Università degli Studi di Ferrara
Dipartimento di Ingegneria

Corso di Laurea Triennale in Ingegneria Elettronica e Informatica

Documentazione Basi di Dati

Progetto E-Commerce: BINS

Studente:
Michele Vaccari
Matricola 121955

Docente:
Prof. Denis Ferraretti

Anno Accademico 2016–2017

INDICE

ELENCO DELLE FIGURE

INTRODUZIONE

BINS (acronimo ricorsivo di "BINS Is Not Shopping") è un applicazione web liberamente disponibile, indicata soprattutto per commercializzare prodotti alimentari. Lo scopo di questo lavoro è di documentare l'applicazione usando l'UML (Unified Model Language).

La documentazione è articolata come segue.

IL PRIMO CAPITOLO contiene le specifiche del progetto descrivendo le caratteristiche e le categorie di utenti dell'applicazione.

IL SECONDO CAPITOLO contiene la descrizione delle singole entità e delle loro associazioni.

IL TERZO CAPITOLO spiega l'algoritmo di traduzione utilizzato per passare dal modello E-R al modello relazionale.

IL QUARTO CAPITOLO spiega il procedimento di normalizzazione e i vari test effettuati per certificare il modello relazionale in 3NF.

IL QUINTO CAPITOLO riporta il codice SQL utilizzato per creare le tabelle che costituiscono la base dati e alcune interrogazioni effettuate.

IL SESTO CAPITOLO mostra le schermate dell'applicazione suddivise per funzionalità e per tipo di utente.

Per il *Presentation Level* si è usato il framework *Bootstrap* sfruttando al meglio il *responsive design* che viene messo a disposizione dal framework. Per quanto riguarda l'usabilità e il design si è scelto l'approccio *mobile first* ma si è anche prestato attenzione all'uso dell'applicazione in modalità desktop.

Per l'application server si è usato *Tomcat* e si è utilizzato il linguaggio di progammazione Java. Per interfacciare l'application server con il database si è utilizzato JDBC.

Il DBMS utilizzato è *MySQL*.

Di seguito si riporta il diagramma approssimato relativo allo sviluppo del lavoro in termini di tempo attraverso un diagramma di Gantt:

1 SPECIFICHE PROGETTO

1.1 CARATTERISTICHE DEL SISTEMA

Si vuole progettare un'applicazione web per la vendita online dei prodotti di un supermercato. L'applicazione deve avere un front-end multilingue (italiano e inglese).

I prodotti del supermercato sono suddivisi in reparti (ortofrutta, macelleria, accessori per la casa, cura della persona, ecc.) e ogni prodotto può avere una data di scadenza.

L'applicazione web deve avere le seguenti caratteristiche:

1. Possibilità di visualizzare il catalogo dei prodotti, navigabile per reparto, caratteristiche, marchio, ricerca libera, ecc. Possibilità di vedere il singolo prodotto con tutti i dettagli;
2. Possibilità di visualizzare la lista delle ricette. Ogni ricetta ha un nome, un autore, un tempo di preparazione, un testo con le specifiche della sua preparazione, una lista di prodotti;
3. Possibilità di inserire prodotti nel carrello e di effettuare l'acquisto di più prodotti in diverse quantità. Predisporre la gestione dei prezzi, del totale carrello, la gestione della disponibilità di magazzino, impedendo di poter acquistare quantità non disponibili e prodotti scaduti;
4. Gestione di una o più shopping list (lista della spesa);
5. Possibilità di inserire o selezionare un indirizzo di consegna da una rubrica personale, di simulare il pagamento e di confermare l'ordine;
6. Possibilità di visualizzare lo stato dell'ordine e lo storico degli ordini effettuati;
7. Gestione di coupon o buoni sconto in fase di acquisto, possibilità di visualizzare il tracking dell'ordine in consegna;
8. Possibilità di gestire il catalogo (inserimento, modifica, blocco prodotti), i reparti, i brand, ecc. Possibilità di gestire la disponibilità di magazzino;
9. Gestione di prodotti in push (per cui spingere la vendita) con inserimento in una "vetrina in home page" o in un'area promo;
10. Possibilità di visualizzare gli utenti, verificare il numero degli ordini per utente, bloccare eventuali utenti, gestire gli altri utenti amministratori;

11. Gestione dei coupon o buoni sconto;
12. Gestione del tracking dell'ordine, con simulazione di tutti i cambi di stato (consegnato al corriere, in viaggio, consegnato al destinatario, ecc.);

1.2 UTENTI DEL SISTEMA

Il sistema prevede che le categorie di utenti sia così rappresentata:

UTENTI PUBBLICI Possono effettuare i punti dal 1 al 2 ed eventualmente registrarsi.

UTENTI REGISTRATI Possono effettuare i punti dal 1 al 7 compresi.

AMMINISTRATORI Possono effettuare i punti dal 8 al 12 compresi.

2 DIAGRAMMA E-R

Una volta analizzate le specifiche del progetto per il quale si vuole realizzare il database, si identificano e si descrivono le singole entità e le loro associazioni.

2.1 DESCRIZIONE DELLE ENTITÀ

2.1.1 Utente Amministratore

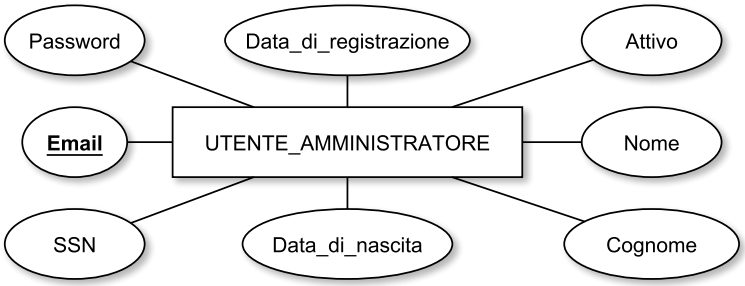

Figura 1: Entità Utente Amministratore

Descrizione Attributi

ID Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

DATA INSERIMENTO Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

NOME Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

DESCRIZIONE Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

DISPONIBILITA Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

PESO Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

PREZZO UNITARIO Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

IMMAGINE Chiave primaria dell'entità che identifica univocamente il prodotto. Il valore di tale attributo viene generato mediante un contatore.

2.1.2 Utente Amministratore

2.1.3 Utente Registrato

2.1.4 Ordine

2.1.5 Pagamento

2.1.6 Fattura

2.1.7 Ricetta

2.1.8 Marchio

2.1.9 Caratteristica

2.1.10 Reparto

2.1.11 Coupon

2.2 DESCRIZIONE DELLE ASSOCIAZIONI

2.3 IL MODELLO E-R COMPLETO

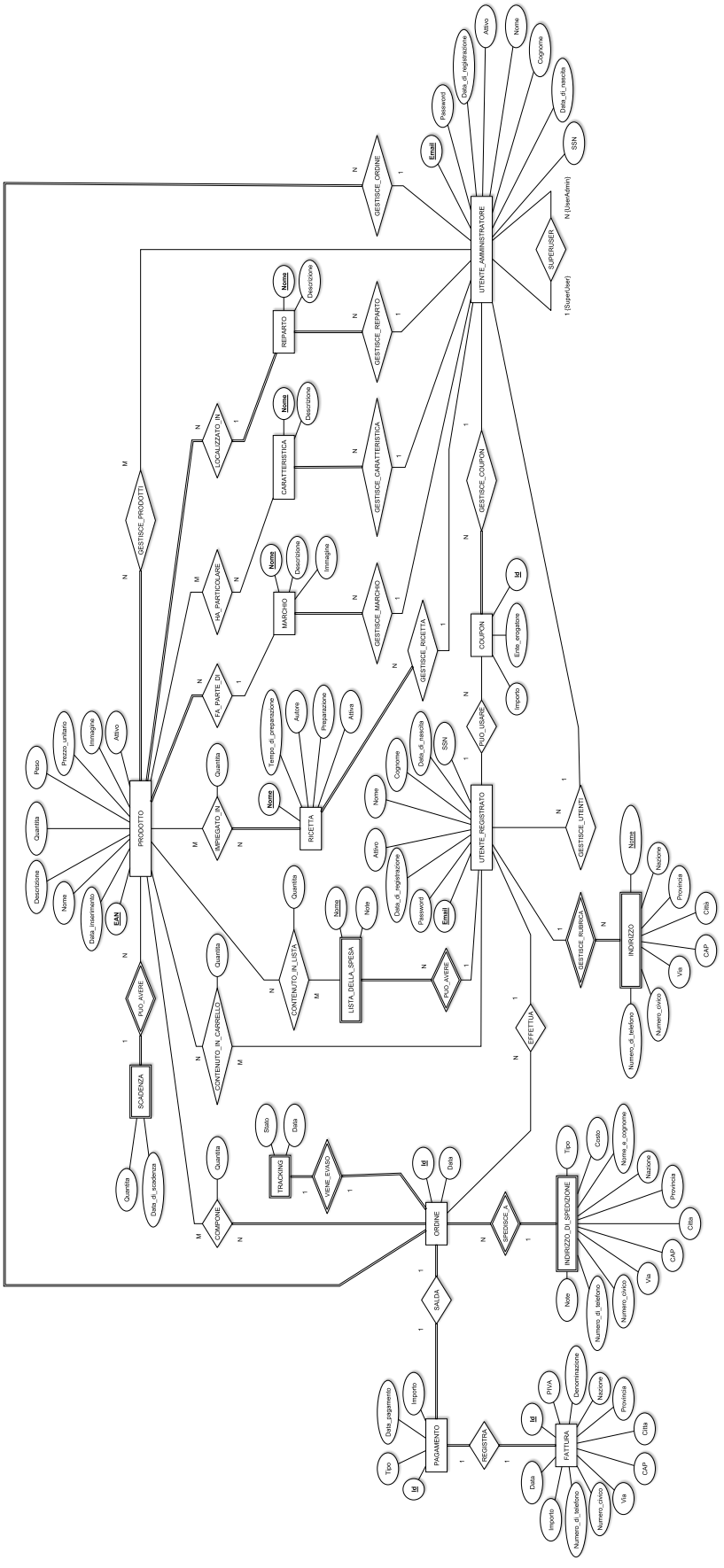

Figura 2: Schema E-R completo

3 MODELLO RELAZIONALE

3.1 L'ALGORITMO DI TRADUZIONE

La fase successiva della progettazione di una base di dati consiste nel passare dalla progettazione concettuale alla progettazione logica. Per passare al modello relazionale, si applica un semplice algoritmo.

L'algoritmo che si utilizza per la traduzione dal modello E-R in modello relazionale è formato dalle seguenti operazioni:

1. Traduzione di tipi di entità;
2. Traduzione di tipi di entità deboli;
3. Traduzione di associazioni binarie di tipo 1 : 1;
4. Traduzione di associazioni binarie di tipo 1 : N;
5. Traduzione di associazioni binarie di tipo N : M;

3.1.1 Traduzione di entità

Per ogni tipo di entità (forte) nello schema E-R si costruisce una relazione che contiene tutti gli attributi semplici dell'entità.

3.1.2 Traduzione di entità deboli

Per ogni tipo di entità debole dello schema E-R con tipo di entità proprietario, si costruisce una relazione e si inseriscono tutti gli attributi semplici dell'entità debole come attributi della relazione. Si inseriscono come attributi di chiave esterna della relazione gli attributi di chiave primaria delle relazioni corrispondenti ai tipi di entità proprietari. La chiave primaria della relazione è data dalla combinazione delle chiavi primarie delle entità proprietarie e dalla chiave parziale del tipo di entità debole (se esiste).

3.1.3 Traduzione di associazioni binarie 1:1

Per ogni tipo di associazione binaria 1 : 1 nello schema E-R si individuano le due relazioni coinvolte dall'associazione. Per la traduzione si usa l'approccio basato su chiavi esterne. Si sceglie una delle due relazioni coinvolte, preferibilmente la relazione corrispondente a un tipo di entità con vincolo di partecipazione totale, e si inserisce come chiave esterna la chiave primaria

della seconda relazione. Infine si inseriscono sulla relazione scelta tutti gli attributi semplici del tipo di associazione 1 : 1.

3.1.4 Traduzione di associazioni binarie 1:N

Per ogni tipo di associazione binaria 1 : N nello schema E-R si individuano le due relazioni coinvolte dall'associazione. Per la traduzione si individuata la relazione che rappresenta il tipo di entità partecipante lato-N del tipo di associazione e si inserisce come chiave esterna la chiave primaria della relazione che rappresenta l'altro tipo di entità partecipante all'associazione. Infine si inseriscono sulla relazione scelta tutti gli attributi semplici del tipo di associazione 1 : N.

3.1.5 Traduzione di associazioni binarie N:M

Per ogni tipo di associazione binaria M : N nello schema E-R si costruisce una nuova relazione che rappresenta l'associazione. Si inseriscono come attributi di chiave esterna della relazione le chiavi primarie delle relazioni che rappresentano i tipi di entità partecipanti, le loro combinazioni formano la chiave primaria della relazione. Infine si inseriscono nella nuova relazione tutti gli attributi semplici del tipo di associazione M : N.

3.2 APPLICAZIONE DELL'ALGORITMO DI TRADUZIONE

Di seguito si riporta la traduzione in modello relazionale del modello E-R analizzato:

PRODOTTO

EAN	Data_inserimento	Nome	Descrizione	Quantita	Peso	Prezzo_unitario	Immagine	Attivo

RICETTA

Nome	**Autore**	Tempo_di_preparazione	Preparazione	Attiva

MARCHIO

Nome	Descrizione	Immagine

CARATTERISTRICA

Nome	Descrizione

REPARTO

Nome	Descrizione

UTENTE_AMMINISTRATORE

Email	Password	Data_di_registrazione	Attivo	Nome	Cognome	Data_di_nascita	SSN

COUPON

Id	Ente_erogatore	Importo

UTENTE_REGISTRATO

Email	Password	Data_di_registrazione	Attivo	Nome	Cognome	Data_di_nascita	SSN

ORDINE

Id	Data

PAGAMENTO

Id	Tipo	Data_pagamento	Importo

FATTURA

Id	PIVA	Denominazione	Nazione	Provincia	Citta	CAP	Via	Numero_civico	Numero_di_telefono	Importo	Data

Figura 3: Traduzione di entità

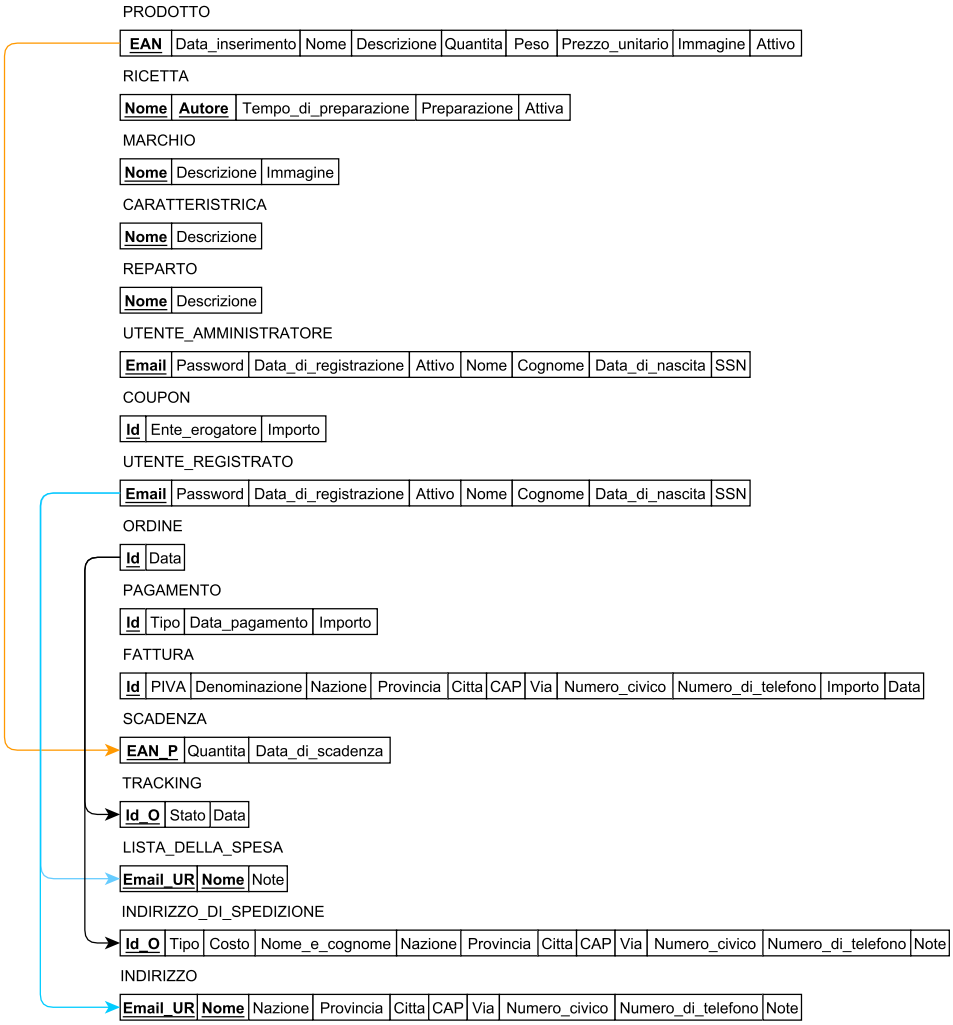

Figura 4: Traduzione di entità deboli

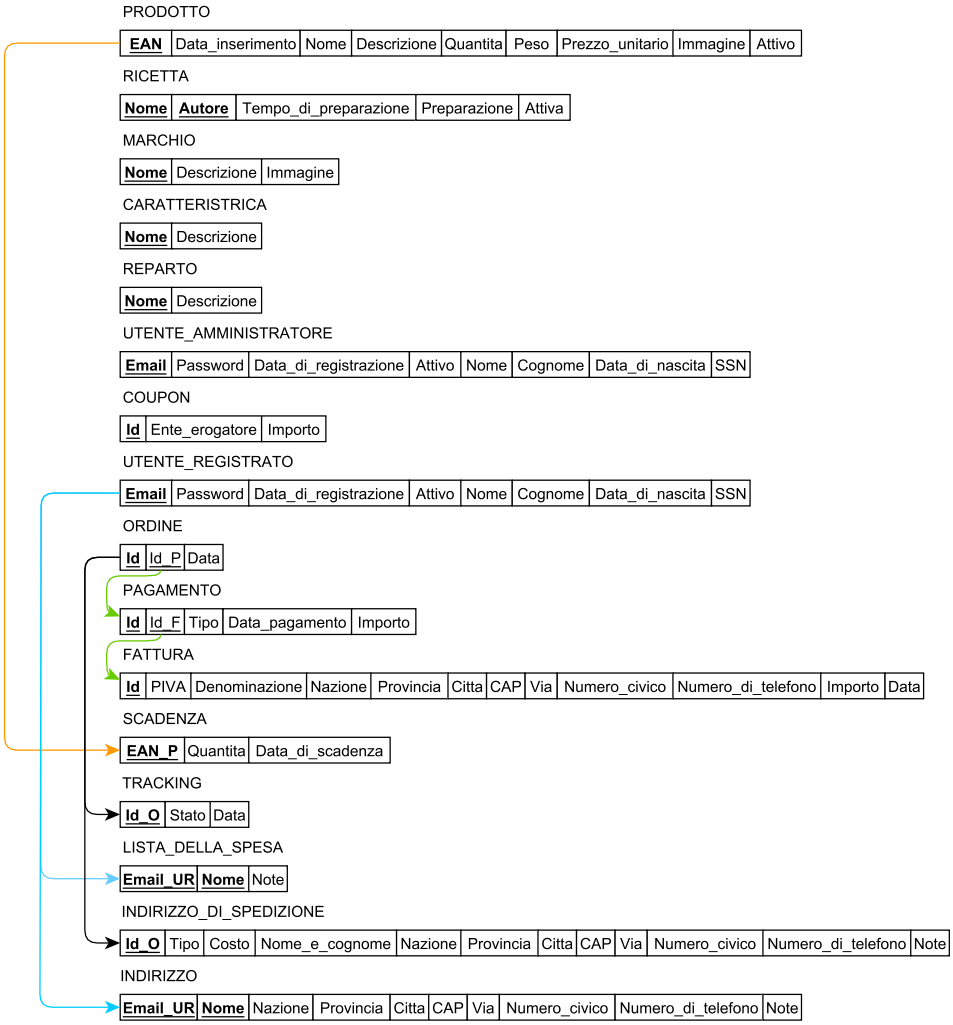

Figura 5: Traduzione di associazioni binarie 1:1

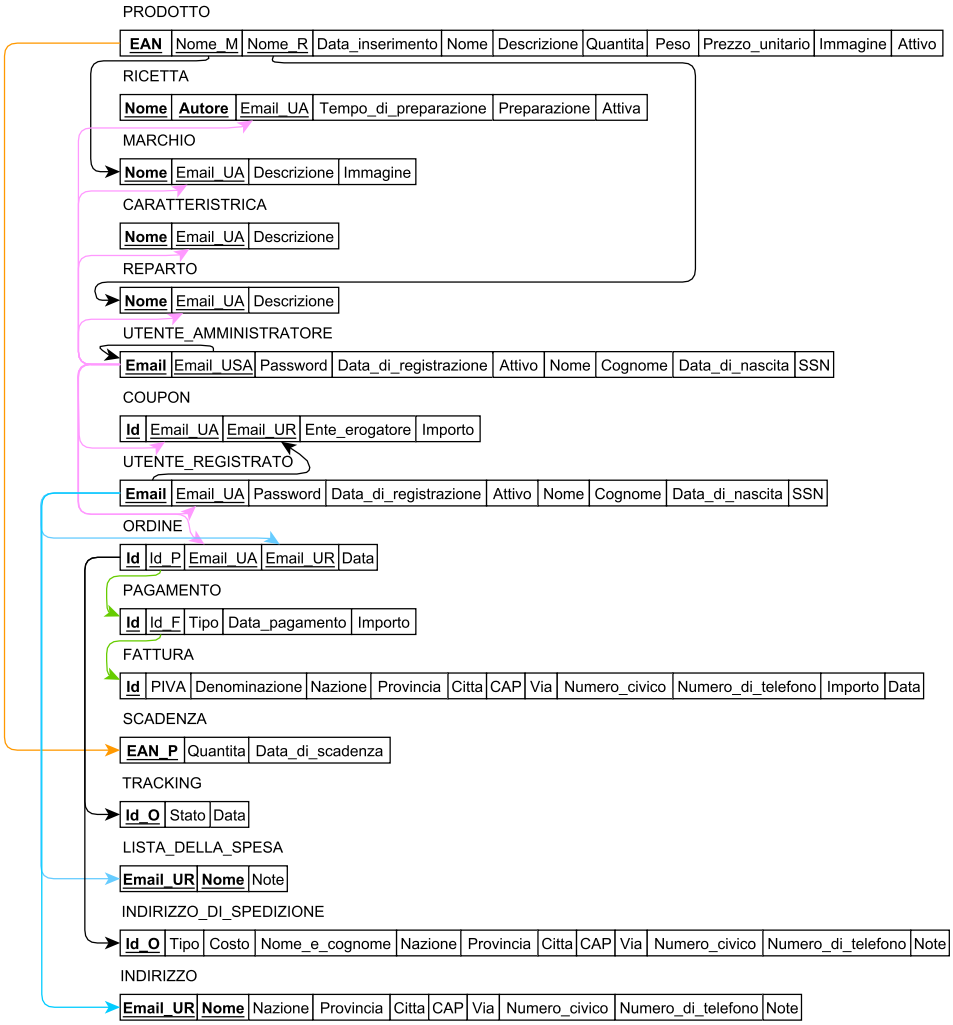

Figura 6: Traduzione di associazioni binarie 1:N

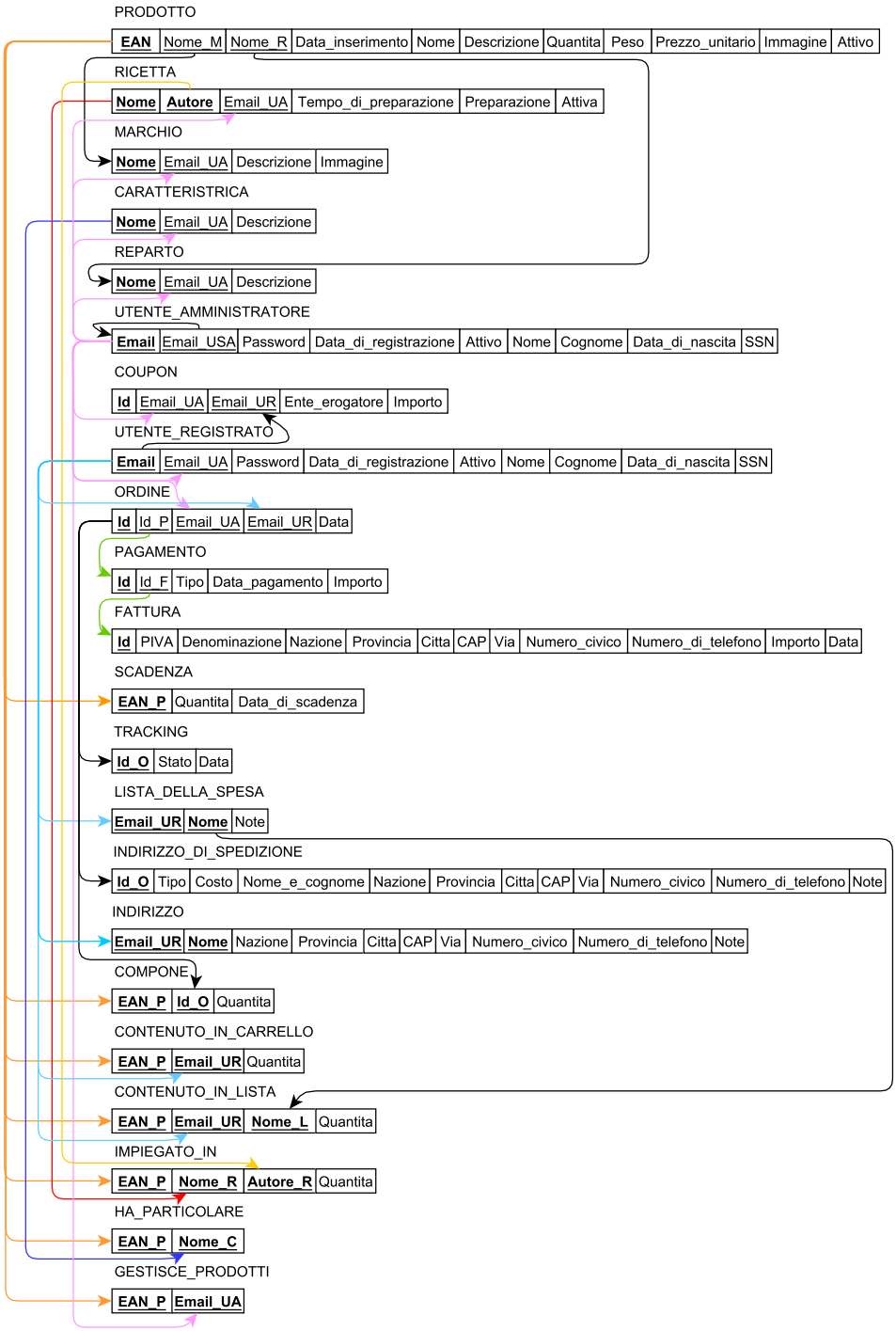

Figura 7: Traduzione di associazioni binarie M:N

4 NORMALIZZAZIONE

4.1 CENNI TEORICI

Per normalizzazione si intende il procedimento che ha lo scopo di eliminare la ridondanza e le eventuali incoerenze del database. Il processo di normalizzazione sottopone uno schema relazionale a una serie di test per certificare se soddisfa una data forma normale.

4.1.1 Prima forma normale (1NF)

Si dice che uno schema relazionale è in prima forma normale se il dominio di un attributo comprende valori atomici e se il valore di un qualsiasi attributo in una tupla sia un singolo valore del dominio, ovvero:

- Tutte le righe della tabella contengono lo stesso numero di colonne;
- Gli attributi rappresentano informazioni elementari;
- I valori che compaiono in una colonna sono dello stesso tipo, cioè appartengono allo stesso dominio;
- Ogni riga è diversa da tutte le altre, cioè non ce ne possono essere due con gli stessi valori nelle colonne;
- L'ordine con il quale le righe compaiono nella tabella è irrilevante.

La prima forma normale (1NF) è già parte integrante della definizione formale di relazione nel modello relazionale.

4.1.2 Seconda forma normale (2NF)

Si dice che uno schema relazionale è in seconda forma normale (2NF) quando è in prima forma normale (1NF) e tutti i suoi attributi non-chiave dipendono dall'intera chiave, cioè non possiede attributi che dipendono soltanto da una parte della chiave.

La seconda forma normale elimina la dipendenza parziale degli attributi dalla chiave e riguarda il caso di relazioni con chiavi composte, cioè formate da più attributi.

4.1.3 Terza forma normale (3NF)

Si dice che uno schema relazionale è in terza forma normale (3NF) quando è in seconda forma normale (2NF) e tutti i suoi attributi non-chiave dipen-

dono direttamente dalla chiave, cioè non possiede attributi non-chiave che dipendono da altri attributi non-chiave.

La terza forma normale elimina la dipendenza transitiva degli attributi dalla chiave.

4.2 PROCESSO DI NORMALIZZAZIONE

Partendo dalla figura 7 a pagina 12, si analizzano le varie chiavi e le varie dipendenze funzionali per ridurre lo schema relazionale precedentemente descritto in 2NF e in 3NF.

Si analizza la relazione FATTURA e si nota che essa risulta essere già in seconda forma normale (2NF); infatti essendoci un solo attributo chiave (*Id*) tutti gli altri attributi dipendono funzionalmente da quest'ultimo. Infine non essendoci dipendenze funzionali transitive, tale relazione risulta essere anche in terza forma normale (3NF).

Lo stesso ragionamento si può applicare alle relazioni SCADENZA, INDIRIZZO_DI_SPEDIZIONE e TRACKING.

Si considerano ora le seguenti relazioni:

- MARCHIO
- CARATTERISTICA
- REPARTO
- UTENTE_REGISTRATO
- PAGAMENTO

Tutte le relazioni sono formate da un unico attributo chiave e da una chiave esterna, pertanto risultano essere in 3NF.

La relazione UTENTE_AMMINISTRATORE è formata da un'unico attributo chiave e presenta l'attributo *Email_USA* non-primo che dipende funzionalmente, direttamente e completamente dalla chiave; quindi ne segue che la relazione è in 3NF.

Si considerano ora le seguenti relazioni:

- PRODOTTO
- COUPON

Entrambe le relazioni sono formate da un unico attributo chiave e da due chiavi esterne, pertanto risultano essere in 3NF.

La relazione ORDINE presenta un attributo chiave e tre chiavi esterne, quindi risulta essere in 3NF.

La relazione RICETTA presenta due attributi chiave e una esterna, quindi risulta essere in 3NF.

La relazioni LISTA_DELLA_SPESA e INDIRIZZO presentano due attributi chiave, quindi risulta essere in 3NF.

Per quanto riguarda le relazioni:

- COMPONE
- CONTENUTO_IN_CARRELLO
- HA_PARTICOLARE
- GESTISCE_PRODOTTI

Tali relazioni sono formate da due attributi chiave che sono chiavi esterne, pertanto risultano essere in 3NF.

Le relazioni CONTENUTO_IN_LISTA e IMPIEGATO_IN presentano tre attributi chiave che sono chiavi esterne, quindi risultano essere in 3NF.

Dall'analisi svolta si deduce che la figura 7 a pagina 12 è in terza forma normale (3NF).

5 CODICE SQL

5.1 DDL

Si riporta il codice SQL (estrapolato da phpMyAdmin) utilizzato per creare le tabelle che costituiscono la base di dati e i vincoli di integrità referenziale.

5.1.1 UTENTE_AMMINISTRATORE

Struttura della tabella *UTENTE_AMMINISTRATORE*:

```
CREATE TABLE IF NOT EXISTS `UTENTE_AMMINISTRATORE` (
        `Email` VARCHAR(100) NOT NULL,
        `Email_UAS` VARCHAR(100) NOT NULL DEFAULT 'admin@bins.com',
        `Password` VARCHAR(32) NOT NULL,
        `Data_di_registrazione` TIMESTAMP NOT NULL,
        `Attivo` BOOLEAN NOT NULL DEFAULT TRUE,
        `Nome` VARCHAR(50) NOT NULL,
        `Cognome` VARCHAR(50) NOT NULL,
        `Data_di_nascita` DATE NOT NULL,
        `SSN` VARCHAR(16) NOT NULL,
        PRIMARY KEY (`Email`),
        FOREIGN KEY (`Email_USA`) REFERENCES UTENTE_AMMINISTRATORE(`
            Email`)
) ENGINE = InnoDB;
```

5.1.2 UTENTE_REGISTRATO

Struttura della tabella *UTENTE_REGISTRATO*:

```
CREATE TABLE IF NOT EXISTS `UTENTE_REGISTRATO` (
        `Email` VARCHAR(100) NOT NULL,
        `Email_UA` VARCHAR(100) NOT NULL DEFAULT 'admin@bins.com',
        `Password` VARCHAR(32) NOT NULL,
        `Data_di_registrazione` TIMESTAMP NOT NULL,
        `Attivo` BOOLEAN NOT NULL DEFAULT TRUE,
        `Nome` VARCHAR(50) NOT NULL,
        `Cognome` VARCHAR(50) NOT NULL,
        `Data_di_nascita` DATE NOT NULL,
        `SSN` VARCHAR(16) NOT NULL,
        PRIMARY KEY (`Email`),
        FOREIGN KEY (`Email_UA`) REFERENCES UTENTE_AMMINISTRATORE(`Email
            `)
) ENGINE = InnoDB;
```

5.1.3 FATTURA

Struttura della tabella *FATTURA*:

```
CREATE TABLE IF NOT EXISTS ‘FATTURA‘ (
        ‘Id‘ INT UNSIGNED NOT NULL AUTO_INCREMENT,
        ‘PIVA‘ VARCHAR(16) NOT NULL,
        ‘Denominazione‘ VARCHAR(100) NOT NULL,
        ‘Nazione‘ VARCHAR(50) NOT NULL,
        ‘Provincia‘ VARCHAR(50) NOT NULL,
        ‘Citta‘ VARCHAR(50) NOT NULL,
        ‘CAP‘ VARCHAR(5) NOT NULL,
        ‘Via‘ VARCHAR(100) NOT NULL,
        ‘Numero_civico‘ VARCHAR(10) NOT NULL,
        ‘Numero_di_telefono‘ VARCHAR(30) NOT NULL,
        ‘Importo‘ DOUBLE(8,2) UNSIGNED NOT NULL DEFAULT ’0.00’,
        ‘Data‘ TIMESTAMP NOT NULL,
        PRIMARY KEY (‘Id‘)
) ENGINE = InnoDB AUTO_INCREMENT=1;
```

5.1.4 PAGAMENTO

Struttura della tabella *PAGAMENTO*:

```
CREATE TABLE IF NOT EXISTS ‘PAGAMENTO‘ (
        ‘Id‘ INT UNSIGNED NOT NULL AUTO_INCREMENT,
        ‘Id_F‘ INT UNSIGNED NOT NULL,
        ‘Tipo‘ VARCHAR(50) NOT NULL,
        ‘Data_pagamento‘ TIMESTAMP NOT NULL,
        ‘Importo‘ DOUBLE(8,2) UNSIGNED NOT NULL DEFAULT ’0.00’,
        PRIMARY KEY (‘Id‘),
        FOREIGN KEY (‘Id_F‘) REFERENCES FATTURA(‘Id‘)
) ENGINE = InnoDB AUTO_INCREMENT=1;
```

5.1.5 ORDINE

Struttura della tabella *ORDINE*:

```
CREATE TABLE IF NOT EXISTS ‘ORDINE‘ (
        ‘Id‘ INT UNSIGNED NOT NULL AUTO_INCREMENT,
        ‘Id_P‘ INT UNSIGNED NOT NULL,
        ‘Email_UA‘ VARCHAR(100) NOT NULL,
        ‘Email_UR‘ VARCHAR(100) NOT NULL,
        ‘Data‘ TIMESTAMP NOT NULL,
        PRIMARY KEY (‘Id‘),
        FOREIGN KEY (‘Email_UA‘) REFERENCES UTENTE_AMMINISTRATORE(‘Email
            ‘),
        FOREIGN KEY (‘Email_UR‘) REFERENCES UTENTE_REGISTRATO(‘Email‘),
        FOREIGN KEY (‘Id_P‘) REFERENCES PAGAMENTO(‘Id‘)
) ENGINE = InnoDB AUTO_INCREMENT=1;
```

5.1.6 TRACKING

Struttura della tabella *TRACKING*:

```
CREATE TABLE IF NOT EXISTS `TRACKING` (
        `Id_O` INT UNSIGNED NOT NULL,
        `Stato` VARCHAR(100) NOT NULL,
        `Data` TIMESTAMP NOT NULL,
        PRIMARY KEY (`Id_O`),
        FOREIGN KEY (`Id_O`) REFERENCES ORDINE(`Id`)
) ENGINE = InnoDB;
```

5.1.7 INDIRIZZO_DI_SPEDIZIONE

Struttura della tabella *INDIRIZZO_DI_SPEDIZIONE*:

```
CREATE TABLE IF NOT EXISTS `INDIRIZZO_DI_SPEDIZIONE` (
        `Id_O` INT UNSIGNED NOT NULL,
        `Tipo` VARCHAR(20) NOT NULL,
        `Costo` DOUBLE(8,2) UNSIGNED NOT NULL DEFAULT '0.00',
        `Nome_e_cognome` VARCHAR(100) NOT NULL,
        `Nazione` VARCHAR(50) NOT NULL,
        `Provincia` VARCHAR(50) NOT NULL,
        `Citta` VARCHAR(50) NOT NULL,
        `CAP` VARCHAR(5) NOT NULL,
        `Via` VARCHAR(100) NOT NULL,
        `Numero_civico` VARCHAR(10) NOT NULL,
        `Numero_di_telefono` VARCHAR(30) NOT NULL,
        `Note` TEXT(1000),
        PRIMARY KEY (`Id_O`),
        FOREIGN KEY (`Id_O`) REFERENCES ORDINE(`Id`)
) ENGINE = InnoDB;
```

5.1.8 LISTA_DELLA_SPESA

Struttura della tabella *LISTA_DELLA_SPESA*:

```
CREATE TABLE IF NOT EXISTS `LISTA_DELLA_SPESA` (
        `Email_UR` VARCHAR(100) NOT NULL,
        `Nome` VARCHAR(50) NOT NULL,
        `Note` TEXT(1000),
        PRIMARY KEY (`Email_UR`, `Nome`),
        FOREIGN KEY (`Email_UR`) REFERENCES UTENTE_REGISTRATO(`Email`)
) ENGINE = InnoDB;
```

5.1.9 INDIRIZZO

Struttura della tabella *INDIRIZZO*:

```
CREATE TABLE IF NOT EXISTS `INDIRIZZO` (
        `Email_UR` VARCHAR(100) NOT NULL,
        `Nome` VARCHAR(50) NOT NULL,
        `Nazione` VARCHAR(50) NOT NULL,
        `Provincia` VARCHAR(50) NOT NULL,
        `Citta` VARCHAR(50) NOT NULL,
        `CAP` VARCHAR(5) NOT NULL,
        `Via` VARCHAR(100) NOT NULL,
        `Numero_civico` VARCHAR(10) NOT NULL,
        `Numero_di_telefono` VARCHAR(30) NOT NULL,
        PRIMARY KEY (`Email_UR`, `Nome`),
        FOREIGN KEY (`Email_UR`) REFERENCES UTENTE_REGISTRATO(`Email`)
) ENGINE = InnoDB;
```

5.1.10 RICETTA

Struttura della tabella *RICETTA*:

```
CREATE TABLE IF NOT EXISTS `RICETTA` (
        `Nome` VARCHAR(50) NOT NULL,
        `Autore` VARCHAR(50) NOT NULL,
        `Email_UA` VARCHAR(50) NOT NULL,
        `Tempo_di_preparazione` TIME NOT NULL,
        `Preparazione` TEXT(22000) NOT NULL,
        `Attiva` BOOLEAN NOT NULL DEFAULT TRUE,
        PRIMARY KEY (`Nome`, `Autore`),
        FOREIGN KEY (`Email_UA`) REFERENCES UTENTE_AMMINISTRATORE(`Email
            `)
) ENGINE = InnoDB;
```

5.1.11 MARCHIO

Struttura della tabella *MARCHIO*:

```
CREATE TABLE IF NOT EXISTS `MARCHIO` (
        `Nome` VARCHAR(50) NOT NULL,
        `Email_UA` VARCHAR(50) NOT NULL,
        `Descrizione` TEXT(2000) NOT NULL,
        `Immagine` VARCHAR(100) NOT NULL,
        PRIMARY KEY (`Nome`),
        FOREIGN KEY (`Email_UA`) REFERENCES UTENTE_AMMINISTRATORE(`Email
            `)
) ENGINE = InnoDB;
```

5.1.12 CARATTERISTICA

Struttura della tabella *CARATTERISTICA*:

```
CREATE TABLE IF NOT EXISTS ‘CARATTERISTICA‘ (
        ‘Nome‘ VARCHAR(50) NOT NULL,
        ‘Email_UA‘ VARCHAR(50) NOT NULL,
        ‘Descrizione‘ TEXT(2000) NOT NULL,
        PRIMARY KEY (‘Nome‘),
        FOREIGN KEY (‘Email_UA‘) REFERENCES UTENTE_AMMINISTRATORE(‘Email
            ‘)
) ENGINE = InnoDB;
```

5.1.13 REPARTO

Struttura della tabella *REPARTO*:

```
CREATE TABLE IF NOT EXISTS ‘REPARTO‘ (
        ‘Nome‘ VARCHAR(50) NOT NULL,
        ‘Email_UA‘ VARCHAR(50) NOT NULL,
        ‘Descrizione‘ TEXT(2000) NOT NULL,
        PRIMARY KEY (‘Nome‘),
        FOREIGN KEY (‘Email_UA‘) REFERENCES UTENTE_AMMINISTRATORE(‘Email
            ‘)
) ENGINE = InnoDB;
```

5.1.14 COUPON

Struttura della tabella *COUPON*:

```
CREATE TABLE IF NOT EXISTS ‘COUPON‘ (
        ‘Id‘ INT UNSIGNED NOT NULL,
        ‘Email_UA‘ VARCHAR(50) NOT NULL,
        ‘Email_UR‘ VARCHAR(50) NOT NULL,
        ‘Ente_erogatore‘ VARCHAR(50) NOT NULL,
        ‘Importo‘ DOUBLE(8,2) NOT NULL DEFAULT ’0.00’,
        PRIMARY KEY (‘Id‘),
        FOREIGN KEY (‘Email_UA‘) REFERENCES UTENTE_AMMINISTRATORE(‘Email
            ‘),
        FOREIGN KEY (‘Email_UR‘) REFERENCES UTENTE_REGISTRATO(‘Email‘)
) ENGINE = InnoDB;
```

5.1.15 PRODOTTO

Struttura della tabella *PRODOTTO*:

```
CREATE TABLE IF NOT EXISTS ‘PRODOTTO‘ (
        ‘EAN‘ VARCHAR(13) NOT NULL,
        ‘Nome_M‘ VARCHAR(50) NOT NULL,
        ‘Nome_R‘ VARCHAR(50) NOT NULL,
        ‘Data_inserimento‘ TIMESTAMP NOT NULL,
        ‘Nome‘ VARCHAR(100) NOT NULL,
        ‘Descrizione‘ TEXT(2000) NOT NULL,
        ‘Quantita‘ INT UNSIGNED NOT NULL,
        ‘Peso‘ INT UNSIGNED NOT NULL,
        ‘Prezzo_unitario‘ DOUBLE(8,2) UNSIGNED NOT NULL DEFAULT ’0.00’,
        ‘Immagine‘ VARCHAR(100) NOT NULL,
        ‘Attivo‘ BOOLEAN NOT NULL DEFAULT TRUE,
        PRIMARY KEY (‘EAN‘),
        FOREIGN KEY (‘Nome_M‘) REFERENCES MARCHIO(‘Nome‘),
        FOREIGN KEY (‘Nome_R‘) REFERENCES RICETTA(‘Nome‘)
) ENGINE = InnoDB;
```

5.1.16 SCADENZA

Struttura della tabella *SCADENZA*:

```
CREATE TABLE IF NOT EXISTS ‘SCADENZA‘ (
        ‘EAN_P‘ VARCHAR(13) NOT NULL,
        ‘Quantita‘ INT UNSIGNED NOT NULL,
        ‘Data_di_scadenza‘ DATE NOT NULL,
        PRIMARY KEY (‘EAN_P‘),
        FOREIGN KEY (‘EAN_P‘) REFERENCES PRODOTTO(‘EAN‘)
) ENGINE = InnoDB;
```

5.1.17 HA_PARTICOLARE

Struttura della tabella *HA_PARTICOLARE*:

```
CREATE TABLE IF NOT EXISTS ‘HA_PARTICOLARE‘ (
        ‘EAN_P‘ VARCHAR(13) NOT NULL,
        ‘Nome_C‘ VARCHAR(50) NOT NULL,
        PRIMARY KEY (‘EAN_P‘, ‘Nome_C‘),
        FOREIGN KEY (‘EAN_P‘) REFERENCES PRODOTTO(‘EAN‘),
        FOREIGN KEY (‘Nome_C‘) REFERENCES CARATTERISTICA(‘Nome‘)
) ENGINE = InnoDB;
```

5.1.18 IMPIEGATO_IN

Struttura della tabella *IMPIEGATO_IN*:

```
CREATE TABLE IF NOT EXISTS `IMPIEGATO_IN` (
        `EAN_P` VARCHAR(13) NOT NULL,
        `Nome_R` VARCHAR(50) NOT NULL,
        `Autore_R` VARCHAR(100) NOT NULL,
        `Quantita` INT UNSIGNED NOT NULL,
        PRIMARY KEY (`EAN_P`, `Nome_R`, `Autore_R`),
        FOREIGN KEY (`EAN_P`) REFERENCES PRODOTTO(`EAN`),
        FOREIGN KEY (`Nome_R`,`Autore_R`) REFERENCES RICETTA(`Nome`,`
            Autore`)
) ENGINE = InnoDB;
```

5.1.19 CONTENUTO_IN_LISTA

Struttura della tabella *CONTENUTO_IN_LISTA*:

```
CREATE TABLE IF NOT EXISTS `CONTENUTO_IN_LISTA` (
        `EAN_P` VARCHAR(13) NOT NULL,
        `Email_UR` VARCHAR(100) NOT NULL,
        `Nome_L` VARCHAR(50) NOT NULL,
        `Quantita` INT UNSIGNED NOT NULL,
        PRIMARY KEY (`EAN_P`, `Email_UR`, `Nome_L`),
        FOREIGN KEY (`EAN_P`) REFERENCES PRODOTTO(`EAN`),
        FOREIGN KEY (`Email_UR`,`Nome_L`) REFERENCES LISTA_DELLA_SPESA(`
            Email_UR`,`Nome`)
) ENGINE = InnoDB;
```

5.1.20 CONTENUTO_IN_CARRELLO

Struttura della tabella *CONTENUTO_IN_CARRELLO*:

```
CREATE TABLE IF NOT EXISTS `CONTENUTO_IN_CARRELLO` (
        `EAN_P` VARCHAR(13) NOT NULL,
        `Email_UR` VARCHAR(100) NOT NULL,
        `Quantita` INT UNSIGNED NOT NULL,
        PRIMARY KEY (`EAN_P`, `Email_UR`),
        FOREIGN KEY (`EAN_P`) REFERENCES PRODOTTO(`EAN`),
        FOREIGN KEY (`Email_UR`) REFERENCES UTENTE_REGISTRATO(`Email`)
) ENGINE = InnoDB;
```

5.1.21 COMPONE

Struttura della tabella *COMPONE*:

```
CREATE TABLE IF NOT EXISTS `COMPONE` (
        `EAN_P` VARCHAR(13) NOT NULL,
        `Id_O` INT UNSIGNED NOT NULL,
        `Quantita` INT UNSIGNED NOT NULL,
        PRIMARY KEY (`EAN_P`, `Id_O`),
        FOREIGN KEY (`EAN_P`) REFERENCES PRODOTTO(`EAN`),
        FOREIGN KEY (`Id_O`) REFERENCES ORDINE(`Id`)
) ENGINE = InnoDB;
```

5.1.22 GESTISCE_PRODOTTI

Struttura della tabella *GESTISCE_PRODOTTI*:

```
CREATE TABLE IF NOT EXISTS `GESTISCE_PRODOTTI` (
        `EAN_P` VARCHAR(13) NOT NULL,
        `Email_UA` VARCHAR(100) NOT NULL,
        PRIMARY KEY (`EAN_P`, `Email_UA`),
        FOREIGN KEY (`EAN_P`) REFERENCES PRODOTTO(`EAN`),
        FOREIGN KEY (`Email_UA`) REFERENCES UTENTE_AMMINISTRATORE(`Email
            `)
) ENGINE = InnoDB;
```

5.2 INTERROGAZIONI

Tra le molte interrogazioni utilizzate dall'applicazione web riguardo il database precedente si riportano quelle che meritano un commento.

6 INTERFACCIA GRAFICA

6.1 ELENCO ENTITÀ

bla bla bla...