

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

---

# Realizzazione Database per Ospedale

---

Azzolini Damiano - Bertagnon Alessandro

---



UNIVERSITÀ  
DEGLI STUDI  
DI FERRARA  
- EX LABORE FRUCTUS -

# INDICE

<b>1</b>	<b>Minimondo</b>	<b>1</b>
1.1	Descrizione . . . . .	1
1.2	Entità . . . . .	2
1.2.1	Utente . . . . .	2
1.2.2	Paziente . . . . .	3
1.2.3	Staff . . . . .	3
1.2.4	Utente - Paziente - Staff . . . . .	3
1.2.5	Reparto . . . . .	4
1.2.6	Sala . . . . .	4
1.2.7	Farmaco . . . . .	4
1.2.8	Prestazione . . . . .	4
1.2.9	Referto . . . . .	5
1.2.10	Permesso . . . . .	5
1.3	Associazioni . . . . .	5
<b>2</b>	<b>Da Modello ER a Modello Relazionale</b>	<b>8</b>
2.1	Traduzione entità forti . . . . .	8
2.2	Traduzione entità deboli e specializzazioni . . . . .	10
2.3	Traduzioni associazioni 1:1 . . . . .	10
2.4	Traduzioni associazioni 1:N . . . . .	10
2.5	Traduzioni associazioni N:M . . . . .	11
<b>3</b>	<b>Normalizzazione</b>	<b>12</b>
<b>4</b>	<b>Codice SQL</b>	<b>16</b>
4.1	Introduzione . . . . .	16
4.2	Codice . . . . .	16
<b>5</b>	<b>Query</b>	<b>19</b>
<b>6</b>	<b>Interfaccia</b>	<b>20</b>

## 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** corrispondente 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. Ad ogni utente sono associati dei **permessi**, ovvero azioni che esso può svolgere sulla piattaforma, i permessi di base assegnati a ciascun utente dipendono dalla tipologia dello stesso e sono:

L'utente **PAZIENTE** potrà:

- Registrarsi sulla piattaforma, fare il login e 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 i farmaci nella lista della farmacia.

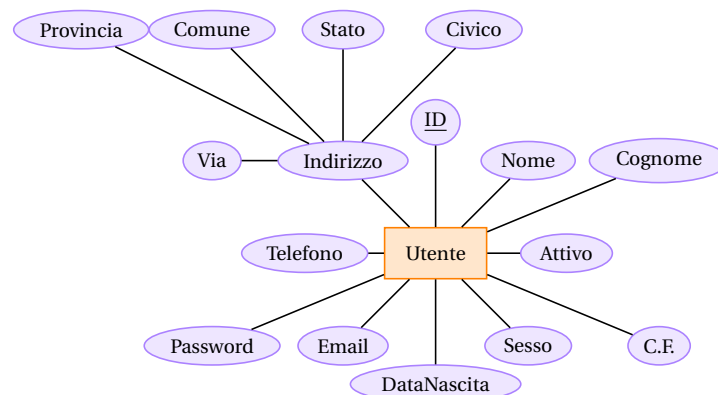
L'utente **AMMINISTRATORE** potrà:

- 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 permessi agli utenti.

## 1.2 ENTITÀ

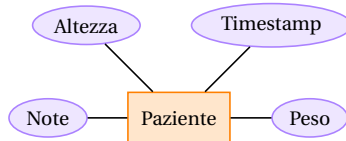
Di seguito vengono analizzate tutte le entità presenti nel database:

### 1.2.1 UTENTE



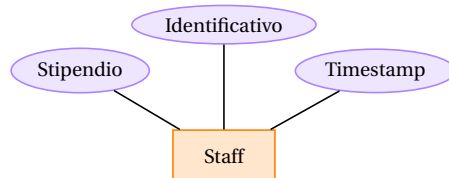
L'entità *Utente* contiene tutte le informazioni riguardo alle persone che usufruiscono di un servizio ospedaliero, sia che siano componenti dello staff che pazienti. Ogni utente è caratterizzato in maniera univoca da un *ID*. Il contenuto degli attributi *email* e *C.F* (codice fiscale) deve essere 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



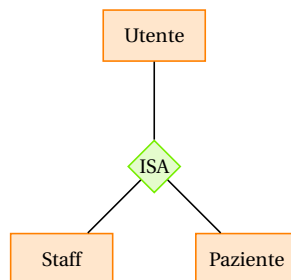
L'entità *Paziente* è una specializzazione della entità *Utente*. Ha come chiave esterna l'*ID* dell'utente al quale si riferisce e presenta anche un attributo *note* per eventuali informazioni aggiuntive.

### 1.2.3 STAFF



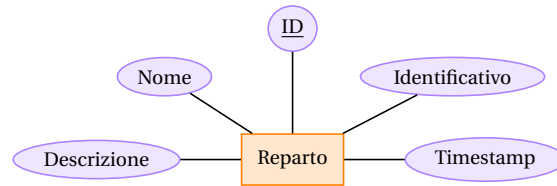
L'entità *Staff*, analogamente ad *Paziente*, è una specializzazione di *Utente*. Ha come chiave esterna l'*ID* dell'utente al quale si riferisce e il reparto di appartenenza. Inoltre presenta un attributo *Identificativo* per specificarne la funzione.

### 1.2.4 UTENTE - PAZIENTE - STAFF



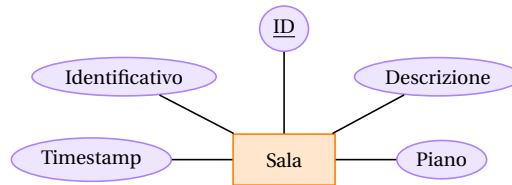
Le entità *Paziente* e *Staff* sono disgiunte: non può esistere un *Utente* nel database che appartenga sia a *Paziente* che *Staff*.

### 1.2.5 REPARTO



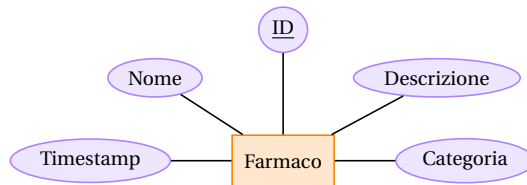
*Reparto* caratterizza un particolare reparto dell'ospedale (cardiologia, pneumologia, ecc) attraverso l'attributo *Nome*. Inoltre ogni reparto è caratterizzato anche da un *ID* unico.

### 1.2.6 SALA



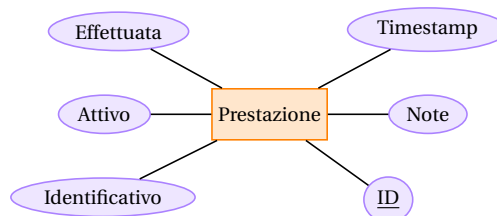
L'entità *Sala* rappresenta le varie sale disponibili nell'ospedale. Ha come chiave esterna l'*ID* del reparto alla quale è assegnata.

### 1.2.7 FARMACO



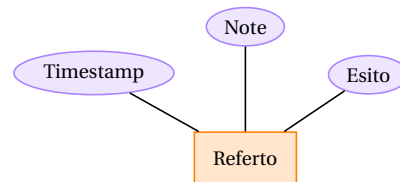
L'entità *Farmaco* rappresenta ciascun farmaco che viene assunto dal paziente. I farmaci possono essere inseriti direttamente dal paziente nel profilo (farmaci assunti regolarmente dovuti a prescrizioni esterne all'ospedale) oppure prescritti a seguito di una *Prestazione* attraverso un *Referto*. Ciascun *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



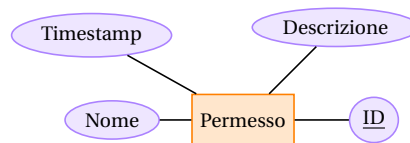
L'entità *Prestazione* rappresenta una prestazione effettuata all'interno dell'ospedale, sia visita medica che operazione chirurgica. Ogni prestazione è caratterizzata da un *ID* univoco e da un *Identificativo* per distinguerne i vari tipi. L'attributo *Attivo* è stato inserito per discriminare le prestazioni 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. *Referto* è stato introdotto come entità debole in quanto non può esistere senza una corrispondente prestazione.

### 1.2.10 PERMESSO



L'entità *Permesso* identifica i permessi sull'applicazione concessi ad un determinato utente.

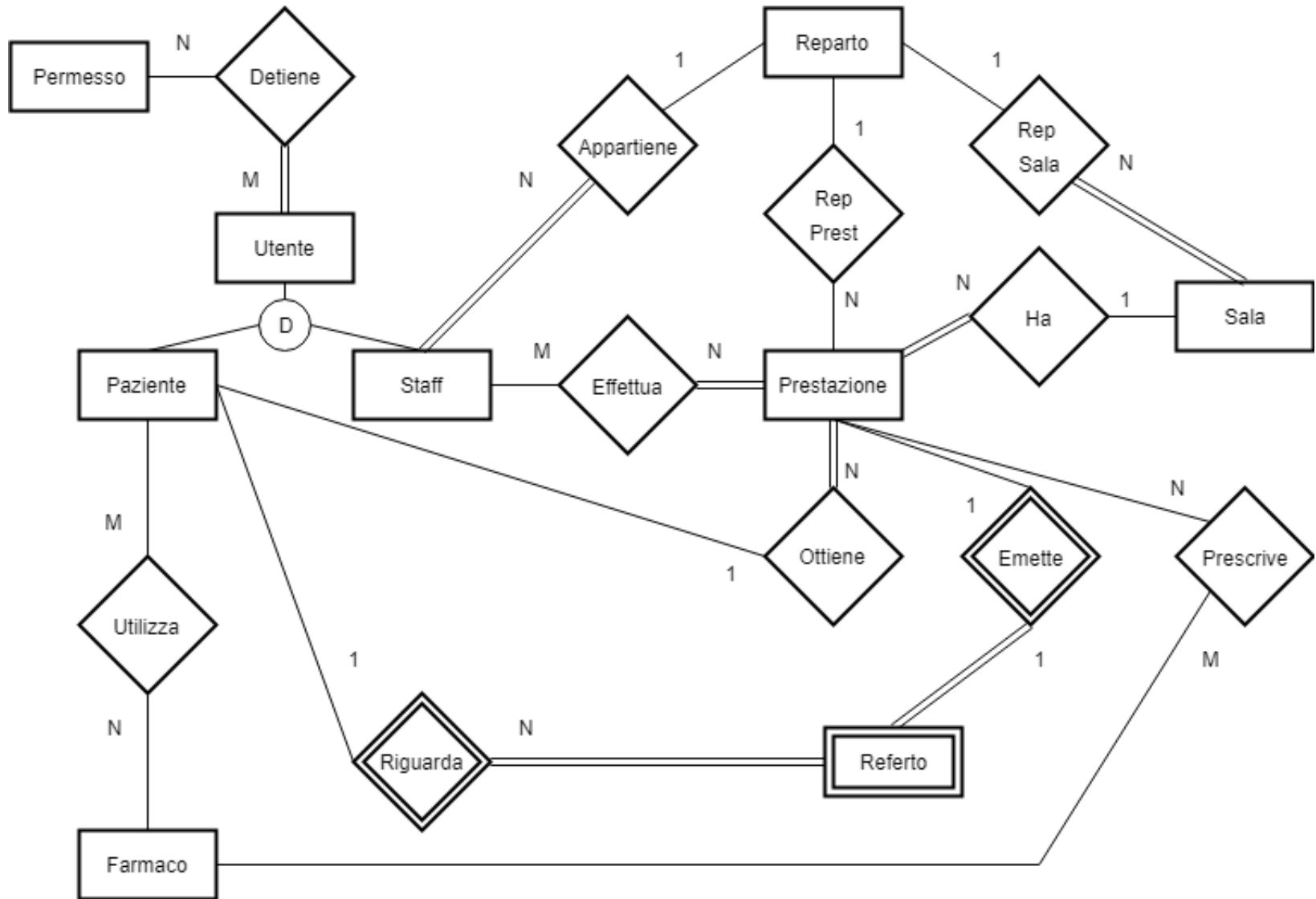
## 1.3 ASSOCIAZIONI

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. Questa associazione è stata introdotta per ridondanza (vedi l'associazione *Emette* tra *Referto* e *Prestazione*)
- 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 mentre ad un *Reparto* sono assegnate più sale.



## 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 e specializzazioni.
- 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.

### 2.1 TRADUZIONE ENTITÀ FORTI

Per ogni tipo di entità forte E nello schema ER, viene costruita una relazione R che contiene tutti gli attributi semplice di E. Di un attributo composto vanno inseriti solamente gli attributi componenti semplici. Come chiave primaria viene scelto uno degli attributi chiavi di E. Le entità forti presenti nello schema sono:

- Utente
- Farmaco
- Reparto
- Sala
- Prestazione
- Permesso

Le chiavi primarie sono evidenziate in giallo, le chiavi esterne in verde e gli attributi unici in azzurro. La loro rappresentazione è la seguente:

## UTENTE

<u>id</u>	nome	cognome	dataNascita	sex	codiceFiscale	email	password	telefono
-----------	------	---------	-------------	-----	---------------	-------	----------	----------

	provincia	stato	comune	via	numeroCivico	created_at	updated_at
--	-----------	-------	--------	-----	--------------	------------	------------

## FARMACO

<u>id</u>	descrizione	nome	categoria	created_at	updated_at
-----------	-------------	------	-----------	------------	------------

## REPARTO

<u>id</u>	nome	identificativo	descrizione	created_at	updated_at
-----------	------	----------------	-------------	------------	------------

## SALA

<u>id</u>	idReparto	identificativo	descrizione	piano	created_at	updated_at
-----------	-----------	----------------	-------------	-------	------------	------------

## PRESTAZIONE

<u>id</u>	idReparto	identificativo	note	attivo	effettuata	created_at
-----------	-----------	----------------	------	--------	------------	------------

## PERMESSO

<u>id</u>	nome	descrizione	created_at	updated_at
-----------	------	-------------	------------	------------

## 2.2 TRADUZIONE ENTITÀ DEBOLI E SPECIALIZZAZIONI

Per ogni tipo di entità debole W dello schema ER con entità proprietaria E, viene costruita una relazione R che ha come attributi, tutti gli attributi semplici di W. Inoltre vengono inseriti come attributi di chiave esterna in R, le chiavi primarie delle relazioni proprietari. La chiave primaria di R è data dalla combinazione della chiave primaria delle varie entità proprietarie e dell'eventuale chiave parziale dell'entità debole W. Nello schema è presente un'unica entità debole: *Referto*

### REFERTO

<u>idPrestazione</u>	<u>idPaziente</u>	esito	note	created_at	updated_at
----------------------	-------------------	-------	------	------------	------------

Gli attributi *idPrestazione* e *idPaziente* sono rispettivamente le chiavi primarie di *Prestazione* e *Paziente*, entità proprietarie di *Referto*. La chiave primaria di *Referto* è quindi la combinazione di *idPrestazione* e *idPaziente* (*Referto* non ha chiave parziale).

Le specializzazioni di *Utente* (*Paziente* e *Staff*) sono rappresentate nel modo seguente:

### PAZIENTE

<u>id</u>	note	altezza	peso	created_at	updated_at
-----------	------	---------	------	------------	------------

### STAFF

<u>id</u>	idReparto	identificativo	stipendio	created_at	updated_at
-----------	-----------	----------------	-----------	------------	------------

dove *id* rappresenta, sia per *Paziente* che per *Staff*, l'identificativo dell'utente (chiave primaria della tabella *Utente*). In *Staff* è stata anche inserita, come chiave esterna (a causa della relazione *N:1* che lega *Staff* a *Reparto*) la chiave primaria di *Reparto* (i.e. *idReparto*).

## 2.3 TRADUZIONI ASSOCIAZIONI 1:1

L'unica relazione *1:1* è *Emette*, che collega *Prestazione* a *Referto*. Essendo *Referto* una entità debole collegata a *Prestazione*, in *Referto* viene inserita la chiave primaria di *Referto*.

## 2.4 TRADUZIONI ASSOCIAZIONI 1:N

Per ogni associazione R binaria del tipo *1:N* nello schema ER, vengono individuate le due relazioni corrispondenti alle due entità partecipanti. Viene quindi inserita nel lato *N*, la chiave primaria della relazione lato *1*. In questo caso sono quindi state inserite le seguenti chiavi:

- *id* di *Paziente* in *Referto*.

- *id* di *Paziente* in *Prestazione*.
- *id* di *Reparto* in *Staff*.
- *id* di *Reparto* in *Sala*.
- *id* di *Sala* in *Prestazione*.

## 2.5 TRADUZIONI ASSOCIAZIONI N:M

Per ogni associazione binaria R del tipo *N:M*, viene costruita una nuova relazione S che la rappresenta. Come attributi di chiave esterna si S, vengono inserite le chiavi primarie delle relazioni rappresentanti le entità partecipanti: la combinazione di queste due chiavi forma la chiave primaria di S. 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*.
- *PermessoUtente*: contiene, come chiavi esterne, le chiavi primarie delle tabelle *Permesso* e *Utente*.

STAFF\_PRESTAZIONE

<u>idStaff</u>	<u>idPrestazione</u>	created_at	updated_at
----------------	----------------------	------------	------------

PAZIENTE\_FARMACO

<u>idPaziente</u>	<u>idFarmaco</u>	created_at	updated_at
-------------------	------------------	------------	------------

FARMACO\_PRESTAZIONE

<u>idFarmaco</u>	<u>idPrestazione</u>	created_at	updated_at
------------------	----------------------	------------	------------

PERMESSO\_UTENTE

<u>id</u>	idPermesso	idUtente	created_at	updated_at
-----------	------------	----------	------------	------------

Per ogni associazione *N:M* viene quindi creata una nuova tabella nel database.

# CAPITOLO 3

## NORMALIZZAZIONE

Il processo di normalizzazione si basa su una serie di test che verificano se uno schema di relazione soddisfa una determinata *forma normale*. Esistono diversi tipi di forma normale:

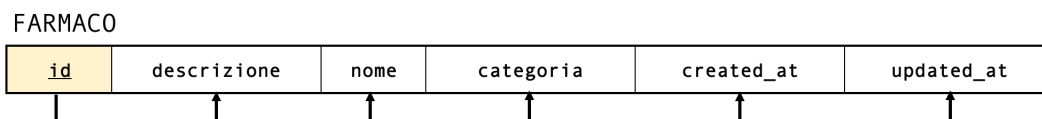
- Prima forma normale (1NF).
- Seconda forma normale (2NF).
- Terza forma normale (3NF).
- Forma normale di Boyce e Codd.
- Quarta forma normale (4NF).
- Quinta forma normale.

L'obiettivo della normalizzazione dei dati è quello di minimizzare la ridondanza e le anomalie dovute all'inserimento, cancellazione o modifica dei dati nel database. Questa caratteristica si ottiene attraverso una analisi degli schemi forniti e una decomposizione degli schemi che non soddisfano certe condizioni, in schemi più piccoli (che verificano le proprietà desiderate). In questa applicazione, si raggiunge solamente la terza forma normale.

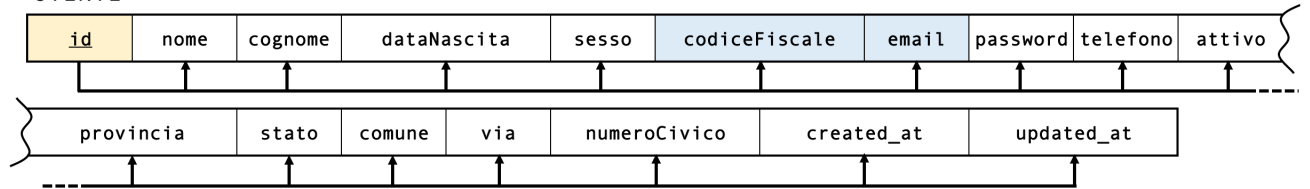
La *prima forma normale* richiede che il dominio di un attributo contenga solamente valori indivisibili e che il valore di un qualsiasi attributo in una tupla sia un valore singolo del dominio. Lo schema presentato è già in prima forma normale.

Per soddisfare la *seconda forma normale*, nello schema di relazione R, ogni attributo non primo di R (quindi non fa parte di una chiave candidata) deve dipendere funzionalmente in modo completo dalla chiave primaria di R. La definizione di dipendenza funzionale completa è la seguente: una dipendenza funzionale  $X \rightarrow Y$  si dice *dipendenza funzionale completa* se la rimozione di un qualsiasi attributo A da X, comporta che la dipendenza funzionale non sussista più. Per ottenere la seconda forma normale, si decompone la relazione principale R in un certo numero di relazioni nelle quali gli attributi non primi sono associati solamente alla parte della chiave primaria da cui sono funzionalmente dipendenti in modo completo.

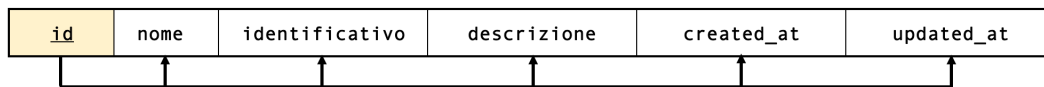
Uno schema di relazione R è in terza forma normale se soddisfa la seconda forma normale e nessun attributo non primo di R dipende in modo transitivo dalla chiave primaria.



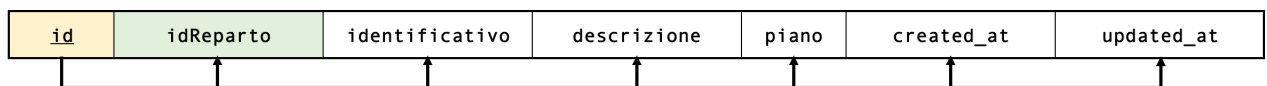
### UTENTE



### REPARTO



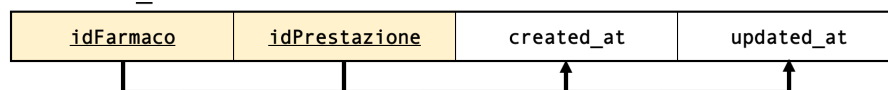
### SALA



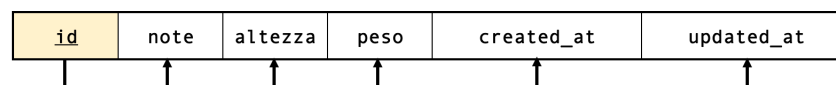
### PRESTAZIONE



### FARMACO\_PRESTAZIONE



### PAZIENTE





PAZIENTE\_FARMACO

<u>idPaziente</u>	<u>idFarmaco</u>	created_at	updated_at
-------------------	------------------	------------	------------

PERMESSO

<u>id</u>	nome	descrizione	created_at	updated_at
-----------	------	-------------	------------	------------

PERMESSO\_UTENTE

<u>id</u>	idPermesso	idUtente	created_at	updated_at
-----------	------------	----------	------------	------------

REFERTO

<u>idPrestazione</u>	<u>idPaziente</u>	esito	note	created_at	updated_at
----------------------	-------------------	-------	------	------------	------------

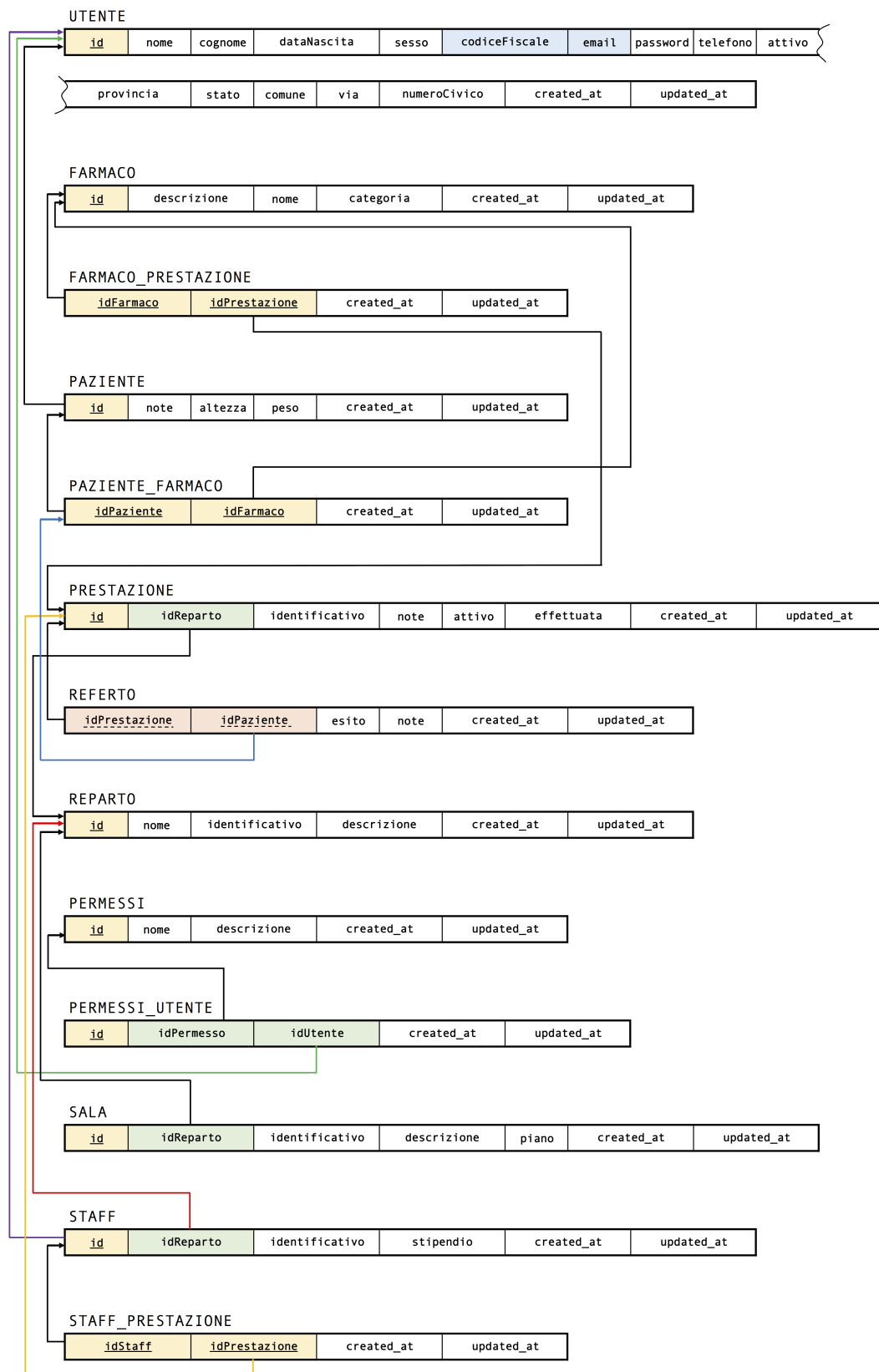
STAFF

<u>id</u>	idReparto	identificativo	stipendio	created_at	updated_at
-----------	-----------	----------------	-----------	------------	------------

STAFF\_PRESTAZIONE

<u>idStaff</u>	<u>idPrestazione</u>	created_at	updated_at
----------------	----------------------	------------	------------

Di seguito lo schema completo in *terza forma normale*.



# CAPITOLO 4

## CODICE SQL

### 4.1 INTRODUZIONE

L'intero progetto è stato sviluppato utilizzando il framework php *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 FOREIGN_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 FOREIGN KEY REFERENCES reparto(id),
        piano INT,
        timestamp TIMESTAMP
    );

```

#### STAFF

```

CREATE TABLE staff (
        id INT FOREIGN_KEY REFERENCES utente(id),
        idReparto INT FOREIGN_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 FOREIGN_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 FOREIGN_KEY REFERENCES prestazione(id),
        idPaziente INT FOREIGN_KEY REFERENCES prestazione(idPaziente),
        identificativo VARCHAR(255) NOT NULL,
        esito TEXT NOT NULL,
        note TEXT,
        timestamp TIMESTAMP
    );

PERMESSO
CREATE TABLE permesso (
    id INT AUTO_INCREMENT PRIMARY_KEY,
    nome VARCHAR(255) NOT NULL,
    descrizione TEXT NOT NULL,
    timestamp TIMESTAMP
);

STAFF_PRESTAZIONE
CREATE TABLE staff_prestazione (
    idPrestazione INT FOREIGN_KEY REFERENCES prestazione(id),
    idStaff INT FOREIGN_KEY REFERENCES staff(id),
    timestamp TIMESTAMP
);

PAZIENTE_FARMACO
CREATE TABLE paziente_farmaco (
    idPaziente INT FOREIGN_KEY REFERENCES paziente(id),
    idFarmaco INT FOREIGN_KEY REFERENCES farmaco(id),
    timestamp TIMESTAMP
);

FARMACO_PRESTAZIONE
CREATE TABLE farmaco_prestazione (
    idPrestazione INT FOREIGN_KEY REFERENCES prestazione(id),
    idFarmaco INT FOREIGN_KEY REFERENCES farmaco(id),
    timestamp TIMESTAMP
);

```

## CAPITOLO 5

# QUERY

La parte relativa alle query verrà inserita non appena il sistema sarà terminato.

## CAPITOLO 6

# INTERFACCIA

La parte relativa all'interfaccia verrà inserita non appena il sistema sarà terminato.