PROGETTO BASI DI DATI

Sito E-Commerce - Nuova Autoricambi srl

Manfrin Fabio - Palasgo Dario - Vettor Federica

2019/2020

Prof. Giorgio Maria Di Nunzio

SOMMARIO

1. ANALISI DEI REQUISITI

1.1 INTRODUZIONE

Si vuole realizzare un database per la gestione del sito E-Commerce dell'azienda Nuova Autoricambi Srl situata a Venezia. L'obiettivo del database è quello di gestire gli ordini effettuati da clienti registrati nel sito. I clienti possono acquistare prodotti da vari cataloghi, pubblicati dai fornitori annualmente, e procedere con l'eventuale reso di prodotti difettosi. Il cliente può scegliere se far spedire l'ordine ad un preciso indirizzo di spedizione o se andarlo a ritirare in negozio; tuttavia deve comunque specificare l'indirizzo di fatturazione per le transazioni effettuate.
I pagamenti non sono gestiti perchè vengono saldati in modo automatico dal conto del cliente ogni fine mese.

1.2 REQUISITI STRUTTURATI

CLIENTE:
Ogni cliente ha un nome, un cognome, un indirizzo e-mail, il codice IBAN del conto sul quale desidera effettuare il saldo degli ordini e delle credenziali per effettuare l'accesso al sito web; tali credenziali sono un codice identificativo e una password (Psw) assegnati automaticamente. Ogni cliente ha esattamente un solo indirizzo di fatturazione e può fare degli ordini.

INDIRIZZO:
L'indirizzo è composto e identificato dalla città, dalla via e dal numero civico. Gli indirizzi possono essere sia di fatturazione che di spedizione.

CITTÀ:
Le città sono identificate dal loro nome. È necessario poter analizzare gli acquisti in relazione alle città di spedizione degli ordini, per poter migliorare il servizio.

ORDINE:
Un ordine è identificato da un ID e ha una data di effettuazione. Un ordine è eseguito da un solo cliente e può contenere un solo indirizzo di spedizione; nel caso quest'ultimo non sia presente, il cliente dovrà andare a ritirare il suo ordine al negozio. Ogni ordine è composto da almeno un prodotto acquistato.

PRODOTTO:
Ogni prodotto è identificato da un codice a barre, ha una descrizione e il suo prezzo attuale. Ogni prodotto appartiene ad una sola categoria e deve appartenere a un solo catalogo.

CATEGORIA:
Una categoria è identificata da un nome.

FORNITORE:
Ogni fornitore è identificato dalla sua partita IVA e ha una ragione sociale.

CATALOGO:
Un catalogo è identificato dall'anno nel quale è stato pubblicato e dal singolo fornitore che lo ha pubblicato. Ogni catalogo, che ha anche il suo nome, deve contenere almeno un prodotto.

PRODOTTO ACQUISTATO:
Ogni specifico prodotto acquistato, identificato da un numero progressivo e dal prodotto di cui fa parte, ha un suo prezzo di acquisto. Ogni prodotto acquistato deve appartenere a un ordine.

RESO:
Se un prodotto acquistato risulta difettoso, è possibile eseguire un reso. Il reso è identificato da un ID, contiene una motivazione e la data di effettuazione della procedura. Il reso inoltre può essere già stato saldato, tramite versamento sull'IBAN del cliente, oppure può essere ancora in sospeso. Ogni reso deve essere riferito a uno specifico prodotto acquistato in un ordine effettuato.

1.3 OPERAZIONI

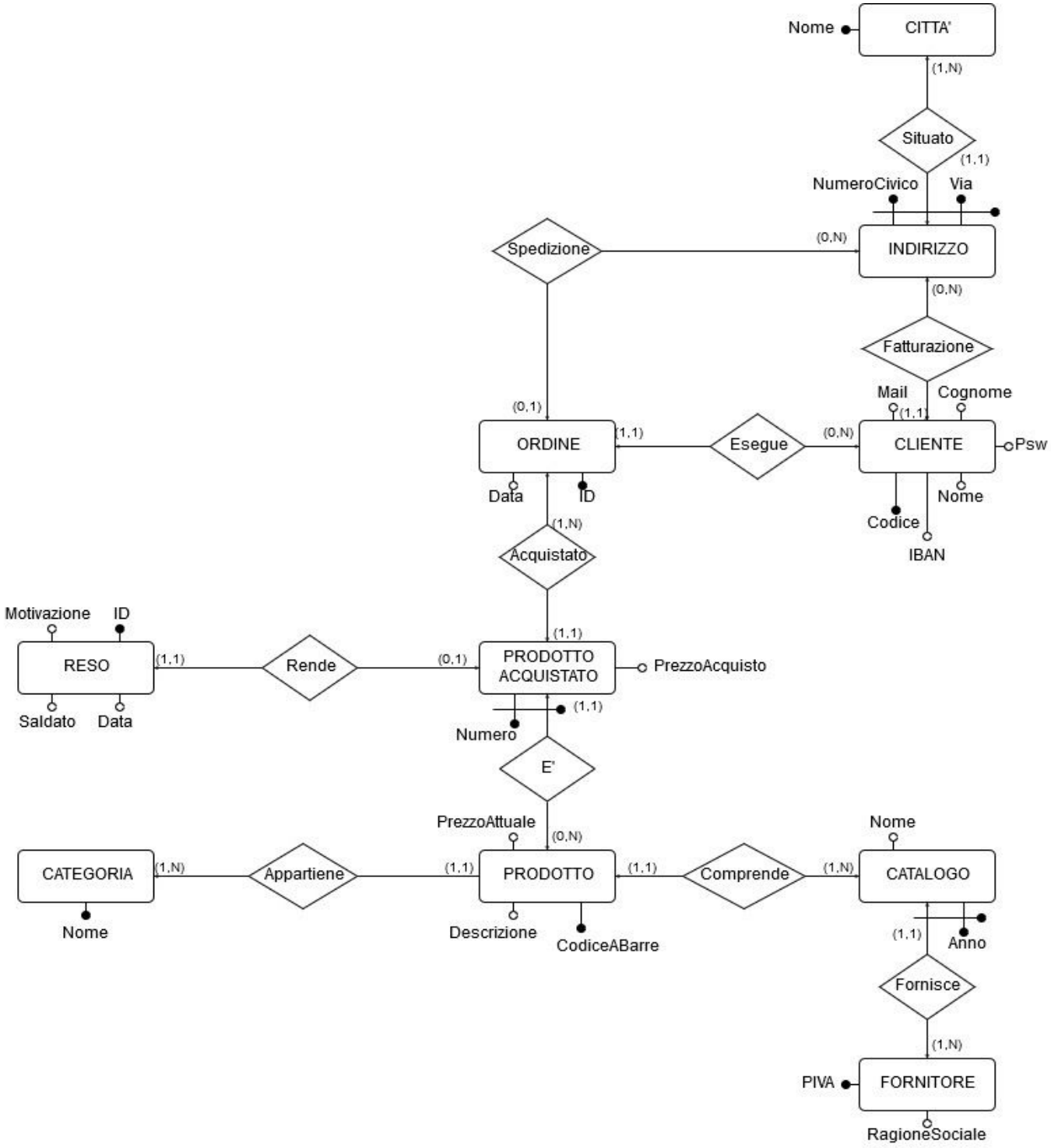
Con il fine di analizzare l'andamento delle vendite, di analizzare le attività sul sito di e-commerce e di aggiornare i dati memorizzati, proponiamo alcune operazioni.

OPERAZIONE	**TIPOLOGIA**	**FREQUENZA**
Calcolare il fatturato annuo per ogni cliente	Interrogazione	1/Anno
Trovare la lista degli articoli più acquistati	Interrogazione	1/Settimana
Trovare la lista degli lista articoli meno acquistati	Interrogazione	1/Settimana
Calcolare il numero di clienti inattivi	Interrogazione	1/Mese
Calcolare il numero di spedizioni per ogni città	Interrogazione	1/Mese
Trovare il fornitore del prodotto reso più volte	Interrogazione	1/Mese
Inserimento di un catalogo	Modifica	10/Anno

Inserimento di un prodotto	Modifica	1000/Anno
Aggiornamento di un reso	Modifica	10/Mese

1. PROGETTAZIONE CONCETTUALE

2.1 MODELLO ER



2.2 REGOLE DI VINCOLO

- Ogni fornitore pubblica un solo catalogo all'anno
- Si è scelto di creare l'entità indirizzo e l'entità città perché è necessario analizzare le destinazioni delle spedizioni per migliorarne l'efficienza
- Tutti gli indirizzi devono essere memorizzati con la stessa convenzione per assicurarne la coerenza
- I nomi delle città devono essere univoci
- Un prodotto acquistato è una specifica istanza di un prodotto
- La cardinalità minima dell'associazione 'Esegue' dal lato del cliente è 0 perchè un cliente che ha appena attivato il suo account non può già aver eseguito un ordine
- Un reso può non essere ancora stato saldato; è stato dunque introdotto il campo booleano Saldato che, se assume valore True, indica che il saldo è avvenuto
- La cardinalità minima dell'associazione 'Spedizione' dal lato dell'ordine è 0 perchè un cliente può scegliere di andare a ritirare il suo ordine al negozio al posto di farlo spedire.

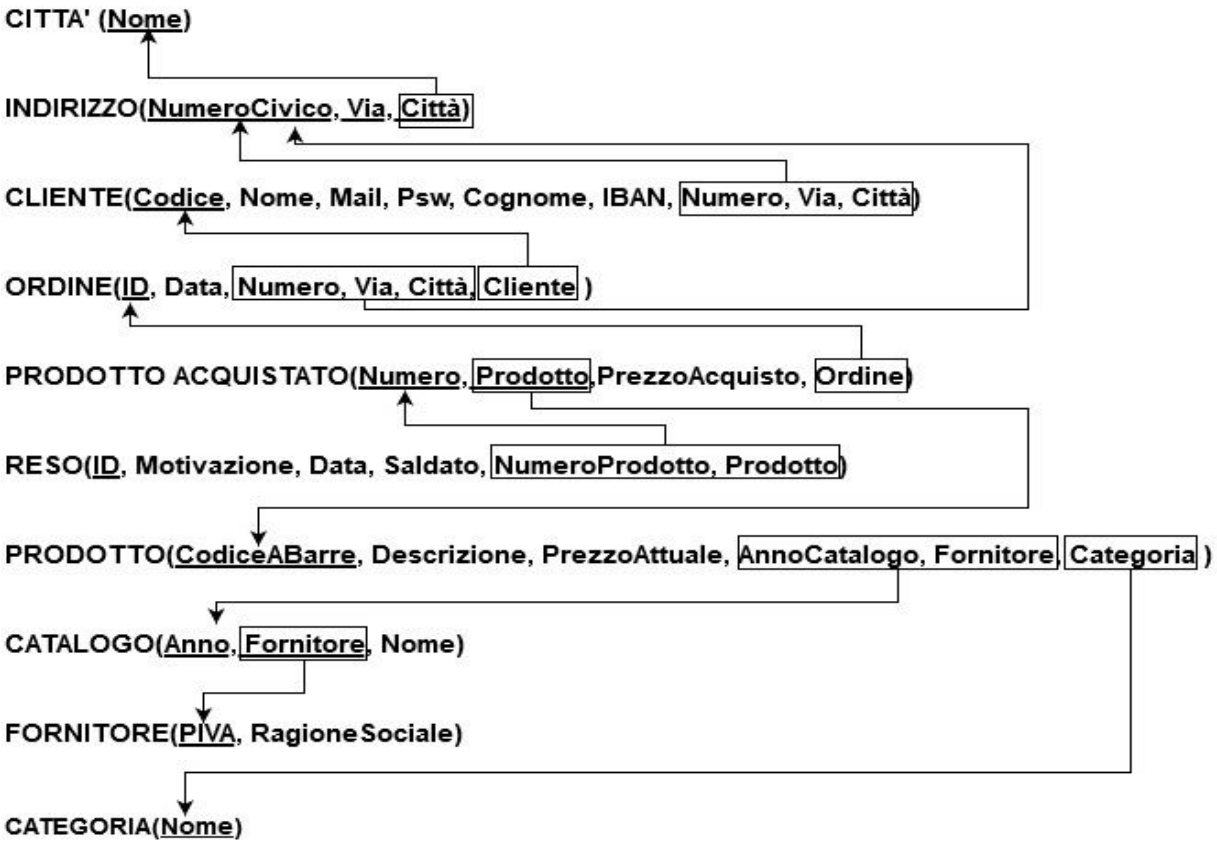
2.3 ATTRIBUTI E TIPI DI DATI

ENTITÀ	**NOME ATTRIBUTO**	**TIPO**
CITTÀ	Nome	VarChar(30)
INDIRIZZO	NumeroCivico	VarChar(5)
INDIRIZZO	Via	VarChar(40)
CLIENTE	Nome	VarChar(30)
CLIENTE	Cognome	VarChar(30)
CLIENTE	Mail	VarChar(50)
CLIENTE	Codice	VarChar(30)
CLIENTE	Psw	VarChar(30)
CLIENTE	IBAN	Char(27)
ORDINE	ID	VarChar(30)
ORDINE	Data	Date
PRODOTTO ACQUISTATO	Numero	Integer

PRODOTTO ACQUISTATO	PrezzoAcquisto	Real
RESO	ID	VarChar(30)
RESO	Motivazione	VarChar(100)
RESO	Saldato	Boolean
RESO	Data	Date
PRODOTTO	Descrizione	VarChar(100)
PRODOTTO	CodiceABarre	VarChar(30)
PRODOTTO	PrezzoAttuale	Real
CATEGORIA	Nome	VarChar(30)
CATALOGO	Anno	Smallint
CATALOGO	Nome	VarChar(30)
FORNITORE	PIVA	Char(11)
FORNITORE	RagioneSociale	VarChar(50)

3. PROGETTAZIONE LOGICA

3.1 MODELLO RELAZIONALE



3.2 REGOLE DI VINCOLO

- Tutti gli attributi sottolineati devono essere non nulli
- Ogni CLIENTE deve avere tutti i suoi attributi non nulli
- Ogni ORDINE deve avere Data e Cliente non nulli
- Ogni PRODOTTO ACQUISTATO deve avere tutti i suoi attributi non nulli
- Ogni RESO deve avere tutti i suoi attributi non nulli
- Ogni PRODOTTO deve avere tutti i suoi attributi non nulli
- Ogni CATALOGO deve avere tutti i suoi attributi non nulli
- Ogni FORNITORE deve avere tutti i suoi attributi non nulli

4. PROGETTAZIONE FISICA

4.1 SQL: STRUTTURA DATABASE

```
CREATE TABLE Citta (
        Nome VARCHAR(30) PRIMARY KEY   );
CREATE TABLE Indirizzo (
        NumeroCivico VARCHAR(5),
        Via VARCHAR(40),
        Citta VARCHAR(30),
        PRIMARY KEY(NumeroCivico,Via,Citta),
        FOREIGN KEY (Citta) REFERENCES Citta(Nome) ON UPDATE CASCADE    );
CREATE TABLE Cliente (
        Codice VARCHAR(30) PRIMARY KEY,
        Nome VARCHAR(30) NOT NULL,
        Cognome VARCHAR(30) NOT NULL,
        Mail VARCHAR(50) NOT NULL,
        Psw VARCHAR(30) CHECK (LENGTH(Psw) >= 8) NOT NULL,
        IBAN CHAR(27) NOT NULL,
        Numero VARCHAR(5) NOT NULL,
        Via VARCHAR(40) NOT NULL,
        Citta VARCHAR(30) NOT NULL,
        FOREIGN KEY (Numero, Via, Citta) REFERENCES Indirizzo(NumeroCivico, Via,    Citta) ON
UPDATE CASCADE ON DELETE RESTRICT    );
CREATE TABLE Ordine (
        ID VARCHAR(30) PRIMARY KEY,
        Data DATE NOT NULL,
```

```
        Numero VARCHAR(5),

        Via VARCHAR(40),

        Citta VARCHAR(30),

        Cliente VARCHAR(30) NOT NULL,

        FOREIGN KEY (Numero, Via, Citta) REFERENCES Indirizzo(NumeroCivico, Via, Citta) ON
UPDATE CASCADE ON DELETE RESTRICT,

        FOREIGN KEY (Cliente) REFERENCES Cliente(Codice) ON UPDATE CASCADE   );

CREATE TABLE Categoria (

        Nome VARCHAR(30) PRIMARY KEY   );

CREATE TABLE Fornitore (

        PIVA CHAR(11) PRIMARY KEY,

        RagioneSociale VARCHAR(50) NOT NULL   );

CREATE TABLE Catalogo (

        Anno SMALLINT CHECK (Anno>1900 AND Anno<3000),

        Fornitore CHAR(11),

        Nome VARCHAR(30) NOT NULL,

        PRIMARY KEY (Anno, Fornitore),

        FOREIGN KEY (Fornitore) REFERENCES Fornitore(PIVA) ON UPDATE CASCADE   );

CREATE TABLE Prodotto (

        CodiceABarre VARCHAR(30) PRIMARY KEY,

        Descrizione VARCHAR(100) NOT NULL,

        AnnoCatalogo SMALLINT NOT NULL,

        Fornitore CHAR(11) NOT NULL,

        Categoria VARCHAR(30) NOT NULL,

        PrezzoAttuale REAL NOT NULL,

        FOREIGN KEY (AnnoCatalogo, Fornitore) REFERENCES Catalogo(Anno, Fornitore) ON
UPDATE CASCADE,

        FOREIGN KEY (Categoria) REFERENCES Categoria(Nome) ON UPDATE CASCADE ON
```

```
DELETE RESTRICT  );

CREATE TABLE ProdottoAcquistato (

        Numero INTEGER,

        Prodotto VARCHAR(30),

        Ordine VARCHAR(30) NOT NULL,

        PrezzoAcquisto REAL NOT NULL,

        PRIMARY KEY (Numero,Prodotto),

        FOREIGN KEY (Prodotto) REFERENCES Prodotto(CodiceABarre) ON UPDATE CASCADE,

        FOREIGN KEY (Ordine) REFERENCES Ordine(ID) ON UPDATE CASCADE   );

CREATE TABLE Reso (

        ID VARCHAR(30) PRIMARY KEY,

        Motivazione VARCHAR(100) NOT NULL,

        Data DATE NOT NULL,

        Saldato BOOLEAN DEFAULT FALSE NOT NULL,

        NumeroProdotto INTEGER NOT NULL,

        Prodotto VARCHAR(30) NOT NULL,

        FOREIGN KEY (NumeroProdotto,Prodotto) REFERENCES
ProdottoAcquistato(Numero,Prodotto) ON UPDATE CASCADE  );
```

4.2 SQL: INTERROGAZIONI

FATTURATO ANNUO PER OGNI CLIENTE:

Algebra relazionale:

$$_{\text{Cliente}}\mathcal{F}_{\text{SUM(PrezzoAcquisto)}}(\sigma_{\text{Data>=2019-01-01 AND Data<=2019-12-31}}(\text{Ordine} \bowtie_{\text{ID=Ordine}} \text{ProdottoAcquistato}))$$

SQL:

```
SELECT Cliente, SUM(P.PrezzoAcquisto)
FROM Ordine AS O JOIN ProdottoAcquistato AS P ON O.ID=P.Ordine
WHERE Data>=’2019-01-01’ AND Data<=’2019-12-31’
GROUP BY Cliente;
```

LISTA ARTICOLI PIÙ ACQUISTATI:

$\varrho_{Tot(Prodotto,\ Qta)}(_{Prodotto}\mathcal{F}_{COUNT(*)}(ProdottoAcquistato))$
$Massimo \leftarrow \mathcal{F}_{MAX(Qta)}(Tot)$
$\pi_{Prodotto}(\sigma_{Qta \in Massimo}(Tot))$

SQL:

```
CREATE VIEW Tot AS(
        SELECT Prodotto, COUNT(*) AS Qta
        FROM ProdottoAcquistato
        GROUP BY Prodotto );

SELECT Prodotto
FROM Tot
WHERE Qta IN (
        SELECT MAX(Qta)
        FROM Tot    );
```

LISTA ARTICOLI MENO ACQUISTATI:

Algebra relazionale:

$\varrho_{Tot(Prodotto,\ Qta)}(_{Prodotto}\mathcal{F}_{COUNT(*)}(ProdottoAcquistato))$
$Minimo \leftarrow \mathcal{F}_{MIN(Qta)}(Tot)$
$\pi_{Prodotto}(\sigma_{Qta \in Minimo}(Tot))$

SQL:

```
CREATE VIEW Tot AS(
        SELECT Prodotto, COUNT(*) AS Qta
        FROM ProdottoAcquistato
        GROUP BY Prodotto    );

SELECT Prodotto
FROM Tot
WHERE Qta IN (
        SELECT MIN(Qta)
        FROM Tot   );
```

FORNITORE DEL PRODOTTO RESO PIÙ VOLTE:

$\varrho_{Resinum(Prod,\ Num)}(_{Prodotto}\mathcal{F}_{COUNT(*)}(Reso))$
$\mathcal{F}_{MAX(Num)}(Resinum) \rightarrow Massimo$

$\pi_{\text{RagioneSociale,PIVA}}(\text{Fornitore} \bowtie_{\text{PIVA=Fornitore}} \text{Prodotto} \bowtie_{\text{CodiceABarre=Prod}} (\sigma_{\text{Num} \in \text{Massimo}}(\text{Resinum}))$

SQL:

```
CREATE VIEW Resinum AS(
        SELECT Prodotto AS Prod, COUNT(*) AS Num
        FROM Reso
        GROUP BY Prodotto    );

SELECT RagioneSociale, PIVA
FROM Fornitore AS F JOIN Prodotto AS P ON F.PIVA=P.Fornitore JOIN Resinum AS R ON
P.CodiceABarre=R.Prod
WHERE Num IN (
        SELECT MAX(Num)
        FROM Resinum   );
```

NUMERO CLIENTI INATTIVI

Algebra relazionale:

$\varrho_{\text{(ClientiInattivi)}}(\mathcal{F}_{\text{COUNT(*)}}(\sigma_{\text{Cliente IS NULL}}(\text{Cliente} \rtimes_{\text{Codice=Cliente}} (\pi_{\text{ID,Cliente}}(\text{Ordine})))))$

SQL:

```
SELECT COUNT(*) AS ClientiInattivi
FROM Cliente AS C LEFT OUTER JOIN Ordine AS O ON C.Codice=O.Cliente
WHERE O.Cliente IS NULL;
```

NUMERO SPEDIZIONI PER CITTA’

Algebra relazionale:

$\varrho_{\text{(Citta,NumSpedizioni)}}(_{\text{Citta}}\mathcal{F}_{\text{COUNT(*)}}(\sigma_{\text{Citta IS NOT NULL}}(\text{Ordine})))$

SQL:

```
SELECT Citta, COUNT(*) AS NumSpedizioni
FROM Ordine
WHERE Citta IS NOT NULL
GROUP BY Citta;
```

5. APPLICAZIONE

L'applicazione che abbiamo creato è sviluppata in linguaggio java. Prima di spiegare le specifiche dell'applicazione è necessario comprenderne lo scopo e chi sono i suoi utilizzatori. Questa applicazione ha come utenti i dipendenti dell'azienda che devono analizzare i dati dell'e-commerce, inserire nuovi dati come ad esempio nuovi prodotti o cataloghi e modificare lo stato dei resi.
Sono dunque state implementate alcune delle operazioni più importanti che devono essere eseguite dall'azienda.

All'esecuzione si apre una finestra nella quale eseguire il login e connettersi al server.

Le interrogazioni, e quindi l'analisi dei dati, sono state implementate nella sezione denominata "Statistiche".
Premendo sul bottone Statistiche del menù principale si apre una schermata nella quale si possono visualizzare il fatturato annuo, il codice a barre (alfanumerico) degli articoli più venduti e la lista degli ordini finora effettuati.

L'inserimento di nuovi dati è stato implementato nella sezione denominata "Aggiungi".
Premendo sul bottone Aggiungi del menù principale si possono aggiungere nuovi cataloghi o nuovi prodotti.
Selezionando Aggiungi catalogo, vengono visualizzati tutti i cataloghi già presenti. In basso si possono invece inserire l'anno del nuovo catalogo, il suo fornitore e il suo nome; premendo il bottone aggiungi, i dati del nuovo catalogo vengono inseriti nella base di dati.
Selezionando aggiungi prodotto vengono visualizzati tutti i prodotti già presenti. In basso si possono invece inserire il codice a barre (alfanumerico) del nuovo prodotto, il suo catalogo di appartenenza, la sua categoria di appartenenza, il suo prezzo attuale e la sua descrizione; premendo il bottone aggiungi, i nuovi dati del nuovo prodotto vengono inseriti nella base di dati.

L'aggiornamento dei dati già presenti è stato implementato nella sezione denominata "Modifica reso".
Premendo sul bottone Modifica reso, si apre una schermata nella quale si possono visualizzare tutti i resi effettuati. Facendo doppio click su uno dei resi, le relative informazioni vengono trasferite nelle caselle in basso. Interagendo con la checkbox è dunque possibile modificare lo stato del reso da Saldato a Non saldato e viceversa. In questa finestra è possibile modificare anche la motivazione del reso. Premendo il bottone aggiungi, le modifiche effettuate vengono salvate nella base di dati.