Centro Benessere *Loto Bianco*

1. ABSTRACT

Il progetto si concentra sulla progettazione di una base di dati per il Centro Benessere Loto Bianco, una struttura dedicata alla cura e al benessere della persona. Il centro offre trattamenti estetici e terapeutici personalizzati, rivolti sia al viso sia al corpo.

Una delle peculiarità della struttura è la presenza di un'esclusiva area termale, accessibile unicamente ai clienti abbonati. Tale area viene gestita e mantenuta da personale specializzato.

Tutti i clienti del centro, siano essi abbonati o meno, hanno la possibilità di prenotare trattamenti specifici. I trattamenti vengono realizzati utilizzando prodotti professionali, forniti da aziende esterne selezionate, la cui gestione è tracciata all'interno del sistema informativo.

Il personale del centro si suddivide in tre principali categorie operative:
- gli specialisti dei trattamenti, che eseguono le prestazioni richieste;
- gli addetti all'area termale, responsabili della manutenzione e del corretto funzionamento degli spazi dedicati. Nel corso del tempo, vengono effettuate varie manutenzioni, di cui si vogliono memorizzare le relative informazionii;
- il personale di segreteria, incaricato di ordinare i prodotti dai fornitori e rilasciare le fatture ai clienti.

La base di dati è progettata per supportare la gestione di tutte queste attività, assicurando un'organizzazione efficiente delle informazioni riguardanti i clienti, trattamenti, dipendenti, fornitori e operazioni amministrative.

2. ANALISI DEI REQUISITI

Il database registrerà i clienti, i dipendenti e la ditta di rifornimento, tenendo traccia dei trattamenti offerti dalla struttura e delle fatture.

Un **cliente** si reca al centro benessere ed è caratterizzato da:

- CF
- nome
- cognome
- numero di telefono
- indirizzo

Il cliente può effettuare la prenotazione di un trattamento. I **trattamenti** sono caratterizzati da:

e sono di due tipi:

- pulizia viso caratterizzata da: tipologia di pelle
- massaggio caratterizzato da: zona e durata

Nel centro benessere lavorano i **dipendenti** caratterizzati da:

- CF
- nome
- cognome
- telefono
- stipendio

I trattamenti vengono svolti dagli **specialisti** del centro che hanno:

- titolo

Per effettuare i trattamenti, gli specialisti hanno bisogno di usare dei prodotti specifici che vengono riforniti da alcune **aziende** selezionate dalla struttura le quali sono caratterizzate da:

- P. IVA
- nome
- sede

Quando i prodotti finiscono, la segreteria si prende in carico la gestione delle ordinazioni di tali **prodotti** aventi:

- marca
- nome
- Ingredienti
- data di scadenza

Un cliente che si è **abbonato** al centro benessere per l'accesso all'area termale è caratterizzato da.

- numero carta
- data di iscrizione

Un cliente abbonato ha diritto ad accedere all'esclusiva **area termale** caratterizzata da:

- ID area
- orario apertura
- orario chiusura

Dell'area termale si occupano gli **addetti di pulizia** e manutenzione che sono caratterizzati da:

- turni
- area

Infine vengono gestite le fatture emesse dai segretari ai clienti che hanno usufruito dei servizi del centro. Una **fattura** è caratterizzata da:

- numero
- costo del trattamento
- modalità di pagamento

3. PROGETTAZIONE CONCETTUALE

——— Lista entità
Il database è formato dalle seguenti tabelle. Quando non è specificato, gli attributi sono NOT NULL.

1. cliente: Rappresenta il cliente del centro benessere.
   - cf: string
   - nome: string
   - cognome: string
   - indirizzo: attributo composto da:
     - città: string
     - via: string
     - cap: string
   - telefono: string UNIQUE

2.abbonato: Rappresenta i clienti che hanno sottoscritto l'abbonamento al centro benessere per l'accesso all'area termale.
   - numero_carta: int
   - data_iscrizione: date

3. trattamento: Rappresenta il trattamento che può essere prenotato.
   - codice: int
   - prezzo: decimal
   - data: date
   - ora: time

ed è di 2 possibili tipi:
   -pulizia_viso
      -tipologia: string
   -massaggio:
      -zona: string
      -durata: int

4. azienda: Rappresenta le ditte che riforniscono il centro benessere.

- piva: string
- nome: string
- sede: string

5. prodotto: Rappresenta il prodotto usato nei trattamenti.
   - marca: int
   - nome: string
   - ingredienti: varchar
   - data_scadenza: date

6. area termale: Rappresenta l'area esclusiva a cui hanno accesso solo i clienti abbonati.
   - id_area: int
   - orario_apertura: varchar
   - orario_chiusura: varchar

7. dipendente: Rappresenta coloro che lavorano nel centro.
   - cf: string
   - nome: string
   - cognome: string
   - telefono: string UNIQUE
   - stipendio: decimal

   segretario

   specialista
   - titolo: varchar

   addetto_pulizie
   - turno: enum
   - area=id_area(area_termale): int

8. fattura: Rappresenta come viene gestita una fattura.
   - numero: int
   - totale: decimal
   - mod_pagamento: string

——— Lista Relazioni

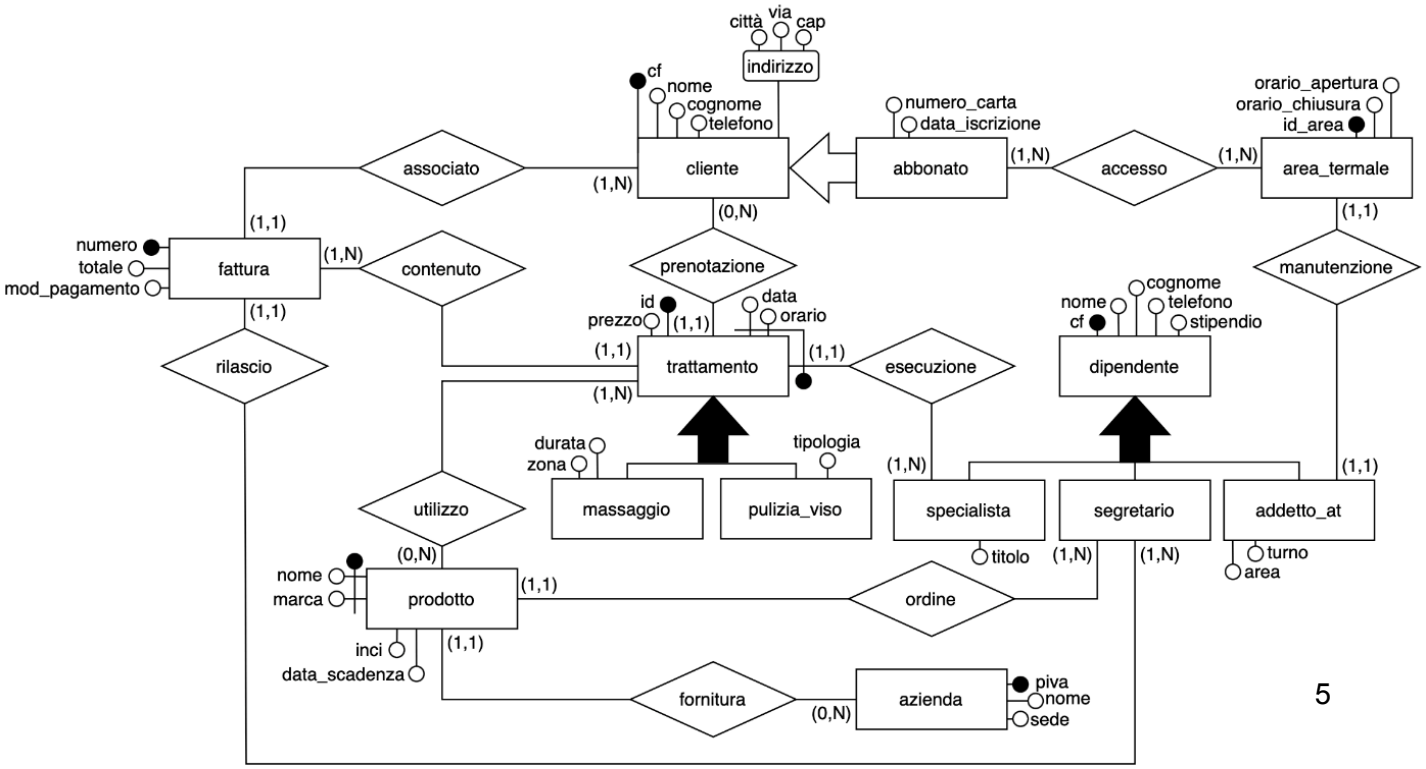
1. cliente-trattamento: prenotazione (1,N)	● Ogni cliente può prenotare nessuno o più trattamenti (0,N) ● Ogni trattamento è prenotato da uno e un solo cliente (1,1)
2. cliente-fattura: associato (1,N)	● Un cliente paga una o più fatture (1,N) ● Una fattura è associata ad un solo cliente (1,1)

3. fattura-segretario: rilascio (1,N)	• Una fattura è effettuata da un unico segretario (1,1) • Un segretario può o meno rilasciare fatture (1,N)
4. fattura-trattamento: pagamento (1,1)	• Esiste una fattura per trattamento (1,1) • Un trattamento produce un'unica fattura (1,1)
fattura trattamento pagamento (1,n)	- a una fattura possono essere associati più trattamenti (1,n) - ogni trattamento è contenuto in una sola fattura (1,1)
5. trattamento-prodotto: utilizzo (N,N)	• durante un trattamento vengono utilizzati uno o più prodotti (1,N) • un prodotto può essere utilizzato durante un trattamento (0,N)
6. prodotto-segreteria: ordinazione (1,N)	• un prodotto viene ordinato da uno e un solo segretario (1,1) • un segretario ordina almeno un prodotto, o più (1,N)
7. prodotto-azienda: fornitura (N,N)	• un prodotto è fornito da una e una sola ditta (1,1) • una ditta può fornire zero o più prodotti (0,N)
8. trattamento-specialista: esecuzione (1,1) /(1,N)	• ogni trattamento viene eseguito da un solo specialista (1,1) • ogni specialista esegue uno e un solo trattamento (1,1)
9. abbonato-area_termale: accesso (1,N) - n.n	• un cliente abbonato ha accesso all'area termale (1,1) • L'area termale da accesso a uno o più abbonati (0,N)
10. area_termale-addetto_at: manutenzione (1,1) n.n	• l'area termale è mantenuta dall'addetto all'area termale (1,1) • l'addetto all'area termale mantiene l'area termale (1,1)

——— Lista Generalizzazioni

- *trattamento* è una generalizzazione totale ed esclusiva di *massaggio* e *pulizia_viso*.
- *dipendente* è una generalizzazione totale ed esclusiva di *specialista*, *segretario* e *addetto_at*.
- *cliente* è una generalizzazione parziale di abbonato.

——— Schema E-R



Nota:

- Addetto fa più manutenzioni (salvare attributi e.g. “data_manutenzione” sulla relazione) per più aree termali (no “1-1)

4. PROGETTAZIONE LOGICA

(Riferimenti: 08-ProgettazioneLogica.pdf)

——— Analisi delle ridondanze

Analizzando lo schema E-R, si osserva che l'attributo *totale* dell'entità fattura potrebbe essere calcolato come la somma dell'attributo *prezzo* dell’entità *trattamento* associato a ciascuna fattura attraverso la relazione *contenuto* (di tipo 1 a N). Dopo l'implementazione, la relazione *contenuto* non esiste come tabella separata, ma è rappresentata come un attributo di *trattamento* che fa riferimento a *fattura* (`trattamento.fattura` ➔ `fattura.numero`). L’attributo *totale* è una ridondanza ed è quindi necessario analizzare le operazioni che lo riguardano per determinare se conviene mantenerlo o eliminarlo.

Operazioni considerate:

- Operazione 1 (1000 alla settimana): inserimento di un nuovo trattamento
- Operazione 2 (100 alla settimana): visualizzare il totale di una fattura

Alle operazioni di scrittura sarà assegnato un costo doppio rispetto alle operazioni di lettura.

Per ciascun trattamento, i prodotti considerati risultano ridondanti in scrittura e lettura, dovendo memorizzare un’informazione legata al costo *totale* delle forniture effettuate nel tempo.

Operazioni considerate:

- Operazione 3 (1000 alla settimana): inserimento di una nuova fornitura

Alle operazioni di scrittura sarà assegnato un costo doppio rispetto alle operazioni di lettura.

Volume dei dati:

Concetto	Construtto	Volume
azienda(fornitura)	Entità	50000
fattura(trattamento)	Entità	20000

Con ridondanza

Operazione 1: *(1*100) accessi in lettura e (1*100+1*100) accessi in scrittura*

Concetto	Costrutto	Operazione di accesso DB	Tipo	Operazioni al giorno
trattamento	Entità	1	S (Scrittura)	x100 volte/giorno
fattura	Entità	1	L (Lettura)	x100 volte/giorno

fattura	Entità	1	S (Scrittura)	x100 volte/giorno

Operazione 2: *(1*10) accessi in lettura*

Concetto	Costrutto	Operazione di accesso DB	Tipo	Operazioni al giorno
fattura	Entità	1	L (Lettura)	x10 volte/giorno

Costo settimanale totale con ridondanza = (100+10)*1 + 200*2 = **510**

Senza ridondanza

Operazione 1: *(1*100) accessi in scrittura*

Concetto	Costrutto	Accesso	Tipo
trattamento	Entità	1	scrittura

Operazione 2: *(5*10) accessi in lettura*

Concetto	Costrutto	Accesso	Tipo
trattamento	Entità	5	lettura

Assumiamo che in media ci siano 5 procedure mediche per ogni fattura.

Costo settimanale totale senza ridondanza = (50)*1 + 100*2 = **250**

Confrontando i costi settimanali, con ridondanza 510 accessi e senza ridondanza 250, eliminare la ridondanza risulta conveniente in termini di accessi settimanali. Pertanto, conviene rimuovere il campo *totale* dalla tabella *fattura* e calcolare dinamicamente dalle entità *trattamento* associate, riducendo il carico complessivo del sistema.

——— Eliminazioni delle generalizzazioni

- *dipendente*: La generalizzazione *dipendente* è stata trasformata in una relazione tra l'entità *dipendente* e le entità figlie *specialista*, *segretario* e *addetto_pulizie*. Questo cambiamento permette una gestione più precisa e dettagliata delle specifiche responsabilità e caratteristiche di ciascun tipo di dipendente, evitando di avere campi NULL (presenti nel caso in cui si avesse preferito accorpare le entità figlie nel padre) o di avere i dati anagrafici di tutti i dipendenti della clinica divisi su più entità (nel caso si avesse preferito dividere la generalizzazione nelle entità figlie). Questa decisione implica l'aggiunta di una caratteristica logica specifica che impedisce a un dipendente di appartenere a più categorie contemporaneamente tra speciialista, segretario e addetto_pulizie.
- *trattamento*: La generalizzazione *trattamento* è stata suddivisa nelle sue classi figlie *trattamento_viso* e *trattamento_corpo*. L'entità *medicinale* contiene informazioni specifiche sui farmaci, mentre l'entità *terapia* include dettagli sulle terapie. Questa suddivisione permette una gestione più precisa e chiara dei trattamenti, evitando la presenza di campi NULL.

Partizionamento/Accorpamento di Entità e Relationship

- Le entità *veterinario* e *addetto_pulizie* possiedono entrambe un attributo composto *stanza*. Poiché le due entità possono condividere la stessa stanza (in particolare è un attributo multivalore per

l'entità *addetto_pulizie*), gli attributi risultano ridondanti. Pertanto, viene creata un'entità *stanza* che ha una relazione 1 a 1 con *veterinario* e una relazione molti a molti con *addetto_pulizie*.

- L'attributo composto *indirizzo*, posseduto unicamente dall'entità *proprietario*, viene accorpato direttamente nell'entità stessa. Al posto dell'attributo *indirizzo*, l'entità *proprietario* ora contiene direttamente gli attributi *città*, *via* e *cap*.

Scelta di identificatori primari

- *stanza*: La nuova entità *stanza* avrà come identificatore primario gli attributi *numero* e *reparto* perchè per diversi reparti ci possono essere stanze con lo stesso numero.
- *Veterinario*, *Segretario*, *Addetto_Pulizie*: ognuna di queste entità ha come identificatore primario la relazione con l'entità *dipendente*.
- *medicinale, terapia*: La divisione della generalizzazione *trattamento* nelle sue entità figlie non comporta alcun cambiamento della loro chiave primaria.

- **Per schema logico**

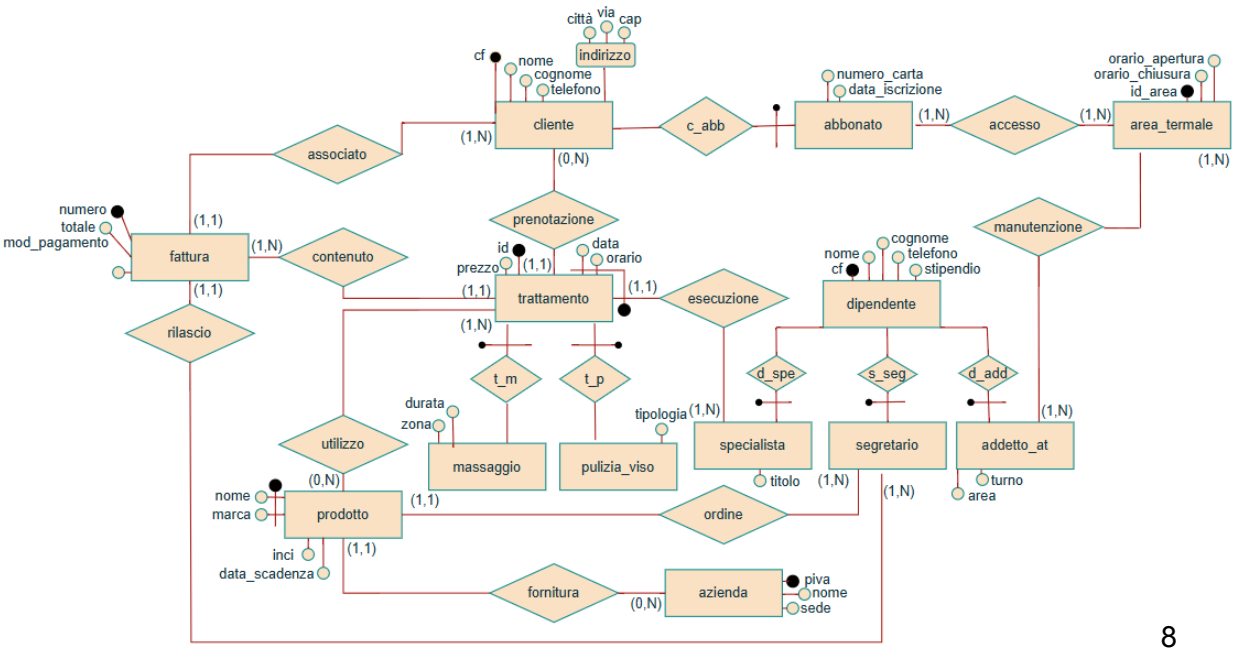
(* = asterisco -> Campi NULL secondo De Leoni)

Cliente(CF, Telefono*,

- **Per schema ER**

Telefono (0,1)

——— Diagramma ER ristrutturato



——— Descrizione schema relazionale

cliente(cf, nome, cognome, indirizzo, telefono, data_nascita)

trattamento(nome, prezzo, data, ora)

trattamento_viso(*nome_tratt*)

trattamento_corpo(*nome_tratt*)

azienda(piva, nome, sede, num_dipendenti, fatturato)

prodotto(codice, nome, ingredienti, data_scadenza)
area_termale(id, orario_apertura, orario_chiusura)
dipendente (cf, nome, cognome, telefono, stipendio)
specialista(*cf_dip*, licenza)
segretario (*cf_dip*, *veterinario*)
addetto_pulizie (*cf_dip*, turno)
stanza (numero, reparto)
pulizia (*n_stanza*, *reparto*, *addetto*)
fattura (numero, mod_pagamento, totale, *cliente, segretario*)

——— Vincoli di integrità referenziale

animale.proprietario ➔ proprietario.cf
animale.specie ➔ specie.nome
visita.animale ➔ animale.microchip
visita.fattura ➔ fattura.numero
visita.veterinario ➔ veterinario.cf_dip
procedura_medica.visita ➔visita.id_visita
procedura_medica.fattura ➔ fattura.numero
efficacia.medicinale ➔ medicinale.nome
efficacia.specie ➔ specie.nome
conclusione_med.visita ➔visita.id_visita
conclusione_med.medicinale ➔medicinale.nome
assegnazione_med.veterinario ➔ veterinario.cf_dip
assegnazione_med.medicinale ➔medicinale.nome
conclusione_ter.visita ➔visita.id_visita
conclusione_ter.terapia ➔terapia.tipo
assegnazione_ter.veterinario ➔ veterinario.cf_dip
assegnazione_ter.terapia ➔terapia.tipo
veterinario.cf_dip ➔ dipendente.cf
veterinario.(n_stanza, reparto) ➔ stanza.(numero, reparto)
addetto_pulizie.cf_dip ➔ dipendente.cf
segretario.cf_dip ➔ dipendente.cf
segretario.veterinario ➔ veterinario.cf_dip
pulizia.(n_stanza, reparto) ➔ stanza.(numero, reparto)
pulizia.addetto➔ addetto_pulizie.cf_dip
fattura.proprietario ➔ proprietario.cf
fattura.segretario ➔ segretario.cf_dip

5. QUERY E INDICI ASSOCIATI

——— Query

1) Clienti che hanno ricevuto più di un trattamento con specialisti diversi nello stesso giorno

```
SELECT c.nome, c.cognome, t.data, COUNT(DISTINCT t.cf_dipendente) AS specialisti_coinvolti
FROM trattamenti t
JOIN clienti c ON t.cf_cliente = c.cf
GROUP BY c.cf, c.nome, c.cognome, t.data
HAVING COUNT(DISTINCT t.cf_dipendente) > 1;
```

-Dipendenti specialisti che hanno eseguito trattamenti utilizzando almeno 3 prodotti diversi in un solo appuntamento
-Fatture con importi superiori alla somma dei costi dei prodotti utilizzati nel trattamento
-Fornitori i cui prodotti sono stati usati in trattamenti che hanno generato in media un incasso superiore del 20% rispetto alla media di tutti i trattamenti dello stesso tipo
-Prodotti utilizzati più frequentemente nei trattamenti per il viso, ma mai nei trattamenti per il corpo
-Trattamenti che hanno utilizzato un prodotto con scadenza entro 30 giorni dalla data del trattamento
-Per ogni specialista, il giorno in cui ha effettuato il maggior numero di trattamenti, con relativo conteggio

——----------

1.

2. Restituire per ogni proprietario il codice fiscale, nome, cognome e il totale che ha pagato per le visite di urgenza 'rosso' e il totale che ha pagato per le visite di tutti i tipi di urgenza. Si ordini poi il risultato in ordine alfabetico rispetto ai codici fiscali dei proprietari.

```
SELECT Tot.cf AS cf_proprietario, Tot.nome, Tot.cognome,
COALESCE(Tot_colore.spese_totali_rosso, 0.00) AS spese_totali_rosso, Tot.spese_totali
FROM (
    SELECT p.cf , SUM(pm.prezzo) AS spese_totali_rosso
    FROM proprietario p JOIN animale a ON p.cf = a.proprietario
    JOIN visita v ON a.microchip = v.animale JOIN procedura_medica pm ON v.id_visita = pm.visita
    WHERE v.liv_urgenza = 'rosso'
    GROUP BY p.cf
) AS Tot_colore
RIGHT JOIN (
    SELECT p.cf, p.nome, p.cognome, SUM(pm.prezzo) AS spese_totali
    FROM proprietario p JOIN animale a ON p.cf = a.proprietario JOIN visita v ON a.microchip = v.animale
    JOIN procedura_medica pm ON v.id_visita = pm.visita
    GROUP BY p.cf, p.nome, p.cognome
) AS Tot
ON Tot.cf = Tot_colore.cf
ORDER BY Tot.cf;
```

cf_proprietario character	nome character varying (50)	cognome character varying (50)	spese_totali_rosso numeric	spese_totali numeric
BLSSMN85E01H501Z	Simone	Balestra	320.00	540.00
BNCLRA70C41H501Z	Laura	Bianchi	1000.00	1000.00
CRSLBR80I01H501Z	Laura	Cerasoli	200.00	200.00
FLLMRC70F01H501Z	Marco	Folli	500.00	500.00
GRFRSS80D01H501Z	Francesca	Guerra	1420.00	1704.00
LNDRGI75A01F205Z	Giorgio	Landri	0.00	426.00
MNNGLL85H01H501Z	Giacomo	Mennini	0.00	100.00
MRTCRL80B01H501Z	Carlo	Martini	600.00	850.00
PLNMLR85C01H501Z	Mara	Pelini	970.00	970.00
RCCSRA75G01H501Z	Sara	Rocca	0.00	50.00
RSSMRA85M01H501Z	Mario	Rossi	0.00	110.00
VRDGPP80A01F205Z	Giuseppe	Verdi	0.00	274.00

3. Trovare per ogni fattura con la somma dei prezzi delle procedure mediche contenute maggiore di 400 euro, il numero di fattura, il numero di procedure mediche contenute e la somma dei prezzi. Si ordini poi il risultato in ordine decrescente rispetto alla somma dei prezzi.

```
SELECT f.numero AS num_fattura, SUM(pm.prezzo) AS costo_totale, COUNT(*) AS num_procedure
FROM fattura f JOIN procedura_medica pm ON f.numero = pm.fattura
GROUP BY f.numero
HAVING SUM(pm.prezzo)>400
ORDER BY costo_totale DESC;
```

num_fattura integer	costo_totale numeric	num_procedure bigint
18	770.00	3
22	620.00	3
15	600.00	2
21	600.00	1
19	500.00	1
9	500.00	1

4.

5. Trovare codice fiscale, nome, cognome, titolo di specializzazione, numero di procedure mediche e totale dei ricavi per i primi cinque veterinari con i ricavi maggiori dell'anno 2023.

```
SELECT vet.cf_dip AS cf_veterinario, d.nome, d.cognome, vet.titolo,
COUNT(*) AS num_procedure, SUM(pm.prezzo) AS totale_ricavi
FROM veterinario vet JOIN dipendente d ON vet.cf_dip = d.cf
JOIN visita v ON vet.cf_dip = v.veterinario JOIN procedura_medica pm ON pm.visita=v.id_visita
WHERE EXTRACT(YEAR FROM v.data) = 2023
GROUP BY vet.cf_dip, d.nome, d.cognome, vet.titolo
ORDER BY totale_ricavi DESC
LIMIT 5;
```

cf_veterinario character	nome character varying (50)	cognome character varying (50)	titolo titolo_vet	num_procedure bigint	totale_ricavi numeric
GRLCST85V01H501Z	Costanza	Giraldi	dermatologo	5	899.00
MRTSTS85L01H501Z	Stefano	Martini	fisioterapista	3	850.00
LMBLRA85K01H501Z	Laura	Lombardi	dermatologo	4	234.00
FRNMLA85J01H501Z	Maria	Ferrini	dentista	2	230.00
BRTDNL85M01H501Z	Daniela	Berti	dentista	1	60.00

6. Trovare per ogni medicinale il numero di prescrizioni in cui è contenuto e il numero di specie per cui è efficace. Si ordini poi il risultato in ordine decrescente rispetto al numero di prescrizioni.

```
SELECT tab_prescrizioni.nome, tab_prescrizioni.num_prescrizioni, tab_specie.num_specie
FROM (
    SELECT m.nome, count(*) AS num_prescrizioni
    FROM medicinale m JOIN conclusione_med cm ON m.nome=cm.medicinale
    GROUP BY m.nome
) AS tab_prescrizioni
JOIN (
    SELECT m.nome, count(*) AS num_specie
    FROM medicinale m JOIN efficacia e ON m.nome=e.medicinale
    GROUP BY m.nome
) AS tab_specie
ON tab_prescrizioni.nome=tab_specie.nome
ORDER BY num_prescrizioni DESC, num_specie DESC;
```

nome [PK] character varying (50)	num_prescrizioni bigint	num_specie bigint
BACOLAM	3	4
CLADAXXA	2	1
AMOXINDOX	1	3
AMOXY ACTIVE	1	2
PREQUILLAN	1	2
CHETAMOL	1	1
KAVMOS	1	1

7. Per ogni dipendente trovare la categoria, lo stipendio, la media degli stipendi della sua categoria e la differenza tra il suo stipendio e la media della categoria. Si ordini poi il risultato in ordine decrescente rispetto allo stipendio.

```
SELECT unione_stipendi.cf_dip AS dipendente, unione_stipendi.categoria, unione_stipendi.stipendio,
ROUND(unione_stipendi.media, 2) AS media_categoria, ROUND(unione_stipendi.differenza_stipendio, 2) AS differenza_stipendio
FROM (
    (SELECT vet_stipendi.cf_dip, 'veterinario' AS categoria, vet_stipendi.stipendio, vet_media.media,
     (vet_stipendi.stipendio - vet_media.media) AS differenza_stipendio
     FROM (
         SELECT v.cf_dip, d.stipendio
         FROM veterinario v JOIN dipendente d ON v.cf_dip=d.cf
     ) AS vet_stipendi, (
         SELECT AVG(d.stipendio) AS media
         FROM veterinario v JOIN dipendente d ON v.cf_dip=d.cf
     ) AS vet_media
    )
    UNION (
        SELECT seg_stipendi.cf_dip, 'segretario' AS categoria, seg_stipendi.stipendio, seg_media.media,
        (seg_stipendi.stipendio - seg_media.media) AS differenza_stipendio
        FROM (
            SELECT s.cf_dip, d.stipendio
            FROM segretario s JOIN dipendente d ON s.cf_dip=d.cf
        ) AS seg_stipendi, (
            SELECT AVG(d.stipendio) AS media
            FROM segretario s JOIN dipendente d ON s.cf_dip=d.cf
        ) AS seg_media
    )
    UNION (
        SELECT pul_stipendi.cf_dip, 'addetto alle pulizie' AS categoria, pul_stipendi.stipendio, pul_media.media,
        (pul_stipendi.stipendio - pul_media.media) AS differenza_stipendio
        FROM (
            SELECT ap.cf_dip, d.stipendio
            FROM addetto_pulizie ap JOIN dipendente d ON ap.cf_dip=d.cf
        ) AS pul_stipendi, (
            SELECT AVG(d.stipendio) AS media
            FROM addetto_pulizie ap JOIN dipendente d ON ap.cf_dip=d.cf
        ) AS pul_media
    )
) AS unione_stipendi
ORDER BY unione_stipendi.stipendio DESC;
```

dipendente character	categoria text	stipendio numeric (10,2)	media_categoria numeric	differenza_stipendio numeric
GRLCST85V01H501Z	veterinario	2900.00	2450.00	450.00
FRNMLA85J01H501Z	veterinario	2700.00	2450.00	250.00
CLLLRA85P01H501Z	veterinario	2600.00	2450.00	150.00
SLLFRS85Q01H501Z	veterinario	2500.00	2450.00	50.00
BRTDNL85M01H501Z	veterinario	2500.00	2450.00	50.00
BGNLFR85M01H501Z	addetto alle pulizie	2400.00	1900.00	500.00
GRGMLL85O01H501Z	veterinario	2300.00	2450.00	-150.00
LMBLRA85K01H501Z	veterinario	2200.00	2450.00	-250.00
BNNCHR85R01H501Z	segretario	2200.00	2000.00	200.00
BRGSLV85U01H501Z	segretario	2200.00	2000.00	200.00
RCCMGN85N01H50…	addetto alle pulizie	2100.00	1900.00	200.00
MLIBLL85T01H501Z	segretario	2100.00	2000.00	100.00
RCCGFR85M01H501Z	segretario	2000.00	2000.00	0.00
VRNGRD85S01H501Z	segretario	2000.00	2000.00	0.00
MRTSTS85L01H501Z	veterinario	1900.00	2450.00	-550.00
LMPLMR85V01H501Z	segretario	1900.00	2000.00	-100.00
FRSGDL85W01H501Z	segretario	1800.00	2000.00	-200.00
VRDBNN85M01H50…	segretario	1800.00	2000.00	-200.00
VRNCLD85L01H501Z	addetto alle pulizie	1200.00	1900.00	-700.00

8. Per ogni tipo di procedura medica, calcolare il numero di specie e il numero di animali a cui è stata applicata.

```
SELECT pm.tipo AS tipo_procedura, COUNT(DISTINCT a.specie) AS numero_specie,
COUNT(DISTINCT v.animale) AS numero_animali
FROM procedura_medica pm JOIN visita v ON pm.visita = v.id_visita
JOIN animale a ON v.animale = a.microchip JOIN specie s ON a.specie = s.nome
GROUP BY pm.tipo;
```

tipo_procedura procedura	numero_specie bigint	numero_animali bigint
check-up	6	9
chirurgia	10	11
esame diagnostico	7	10
fisioterapia	2	3

——— Indici

Poiché i dati delle visite effettuate in determinate date vengono letti molto più frequentemente rispetto a quando vengono aggiornati, possiamo rendere più efficiente la ricerca creando un indice sulla tabella *visita* sugli attributi *veterinario* e *data*. L'indice da creare è:

```
CREATE INDEX idx_visita_veterinario_data ON visita (veterinario, data);
```

La query 3 coinvolge un JOIN tra le tabelle veterinario, dipendente, visita e procedura_medica. L'indice composito su *veterinario* e *data* nella tabella *visita* accelera il processo di ricerca delle visite effettuate dai veterinari in un determinato anno. In particolare è utile perché la query filtra le visite basate sull'anno (WHERE EXTRACT(YEAR FROM v.data) = 2023). Filtrando le righe per l'anno 2023 e per ciascun veterinario, l'indice composto riduce significativamente il numero di righe che devono essere esaminate.

6. SOFTWARE C PER L'ACCESSO AL DATABASE

Il codice `SoftwareClinica.c` presenta il main e le funzioni:

- `getInputInt(const char *prompt, int *value)` che assicura che l'input dell'utente sia un intero
- void getInputFloat(const char *prompt, float *value) che assicura che l'input dell'utente sia un float
- void printTable(PGresult *res) che stampa la query passata come parametro
- void executeQuery(PGconn *conn, PGresult **res, const char *query) che esegue la query e chiama printTable per stampare i risultati
- void showMenu() che stampa il menu delle query tra cui scegliere

Prima di compilare il programma bisogna ridefinire le costanti usate per l'accesso al database con quelle della macchina corrente. Tali costanti sono: HOST, USER, DB, PASS, PORT.

Per compilare il programma basta eseguire il comando:

```
gcc SoftwareClinica.c -L dependencies/lib -lpq -o SoftwareClinica
```

assicurandosi di essere nella cartella giusta, e per eseguirlo:

```
.\SoftwareClinica
```

Il programma mostrerà a schermo un menu da cui si potrà scegliere una query digitando il numero corrispondente. Nel caso si selezionino la query 1, 2 o 3 verrà richiesto all'utente di inserire un valore (nel caso 1 si consiglia di inserire "rosso", nel caso 2 di inserire un valore minore di 770 e nel caso 3 l'anno "2023").