iCommerce

Progetto di Basi di Dati A.A 2017/18

Mirko Franco Matr. 1138070
Marco Chilese Matr. 1143012

2 Febbraio 2018

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Il progetto modella una basi di dati che rappresenti un possibile sito di e-commerce online, focalizzandosi in particolare sui clienti, gli ordini, le spedizioni e possibili resi.
Le operazioni tipiche in una base di dati così modellata sono la configurazione di un ordine, che consta nella scelta di uno o più prodotti tra i disponibili, la scelta di un metodo di pagamento, di un indirizzo di fatturazione e spedizione per il suddetto ordine. Inoltre, da parte del cliente, deve essere possibile rendere uno o più prodotti tra quelli ordinati.
Le spedizioni dei prodotti sono organizzate internamente in base al magazzino presso cui l'oggetto scelto risulta disponibile. Pertanto un singolo ordine può essere associato a più spedizioni.

Capitolo 1

Raccolta e analisi dei requisiti

Si vuole realizzare una base di dati[1] che contenga e gestisca le informazione relative a un organizzazione che si occupa di commercio elettronico online. In particolare ci si vuole concentrare sui prodotti offerti, sui clienti, gli ordini da questi effettuati e la possibilità di effettuare resi.

Ogni **prodotto** è caratterizzato da:

- un identificativo unico all'interno dell'organizzazione;
- un nome;
- un prezzo;
- una percentuale di sconto.

Ogni prodotto appartiene ad una categoria, è venduto da un certo venditore ed è localizzato in uno o più magazzini.

Ogni **categoria** ha:

- un indentificativo unico;
- un nome.

Ogni **magazzino** ha:

- un codice unico, che lo indentifica;
- un **indirizzo**, caratterizzato da:
  - un codice identificativo;
  - una provincia;
  - una via e un numero civico;
  - un CAP

Ogni indirizzo appartiene a una **nazione**, ed ognuna di esse comprende:

- un codice identificativo unico;
- un nome;
- il nome del continente a cui appartiene.

Ogni **venditore** viene caratterizzato da:

- un identificativo unico;

[1] collezione di dati gestita da un DBMS

- il suo nome.

Ogni **cliente** ha:

- un codice fiscale, che lo identifica;
- una e-mail;
- una password;
- un nome e un cognome;
- il sesso;
- la data di nascita;
- un indirizzo di fatturazione;
- un insieme di indirizzi di consegna;
- un insieme di metodi di pagamento;
- un indicatore che permetta di distinguere un utente attivo da uno che si è disiscritto dal sito.

Ogni cliente può essere un privato o un'azienda ed effettuare ordini di alcuni prodotti offerti. Nel caso del cliente di tipo azienda è di interesse anche la partiva IVA. Inoltre, i metodi di pagamento inseriti possono non necessariamente venir utilizzati per gli ordini: un utente può inserire un insieme di metodi di pagamento che magari non utilizzerà. Ogni metodo di pagamento viene identificato da un codice interno univoco e appartiene a un insieme di clienti. I metodi di pagamento si dividono in contrassegno e carta di credito. Della carta di credito è di interesse conoscere il nome dell'intestatario, il numero della carta e la sua data di scadenza.
Ogni **ordine** è effettuato da un cliente ed è caratterizzato da:

- un codice univoco, che lo identifica;
- la data in cui viene effettuato;
- uno stato;
- un metodo di pagamento;
- un indirizzo di consegna.

Ogni **stato** è caratterizzato da un indentificativo e da un nome autoesplicativo. Ogni ordine può essere spedito in più spedizioni. Di ogni prodotto appartenente all'ordine può essere richiesto un reso dal cliente che lo ha effettuato, entro 14 giorni dalla consegna del prodotto.
Ogni **reso** è caratterizzato da un identificatore univoco, è correlato a un ordine e a un prodotto appertenente all'ordine a cui è riferito ed è di interesse conoscerne la data di richiesta. Il reso si considera possibile se rispetta il vincolo descritto sopra.
Ogni **spedizione** può essere internazionale o nazionale ed è composta da:

- l'ordine a cui è riferita;
- un insieme di prodoti;
- un indirizzo di consegna;
- una data di consegna.

Le operazioni fondamentali che si prevede possano essere effettuate sono la registrazione - e quindi l'inserimento nella base di dati - di un cliente, l'inserimento di prodotti e l'ordine di essi da parte di un cliente. Un cliente può inoltre richiedere un reso di un prodotto sotto le condizioni sopra citate. La base di dati mira a gestire in modo coerente le operazioni di ordine e reso che il cliente può effettuare. È prevista inoltre una piccola gestione dello stato dell'ordine, delle spedizioni e dei metodi di pagamento.

Capitolo 2

Progettazione concettuale

La progettazione concettuale di una base di dati consiste nella rappresentazione delle specifiche informali della realtà di interesse in una modalità formale, ma indipendente dai criteri di rappresentazione utilizzati a livello logico. Mostriamo di seguito la rappresentazione con un modello Entity-Relationship[1] della realtà descritta nel precedente capitolo. Successivamente riportiamo la documentazione allegata al modello Entity-Relationship, ovvero il dizionario dei dati e le regole aziendali. Le regole aziendali rappresentato le proprietà che non sono rappresentabili graficamente nello schema Entity-Relationship.

2.1 Documentazione dello schema Entity-Relationship

[1] modello concettuale di dati che fornisce strumenti per descrivere una realtà di interesse in un modo facile da comprendere e che non dipende dai criteri di organizzazione dei dati nei calcolatori

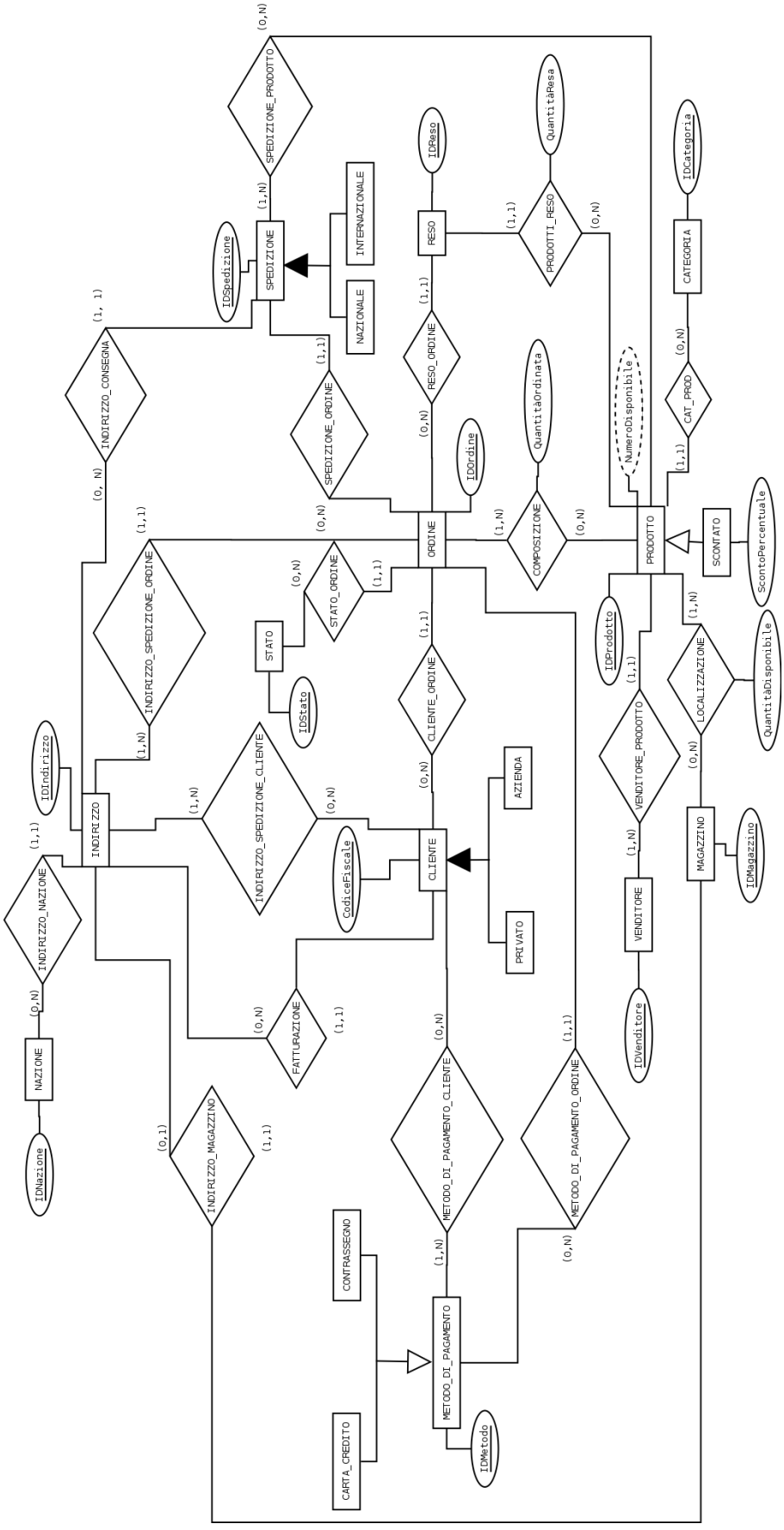

Figura 2.1: *Schema Entity-Relationship*

Entità	Descrizione	Attributi	Identicatore
CATEGORIA	Categorie di prodotti.	IDCategoria Nome	IDCategoria
VENDITORE	Venditori e informazioni ad essi associate.	IDVenditore Nome	IDVenditore
MAGAZZINO	Magazzini dove sono ubicati i prodotti.	IDMagazzino Indirizzo	IDMagazzino
PRODOTO	Prodotti in vendita nel negozio online.	IDProdotto IDVenditore Categoria Nome Prezzo QuantitàDisponibile	IDProdotto
SCONTATO	Prodotti iscontati n vendita nel negozio online.	IDProdotto IDVenditore Categoria Nome Prezzo QuantitàDisponibile ScontoPercentuale	IDProdotto
METODO_DI_PAGAMENTO	Metodi di pagamento dei clienti.	IDMetodo Cliente	IDMetodo
CARTA_DI_CREDITO	Carte di credito dei clienti.	IDMetodo NomeIntestatario NumeroCarta	IDMetodo
CLIENTE	Cliente che si sono iscritti nel negozio online.	CodiceFiscale Nome Cognome DataNascita Sesso IndirizzoFatturazione Mail Password	CodiceFiscale
AZIENDA	Cliente che si sono iscritti nel negozio online.	CodiceFiscale Nome Cognome DataNascita Sesso IndirizzoFatturazione Mail Password PartitaIVA	CodiceFiscale
STATO	Stato dell'ordine.	IDStato Nome	IDStato
ORDINE	Ordini effettuati dai clienti.	IDOrdine Cliente Data MetodoDiPagamento IndirizzoDiSpedizione StatoOrdine	IDOrdine

Entità	Descrizione	Attributi	Identicatore
RESO	Resi richiesti dai clienti.	IDReso Ordine Prodotto Data	IDReso
NAZIONE	Nazioni di spedizione degli ordini, di fatturazione dei clienti, dei magazzini.	IDNazione Nome Continente	IDOrdine
SPEDIZIONE	Spedizioni degli ordini effettuati dai clienti.	IDSpedizione DataConsegna Ordine	IDSpedizione
INDIRIZZO	Indirizzi di spedizione degli ordini, di fatturazione dei clienti, dei magazzini.	IDIndirizzo Via NumeroCivico Provincia CAP Nazione	IDIndirizzo

Tabella 2.1: *Dizionario delle entità dello schema Entity-Relationship*

Relazione	Descrizione	Entità coinvolte	Atributi
LOCALIZZAZIONE	Associa a ogni magazzino i prodotti presenti.	PRODOTTO (1,N) MAGAZZINO(0,N)	QuantitàDisponibile
VENDITORE_PRODOTTO	Associa a ogni prodotto il suo venditore.	PRODOTTO (1,1) VENDITORE(1,N)	
CAT_PROD	Associa a ogni prodotto la sua categoria.	PRODOTTO (1,1) CATEGORIA(0,N)	
COMPOSIZIONE	Associa a ogni ordine i prodotti da cui è composto.	ORDINE (1,N) PRODOTTO (0,N)	QuantitàOrdinata
PRODOTTI_RESO	Associa a ogni reso il prodetto che si deve rendere.	PRODOTTO (0,N) RESO (1,1)	QuantitàResa
RESO_ORDINE	Associa a ogni reso un ordine.	RESO (1,1) ORDINE(0,N)	
SPEDIZIONE_PRODOTTO	Associa a ogni spedizione un insieme di prodotti.	PRODOTTO (0,N) SPEDIZIONE (1,N)	QuantitàSpedita
SPEDIZIONE_ORDINE	Associa ad ogni spedizione un ordine.	SPEDIZIONE (1,1) ORDINE(0,N)	
METODO_DI_PAGAMENTO_ORDINE	Associa a ogni ordine un metodo di pagamento.	ORDINE (1,1) METODO_DI_ PAGAMENTO (0.N)	
METODO_DI_PAGAMENTO_CLIENTE	Associa a ogni cliente un insieme di metodi di pagamento.	CLIENTE (0,N) METODO_DI_ PAGAMENTO (1,N)	
CLIENTE_ORDINE	Associa a ordine il cliente che l'ha effettuato.	ORDINE (1,1) CLIENTE (0,N)	
STATO_ORDINE	Associa a ogni ordine il suo stato.	ORDINE (1,1) STATO (0,N)	
INDIRIZZO_SPEDIZIONE_CLIENTE	Associa a ogni cliente un insieme di indirizzi di spedizione.	CLIENTE (0,N) INDIRIZZO(1,N)	
INDIRIZZO_SPEDIZIONE_ORDINE	Associa a ogni ordine il suo indirizzo di spedizione.	ORDINE (1,1) INDIRIZZO_DI_ SPEDIZIONE (1,N)	
INDIRIZZO_CONSEGNA	Associa a ogni spedizione un indirizzo di consegna.	INDIRIZZO(0,N) SPEDIZIONE(1,1)	

Relazione	Descrizione	Entità coinvolte	Atributi
FATTURAZIONE	Associa a ogni cliente un indirizzo di fatturazione	INDIRIZZO (0,N) CLIENTE (1,1)	
INDIRIZZO_MAGAZZINO	Associa a magazzino il suo indirizzo.	INDIRIZZO (0,1) MAGAZZINO(1,1)	
INDIRIZZO_NAZIONE	Associa a ogni indirizzo la sua nazione di appartenenza.	INDIRIZZO (1,1) NAZIONE (0,N)	

Tabella 2.2: Dizionario delle relazioni dello schema Entity-Relationship

Regole di vincolo	
(RV1)	Il metodo di pagamento di un ordine deve essere un metodo di pagamento del cliente che ha effettuato l'ordine.
(RV2)	L'indirizzo di spedizione di un ordine deve essere un indirizzo del cliente che ha effettuato l'ordine
(RV3)	Un prodotto di cui viene effettuato il reso deve essere un prodotto presente nell'ordine a cui si riferisce il reso.
(RV4)	La quantità di pezzi di un prodotto di cui viene effettuato il reso deve essere uguale o inferiore al numero di pezzi ordinati nell'ordine a cui si riferisce il reso.
(RV5)	La data di spedizione deve essere uguale o successiva alla data in cui è stato effettuato l'ordine.
(RV6)	La data di richiesta di un reso non può essere distante più di 14 giorni dalla data di ricezione della spedizione del prodotto a cui si riferisce il reso.
(RV7)	I prodotti che compongono una spedizione devono essere prodotti presenti nell'ordine a cui la spedizione è riferita.
(RV8)	L'indirizzo di consegna di una spedizione deve essere l'indirizzo di spedizione dell'ordine al quale si riferisce.
(RV9)	La quantità ordinata di un prodotto deve essere minore o uguale alla quantità di quel prodotto presente in magazzino.

Tabella 2.3: *Regole di vincolo*

Regole di derivazione
(RD1) Il numero di pezzi disponibili di un dato prodotto deve essere la somma delle singole disponibilità del prodotto nei magazzini dove è presente.

Tabella 2.4: *Regole di derivazione*

Capitolo 3

Progettazione logica

La progettazione logica di una base di dati consiste nella riorganizzazione e ottimizzazione dello schema E-R in output dalla fase di progettazione concettuale, in modo da renderlo naturalmente traducibile nel modello logico, nel nostro caso il modello relazionale[1]. Essa si divide in una prima fase di **ristrutturazione** dello schema E-R e poi nella vera e propria **traduzione** nel modello logico di riferimento.

3.1 Ristrutturazione

Nel processo di ristrutturazione dello schema E-R, come primo passo si decide di risolvere le generalizzazioni presenti nello schema.
In ordine di rilevanza concettuale:

Generalizzazione cliente privato-azienda: le caratteristiche peculiari delle entità figlie sono state accorpate nell'entità genitore, consentendo quindi valori `NULL` per gli attributi specifici di entrambi. Quindi otteniamo un'entità con le seguenti caratteristiche:

`CLIENTE`(CodiceFiscale, Nome, Cognome*, Mail, Password, DataNascita, Sesso*, isAzienda, PartitaIVA, IndirizzoFatturazione)

dove gli attributi marcati con * possono avere valori nulli, in particolare `"Cognome"` può assumere valori nulli poichè un'azienda non potrà possedere un cognome.

Generalizzazione spedizione nazionale-internazionale: tale generalizzazione è stata ristrutturata semplicemente aggiungendo un attributo di tipo booleano (`isInternazionale`) all'entità genitore: in questo modo è possibile distinguere una spedizione internazionale da una nazionale. Otteniamo quindi un'entità con le seguenti caratteristiche:

`SPEDIZIONE`(IDSpedizione, Ordine, isInternazionale, DataConsegna).

Generalizzazione dei metodi di pagamento: tale generalizzazione permette la distinzione tra i metodi di pagamento offerti dal negozio: contrassegno e carta di credito. Il pagamento mediante carta di credito è caratterizzato dal numero di carta e dal nome dell'intestatario. La ristrutturazione di tale struttura prevede l'accorpamento di entrambe le entità figlie all'entità genitore, con relativi attributi, e l'inserimento di un attributo `"tipo"` che permetta la distinzione tra i vari metodi di pagamento offerti, garantendo l'introduzione futura di nuovi metodi di pagamento. Pertanto, l'entità risultante sarà:

`METODO_DI_PAGAMENTO`(IDMetodo, Tipo, NumeroCarta*, IntestatarioCarta*)

dove gli attributi marcati con * possono avere valori nulli (ad. es. in caso di contrassegno).

Generalizzazione prodotto scontato: la generalizzazione attraverso cui si specificava se un prodotto fosse scontato o meno, viene risolta mediante l'accorpamento dell'entità figlia con l'entità genitore. Introducendo quindi l'attributo `sconto_percentuale` permettendo che sia a `NULL` nel caso in cui non ci sia alcun sconto applicato. Risulterà quindi:

`PRODOTTO`(IDProdotto, Categoria, Nome, Venditore, Prezzo, ScontoPercentuale*, QuantitaDisponibile)

dove gli attributi marcati con * possono avere valori nulli.

[1] modello logico che permette di definire i dati per mezzo del costruttore relazione

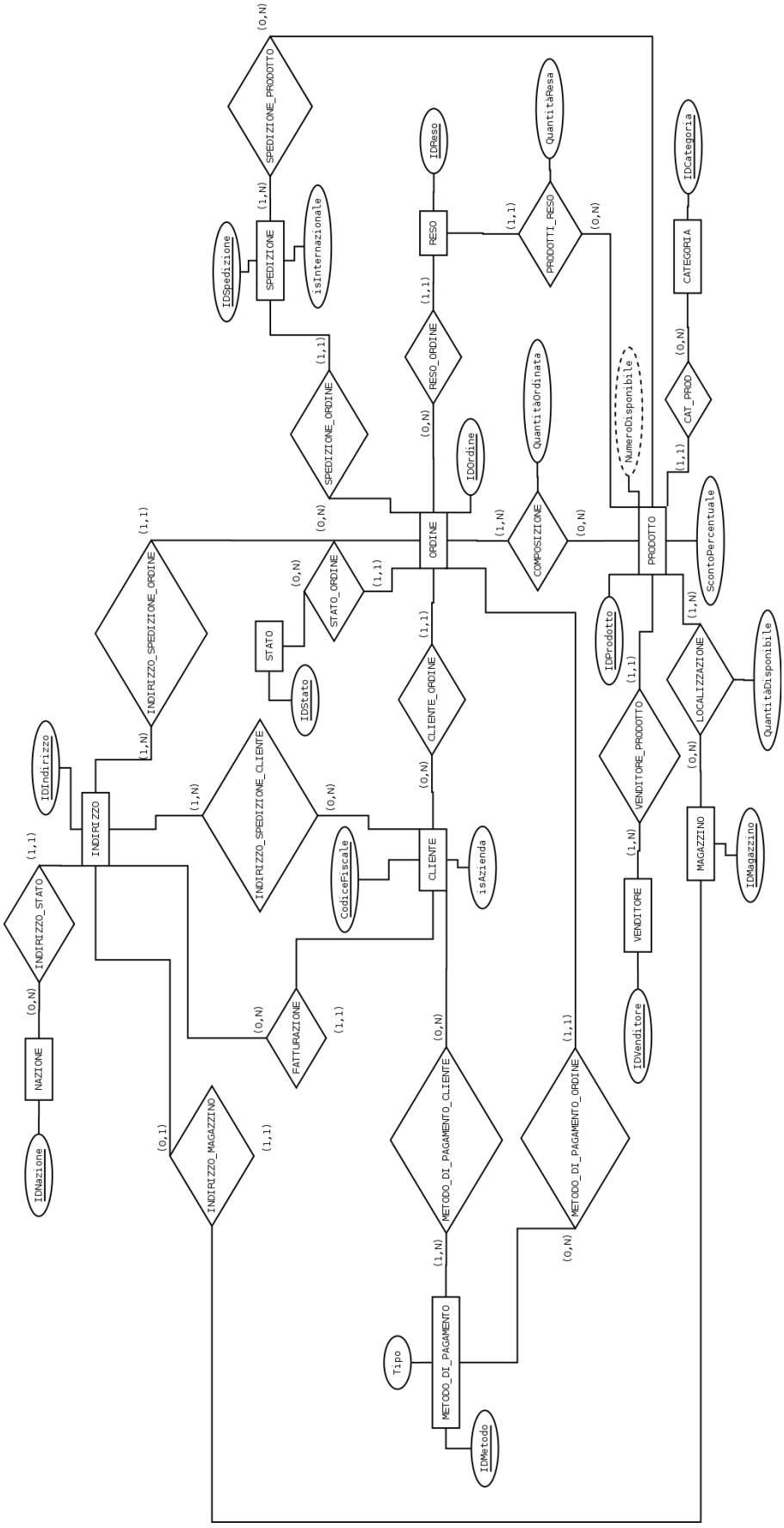

Figura 3.1: *Schema Entity-Relationship ristrutturato*

Nel processo di ristrutturazione è stata inoltre rimossa la relazione `INDIRIZZO_CONSEGNA` che permetteva di correlare una spedizione con il proprio indirizzo di consegna, che è altresì specificato nell'ordine. Rimuovendo tale relazione si evita ridondanza, infatti mediante la relazione `SPEDIZIONE_ORDINE` è possibile ottenere ugualmente l'indirizzo di spedizione.

3.2 Traduzione verso il modello logico

Dopo aver concluso la ristrutturazione dello schema E-R abbiamo proceduto a tradurre le varie entità e relazioni secondo il modello logico. Ecco il risultato:

`CATEGORIA` (IDCategoria, Nome)
PK(IDCategoria)

`VENDITORE` (IDVenditore, Nome)
PK(IDVenditore)

`NAZIONE` (IDNazione, Nome, Continente)
PK(IDNazione)

`INDIRIZZO` (IDIndirizzo, Via, NumeroCivico, CAP, Provincia, Nazione)
PK(IDIndirizzo)
Nazione FK(Nazione(IDNazione))

`MAGAZZINO` (IDMagazzino, Indirizzo)
PK(IDMagazzino)
Indirizzo FK(INDIRIZZO(IDIndirizzo))

`PRODOTTO` (IDProdotto, Categoria, Nome, Venditore, Prezzo, Scontopercentuale*, QuantitàDisponibile)
PK(IDProdotto)
Categoria FK(CATEGORIA(IDCategoria))
Venditore FK(Venditore(IDVenditore))
Altri vincoli:
La quantità disponibile deve essere la somma delle quantità presenti nei vari magazzini del prodotto stesso

`CLIENTE` (CodiceFiscale, Nome, Cognome*, Mail, Password, DataNascita*, Sesso*, isAzienda, PartitaIVA*, IndirizzoFatturazione)
PK(CodiceFiscale)
IndirizzoFatturazione FK(INDIRIZZO(IDIndirizzo))

`METODO_DI_PAGAMENTO` (IDMetodo, Tipo, NumeroCarta*, NomeIntestatario*)
PK(IDMetodo)

`METODO_DI_PAGAMENTO_CLIENTE` (Cliente, MetodoPagamento)
PK(Cliente, MetodoPagamento)
Cliente FK(CLIENTE(CodiceFiscale))
MetodoPagamento FK(METODO_DI_PAGAMENTO(IDMetodo))

`STATO` (IDStato, Nome)
PK(IDStato)

`ORDINE` (IDOrdine, Cliente, Data, MetodoPagamento, IndirizzoSpedizione, StatoOrdine)
PK(IDOrdine)
Cliente FK(CLIENTE(CodiceFiscale))
MetodoPagamento FK(METODO_DI_PAGAMENTO(IDMetodo))
IndirizzoSpedizione FK(INDIRIZZO(IDIndirizzo))
StatoOrdine FK(STATO(IDStato))
Altri vincoli:
Il metodo di pagamento dell'ordine deve essere un metodo di pagamento appartenente al cliente.
L'indirizzo di spedizione dell'ordine deve essere un indirizzo di spedizione del cliente che ha effettuato l'ordine

`SPEDIZIONE` (IDSpedizione, Ordine, isInternazionale, DataConsegna)
PK(IDSpedizione)
Ordine FK(ORDINE(IDOrdine))
Altri vincoli:
La data di spedizione deve essere uguale o successiva alla data in cui è stato effettuato l'ordine a cui la spedizione si riferisce

`RESO` (IDReso, Prodotto, Ordine, Data, QuantitàResa)
PK(IDReso)
Prodotto FK(PRODOTTO(IDProdotto))
Ordine FK(ORDINE(IDOrdine))
Altri vincoli:
Il prodotto di cui viene effettuato il reso deve essere un prodotto presente nell'ordine a cui il reso si riferisce
La quantità di prodotto resa deve essere uguale o inferiore alla quantità di prodotto ordinata
La data del reso non può essere distante più di 14 giorni dalla data di cinsegna del prodotto che si vuole rendere.

`INDIRIZZO_SPEDIZIONE_CLIENTE` (Indirizzo, Cliente)
PK(Indirizzo, Cliente)
Indirizzo FK(INDIRIZZO(IDIndirizzo))
Cliente FK(CLIENTE(CodiceFiscale))

`SPEDIZIONE_PRODOTTO` (Spedizione, Prodotto, QuantitàSpedita)
PK(Spedizione, Prodotto)
Spedizione FK(SPEDIZIONE(IDSpedizione))
Prodotto FK(PRODOTTO(IDProdotto))
Altri vincoli:
I prodotti di una spedizione devono essere prodotti presenti nell'ordine a cui la spedizione si riferisce.

`COMPOSIZIONE` (Ordine, Prodotto, QuantitàDisponibile)
PK(Ordine, Prodotto)
Ordine FK(ORDINE(IDOrdine))
Prodotto FK(PRODOTTO(IDProdotto))

`LOCALIZZAZIONE` (Magazzino, Prodotto, QuantitàDisponibile)
PK(Magazzino, Prodotto)
Magazzino FK(MAGAZZINO(IDMagazzino))
Prodotto FK(PRODOTTO(IDProdotto))

Capitolo 4

Implementazione

In questo capitolo verrà presentata l'implementazione effettiva della base di dati in un DBMS reale. Per questo progetto è stato utilizzato un server di prova locale Apache 2.4.29 equipaggiato con MariaDB 10.1.28. Il codice della creazione dello schema viene ommesso. Vengono invece riportati triggers, stored procedures, funzioni e query codificate attraverso il linguaggio PL/SQL.

4.1 Stored Procedure

4.1.1 SP1

La seguente procedura impone la quantià disponibile di un prodotto uguale alla somma delle quantità di quel prodotto nei vari magazzini.

```
CREATE PROCEDURE aggiornaDisponibilita(IN codiceProdotto varchar(10))
        UPDATE PRODOTTO
        SET QuantitaDisponibile = (
                SELECT SUM(QuantitaDisponibile)
                FROM LOCALIZZAZIONE
                WHERE Prodotto = codiceProdotto
                GROUP BY codiceProdotto)
        WHERE IDProdotto = codiceProdotto
```

4.1.2 SP2

La seguente procedura aggiunge pezzi di un certo prodotto in un certo magazzino.
Si ritengono, per precondizione, validi i dati in input.

```
CREATE PROCEDURE aggiungPezziMagazzino(IN codiceMagazzino varchar(20),
        IN codiceProdotto varchar(10), IN daAggiungere integer)
BEGIN
        START TRANSACTION;
                UPDATE LOCALIZZAZIONE
                SET QuantitaDisponibile = QuantitaDisponibile + daMuovere
                WHERE Magazzino = codiceMagazzino AND
                  Prodotto = codiceProdotto;

        UPDATE PRODOTTO
        SET QuantitaDisponibile = QuantitaDisponibile + daMuovere
        WHERE IDProdotto = codiceProdotto;
COMMIT WORK;
END;
```

4.1.3 SP3

Questa procedura produce la lista per un dato cliente c in input la lista degli ordini effettuati ma non ancora evasi.

```
CREATE PROCEDURE ordiniNonEvasi (IN cliente varchar(11))
BEGIN
        SELECT O.IDOrdine
        FROM ORDINE AS O
        WHERE O.Cliente = cliente AND O.StatoOrdine <> 5 ;
END
```

4.1.4 SP4

La seguente funzione stampa tutti i prodotti che un cliente c ha ordinato

```
CREATE PROCEDURE stampaProdottiOrdinati(IN cliente varchar(11))
        SELECT O.IDOrdine, O.Cliente, C.Nome, C.Cognome, O.Data, P.Nome
        FROM ORDINE AS O
        INNER JOIN CLIENTE AS C ON O.Cliente = C.CodiceFiscale
        INNER JOIN PRODOTO AS P ON O.Prodotto = P.IDProdotto
        WHERE O.Cliente = cliente
```

4.2 Function

4.2.1 F1

Questa funzione ritorna il numero di ordini effettuati dal cliente c negli ultimi n mesi.
Tale funzione trova applicazione nella Query `Q2`.

```
CREATE FUNCTION numeroOrdini(cliente varchar(11), numeroMesi integer)
RETURNS INTEGER
BEGIN
        DECLARE numero INTEGER;

        SELECT COUNT(*) INTO numero
        FROM ORDINE AS O
        WHERE O.Cliente = cliente AND DATADIFF(CURDATE(), O.Data) < numeroMesi * 30;

        RETURN numero;
END;
```

4.2.2 F2

La seguente funzione, dato un codice prodotto p ritorna il suo prezzo.

```
CREATE FUNCTION prezzoProdotto(codiceProdotto varchar(10))
RETURNS FLOAT(2)
BEGIN
        DECLARE scontoPercentuale INTEGER;
        DECLARE prezzo FLOAT(2);

        SELECT P.ScontoPercentuale INTO scontoPercentuale
        FROM PRODOTTO AS P
        WHERE P.IDProdotto = codiceProdotto ;

        SELECT P.Prezzo INTO prezzo
        FROM PRODOTTO AS P
```

```
        WHERE P.IDProdotto =  codiceProdotto;

        RETURN prezzo - (scontoPercentuale/100 * prezzo);
END;
```

4.2.3 F3

La seguente funzione, dato un cliente c, ritorna il totale speso all'interno del sito.

```
CREATE FUNCTION totaleSpesoCliente(cliente  varchar(11))
RETURNS FLOAT(2)
BEGIN
        DECLARE totale FLOAT(2);
        DECLARE numeroOrdini INTEGER;

        SELECT SUM(CM.QuantitaOrdinata * prezzoProdotto(CM.Prodotto)) INTO totale
        FROM ORDINE AS O
                INNER JOIN CLIENTE AS C ON O.Cliente = C.CodiceFiscale
                INNER JOIN COMPOSIZIONE AS CM ON CM.Ordine = O.IDOrdine
        WHERE C.CodiceFiscale = cliente ;

        SELECT COUNT(*) INTO numeroOrdini
        FROM ORDINE AS O
        WHERE O.Cliente = cliente;

        IF numeroOrdini = 0 THEN
                RETURN 0;
        END IF;

        RETURN totale;
END;
```

4.3 Triggers

4.3.1 T1

Il seguente trigger si impone di verificare prima dell'inserimento di un nuovo prodotto in `PRODOTTO`, che `quantitaDisponibile` sia maggiore di 0. Nel caso non lo fosse, invia un segnale d'errore che informa dell'accaduto.

```
CREATE TRIGGER quantitaNonNegativa
BEFORE INSERT ON PRODOTTO
FOR EACH ROW
BEGIN
IF new.QuantitaDisponibile < 0 THEN
        SIGNAL SQLSTATE VALUE '45000'
                SET MESSAGE_TEXT = '[TABLE: PRODOTTO] -
                        QuantitaDisponibile non valida';
END IF;
SET NEW.QuantitaDisponibile = 0;
END;
```

4.3.2 T2

Il seguente trigger si impone di verificare prima dell'inserimento di un prodotto in magazzino, che la quantità inserita sia maggiore di 0. Nel caso in cui non lo fosse, informa dell'accaduto.

```
CREATE TRIGGER verficaLocalizzazione
BEFORE INSERT ON LOCALIZZAZIONE
FOR EACH ROW
BEGIN
IF new.QuantitaDisponibile < 0 THEN
        SIGNAL SQLSTATE VALUE '45000'
                SET MESSAGE_TEXT = '[TABLE: LOCALIZZAZIONE] -
                        QuantitaDisponibile non valida';
END IF;
END;
```

4.3.3 T3

Il seguente trigger aggiorna la quantità disponibile di un prodotto quando viene inserito in uno dei magazzini. Tale aggiornamento viene effettuato mediante una `CALL` alla procedura `SP1 - aggiornaDisponibilita`.

```
CREATE TRIGGER aggiornaDisponibilita
AFTER INSERT ON LOCALIZZAZIONE
FOR EACH ROW
BEGIN
        CALL aggiornaDisponibilita(NEW.Prodotto);
END;
```

4.3.4 T4

Il seguente trigger verifica che l'indirizzo di spedizione dell'ordine e il metodo di pagamento appartengano al cliente che ha effettuato l'ordine.

```
CREATE TRIGGER verificaOrdine
BEFORE INSERT ON ORDINE
FOR EACH ROW
DECLARE
        numeroIndirizzi INTEGER;
        numeroMetodi INTEGER;
BEGIN
SELECT COUNT(*) INTO numeroIndirizzi
FROM INDIRIZZO_SPEDIZIONE_CLIENTE AS IC
WHERE IC.Cliente = NEW.Cliente AND
        IC.Indirizzo = NEW.IndirizzoSpedizione;

IF numeroIndirizzi = 0
        SIGNAL SQL STATE '45000'
                SET MESSAGE_TEXT = '[TABLE : ORDINI] -
                        L'indirizzo di spedizione deve
                        essere un indirizzo del cliente';

SELECT COUNT(*) INTO numeroMetodi
FROM METODO_DI_PAGAMENTO_CLIENTE AS MC
WHERE MC.Cliente = NEW.Cliente AND
        MC.MetodoPagamento = NEW.MetodoPagamento;

IF numeroMetodi = 0
        SIGNAL SQL STATE '45000'
        SET MESSAGE_TEXT = '[TABLE : ORDINI] -
                Il metodo di pagamento dell'ordine deve
                appartenere al cliente ';
```

```
END;
```

4.3.5 T5

Tale trigger verifica che la data in cui si effettua il reso rispetti le regole anziendali, ossia entro 14 giorni dalla di consegna. Inoltre la quantità di un prodotto da rendere deve essere al massimo pari a quella ordinata, ed ovviamente il prodotto reso deve appartenere all'ordine a cui si riferisce.

```
CREATE TRIGGER verificaReso
BEFORE INSERT ON RESO
FOR EACH ROW
BEGIN

        /* Il prodotto deve appartenere all'ordine a cui il reso si riferisce */
        DECLARE numeroProdotti INTEGER;

        SELECT COUNT(*) INTO numeroProdotti
        FROM COMPOSIZIONE AS C
        WHERE C.Prodotto = NEW.Prodotto AND C.Ordine = NEW.Ordine;

        IF numeroProdotti = 0 THEN
                SIGNAL SQLSTATE VALUE '45000'
                        SET MESSAGE_TEXT = '[TABLE : RESO] - Il prodotto deve
                        appartenere all'ordine a cui il reso si riferisce';
        END IF;

        /*La quantita' di prodotto resa deve essere minore o uguale a quella ordinata */
        IF NEW.QuantitaResa > (
                SELECT C.QuantitaOrdinata
                FROM COMPOSIZIONE
                WHERE NEW.Ordine = C.Ordine AND NEW.Prodotto = C.Prodotto)
        THEN
                SIGNAL SQLSTATE VALUE '45000'
                        SET MESSAGE_TEXT = '[TABLE : RESO] - La
                  quantita' resa deve essere minore o
                          uguale alla quantita' ordinata'';
        END IF;

        /* Il reso di un prodotto puo' essere richiesto al massimo dopo 14 giorni
        dalla data di consegna */
        IF DATEDIFF(DataReso, (
                SELECT S.DataConsegna
                FROM SPEDIZIONE AS S
                INNER JOIN SPEDIZIONE_PRODOTTO AS SP ON
                  S.IDSpedizione = SP.Spedizione
                WHERE SP.Prodotto = NEW.Prodotto )) > 14
        THEN
                SIGNAL SQLSTATE VALUE '45000'
                        SET MESSAGE_TEXT = '[TABLE : RESO] - Il reso puo' essere
                          richiesto al massimo dopo 14 giorni';
        END IF;
END
```

4.3.6 T6

Tale trigger, all'inserimento di un nuovo cliente nella basi di dati, associa a tale cliente un metodo di pagamento di default: il contrassegno.

```
CREATE TRIGGER `contrassegnoDefault`
AFTER INSERT ON `CLIENTE`
FOR EACH ROW
BEGIN
        INSERT INTO METODO_DI_PAGAMENTO_CLIENTE
        VALUES(new.CodiceFiscale, '1');
END;
```

4.4 View

4.4.1 V1

La seguente vista rappresenta il numero di ordini effettuati da ciascun cliente.

```
CREATE VIEW numeroOrdini AS
        SELECT C.CodiceFiscale, COUNT(*) AS NumeroOrdini
        FROM CLIENTE AS C
                INNER JOIN ORDINE AS O ON C.CodiceFiscale = O.Cliente
                GROUP BY C.CodiceFiscale;
```

4.5 Queries

4.5.1 Q1

Questa query produce una lista dei prodotti disponibili con la relativa disponibilità per ogni magazzino.

```
SELECT L.Magazzino, L.Prodotto AS CodiceProdotto, P.Nome, L.QuantitaDisponibile
FROM LOCALIZZAZIONE AS L
INNER JOIN PRODOTTO AS P ON L.Prodotto = P.IDProdotto
ORDER BY L.Magazzino;
```

4.5.2 Q2

Questa query produce il numero di ordini effettuati negli ultimi sei mesi e nell'ultimo anno da ciascun cliente attivo.
Si utilizza la funzione `F1` precedentemente definita.

```
SELECT C.CodiceFiscale, C.Nome, C.Cognome, C.PartitaIVA,
   numeroOrdini(C.CodiceFiscale, 6) AS NumeroOrdiniSeiMesi,
   numeroOrdini(C.CodiceFiscale, 12) AS NumeroOrdiniUltimoAnno
FROM CLIENTE AS C
WHERE C.isActive = true;
```

4.5.3 Q3

La seguente query stampa tutti i prodotti (e gli ordini di appartenenza) ordinati da un certo cliente.
Avendo la necessità di rendere la query parametrica viene implementata come stored procedure (descritta in Stored Procedure-SP4).

```
call stampaProdottiOrdinati("CODFISC0001");
```

Magazzino	CodiceProdotto	Nome	QuantitaDisponibile
Mag1	COD1	Nome Prodotto1	7
Mag1	COD14	Nome Prodotto14	7
Mag1	COD3	Nome Prodotto3	7
Mag1	COD8	Nome Prodotto8	7
Mag2	COD13	Nome Prodotto13	5
Mag2	COD2	Nome Prodotto2	5
Mag2	COD4	Nome Prodotto4	5
Mag2	COD8	Nome Prodotto8	5
Mag2	COD9	Nome Prodotto9	5
Mag3	COD10	Nome Prodotto10	19
Mag3	COD12	Nome Prodotto12	19
Mag3	COD3	Nome Prodotto3	19
Mag3	COD5	Nome Prodotto5	19
Mag3	COD9	Nome Prodotto9	19
Mag4	COD10	Nome Prodotto10	12
Mag4	COD11	Nome Prodotto11	12
Mag4	COD15	Nome Prodotto15	12
Mag4	COD4	Nome Prodotto4	12
Mag4	COD6	Nome Prodotto6	12
Mag5	COD10	Nome Prodotto10	1
Mag5	COD11	Nome Prodotto11	1
Mag5	COD12	Nome Prodotto12	1
Mag5	COD5	Nome Prodotto5	1
Mag5	COD7	Nome Prodotto7	1
Mag6	COD12	Nome Prodotto12	2
Mag6	COD13	Nome Prodotto13	2
Mag6	COD6	Nome Prodotto6	2
Mag6	COD8	Nome Prodotto8	2
Mag6	COD9	Nome Prodotto9	2
Mag7	COD13	Nome Prodotto13	3
Mag7	COD14	Nome Prodotto14	3
Mag7	COD7	Nome Prodotto7	3
Mag7	COD8	Nome Prodotto8	3
Mag7	COD9	Nome Prodotto9	3

Figura 4.1: Esempio di output della Query 1

CodiceFiscale	Nome	Cognome	PartitaIVA	NumeroOrdiniSeiMesi	NumeroOrdiniUltimoAnno
CODFISC0001	Mario	Rossi	NULL	3	3
CODFISC0002	Antonio	Verdi	NULL	0	0
CODFISC0003	Giulia	Milano	NULL	0	0
CODFISC10	Nome10	Cognome10	NULL	0	0
CODFISC11	Nome11	Cognome11	NULL	0	0
CODFISC12	Nome12	Cognome12	NULL	0	0
CODFISC13	Nome13	Cognome13	NULL	0	0
CODFISC14	Nome14	Cognome14	NULL	0	0
CODFISC15	NomeAzienda15	NULL	PIVA15	0	0
CODFISC16	NomeAzienda16	NULL	PIVA16	0	0
CODFISC17	NomeAzienda17	NULL	PIVA17	0	0
CODFISC18	NomeAzienda18	NULL	PIVA18	0	0
CODFISC19	NomeAzienda19	NULL	PIVA19	0	0
CODFISC4	Nome4	Cognome4	NULL	2	2
CODFISC5	Nome5	Cognome5	NULL	1	1
CODFISC6	Nome6	Cognome6	NULL	1	1
CODFISC7	Nome7	Cognome7	NULL	0	0
CODFISC8	Nome8	Cognome8	NULL	0	0
CODFISC9	Nome9	Cognome9	NULL	0	0

Figura 4.2: Esempio di output della Query 2

IDOrdine	Cliente	Nome	Cognome	Data	Nome
1	CODFISC0001	Mario	Rossi	2018-01-15	Nome Prodotto1
1	CODFISC0001	Mario	Rossi	2018-01-15	Nome Prodotto12
1	CODFISC0001	Mario	Rossi	2018-01-15	Nome Prodotto3
1	CODFISC0001	Mario	Rossi	2018-01-15	Nome Prodotto9

Figura 4.3: Esempio di output della Query 3

4.5.4 Q4

La sequente query ritorna l'elenco dei codici dei prodotti che non sono mai stati ordinati.

```
SELECT P.IDProdotto
FROM PRODOTTO AS P
WHERE P.IDProdotto <> ALL (
        SELECT C.Prodotto
        FROM COMPOSIZIONE AS C)
```

IDProdotto
AAPL1
COD1
AAPL2
COD9
MS1
COD8
COD4
COD16
COD11
COD5
COD7
COD15
COD3
COD12
COD13
COD14
COD6
COD10
COD2

Figura 4.4: Esempio di output della Query 4

4.5.5 Q5

La sequente query ritorna la lista dei cliente che sono di tipo azienda.

```
SELECT C.PartitaIVA, C.Nome, C.IndirizzoFatturazione
FROM CLIENTE AS C
WHERE C.PartitaIVA IS NOT NULL
```

PartitaIVA	Nome	IndirizzoFatturazione
PIVA15	NomeAzienda15	4
PIVA16	NomeAzienda16	2
PIVA17	NomeAzienda17	6
PIVA18	NomeAzienda18	8
PIVA19	NomeAzienda19	1

Figura 4.5: Esempio di output della Query 5

4.5.6 Q6

La sequente query ritorna la lista degli ordini di un cliente non ancora evasi.
La query è resa parametrica mediante una stored procedure (SP3).

```
call ordiniNonEvasi("CODFISC0001");
```

```
+-----------+
| IDOrdine  |
+-----------+
|         1 |
+-----------+
```

Figura 4.6: Esempio di output della Query 6

4.5.7 Q7

La sequente query ritorna il cliente che ha effettuato il maggior numero di ordini.
Si utilizza la vista V1 precedentemente descritta.

```
SELECT NO.CodiceFiscale, NO.NumeroOrdini
FROM   numeroOrdini AS NO
WHERE NO.NumeroOrdini >= ALL(
          SELECT NO.NumeroOrdini
          FROM numeroOrdini as NO)
```

Esempio di risultato:

```
+---------------+--------------+
| CodiceFiscale | NumeroOrdini |
+---------------+--------------+
| CODFISC0001   |            3 |
+---------------+--------------+
```

Figura 4.7: Esempio di output della Query 7

Capitolo 5

Note finali

Durante la progettazione e l'implementazione di questo progetto varie volte è sorto il problema di definire il confine tra le procedure di competenza della base di dati e quelle di competenza dell'applicazione[1]. In particolare, riteniamo che il soddisfacimento della regola di vincolo (`RV9`) sia di competenza dell'applicazione in quanto la gestione, a livello di base di dati, fa nascere il rischio di inserimento di ordini vuoti che non devono essere permessi.
Come sviluppi futuri, per ovviare questo problema, riteniamo interessante lo sviluppo di procedure applicative eventualmente supportate da procedure e/o funzioni interne alla base di dati che impediscano l'inserimento di ordini vuoti e che controllino il soddisfacimento di (`RV9`).

È degno di nota il fatto che il ciclo presente nello schema ER (tra le entità `METODO_DI_PAGAMENTO`, `CLIENTE` e `ORDINE`) non generi ridondanza, poiché un metodo di pagamento può appartenere a più clienti. Analogamente per il ciclo `CLIENTE`, `INDIRIZZO` e `ORDINE`.

[1] intesa sia come utente che come vero e proprio softwre applicativo che usa la base di dati