UNIVERSITÀ DEGLI STUDI DI FERRARA
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE
BASI DI DATI

Realizzazione Database per Social Network

Azzolini Damiano - Bertagnon Alessandro

UNIVERSITÀ
DEGLI STUDI
DI FERRARA
- EX LABORE FRUCTUS -

INDICE

ELENCO DELLE FIGURE

CAPITOLO 1

MINIMONDO

1.1 DESCRIZIONE

Il progetto di basa sulla realizzazione di una applicazione web per la gestione di una clinica privata. Alla piattaforma possono accedere 5 tipi di utente:

- paziente
- medico
- infermiere
- impiegato
- amministratore

La clinica in questione eroga diversi tipi di **prestazioni** ai suoi utenti, ad esempio: visite specialistiche, esami diagnostici, day surgery e terapie. Ogni prestazione può essere effettuata da uno o più membri dello **staff** (a seconda della complessità) in una delle **sale** della clinica. Al termine di ogni prestazione il medico compila un **referto** corrispondete alla prestazione appena effettuata. Il sistema deve anche gestire i **farmaci** assunti dagli utenti e utilizzati durante le prestazioni. Per motivi di organizzazione interna ogni membro del personale e ogni sala afferisce a uno specifico **reparto** della clinica. Più in dettaglio:

L'utente **PAZIENTE** potrà:

- Registrarsi sulla piattaforma e fare il login. Modificare il proprio profilo.
- Aggiungere/rimuovere i farmaci che assume regolarmente.
- Visionare le prestazioni effettuate con i referti corrispondenti.

L'utente **MEDICO** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Visionare le schede personali dei pazienti (compresi i farmaci assunti).
- Visionare le prestazioni e i relativi referti.
- Aggiungere/Modificare/Cancellare i referti delle prestazioni a cui ha preso parte.
- Aggiungere/Rimuovere i farmaci utilizzati nelle prestazioni a cui ha preso parte.
- Aggiungere personale alle prestazioni che gli sono state assegnate.

L'utente **INFERMIERE** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Visionare le schede personali dei pazienti (compresi i farmaci assunti).
- Visionare le prestazioni assegnate.
- Aggiungere/Rimuovere i farmaci utilizzati nelle prestazioni alle quali ha preso parte.

L'utente **IMPIEGATO** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Prenotare le prestazioni per i pazienti associando ad esse i medici che dovranno effettuarle.
- Visualizzare lo storico delle prestazioni effettuate dai pazienti (ma non i referti).
- Gestire il personale:
  - Modificare lo stipendio dei vari membri dello staff.
  - Modificare il reparto di appartenenza.
- Aggiungere/Modificare/Cancellare le tipologie di prestazioni.
- Aggiungere/Modificare/Cancellare i farmaci nella lista della farmacia.

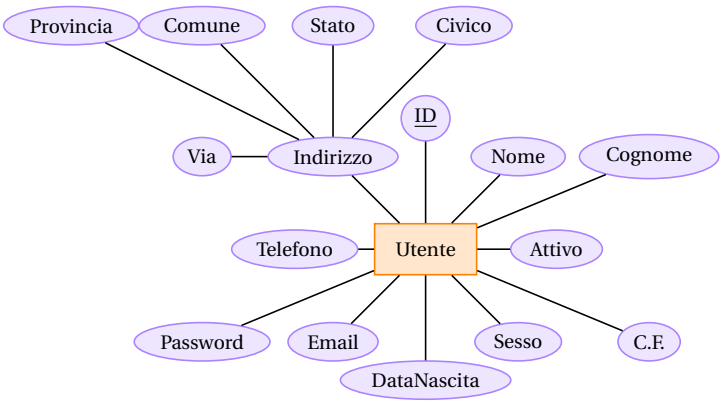
L'utente **AMMINISTRATORE** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Fare tutto quello che fanno gli utenti precedenti.
- Aggiungere/Modificare/Cancellare gli utenti Staff della clinica.
- Aggiungere/Modificare/Cancellare le sale della clinica.
- Aggiungere/Modificare/Cancellare i reparti della clinica.
- Gestire tutta la base utenti.
- Aggiungere/Rimuovere i singoli ruoli (permessi) agli utenti.

1.2 Entità

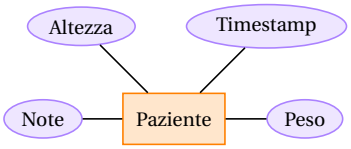
Di seguito vengono analizzate tutte le entità presenti nel database:

1.2.1 Utente



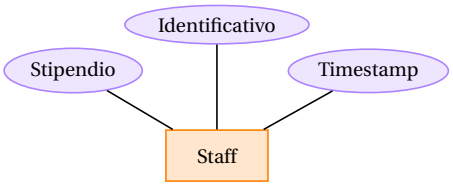
L'entità *Utente* contiene tutte le informazioni riguardo alle persone che usufruiscono di un servizio ospedaliero, sia che siano componenti dello staff che pazienti. Ogni utente è caratterizzato in maniera univoca da un *ID*. Il contenuto degli attributi *email* e *C.F.* (codice fiscale) deve essere unico (non possono esserci due utenti con lo stesso valore nel campo *mail* e/o *C.F.*). L'attributo *Attivo* viene utilizzato per indicare se un utente è attivo oppure no (l'utente ha la possibilità di cancellarsi dall'ospedale, tuttavia non viene eliminata la riga corrispondente dal database, ma viene impostato a 0 l'attributo *attivo*).

1.2.2 Paziente



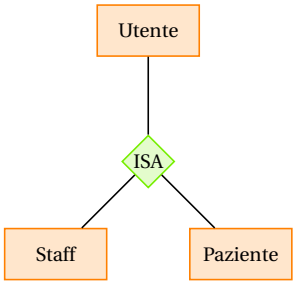
L'entità *Paziente* è una specializzazione della entità *Utente*. Ha come chiave esterna l'*ID* dell'utente al quale si riferisce e presenta anche un attributo *note* per eventuali informazioni aggiuntive.

1.2.3 Staff



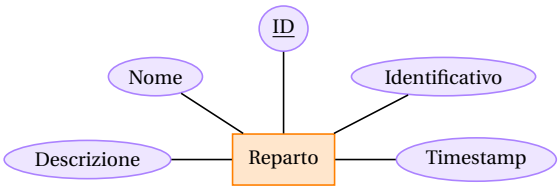
L'entità *Staff*, analogamente ad *Paziente*, è una specializzazione di *Utente*. Ha come chiave esterna l'*ID* dell'utente al quale si riferisce e il reparto di appartenenza. Inoltre presenta un attributo *Identificativo* per specificarne la funzione.

1.2.4 Utente - Paziente - Staff



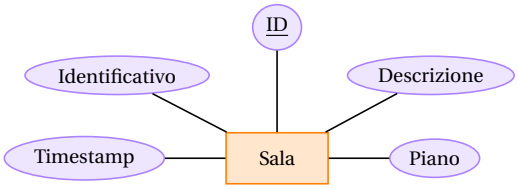
Le entità *Paziente* e *Staff* sono disgiunte: non può esistere un *Utente* nel database che appartenga sia a *Paziente* che *Staff*.

1.2.5 REPARTO



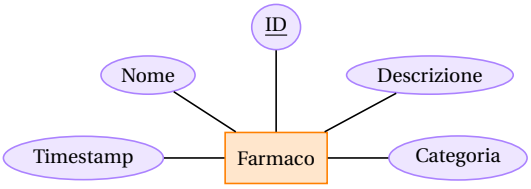
Reparto caratterizza un particolare reparto dell'ospedale (cardiologia, pneumologia, ecc) attraverso l'attributo *Nome*. Tuttavia ogni reparto è caratterizzato anche da un *ID* unico.

1.2.6 SALA



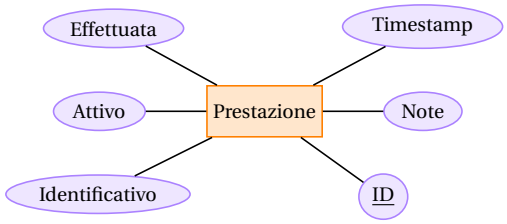
L'entità *Sala* rappresenta le varie sale disponibili nell'ospedale e ha come chiave esterna l'*ID* del reparto alla quale è assegnata.

1.2.7 FARMACO



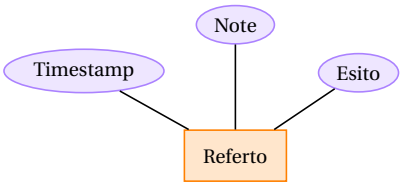
L'entità *Farmaco* rappresenta ciascun farmaco che viene assunto dal paziente (eventualmente anche prima di essere diventato un paziente dell'ospedale). I farmaci possono essere prescritti a seguito di una *Prestazione* attraverso un *Referto*. Ciascun *Farmaco* è caratterizzato da una categoria (salvavita, pressione, ecc) ed include un campo di testo *Descrizione* dove può essere inserita la posologia.

1.2.8 PRESTAZIONE



L'entità *Prestazione* rappresenta una prestazione effettuata all'interno dell'ospedale, sia visita medica che operazione chirurgica. Ogni prestazione è caratterizzata da un *ID* univoco e da un *Identificativo* per distinguerne i vari tipi. L'attributo *Attivo* è stato inserito per discriminare le prestazioni prenotate (in questo caso *Attivo* viene posto a 1) e cancellate (*Attivo* a 0). *Effettuata* viene impostato a 1 se la prestazione è stata effettuata, 0 se invece deve essere ancora effettuata: in entrambi i casi, la prestazione non deve essere stata cancellata (*Attivo* deve essere impostato a 1).

1.2.9 REFERTO



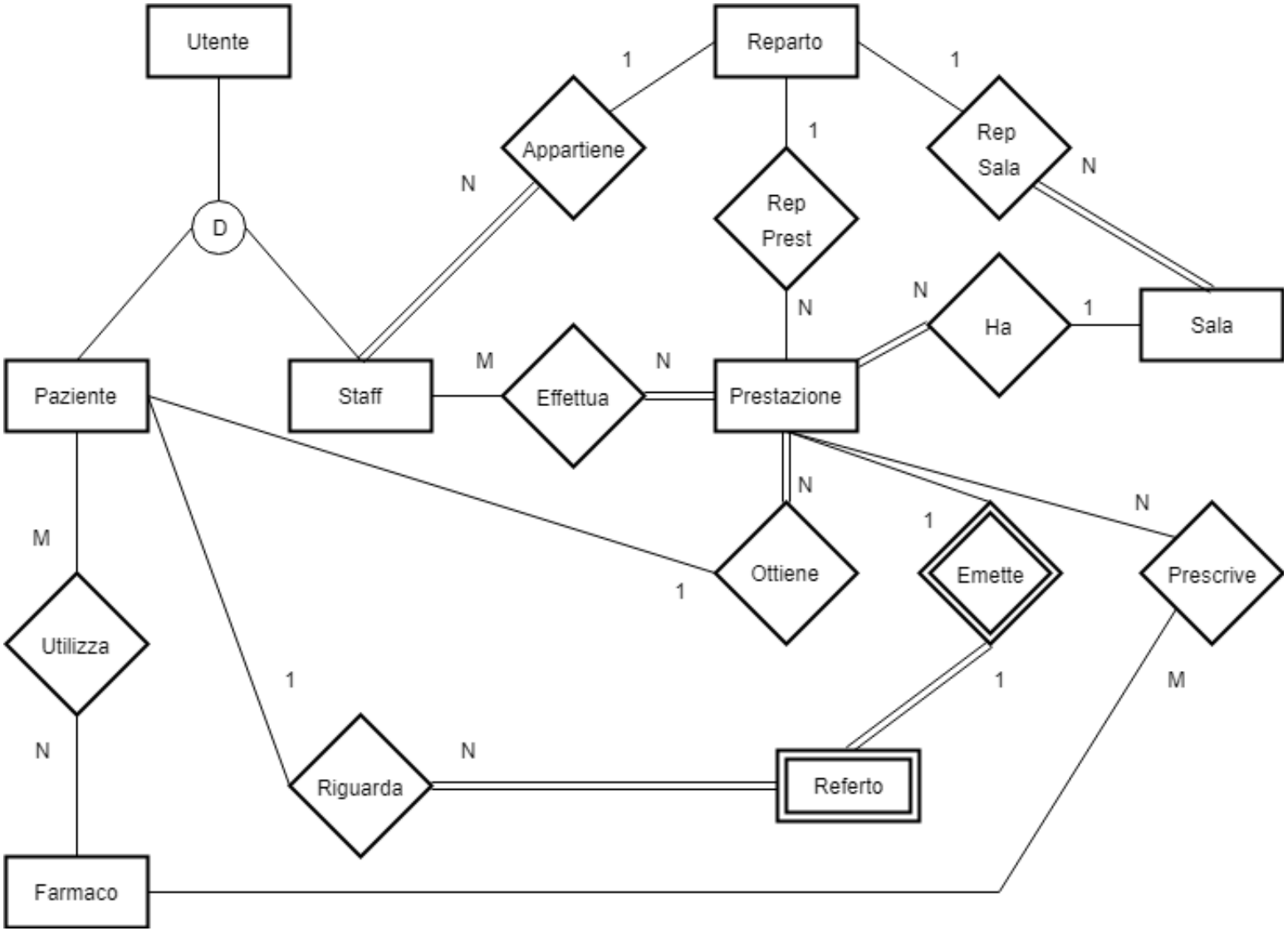
L'entità *Referto* è una entità debole, collegata a prestazione. Infatti non ha chiave primaria ed ha come chiavi esterne l'*ID* del paziente e l'*ID* della prestazione alla quale fa riferimento.

1.3 ASSOCIAZIONI

Le associazioni rappresentano i vari legami che intercorrono tra le varie entità. Sono le seguenti

- Utilizza: associazione *N:M* tra *Paziente* e *Farmaco*: un *Paziente* può utilizzare diversi farmaci e analogamente un *Farmaco* può essere utilizzato da diversi pazienti.
- Riguarda: relazione *1:N* tra *Paziente* e *Referto*: un *Paziente* può avere associati ad esso diversi referti ma uno specifico *Referto* è associato ad un solo paziente. Inoltre un referto deve essere necessariamente essere associato ad un paziente, infatti nello schema ER è legato da un vincolo di partecipazione totale.
- Ottiene: relazione *1:N* tra *Paziente* e *Prestazione*: un *Paziente* può ottenere diverse prestazioni, ma una specifica *Prestazione* deve essere associata univocamente ad un utente, viene quindi rappresentata con un vincolo di partecipazione totale.
- Effettua: relazione *M:N* tra *Staff* e *Prestazione*: un membro di uno *Staff* può effettuare più prestazioni e ciascuna *Prestazione* può essere effettuata da più membri dello staff (si pensi per esempio ad una operazione chirurgica che coinvolge diversi membri dello staff come anestesista e chirurgo). *Prestazione* ha un vincolo di partecipazione totale.
- Appartiene: relazione *N:1* tra *Staff* e *Reparto*: un membro dello *Staff* deve appartenere necessariamente (vincolo di partecipazione totale) ad un solo reparto ma un *Reparto* comprende più membri dello staff.
- Emette: relazione *1:1* tra *Prestazione* e *Referto*: ogni *Prestazione* ha un unico referto ed uno specifico *Referto* è associato ad una sola prestazione. *Referto* è entità debole (non può esistere senza una prestazione) quindi è caratterizzato da un vincolo di partecipazione totale. Essendo una relazione *1 - 1*, si sarebbero potuti includere gli attributi di *Referto* all'interno di *Prestazione*. Tuttavia è stata creata l'entità *Prestazione* per poter semplificare le query e poter risalire, per esempio, in maniera veloce a tutti i referti riguardanti un determinato utente.
- Prescrive: relazione *N:M* tra *Prestazione* e *Farmaco*: una *Prestazione* può descrivere uno o più farmaci e un *Farmaco* può essere prescritto da una prestazione (ricordiamo che nel database possono essere inseriti anche farmaci non prescritti da una prestazione, che l'utente assumeva già in maniera autonoma).

- Ha: relazione *N:1* tra *Prestazione* e *Sala*: una *Prestazione* deve (vincolo partecipazione totale) utilizzare una sola sala ma una *Sala* viene utilizzata per più prestazioni.
- Rep Prest: relazione *N:1* tra *Prestazione* e *Reparto*: una *Prestazione* deve (vincolo di partecipazione totale) essere effettuata in un reparto ma in un *Reparto* possono essere effettuate più prestazioni.
- Rep Sala: relazione *N:1* tra *Sala* e *Reparto*: una *Sala* deve (vincolo di partecipazione totale) essere assegnata ad un reparto mente ad un *Reparto* sono assegnate più sale.


Utente
Reparto
Appartiene
Rep
Sala
Rep
Prest
D
Paziente
Staff
Effettua
Prestazione
Ha
Sala
Ottiene
Emette
Prescrive
Utilizza
Riguarda
Referto
Farmaco
1
N
M

CAPITOLO 2

DA MODELLO ER A MODELLO RELAZIONALE

Dopo aver completato lo schema ER, è necessario mappare le entità e le relazioni sul database. Questa procedura avviene secondo i seguenti passi:

- Traduzione di tipi di entità
  - Traduzione entità forti.
  - Traduzione entità deboli.
- Traduzione di tipi di associazioni binarie.
  - Traduzioni associazioni 1:1.
  - Traduzione associazioni 1:N.
  - Traduzione associazioni N:M.
- Traduzione di attributi multivalore.
- Traduzioni di tipi di associazione N-arie.

2.0.1 TRADUZIONE ENTITÀ FORTI

2.0.2 TRADUZIONE ENTITÀ DEBOLI

2.0.3 TRADUZIONI ASSOCIAZIONI 1:1

2.0.4 TRADUZIONI ASSOCIAZIONI 1:N

2.0.5 TRADUZIONI ASSOCIAZIONI N:M

Per ogni associazione *N:M* è stata creata una nuova tabella nel database, in particolare:

- StaffPrestazione: contiene, come chiavi esterne, le chiavi primarie delle tabelle *Staff* e *Prestazione*.
- PazienteFarmaco: contiene, come chiavi esterne, le chiavi primarie delle tabelle *Paziente* e *Farmaco*.
- PrestazioneFarmaco: contiene, come chiavi esterne, le chiavi primarie delle tabelle *Prestazione* e *Farmaco*.

Capitolo 3

Normalizzazione

CAPITOLO 4

CODICE SQL

4.1 INTRODUZIONE

Come detto nella descrizione, l'intero progetto è stato sviluppato utilizzando il framework *laravel*. Le tabelle del database sono state create utilizzando le `migration`. Per ogni tabella del database è stato eseguito il comando `php artisan make:migration create_table_nomeTabella`. Questo comando genera una classe `migration` all'interno del file `create_table_nomeTabella.php` nella quale sono definiti i metodi `up()` e `down()`. All'interno di `up()` vengono inseriti tutti i comandi per la creazione delle tabelle. Generate le migrations per tutte le tabelle, il comando `php artisan migrate` traduce i comandi specificati nel metodo `up`, in comandi SQL per la creazione delle tabelle. Di seguito viene riportato il codice SQL generato dalle migrations.

4.2 CODICE

```
UTENTE
CREATE TABLE utente (
        id INT AUTO_INCREMENT PRIMARY_KEY,
        nome VARCHAR(255) NOT NULL,
        cognome VARCHAR(255) NOT NULL,
        dataNascita DATE NOT NULL,
        sesso BIT NOT NULL,
        codiceFiscale VARCHAR(255) NOT NULL UNIQUE,
        email VARCHAR(255) NOT NULL UNIQUE,
        password VARCHAR(255) NOT NULL,
        telefono VARCHAR(255) NOT NULL,
        attivo BIT NOT NULL,
        provincia VARCHAR(255) NOT NULL,
        stato VARCHAR(255) NOT NULL,
        comune VARCHAR(255) NOT NULL,
        via VARCHAR(255) NOT NULL,
        numeroCivico INT NOT NULL,
        timestamp TIMESTAMP
);

PAZIENTE
CREATE TABLE paziente (
        id INT FOREGIN_KEY REFERENCES utente(id),
        note TEXT,
        altezza INT NOT NULL,
```

```
        peso INT NOT NULL,
        timestamp TIMESTAMP
);

REPARTO
CREATE TABLE reparto (
id INT PRIMARY_KEY,
    nome VARCHAR(255) NOT_NULL,
    identificativo VARCHAR(255) NOT NULL,
    descrizione TEXT,
    timestamp TIMESTAMP,
);

SALA
CRATE TABLE sala (
        id INT AUTO INCREMENT PRIMARY KEY,
        identificativo VARCHAR(255) NOT NULL,
        idReparto INT FOREGIN KEY REFERENCES reparto(id),
        piano INT,
        timestamp TIMESTAMP
);

STAFF
CREATE TABLE staff (
        id INT FOREGIN_KEY REFERENCES utente(id),
        idReparto INT FOREGIN_KEY REFERENCES utente(idReparto),
        identificativo VARCHAR(255) NOT NULL,
        contenuto TETX NOT_NULL,
        timestamp TIMESTAMP
);

FARMACO
CREATE TABLE farmaco (
        id INT AUTO INCREMENT PRIMARY KEY,
        descrizione TEXT NOT NULL,
        nome VARCHAR(255) NOT NULL,
        categoria VARCHAR(255) NOT NULL,
        timestamp TIMESTAMP
);

PRESTAZIONE
CREATE TABLE prestazione (
        id INT AUTO_INCREMENT PRIMARY_KEY,
        idReparto INT FOREGIN_KEY REFERENCES reparto(idReparto),
        identificativo VARCHAR(255) NOT NULL,
        note TEXT,
        attivo BIT NOT NULL,
        effettuata BIT NOT NULL,
        timestamp TIMESTAMP
);

REFERTO
CREATE TABLE referto (
```

```
        idPrestazione INT FOREGIN_KEY REFERENCES prestazione(id),
        idPaziente INT FOREGIN_KEY REFERENCES prestazione(idPaziente),
        identificativo VARCHAR(255) NOT NULL,
        esito TEXT NOT NULL,
        note TEXT,
        timestamp TIMESTAMP
);
```

Capitolo 5

Query

Capitolo 6

Interfaccia