
                 AND v.Animale IN(SELECT s.Idanimale
                                                  FROM SchedeCliniche s
                                                  WHERE s.Vaccinato=FALSE);
        IF(conta!=0)
                THEN
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
        END IF;
END$
DELIMITER ;
```

6. Funzione che verifica se ci sono nuove visite di sterilizzazione da registrare.

```
DROP FUNCTION IF EXISTS Nuovester;
DELIMITER $
CREATE FUNCTION Nuovester() RETURNS BOOLEAN
BEGIN
        DECLARE conta INTEGER;
        SELECT COUNT(*) INTO conta
        FROM Visite v
        WHERE v.Datavisita<CURDATE() AND v.Tipovisita=2
                 AND v.Animale IN(SELECT s.Idanimale
                                          FROM SchedeCliniche s
                                          WHERE s.Sterilizzato=FALSE);
        IF(conta!=0)
                THEN
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
        END IF;
END$
DELIMITER ;
```

7 Interfaccia Web

È stata realizzata una semplice interfaccia web che permette di inserire, modificare, cancellare e ricercare record all'interno della base di dati.
Il sito è stato caricato nello spazio web al link:
`http://basidati/basidati/~aadami/progetto.`
Nello sviluppo del sito web è stato scelto di creare un menù di navigazione per facilitare all'utente l'utilizzo del sito.
Una parte del sito è riservata agli utenti che possiedono un account e per accedervi l'utente deve prima autenticarsi.
Se un utente non possiede un account, può accedere solamente alle pagine delle query e visualizzare i veterinari. Un utente che non possiede un account può comunque registrarsi e ottenere così un account.
Le pagine di ricerca(Cercaanimali.php e cercavet.php) inviano i dati mediante il metodo GET, mentre tutte le altre pagine inviano i dati mediante il metodo POST.
Ho scelto di salvare l'username e le password in una tabella Account del database; inoltre le password sono state salvate in scuro per la sicurezza dei dati.
Nella maggior parte delle pagine rivolte agli utenti autenticati viene testato se c'è qualche utente connesso, e in caso negativo non permette di accedere alla pagina e stampa un messaggio di "Accesso negato".
Le pagine per prenotare una nuova visita e registrare un nuovo animale nel database mantengono lo stato tra le pagine facendo uso degli hidden fields.
Sono state anche create delle pagine di dettaglio per gli animali(dettaglianimali.php), i veterinari(Dettaglivet.php) e le visite(dettaglivisita.php) che fanno uso di query string.
Tutte le funzioni utilizzate nelle varie pagine sono contenute nel file Utility.php, che contiene le seguenti funzioni:

- page_start è una funzione che inizia una pagina html
- page_end è una funzione che chiude una pagina html
- table_start è una funzione che inizia una tabella e ne crea l'intestazione
- table_row stampa una riga in una tabella

- table_end termina la tabella
- solonumeri controlla se la stringa passata è composta da soli numeri utilizzando le espressioni regolari
- solocaratteri è analoga alla funzione precedente, ma in questo caso viene controllato se la stringa passata è composta da soli caratteri
- lunghezzadata verifica se una stringa passata è lunga esattamente n
- headerdiv crea l'header delle pagine principali
- pathdiv crea un div che mostra l'area in cui l'utente si trova
- navdiv crea la barra di navigazione che mostra le pagine alle quali l'utente può accedere