Documentazione

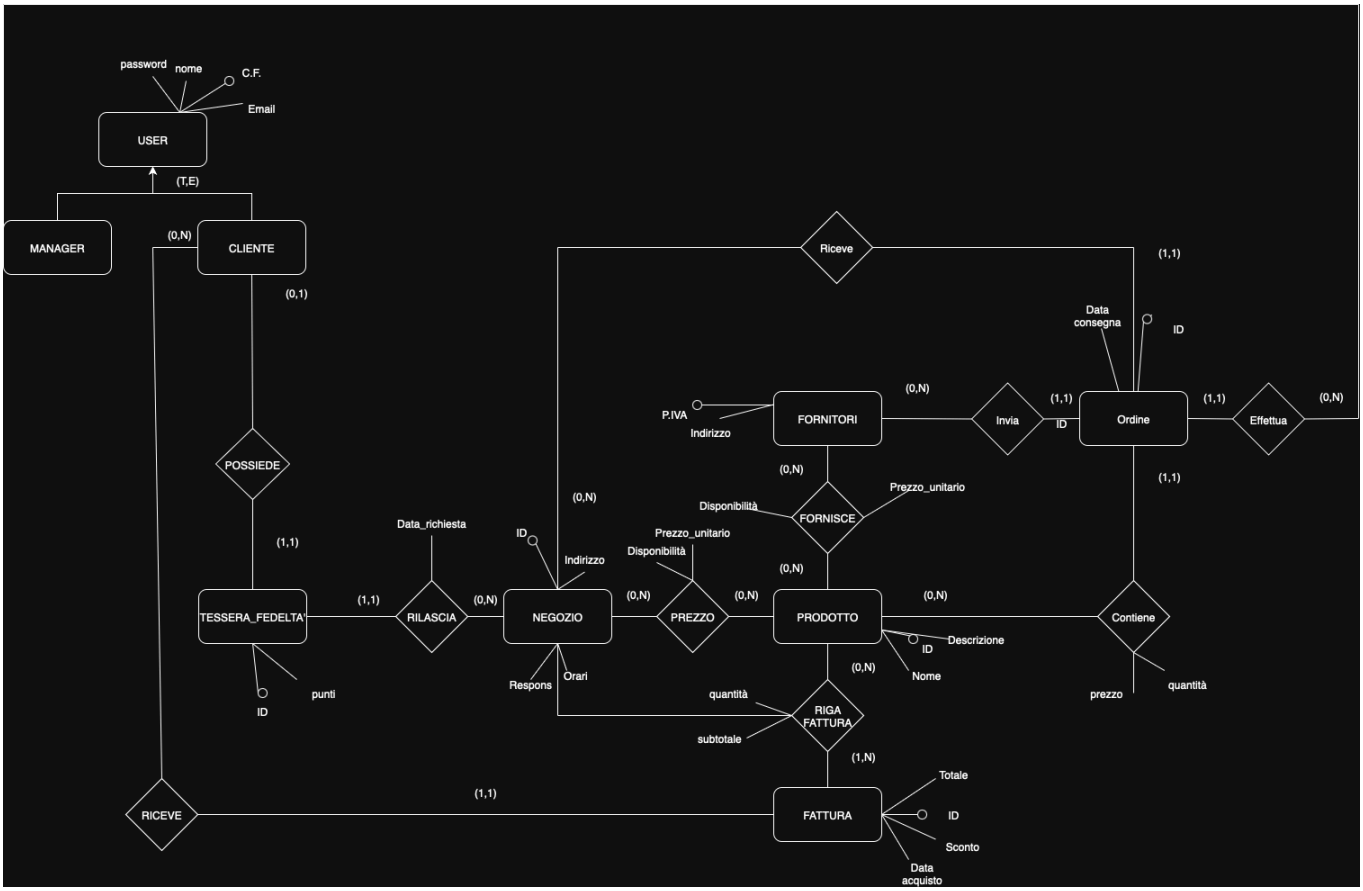
Documentazione progetto

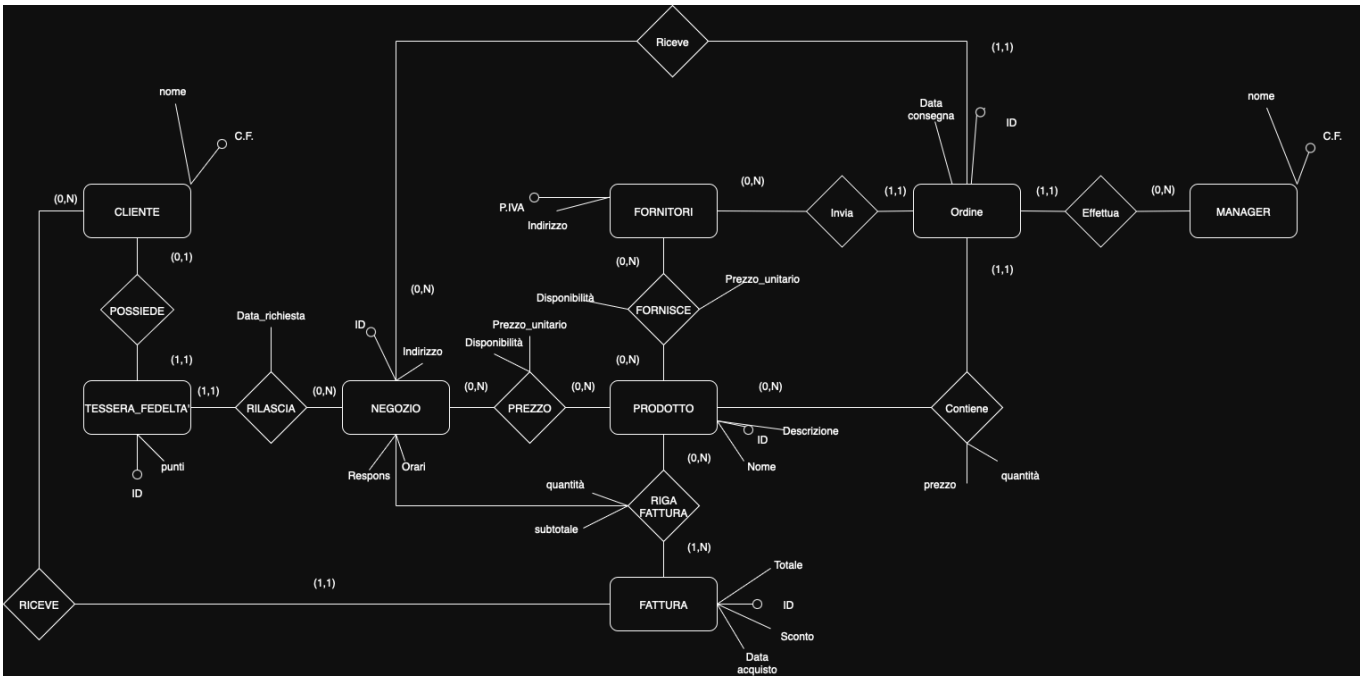
Indice

Introduzione

Il seguente progetto è stato creato secondo le specifiche dell'anno 2025/2026 per il corso di Basi di dati.
Riguarda la creazione di un gestionale di negozi e fornitori che permetta l'uso sia da parte di un manager (per gestire i vari negozi e fornitori oltre che i prodotti presenti), sia per i clienti che potranno acquistare prodotti dai vari negozi usufruendo degli sconti in caso richiedessero una tessera fedeltà ad un negozio.

1. Schema-ER





Lo schema ER ha subito una ristrutturazione per quanto riguarda la gerarchia di generalizzazione tra **Clienti** e **Manager** (gerarchia **User**), le due entità sono state suddivise in modo da permetterne un'implementazione facilitata (andando anche a suddividere meglio le relazioni non in comune tra le due).
Un'altra ristrutturazione riguarda invece la relazione **Ordine**, questa relazione è stata trasformata in un'entità a parte avente relazioni differenti con le entità **Fornitori**, **Manager**, **Negozio** e **Prodotto**.

2. Schema Logico

Per gli schemi logici di ogni tabella adotterò questi simboli:

- 'Grassetto'->Primary key
- '*'-> per valori nullable
- 'Corsivo' -> per Foreign Keys

1. cliente(**c_f**,nome,email,password) e manager(**c_f**,nome,email,password):
   in queste entità c_f è la chiave primaria mentre l'email è unica, non possono esserci quindi tuple con la stessa email.
2. prodotti(**id**,name,description)
   rileggendo la consegna, description sembra essere un elemento indispensabile e non può quindi essere nullo, id è la chiave primaria.
3. negozi(**id**,name,address,responsabile,orario_apertura,orario_chiusura)
4. fatture(**id**,*cliente*,sconto,data_emissione,totale)
   Questa entità presenta una foreign key (cliente) con vincolo ON DELETE CASCADE, inoltre lo sconto e il totale presentano entrambe un vincolo di dominio CHECK(>=0).
5. fornitori(**p_iva**,name,indirizzo)
6. fidelity_card(**id**,*cliente*,punti,*negozio*,data_rilascio)
   l'attributo punti presenta un vincolo di dominio >=0.
7. prodotto_negozio(**(*prodotto*,*negozio*)**,prezzo_unitario,disponibilita)
   Rappresenta la relazione **possiede** tra negozio e prodotto. La primary key è formata da prodotto e negozio, inoltre prezzo_unitario e disponibilita presentano vincoli di dominio >=0.
8. prodotto_fornitore(**(*prodotto*,*fornitore*)**,prezzo_unitario,disponibilita)
   Come prodotto_negozio ma rappresenta la tabella dei prodotti posseduti dai vari fornitori.
9. ordini(**id**,*fornitore*,*manager_richiedente*,*negozio*,*prodotto*, quantita, prezzo, data_consegna)
   Questa entità rappresenta gli ordini dei singoli prodotti effettuata presso i vari fornitori, l'eliminazione di una qualsiasi delle foreign key comporta la cancellazione dell'ordine, per quantita e prezzo vale sempre il vincolo di dominio >=0.
10. riga_fattura(**(*fattura*,*prodotto*,*negozio*)**,quantita,subtotale*)
    per quantita e subtotale vale il vincolo di dominio >=0, mentre la primary key è composta da fattura, prodotto e negozio, è presente inoltre una foreign key formata da prodotto e negozio che impedisce di aggiungere tuple che hanno prodotti non appartenenti al negozio indicato.
11. storico_tessere(**tessera_id_originale**,negozio_id,negozio_nome,*cliente*,punti,data_rilascio, data_eliminazione_negozio)

3. Documentazione funzionalità

Le varie funzionalità (trigger, viste, funzioni) saranno elencate seguendo l'ordine del file init.sql:

1. Trigger hash_password

Usato per clienti e manager, va a fare l'hashing della password prima dell'inserimento nelle rispettive tabelle, utilizza la funzione *md5* di psql.

2. Trigger normalizeInput

usato per poter normalizzare email e codice fiscale con LOWER e UPPER

3. Trigger validate_fidelity_card_trigger

Permette, attraverso la sua funzione di validazione, di controllare se i punti della fidelity card da inserire sono >0 e se il negozio e il cliente inseriti esistono prima che la tessera possa essere inserita nella sua tabella.

4. View catalogo_negozi

View che permette di avere un catalogo con descrizione per ogni negozio dei prodotti facendo una join tra negozi, prodotto_negozio e prodotti.

5. View tessere_maggiori_punti

View che restituisce i clienti con una tessera fedeltà avente più di 300 punti.

6. Trigger aggiorna_punti_fidelity_trigger

Un trigger che, attraverso la sua funzione **aggiorna_punti_fidelity**, controlla, ad ogni fattura inserita, se il cliente possiede una tessera fedeltà, in caso fosse così va ad aumentare il saldo punti come da specifiche.

7. Funzione lista_tessere_rilasciate(negozio_id INT)

Questa funzione ha come parametro *negozio_id* e va a restituire la lista delle tessere rilasciate dallo stesso, in caso non fossero presenti tessere allora avremo una tabella vuota.

8. Trigger convalida_ordine_prodotto

Questo trigger utilizza la funzione convalida_ordine_prodotto() prima dell'inserimento di una tupla nella tabella ordini, quando questi vengono inseriti la funzione ricerca il fornitore avente il prezzo più basso e con disponibilità >= alla quantita di prodotto richiesto (viene utilizzato un limit 1 per prendere il primo fornitore in caso ci fossero fornitori con stesso prezzo e abbastanza disponibilita) successivamente viene calcolato il totale e viene aggiunto il fornitore alla tupla (se presente, altrimenti il tutto viene annullato con una exception).

9. Trigger aggiorna_disponibilita_prodotto_fornitore

Questo trigger va a modificare la disponibilità di un certo prodotto presso un certo fornitore dopo l'inserimento dell'ordine (i controlli sulla disponibilità sono stati già fatti dal trigger precedente)

10. View sconti_clienti

Restituisce una tabella con gli sconti utilizzabili dai vari clienti in base al loro punteggio (se possiedono una carta fedeltà), l'utilizzo di una view permette al cliente di scegliere se utilizzare o no lo sconto a lui concesso (oltre a fare i relativi controlli).

11. Trigger aggiungi_prodotto_negozio

Questo trigger permette, all'inserimento di un ordine, di andare ad aggiungere nella tabella prodotto_negozio il prodotto ordinato da un certo negozio se il negozio non lo possiede ancora altrimenti va a modificare la quantità di prodotto presente in quel negozio.

12. Triggger aggiorna_disponibilita_prodotto_negozio

Questo trigger va a diminuire la quantità di un prodotto posseduto da un negozio ogni volta che viene fatta una insert in *riga_fattura*.

13. Trigger aggiorna_tessere_negozi_eliminati

Questo trigger si attiva prima dell'eliminazione di un negozio dalla tabella negozi, va a salvare tutte le tessere del negozio che sta per essere eliminato in una tabella di storico (i punti sulle tessere non potranno essere più utilizzati e i clienti dovranno richiederne di nuove), assieme ad esse sarà salvato anche id e nome del negozio,oltre che la data di eliminazione dello stesso.

14. Trigger Calcola_subtotale_riga_fattura

Questo trigger calcola internamente al db il subtotale per la riga_fattura che deve essere inserita, così da mantenere integrità rispetto al prezzo del negozio e alla quantità ed evitando che possano essere aggiunti valori inadeguati (-1 o 0). Il controllo della presenza del prodotto in un certo negozio viene fatto dalla foreign key della tabella riga_fattura.

14. Trigger valida_fattura()

Il trigger valida_fattura si attiva prima dell'inserimento di una fattura, questa avrà già lo sconto richiesto inserito. Valida fattura va quindi a controllare se lo sconto inserito è disponibile per il cliente e in caso lo applica al totale (aggiornato con NEW.totale=) e andrà a decurtare i punti

necessari allo sconto dalla tessera, in caso non fosse presente alcuna tessera fedeltà allora non verrà applicato alcuno sconto.

15. View storico_ordine_approfondito

Una view che restituisce lo storico ordine dei fornitori con approfondimenti riguardanti il fornitore stesso, il prodotto e il negozio.

16. Function storicoOrdiniFornitore

Funzione che, in base alla p_iva inserita, restituisce lo storico ordini approfondito del fornitore a cui appartiene, utilizza la view **storico_ordine_approfondito**.

4. Immagini funzionamento funzionalità richieste

4.1 Aggiornamento saldo punti su tessera fedeltà

Con tessera vuota:

```
mydb=# select * from fidelity_card;
 id |   cliente    | punti | negozio | data_rilascio
----+--------------+-------+---------+---------------
  1 | GRAZ12345678 |     0 |       2 | 2025-09-11
(1 row)

mydb=# _
```

Durante processo:

```
mydb=# select * from fidelity_card;
 id |   cliente    | punti | negozio | data_rilascio
----+--------------+-------+---------+---------------
  1 | GRAZ12345678 |     0 |       2 | 2025-09-11
(1 row)

mydb=# _
```

Processo completato:

```
mydb=# select * from fidelity_card;
 id |   cliente    | punti | negozio | data_rilascio
----+--------------+-------+---------+---------------
  1 | GRAZ12345678 |     0 |       2 | 2025-09-11
(1 row)

mydb=# _
```

4.2 Applicazione sconto sulla spesa

Con tessera vuota o senza tessera:

```
mydb=# Insert into fatture (cliente ,sconto,totale) values('MARIA12345678',5,100);
NOTICE:  Sconto non applicato per il cliente MARIA12345678
NOTICE:  Fattura rilasciata per il cliente MARIA12345678: totale originale €100.00, totale finale €100.00,sconto percentuale %0.00
NOTICE:  Cliente MARIA12345678 non ha una tessera fedeltà, punti non assegnati
INSERT 0 1
mydb=# select * from fatture;
 id |    cliente    | sconto | data_emissione | totale
----+---------------+--------+----------------+--------
  1 | MARIA12345678 |   0.00 | 2025-09-11     | 100.00
(1 row)
```

Aggiunta punti tessera:

```
NOTICE:  Sconto non applicato per il cliente GRAZ12345678
NOTICE:  Fattura rilasciata per il cliente GRAZ12345678: totale originale €300.00, totale finale €300.00,sconto percentuale %0.00
NOTICE:  Tessera fedeltà rilasciata al cliente GRAZ12345678 dal 2
NOTICE:  Punti aggiornati per il cliente GRAZ12345678: aggiunti 300 punti
INSERT 0 1
mydb=# select * from fatture;
 id |    cliente    | sconto | data_emissione | totale
----+---------------+--------+----------------+--------
  1 | MARIA12345678 |   0.00 | 2025-09-11     | 100.00
  2 | GRAZ12345678  |   0.00 | 2025-09-11     | 300.00
(2 rows)
```

Inserimento fattura:

```
mydb=# Insert into fatture (cliente ,sconto,totale) values('GRAZ12345678',15,100);
NOTICE:  Tessera fedeltà rilasciata al cliente GRAZ12345678 dal 2
NOTICE:  Sconto applicato per il cliente GRAZ12345678: €15.00
NOTICE:  Fattura rilasciata per il cliente GRAZ12345678: totale originale €100.00, totale finale €85.00,sconto percentuale %15.00
NOTICE:  Tessera fedeltà rilasciata al cliente GRAZ12345678 dal 2
NOTICE:  Punti aggiornati per il cliente GRAZ12345678: aggiunti 85 punti
INSERT 0 1
```

Decurtazione punti:

```
mydb=# select * from fidelity_card;
 id |   cliente    | punti | negozio | data_rilascio
----+--------------+-------+---------+---------------
  1 | GRAZ12345678 |    95 |       2 | 2025-09-11
(1 row)
```

4.3 Lista tesserati dato un negozio

```
mydb=# select * from lista_tessere_rilasciate(1);
 tessera_id | cliente_c_f | punti | data_rilascio
------------+-------------+-------+---------------
(0 rows)

mydb=# select * from lista_tessere_rilasciate(2);
 tessera_id | cliente_c_f  | punti | data_rilascio
------------+--------------+-------+---------------
          1 | GRAZ12345678 |    95 | 2025-09-11
(1 row)
```

4.4 Saldo punti > 300

Punti prima:

```
mydb=# select * from fidelity_card;
 id |    cliente    | punti | negozio | data_rilascio
----+---------------+-------+---------+---------------
  1 | GRAZ12345678  |   195 |       2 | 2025-09-11
  2 | MARIA12345678 |   300 |       1 | 2025-09-11
(2 rows)
```

Tessere > 300:

```
mydb=# select * from tessere_maggiori_punti;
  cliente_c_f  | cliente_nome | tessera_id | punti | data_rilascio
---------------+--------------+------------+-------+---------------
 MARIA12345678 | Maria        |          2 |   300 | 2025-09-11
(1 row)
```

4.5 Storico ordini a fornitori

Ordini totali:

```
mydb=# select * from ordini;
 id |   fornitore   | manager_richiedente | negozio | prodotto | quantita | prezzo | data_consegna
----+---------------+---------------------+---------+----------+----------+--------+---------------
  1 | IT11111111111 | MGMT12345678        |       2 |        5 |        5 |     11 | 2025-09-18
  3 | IT22222222222 | MGMT12345678        |       1 |        4 |        9 |      7 | 2025-09-18
  4 | IT22222222222 | MGMT12345678        |       1 |        2 |       80 |     40 | 2025-09-18
(3 rows)
```

Ordini fornitore 'IT11111111111':

```
mydb=# select * from storicoordinifornitore('IT11111111111');
 id | id_fornitore  |     nome_fornitore      | manager_richiedente | id_negozio | nome_negozio | id_prodotto | nome_prodotto | quantita | prezzo | data_consegna
----+---------------+-------------------------+---------------------+------------+--------------+-------------+---------------+----------+--------+---------------
  1 | IT11111111111 | Cartoleria Centrale SRL | MGMT12345678        |          2 | Negozio 2    |           5 | Righello 30cm |        5 |     11 | 2025-09-18
(1 row)
```

4.6 Mantenimento storico tessere

Cosiderando:

```
mydb=# select * from fidelity_card;
 id |    cliente    | punti | negozio | data_rilascio
----+---------------+-------+---------+---------------
  1 | GRAZ12345678  |   195 |       2 | 2025-09-11
  2 | MARIA12345678 |   300 |       1 | 2025-09-11
(2 rows)
```

Elimino il negozio 2:

```
mydb=# DELETE FROM negozi where id = 2;
NOTICE:  Tessera 1 del cliente GRAZ12345678 spostata nello storico
DELETE 1
mydb=# _
```

Tabella di storico:

```
mydb=# select * from storico_tessere;
 tessera_id_originale | negozio_id | negozio_nome |   cliente    | punti | data_rilascio | data_eliminazione_negozio
----------------------+------------+--------------+--------------+-------+---------------+---------------------------
                    1 |          2 | Negozio 2    | GRAZ12345678 |   195 | 2025-09-11    | 2025-09-11
(1 row)
```

Tessere rimaste:

![Tessere rimaste](./Immagini_funzioni/Tessere_rimaste.

4.7 e 4.8 Ordine prodotti da fornitore e aggiornamento disponibilità prodotti da fornitore

Considero il seguente prodotto da ordinare:

```
mydb=# select * from prodotti where id = 3;
 id |   name    |                    description
----+-----------+-----------------------------------------------------
  3 | Matita HB | Matita in grafite durezza HB per disegno e scrittura
(1 row)
```

E il prezzo da tutti i fornitori:

```
mydb=# select * from prodotto_fornitore where prodotto = 3;
 prodotto |   fornitore   | prezzo_unitario | disponibilita
----------+---------------+-----------------+---------------
        3 | IT11111111111 |            0.40 |           800
        3 | IT22222222222 |            0.45 |           600
(2 rows)
```

Ordino 365 unità di questo prodotto per il negozio 1 che attualmente ha 150 unità:

```
mydb=# select * from prodotto_negozio where prodotto = 3;
 prodotto | negozio | prezzo_unitario | disponibilita
----------+---------+-----------------+---------------
        3 |       1 |            0.60 |           150
(1 row)
```

La funzione ha scelto il prodotto dal fornitore con prezzo_unitario minore ovvero:

```
mydb=# INSERT INTO ordini (negozio,prodotto,quantita,manager_richiedente) values (1,3,365,'MGMT12345678');
NOTICE:  Ordine 6 creato: 365 unità del prodotto 3 dal fornitore IT11111111111 al prezzo di €0.40 per unità. Manager: MGMT12345678.
NOTICE:  Disponibilità aggiornata per il prodotto 3 dal fornitore IT11111111111: -365 unità
INSERT 0 1
mydb=# _
```

Controlliamo ora la disponibilità aggiornata dal negozio:

```
mydb=# select * from prodotto_negozio where prodotto = 3;
 prodotto | negozio | prezzo_unitario | disponibilita
----------+---------+-----------------+---------------
        3 |       1 |            0.60 |           515
(1 row)
```

E dal fornitore:

```
mydb=# select * from prodotto_fornitore where prodotto = 3 and fornitore = 'IT11111111111';
 prodotto |   fornitore   | prezzo_unitario | disponibilita
----------+---------------+-----------------+---------------
        3 | IT11111111111 |            0.40 |           435
(1 row)
```