Progetto di Basi di Dati

Giulia Amato, matricola 1075626
Paolo Broglio, matricola 1073874

06/2016

Accesso

Accesso Dottore
Nome Utente: **giovanniadami76**
Password: **HB51VWT6**

Accesso Admin
Nome Utente: **taddeobaghi86**
Password: **admin7**

Accesso CUP
Nome Utente: **CPADOVA01**
Password: **cuppadova**

Codice Fiscale di un Paziente dell' ASL di Padova
CF: **DRSSBR79M64I107Z**

Account per accesso PHPMYADMIN: **pbroglio**
Account per visualizzazione Sito Web: **pbroglio**

Abstract

Il progetto modella la gestione delle visite mediche relative ad un'agenda medica per le principali ASL del territorio Veneto. In particolare, tale agenda potrà essere visualizzata da tre tipi di utenti grazie all'interfaccia web sviluppata. Essi potranno effettuare alcune operazioni tipiche attraverso l'utilizzo di un proprio account:

1. Dottori, che hanno la possibilità di creare dei referti e di visualizzarli, visualizzare le visite mediche secondo una tale data e le informazioni dei pazienti
2. Amministratori, che gestiscono la parte amministrativa. Aggiungono nuovi dottori o infermieri, ma possono anche modificarne i dati o eliminarli
3. CUP (Centro Unico Prenotazioni) che ha l'incarico di gestire le prenotazioni attraverso la creazione, modifica e cancellazione.

Descrizione dei requisiti

Il progetto in esame vuole analizzare una base di dati che contenga e gestisca le informazioni relative ad un'agenda medica per la gestione delle visite mediche, ed in particolare si vogliono conoscere:

1. dati relativi allo svolgimento della visita medica
2. dati relativi al paziente

3. dati relativi al dottore
4. dati relativi all'infermiere
5. dati relativi alla ASL di appartenenza.

Per ordine, una **visita medica** è identificata dalle seguenti informazioni:

- Un codice, che la identifica univocamente
- Una data
- Un'ora
- Una priorità con la quale si potranno eventualmente ordinare le visite

Una visita potrà dunque rientrare in due stati (*effettuata* o *prenotata*), ed inoltre una visita potrà essere descritta come *visita di controllo* o come *visita medica*. Al termine di una visita sarà prodotto un **referto**, dotato di *codice* che lo identifica univocamente e un *testo* che descrive l'avvenuta visita. Ogni visita verrà svolta in un certo **ambulatorio** contenuto in una data **ASL** del territorio del Veneto. Una **ASL** sarà identificata da un *codice* univoco, e avrà altri dati significativi quali *indirizzo* e *contatti* (email e telefono). Inoltre, la gestione delle prenotazioni di visite mediche per ogni **ASL** verrà effettuata dal **CUP** (*Centro Unico Prenotazioni*), il quale è identificato da un *codice* univoco e da una *password*, che verranno usati per accedere nella base di dati nell'interfaccia web. Ad ogni **ASL** afferiscono un certo numero di **dottori** e **infermieri**, nonché **amministratori di sistema**. L'**amministratore** e il **dottore** avranno accesso al database per la gestione delle visite, perciò verranno identificati da:

- Un nome utente, univoco
- Una password
- Una data di scadenza della password, che dovrà essere rinnovata
- Uno stipendio

Per quanto riguarda il **dottore** è molto importante essere a conoscenza di dati aggiuntivi quali:

- La specializzazione, affinché le visite di una certa impronta possano essere indirizzate ai dottori competenti su quel settore
- Orario di inizio e fine turni di lavoro, importante per l'inserimento delle visite in un dato range di tempo

Per quanto riguarda l'**infermiere**, anch'esso avrà come dati significativi lo *stipendio*, ma anche l'attributo *tirocinante*, che identifica la qualifica di un dato infermiere. Tali categorie di utenza avranno tutte una serie di informazioni che caratterizzano un singolo individuo:

- Un codice fiscale, che identifica univocamente un utente
- Un nome ed un cognome
- Una data di nascita
- Il sesso
- Contatti quali email e telefono che identificano univocamente un singolo utente
- Indirizzo, con via e civico
- Città di nascita e città di residenza

Ogni **città** ha inoltre un *nome univoco*, una *provincia*, un *CAP* e una *regione*, anch'essa con un *nome univoco*.

Progettazione concettuale

Lista delle classi

Tutti gli attributi delle classi sono NOT NULL poiché sono elementi indispensabili per la gestione della base di dati. I soli attributi che possono assumere valore NULL sono: Email in Utente, Codice Esenzione in Paziente, Priorita' in Visita Medica.

UTENTE: rappresenta una persona fisica

- Nome: varchar(20)
- Cognome: varchar(20)
- Codice Fiscale: varchar(16)
- Data Nascita: date
- Sesso: char
- Email: varchar(45)
- Telefono: varchar(20)
- Indirizzo (Via, Civico): varchar(45)

Sono definite le seguenti sottoclassi di Utente (con vincolo di partizionamento):

1) **PAZIENTE**: rappresenta i pazienti

- Codice Esenzione: varchar(3)

2) **INFERMIERE:** rappresenta gli infermieri di una data ASL

- Stipendio: smallint
- Tirocinante: bool

3) **ACCESSO:** rappresentano gli utenti che possono accedere al database

- Nome utente: varchar(20)
- Password: varchar(20)
- DataScadenza: date

A sua volta, definito in sottoclassi:
1) **ADMIN:** Definisce l'admin del sistema
2) **DOTTORE:** Definisce i dottori di una data ASL

- Specializzazione varchar(30)
- Orario Lavoro (Orario Inizio, Orario Fine) time

3) **CUP:** Ente che gestisce le prenotazioni di una data visita medica

- Codice: varchar(10)
- Password: varchar(10)

CITTA': rappresenta le città di una sede o la residenza di un utente

- Nome: varchar(45)
- Provincia: varchar(2)
- CAP: varchar(5)

REGIONE: rappresenta una regione di appartenenza di una città

- Nome: varchar(15)

ASL: modella le informazioni di una ASL

- Indirizzo (Via, Civico): varchar(45)
- Contatti (Email): varchar(45)
- Contatti (Telefono): varchar(20)

AMBULATORIO: rappresenta il luogo in cui verrà eseguita una visita medica

- Nome: varchar(5)

VISITA MEDICA: modella le visite mediche

- Priorità: char

- Ora: time
- Data: date
- Codice: int
- Visita / Controllo: bool
- Prenotata / Effettuata: bool

REFERTO: rappresenta un referto creato dal dottore a seguito di una visita medica

- Testo: longtext
- Codice Referto: int

Lista delle associazioni

UTENTE-CITTA':*è nato in*

Ogni utente ha una e una sola città di nascita. Una città può avere 0 o più persone nate in quella città. Molteplicità 1:N

UTENTE-CITTA': *è residente in*

Ogni utente ha una e una sola città di residenza. Una città può avere 0 o più persone residenti in quella città. Molteplicità 1:N

UTENTE-ASL: *afferisce in*

Un utente afferisce ad una ed una sola ASL, mentre una ASL puo' avere da 0 a N utenti che afferiscono ad essa. Molteplicita' 1:N

CITTA'-REGIONE: *appartiene a*

Ogni Città appartiene ad una e una sola Regione. Una regione può avere una città, o più. Molteplicità: 1:N

CITTA'-ASL: *sede di*

Ogni città può essere sede di zero ASL, o al più una. Una ASL appartiene ad una ed una sola città. Molteplicità: 1:1

AMBULATORIO-ASL: *contenuto in*

Ogni ambulatorio è contenuto in zero ASL, o al più N. Ogni ASL può avere un ambulatorio, o più. Molteplicità: N:N

VISITA MEDICA-AMBULATORIO: *svolta in*

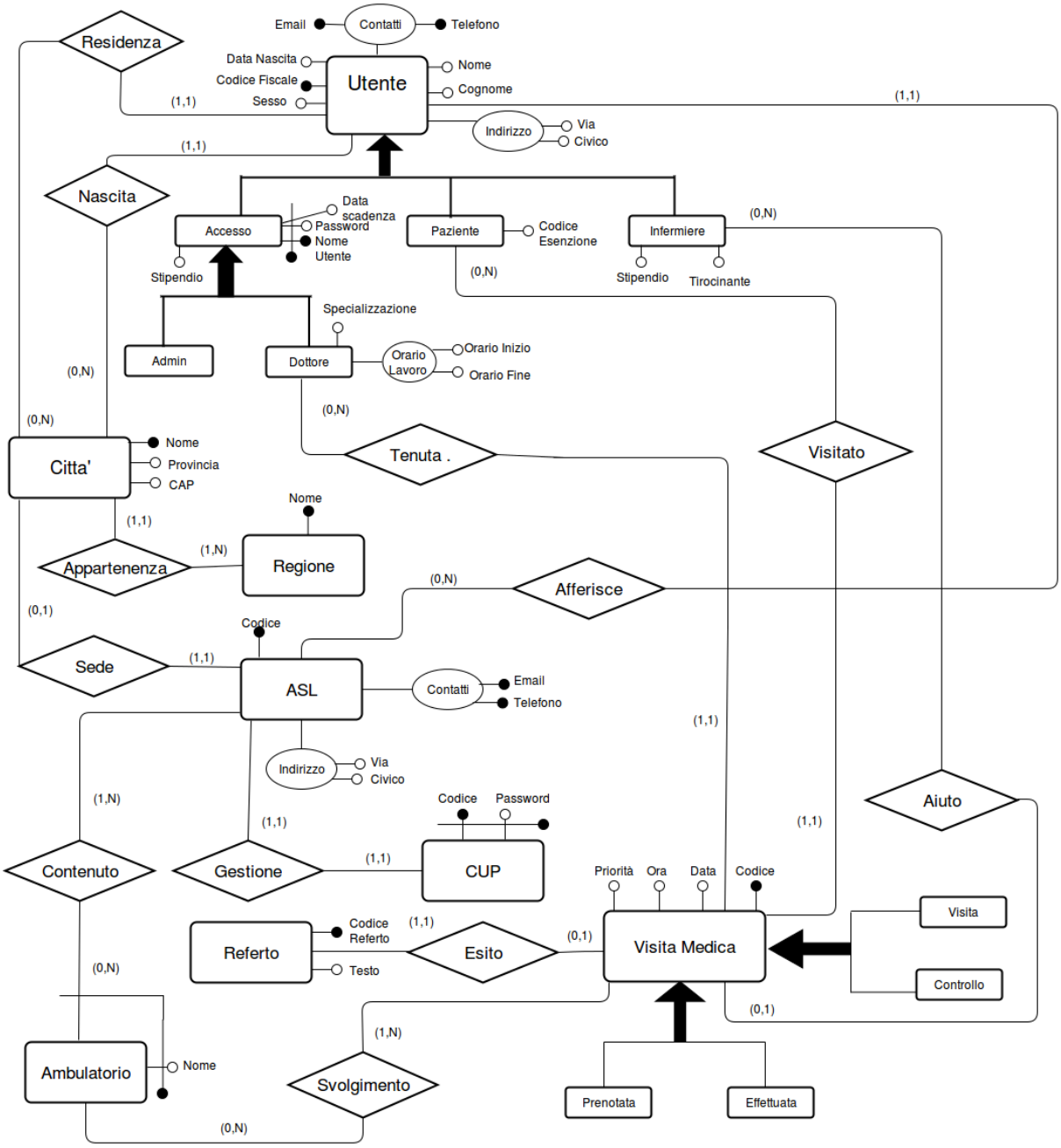

Figura 1: Schema Concettuale

Ogni visita medica si svolge in un ambulatorio, o al più N. In un ambulatorio può essere svolta nessuna visita medica, o più. Molteplicità N:N

VISITA MEDICA-REFERTO: *esito di*

Un referto può essere un esito di una e una sola visita medica. Una visita medica può avere zero referti, o al più uno. Molteplicità 1:1

ASL-CUP: *gestisce*

Ogni ASL ha le prenotazioni gestite da uno e un solo CUP. Un CUP può gestire una e una sola ASL. Molteplicità 1:1

INFERMIERE-VISITA MEDICA: *aiuto*

Ogni infermiere può prendere parte a 0 o N visite mediche. Una visita medica può avere 0 o 1 infermiere che presta servizio. Molteplicità N:1

PAZIENTE-VISITA MEDICA: *visitato*

Ogni paziente può prendere parte di nessuna visita medica, o al più N. Una visita medica invece può avere uno ed un solo paziente. Molteplicità: 1:N

DOTTORE-VISITA MEDICA: *tenuta*

Ogni visita medica è tenuta da uno ed un solo dottore, mentre un dottore può tenere 0 o N visite mediche. Molteplicità: 1:N

Descrizione della gerarchia tra le classi

Esiste una gerarchia totale, in cui ogni occorrenza dell'entita' genitore e' una occorrenza di almeno una delle entita' figlie, nell'entita' Utente. Essa infatti ha come figli le entita' Paziente, Infermiere e Accesso. Quest'ultima a sua volta e' genitore di un'altra generalizzazione, poiche' Accesso denota la tipologia di utenti che possono avere accesso alla Base Dati, ovvero Admin e Dottore. Inoltre, esiste anche una seconda gerarchia totale, che riguarda l'entita' Visita Medica. Ci sono infatti due generalizzazioni totali. La prima, esclusiva, permette di descrivere una Visita Medica come Prenotata o Effettuata. La seconda, anch'essa esclusiva, permette di descrivere una Visita Medica come Visita o come Controllo.

Progettazione logica

La generalizzazione della classe Utente è stata tradotta tramite sostituzione della generalizzazione con associazioni dato che gli altri due metodi (accorpamento delle figlie della generalizzazione nel genitore, e accorpamento del genitore della generalizzazione nelle figlie) avrebbe appesantito la specificità delle varie entità, provocando diversi svantaggi. Con la sostituzione con associazioni, invece, la generalizzazione si trasforma in associazioni 1 a 1 che legano rispettivamente l'entità genitore con le entità figlie. Non ci sono trasferimenti di attributi. Viene dunque eliminata la classe Utente, e sostituita dalla classe Dati.

Associazioni

DATI - CITTA': *residenza*

Ogni utente è residente di una sola città, mentre una città può avere da 0 a N residenti. La molteplicità è 1 - N.

DATI - CITTA': *nascita*

Ogni utente è nato di una sola città, mentre una città può avere da 0 a N nati nel proprio territorio. La molteplicità è 1 - N.

DATI - ADMIN: *informazioni*

Un utente può essere o non essere un admin, mentre un admin è sicuramente un utente. La molteplicità è 1 - 1.

ADMIN - CREDENZIALI D'ACCESSO: *accesso*

Un admin può avere una e una sola credenziale d'accesso. Una credenziale d'accesso puo' appartenere ad un solo admin. La molteplicita' è 1 - 1.

DOTTORE - CREDENZIALI D'ACCESSO: *accesso*

Un dottore può avere una e una sola credenziale d'accesso. Una credenziale d'accesso puo' appartenere ad un solo dottore. La molteplicita' è 1 - 1.

DATI - PAZIENTE: *informazioni*

Un utente può essere o non essere un paziente, mentre un paziente è sicuramente un utente. La molteplicità è 1 - 1.

DATI - INFERMIERE: *informazioni*

Un utente può essere o non essere un infermiere, mentre un infermiere è sicuramente un utente. La molteplicità è 1 - 1.

DATI - DOTTORE: *informazioni*

Un utente può essere o non essere un dottore, mentre un dottore è sicuramente un utente. La molteplicità è 1 - 1.

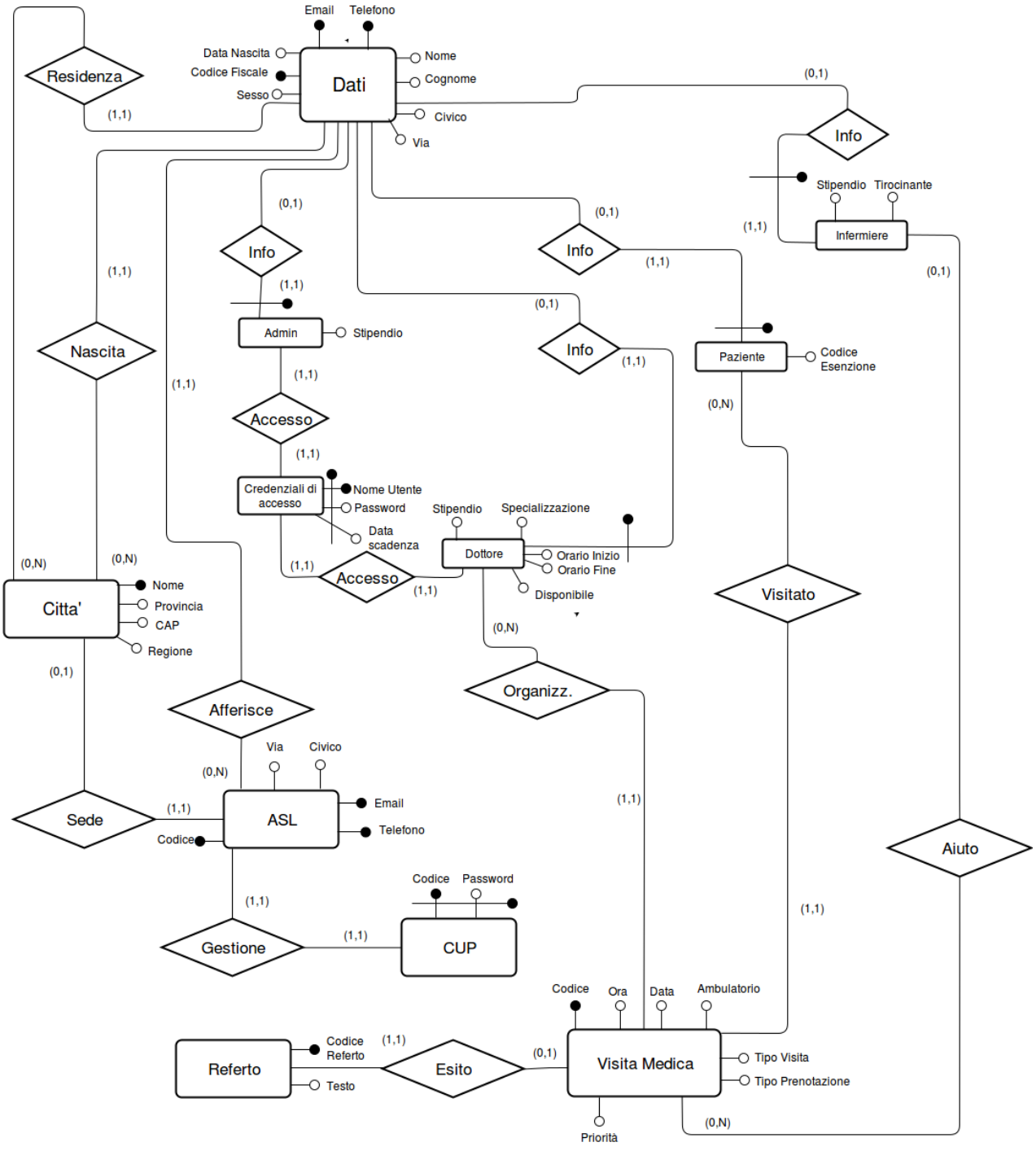

Figura 2: Schema Logico

DATI - ASL: *afferisce*

Un utente può afferire ad una e una sola ASL, mentre una ASL può avere da 0 a più utenti. La molteplicità è 1 - N.

CITTA' - ASL: *sede*

Un città può essere o non essere sede di una ASL, considerando i piccoli comuni. Mentre una ASL ha sicuramente una sede in una città. La molteplicità è 1 - 1.

ASL - CUP: *gestione*

Le prenotazioni per una ASL vengono gestite da uno e un solo CUP, e un CUP può gestire una e una sola ASL. La molteplicità è 1 - 1.

DOTTORE - VISITA MEDICA: *organizzazione*

Un dottore può effettuare da zero a più visite mediche, mentre una visita medica è sicuramente svolta da uno e un solo dottore. La molteplicità è N - 1.

VISITA MEDICA - REFERTO: *esito*

Un visita medica può avere un referto quando si trova nello stato "effettuata", o puo' non averlo se si trova nello stato "prenotata" , mentre un referto appartiene sicuramente ad un unica visita. La molteplicità è 1 - 1.

PAZIENTE - VISITA MEDICA: *visitato*

Un paziente presente nella base dati può essere stato visitato N volte, o può ancora non essere stato visitato se la visita e' in stato "prenotato". Mentre una visita medica riguarda uno ed un solo paziente. La molteplicità è N - 1.

INFERMIERE - VISITA MEDICA: *aiuto*

Una visita medica può avere o meno un infermiere che appoggia il dottore, e una infermiere può essere di aiuto in 0 o N visite mediche. La molteplicità è 1 - N.

Implementazione della base di dati

Qui di seguito verrà riportato il codice relativo alla base di dati. Verranno definite le tabelle:

```
 CREATE TABLE Informazioni (

CodiceFiscale  VARCHAR(16) NOT NULL PRIMARY KEY,
DataNascita    DATE NOT NULL,
Nome     VARCHAR(20) NOT NULL,
```

```
Cognome    VARCHAR(20) NOT NULL,
Email    VARCHAR(45),
Sesso    CHAR NOT NULL,
Telefono    VARCHAR(20) NOT NULL,
CittaResidenza VARCHAR(45) NOT NULL,
CittaNascita   VARCHAR(45) NOT NULL,
Indirizzo    VARCHAR(45) NOT NULL,
CodiceASL     INT NOT NULL,

FOREIGN KEY (CittaResidenza) REFERENCES Citta(Nome)
ON UPDATE CASCADE,
FOREIGN KEY (CittaNascita) REFERENCES Citta(Nome)
ON UPDATE CASCADE,
FOREIGN KEY (CodiceASL) REFERENCES ASL(Codice)



) ENGINE=INNODB;

CREATE TABLE Citta (

Nome    VARCHAR(45) NOT NULL PRIMARY KEY,
Provincia VARCHAR(2) NOT NULL,
CAP   VARCHAR(5) NOT NULL,
Regione   VARCHAR(15) NOT NULL


)ENGINE=INNODB;

CREATE TABLE ASL (

Codice    INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
Indirizzo VARCHAR(45) NOT NULL,
Email     VARCHAR(45) NOT NULL,
Telefono  VARCHAR(20) NOT NULL,
CittaSede VARCHAR(45) NOT NULL,

FOREIGN KEY (CittaSede) REFERENCES Citta(Nome)
ON UPDATE CASCADE


)ENGINE=INNODB;


CREATE TABLE CUP (
```

```
Codice VARCHAR(10) NOT NULL PRIMARY KEY,
Password VARCHAR(10) NOT NULL,
CodiceASL INT NOT NULL,

FOREIGN KEY (CodiceASL) REFERENCES ASL(Codice)


) ENGINE=INNODB;


CREATE TABLE DatiAccesso (

CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
NomeUtente VARCHAR(20) NOT NULL UNIQUE,
Password VARCHAR(20) NOT NULL,
DataScadenza DATE NOT NULL,

FOREIGN KEY (CodiceFiscale) REFERENCES Informazioni(CodiceFiscale)
ON DELETE CASCADE

)ENGINE=INNODB;


CREATE TABLE Admin (

CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
NomeUtente VARCHAR(20) NOT NULL,
Stipendio SMALLINT NOT NULL,

FOREIGN KEY (CodiceFiscale) REFERENCES Informazioni(CodiceFiscale)
ON DELETE CASCADE


) ENGINE=INNODB;

CREATE TABLE Dottore (

CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
NomeUtente VARCHAR(20) NOT NULL,
Stipendio SMALLINT NOT NULL,
Specializzazione VARCHAR(30) NOT NULL,
OraInizio TIME NOT NULL,
OraFine TIME NOT NULL,
Disponibile BOOL NOT NULL,

FOREIGN KEY (CodiceFiscale) REFERENCES Informazioni(CodiceFiscale)
```

```
ON DELETE CASCADE


) ENGINE=INNODB;

CREATE TABLE Paziente (

CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
CodiceEsenzione VARCHAR(3),

FOREIGN KEY (CodiceFiscale) REFERENCES Informazioni(CodiceFiscale)
ON DELETE CASCADE


) ENGINE=INNODB;

CREATE TABLE Infermiere (

CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
Stipendio SMALLINT NOT NULL,
Tirocinante BOOL NOT NULL,

FOREIGN KEY (CodiceFiscale) REFERENCES Informazioni(CodiceFiscale)
ON DELETE CASCADE

)ENGINE=INNODB;


CREATE TABLE VisitaMedica (

CodiceVisita INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
Data DATETIME NOT NULL,
NomeAmbulatorio VARCHAR(5) NOT NULL,
TipoVisita BOOLEAN NOT NULL,
TipoPrenotazione BOOLEAN NOT NULL,
Priorita CHAR,
CFDottore VARCHAR(16),
CFInfermiere VARCHAR(16),
CFPaziente VARCHAR(16) NOT NULL,
CodiceReferto INT,

FOREIGN KEY (CFDottore) REFERENCES Dottore(CodiceFiscale)
ON DELETE SET NULL,
FOREIGN KEY (NomeAmbulatorio) REFERENCES Ambulatorio(Nome),
FOREIGN KEY (CFInfermiere) REFERENCES Infermiere(CodiceFiscale)
ON DELETE SET NULL,
```

```
FOREIGN KEY (CFPaziente) REFERENCES Paziente(CodiceFiscale)
ON DELETE CASCADE,
FOREIGN KEY (CodiceReferto) REFERENCES Referto(Codice)
ON DELETE SET NULL

) ENGINE=INNODB;

CREATE TABLE Referto (

Codice INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
Contenuto LONGTEXT NOT NULL


)ENGINE=INNODB;

CREATE TABLE Ambulatorio (

Nome VARCHAR(5) PRIMARY KEY NOT NULL,
CodiceASL INT NOT NULL,
FOREIGN KEY (CodiceASL) REFERENCES ASL(Codice)

)ENGINE=INNODB;
```

Triggers

1) Questo trigger viene attivato quando un dottore viene eliminato. Nella tabella delle visite mediche tutte quelle prenotate vengono cancellate. Quelle effettuate invece avranno l'attributo CFDottore = NULL poiché il referto deve rimanere disponibile.

```
DELIMITER $$
DROP IF EXISTS TRIGGER controlla_visita $$
CREATE TRIGGER controlla_visita
 BEFORE DELETE ON Dottore
 FOR EACH ROW
 BEGIN
  DELETE FROM VisitaMedica
  WHERE CFDottore=OLD.CodiceFiscale
  AND TipoPrenotazione=0;
 END $$
DELIMITER;
```

2) Quando una visita medica diventa di tipo "effettuata"(tipoPrenotazione=1) allora controlla il numero di visite a cui a partecipato l'infermiere (se c'è) e se questo numero è uguale a 4 allora aumenta di 200 lo stipendio.

```
DELIMITER $$
DROP IF EXISTS TRIGGER premio_infermiere $$
CREATE TRIGGER premio_infermiere
BEFORE UPDATE ON VisitaMedica
FOR EACH ROW
BEGIN
IF conta_visite(OLD.CFInfermiere) = 4
THEN
UPDATE Infermiere
SET Infermiere.Stipendio = Infermiere.Stipendio+200
WHERE Infermiere.CodiceFiscale=OLD.CFInfermiere;
END IF;
END $$
DELIMITER;
```

Funzioni

1) Funzione che permette di calcolare quante visite mediche ha effettuato un certo infermiere.

```
CREATE FUNCTION ‘Conta_Visite‘(CF_infermiere VARCHAR(16))
RETURNS smallint(5) unsigned
BEGIN
DECLARE tot_visite SMALLINT UNSIGNED;
SELECT Count(*) into tot_visite
FROM VisitaMedica VM
WHERE VM.CFinfermiere = CF_infermiere AND VM.TipoPrenotazione = 1
GROUP BY VM.CFinfermiere;
RETURN tot_visite;
END;
```

2) Funzione che verifica se un dottore con una certa specializzazione è disponibile in una certa ora. La funzione ritorna il codice fiscale di un solo dottore.

```
DELIMITER $$
CREATE FUNCTION ‘controlla_dottore‘(‘ora‘ TIME, ‘specializzazione‘ VARCHAR(30))
 RETURNS varchar(16)
BEGIN
DECLARE Disponibilita BOOL;
DECLARE cf VARCHAR(16);
SELECT D.CodiceFiscale,D.Disponibile INTO cf, Disponibilita
FROM Dottore D
```

```
WHERE ora BETWEEN D.OraInizio AND D.OraFine
AND D.Specializzazione = specializzazione
LIMIT 1;
IF(Disponibilita=1)
THEN RETURN cf;
ELSE RETURN NULL;
END IF;
END$$
DELIMITER ;
```

Query e Procedure

In questa sezione verranno proposte le query implementate:

1. Query che permette di ottenere il massimo numero di visite mediche effettuate da un infermiere tirocinante,il massimo numero di visite mediche effettuate da un dottore, in cui la priorità delle visite e' al livello minore (L), e il massimo numero di visite mediche effettuate da un dottore, in cui la priorità delle visite e' al livello maggiore (H)

```
CREATE VIEW visite_effettuate AS

SELECT MAX(VisiteEffettuate) AS Visite,
T.CFInfermiere AS CodiceFiscale
FROM (
SELECT VM.CFInfermiere, VM.CFDottore, COUNT(*) VisiteEffettuate
FROM Infermiere i, VisitaMedica VM
WHERE i.Tirocinante = 1 AND
VM.CFInfermiere = i.CodiceFiscale
GROUP BY VM.CFInfermiere, VM.CFDottore ) AS T
UNION
SELECT MAX(VisiteEffettuate), T2.CFDottore
FROM (
SELECT VM.CFDottore, COUNT(*) VisiteEffettuate
FROM VisitaMedica VM
WHERE VM.TipoPrenotazione = 1 AND
 VM.Priorita = 'L'
 GROUP BY VM.CFDottore ) AS T2
UNION
SELECT MAX(VisiteEffettuate), T3.CFDottore
FROM (
SELECT VM.CFDottore, COUNT(*) VisiteEffettuate
FROM VisitaMedica VM
WHERE VM.TipoPrenotazione = 1 AND
VM.Priorita = 'H'
GROUP BY VM.CFDottore ) AS T3;
```

2. Query che, dati i codici fiscali dei dottori che non hanno mai fatto visite mediche, mostra il nome utente, il codice fiscale e lo stipendio di questi dottori, selezionando coloro che hanno la data scadenza dell'account tra il 1 Gennaio 2016 e il 1 Ottobre 2016, il cui stipendio e' maggiore di 3000, il nome utente comprende una ca_ , e con l'orario di inizio del turno del lavoro compreso tra le 9 e le 13, e il turno di fine compreso tra le 13 e le 15.45

```
CREATE VIEW mostra_dottri_stipendio AS
SELECT D.NomeUtente, D.CodiceFiscale, Doct.Stipendio
FROM DatiAccesso D JOIN Dottore Doct
ON D.CodiceFiscale = Doct.CodiceFiscale AND
D.DataScadenza BETWEEN "2016-01-01" AND "2016-10-01"
AND Doct.Stipendio>30000 AND
D.NomeUtente LIKE "%ca_%" AND
Doct.OraInizio BETWEEN "9:00:00" AND "13:30:00"
AND Doct.OraFine BETWEEN "13:00:00" AND "15:45:00"
AND Doct.CodiceFiscale NOT IN (
SELECT VM.CFDottore
FROM VisitaMedica VM, Informazioni I
WHERE VM.CFdottore=I.CodiceFiscale);
```

3. Query che restituisce il codice fiscale dei dottori, il nome, il cognome e, per ogni dottore presente nel base dati, a prescindere dunque delle ASL di appartenenza, conta il numero di visite mediche effettuate nel mese di Maggio 2016

```
CREATE VIEW conta_visite AS
SELECT VM.CFdottore, I.Nome, I.Cognome, COUNT(VM.CFdottore) Visite_Effettuate
FROM VisitaMedica VM JOIN Informazioni I
ON VM.CFdottore = I.CodiceFiscale
WHERE VM.CFdottore IN (
SELECT VM.CFdottore
FROM VisitaMedica VM
WHERE VM.TipoPrenotazione = 1 AND
VM.Data BETWEEN '2016-05-01 00:00:00' AND '2016-05-31 23:59:59')
GROUP BY VM.CFdottore;
```

4. Query che permette di ottenere il maggior numero di infermieri tirocinanti uomini nati in una data posteriore al 1 Gennaio 1970, e di ottenere il maggior numero di infermieri tirocinanti donne, nate anch'esse in una data posteriore a quella già espressa

```
CREATE VIEW tirocinanti AS
```

```
SELECT MAX(T1.CountInf) AS conta
FROM (
SELECT COUNT(Infermiere.CodiceFiscale) AS CountInf,Informazioni.CodiceASL
FROM Infermiere JOIN Informazioni
ON Infermiere.CodiceFiscale=Informazioni.CodiceFiscale
WHERE Infermiere.Tirocinante=1
AND CAST(Informazioni.DataNascita as date)>1970-01-01 AND Informazioni.Sesso='M'
GROUP BY Informazioni.CodiceASL) T1
UNION
SELECT MAX(T2.CountInf) AS conta
FROM (
SELECT COUNT(Infermiere.CodiceFiscale) AS CountInf,Informazioni.CodiceASL
FROM Infermiere JOIN Informazioni ON
Infermiere.CodiceFiscale=Informazioni.CodiceFiscale
WHERE Infermiere.Tirocinante=1 AND
CAST(Informazioni.DataNascita as date)>1970-01-01
AND Informazioni.Sesso='F'
GROUP BY Informazioni.CodiceASL) T2;
```

5. Query che restituisce nome, cognome, codice dell'ASL di appartenenza e numero di visite effettuate dei dottori che nel mese precedente hanno effettuato meno di 5 visite mediche

```
CREATE VIEW visite_mese_precedente AS
SELECT DISTINCT Informazioni.Nome, Informazioni.Cognome, Informazioni.CodiceASL,
Informazioni.CodiceFiscale, COUNT(*) NumeroVisite
FROM VisitaMedica JOIN Informazioni
ON Informazioni.CodiceFiscale = VisitaMedica.CFDottore
WHERE TipoPrenotazione=1
AND
EXTRACT(YEAR FROM VisitaMedica.Data)=
EXTRACT(YEAR FROM DATE_SUB(CURDATE(),INTERVAL 1 MONTH))
AND
EXTRACT(MONTH FROM VisitaMedica.Data)=
EXTRACT(MONTH FROM DATE_SUB(CURDATE(),INTERVAL 1 MONTH))
GROUP BY CFDottore HAVING COUNT(*)<5;
```

6. Query che rende possibile ordinare le visite mediche ancora da effettuare per anzianità, mostrando il nome e il cognome del paziente, ma anche la data di nascita e il codice fiscale. Avremo a disposizione anche le informazioni relative alla visita medica prenotata, mostrano il nome dell'ambulatorio in cui verra' eseguita la visita, il tipo di visita, i codici fiscali dei dottori e degli infermieri

```
CREATE VIEW sort_anzianita AS
SELECT I.Nome, I.Cognome, I.DataNascita,
VM.CFPaziente, I.CodiceASL, VM.CodiceVisita,
```

```
VM.Data, VM.NomeAmbulatorio, VM.TipoVisita,
VM.Priorita, VM.CFDottore, VM.CFInfermiere,
VM.CodiceReferto
FROM VisitaMedica VM  JOIN Informazioni I
WHERE VM.CFpaziente = I.CodiceFiscale AND
VM.TipoPrenotazione = 0
ORDER BY I.DataNascita;
```

7. Procedura che inserisce il referto e aggiorna la visita medica inserita con il codice del referto appena aggiunto

```
DELIMITER $$
CREATE PROCEDURE insert_referto(IN testo_referto TEXT, IN cod_visita INT)
BEGIN
INSERT INTO ‘Referto‘(‘Contenuto‘)
VALUES (testo_referto);
SET @last_id_inserito = LAST_INSERT_ID();
UPDATE ‘VisitaMedica‘
SET ‘CodiceReferto‘=@last_id_inserito
WHERE ‘CodiceVisita‘= cod_visita;
END $$
DELIMITER;
```

8. Procedura che permette di effettuare un ordinamento per data (crescente o decrescente) delle visite mediche prenotate ancora da effettuare

```
DELIMITER $$
CREATE PROCEDURE ‘Ordina_Data‘(IN Ordinamento BOOLEAN)
BEGIN
SELECT *
FROM VisitaMedica
WHERE VisitaMedica.TipoPrenotazione=0
ORDER BY
CASE WHEN Ordinamento = 0 THEN Data END,
CASE WHEN Ordinamento = 1 THEN Data END DESC;
END $$
DELIMITER;
```

9. Procedura che permette di effettuare un ordinamento per priorità (H o L) delle visite mediche prenotate ancora da effettuare

```
DELIMITER $$
CREATE PROCEDURE ‘Ordina_Priorita‘(IN Prio CHAR(1))
BEGIN
SELECT *
FROM VisitaMedica
ORDER BY
CASE WHEN Prio = ’L’ THEN Priorita END DESC,
```

```
CASE WHEN Prio = 'H' THEN Priorita END,
CASE WHEN Prio NOT IN ('L', 'H') THEN Priorita END DESC;
END $$
DELIMITER;
```

10. Procedura che restituisce gli ambulatori disponibili in una certa data e di una certa ASL

```
#
# VIEW NECESSARIA
#
CREATE VIEW ambulatori_prenotati AS
SELECT Ambulatorio.Nome, VisitaMedica.Data
FROM Ambulatorio
JOIN VisitaMedica
ON Ambulatorio.Nome=VisitaMedica.NomeAmbulatorio
WHERE VisitaMedica.TipoPrenotazione = 0;

DELIMITER $$
CREATE PROCEDURE ambulatori_disp(IN data_scelta DATE,
IN ora_scelta TIME, IN cf_dottore VARCHAR(16))
BEGIN
DECLARE conta INTEGER;
CREATE TEMPORARY TABLE temp (nome VARCHAR(5));
INSERT INTO temp
SELECT ambulatori_prenotati.Nome
FROM ambulatori_prenotati
JOIN Ambulatorio
ON ambulatori_prenotati.Nome = Ambulatorio.Nome
WHERE DATE_FORMAT(ambulatori_prenotati.Data,'%Y-%m-%d')
= data_scelta
AND Ambulatorio.CodiceASL =
(SELECT CodiceASL FROM Informazioni W
HERE Informazioni.CodiceFiscale=cf_dottore)
AND DATE_FORMAT(ambulatori_prenotati.Data, "%T")
BETWEEN OraInizio(cf_dottore)
AND OraFine(cf_dottore)
AND DATEDIFF(OraFine(cf_dottore),
DATE_FORMAT(ambulatori_prenotati.Data,"%T")) > '00:30:00'
AND
(DATEDIFF(DATE_FORMAT(ambulatori_prenotati.Data,"%T"),
ora_scelta) > '00:30:00'
OR
DATEDIFF(ora_scelta,
DATE_FORMAT(ambulatori_prenotati.Data,"%T")) > '00:30:00');

SELECT COUNT(*) INTO conta FROM temp;
```

```
IF conta > 0
THEN SELECT * FROM temp;
ELSE
SELECT Ambulatorio.Nome
FROM Ambulatorio WHERE Ambulatorio.CodiceASL=
(SELECT CodiceASL FROM Informazioni WHERE Informazioni.CodiceFiscale=cf_dottore)
AND Ambulatorio.Nome
NOT IN (SELECT VisitaMedica.NomeAmbulatorio
FROM VisitaMedica WHERE VisitaMedica.TipoPrenotazione=0);
END IF;

END $$
DELIMITER;
```

Interfaccia Web

Di seguito sono descritte le pagine Web che permettono ai tre utenti della base di dati di accedere e operare secondo i permessi a loro concessi. Per ognuno è stata creata una pagina di login che utilizza una form e delle query che controllano l'esistenza della combinazione username/password per l'utente che effettua l'accesso. Inoltre a seguito dell'autenticazione viene eseguito un redirect alla pagina index fatta appositamente per ogni utente.

Dottore

Il dottore è l'utente che utilizza l'Agenda Medica in ogni suo aspetto. Attraverso la pagina indexdottore.php è possibile utilizzare l'agenda e visualizzare quindi la lista delle visite mediche effettuate dal dottore e quelle prenotate a suo nome. Inoltre è disponibile una tabella contenente i dati dei pazienti che afferiscono alla Associazione Sanitaria Locale in cui lavora il dottore stesso. Sono forniti due metodi per ordinare le visite mediche marcate come prenotate: uno per priorità, partendo da quella più alta 'H' fino a quella più bassa 'L' e l'altro per anzianità del paziente. L'inserimento di una nuova visita medica viene fatto attraverso il bottone apposito. La pagina nuovavisita.php permette attraverso una form da compilare l'inserimento dei dettagli necessari che caratterizzano ogni visita presente nel database. Per conoscere in tempo reale l'occupazione degli ambulatori conoscendo la data e l'ora in cui si vuole eseguire la visita è stata usata una chiamata AJAX che inserisce nel campo select degli ambulatori quelli disponibili. Questa funzione JavaScript utilizza un modulo php che a sua volta usa la procedura 'ambulatoridisp' per ottenere i dati interessati. Per il resto si tratta di compilare manualmente i campi disponibili, o selezionare quelli forniti nel caso di campi select.

Amministratore

L'amministratore attraverso il suo pannello a cui accede in seguito alla login può operare direttamente sui dottori e sugli infermieri che lavorano nella ASL di cui è amministratore. Sono possibili le normali operazioni di inserimento e cancellazione e in più una pagina, infoutili.php, che fornisce dati interessanti a proposito delle ASL del Veneto, accessibile tramite il link presente nel pannello. Per quanto riguarda l'inserimento di nuovi medici o infermieri è necessario indicare anche dati sensibili che solo gli amministratori conoscono al momento dell'inserimento, come la password e il nome utente di un dottore.

CUP

Coloro che accedono come utenti CUP hanno a disposizione il potere di creare immediatamente una visita medica indicando il dottore che deve effettuarla, l'infermiere che deve partecipare e il paziente che ha richiesto la visita stessa. Utilizzando la medesima funzione JavaScript usata per consentire l'inserimento di una nuova visita al medico è possibile ottenere la lista degli ambulatori disponibili nella data e nell'ora fornite.