ECloth Mountain

1 Abstract

"ECloth Mountain" è un'azienda speccializzata nella vendita di abbigliamento tecnico da montagna, ha appena aperto il proprio sito di e-commerce e ha commissionato il proprio database.
Dal sito di "ECloth Mountain" è possibile registrarsi e creare un account, per salvare i propri dati e per poter effettuare acquisti. Inoltre è possibile creare varie liste di prodotti, per poterli salvare e visualizzare in un secondo momento.
L'utente per registrarsi deve inserire i propri dati personali, mentre l'indirizzo e il metodo di pagamento non sono obbligatori. L'indirizzo e il metodo di pagamento sono necessari al momento dell'acquisto, ma l'utente può decidere di non salvarli. L'utente attraverso il proprio account può rivedere gli acquisti effettuati e le liste di prodotti salvate.

2 Analisi dei requisiti

2.1 Descrizione testuale

Nella base di dati sono presenti i dati degli **utenti**, che si registrano sul sito per effettuare gli acquisti. Di ogni utente sono noti:

- nome
- cognome
- email
- password

Ogni utente può memorizzare i dati di una **carta di credito**. Di ogni carta di credito sono noti:

- numero della carta
- intestatario della carta
- data di scadenza
- codice di sicurezza (CVV)

Ogni utente può memorizzare i dati di vari indirizzi di spedizione. Di ogni **indirizzo** sono noti:

- via
- numero civico
- città
- CAP
- stato

Ogni utente può avere diversi carrelli. Di ogni **carrello** sono noti:

- nome del carrello

- costo totale
- data di creazione
- data di ultima modifica
- prodotti contenuti

Ogni utente può effettuare un **ordine**, effettuando il *checkout* del carrello desiderato. Di ogni ordine sono noti:

- numero dell'ordine
- costo totale
- data di creazione
- data di ultima modifica
- prodotti contenuti
- provider del pagamento
- status dell'ordine
- indirizzo di spedizione

I pagamenti sono gestiti mediante un provider esterno, che si occupa di effettuare il pagamento e di notificare il sito di "ECloth Mountain" del risultato dell'operazione.
Gli utenti possono acquistare dei prodotti, presenti sul sito. Di ogni **prodotto** sono noti:

- nome
- descrizione
- prezzo
- quantità
- tag

Non è possibile acquistare un prodotto se la quantità disponibile è minore di quella richiesta.

Termine	**Descrizione**	**Collegamenti**
Utente	Persona che si registra sul sito	Carta di credito, Lista, Indirizzo
Carta di credito	Carta di credito di un utente	Utente
Indirizzo	Indirizzo di spedizione di un utente	Utente, Ordine
Lista	Lista di prodotti, con relativo costo totale	Utente, Prodotto
Carrello	Lista con un nome	Entità figlia di Lista
Ordine	Lista con un identificativo e i dettagli di pagamento	Indirizzo, Entità figlia di Lista
Prodotto	Prodotto in vendita sul sito	Lista, Categoria
Categoria	Categoria di prodotti	Prodotto

Table 1: Glossario dei termini

Operazione	Tipo	Frequenza
Inserimento di un nuovo Utente	U	1 volta al giorno
Inserimento di una Carta di Credito	U	5 volte al giorno
Inserimento di un Indirizzo	U	20 volte al giorno
Inserimento di un nuovo Carrello	U	20 volte al giorno
Cancellazione di un Carrello	U	10 volte al giorno
Visione di un Carrello	U	100 volte al giorno
Aggiungi un Prodotto al Carrello	U	100 volte al giorno
Rimuovi un Prodotto dal Carrello	U	90 volte al giorno
Inserimento di un nuovo Ordine	U	10 volte al giorno
Cancellazione di un Ordine	U	1 volte al giorno
Visione di un Ordine	U	10 volte al giorno
Inserimento di un nuovo Prodotto	B	10 volte alla settimana
Calcolare il guadagno mensile	B	1 volta al mese
Calcolare il guadagno quotidiano	B	1 volta al giorno

Table 2: Operazioni

3 Progettazione concettuale

3.1 Lista delle entità

- Utente
  - email VARCHAR(255) PK NOT NULL
  - password VARCHAR(255) NOT NULL
  - nome VARCHAR(255) NOT NULL
  - cognome VARCHAR(255) NOT NULL
- Carta di credito
  - numero VARCHAR(16) PK NOT NULL
  - intestatario VARCHAR(255) NOT NULL
  - scadenza DATE NOT NULL
  - cvv INT NOT NULL
- Indirizzo
  - via VARCHAR(255) PK NOT NULL
  - numero PK INT NOT NULL
  - cap VARCHAR(5) PK NOT NULL
  - citta VARCHAR(255) NOT NULL
  - stato VARCHAR(255) NOT NULL
- Lista
  - id utente INT PK NOT NULL
  - data creazione DATE PK NOT NULL
  - data modifica DATE NOT NULL

  - costo totale FLOAT NOT NULL

    L'entità Lista si specializza in due sottocategorie con una generalizzazione totale:

    - Carrello
      - nome VARCHAR(255) NOT NULL
    - Ordine
      - numero INT NOT NULL
      - dettagli pagamento: attributo composto:
        - ☐ provider VARCHAR(255) NOT NULL
        - ☐ status VARCHAR(255) NOT NULL

- Prodotto
  - nome VARCHAR(255) PK NOT NULL
  - tag VARCHAR(255) PK NOT NULL
  - descrizione VARCHAR(255)
  - prezzo FLOAT NOT NULL
  - quantità INT NOT NULL
- Categoria
  - nome VARCHAR(255) PK NOT NULL
  - descrizione VARCHAR(255)

Relazione	**Entità coinvolte**	**Descrizione**	**Attributi**
Metodo pagamento	Carta di credito (1, 1), Utente (0, 1)	Un utente può avere al più una carta di credito, una carta di credito si riferisce ad un solo cliente	Nessuno
Indirizzo spedizione	Indirizzo (0, N), Utente (0, N)	Un utente può avere più indirizzi di spedizione, o non averne uno affatto. Un indirizzo può essere associato a più clienti e può non avere nessun cliente	Nessuno
	Lista (1, 1), Utente (0, N)	Un utente può avere più liste, o non averne affatto. Una lista si riferisce ad un utente	Nessuno
Contenuto	Lista (0, N), Prodotto (0, N)	Una lista può contenere più prodotti, un prodotto può essere contenuto in più liste	quantità INT > 0
Spedizione	Ordine (1, 1), Indirizzo (0, N)	Un ordine può essere spedito ad un solo indirizzo. Un indirizzo può essere associato a più ordini	Nessuno
Tag	Prodotto (1, 1), Categoria (0, N)	Un prodotto appartiene ad una categoria, una categoria può avere più prodotti	Nessuno

Table 3: Tabella delle relazioni

3.2 Schema Concettuale

Vincoli non rappresentabili tramite schema E-R:

- Un indirizzo è associato ad un utente o ad un ordine (o non esclusivo).

Vincoli di derivazione:

- L'attributo costo di una lista è dato dalla somma dei prezzi dei prodotti contenuti nella lista moltiplicati per la quantità di ciascuno di essi.

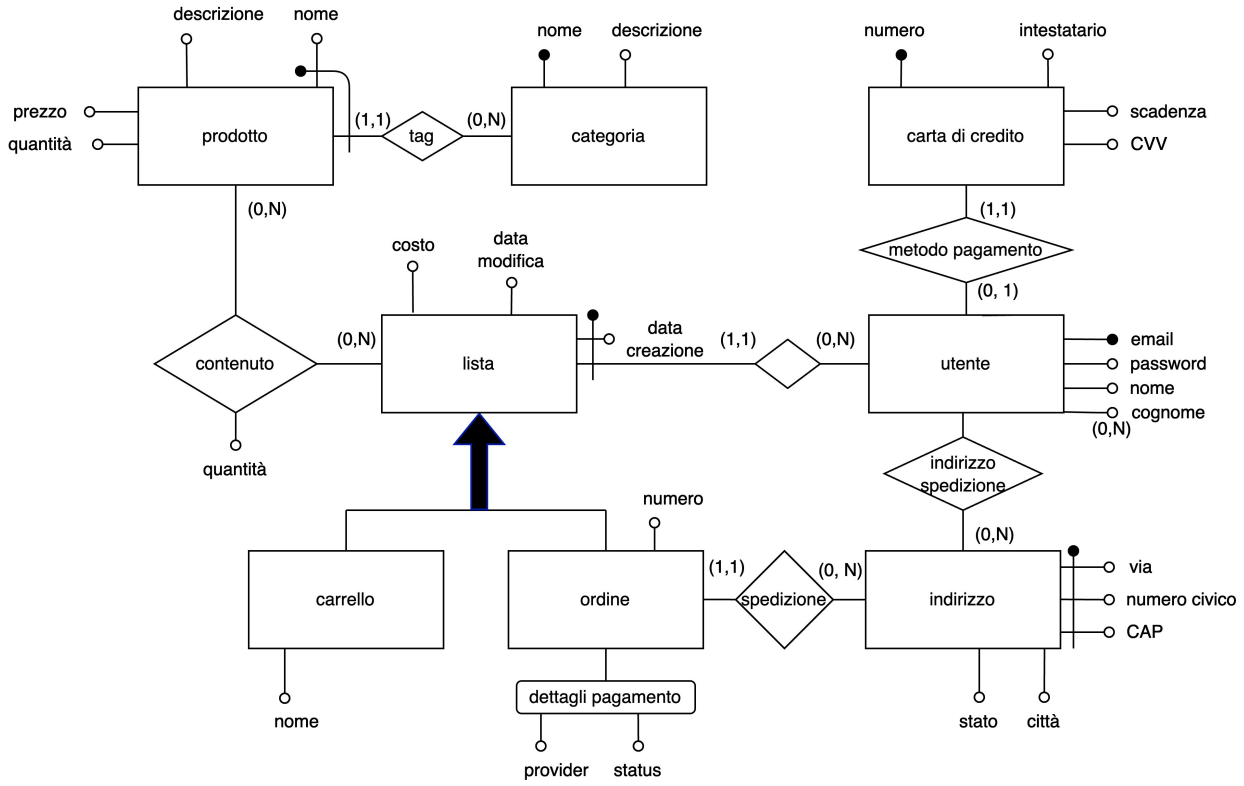

Figure 1: Schema concettuale

4 Progettazione logica

4.1 Ristrutturazione

Analisi delle ridondanze

L'attributo "Costo" dell'entità Lista è ridondante, in quanto può essere calcolato come somma dei prezzi dei prodotti contenuti nella lista moltiplicati per la quantità di ciascuno di essi. L'attributo "Costo" viene utilizzato nelle operazioni:

1. Aggiungi Prodotto al Carrello
2. Rimuovi Prodotto dal Carrello
3. Visualizza Carrello
4. Visualizza Ordine

Poichè le operazioni 1 e 2 modificano l'attributo "Costo" dell'entità Lista e entrambe queste due operazioni sono molto più frequenti dell'operazione 3, si toglie l'attributo "Costo" dall'entità Lista e si calcola il costo della lista ogni volta che viene richiesta l'operazione 3. Si potrebbe pensare di aggiungere l'attributo "Costo" all'entità Ordine, ma l'operazione 4 è poco frequente.

Generalizzazione	**Risoluzione**
Lista ← Carrello, Ordine	L'entità Carrello è accorpata in Lista, con l'aggiunta di un attributo "nome VARCHAR(255) not null". L'entità Ordine è rappresentata mediante una relazione tra Lista (0,1) e Indirizzo (0, N), dove gli attributi di Ordine diventano attributi della relazione.

Table 4: Eliminazione delle generalizzazioni

Scelta degli identificatori primari

È stato introdotto l'attributo "ID" per le entità Prodotto, Indirizzo e Lista, dunque è stato rimosso l'attributo "Numero" dall'entità Ordine.

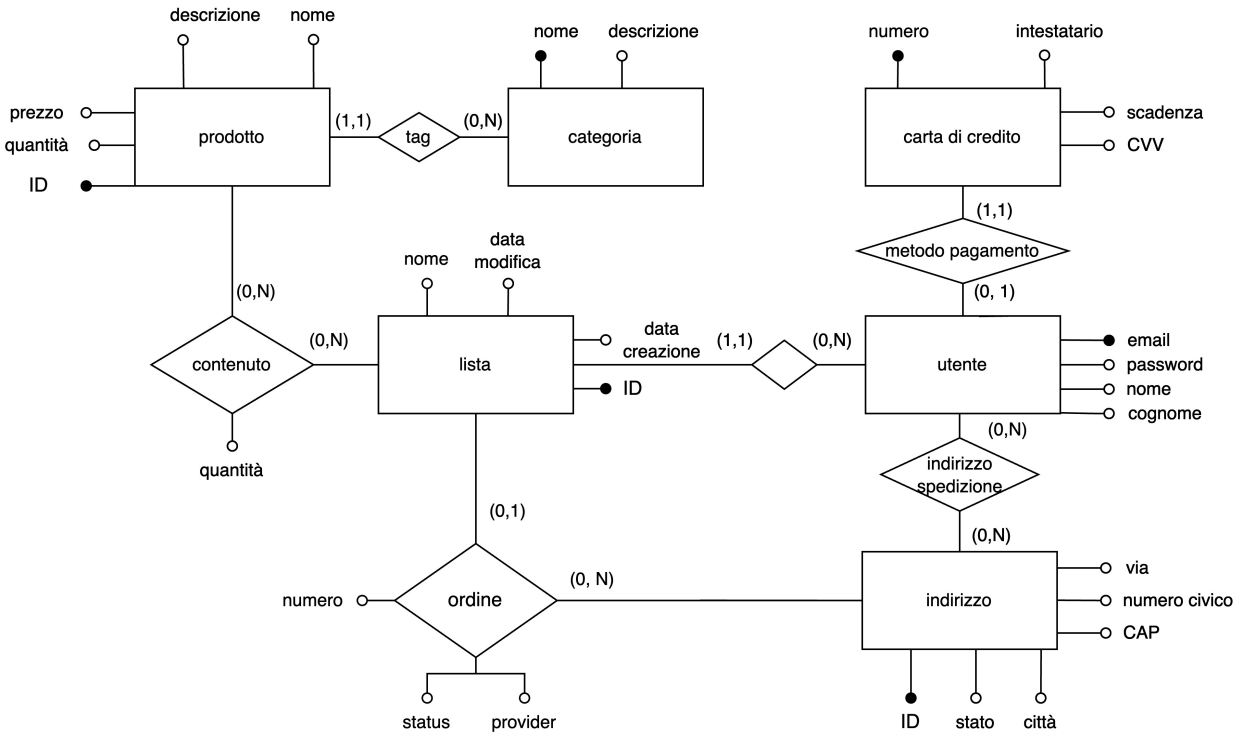

Figure 2: Schema E-R ristrutturato

Creazione delle tabelle

$A \rightarrow B$ indica che B è chiave esterna di A.

Carta di credito (numero, intestatario, scadenza, CVV, utente → Utente.email)

Utente (email, password, nome, cognome)

Indirizzo spedizione (utente → Utente.email, indirizzo → Indirizzo.ID)

Indirizzo (ID, via, numero civico, CAP, città, stato)

Ordine (lista → Lista.ID, indirizzo → Indirizzo.ID, status, provider)

Lista (ID, nome, utente → Utente.email, data creazione, data modifica)

Contenuto (lista → Lista.ID, prodotto → Prodotto.ID, quantità)

Prodotto (ID, nome, descrizione, prezzo, quantità, tag → Categoria.nome)

Categoria (nome, descrizione)

5 Here I write the queries

```
SELECT lista.utente, sum(contenuto.quantita)
FROM lista, contenuto
WHERE lista.id = contenuto.lista
AND lista.data_creazione > '2023-01-01'
GROUP BY (lista.utente);

SELECT utente.email, count(utente.email)
FROM utente, lista, ordine
WHERE utente.email = lista.utente
AND lista.id = ordine.lista
GROUP BY (utente.email)

SELECT email, count(email)
FROM ((
        SELECT utente.email, lista.id from utente, lista
        WHERE utente.email = lista.utente) as tab
        LEFT OUTER JOIN ordine
        ON tab.id = ordine.lista)
WHERE lista IS NULL
GROUP BY email

SELECT ordine.lista,
        sum(prodotto.prezzo * contenuto.quantita) as totale
FROM ordine, prodotto, contenuto
WHERE ordine.lista = contenuto.lista
AND prodotto.id = contenuto.prodotto
GROUP BY ordine.lista

SELECT lista.nome, totale
FROM (
        SELECT tab.id, sum(prezzo * quantita) as totale
        FROM (
                SELECT lista.id as id, prodotto.prezzo,
                        contenuto.quantita
                FROM lista, prodotto, contenuto
```

```
                WHERE lista.id = contenuto.lista
                AND prodotto.id = contenuto.prodotto
        ) as tab
        LEFT JOIN ordine
        ON tab.id = lista
        WHERE indirizzo is NULL
        GROUP BY tab.id) as tab2, lista
WHERE tab2.id = lista.id;

SELECT DATE(data_creazione) AS day, sum(totale)
FROM lista, (
        SELECT ordine.lista,
                sum(prodotto.prezzo * contenuto.quantita) AS totale
        FROM ordine, prodotto, contenuto
        WHERE ordine.lista = contenuto.lista
        AND prodotto.id = contenuto.prodotto
        GROUP BY ordine.lista) as costo_ordini
WHERE lista.id = lista
GROUP BY day;

SELECT EXTRACT(YEAR from data_creazione) AS year,
        EXTRACT(MONTH from data_creazione) AS month, sum(totale)
FROM lista, (
        SELECT ordine.lista,
                sum(prodotto.prezzo * contenuto.quantita) AS totale
        FROM ordine, prodotto, contenuto
        WHERE ordine.lista = contenuto.lista
        AND prodotto.id = contenuto.prodotto
        GROUP BY ordine.lista) AS costo_ordini
WHERE lista.id = lista
GROUP BY year, month
ORDER BY year, month;

SELECT p.tag, sum(p.prezzo * c.quantita) as totale
FROM prodotto as p, contenuto as c, ordine as o
WHERE p.id = c.prodotto
AND o.lista = c.lista
GROUP BY p.tag
HAVING sum(p.prezzo * c.quantita) > 100
ORDER BY totale DESC;
```

6 Esecuzione queries in codice in cpp

6.1 Strutturazione

Per l'esecuzione delle queries precedenti, si è scelto di scrivere un singolo file cpp contenente il codice in grado di svolgere le seguenti funzionalità:

- connessione al database.
- esecuzione di ogni interrogazione.
- stampa dei risultati per una generica interrogazione.

Ogni interrogazione è stata messa all'interno di una procedura avente la firma query(PGconn*); all'interno di essa si trova il codice necessario per poterla eseguire la query e stamparne i risultati.

6.2 Esecuzione

Per poter eseguire una delle interrogazioni disponibili, all'interno del main viene creata una mappa di puntatori a funzione aventi la firma query(PGconn*); in particolare la mappa è del tipo: std::map¡unsigned int, void (*)(PGconn*)¿.

Viene poi creata una mappa del tipo std::map¡unsigned int, std::string¿ contenente le singole descrizioni del risultato di ogni query.

A questo punto viene instaurata la connessione al database mediante la funzione checkConnection(PGconn*); nel caso in cui la connessione sia andata a buon fine, si entra in un ciclo do while che esegue le seguenti istruzioni:

1. stampa della mappa contenente le descrizioni delle query.
2. viene chiesto all'utente di immettere un numero di quelli specificati dalle descrizioni precedentemente stampate.
3. tramite il numero ottenuto, viene eseguita la query associata ad esso mediante la mappa di puntatori a funzine

Il codice contenuto nel main è il seguente.

```
int main() {
        std::map<unsigned int, std::string> messaggi;
        // inizializzazione messaggi
        std::map<unsigned int, void (*)(PGconn*)> queries;
        // inizializzazione queries


        // SETUP CONNESSIONE al DB
        PGconn* conn;
        char conninfo [250];
        sprintf(conninfo,
        "user=%s␣password=%s␣dbname=%s␣hostaddr=%s␣port=%d" ,
        PG_USER , PG_PASS , PG_DB , PG_HOST , PG_PORT ) ;

        // APERTURA DELLA CONNESSIONE
        conn = PQconnectdb(conninfo);

        // CONTROLLO DELLA CONNESSIONE
        checkConnection(conn);


        unsigned int operazione = 0;
        do {
                print_messages(messaggi);
                std::cin >> operazione;
                if (operazione != 0) {
                        queries[operazione](conn);
                }
```

```
        } while(operazione != 0);
}
```