Ranjeet kumar 0512110143

PROGETTO DI BASE DI DATI

ECOMMERCE

Descrizione

Si vuole progettare una base di dati che gestisca le informazioni relative ad un sito ecommerce.

L'obiettivo principale dell'ecommerce è creare un ambiente di shopping online conveniente per i clienti.Ogni cliente è identificato da idCliente.Di un cliente si vogliono conoscere anche tutte le informazioni essenziali come nome,cognome,numero di telefono,username e password.Gli attributi "username" e "password" consentono al cliente di autenticarsi e accedere al proprio account, garantendo la privacy e la sicurezza delle informazioni personali.

Dopo aver effettuato l'accesso, i clienti avranno la possibilità di visualizzare un elenco completo dei prodotti disponibili per l'acquisto, consentendo di effettuare ordini .Ogni prodotto ha come identificatore univoco un idProdotto,inoltre si vuole conoscere anche il nome,una breve descrizione,il prezzo,il peso e la disponibilità.I prodotti appartengono ad una categoria.quest'ultima è identificato attraverso un idCategoria,sono caratterizzati da un nome e da una breve descrizione della categoria stessa.

Ogni prodotto può essere fisicamente disponibile in uno o più magazzini.

Ordini sono identificati univocamente da un idOrdine ed interessa conoscere la data .Dopo aver confermato l'ordine si passa al pagamento.

Il pagamento viene identificato univocamente da un idPagamento ,si vuole conoscere anche la data in cui è stato effettuato il pagamento e l'importo totale dell'ordine .Il pagamento può essere di due tipi:contrassegno e carta di credito.

Specifiche della realtà d’interesse

La realtà che andiamo a rappresentare riguarda la gestione di un ecommerce sul quale è possibile effettuare ordini da parte di clienti iscritti.

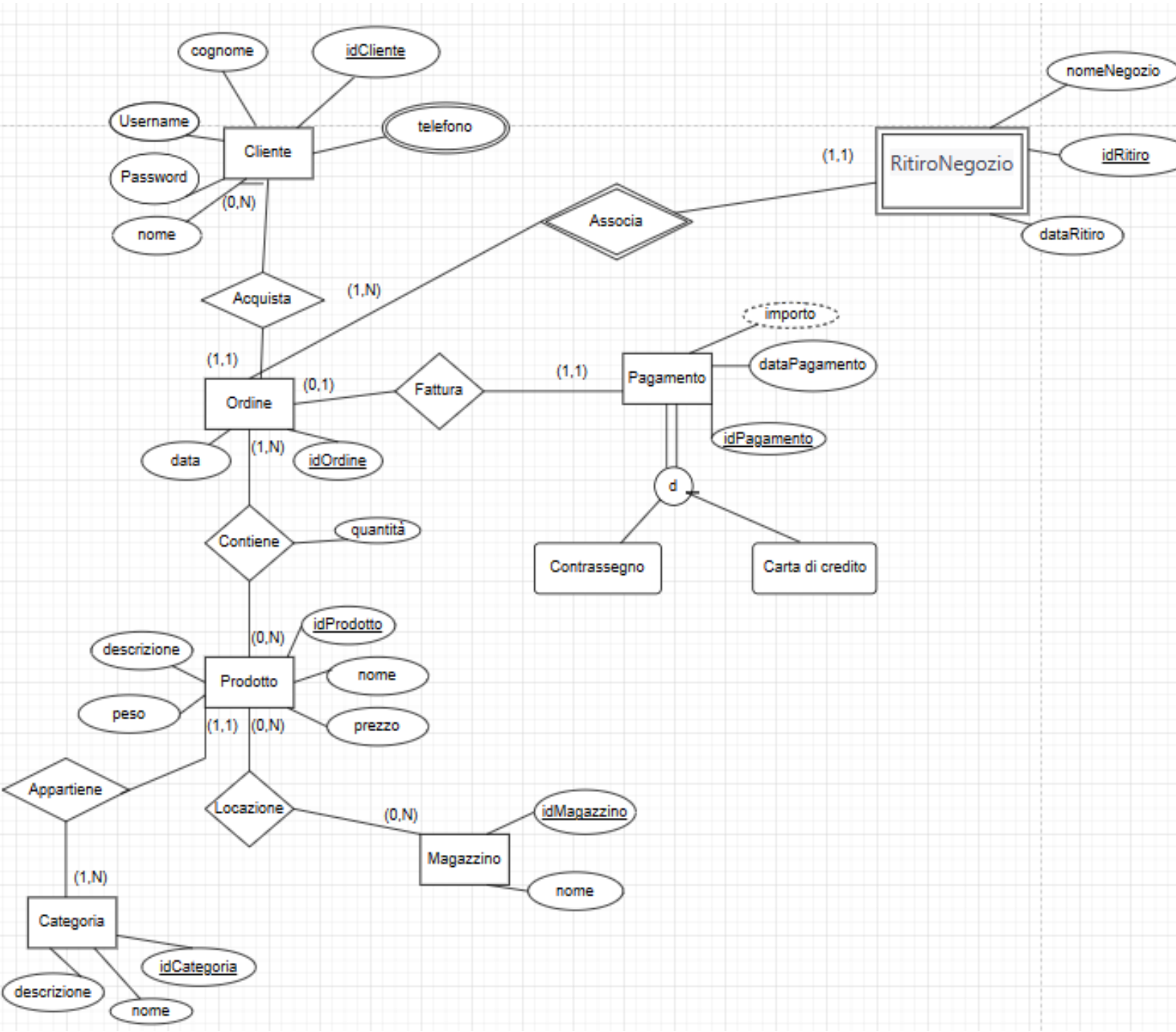
All'interno di ecommerce, i clienti avranno accesso a una vasta gamma di prodotti suddivisi in diverse categorie, offrendo una soluzione completa per soddisfare le esigenze di ogni utente. La struttura di categorie agevolerà la ricerca dei prodotti desiderati, semplificando l'esperienza di acquisto per gli utenti. quando un cliente effettua un ordine, i prodotti richiesti devono essere prelevati dal magazzino dell'azienda. Il "magazzino" è il luogo fisico dove vengono archiviati i prodotti mentre il "ritiro in negozio" è un servizio che consente ai clienti di ritirare fisicamente i loro ordini presso un negozio fisico dell'azienda
Ogni entità necessità di un identificativo per l'identificazione univoca.

GLOSSARIO DEI TERMINI

Termine	Significato
Cliente	Persona che effettua almeno un ordine nel sito.
Ordine	Insieme di tutte le informazioni relative ad un acquisto da parte di un Cliente.
RitiroNegozio	rappresenta le informazioni relative al ritiro di un ordine presso un negozio fisico
Pagamento	Pagamento relativo ad un acquisto.
Contrassegno	Modalità di pagamento
Carta di Credito	Modalità di pagamento
Prodotto	Tutte le informazioni relative ad un prodotto
Magazzino	Magazzino nel quale sono disponibili i prodotti.

Categoria	Categoria nella quale sono organizzati prodotti dello stesso tipo.

2. Progettazione concettuale della base di dati Schema EER



Dizionario delle entità

Legenda: sotto-entità, attributo multivalore, attributo ridondante, entità debole, chiave candidata

Entità	Descrizione	Attributi	Identificatore
Cliente	Persona che effettua almeno un ordine nel sito.	idCliente,nome,cognome,Username, password,telefono	idCliente
Ordine	Insieme di tutte le informazioni relative ad un acquisto da parte di un Cliente.	idOrdine ,data	idOrdine
RitiroNegozio	rappresenta le informazioni relative al ritiro di un ordine presso un negozio fisico	idRitiro ,dataRitro, nomeNegozio	idRitiro
Pagamento	Pagamento relativo ad un acquisto.	idPagamento, dataPagamento, importo	idPagamento
Contrassegno	Modalità di pagamento	/	/
Carta di Credito	Modalità di pagamento	/	/
Prodotto	Tutte le informazioni relative ad un prodotto	idProdotto,nome, descrizione,prezzo ,peso	idProdotto
Magazzino	Magazzino nel quale sono disponibili i prodotti venduti.	idMagazzino,nome Magazzino	idMagazzino

Categoria	Categoria nella quale sono organizzati prodotti dello stesso tipo.	idCategoria, nomeCategoria, descrizione	idCategoria

Dizionario delle relazioni

Relazione	Descrizione	Entità Coinvolte	Attributi
Associa	Ogni dettaglio di ritiro in negozio è associato a un ordine specifico	Ordine(1,N) RitiroNegozio(1,1)	/
Acquista	l'immissione di un ordine da parte di un cliente	Cliente(0,N) Ordine(1,1)	/
Fattura	definisce il pagamento dell'ordine	Ordine(1,1) Pagamento(1,1)	/
Contiene	la presenza di determinati prodotti all'interno di un ordine	Ordine(1,N) Prodotto(0,N)	quantità
Appartiene	un prodotto appartiene ad una categoria	Prodotto(1,1) Categoria(1,N)	/
locazione	la posizione presso la quale sono presenti i prodotti	Prodotto(0,N) Magazzino(0,N)	/

Vincoli non esprimibili nello schema

Come sappiamo lo schema concettuale ER è sufficientemente espressivo per rappresentare i dati, ma non lo è altrettanto per la rappresentazioni di vincoli di integrità o di derivazione, di seguito elencati.

- L'attributo "descrizione" deve avere numero di caratteri minore o uguale a 250.

Definizione delle procedure per la gestione della base di dati

Tavola dei volumi

Definiamo di seguito la tavola dei volumi della base di dati.

Concetto	Tipo	Carico Applicativo
Cliente	E	50
RitiroNegozio	E	350
Ordine	E	250
Pagamento	E	250
Prodotto	E	400
Magazzino	E	2
Categoria	E	10
Acquista	R	250
Associa	R	350

Contiene	R	500
Fattura	R	250
Appartiene	R	40
locazione	R	500
Contrassegno	E	100
Carta di Credito	E	150

Tavola delle operazioni

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

Operazione	Tipo	Frequenza
Op1: Aggiungere un nuovo cliente	I	20/mm
Op2: Aggiungere un nuovo prodotto	I	20/aa
Op3: Visualizzare tutti i prodotti	I	15/mm
Op4: Trovare i prodotti più venduti	B	4/aa
Op5: Trovare i prodotti meno venduti	B	4/aa
Op6: Trovare il prodotto più costoso	B	4/aa
Op7 :Trovare i clienti che hanno effettuato più di 1 ordine	I	3/aa
Op 8:Elencare i prodotti in ordine di prezzo dal più basso al più alto.	I	4/mm
Op9: Selezionare gli importi effettuati da un dato cliente	I	5/mm
Op10 : Trovare i prodotti con prezzo maggiore di 50 euro.	B	3/aa

Op11: Trovare i prodotti che non sono mai stati venduti.	B	4/aa
Op12: Trovare la data dell'ultimo ordine effettuato da ciascun cliente.	B	2/mm
Op13:Trovare le categorie con la maggior varietà di prodotti distinti.	B	5/aa
Op14:Selezionare tutti gli ordini pagati con Carta di credito.	B	2/mm
Op15: Visualizzare tutti gli ordini eseguiti in un anno.	I	2/aa
Op16:Seleziona i ritiri nei negozi e il cliente associato con il maggior numero totale di ritiri	B	5/aa

Progettazione logica

Analisi delle ridondanze

Il dato ridondante è l'attributo "importo" e ci la quantità di denaro che il cliente deve spendere per effettuare l'acquisto.Questo è un dato ridondante perchè tramite dei calcoli è possibile risalire al valore del dato ma è da verificare se effettivamente ,conviene eliminarlo.

Tavola degli accessi

Operazione 9

Calcolo con ridondanza			
Tabella	**Tipo**	**Accessi**	**Tipo accessi**
Cliente	E	1	L
Acquista	R	5	L
Ordine	E	5	L
Fattura	R	5	L
Pagamento	E	5	L
Totale		(21L)=21*5/mm=105 a/mm	

Calcolo senza ridondanza			
Tabella	**Tipo**	**Accessi**	**Tipo accessi**
Cliente	E	1	L
Acquista	R	5	L

Ordine	E	5	L
Contiene	R	5*2=10	L
Prodotto	R	5*2=10	L
Totale		(31L)=31*5/mm=155 a/mm	

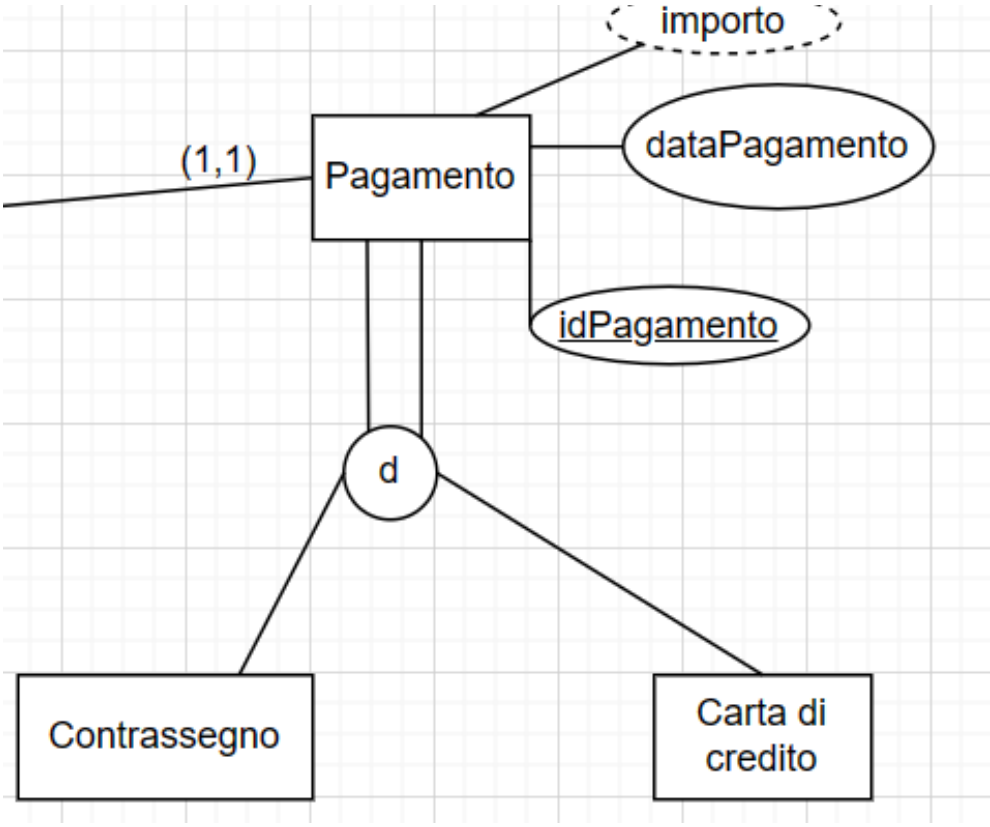
Totale accessi con ridondanza : 105a/mm +1000 byte

Totale accessi senza ridondanza: 155 a/mm

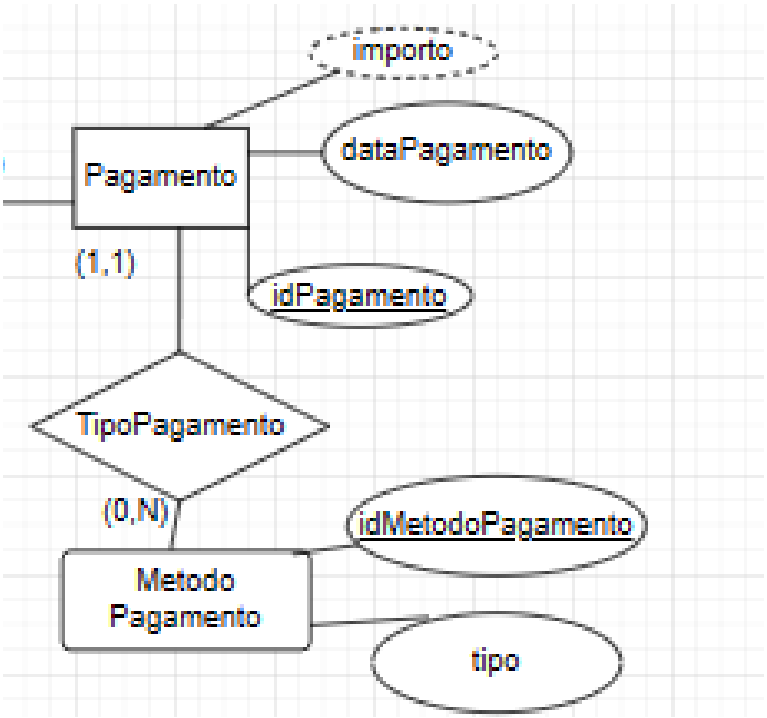
Considerati i dati precedenti sugli accessi: 105 accessi con attributo ridondante e 155 accessi senza attributo ridondante,ho scelto di non eliminare la ridondanza.

Eliminazione delle gerarchie

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

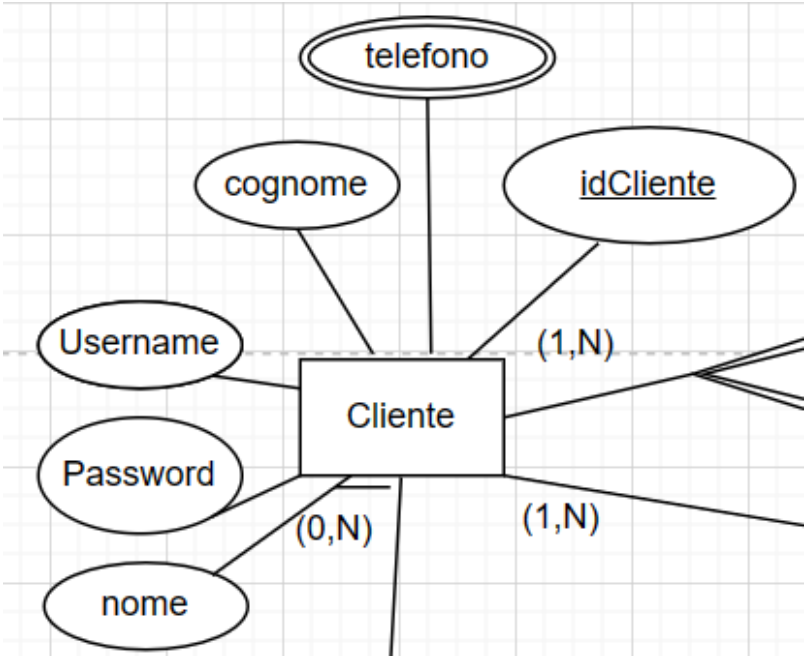


La gerarchia "Pagamento" viene risolta mantenendo l'entità padre "Pagamento" e accorpando le entità figlie in una nuova entità "MetodoPagamento"

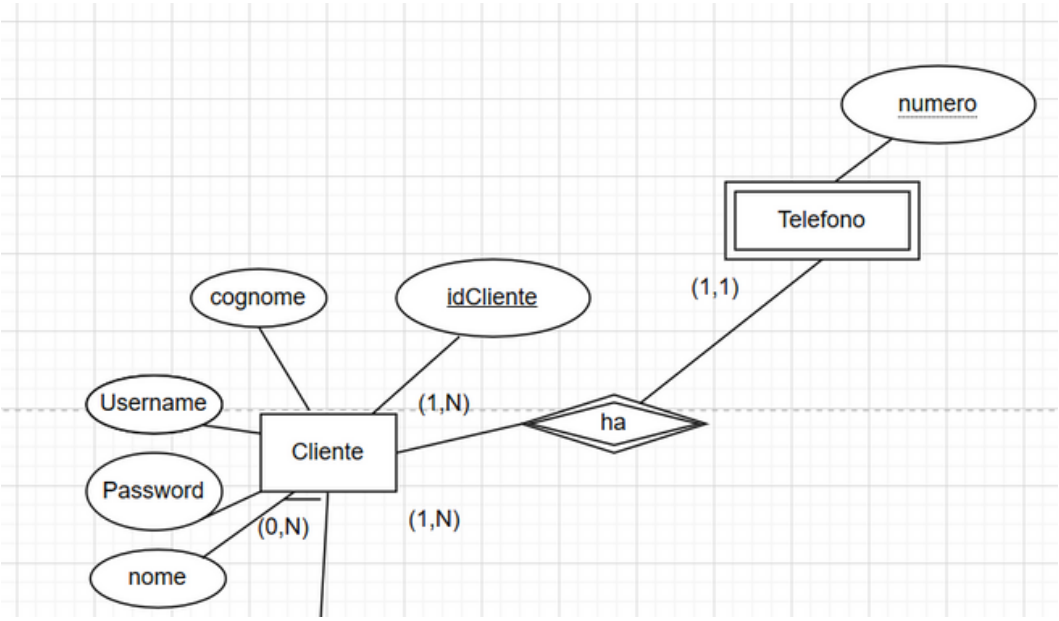


Eliminazione dell’attributo multivalore

Nello schema inizialmente elaborato, compare un attributo multivalore:

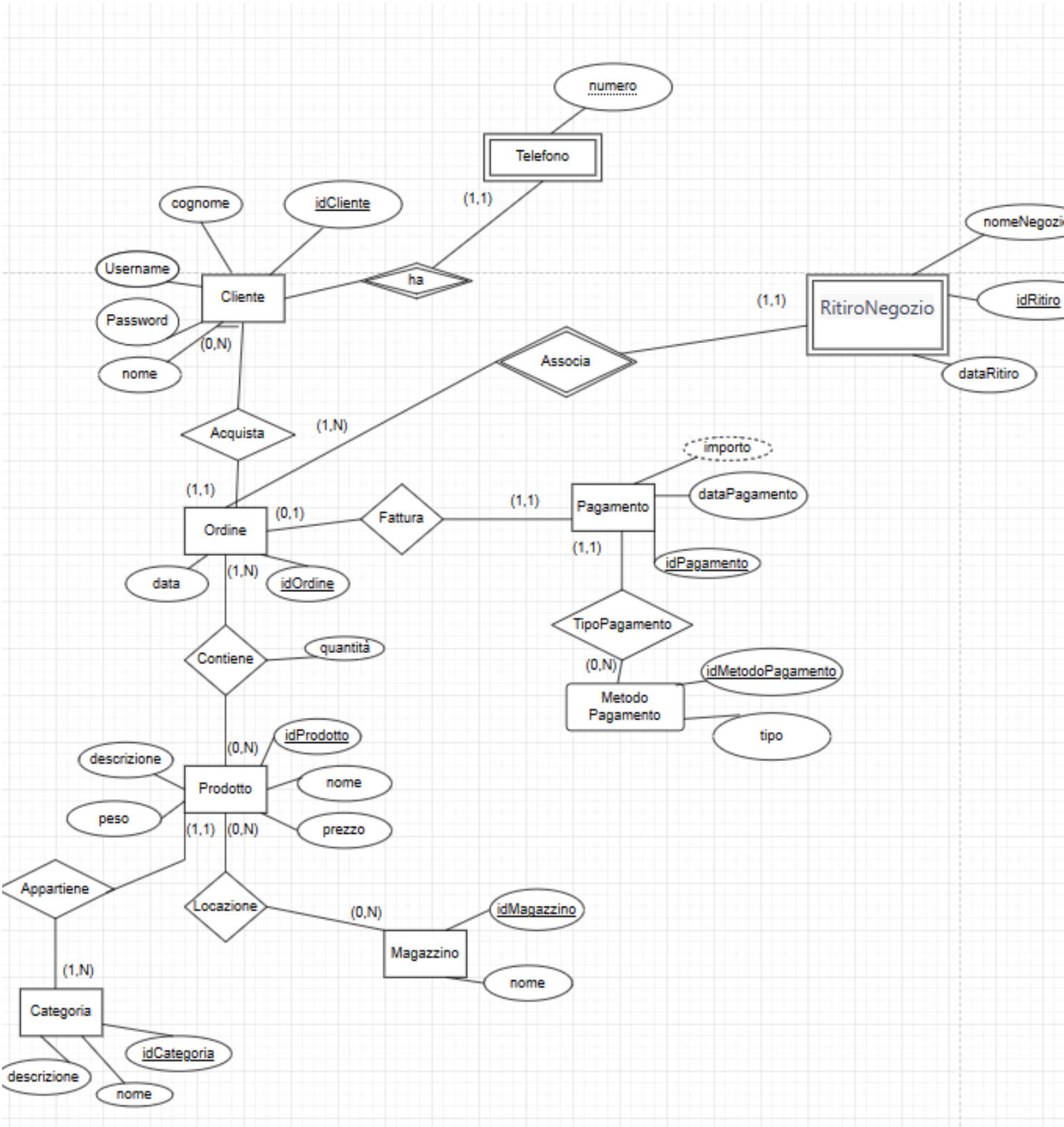


Tale forma di attributo va risolto in maniera differente in fase di progettazione logica.
Si sceglie quindi di definire una nuova entità debole “Telefono”, in relazione con l’entità “Cliente”:



Schema EER ristrutturato

Al termine della fase di ristrutturazione, lo schema EER completo che ne deriva è il seguente:



Schema relazionale

Si procede al mapping della base di dati:

- **Cliente**(idCliente,nome,cognome,username,password)
- **Ordine**(idOrdine,data,Cliente.idCliente↑)

- **RitiroNegozio**(idRitiro,Ordine.idOrdine↑,nomeNegozio,dataRitiro)

- **MetodoPagamento**(idMetodoPagamento,tipo)

- **Pagamento**(idPagamento,dataPagamento,importo,Ordine.idOrdine↑,MetodoPagamento.idMetodoPagamento↑)
- **Telefono**(numero,Cliente.idCliente↑)
- **Categoria**(idCategoria,nome,descrizione)

- **Prodotto**(idProdotto,nome,descrizione,prezzo,peso,Categoria.idCategoria↑)
- **Contiene**(Ordine.idOrdine↑,Prodotto.idProdotto↑,quantità)
- **Magazzino(**idMagazzino,nome)
- **Locazione**(Magazzino.idMagazzino↑,Prodotto.idProdotto↑)

Normalizzazione

Il database si presenta già normalizzato.
È infatti **in prima forma normale** in quanto tutti gli attributi sono atomici dopo la ristrutturazione (è stato infatti eliminato l'attributo multivalore 'telefono' nell'entità Cliente).
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

Realizzazione della base di dati con MySQL

```sql
create schema ecommerce;
use ecommerce;

create table Cliente
(
    idCliente INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome varchar(50) NOT NULL,
    cognome varchar(50) NOT NULL,
    username varchar(50) NOT NULL,
    password varchar(50) NOT NULL
);


create table Ordine
(
    idOrdine INT NOT NULL AUTO_INCREMENT,
    data DATE NOT NULL,
    idCliente INT NOT NULL,
    PRIMARY KEY(idOrdine),
    FOREIGN KEY(idCliente) REFERENCES Cliente(idCliente)
);

```

```sql
create table RitiroNegozio
(
    idRitiro INT NOT NULL AUTO_INCREMENT,
    nomeNegozio VARCHAR(30) NOT NULL,
    dataRitiro DATE NOT NULL,
    idOrdine INT NOT NULL,
    PRIMARY KEY(idRitiro,idOrdine),
    FOREIGN KEY(idOrdine) REFERENCES Ordine(idOrdine)
);

create table MetodoPagamento
(
    idMetodoPagamento INT NOT NULL AUTO_INCREMENT,
    tipo VARCHAR(30) NOT NULL,
    PRIMARY KEY(idMetodoPagamento)
);

create table Pagamento
(
    idPagamento INT NOT NULL AUTO_INCREMENT,
    dataPagamento DATE NOT NULL,
    importo INT NOT NULL,
    idOrdine INT NOT NULL,
    idMetodoPagamento INT NOT NULL,
    PRIMARY KEY(idPagamento),
    FOREIGN KEY(idOrdine) REFERENCES Ordine(idOrdine),
    FOREIGN KEY(idMetodoPagamento) REFERENCES MetodoPagamento(idMetodoPagamento)
);

create table Telefono
(
    numero VARCHAR(20) NOT NULL,
    idCliente INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(numero,idCliente),
    FOREIGN KEY(idCliente) REFERENCES Cliente(idCliente)
);


create table Categoria
(
    idCategoria INT NOT NULL AUTO_INCREMENT,
    descrizione VARCHAR(250) NOT NULL,
    nome VARCHAR(30) NOT NULL,
    PRIMARY KEY(idCategoria)
);
```

```sql
create table Prodotto
(
    idProdotto INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR(30) NOT NULL,
    descrizione VARCHAR(250) NOT NULL,
    prezzo INT NOT NULL,
    peso FLOAT NOT NULL,
    idCategoria INT NOT NULL,
    PRIMARY KEY(idProdotto),
    FOREIGN KEY(idCategoria) REFERENCES Categoria(idCategoria)
);

create table contiene
(
    idOrdine INT NOT NULL,
    idProdotto INT NOT NULL,
    quantità INT NOT NULL,
    PRIMARY KEY(idOrdine,idProdotto),
    FOREIGN KEY(idOrdine) REFERENCES Ordine(idOrdine),
    FOREIGN KEY(idProdotto) REFERENCES Prodotto(idProdotto)
);

create table Magazzino
(
    idMagazzino INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR(30) NOT NULL,
    PRIMARY KEY(idMagazzino)
);

create table Locazione
(
    idMagazzino INT NOT NULL,
    idProdotto INT NOT NULL,
    PRIMARY KEY(idMagazzino,idProdotto),
    FOREIGN KEY(idMagazzino) REFERENCES Magazzino(idMagazzino),
    FOREIGN KEY(idProdotto) REFERENCES Prodotto(idProdotto)
);
```

Implementazione query SQL

Operazione 1:

```
INSERT INTO cliente(idCliente,nome,cognome,username,password)
VALUES(?,?,?,?,?);
```

Operazione 2:

```
INSERT INTO
prodotto(idProdotto,nome,descrizione,prezzo,peso,idCategoria)
VALUES(?,?,?,?,?,?);
```

Operazione 3:

```
SELECT * FROM prodotto;
```

Operazione 4:

```
SELECT p.idProdotto, p.nome AS nomeProdotto, SUM(c.quantità) AS
quantità_venduta
        FROM Prodotto as p
        JOIN Contiene as c ON p.idProdotto = c.idProdotto
        JOIN Ordine as o ON c.idOrdine = o.idOrdine
        WHERE o.data BETWEEN '2021-04-21' AND '2023-09-01'
        GROUP BY p.idProdotto, p.nome
        ORDER BY quantità_venduta DESC
```

Operazione 5:

```
SELECT p.idProdotto, p.nome AS nomeProdotto,
COALESCE(SUM(c.quantità), 0) AS quantità_venduta
 FROM Prodotto as p
 LEFT JOIN Contiene as c ON p.idProdotto = c.idProdotto
 LEFT JOIN Ordine as o ON c.idOrdine = o.idOrdine AND o.data BETWEEN
'2021-04-21' AND '2023-09-01'
 GROUP BY p.idProdotto, p.nome
  ORDER BY quantità_venduta ASC;
```

Operazione 6:

```
SELECT idProdotto, nome, prezzo
FROM Prodotto
ORDER BY prezzo DESC
 LIMIT 1;
```

Operazione 7:

```
SELECT c.idCliente, c.nome
 FROM Cliente c
JOIN Ordine o ON c.idCliente = o.idCliente
 GROUP BY c.idCliente, c.nome
 HAVING COUNT(o.idOrdine) > 1;
```

Operazione 8:

```
SELECT idProdotto, nome, prezzo
 FROM Prodotto
ORDER BY prezzo ASC;
```

Operazione 9:

```
SELECT c.nome, c.cognome, COALESCE(SUM(p.importo), 0) AS
importo_totale
 FROM Cliente c
JOIN Ordine o ON c.idCliente = o.idCliente
 JOIN Pagamento p ON o.idOrdine = p.idOrdine
 WHERE c.idCliente = ?
GROUP BY c.nome, c.cognome;
```

Operazione 10:

```
SELECT idProdotto, nome, prezzo
FROM Prodotto
 WHERE prezzo > 50
```

Operazione 11:

```
SELECT p.idProdotto, p.nome
  FROM Prodotto p
 LEFT JOIN Contiene c ON p.idProdotto = c.idProdotto
 WHERE c.idProdotto IS NULL;
```

Operazione 12:

```
SELECT c.nome, MAX(o.data) AS ultima_data_ordine
 FROM Cliente c
JOIN Ordine o ON c.idCliente = o.idCliente
GROUP BY c.nome;
```

Operazione 13:

```
SELECT cat.idCategoria, cat.nome, COUNT(DISTINCT p.idProdotto) AS
varietà_prodotti
FROM Categoria cat
 JOIN Prodotto p ON cat.idCategoria = p.idCategoria
GROUP BY cat.idCategoria, cat.nome
ORDER BY varietà_prodotti DESC;
```

Operazione 14:

```
SELECT o.idOrdine, o.data
 FROM Ordine o
 JOIN Pagamento p ON o.idOrdine = p.idOrdine
 JOIN MetodoPagamento mp ON p.idMetodoPagamento =
mp.idMetodoPagamento
 WHERE mp.tipo = 'Carta di credito';
```

Operazione 15:

```
SELECT idOrdine, data
  FROM Ordine
  WHERE YEAR(data) = ?;
```

Operazione 16:

```
SELECT rn.nomeNegozio, c.idCliente, c.nome, c.cognome, COUNT(*) AS
NumeroRitiri
FROM RitiroNegozio rn
JOIN Ordine o ON rn.idOrdine = o.idOrdine
JOIN Cliente c ON o.idCliente = c.idCliente
GROUP BY rn.nomeNegozio, c.idCliente, c.nome, c.cognome
ORDER BY NumeroRitiri DESC
LIMIT 1;
```

Test dell’applicazione Java