Università degli Studi di Salerno

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

Progetto Basi di Dati

Piattaforma di Cartelle Cliniche Elettroniche (CCE)

Tozza Gennaro Carmine
Matricola: 0512120382

Anno Accademico 2024-2025

Indice

Capitolo 1

Fase 1

1.1 Descrizione della realtà di interesse

[1] La realtà che si vuole presentare riguarda la gestione di una **piattaforma CCE.**

La **Cartella Clinica Elettronica (CCE)** costituisce un'evoluzione della **Cartella Clinica Cartacea (CCC)** ovvero è lo strumento per la gestione organica e strutturata dei dati, riferiti alla **storia clinica** di un paziente, che garantisce il **supporto** dei processi clinici (diagnostico-terapeutici) e assistenziali nei singoli episodi di cura e che favorisce la **continuità** di cura del paziente tra diversi episodi di cura, afferenti alla stessa struttura ospedaliera, mediante la condivisione e il recupero dei dati clinici in essi registrati.

La **Cartella Clinica** viene definita come **"diario diagnostico-terapeutico in cui sono annotati i dati anagrafici ed anamnestici del paziente, gli esami obiettivi, di laboratorio e specialistici, le terapie praticate, nonché l'andamento, gli esiti e gli eventuali postumi della malattia".**

La cartella clinica, come **insieme di dati, informazioni** e atti, svolge un ruolo di **importanza centrale** all'interno delle strutture sanitarie poiché è uno strumento di lavoro per la raccolta delle informazioni relative alle attività assistenziali.

Il **progressivo processo di informatizzazione del sistema sanitario** risponde alle crescenti necessità di memorizzazione, elaborazione e trasmissione dei dati clinici attraverso l'uso dei calcolatori e delle reti telematiche.

Alla luce di tutto ciò si fa **più forte** l'esigenza di trattare tali dati in un **lin-**

[1] Fonte:https://etd.adm.unipi.it/theses/available/etd-11282013-094013/unrestricted/definitiva_Lanciani.pdf

guaggio informatico attraverso l'uso di una cartella elettronica dove è lo stesso **database** creatosi a fornire la raccolta dati.

1.2 Specifiche complete che descrivono in modo compiuto la realtà di interesse

Si vuole progettare una base di dati per la **gestione** di una **piattaforma di cartelle cliniche elettroniche**.

L'accesso alla piattaforma si effettua tramite la creazione di un **account** che prevede un id, nome, cognome, data e luogo di nascita, eta, un email, un username, una password e un numero di telefono.

Ogni **medico** è caratterizzato da: id, genere, specializzazione, il numero di pazienti che segue, la sua **carriera** e il **servizio** che offre.

Un medico può avere **più carriere**, ciò serve a tenere traccia delle attività storiche, consentendo di tracciare la cronologia professionale del medico.
Prevede un id, ruolo, data_inizio, data_fine, l'ospedale o l'organizzazione presso cui ha lavorato (un luogo), una breve descrizione delle attività.

Mentre il **servizio** contiene informazioni dettagliate sui servizi offerti da un medico, consentendo flessibilità nell'aggiungere altre informazioni in base alle necessità.

In questo modo, un medico può offrire **più servizi** e ogni servizio può avere una descrizione dettagliata.
Un servizio, infatti, è composto da id, nome del servizio, descrizione del servizio e in più le lingue in cui viene offerto.

In particolare, ogni medico dispone di un'email, per accedere alla piattaforma, che gli consente di eseguire operazioni aggiuntive rispetto a quelle disponibili per i pazienti.

Mentre del **paziente** si tiene traccia dell'id e del codice fiscale.

Il **paziente** può avere una **cartella clinica** ed eventualmente prenotare **uno o più appuntamenti** di cui viene specificato l'id, la data, l'ora, il luogo.

Il medico, quindi tramite appuntamento, visita i pazienti e può aggiornare, modificare o inserire (se è la prima volta) la cartella clinica di quest'ultimi sulla piattaforma.

La **cartella clinica** prevede un id, informazioni sulla storia passata del paziente, farmaci e trattamenti prescritti, risultati di esami di laboratorio, procedure diagnostiche, come elettrocardiogramma (ECG) o colonscopia e osservazioni del medico.

In relazione ai risultati degli esami di laboratorio, il medico può commissionare **uno o più test** e si tiene traccia oltre che dell'id e degli esiti anche il tipo di test e la data in cui viene effettuato.

A fine appuntamento vengono generati **uno o più pagamenti** in base ai servizi forniti e possono includure anche addebiti dei risultati di laboratorio.

Di quest'ultimi si sanno l'id, l'importo, il tipo di pagamento se in contanti o con carte, la causale ovvero il motivo del pagamento e la data in cui viene realizzato.

N.B: Ogni identificativo (id) è univoco.

1.3 Glossario dei termini

Account → identità di un utente presso la piattaforma, con la creazione del quale si ha accesso alla piattaforma.

Paziente → account che visualizza la cartella clinica, i test del laboratorio, prenotare un appuntamento con il medico e infine effettuare e visualizzare i pagamenti.

Medico → account che inserisce, modifica e aggiorna la cartella clinica del paziente, commissiona i test in laboratorio e può vedere il loro calendario di appuntamenti.

Carriera → storico relativo alle attività del medico.

Servizio → informazioni dettagliate sullo stesso.

Cartella Clinica → contiene il "diario diagnostico-terapeutico" del paziente.

Test → informazioni riguardanti gli esami di laboratorio.

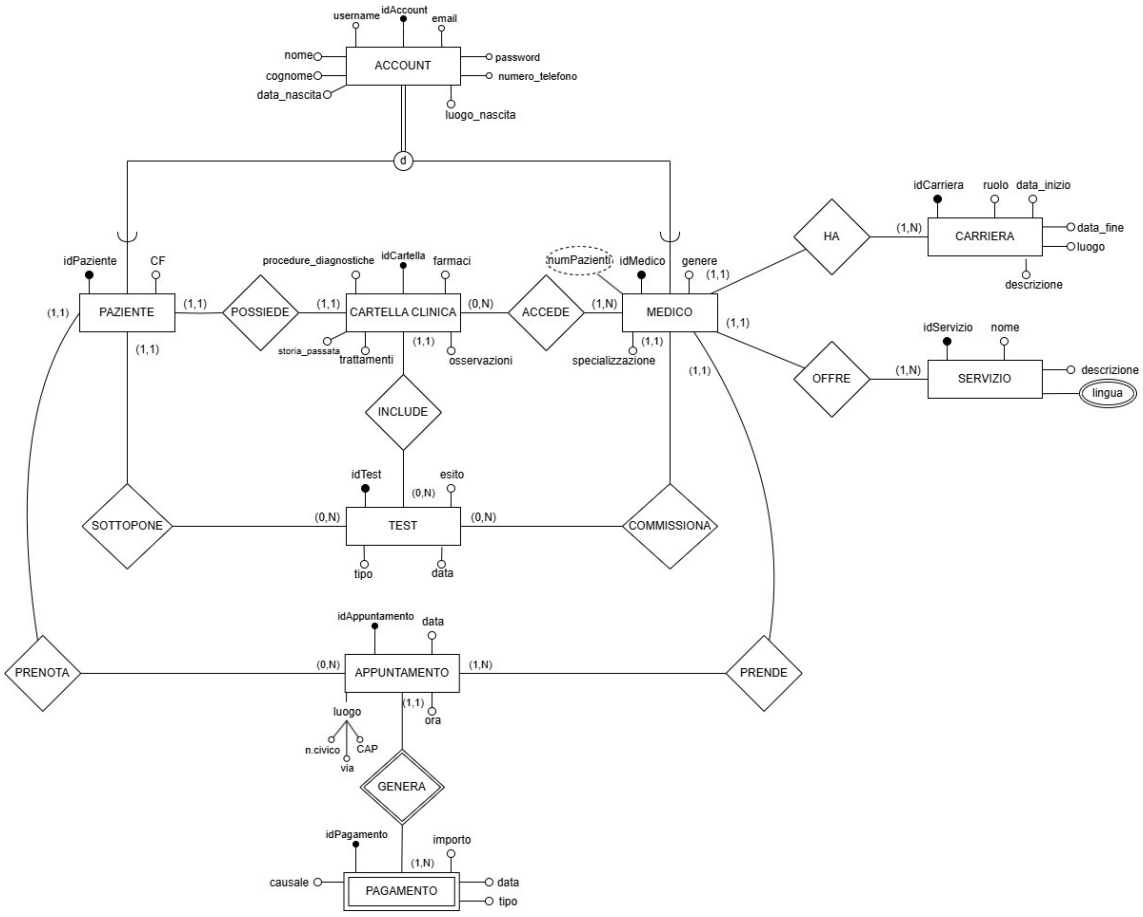
Appuntamento → informazioni riguardanti l'appuntamento tra il paziente e il medico.

Pagamento → importo che il paziente deve pagare per la visita ricevuta.

Capitolo 2

Fase 2

2.1 Schema ER/EER



2.2 Dizionario delle entità

Entità	Descrizione	Attributi	Identificatore
Account	Identità di un utente sulla piattaforma	idAccount, nome, cognome, data_nascita, luogo_nascita, email, username, password, numero_telefono	idAccount
Paziente	Rappresenta il paziente che utilizza la piattaforma	idPaziente, codice_fiscale (CF)	idPaziente
Medico	Rappresenta il medico che utilizza la piattaforma	idMedico, genere, specializzazione, numPazienti	idMedico
Carriera	Registra i dettagli della carriera del medico	idCarriera, ruolo, data_inizio, data_fine, luogo, descrizione	idCarriera
Servizio	Rappresenta i servizi offerti da un medico	idServizio, nome, descrizione, lingua	idServizio
Cartella Clinica	Memorizza le informazioni mediche del paziente	idCartella, storia_passata, farmaci, trattamenti, procedure_diagnostiche, risultati_lab, osservazioni	idCartella
Test	Rappresenta gli esami di laboratorio richiesti per un paziente	idTest, esito, tipo, data	idTest
Appuntamento	Rappresenta una visita tra un paziente e un medico	idAppuntamento, data, ora, luogo	idAppuntamento
Pagamento	Rappresenta il pagamento per servizi o test	idPagamento, importo, data, tipo, causale	idPagamento

L'entità "Account" rappresenta un tipo **generale** di utente sulla piattaforma e può essere suddivisa in sottogruppi in base a caratteristiche specifiche dei suoi utenti.
Questi sottogruppi, detti **sottoclassi**, sono Paziente e Medico.
Il legame tra Account e le sue sottoclassi è noto come **relazione classe/sottoclasse o relazione IS-A.**

Ciascuna sottoclasse **eredita** gli attributi comuni della superclasse Account (come email, username e password etc.) ma aggiunge anche **attributi specifici**, come il codice fiscale per il Paziente e la specializzazione per il Medico.

Il vincolo di **disgiunzione** specifica che le sottoclassi di una specializzazione devono essere mutualmente esclusive, ovvero disgiunte.

Ciò significa che un'entità Account può appartenere solo a una delle sottoclassi definite: **un account può rappresentare esclusivamente un Paziente o un Medico, ma non entrambi contemporaneamente.**

"numPazienti" è un attributo **ridondante**.

"luogo" è un attributo **composto**, diviso nelle sottoparti: n.civico, via e CAP.

"Pagamento" è un'**entità debole**.

Le entità deboli dipendono da un'altra entità (detta **"possessore di identificazione"**) per essere indentificate in modo univoco, in combinazione con alcuni valori dei suoi attributi.

La relazione tra un'entità debole e il suo possessore è chiamata **relazione di identificazione**, ed è caratterizzata da un vincolo di partecipazione totatale: ogni entità debole deve essere associata a un'entità specifica del possessore.

In questo caso, "Pagamento" è un'entità debole che dipende dall'entità "Appuntamento".

L'attributo "lingua" è **multivalued**, cioè può avere un insieme di valori per la stessa entità.
In questo caso un servizio può essere offerto in una o più lingue.

2.3 Dizionario delle relazioni

Relazione	Descrizione	Entità coinvolte	Attributi
Possiede	Un paziente possiede una cartella clinica	Paziente(1,1), Cartella Clinica (1,1)	/
Accede	Un medico accede alle cartelle cliniche	Medico(1,N), Cartella Clinica (1,N)	/
Ha	Un medico ha una o più carriere	Medico (1,N), Carriera (1,1)	/
Offre	Un medico offre uno o più servizi	Medico (1,N), Servizio (1,1)	/
Sottopone	Un paziente si può sottoporre a dei test	Paziente (0,N), Test (1,1)	/
Commissiona	Un medico può commissionare dei test	Medico (0,N), Test (1,1)	/
Include	Una cartella clinica include dei test	Cartella Clinica (0,N), Test (1,1)	/
Prenota	Un paziente può prenotare degli appuntamenti con un medico	Paziente (0,N), Appuntamento (1,1)	/
Prende	Un medico prende un appuntamento con un paziente	Medico(1,N), Appuntamento(1,1)	/
Genera	Ogni appuntamento genera uno o più pagamenti	Appuntamento(1,N), Pagamento(1,1)	/

2.4 Vincoli dello schema non esprimibili dal modello ER/EER

Oltre ciò che è deducibile dallo schema EER, si tenga conto dei seguenti vincoli:

- Le date devono essere coerenti cronologicamente;
- L'attributo "osservazioni" dell'entità Cartella Clinica deve avere un numero di caratteri minore o uguale a 1.000;
- L'attributo "codice_fiscale" di Paziente deve avere 16 caratteri mentre l'attributo "numero_telefono" di Account deve averne 10;
- L'attributo "genere" di Medico deve essere o "M" o "F";

2.5 Definizione delle procedure per la gestione della base di dati

2.5.1 Tavola dei volumi

Si definisce di seguito la tavola dei volumi della base di dati.

Concetto	**Tipo**	**Volume**
Account	E	110
Paziente	SE	100
Medico	SE	10
Cartella Clinica	E	100
Appuntamento	E	120
Pagamento	E	180
Test	E	60
Carriera	E	30
Servizio	E	100
Possiede	R	100
Accede	R	120
Ha	R	100
Offre	R	100
Sottopone	R	60
Commissiona	R	60
Include	R	60
Prenota	R	120
Prende	R	120
Genera	R	180

2.5.2 Tavola delle operazioni

Si definisce di seguito la tavola delle operazioni per la gestione dei dati memorizzati nella base di dati.

n°	Operazione	Tipo	Frequenza
1	Inserire un nuovo account	I	5/mese
2	Inserire una cartella clinica per un paziente	I	20/mese
3	Visualizzare la cartella clinica di un paziente	I	10/mese
4	Inserire un nuovo paziente	I	10/giorno
5	Aggiornare la cartella clinica con nuovi trattamenti/test	I	1/anno
6	Inserire un nuovo appuntamento	I	2/mese
7	Selezionare gli appuntamenti per un paziente o medico in una spefica data	I	1/mese
8	Inserire un nuovo pagamento	I	1/mese
9	Generare il riepilogo dei pagamenti per un paziente	I	2/anno
10	Commissionare un test di laboratorio	I	3/mese
11	Selezionare i risultati di test di laboratorio per un paziente	I	2/mese
12	Inserire una nuova carriera per un medico	B	6/anno
13	Selezionare il numero di pazienti che segue un medico	I	1/giorno
14	Rimuovere un appuntamento in una certa data	I	4/mese

Capitolo 3

Fase 3

3.1 Analasi delle ridondanze

Il dato ridondante è l'attributo **"numPazienti"** dell'entità Medico.
Infatti, sarebbe possibile ottenere il numero dei pazienti che segue un medico attraverso il conto delle partecipazioni di un determinato Medico nella relazione "Medico accede Cartella Clinica", poiché una cartella clinica è associata ad un unico paziente.

Supponendo che l'attributo ridondante abbia un peso di 4 byte, essendo un normale intero, e considerato che il volume dell'entità Medico è uguale a 10, il dato andrebbe ad occupare uno spazio totale di circa **40 byte**.

Per decidere se mantenere o meno il dato ridondante è necessario calcolare, per le operazioni che lo coinvolgono, la differenza nel numero di accessi con e senza quest'ultimo.

3.1.1 Tavola degli accessi

Operazione coinvolte: Op4, Op13.
Si ricorda che 1L = 1, 1S = 2.

Operazione 4

Calcolo con ridondanza				**Calcolo senza ridondanza**			
Tabella	**Tipo**	**Accessi**	**Tipo accessi**	**Tabella**	**Tipo**	**Accessi**	**Tipo accessi**
Paziente	E	1	S	Paziente	E	1	S
Cartella Clinica	E	1	S	Cartella Clinica	E	1	S
Accede	R	12	L				
Medico	E	12	L				
Medico	E	12	S				
Totale $14S + 24L = (28 + 24) \times 10 = 520 accessi$				**Totale** $2S = 4 \times 10 = 40 accessi$			

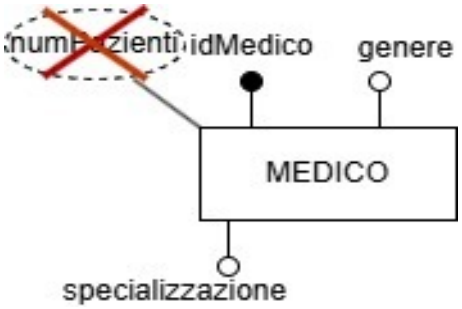
media medici che accedono ad una cartella clinica: 120/10 = 12.

Operazione 13

Calcolo con ridondanza				Calcolo senza ridondanza			
Tabella	**Tipo**	**Accessi**	**Tipo accessi**	**Tabella**	**Tipo**	**Accessi**	**Tipo accessi**
Medico	E	1	L	Medico	E	1	L
				Accede	R	12	L
Totale $1L = 1 \times 1 = 1accessi$				**Totale** $13L = 13 \times 1 = 13accessi$			

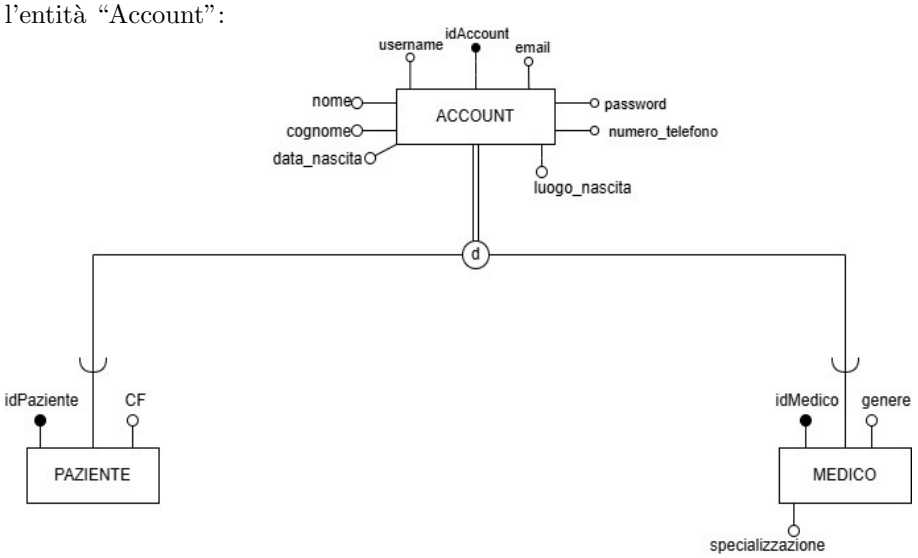
Totale accessi con ridondanza = 521 + 40 byte
Totale accessi senza ridondanza = 53

Dato il minor numero di accessi e lo spreco di spazio pari a 40 byte, è più efficiente scegliere di **non mantenere** il dato ridondante "numPazienti ".



3.2 Eliminazione delle gerarchie

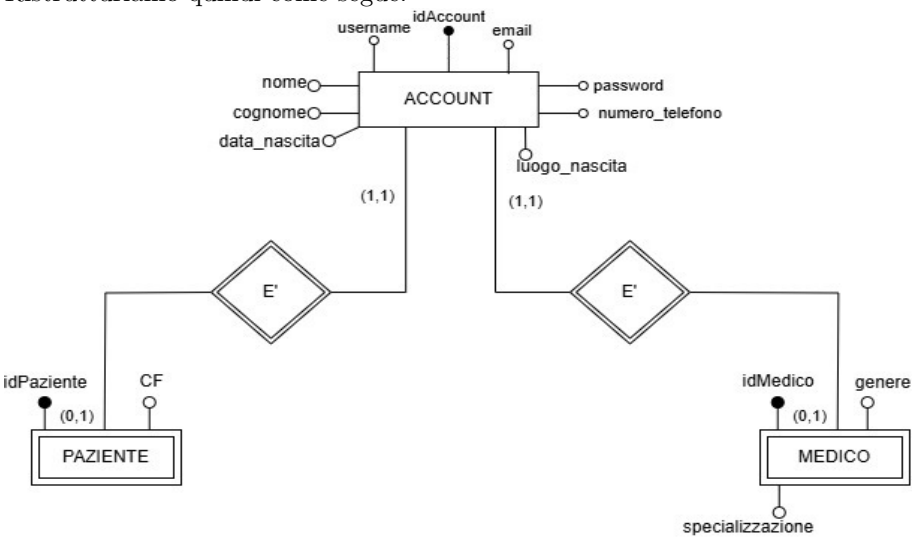
Nello schema inizialmente elaborato, è presente la seguente specializzazione dell'entità "Account":



In questa fase di progettazione logica, è necessario individuare un metodo efficace di ristrutturazione che permetta l'eliminazione di questa gerarchia.

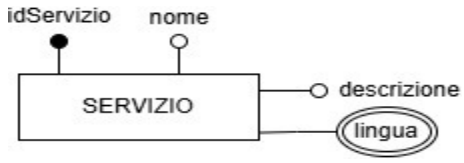
La scelta effettuata è la sostituzione della generalizzazione con associazioni. Questo approccio è applicabile quando ci sono operazioni che fanno distinzione tra entità padre ed entità figlie. Assenza di valori nulli e incremento degli accessi.

Ristrutturiamo quindi come segue:



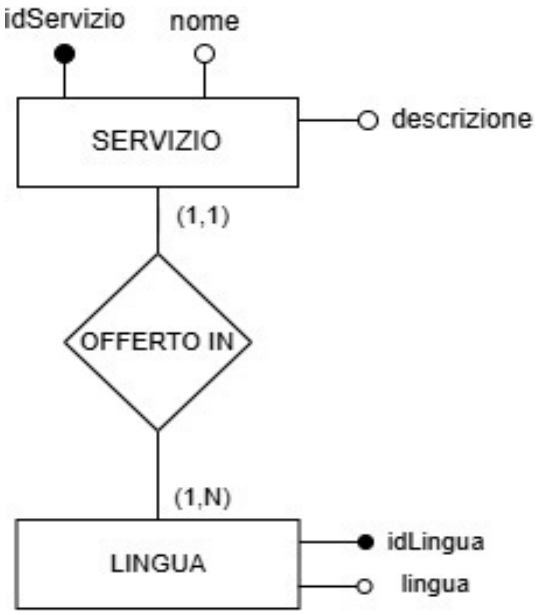
3.3 Eliminazione dell'attributo multivalore

Nello schema inizialmente elaborato, compare un attributo multivalore:



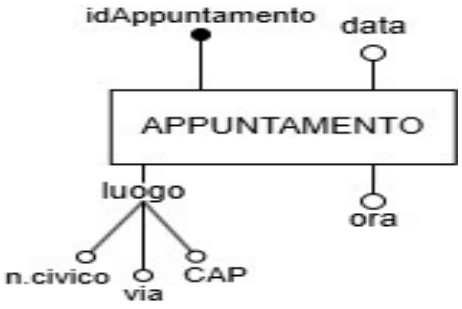
Tale forma di attributo va risolto in maniera differente in fase di progettazione logica.

Si sceglie quindi di definire una nuova entità "Lingua", in relazione con l'entità "Servizio":

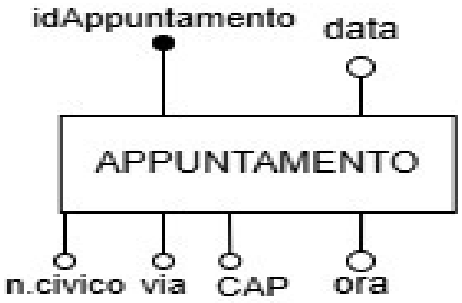


Eliminazione attributo composto

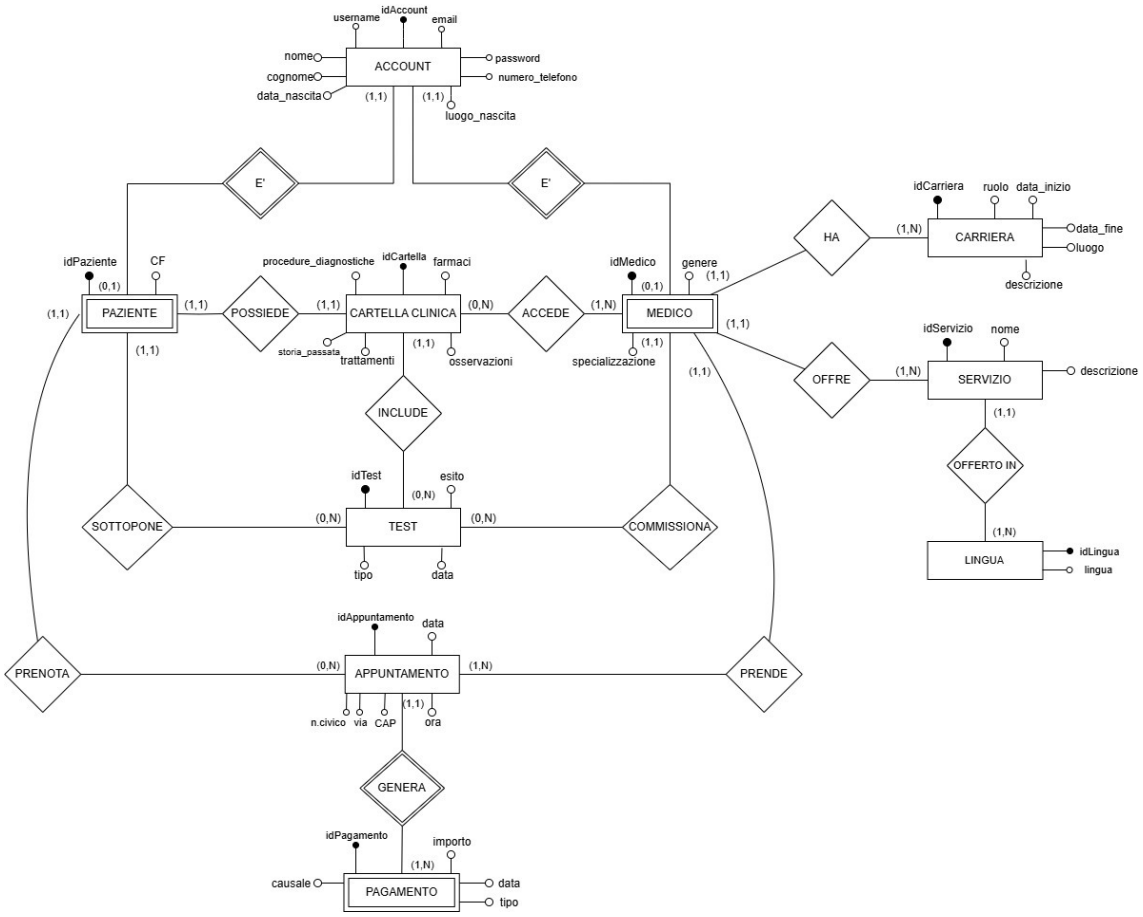
Nello schema inizialmente elaborato, compare un attributo composto:



Anche in questo caso tale forma di attributo va risolto in maniera differente in fase di mapping.
Per questo si decide di ristrutturare come segue:



3.4 Schema EER ristrutturato



3.5 Schema relazionale

Si procede al mapping della base di dati:

Account(idAccount, email, username, nome, cognome, data_nascita, password, numero_telefono, luogo_nascita)
Medico(idMedico, idAccount↑, genere, specializzazione)
Accede(idMedico↑,idCartella↑)
Cartella Clinica(idCartella, procedure_diagnostiche, farmaci, storia_passata, trattamenti, osservazioni, idPaziente↑)
Paziente(idPaziente, idAccount↑,CF)
Carriera(idCarriera, ruolo, data_inizio, data_fine, luogo, descrizione, idMedico↑)
Servizio(idServizio, nome, descrizione, idMedico↑)
Lingua(idLingua, lingua, idServizio↑)
Test(idTest, esito, tipo, data, idMedico↑, idPaziente↑, idCartella↑)
Appuntamento(idAppuntamento, data, n.civico, via, CAP, ora, idMedico↑, idPaziente↑)
Pagamento(idPagamento, idAppuntamento↑, importo, data, tipo, causale)

3.6 Normalizzazione

Il database si presenta già normalizzato.

È infatti in prima forma normale in quanto tutti gli attributi sono atomici.

È in seconda forma normale perché, oltre ad essere già in 1NF, quando è presente una chiave primaria composta da più attributi tutte le dipendenze funzionali che la riguardano sono piene e non parziali.

È in terza forma normale perché, oltre ad essere già in 2NF, in tutte le tabelle non sono presenti dipendenze transitive fra attributi non chiave e la chiave primaria.

Capitolo 4

Fase 4

4.1 Realizzazione della base di dati

```
1  SET GLOBAL local_infile = 1;
2  SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
3  SET GLOBAL time_zone = '+00:00';
4  SET time_zone = '+00:00';
5
6  START TRANSACTION;
7
8  -- Creazione del database
9  CREATE DATABASE IF NOT EXISTS `cce`;
10 USE `cce`;
11
12 -- Struttura della tabella `account`
13 CREATE TABLE `account` (
14    `idAccount` INT(11) NOT NULL AUTO_INCREMENT,
15    `email` VARCHAR(50) NOT NULL UNIQUE,
16    `username` VARCHAR(20) NOT NULL UNIQUE,
17    `nome` VARCHAR(30) NOT NULL,
18    `cognome` VARCHAR(30) NOT NULL,
19    `data_nascita` DATE NOT NULL,
20    `password` VARCHAR(255) NOT NULL,
21    `numero_telefono` CHAR(10) NOT NULL,
22    `luogo_nascita` VARCHAR(30) NOT NULL,
23   PRIMARY KEY (`idAccount`)
24 );
25
26 -- Struttura della tabella `medico`
27 CREATE TABLE `medico` (
28    `idMedico` INT(11) NOT NULL AUTO_INCREMENT,
```

```
29    `genere` ENUM('M', 'F') NOT NULL,
30    `specializzazione` VARCHAR(255) NOT NULL,
31    `idAccount` INT(11) NOT NULL,
32    PRIMARY KEY (`idMedico`),
33    FOREIGN KEY (`idAccount`) REFERENCES `account`
          (`idAccount`) ON DELETE CASCADE
34  );
35
36  -- Struttura della tabella `paziente`
37  CREATE TABLE `paziente` (
38    `idPaziente` INT(11) NOT NULL AUTO_INCREMENT,
39    `idAccount` INT(11) NOT NULL,
40    `CF` CHAR(16) NOT NULL UNIQUE,
41    PRIMARY KEY (`idPaziente`),
42    FOREIGN KEY (`idAccount`) REFERENCES `account`
          (`idAccount`) ON DELETE CASCADE
43  );
44
45  -- Struttura della tabella `cartella_clinica`
46  CREATE TABLE `cartella_clinica` (
47    `idCartella` INT(11) NOT NULL AUTO_INCREMENT,
48    `procedure_diagnostiche` VARCHAR(255) NOT NULL,
49    `farmaci` VARCHAR(255) NOT NULL,
50    `storia_passata` VARCHAR(255) NOT NULL,
51    `trattamenti` VARCHAR(255) NOT NULL,
52    `osservazioni` VARCHAR(1000) NOT NULL,
53    `idPaziente` INT(11) NOT NULL,
54    PRIMARY KEY (`idCartella`),
55    FOREIGN KEY (`idPaziente`) REFERENCES `paziente`
          (`idPaziente`) ON DELETE CASCADE
56  );
57
58  -- Struttura della tabella `accede`
59  CREATE TABLE `accede` (
60    `idMedico` INT(11) NOT NULL,
61    `idCartella` INT(11) NOT NULL,
62    PRIMARY KEY (`idMedico`, `idCartella`),
63    FOREIGN KEY (`idMedico`) REFERENCES `medico`
          (`idMedico`) ON DELETE CASCADE,
64    FOREIGN KEY (`idCartella`) REFERENCES
          `cartella_clinica` (`idCartella`) ON DELETE CASCADE
65  );
66
67  -- Struttura della tabella `appuntamento`
68  CREATE TABLE `appuntamento` (
69    `idAppuntamento` INT(11) NOT NULL AUTO_INCREMENT,
```

```
70    `data` DATE NOT NULL,
71    `n.civico` VARCHAR(3) NOT NULL,
72    `via` VARCHAR(255) NOT NULL,
73    `CAP` INT(5) NOT NULL,
74    `ora` TIME NOT NULL,
75    `idMedico` INT(11) NOT NULL,
76    `idPaziente` INT(11) NOT NULL,
77    PRIMARY KEY (`idAppuntamento`),
78    FOREIGN KEY (`idMedico`) REFERENCES `medico`
          (`idMedico`) ON DELETE CASCADE,
79    FOREIGN KEY (`idPaziente`) REFERENCES `paziente`
          (`idPaziente`) ON DELETE CASCADE
80  );
81
82  -- Struttura della tabella `carriera`
83  CREATE TABLE `carriera` (
84    `idCarriera` INT(11) NOT NULL AUTO_INCREMENT,
85    `ruolo` VARCHAR(255) NOT NULL,
86    `data_inizio` DATE NOT NULL,
87    `data_fine` DATE,
88    `descrizione` VARCHAR(500) NOT NULL,
89    `idMedico` INT(11) NOT NULL,
90    PRIMARY KEY (`idCarriera`),
91    FOREIGN KEY (`idMedico`) REFERENCES `medico`
          (`idMedico`) ON DELETE CASCADE
92  );
93
94  -- Struttura della tabella `servizio`
95  CREATE TABLE `servizio` (
96    `idServizio` INT(11) NOT NULL AUTO_INCREMENT,
97    `nome` VARCHAR(255) NOT NULL,
98    `descrizione` VARCHAR(500) NOT NULL,
99    `idMedico` INT(11) NOT NULL,
100   PRIMARY KEY (`idServizio`),
101   FOREIGN KEY (`idMedico`) REFERENCES `medico`
          (`idMedico`) ON DELETE CASCADE
102 );
103
104 -- Struttura della tabella `lingua`
105 CREATE TABLE `lingua` (
106     `idLingua` INT(11) NOT NULL AUTO_INCREMENT,
107     `lingua` VARCHAR(255) NOT NULL,
108     `idServizio` INT(11) NOT NULL,
109     PRIMARY KEY (`idLingua`),
110     FOREIGN KEY (`idServizio`) REFERENCES `servizio`
          (`idServizio`) ON DELETE CASCADE
```

```
111 );
112
113
114 -- Struttura della tabella `pagamento`
115 CREATE TABLE `pagamento` (
116   `idPagamento` INT(11) NOT NULL AUTO_INCREMENT,
117   `importo` DOUBLE NOT NULL,
118   `data` DATE NOT NULL,
119   `tipo` ENUM('contanti', 'carta') NOT NULL,
120   `causale` VARCHAR(255) NOT NULL,
121   `idAppuntamento` INT(11) NOT NULL,
122   PRIMARY KEY (`idPagamento`),
123   FOREIGN KEY (`idAppuntamento`) REFERENCES
        `appuntamento` (`idAppuntamento`) ON DELETE CASCADE
124 );
125
126 -- Struttura della tabella `test`
127 CREATE TABLE `test` (
128   `idTest` INT(11) NOT NULL AUTO_INCREMENT,
129   `esito` VARCHAR(255) NOT NULL,
130   `tipo` VARCHAR(255) NOT NULL,
131   `data` DATE NOT NULL,
132   `idMedico` INT(11) NOT NULL,
133   `idPaziente` INT(11) NOT NULL,
134   `idCartella` INT(11) NOT NULL,
135   PRIMARY KEY (`idTest`),
136   FOREIGN KEY (`idMedico`) REFERENCES `medico`
        (`idMedico`) ON DELETE CASCADE,
137   FOREIGN KEY (`idPaziente`) REFERENCES `paziente`
        (`idPaziente`) ON DELETE CASCADE,
138   FOREIGN KEY (`idCartella`) REFERENCES
        `cartella_clinica` (`idCartella`) ON DELETE CASCADE
139 );
140
141 COMMIT;
```

4.2 Script in SQL che permette di popolare da zero la base di dati realizzata

```
1  -- Popolare la tabella account
2  INSERT INTO `account` (`idAccount`, `email`, `username`,
       `nome`, `cognome`, `data_nascita`, `password`,
       `numero_telefono`, `luogo_nascita`) VALUES
3  (1, 'mario.rossi@example.com', 'mario.rossi', 'Mario',
       'Rossi', '1980-01-15', SHA1('password1'), 3331234567,
       'Roma'),
4  (2, 'luigi.bianchi@example.com', 'luigi.bianchi',
       'Luigi', 'Bianchi', '1990-02-25', SHA1('password2'),
       3349876543, 'Milano'),
5  (3, 'anna.verdi@example.com', 'anna.verdi', 'Anna',
       'Verdi', '1985-05-10', SHA1('password3'), 3455678901,
       'Napoli');
6
7  -- Popolare la tabella paziente
8  INSERT INTO `paziente` (`idPaziente`, `idAccount`, `CF`)
       VALUES
9  (1, 1, 'RSSMRA80A01H501X'),
10 (2, 2, 'BNCLGU90B25F205P');
11
12 -- Popolare la tabella medico
13 INSERT INTO `medico` (`idMedico`, `genere`,
       `specializzazione`, `idAccount`) VALUES
14 (1, 'M', 'Cardiologia', 3),
15 (2, 'F', 'Pediatria', 2);
16
17 -- Popolare la tabella carriera
18 INSERT INTO `carriera` (`idCarriera`, `ruolo`,
       `data_inizio`, `data_fine`, `descrizione`,
       `idMedico`) VALUES
19 (1, 'Medico-Generico', '2010-01-01', '2015-12-31',
       'Esperienza-in-clinica-generale', 1),
20 (2, 'Specialista-Cardiologia', '2016-01-01',
       '2024-12-31', 'Lavoro-in-ospedale', 1);
21
22 -- Popolare la tabella appuntamento
23 INSERT INTO `appuntamento` (`idAppuntamento`, `data`,
       `n.civico`, `via`, `CAP`, `ora`, `idMedico`,
       `idPaziente`) VALUES
24 (1, '2024-12-25', '25', 'Via-Roma', '84091', '10:00:00',
       1, 1),
```

```
25  (2, '2024-12-26', '30', 'Via Milano', '80053',
        '14:30:00', 2, 2);
26
27  -- Popolare la tabella pagamento
28  INSERT INTO `pagamento` (`idPagamento`, `importo`,
        `data`, `tipo`, `causale`, `idAppuntamento`) VALUES
29  (1, 50.00, '2024-12-25', 'contanti', 'Visita
        specialistica', 1),
30  (2, 75.00, '2024-12-26', 'carta', 'Controllo
        pediatrico', 2);
31
32  -- Popolare la tabella cartella clinica
33  INSERT INTO `cartella_clinica` (`idCartella`,
        `procedure_diagnostiche`, `farmaci`,
        `storia_passata`, `trattamenti`, `osservazioni`,
        `idPaziente`) VALUES
34  (1, 'ECG', 'Aspirina', 'Ipertensione', 'Terapia
        antipertensiva', 'Buona risposta alla terapia', 1),
35  (2, 'Visita pediatrica', 'Ibuprofene', 'Febbre
        ricorrente', 'Antibiotici', 'Monitorare la febbre',
        2);
36
37  -- Popolare la tabella accede
38  INSERT INTO `accede` (`idMedico`, `idCartella`) VALUES
39  (1, 1),
40  (2, 2);
41
42  -- Popolare la tabella test
43  INSERT INTO `test` (`idTest`, `esito`, `tipo`, `data`,
        `idMedico`, `idPaziente`, `idCartella`) VALUES
44  (1, 'Normale', 'ECG', '2024-12-20', 1, 1, 1),
45  (2, 'Negativo', 'Test influenza', '2024-12-22', 2, 2, 2);
46
47  -- Popolare la tabella servizio
48  INSERT INTO `servizio` (`idServizio`, `nome`,
        `descrizione`, `idMedico`) VALUES
49  (1, 'Cardiologia', 'Visite cardiologiche', 1),
50  (2, 'Pediatria', 'Cure pediatriche', 2);
51
52  -- Popolare la tabella lingua
53  INSERT INTO `lingua` (`idLingua`, `lingua`,
        `idServizio`) VALUES
54  (1, 'Italiano, Inglese', 1),
55  (2, 'Italiano, Francesce, Inglese, Spagnolo', 2);
```

Capitolo 5

Fase 5

5.1 Implementazione query SQL

Operazioni CRUD implementate utilizzando **JDBC**.

Operazione 1:

Visualizza account:

```
SELECT * FROM account
```

Operazione 2:

Visualizza account data l'email:

```
SELECT * FROM account WHERE email = ?
```

Operazione 3:

Visualizza account data l'email e il numero di telefono:

```
SELECT * FROM account WHERE email = ? AND  numero_telefono = ?
```

Operazione 4:

Cancella account:

```
DELETE FROM account WHERE idAccount = 3
```

Operazione 5:

Cancella account dato l'id:

```
DELETE FROM account WHERE idAccount ='" + param_id + "'
```

Operazione 6:

Crea tabella:

```
CREATE TABLE ‘account_1‘ (""‘idAccount‘ INT(11) NOT NULL AUTO_INCREMENT, "
+ "‘email‘ VARCHAR(50) NOT NULL UNIQUE, "
+ "‘username‘ VARCHAR(20) NOT NULL UNIQUE, "
+ "‘nome‘ VARCHAR(30) NOT NULL, "
+ "‘cognome‘ VARCHAR(30) NOT NULL, "
+ "‘data_nascita‘ DATE NOT NULL, "
+ "‘password‘ VARCHAR(255) NOT NULL, "
+ "‘numero_telefono‘ CHAR(10) NOT NULL, "
+ "‘luogo_nascita‘ VARCHAR(30) NOT NULL, "
+ "PRIMARY KEY (‘idAccount‘)"
+ ");
```

Operazione 7:

Cancella tabella:

```
DROP TABLE account_1
```

Operazione 8:

Inserimento account:

```
INSERT INTO account VALUES (4, 'gennaro.tozza@example.com', 'gennaro.tozza',
'Gennaro', 'Tozza', '2004-04-25', SHA1('password4'), 3928011093, 'Battipaglia')
```

Operazione 9:

Inserimento account con dati:

```
INSERT INTO account VALUES (?,?,?,?,?,?,?,MD5(?),?,?)
```

Operazione 10:

Aggiornamento account:

```
UPDATE account SET email = 'gennarocarmine@example.com'
WHERE email = 'gennaro.tozza@example.com'
```

Operazione 11:

Visualizza tabella dato il nome:

```
SELECT * FROM " + tableName
```

Operazione 12:

Elenca tabelle:

```
SHOW TABLES
```

Altre query:

Visualizza informazioni medico in base a un dato servizio:

```
SELECT  m.idMedico, a.nome, a.cognome, a.email
FROM medico AS m, servizio AS s, account AS a
WHERE s.nome = ? AND m.idAccount = a.idAccount AND m.idMedico = s.idMedico;
```

-

Visualizza i medici con più appuntamenti e mostra il loro nome, cognome, specializzazione e il numero di appuntamenti:

```
SELECT m.idMedico, a.nome, a.cognome, m.specializzazione,
       COUNT(app.idAppuntamento) AS num_appuntamenti
FROM medico AS m, account AS a, appuntamento AS app
WHERE m.idAccount = a.idAccount AND m.idMedico = app.idMedico
GROUP BY m.idMedico, a.nome, a.cognome, m.specializzazione
ORDER BY num_appuntamenti DESC;
```

-

Elenca i pazienti che non hanno mai avuto appuntamenti con medici della specializzazione data:

```
SELECT p.idPaziente, a.nome AS nome_paziente, a.cognome AS cognome_paziente
FROM paziente AS p, account AS a
WHERE p.idAccount = a.idAccount AND p.idPaziente NOT IN (
                SELECT app.idPaziente
                FROM appuntamento app, medico m
                WHERE app.idMedico = m.idMedico AND m.specializzazione = ?);
```

-

Calcola l'importo totale dei pagamenti ricevuti da ciascun medico, raggruppando per tipo di pagamento:

```
SELECT m.idMedico, a.nome AS nome_medico, a.cognome AS cognome_medico, pag.tipo,
    SUM(pag.importo) AS totale_importo
FROM medico AS m, account AS a, appuntamento AS app, pagamento AS  pag
WHERE m.idAccount = a.idAccount AND m.idMedico = app.idMedico
AND app.idAppuntamento = pag.idAppuntamento
GROUP BY m.idMedico, a.nome, a.cognome, pag.tipo
ORDER BY m.idMedico, pag.tipo;
```

Capitolo 6

Fase 6

6.1 Test dell'applicazione

```
*** Cce DB ***
Connessione OK

Mon Dec 30 15:54:38 CET 2024 WARN: Caught while disconnecting...

EXCEPTION STACK TRACE:



** BEGIN NESTED EXCEPTION **

javax.net.ssl.SSLException
MESSAGE: closing inbound before receiving peer's close_notify

STACKTRACE:

javax.net.ssl.SSLException Create breakpoint : closing inbound before receiving peer's close_notify
    at java.base/sun.security.ssl.SSLSocketImpl.shutdownInput(SSLSocketImpl.java:838)
    at java.base/sun.security.ssl.SSLSocketImpl.shutdownInput(SSLSocketImpl.java:817)
    at com.mysql.cj.protocol.a.NativeProtocol.quit(NativeProtocol.java:1319)
    at com.mysql.cj.NativeSession.quit(NativeSession.java:182)
    at com.mysql.cj.jdbc.ConnectionImpl.realClose(ConnectionImpl.java:1750)
    at com.mysql.cj.jdbc.ConnectionImpl.close(ConnectionImpl.java:720)
    at cce.Cce.releaseConnection(Cce.java:31)
    at cce.Cce.testConnection(Cce.java:39)
    at cce.Cce.main(Cce.java:806)


** END NESTED EXCEPTION **
```

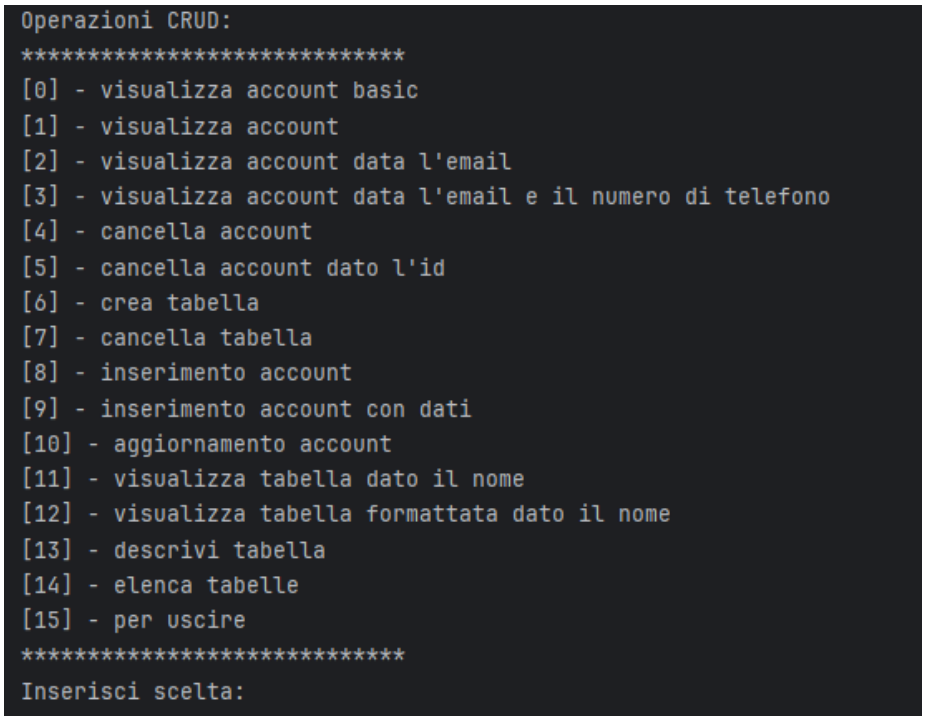


Figura 6.1: Screenshot dell'avvio dell'applicazione.

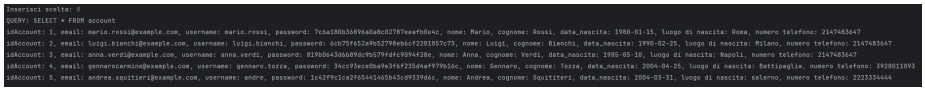


Figura 6.2: Screenshot dei risultati della query SQL.

Figura 6.3: Screenshot dei risultati della query SQL.

Figura 6.4: Screenshot dei risultati della query SQL.

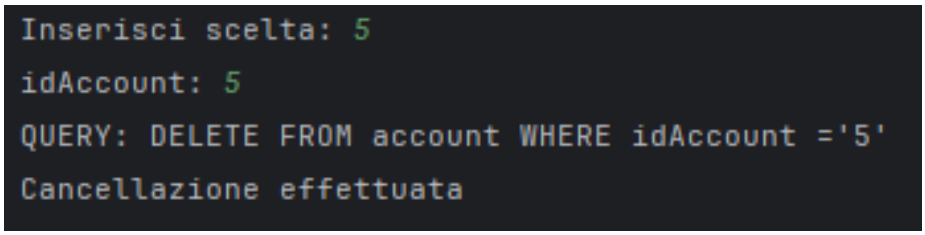


Figura 6.5: Screenshot dei risultati della query SQL.

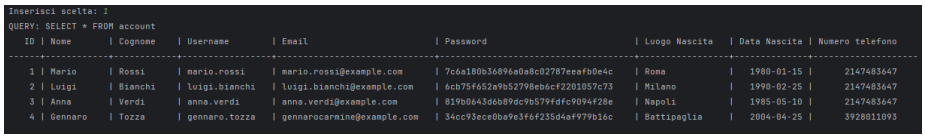


Figura 6.6: Screenshot dei risultati della query SQL dopo la cancellazione.

Figura 6.7: Screenshot dei risultati della query SQL.

Figura 6.8: Screenshot dei risultati della query SQL.

```
Inserisci scelta: 12
Nome tabella: medico
QUERY: SELECT * FROM medico
Printing 2 rows from table medico
+-----------+--------+------------------+-----------+
| idMedico | genere | specializzazione | idAccount |
+-----------+--------+------------------+-----------+
|         1 | M      | Cardiologia      |         3 |
|         2 | F      | Pediatria        |         2 |
+-----------+--------+------------------+-----------+
```

Figura 6.9: Screenshot dei risultati della query SQL.

```
mysql> SELECT m.idMedico, a.nome, a.cognome, m.specializzazione,
    ->                COUNT(app.idAppuntamento) AS num_appuntamenti
    ->         FROM medico AS m, account AS a, appuntamento AS app
    ->         WHERE m.idAccount = a.idAccount AND m.idMedico = app.idMedico
    ->         GROUP BY m.idMedico, a.nome, a.cognome, m.specializzazione
    ->         ORDER BY num_appuntamenti DESC;
+-----------+-------+---------+------------------+------------------+
| idMedico | nome  | cognome | specializzazione | num_appuntamenti |
+-----------+-------+---------+------------------+------------------+
|         1 | Anna  | Verdi   | Cardiologia      |                1 |
|         2 | Luigi | Bianchi | Pediatria        |                1 |
+-----------+-------+---------+------------------+------------------+
2 rows in set (0.02 sec)
```

Figura 6.10: Screenshot dei risultati della query SQL.

```
mysql> SELECT m.idMedico, a.nome AS nome_medico, a.cognome AS cognome_medico, pag.tipo,
    ->             SUM(pag.importo) AS totale_importo
    ->         FROM medico AS m, account AS a, appuntamento AS app, pagamento AS  pag
    ->         WHERE m.idAccount = a.idAccount AND m.idMedico = app.idMedico
    ->         AND app.idAppuntamento = pag.idAppuntamento
    ->         GROUP BY m.idMedico, a.nome, a.cognome, pag.tipo
    ->         ORDER BY m.idMedico, pag.tipo;
+-----------+-------------+----------------+----------+----------------+
| idMedico | nome_medico | cognome_medico | tipo     | totale_importo |
+-----------+-------------+----------------+----------+----------------+
|         1 | Anna        | Verdi          | contanti |             50 |
|         2 | Luigi       | Bianchi        | carta    |             75 |
+-----------+-------------+----------------+----------+----------------+
2 rows in set (0.01 sec)
```

Figura 6.11: Screenshot dei risultati della query SQL.