

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI E SISTEMI INFORMATIVI I

Progettazione e sviluppo di una base di dati relazionale per un “sistema di gestione per le consegne a domicilio”

Autore:

Fabiola SALOMONE

MATRICOLA: N86/2870

fab.salomone@studenti.unina.it

Docenti:

Prof. Adriano PERON

Prof. Silvio BARRA

Indice

Capitolo I – Descrizione del progetto	5
1.1 Introduzione	5
1.2 Risultati	6
1.2.1 – Schermata Login	6
1.2.2 – Schermata Registrazione	6
1.2.3 – Schermata Password dimenticata	7
1.2.4 – Schermata Scelta Ristorante-Veicolo	7
1.2.5 – Schermata Menù	8
1.2.6 – Schermata ordine	8
Capitolo II – Progettazione Concettuale	9
2.1 Class diagram	9
2.2 Class diagram ristrutturato	10
2.2.1 Analisi delle ridondanze	10
2.2.2 Eliminazione delle generalizzazioni	10
2.2.3 Eliminazione Attributi Multivalore	10
2.2.4 Eliminazione Attributi Strutturati	11
2.2.5 Partizionamento/accoppiamento di entità e associazioni	11
2.2.6 Scelta degli identificatori primari	11
2.3 Dizionari	13
2.3.1 Dizionario delle classi	13
2.3.2 Dizionario delle associazioni	15
2.3.3 Dizionario dei vincoli	17
Capitolo III – Progettazione Logica	18
3.1 Schema Logico	18
Capitolo IV – Progettazione Fisica	19
4.1 Creazione SEQUENCES	19
4.1.1 Sequences	19
4.2 Definizioni delle tabelle e vincoli semplici	23

4.1.2 Definizione della tabella indirizzo	23
4.1.2 Definizione della tabella utente	24
4.1.3 Definizione della tabella ristorante	24
4.1.4 Definizione enumerazione TIPO_PRODOTTO	25
4.1.5 Definizione enumerazione CATEGORIA_PRODOTTO	25
4.1.6 Definizione tabella prodotto	26
4.1.7 Definizione tabella fornitura	26
4.1.8 Definizione tabella rider	27
4.1.9 Definizione enumerazione TIPO_VEICOLO	27
4.1.10 Definizione tabella veicolo	28
4.1.11 Definizione tabella corrierePer	28
4.1.12 Definizione tabella carrello	29
4.1.13 Definizione tabella composizioneCarrello	29
4.1.14 Definizione enumerazione STATO_ORDINE	30
4.1.15 Definizione tabella ordine	30
4.2 Viste	31
4.2.1 riderVeicoloView	31
4.3 Definizione Trigger	31
4.3.1 Definizione Trigger verifico_email_password	31
4.3.2 Definizione Trigger controllo_quantitaProdotto	32
4.3.3 Definizione Trigger modifica_password	33
4.3.4 Definizione Trigger controllo_attivitarider	33
4.3.5 Definizione Trigger controllo_statoOrdine	34
4.3.6 Definizione Trigger controllo_costoTotaleOrdine	34
4.4 Definizione Procedure	35
4.4.1 Definizione Procedura inserisci_indirizzo	35
4.4.2 Definizione Procedura inserisci_utente	35
4.4.3 Definizione Procedura inserisci_ristorante	36
4.4.4 Definizione Procedura inserisci_prodotto	36
4.4.5 Definizione Procedura inserisci_fornitura	37

4.4.6 Definizione Procedura inserisci_rider	37
4.4.7 Definizione Procedura inserisci_veicolo	38
4.4.8 Definizione Procedura inserisci_carrello	38
4.4.9 Definizione Procedura inserisci_ordine	38
4.4.10 Definizione Procedura cancella_indirizzo	39
4.4.11 Definizione Procedura cancella_utente	39
4.4.12 Definizione Procedura cancella_ristorante	39
4.4.13 Definizione Procedura cancella_prodotto	40
4.4.14 Definizione Procedura cancella_fornitura	40
4.4.15 Definizione Procedura cancella_rider	40
4.4.16 Definizione Procedura cancella_veicolo	41
4.4.17 Definizione Procedura cancella_carrello	41
4.4.18 Definizione Procedura cancella_ordine	41
4.4.19 Definizione Procedura aggiorna_indirizzo	42
4.4.20 Definizione Procedura aggiorna_utente	42
4.4.21 Definizione Procedura aggiorna_ristorante	43
4.4.22 Definizione Procedura aggiorna_veicolo	43
4.4.23 Definizione Procedura aggiorna_rider	44
4.4.24 Definizione Procedura aggiorna_prodotto	44
4.4.25 Definizione Procedura aggiorna_fornitura	45
4.4.26 Definizione Procedura aggiorna_carrello	45
4.4.27 Definizione Procedura aggiorna_ordine	45
4.5 Popolamento con dati di esempio	47
Capitolo V – Glossario	51

Capitolo I – Descrizione del progetto

Descrizione Progetto e Risultato

1.1 Introduzione

Dalle specifiche dettagliate dal committente, sono emerse le seguenti richieste:

Si vuole fornire una base di dati che consente di centralizzare una serie di servizi per la gestione di una piattaforma di consegne a domicilio.

Per quanto riguarda la gestione dei dati di interesse del sistema, si vuole progettare una base di dati atta a contenere le informazioni relative a:

1. Ristorante, compreso di nome, numero Telefonico, indirizzo e una breve descrizione;
2. Pietanze con relativo nome, prezzo, tipo di prodotto, codice seriale, e categoria della pietanza;
3. Utenti, con nome, cognome, email, password, indirizzo, numero Telefonico;
4. Rider, con nome, cognome, biografia, numero di attività ad esso associate;
5. Veicoli, con marca, modello, anno di immatricolazione;
6. Ordinanze, con rispettivo prezzo, data dell'acquisto, e stato, che indica se l'ordine è stato consegnato con successo o meno;
7. Carrello, con data e codice carrello.

In particolare viene richiesto di tener traccia:

- o Delle consegne effettuate;
- o Del negozio da cui sono partite le consegne;
- o Degli utenti che hanno effettuato le ordinazioni;
- o Dei Rider se ha completato o no le consegne.

Si è progettata ed implementata una base di *dati relazionali*^[1], relativa alle consegne a domicilio di cibo, denominata Food.

La piattaforma consentirà agli utenti di:

1. potersi registrare e successivamente effettuare il login;
2. selezionare il ristorante da cui si desidera fare acquisti ed impostare il tipo di veicolo che il rider dovrà utilizzare per effettuare la consegna
3. selezionare i prodotti da acquistare divisi per categoria, tra cui:
 - Alcolico
 - Analcolico
 - Antipasti

- Primi
- Secondi
- Contorni
- Dolci

4. Scegliere i prodotti da acquistare e le relative quantità;
5. Scegliere i prodotti da acquistare in base ad un filtraggio per prezzo;
6. Di aggiungere il prodotto/pietanza all'ordine;
7. Di concludere l'ordine.

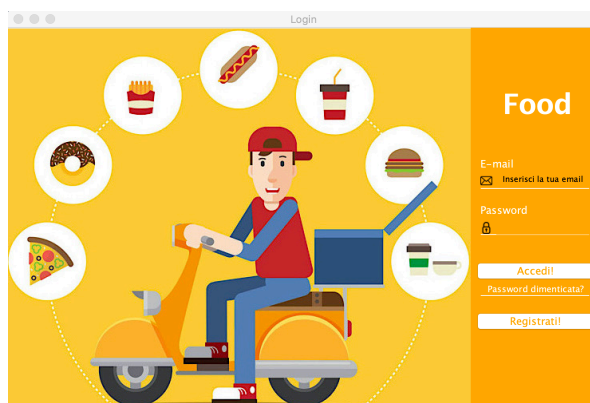
Per finire, il sistema mostrerà un'interfaccia che porterà alla vera e propria conclusione dell'ordine.

Inoltre il sistema dovrà rispettare un particolare vincolo ossia che ogni rider potrà essere associato solo a massimo a tre attività (ristoranti).

Per la realizzazione della piattaforma si è usati, l'ambiente di sviluppo *Eclipse* in collaborazione con un *DBMS*^[2] relazionale ad oggetti, *Postgres*, Version 2.4.2. Il DBMS è stato realizzato attraverso l'utilizzo dell'interfaccia grafica *pgAdmin4*.

1.2 Risultati

1.2.1 – Schermata Login

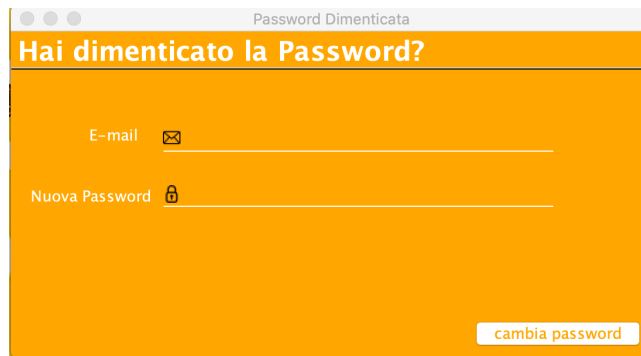


Il “*Login*” è l'interfaccia principale del sistema, dove l'utente può tramite un form registrarsi al servizio e di conseguenza accedere per effettuare acquisti. L'interfaccia dispone di due pulsanti: uno conduce ad una nuova interfaccia per il recupero di password e l'altro ad una nuova interfaccia per la registrazione.

1.2.2 – Schermata Registrazione

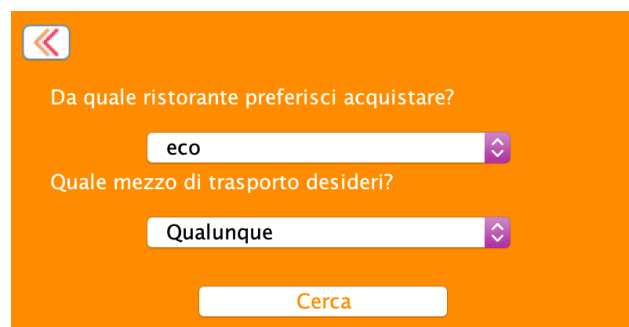
L'interfaccia "*Iscrizione*" dà la possibilità di registrarsi al sistema. È un form, in cui l'utente deve inserire le sue informazioni personali e la password da utilizzare per accedere al sistema. Quando viene cliccato il pulsante 'Iscriviti' si procede alla registrazione dell'utente, la quale consiste nell'inserire i dati immessi dall'utente in una tabella riservata del database. I dati, prima di essere immessi verranno controllati.

1.2.3 – Schermata Password dimenticata



L'interfaccia "*Password Dimenticata*" dà la possibilità di modificare la password, senza registrarsi nuovamente, nel momento in cui l'utente la smarrisce o la dimentica. Nel form l'utente dovrà inserire l'email inserita all'atto della registrazione e la nuova password. Quando viene cliccato il pulsante 'cambia password' si procede all'aggiornamento della password, che consiste nell'inserire la nuova password relativa alla specifica email in una tabella riservata del database inserita sul form. I dati, ovviamente, prima di essere immessi verranno controllati.

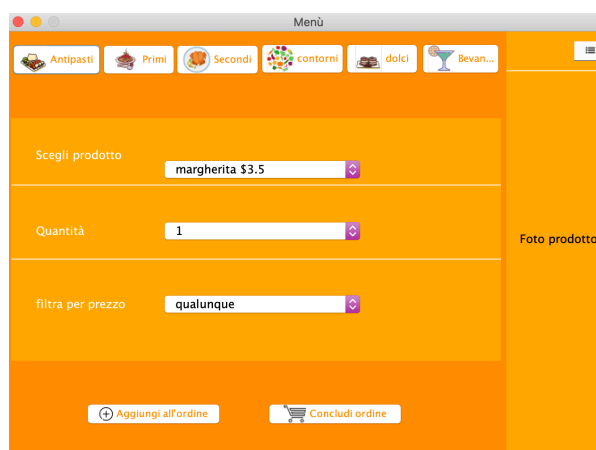
1.2.4 – Schermata Scelta Ristorante-Veicolo



Tale interfaccia si presenta all'utente dopo che ha effettuato l'accesso alla piattaforma.

In questa interfaccia all'utente verrà fatto scegliere il ristorante da cui acquistare e il veicolo che preferisce che il rider dovrà utilizzare per la consegna

1.2.5 – Schermata Menù



L'interfaccia “Menu” si presenta all'utente dopo scelto il ristorante e il veicolo.

In questa interfaccia utente potrà scegliere:

- La categoria da cui acquistare pietanze;
- Il prodotto;
- La quantità.

Inoltre potrà effettuare un filtraggio per prezzo e proseguire o aggiungendo altri prodotti all'ordine oppure concludendolo.

1.2.6 – Schermata ordine



L'interfaccia “Dettagli fattura” dà la possibilità di concludere l'ordine ma offre anche la possibilità di far scegliere all'utente il rider che preferisce per la consegna. Come si può vedere si tratta di un form pre-compilato dove si è fatto in modo che le informazioni associate al profilo del particolare utente vengano recuperate dal database ed inserite in modo automatico dal sistema. Inoltre, come possiamo vedere, viene mostrato un carrello con tutti i prodotti selezionati dall'utente, ciò è stato creato facendo in modo che i prodotti selezionati si memorizzino in una tabella riservata del database e poi vengano mostrati nuovamente dal sistema. Infine viene data la possibilità all'utente di annullare tutto oppure effettuare l'operazione. Quando viene cliccato il pulsante ‘Effettua Ordine’ si procede col inserire l'ordine in una tabella riservata del database e sarà compito di chi possiede la base di dati ad indicare nel database se l'ordine è stato effettuato oppure no.

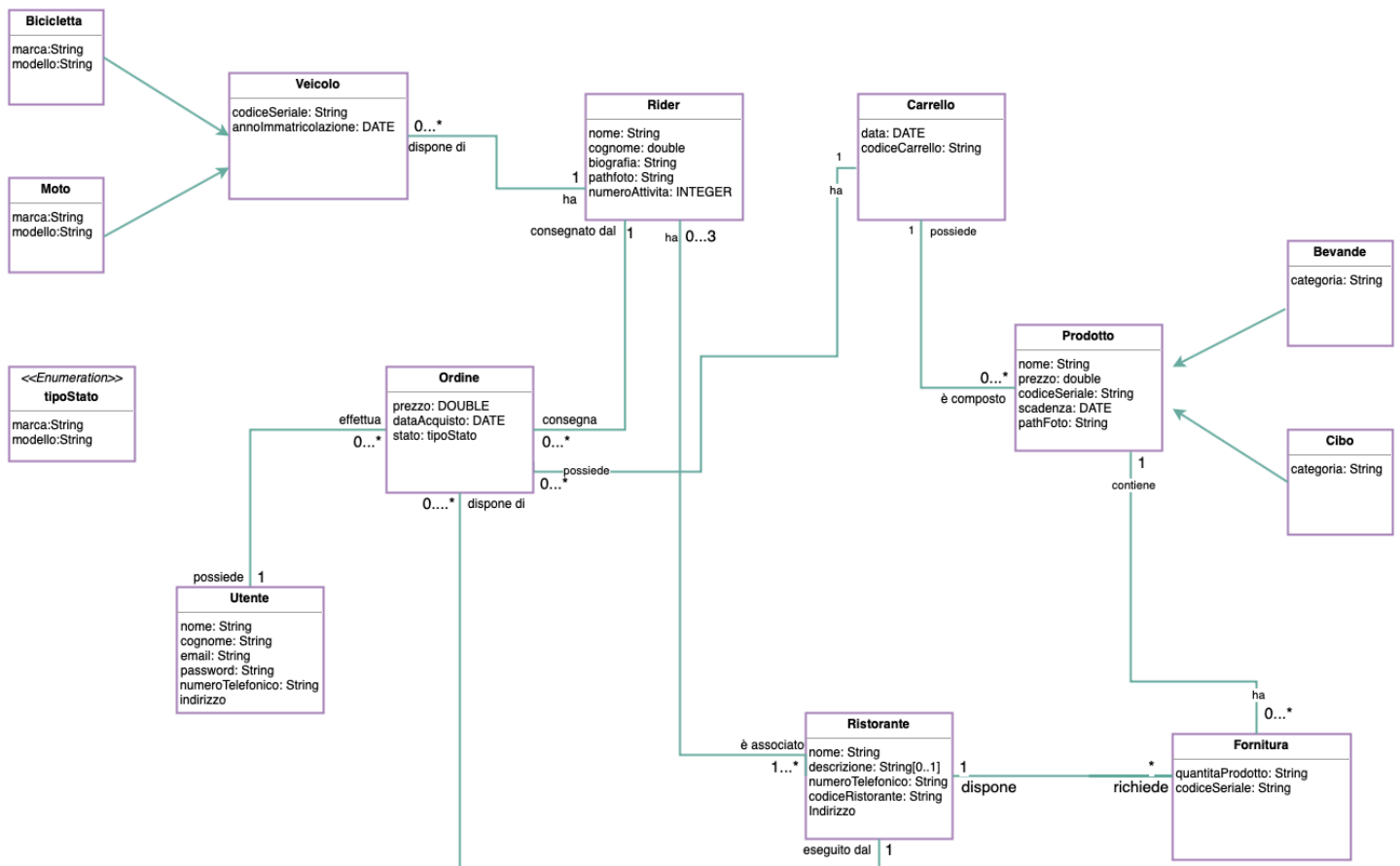
Capitolo II – Progettazione Concettuale

Progettazione Concettuale

In questo capitolo sarà trattata la fase di progettazione della base di dati, dove partendo dal risultato dell'analisi dei requisiti si arriverà ad uno *schema concettuale*^[3], che rappresenterà, usando un *Class Diagram UML*^[4], le entità rilevanti e le relazioni che intercorrono tra esse.

2.1 Class diagram

Viene ora indicato il *Class diagram UML* relativo alla piattaforma:



2.2 Class diagram ristrutturato

Viene ora indicato il Class diagram UML ristrutturato relativo alla piattaforma, dove l'obiettivo sarà quello di rendere lo schema efficiente per l'implementazione prefiggendo alcuni scopi:

- Analisi delle ridondanze
- Eliminazione delle generalizzazioni
- Eliminazione Attributi Multivalore
- Eliminazione Attributi Strutturati
- Partizionamento/accoppiamento di entità e associazioni
- Scelta degli identificatori primari

2.2.1 *Analisi delle ridondanze*

Consiste nell'esaminare tutti gli attributi di tutte le entità e associazioni e verificare se alcuni di essi sono ridondanti o sono ricavabili in funzione di altri attributi.

Questa prima fase del lavoro risulta abbastanza semplice in quanto già nella prima fase si erano scelti una serie di attributi non ridondanti. Infatti, non essendoci ancora una precisa idea dei carichi in gioco, nonché delle particolari query richieste dal sistema, non si è ritenuto di dover aggiungere attributi al solo fine di ottenere una futura efficienza in alcune operazioni.

2.2.2 *Eliminazione delle generalizzazioni*

Le gerarchie di generalizzazioni non sono un costrutto previsto dai database relazionali. È necessario perciò una loro traduzione in classi e associazioni. Si presentano in generale tre alternative:

1. Unificare il padre della generalizzazione nei figli. In questo caso gli attributi e le associazioni relative al padre vengono duplicati per tutti i figli.
2. Unificare i figli della generalizzazione nel padre. In questo caso gli attributi di tutti i figli vengono aggiunti al padre, così come le relazioni. Inoltre deve essere aggiunto nel padre un nuovo attributo che identifichi il tipo del figlio coinvolto in ogni tupla.
3. Introdurre un'ulteriore relazione, con cardinalità uno a uno dal padre verso ogni figlio.

Per eliminare le gerarchie presenti nel modello si è dovuto applicare la seconda alternativa, in particolare:

- Nel modello iniziale vi era la gerarchia Prodotto con cibo e bevande, ora invece vi è un attributo 'tipoProdotto' che può assumere solo uno dei valori indicati nell'enumerazione ossia o cibo o bevanda.
- Nel modello iniziale vi era la gerarchia veicolo con moto e bicicletta, ora invece vi è un attributo 'tipoVeicolo' che può assumere solo uno dei valori indicati nell'enumerazione ossia o moto o bicicletta

2.2.3 *Eliminazione Attributi Multivalore*

Questa fase del lavoro risulta abbastanza semplice in quanto già nella prima fase si erano scelti una serie di attributi senza valore multiplo.

2.2.4 Eliminazione Attributi Strutturati

Risulta conveniente ai fini dell'efficienza andare a modificare gli attributi strutturati con delle entità. Ciò viene effettuato per il campo 'Indirizzo' che è formato da più attributi. Si è proceduto con la creazione di una nuova entità denominata 'Indirizzo' contenente: 'nomeVia', 'numeroCivico', 'CAP', 'città', 'provincia' e si è attribuita una relazione 1 a molti tra l'entità 'Indirizzo' e 'Utente' e una relazione uno a uno tra l'entità 'Indirizzo' e 'Ristorante', per esplicitare che:

1. Utente - Indirizzo:

- L'utente possiede al più un solo CAP, un solo numero civico, un solo nome via, una sola città e una sola provincia;
- L'indirizzo possiede più utenti.

2. Indirizzo - Ristorante:

- Il Ristorante possiede al più un solo CAP, un solo numero civico, un solo nome via, una sola città e una sola provincia;
- L'indirizzo possiede al più un solo Ristorante

2.2.5 Partizionamento/accoppiamento di entità e associazioni

Questa fase del lavoro risulta abbastanza semplice in quanto non si è necessitati di effettuare operazioni di Partizionamento/accoppiamento di entità e associazioni.

2.2.6 Scelta degli identificatori primari

Risulta conveniente ai fini dell'efficienza in fase di implementazione l'introduzione di alcune chiavi "tecniche" in ogni entità. Tali chiavi tecniche altro non saranno che identificativi numerici che permetteranno di discriminare con maggiore facilità le istanze.

2.3 Dizionari

2.3.1 Dizionario delle classi

CLASSE	DESCRIZIONE	ATTRIBUTI
Utente	Descrive tutti gli utenti registrati alla piattaforma	nome (1,1) [VARCHAR(32)] : nome utente cognome (1,1) [VARCHAR(32)] : cognome dell'utente email (1,*) [VARCHAR(64)] : email dell'utente password (1,*) [VARCHAR(32)] : password dell'utente numeroTelefonico (1,*) [CHAR(10)] : numero telefonico dell'utente
Indirizzo	Descrive l'indirizzo degli utenti e del ristorante	nomeVia (1,1) [VARCHAR(32)] : nome della via in cui risiede l'utente o il ristorante numeroCivico (1,1) [INTEGER] : numero civico in cui risiede l'utente o il ristorante CAP (1,1) [CHAR(5)] : CAP dell'utente o del ristorante citta (1,1) [VARCHAR(64)] : citta in cui risiede l'utente o il ristorante provincia (1,1) [CHAR(2)] : provincia in cui risiede l'utente o il ristorante codiceIndirizzo (0,N) con $N \geq 1$ [INTEGER] : identifica univocamente un indirizzo
Ristorante	Descrive il ristorante da cui l'utente potrà effettuare ordinazioni	nome (1,1) [VARCHAR(32)] : nome del ristorante descrizione (1,1) [VARCHAR(32)] : breve descrizione del ristorante numeroTelefonico (1,1) [CHAR(10)] : numero telefonico del ristorante codiceRistorante (0,N) con $N \geq 1$ [INTEGER] : identifica univocamente il ristorante Consegna (0,1) [VARCHAR(2)]: indica se la consegna è stata effettuata
Fornitura	Descrive una serie di prodotti necessari per lo svolgimento dell'attività	quantitaProdotto (0...*) [INTEGER] : quantità di un prodotto
Ordine	Descrive l'ordine	prezzo (1,1) : [DOUBLE] : prezzo totale dataAcquisto (1,1) [DATE] : data in cui è stato effettuato l'ordine Stato (1,1) [BOOLEAN] : indica se uno stato è arrivato, non arrivato o non consegnato.

CLASSE	DESCRIZIONE	ATTRIBUTI
Prodotto	Descrive i prodotti disponibili	nome (1,1) [VARCHAR(32)] : nome del prodotto prezzo (1,1) [REAL] : prezzo del prodotto codiceSeriale (0,N) con N \geq 1 [SERIAL] : identifica univocamente un prodotto scadenza (1,1) [DATE] : data di scadenza del prodotto pathFoto (1,1) [VARCHAR(64)] : foto del prodotto tipoProdotto (0,1) [TIPO_PRODOTTO] : indica il tipo del prodotto categoria (0,1) [CATEGORIA_TIPO] : indica la categoria del prodotto
Rider	Descrive il rider ossia colui che dovrà effettuare le consegne agli utenti	nome (1,1) [VARCHAR(32)] : nome del rider cognome (1,1) [VARCHAR(32)] : cognome del rider biografia (1,1) [VARCHAR(512)] : biografia del rider pathFoto (1,1) [VARCHAR(32)] : foto del rider numeroAttività (0,3) [INTEGER] : indica il numero di attività svolte dal rider
Veicolo	Descrive i veicoli	annoImmatricolazione (0,N) con N \geq 1: [DATE]: anno di immatricolazione del veicolo marca (1,1) : [VARCHAR(32)]: marca del veicolo modello (1,1) : [VARCHAR(32)]: modello del veicolo tipoVeicolo (0,1) : [TIPO_VEICOLO]: indica il tipo di veicolo

2.3.2 Dizionario delle associazioni

NOME	DESCRIZIONE	CLASSI COINVOLTE
Utente-Indirizzo	Esprime l'appartenenza dell'indirizzo ad un utente	Utente [0...*] ruolo possiede : indica l'utente a cui appartiene l'indirizzo Indirizzo [1] ruolo ha : indica l'indirizzo di un utente
Utente-Ordine	Esprime l'appartenenza dell'utente ad un ordine	Utente [1] ruolo possiede : indica l'utente a cui appartiene l'ordine Ordine [0...*] ruolo effettua : indica l'ordine effettuato dall'utente
Indirizzo-Ristorante	Esprime l'appartenenza di un Ristorante ad un indirizzo	Indirizzo [1] ruolo ha : indica l'indirizzo di un ristorante Ristorante [1] ruolo gode : indica le varie attività di appoggio della piattaforma
Ordine-Ristorante	Esprime l'appartenenza di un Ordine ad un Ristorante	Ordine [0...*] ruolo dispone : indica l'ordine che viene eseguito da un particolare ristorante Ristorante [1] ruolo eseguito : indica le varie attività di appoggio della piattaforma che effettuano l'ordine
Ordine-Rider	Esprime l'appartenenza di un Ordine ad un Rider	Ordine [0..*] ruolo consegna : indica l'ordine che viene consegnato Rider [1] ruolo consegnato : indica il rider che si occupa di consegnare l'ordine
Ordine-Carrello	Esprime l'appartenenza di un Ordine ad un Carrello	Ordine [0..*] ruolo possiede : indica l'ordine richiesto dall'utente Carrello [1] ruolo ha : indica il carrello di ogni ordine richiesto dall'utente
Veicolo-Rider	Esprime l'appartenenza di un Veicolo ad un Rider	Rider [1] ruolo ha : indica il rider a cui è associato il veicolo Veicolo [0...*] ruolo dispone : indica il veicolo associato al rider

NOME	DESCRIZIONE	CLASSI COINVOLTE
Rider-Ristorante	Esprime l'appartenenza di un Rider ad un Ristorante	<p>Rider [0...3] ruolo ha: indica il rider a cui è associato il ristorante</p> <p>Ristorante [1...*] ruolo è_associato: indica le varie attività di appoggio della piattaforma a cui è associato un rider</p>
Carrello-Prodotto	Esprime l'appartenenza di un prodotto ad un carrello	<p>Carrello [*] ruolo possiede: indica il contenitore che contiene i prodotti che l'utente desidera acquistare</p> <p>Prodotto [*] ruolo è_composto: indica il prodotto che l'utente desidera acquistare</p>
Prodotto-Fornitura	Esprime la quantità di un prodotto	<p>Prodotto [1] ruolo contiene: indica il prodotto che l'utente desidera acquistare</p> <p>Fornitura [0...*] ruolo ha: indica la quantità disponibile di un particolare prodotto nella piattaforma</p>
Ristorante-Fornitura	Esprime la quantità di un particolare prodotto nel ristorante	<p>Ristorante [1] ruolo dispone: indica le varie attività che mettono a disposizione le proprie pietanze nella piattaforma</p> <p>Fornitura [*] ruolo richiede: indica la quantità disponibile di un particolare prodotto nel ristorante</p>

2.3.3 Dizionario dei vincoli

VINCOLO	DESCRIZIONE
chk_nome	I nomi degli utenti devono contenere almeno un carattere da a-z o A-Z.
chk_numerotelefono	I numeri di telefono degli utenti possono contenere solo otto valori numerici da 0 a 9.
chk_cognome	I cognomi degli utenti devono contenere almeno un carattere maiuscolo da a-z o A-Z.
chk_cap	I CAP in cui risiedono gli utente devono contenere solo 5 valori numerici da 0 a 9.
chk_citta	I nomi di città in cui risiedono gli utente devono contenere almeno un carattere da a-z o A-Z.
chk_nomevia	I nomi di via in cui risiedono gli utente devono contenere almeno un carattere da a-z o A-Z.
chk_numerocivico	I numeri civici in cui risiedono gli utente devono contenere almeno un caratteri da a-z o A-Z oppure una cifra numerica da 0 a 9.
chk_provincia	Le province in cui risiedono gli utente, devono contenere almeno un carattere da a-z o A-Z.
riderVeicoloView (Eliminare)	La vista "riderVeicoloView" mostra per ciascun rider, il veicolo che gli appartiene.
chk_tipo_Prodotto	Un prodotto può essere di tipo 'bevanda' o 'cibo', ma non entrambi contemporaneamente.
chk_tipo_categoria	Una categoria può essere di tipo 'alcolico' o 'analcolico' o 'antipasti', 'primi' o 'secondi' o 'contorni' o 'dolci', ma non entrambi contemporaneamente.
chk_tipoVeicolo	Un veicolo può essere di tipo 'moto' o 'bicicletta', ma non entrambi contemporaneamente.
controllo_quantitaProdotto	Controlla che la quantità di un prodotto non deve essere minore o uguale a 0.
verifico_email_password	Gli indirizzi email degli utenti devono essere indirizzi email di forma legittima, ovvero deve includere almeno un carattere alfanumerici e simbolo prima della @, almeno un carattere alfanumerici e simbolo dopo la @, un punto e almeno un carattere. Inoltre la password deve contenere almeno una lettera maiuscola, almeno una lettera minuscola, almeno una cifra numerica, almeno un carattere speciale.
modifica_password	Aggiorna la vecchia password dell'utente con una nuova
controllo_attivitarider	Controlla che ogni rider sia fatto corrispondere a non più di tre ristoranti

Capitolo III – Progettazione Logica

Progettazione Logica

In questo capitolo è trattata la fase di progettazione della base di dati. Si tradurrà lo schema concettuale, dopo la ristrutturazione, in uno *schema logico*^[5]. Negli schemi relazionali che seguiranno le *chiavi primarie*^[6] sono indicate con una singola sottolineatura mentre le *chiavi esterne*^[7] con una doppia sottolineatura.

3.1 Schema Logico

Indirizzo (nomevia, numerocivico, Cap, città, provincia, codiceindirizzo)

Utente(nome, cognome, email, password, numeroTelefonico, codiceindirizzo)

Ristorante(nome, descrizione, numerotelefonico, codiceristorante, codiceindirizzo)

Prodotto(nome, prezzo, codiceseriale, scadenza, pathFoto, tipoprodotto, categoria)

Fornitura(quantitaprodotto, codiceseriale, codiceristorante)

Rider(nome, cognome, biografia, pathFoto, codicerider)

Veicolo(codiceseriale, annoimmatricolazione, marca, modello, tipoveicolo, codicerider)

CorrierePer(codiceristorante, codicerider)

Carrello(data, codiceCarrello)

Composizione(codicecarrello, codiceseriale, quantita)

Ordine(codicecarrello, email, codiceristorante, codicerider)

Capitolo IV – Progettazione Fisica

Progettazione Fisica

Seguono le definizioni delle tabelle estratte dallo script di creazione del database.

4.1 Creazione SEQUENCES

4.1.1 Sequences

```
1. /**
2.  * SEQUENCES: per alcuni attributi presenti nelle definizioni
3.  * delle tabelle
4.  */
5. CREATE SCHEMA IF NOT EXISTS food;
6.
7. SET search_path TO food;
8.
9. --CREAZIONE food.carrello_codicecarrello_seq
10. CREATE SEQUENCE food.carrello_codicecarrello_seq
11.     INCREMENT 1
12.     START 1
13.     MINVALUE 1
14.     MAXVALUE 2147483647
15.     CACHE 1;
16.
17. --CREAZIONE food.composizionecarrello_codicecarrello_seq
18. CREATE SEQUENCE food.composizionecarrello_codicecarrello_seq
19.     INCREMENT 1
20.     START 1
21.     MINVALUE 1
22.     MAXVALUE 2147483647
23.     CACHE 1;
24.
25. --CREAZIONE food.composizionecarrello_codiceseriale_seq
26. CREATE SEQUENCE food.composizionecarrello_codiceseriale_seq
27.     INCREMENT 1
28.     START 1
29.     MINVALUE 1
30.     MAXVALUE 2147483647
31.     CACHE 1;
32.
33. --CREAZIONE food.corriereper_codicerider_seq
34. CREATE SEQUENCE food.corriereper_codicerider_seq
```

```

35.      INCREMENT 1
36.      START 1
37.      MINVALUE 1
38.      MAXVALUE 2147483647
39.      CACHE 1;
40.
41. --CREAZIONE food.corriereper_codiceristorante_seq
42. CREATE SEQUENCE food.corriereper_codiceristorante_seq
43.      INCREMENT 1
44.      START 1
45.      MINVALUE 1
46.      MAXVALUE 2147483647
47.      CACHE 1;
48.
49. --CREAZIONE food.fornitura_codiceristorante_seq
50. CREATE SEQUENCE food.fornitura_codiceristorante_seq
51.      INCREMENT 1
52.      START 1
53.      MINVALUE 1
54.      MAXVALUE 2147483647
55.      CACHE 1;
56.
57. --CREAZIONE food.fornitura_codiceseriale_seq
58. CREATE SEQUENCE food.fornitura_codiceseriale_seq
59.      INCREMENT 1
60.      START 1
61.      MINVALUE 1
62.      MAXVALUE 2147483647
63.      CACHE 1;
64.
65. --CREAZIONE food.indirizzo_codiceindirizzo_seq
66. CREATE SEQUENCE food.indirizzo_codiceindirizzo_seq
67.      INCREMENT 1
68.      START 1
69.      MINVALUE 1
70.      MAXVALUE 2147483647
71.      CACHE 1;
72.
73. --CREAZIONE food.ordine_codicecarrello_seq
74. CREATE SEQUENCE food.ordine_codicecarrello_seq
75.      INCREMENT 1
76.      START 1
77.      MINVALUE 1
78.      MAXVALUE 2147483647
79.      CACHE 1;
80.
81. --CREAZIONE food.ordine_codicerider_seq
82. CREATE SEQUENCE food.ordine_codicerider_seq
83.      INCREMENT 1
84.      START 1
85.      MINVALUE 1
86.      MAXVALUE 2147483647

```

```

87.    CACHE 1;
88.
89. --CREAZIONE food.ordine_codiceristorante_seq
90. CREATE SEQUENCE food.ordine_codiceristorante_seq
91.    INCREMENT 1
92.    START 1
93.    MINVALUE 1
94.    MAXVALUE 2147483647
95.    CACHE 1;
96.
97. --CREAZIONE food.prodotto_codiceseriale_seq
98. CREATE SEQUENCE food.prodotto_codiceseriale_seq
99.    INCREMENT 1
100.   START 1
101.   MINVALUE 1
102.   MAXVALUE 2147483647
103.   CACHE 1;
104.
105. --CREAZIONE food.rider_codicerider_seq*/
106. CREATE SEQUENCE food.rider_codicerider_seq
107.    INCREMENT 1
108.    START 1
109.    MINVALUE 1
110.    MAXVALUE 2147483647
111.    CACHE 1;
112.
113. --CREAZIONE food.ristorante_codiceindirizzo_seq
114. CREATE SEQUENCE food.ristorante_codiceindirizzo_seq
115.    INCREMENT 1
116.    START 1
117.    MINVALUE 1
118.    MAXVALUE 2147483647
119.    CACHE 1;
120.
121. --CREAZIONE food.ristorante_codiceristorante_seq
122. CREATE SEQUENCE food.ristorante_codiceristorante_seq
123.    INCREMENT 1
124.    START 1
125.    MINVALUE 1
126.    MAXVALUE 2147483647
127.    CACHE 1;
128.
129. --CREAZIONE food.utente_codiceindirizzo_seq
130. CREATE SEQUENCE food.utente_codiceindirizzo_seq
131.    INCREMENT 1
132.    START 1
133.    MINVALUE 1
134.    MAXVALUE 2147483647
135.    CACHE 1;
136.
137. --CREAZIONE food.veicolo_codicerider_seq
138. CREATE SEQUENCE food.veicolo_codicerider_seq

```

```
139.      INCREMENT 1
140.      START 1
141.      MINVALUE 1
142.      MAXVALUE 2147483647
143.      CACHE 1;
144.
145.--CREAZIONE food.veicolo_codiceseriale_seq
146.CREATE SEQUENCE food.veicolo_codiceseriale_seq
147.      INCREMENT 1
148.      START 1
149.      MINVALUE 1
150.      MAXVALUE 2147483647
151.      CACHE 1;
152.
```

4.2 Definizioni delle tabelle e vincoli semplici

4.1.2 Definizione della tabella indirizzo

```
1.  /**
2.  * TABELLA: indirizzo
3.  * Crea generatore automatico di numeri per 'codiceindirizzo',
4.  * crea la tabella e implementa i vincoli più semplici
5.  */
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella indirizzo
11. CREATE TABLE indirizzo(
12.     nomevia          VARCHAR(32),
13.     numerocivico     VARCHAR(5),
14.     Cap              CHAR(5),
15.     citta            VARCHAR(64),
16.     provincia        CHAR(2),
17.     codiceindirizzo  SERIAL PRIMARY KEY
18. );
19.
20. - Vincolo per cap
21. ALTER TABLE indirizzo
22.     ADD CONSTRAINT chk_cap CHECK (cap ~* '[0-9]{5}$');
23.
24.
25. --Vincolo per citta
26. ALTER TABLE indirizzo
27.     ADD CONSTRAINT chk_citta CHECK (citta ~* '[a-zA-Z]{1,}$');
28.
29. --Vincolo per nome via
30. ALTER TABLE indirizzo
31.     ADD CONSTRAINT chk_nomevia CHECK (nomevia ~* '[a-zA-Z]{1,}$');
32.
33. --Vincolo per numero_Civico
34. ALTER TABLE indirizzo
35.     ADD CONSTRAINT chk_numerocivico CHECK (numerocivico ~* '[a-zA-Z0-9]{1,}$');
36.
37. --Vincolo per provincia
38. ALTER TABLE indirizzo
39.     ADD CONSTRAINT chk_provincia CHECK (provincia ~* '[a-zA-Z]{2}$');
40.
```

4.1.2 Definizione della tabella utente

```
1. /**
2.  * TABELLA: utente
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6. CREATE SCHEMA IF NOT EXISTS food;

7. SET search_path TO food;
8.
9. -- Creazione tabella utente
10. CREATE TABLE utente(
11.     nome                VARCHAR(32),
12.     cognome             VARCHAR(32),
13.     email               VARCHAR(64) PRIMARY KEY,
14.     password            VARCHAR(32),
15.     numerotelefono      VARCHAR(10),
16.     codiceindirizzo     SERIAL NOT NULL,
17.     CONSTRAINT fk_indirizzo
18.     FOREIGN KEY (codiceindirizzo) REFERENCES
    indirizzo(codiceindirizzo)
19.     ON UPDATE CASCADE
20.     ON DELETE RESTRICT
21. );
22.
23. --Vincolo per nome
24. ALTER TABLE utente
25.     ADD CONSTRAINT chk_nome CHECK (nome ~* '[a-zA-Z]$');
26.
27. --Vincolo per numero telefonico
28. ALTER TABLE utente
29.     ADD CONSTRAINT chk_numerotelefono CHECK (numerotelefono ~*
    '[0-9]{8}$');
30.
31. --Vincolo per cognome
32. ALTER TABLE utente
33.     ADD CONSTRAINT chk_cognome CHECK (cognome ~* '[a-zA-Z]$');
34.
35.
```

4.1.3 Definizione della tabella ristorante

```
1. /**
2.  * TABELLA: ristorante
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6. CREATE SCHEMA IF NOT EXISTS food;
7.
```



```

8. SET search_path TO food;
9.
10. – Creazione tabella ristorante
11. CREATE TABLE ristorante(
12.     nome VARCHAR(32),
13.     descrizione VARCHAR(512),
14.     numerotelefono CHAR(10),
15.     codiceristorante SERIAL PRIMARY KEY,
16.     codiceindirizzo SERIAL NOT NULL,
17. CONSTRAINT fk_indirizzo
18.     FOREIGN KEY (codiceindirizzo) REFERENCES
indirizzo(codiceindirizzo)
19.     ON UPDATE CASCADE
20.     ON DELETE RESTRICT
21. );
22.

```

4.1.4 Definizione enumerazione TIPO_PRODOTTO

```

1. /**
2. * Enumerazione: TIPO_PRODOTTO
3. * Crea la tabella e implementa i vincoli più semplici
4. */
5.
6. CREATE SCHEMA IF NOT EXISTS food;
7.
8. SET search_path TO food;
9.
10. CREATE TYPE TIPO_PRODOTTO AS ENUM('bevanda', 'cibo');

```

4.1.5 Definizione enumerazione CATEGORIA_PRODOTTO

```

11. /**
12. * Enumerazione: CATEGORIA_PRODOTTO
13. * Crea la tabella e implementa i vincoli più semplici
14. */
15.
16. CREATE SCHEMA IF NOT EXISTS food;
17.
18. SET search_path TO food;
19.
20. CREATE TYPE CATEGORIA_PRODOTTO AS ENUM('alcolico',
'analcolico', 'antipasti', 'primi', 'secondi', 'contorni',
'dolci');

```

4.1.6 Definizione tabella prodotto

```
1.  /**
2.  * TABELLA: CATEGORIA_PRODOTTO
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella prodotto
11. CREATE TABLE prodotto(
12.     nome                                VARCHAR(32),
13.     prezzo                              REAL,
14.     codiceseriale                       SERIAL PRIMARY KEY,
15.     scadenza                           DATE,
16.     pathfoto                            VARCHAR(64),
17.     tipoprodotto TIPO_PRODOTTO          NOT NULL,
18.     categoria CATEGORIA_PRODOTTO       NOT NULL
19. );
20.
21. --Vincolo tipoProdotto
22. ALTER TABLE prodotto
23. ADD CONSTRAINT chk_tipo_Prodotto CHECK (prodotto.tipoprodotto
    IN ('bevanda', 'cibo'));
24.
25. --Vincolo categoria
26. ALTER TABLE prodotto
27. ADD CONSTRAINT chk_categoria CHECK (prodotto.categoria IN
    ('alcolico', 'analcolico', 'antipasti', 'primi', 'secondi',
    'contorni', 'dolci'));
```

4.1.7 Definizione tabella fornitura

```
1.  /**
2.  * TABELLA: fornitura
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella fornitura
11. CREATE TABLE fornitura(
12.     quantitaprodotto INTEGER NOT NULL CHECK (quantitaProdotto > 0),
13.     codiceseriale     SERIAL ,
14.     codiceristorante  SERIAL ,
```

```

15.     PRIMARY KEY (codiceseriale, codiceristorante),
16.     CONSTRAINT fk_prodotto
17.     FOREIGN KEY (codiceseriale) REFERENCES
    prodotto(codiceseriale)
18.     ON DELETE CASCADE
19.     ON UPDATE CASCADE
20. );

```

4.1.8 Definizione tabella rider

```

1.  /**
2.  * TABELLA: rider
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella rider
11. CREATE TABLE rider(
12.     nome                VARCHAR(32),
13.     cognome              VARCHAR(32),
14.     biografia            VARCHAR(512),
15.     pathfoto             VARCHAR(64),
16.     Numeroattivit       INTEGER,
17.     codicerider          SERIAL PRIMARY KEY
18. );
19.

```

4.1.9 Definizione enumerazione TIPO_VEICOLO

```

1.  /**
2.  * TABELLA: TIPO_VEICOLO
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. CREATE TYPE TIPO_VEICOLO AS ENUM ('moto', 'bicicletta');

```

4.1.10 Definizione tabella veicolo

```
1. /**
2. * TABELLA: veicolo
3. * Crea la tabella e implementa i vincoli più semplici
4. */
5.
6. CREATE SCHEMA IF NOT EXISTS food;
7.
8. SET search_path TO food;
9.
10. -- Creazione tabella veicolo
11. CREATE TABLE veicolo(
12.     codiceseriale          SERIAL PRIMARY KEY,
13.     annoimmatricolazione  DATE,
14.     marca                  VARCHAR(32),
15.     modello                VARCHAR(32),
16.     tipoveicolo TIPO_VEICOLO NOT NULL,
17.     codicerider            SERIAL NOT NULL,
18.     CONSTRAINT fk_rider FOREIGN KEY(codicerider) REFERENCES
rider(codicerider)
19.     ON UPDATE CASCADE
20.     ON DELETE CASCADE
21. );
22.
23. --Vincolo tipoVeicolo
24. ALTER TABLE veicolo
25. ADD CONSTRAINT chk_tipoVeicolo CHECK (veicolo.tipoveicolo IN
('moto','bicicletta'));
26.
```

4.1.11 Definizione tabella corrierePer

```
1. /**
2. * TABELLA: corrierePer
3. * Crea la tabella e implementa i vincoli più semplici
4. */
5.
6. CREATE SCHEMA IF NOT EXISTS food;
7.
8. SET search_path TO food;
9.
10. -- Creazione tabella corrierePer
11. CREATE TABLE corrierePer(
12.     codiceristorante SERIAL,
13.     codicerider       SERIAL,
14.     CONSTRAINT pk_corriere PRIMARY KEY(codicerider,
codiceristorante),
```

```

15.     CONSTRAINT fk_rider FOREIGN KEY(codicerider) REFERENCES
    rider(codicerider)
16.     ON UPDATE CASCADE
17.     ON DELETE CASCADE,
18.     CONSTRAINT fk_ristorante FOREIGN KEY(codiceristorante)
REFERENCES ristorante(codiceristorante)
19.     ON UPDATE CASCADE
20.     ON DELETE CASCADE
21. );
22.

```

4.1.12 Definizione tabella carrello

```

1.  /**
2.  * TABELLA: carrello
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella carrello
11. CREATE TABLE carrello(
12.     data          DATE NOT NULL,
13.     codicecarrello SERIAL PRIMARY KEY
14. );
15.

```

4.1.13 Definizione tabella composizioneCarrello

```

1.  /**
2.  * TABELLA: composizioneCarrello
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. - Creazione tabella composizioneCarrello
11. CREATE TABLE composizioneCarrello(
12.     codicecarrello SERIAL,
13.     codiceseriale  SERIAL,
14.     quantita       INTEGER CHECK(quantita IS NOT NULL AND
    quantita > 0),
15.     CONSTRAINT pk_composizioneCarrello PRIMARY
    KEY(codiceSeriale, codicecarrello),
16.     CONSTRAINT fk_prodotto FOREIGN KEY(codiceseriale)
REFERENCES prodotto(codiceseriale)

```

```

17.     ON UPDATE CASCADE
18.     ON DELETE CASCADE,
19.     CONSTRAINT fk_carrello FOREIGN KEY(codicecarrello)
REFERENCES carrello(codicecarrello)
20.     ON UPDATE CASCADE
21.     ON DELETE CASCADE
22. );

```

4.1.14 Definizione enumerazione STATO_ORDINE

```

1.  /**
2.  * TABELLA: TIPO_VEICOLO
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. CREATE TYPE STATO_ORDINE AS ENUM ('A', 'N', 'NA');

```

4.1.15 Definizione tabella ordine

```

1.  /**
2.  * TABELLA: ordine
3.  * Crea la tabella e implementa i vincoli più semplici
4.  */
5.
6.  CREATE SCHEMA IF NOT EXISTS food;
7.
8.  SET search_path TO food;
9.
10. – Creazione tabella ordine
11. CREATE TABLE ordine(
12.     codicecarrello      SERIAL,
13.     email                VARCHAR(64),
14.     codiceristorante     SERIAL,
15.     codicerider          SERIAL,
16.     prezzo               REAL,
17.     dataacquisto        DATE,
18.     stato STATO_ORDINE  NOT NULL,
19.     PRIMARY KEY(codicecarrello, email, codiceristorante,
codicerider),
20.     CONSTRAINT fk_carrello FOREIGN KEY(codicecarrello)
REFERENCES carrello(codicecarrello)
21.     ON UPDATE CASCADE
22.     ON DELETE CASCADE,

```

```

23.     CONSTRAINT fk_utente FOREIGN KEY(email) REFERENCES
       utente(email)
24.     ON UPDATE CASCADE
25.     ON DELETE CASCADE,
26.     CONSTRAINT fk_rider FOREIGN KEY(codicerider) REFERENCES
       rider(codicerider)
27.     ON UPDATE CASCADE
28.     ON DELETE CASCADE,
29.     CONSTRAINT fk_ristorante FOREIGN KEY(codiceristorante)
       REFERENCES ristorante(codiceristorante)
30.     ON UPDATE CASCADE
31.     ON DELETE CASCADE
32. );
33.
34. --Vincolo ordine
35. ALTER TABLE ordine
36. ADD CONSTRAINT chk_stato CHECK (ordine.stato IN ('A', 'N', 'NA'));

```

4.2 Viste

La vista mostra, per ciascun rider, il veicolo che gli appartiene.

4.2.1 riderVeicoloView

```

1. --Vincolo per associare ad ogni rider un veicolo
2. CREATE VIEW riderVeicoloView AS
3. SELECT R.codicerider, V.tipoveicolo
4. FROM rider AS R NATURAL JOIN veicolo AS V;

```

4.3 Definizione Trigger

4.3.1 Definizione Trigger verifico_email_password

Il Trigger **verifico_email_password** verifica che l'email abbia:

- almeno un carattere o una cifra numerica o un carattere speciale prima della '@'
- almeno un carattere o una cifra numerica o un carattere speciale dopo la '@' e prima del punto
- dopo il punto almeno un carattere

Verifica, inoltre, anche che la password sia costituita da:

- almeno un carattere maiuscolo,
- almeno un carattere minuscolo,
- almeno una cifra numerica

- almeno un carattere speciale

Nel caso in cui l'utente inserisce un formato email e password non conformi si solleva un eccezione.

```

1. CREATE FUNCTION food.controllo_email_password()
2. RETURNS TRIGGER
3. LANGUAGE 'plpgsql'
4. COST 100
5. VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN
8. IF (NEW.email !~* '^[_A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-
  Za-z]+$') THEN
9. RAISE EXCEPTION 'Errore: formato email non valido.';
10. END IF;
11.
12. IF (NEW.password ~* '^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])
  (?!.*?[@$%^&*~]).{8,}$') THEN
13. RAISE EXCEPTION 'Errore: la password non valida';
14. END IF;
15. RETURN NEW;
16. END;
17. $BODY$;
18.
19. CREATE TRIGGER verifico_email_password
20. BEFORE INSERT
21. ON food.utente
22. FOR EACH ROW
23. EXECUTE PROCEDURE food.controllo_email_password();
24.

```

4.3.2 Definizione Trigger controllo_quantitaProdotto

Il Trigger **controllo_quantitaProdotto** verifica che quando vengono inseriti o aggiornate le quantità dei prodotti non siano minori o uguale a zero. Nel caso si inseriscono una quantità di prodotto non conformi si solleva un eccezione.

```

1. CREATE FUNCTION food.controllo_quantitaProdotto()
2. RETURNS TRIGGER
3. LANGUAGE 'plpgsql'
4. COST 100
5. VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN
8. IF (NEW.quantitaprodotto <= 0) THEN
9. RAISE EXCEPTION 'la quantità di prodotto deve essere
  maggiore di zero.';
10. END IF;
11. END;
12. $BODY$;
13.

```



```

14. CREATE TRIGGER controllo_quantitaProdotto
15.     BEFORE INSERT OR UPDATE
16.     ON food.fornitura
17.     FOR EACH ROW
18.     EXECUTE PROCEDURE food.controllo_quantitaProdotto();
19.

```

4.3.3 Definizione Trigger modifica_password

Il Trigger **modifica_password** aggiorna la password quando l'utente vuole inserire una nuova password, in particolare controlla prima che la password che l'utente desidera inserire sia diversa da quella già presente nel database e in caso di errore lancia un exception.

```

1. CREATE FUNCTION food.modifica_password()
2.     RETURNS TRIGGER
3.     LANGUAGE 'plpgsql'
4.     COST 100
5.     VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN
8.     IF (NEW.password != OLD.password) THEN
9.         UPDATE utente SET password = NEW.password WHERE email =
10.        OLD.email;
11.    ELSE
12.        RAISE EXCEPTION 'Si prega di inserire una NUOVA password
13.        diversa dalla vecchia';
14.    END IF;
15.    RETURN NEW;
16. END;
17. $BODY$;
18.
19. CREATE TRIGGER modifica_password
20.     BEFORE UPDATE
21.     ON food.utente
22.     FOR EACH ROW
23.     EXECUTE PROCEDURE food.modifica_password();
24.

```

4.3.4 Definizione Trigger controllo_attivitarider

Il Trigger **controllo_attivitarider** controlla il numero di ristoranti (attività) associati ai rider. Quando si fa corrispondere ad un rider più di tre attività si solleva un eccezione.

```

1. CREATE FUNCTION food.controllo_attivitarider()
2.     RETURNS TRIGGER
3.     LANGUAGE 'plpgsql'
4.     COST 100
5.     VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN
8.     IF (NEW.numeroattivitaa>3) THEN

```

```

9.      RAISE EXCEPTION 'Errore: non è possibile associare al
      rider più di tre attività';
10.    END IF;
11. END;
12. $BODY$;
13.
14. CREATE TRIGGER controllo_attivitarider
15.    BEFORE INSERT OR UPDATE
16.    ON food.rider
17.    FOR EACH ROW
18.    WHEN (NEW.numeroattivit  > 3)
19.    EXECUTE PROCEDURE food.controllo_attivitarider();
20.

```

4.3.5 Definizione Trigger controllo_statoOrdine

Il Trigger **controllo_statoOrdine** da errore nel momento in cui lo stato dell'ordine è NA, cioè non è arrivato a destinazione.

```

1. CREATE FUNCTION food.controllo_stato()
2.    RETURNS TRIGGER
3.    LANGUAGE 'plpgsql'
4.    COST 100
5.    VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN
8.    RAISE EXCEPTION 'ERRORE: ordine non consegnato';
9.    RETURN NEW;
10. END
11. $BODY$;
12.
13. CREATE TRIGGER "controllo_statoOrdine"
14.    AFTER UPDATE OF stato
15.    ON food.ordine
16.    FOR EACH ROW
17.    WHEN (NEW.stato = 'NA'::food.stato_ordine)
18.    EXECUTE PROCEDURE food.controllo_stato();
19.

```

4.3.6 Definizione Trigger controllo_costoTotaleOrdine

Il Trigger **controllo_costoTotaleOrdine** da errore nel momento in cui il costo risultante dell'ordine non calcola correttamente il costo totale dell'ordine

```

1. CREATE FUNCTION food.controllo_prezzo_ordine()
2.    RETURNS TRIGGER
3.    LANGUAGE 'plpgsql'
4.    COST 100
5.    VOLATILE NOT LEAKPROOF
6. AS $BODY$
7. BEGIN

```

```

8.      IF (NEW.prezzo <= 0) THEN
9.          RAISE EXCEPTION 'prezzo totale ordine non corretto.';
10.     ELSE
11.         RETURN NEW;
12.     END IF;
13. END;
14. $BODY$;
15.
16. ALTER FUNCTION food.controllo_prezzo_ordine()
17.     OWNER TO postgres;
18.
19. CREATE TRIGGER "controllo_costoTotaleOrdine"
20.     BEFORE INSERT
21.     ON food.ordine
22.     FOR EACH ROW
23.     EXECUTE PROCEDURE food.controllo_prezzo_ordine();
24.

```

4.4 Definizione Procedure

4.4.1 Definizione Procedura inserisci_indirizzo

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_indirizzo(
2.     nomevia CHARACTER VARYING,
3.     cap CHARACTER,
4.     citta CHARACTER VARYING,
5.     provincia CHARACTER,
6.     codiceindirizzo INTEGER,
7.     numerocivico INTEGER)
8.
9. LANGUAGE 'plpgsql'
10. AS $BODY$
11. BEGIN
12.
13.     INSERT INTO food.indirizzo VALUES(nomevia, cap, citta, provincia,
        codiceindirizzo, numerocivico);
14. COMMIT;
15.
16. END;
17. $BODY$;
18.

```

4.4.2 Definizione Procedura inserisci_utente

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_utente(
2.     nome CHARACTER VARYING,
3.     cognome CHARACTER VARYING,
4.     email CHARACTER VARYING,

```

```

5.      password CHARACTER VARYING,
6.      numerotelefono CHARACTER,
7.      codiceindirizzo INTEGER)
8.
9. LANGUAGE 'plpgsql'
10. AS $BODY$
11. BEGIN
12.
13.   INSERT INTO food.utente VALUES(nome, cognome, email,
      password, numerotelefono, codiceindirizzo);
14. COMMIT;
15.
16. END;
17. $BODY$;
18.

```

4.4.3 Definizione Procedura inserisci_ristorante

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_ristorante(
2.     nome CHARACTER VARYING,
3.     descrizione CHARACTER VARYING,
4.     numerotelefonico CHARACTER,
5.     codiceristorante INTEGER,
6.     codiceindirizzo INTEGE)
7.
8. LANGUAGE 'plpgsql'
9. AS $BODY$
10. BEGIN
11.
12.   INSERT INTO food.ristorante VALUES(nome, descrizione,
      numerotelefonico, codiceristorante, codiceindirizzo, consegnato
      );
13. COMMIT;
14.
15. END;
16. $BODY$;
17.

```

4.4.4 Definizione Procedura inserisci_prodotto

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_prodotto(
2.     nome CHARACTER VARYING,
3.     prezzo REAL,
4.     codiceseriale INTEGER,
5.     scadenza DATE,
6.     pathfoto CHARACTER VARYING,
7.     tipoprodotto food.tipo_prodotto,
8.     categoria food.categoria_prodotto)
9. LANGUAGE 'plpgsql'
10. AS $BODY$
11.

```

```

12. BEGIN
13.   INSERT INTO food.prodotto VALUES(nome, prezzo, codiceseriale,
    scadenza, pathfoto, tipoprodotto,categoria);
14. COMMIT;
15.
16. END;
17. $BODY$;

```

4.4.5 Definizione Procedura inserisci_fornitura

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_fornitura(
2.   quantitaprodotto INTEGER,
3.   codiceseriale INTEGER,
4.   codiceristorante INTEGER)
5. LANGUAGE 'plpgsql'
6. AS $BODY$
7.
8. BEGIN
9.
10.  INSERT INTO food.fornitura VALUES(quantitaprodotto,
    codiceseriale, codiceristorante);
11. COMMIT;
12.
13. END;
14. $BODY$;

```

4.4.6 Definizione Procedura inserisci_rider

```

1. CREATE OR REPLACE PROCEDURE food.inserisci_rider(
2.   nome CHARACTER VARYING,
3.   cognome CHARACTER VARYING,
4.   biografia CHARACTER VARYING,
5.   pathfoto CHARACTER VARYING,
6.   codicerider INTEGER,
7.   numeroattivita INTEGER)
8. LANGUAGE 'plpgsql'
9. AS $BODY$
10.
11. BEGIN
12.
13.  INSERT INTO food.rider VALUES(nome, cognome, biografia,
    pathfoto, codicerider, numeroattivita);
14. COMMIT;
15.
16. END;
17. $BODY$;
18.

```

4.4.7 Definizione Procedura inserisci_veicolo

```
1. CREATE OR REPLACE PROCEDURE food.inserisci_veicolo(  
2.     codiceseriale INTEGER,  
3.     annoimmatricolazione DATE,  
4.     marca CHARACTER VARYING,  
5.     modello CHARACTER VARYING,  
6.     tipoveicolo food.tipo_veicolo,  
7.     codicerider INTEGER)  
8. LANGUAGE 'plpgsql'  
9. AS $BODY$  
10.  
11. BEGIN  
12.  
13.     INSERT INTO food.veicolo VALUES(codiceseriale,  
14.     annoimmatricolazione, marca, modello, tipoveicolo,  
15.     codicerider);  
16. COMMIT;  
17. END;  
18. $BODY$;
```

4.4.8 Definizione Procedura inserisci_carrello

```
1. CREATE OR REPLACE PROCEDURE food.inserisci_carrello(  
2.     data DATE,  
3.     codicecarrello INTEGER)  
4. LANGUAGE 'plpgsql'  
5. AS $BODY$  
6.  
7. BEGIN  
8.  
9.     INSERT INTO food.carrello VALUES(DATE, codicecarrello);  
10. COMMIT;  
11.  
12. END;  
13. $BODY$;
```

4.4.9 Definizione Procedura inserisci_ordine

```
1. CREATE OR REPLACE PROCEDURE food.inserisci_ordine(  
2.     codicecarrello INTEGER,  
3.     email CHARACTER VARYING,  
4.     codiceristorante INTEGER,  
5.     codicerider INTEGER,  
6.     prezzo REAL,  
7.     dataacquisto DATE,  
8.     stato food.stato_ordine)
```

```

9.  LANGUAGE 'plpgsql'
10. AS $BODY$
11.
12. BEGIN
13.
14.     INSERT INTO food.ordine VALUES(codicecarrello, email,
        codiceristorante, codicerider ,prezzo, dataAqusto, stato);
15. COMMIT;
16.
17. END;
18. $BODY$;
19.

```

4.4.10 Definizione Procedura cancella_indirizzo

```

1.  CREATE OR REPLACE PROCEDURE food.elimina_indirizzo(
2.      codindirizz INTEGER)
3.  LANGUAGE 'plpgsql'
4.  AS $BODY$
5.
6.  BEGIN
7.
8.      DELETE FROM food.indirizzo WHERE codiceindirizzo=codindirizz;
9.
10. END;
11. $BODY$;

```

4.4.11 Definizione Procedura cancella_utente

```

1.  CREATE OR REPLACE PROCEDURE food.elimina_utente(
2.      emailutente CHARACTER VARYING)
3.  LANGUAGE 'plpgsql'
4.  AS $BODY$
5.
6.  BEGIN
7.
8.      DELETE FROM food.utente WHERE email=emailutente;
9.
10. END;
11. $BODY$;
12.

```

4.4.12 Definizione Procedura cancella_ristorante

```

1.  CREATE OR REPLACE PROCEDURE food.elimina_ristorante(
2.      codcristorante INTEGER)
3.  LANGUAGE 'plpgsql'
4.  AS $BODY$
5.
6.  BEGIN

```

```

7.
8.      DELETE FROM food.ristorante WHERE
      codiceristorante=codCRistorante;
9.
10. END;
11. $BODY$;
12.

```

4.4.13 Definizione Procedura cancella_prodotto

```

1. CREATE OR REPLACE PROCEDURE food.elimina_prodotto(
2.     nomeprodotto CHARACTER VARYING)
3. LANGUAGE 'plpgsql'
4. AS $BODY$
5.
6. BEGIN
7.
8.     DELETE FROM food.prodotto WHERE nome=nomeProdotto;
9.
10. END;
11. $BODY$;

```

4.4.14 Definizione Procedura cancella_fornitura

```

1. CREATE OR REPLACE PROCEDURE food.elimina_fornitura(
2.     codseriale INTEGER)
3. LANGUAGE 'plpgsql'
4. AS $BODY$
5.
6. BEGIN
7.
8.     DELETE FROM food.fornitura WHERE codiceseriale=codseriale;
9.
10. END;
11. $BODY$;
12.
13.

```

4.4.15 Definizione Procedura cancella_rider

```

1. CREATE OR REPLACE PROCEDURE food.elimina_rider(
2.     codrider INTEGER)
3. LANGUAGE 'plpgsql'
4. AS $BODY$
5.
6. BEGIN
7.
8.     DELETE FROM food.rider WHERE codicerider=codrider;
9.
10. END;
11. $BODY$;

```


12.
13.

4.4.16 Definizione Procedura cancella_veicolo

```
1. CREATE OR REPLACE PROCEDURE food.elimina_veicolo(  
2.     codiceserialeveicolo INTEGER)  
3. LANGUAGE 'plpgsql'  
4. AS $BODY$  
5.  
6. BEGIN  
7.  
8.     DELETE FROM food.veicolo WHERE  
       codiceseriale=codiceserialeveicolo;  
9.  
10. END;  
11. $BODY$;  
12.
```

4.4.17 Definizione Procedura cancella_carrello

```
1. CREATE OR REPLACE PROCEDURE food.elimina_carrello(  
2.     codcarrello INTEGER)  
3. LANGUAGE 'plpgsql'  
4. AS $BODY$  
5.  
6. BEGIN  
7.  
8.     DELETE FROM food.carrello WHERE codicecarrello=codCarrello;  
9.  
10. END;  
11. $BODY$;  
12.
```

4.4.18 Definizione Procedura cancella_ordine

```
1. CREATE OR REPLACE PROCEDURE food.elimina_ordine(  
2.     codcarrello INTEGER,  
3.     emailutente CHARACTER VARYING,  
4.     codristorante INTEGER,  
5.     codrider INTEGER,  
6.     prezzoordine REAL,  
7.     dataacquistoordine DATE,  
8.     statoordine food.stato_ordine)  
9. LANGUAGE 'plpgsql'  
10. AS $BODY$  
11.  
12. BEGIN
```

```

13.
14.  DELETE FROM food.ordine WHERE codicecarrello=codcarrello AND
    email = emailutente AND codiceristorante=codristorante AND
    codicerider=codrider AND prezzo=prezzoordine AND
    dataacquisto=dataacquistoordine AND stato=statoordine;
15.
16. END;
17. $BODY$;
18.
19.

```

4.4.19 Definizione Procedura aggiorna_indirizzo

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_indirizzo(
2.      nomevia_utente CHARACTER VARYING,
3.      cap_utente CHARACTER,
4.      citta_utente CHARACTER VARYING,
5.      provincia_utente CHARACTER,
6.      codiceindirizzo_utente INTEGER,
7.      numerocivico_utente INTEGER)
8.  LANGUAGE 'plpgsql'
9.  AS $BODY$
10.
11. BEGIN
12.
13.  UPDATE food.indirizzo SET nomevia=nomevia_utente ,
    cap=cap_utente, citta=citta_utente, provincia=provincia_utente,
    numerocivico=numerocivico_utente WHERE codiceindirizzo =
    codiceindirizzo_utente;
14. COMMIT;
15.
16. END;
17. $BODY$;
18.

```

4.4.20 Definizione Procedura aggiorna_utente

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_utente(
2.      nomeutente CHARACTER VARYING,
3.      cognomeutente CHARACTER VARYING,
4.      emailutente CHARACTER VARYING,
5.      passwordutente CHARACTER VARYING,
6.      numerotelefonoutente CHARACTER,
7.      codiceindirizzoutente INTEGER)
8.  LANGUAGE 'plpgsql'
9.  AS $BODY$
10.
11. BEGIN
12.

```

```

13.  UPDATE food.utente SET nome = nomeutente,
    cognome=cognomeutente, password=passwordutente,
    numerotelefono=numerotelefonoutente,
    codiceindirizzo=codiceindirizzoutente WHERE email=emailutente;
14.
15. END;
16. $BODY$;
17.

```

4.4.21 Definizione Procedura aggiorna_ristorante

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_ristorante(
2.      nome_ristorante CHARACTER VARYING,
3.      descrizione_ristorante CHARACTER VARYING,
4.      numerotelefono_ristorante CHARACTER,
5.      codiceristorante_ristorante INTEGER,
6.      codiceindirizzo_ristorante INTEGER
7.  )
8.  LANGUAGE 'plpgsql'
9.  AS $BODY$
10.
11. BEGIN
12.
13. UPDATE food.ristorante SET nome=nome_ristorante,
    descrizione=descrizione_ristorante,
    numerotelefono=numerotelefono_ristorante,
    codiceindirizzo=codiceindirizzo_ristorante WHERE
    codiceristorante=codiceristorante_ristorante;
14. COMMIT;
15.
16. END;
17. $BODY$;
18.
19.

```

4.4.22 Definizione Procedura aggiorna_veicolo

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_veicolo(
2.      codiceserialeveicolo INTEGER,
3.      annoimmatricolazioneveicolo DATE,
4.      marcaveicolo CHARACTER VARYING,
5.      modelloveicolo CHARACTER VARYING,
6.      tipoveicolo_v food.tipo_veicolo,
7.      codiceriderveicolo INTEGER)
8.  LANGUAGE 'plpgsql'
9.  AS $BODY$
10.

```

```

11. BEGIN
12.
13.     UPDATE food.veicolo SET codicerider=codiceriderveicolo,
        annoimmatricolazione=annoinmatricolazioneveicolo,
        marca=marcaveicolo, modello=modelloveicolo,
        tipoveicolo=tipoveicolo_v WHERE codiceseriale =
        codiceserialeveicolo;
14. COMMIT;
15.
16. END;
17. $BODY$;

```

4.4.23 Definizione Procedura aggiorna_rider

```

1. CREATE OR REPLACE PROCEDURE food.aggiorna_rider(
2.     nome_r CHARACTER VARYING,
3.     cognome_r CHARACTER VARYING,
4.     biografia_r CHARACTER VARYING,
5.     pathfoto_r CHARACTER VARYING,
6.     codicerider_r INTEGER,
7.     numeroattivit _r INTEGER)
8. LANGUAGE 'plpgsql'
9. AS $BODY$
10.
11. BEGIN
12.     UPDATE food.rider SET nome=nome_r, cognome=cognome_r,
        biografia=biografia_r, pathfoto=pathfoto_r,
        numeroattivit =numeroattivit _r WHERE
        codicerider=codicerider_r;
13. COMMIT;
14.
15. END;
16. $BODY$;
17.

```

4.4.24 Definizione Procedura aggiorna_prodotto

```

1. CREATE OR REPLACE PROCEDURE food.aggiorna_prodotto(
2.     nome_prodotto CHARACTER VARYING,
3.     prezzo_prodotto REAL,
4.     codiceseriale_prodotto INTEGER,
5.     scadenza_prodotto DATE,
6.     pathfoto_prodotto CHARACTER VARYING,
7.     tipoprodotto_prodotto food.tipo_prodotto,
8.     categoria_prodotto food.categoria_prodotto)
9. LANGUAGE 'plpgsql'
10. AS $BODY$
11.
12. BEGIN

```

```

13.
14.  UPDATE food.prodotto SET nome = nome_prodotto,
    prezzo=prezzo_prodotto, pathfoto=pathfoto_prodotto,
    tipoprodotto=tipoprodotto_prodotto,
    categoria=categoria_prodotto WHERE codiceseriale =
    codiceseriale_prodotto;
15.
16. END;
17. $BODY$;

```

4.4.25 Definizione Procedura aggiorna_fornitura

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_fornitura(
2.      f_quantitaprodotto INTEGER,
3.      f_codiceseriale INTEGER,
4.      f_codiceristorante INTEGER)
5.  LANGUAGE 'plpgsql'
6.  AS $BODY$
7.  BEGIN
8.
9.      UPDATE food.fornitura SET quantitaprodotto =
    F_quantitaprodotto, codiceristorante = F_codiceristorante WHERE
    codiceseriale = F_codiceseriale;
10.
11. END;
12. $BODY$;
13.

```

4.4.26 Definizione Procedura aggiorna_carrello

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_carrello(
2.      datacarrello DATE,
3.      codcarrello INTEGER)
4.  LANGUAGE 'plpgsql'
5.  AS $BODY$
6.
7.  BEGIN
8.      UPDATE food.carrello SET DATA = datacarrello WHERE
    codicecarrello = codcarrello;
9.
10. END;
11. $BODY$;
12.

```

4.4.27 Definizione Procedura aggiorna_ordine

```

1.  CREATE OR REPLACE PROCEDURE food.aggiorna_ordine(

```

```
2.         codicecarrello_ordine INTEGER,
3.         email_ordine CHARACTER VARYING,
4.         codiceristorante_ordine INTEGER,
5.         codicerider_ordine INTEGER,
6.         prezzoordine REAL,
7.         dataacquistoordine DATE,
8.         statoordine food.stato_ordine)
9. LANGUAGE 'plpgsql'
10. AS $BODY$
11.
12. BEGIN
13.
14. UPDATE food.ordine SET codicecarrello = codicecarrello_ordine,
    codiceristorante=codiceristorante_ordine,
    codicerider=codicerider_ordine, consegnato=consegnato_ordine,
    prezzo =prezzoordine, dataacquisto = dataacquistoordine, stato =
    statoordine WHERE email=email_ordine;
15.
16. END;
17. $BODY$;
18.
```

4.5 Popolamento con dati di esempio

```
1.  -- SCRIPT PER IL POPOLAMENTO DELLA BASE DI DATI
2.  -----
3.  -- Universita degli Studi di Napoli Federico II
4.  -- Insegnamento di Basi di Dati e Sistemi informativi
5.  -----
6.  -- Questo script popola il database con dati fittizi.
7.  -----
8.
9.  SET search_path TO food;
10.
11. --Popolamento tabella indirizzo
12. INSERT INTO indirizzo(nomeVia, numeroCivico, CAP, citta, provincia)
    VALUES
13. ('coolstreet', 3, '84012', 'coolcity', 'SA'),
14. ('dundermifflin', 9, '18512', 'scranton', 'US'),
15. ('nazionale', 29, '84070', 'sangiovanniapiro', 'SA'),
16. ('leopardi', 24, '80125', 'napoli', 'NA'),
17. ('claudio', 21, '80125', 'napoli', 'NA'),
18. ('sucuzzone', 34, '71420', 'maratea', 'PZ'),
19. ('answer', 42, '50003', 'question', 'EN'),
20. ('guercet', 33, '57129', 'vallais', 'CH');
21.
22.
23. --Popolamento tabella utente
24. INSERT INTO utente(nome, cognome, email, password, numeroTelefono,
    codiceIndirizzo) VALUES
25. ('antonio', 'verdi', 'fakemail@domain.com', 'Secure9@',
    '3333337890', 4),
26. ('michael', 'scoot', 'bestboss@ever.com', 'passwo8#', '1231234567',
    3),
27. ('dwight', 'schrute', 'assistant@live.it', 'beetsf8@', '4454356760',
    3),
28. ('arthur', 'dent', 'dont@panic.com', 'dontpa4#', '3456783331', 8),
29. ('achille', 'tartaruga', 'geb@egb.com', 'granch3@', '3333333333',
    2),
30. ('prof', 'dieti', 'ti@boccio.com', 'forsen5#', '1231233210', 6),
31. ('isaac', 'asimov', 'sci@fi.com', 'fondaz2@', '7345689231', 2),
32. ('puja', 'suez', 'papa@pujaz.it', 'sgaper7@', '4572891206', 4),
33. ('jorge', 'borges', 'book@ofsand.com', 'alephw1#', '0123456789',
    5),
34. ('gregor', 'samsa', 'blatta@vender.com', 'passwo5@', '2344327896',
    7);
35.
36.
37. --Popolamento tabella prodotto
38. INSERT INTO prodotto(nome, prezzo, scadenza, pathFoto,
    tipoProdotto, categoria) VALUES
39. ('margherita', 3.50, '2021-05-23', '/', 'cibo', 'primi'),
40. ('diavola', 4.50, '2021-04-09', '/', 'cibo', 'primi'),
41. ('bianca', 3.00, '2021-04-25', '/', 'cibo', 'primi'),
42. ('spaghetti', 5.00, '2021-04-16', '/', 'cibo', 'primi'),
43. ('fiorentina', 30.00, '2021-03-31', '/', 'cibo', 'secondi'),
```

```

44. ('filetto', 35.00, '2021-03-28', '/', 'cibo', 'secondi'),
45. ('tbone', 27.00, '2021-03-30', '/', 'cibo', 'secondi'),
46. ('salmone', 25.00, '2021-03-24', '/', 'cibo', 'secondi'),
47. ('tonno', 26.00, '2021-03-22', '/', 'cibo', 'secondi'),
48. ('falafel', 15.00, '2021-04-18', '/', 'cibo', 'secondi'),
49. ('nuggets', 7.00, '2022-02-13', '/', 'cibo', 'secondi'),
50. ('insalata', 3.00, '2021-03-13', '/', 'cibo', 'contorni'),
51. ('patatine', 4.00, '2021-06-21', '/', 'cibo', 'contorni'),
52. ('pomodori', 3.00, '2021-03-13', '/', 'cibo', 'contorni'),
53. ('zeppole', 5.00, '2021-03-13', '/', 'cibo', 'dolci'),
54. ('torta', 7.00, '2021-03-17', '/', 'cibo', 'dolci'),
55. ('panettone', 4.50, '2021-03-19', '/', 'cibo', 'dolci'),
56. ('mare', 7.00, '2021-03-14', '/', 'cibo', 'antipasti'),
57. ('monti', 6.50, '2021-03-17', '/', 'cibo', 'antipasti'),
58. ('mare-monti', 10.00, '2021-03-14', '/', 'cibo', 'antipasti'),
59. ('vegano', 6.00, '2021-06-25', '/', 'cibo', 'antipasti'),
60. ('acqua', 1.00, '2028-01-01', '/', 'bevanda', 'analcolico'),
61. ('coca', 2.00, '2022-09-20', '/', 'bevanda', 'analcolico'),
62. ('aranciata', 2.00, '2022-10-27', '/', 'bevanda', 'analcolico'),
63. ('gassosa', 2.00, '2022-08-14', '/', 'bevanda', 'analcolico'),
64. ('succo-frutta', 2.50, '2021-05-20', '/', 'bevanda', 'analcolico'),
65. ('birra', 1.50, '2021-12-25', '/', 'bevanda', 'alcolico'),
66. ('vino-rosso', 2.50, '2021-09-25', '/', 'bevanda', 'alcolico'),
67. ('vino-bianco', 3.50, '2021-07-10', '/', 'bevanda', 'alcolico'),
68. ('mojito', 4.00, '2021-11-03', '/', 'bevanda', 'alcolico');
69.
70.
71. --Popolamento tabella ristorante
72. INSERT INTO ristorante(nome, descrizione, numeroTelefono,
codiceIndirizzo) VALUES
73. ('gnab-gib', 'ristorante al termine dell'universo', '0001112229', 7),
74. ('eco', 'il nome sta per economico', '8000774563', 6),
75. ('enigma', 'il nome e` il vero enigma', '2340255363', 3);
76.
77.
78. --Popolamento tabella rider
79. INSERT INTO rider(nome, cognome, biografia, pathFoto) VALUES
80. ('abdul', 'alhazred', 'dice di essere un rider veloce ed
efficiente', '/', 2),
81. ('philip', 'fry', 'ha paura di cadere nei frigoriferi', '/', 1),
82. ('gennaro', 'esposito', 'scegli me non te ne pentirai', '/', 1),
83. ('pasquale', 'esposito', 'se vuoi un pasto caldo in 5 minuti chiama
me', '/', 2),
84. ('palmer', 'eldrich', 'consegna anche can-d', '/', 3),
85. ('wow', 'bagger', 'ha avuto un problema con degli elastici, un
pranzo liquido e un acceleratore di particelle', '/', 2),
86. ('corrado', 'corriere', 'dice di essere predestinato a questo
lavoro', '/', 1);
87.
88.
89. --Popolamento tabella veicolo
90. INSERT INTO veicolo(annoImmatricolazione, marca, modello,
tipoVeicolo, codiceRider) VALUES
91. ('2016-02-17', 'lancia', 'economica', 'bicicletta', 7),
92. ('2012-06-27', 'lancia', 'economica_motorizzata', 'moto', 7),
93. ('2018-01-02', 'post', 'gres', 'moto', 3),

```



```

94. ('2000-03-12', 'fiat', 'mezza_panda', 'moto', 2),
95. ('2009-12-08', 'piaggio', 'liberty', 'moto', 4),
96. ('2009-12-08', 'piaggio', 'zip', 'moto', 4),
97. ('1997-12-08', 'marin', 'graziella', 'bicicletta', 4),
98. ('2008-04-09', 'trek', 'graziella', 'bicicletta', 5),
99. ('2008-07-11', 'cthulhu', 'squid', 'moto', 1),
100. ('2020-01-18', 'dreaming', 'nightmare', 'bicicletta', 1),
101. ('2020-01-18', 'space', 'invaders', 'bicicletta', 6),
102. ('2020-01-18', 'space', 'milkyway', 'moto', 6),
103. ('2020-01-18', 'volta', 'greenpower', 'bicicletta', 6),
104. ('2020-01-18', 'volta', 'semigreenpower', 'moto', 6);
105.
106.
107. --Popolamento tabella corrierePer
108. INSERT INTO corrierePer(codiceRider, codiceRistorante) VALUES
109. (1, 1),
110. (1, 2),
111. (1, 3),
112. (2, 1),
113. (2, 3),
114. (3, 2),
115. (4, 1),
116. (4, 2),
117. (5, 2),
118. (5, 3),
119. (6, 1),
120. (6, 2),
121. (6, 3),
122. (7, 1);
123.
124.
125. --Popolamento tabella ordine
126. INSERT INTO ordine(codicecarrello, email, codiceristorante,
127. codiceRider, prezzo, dataAcquisto, stato) VALUES
128. (1, 'assistant@live.it', 2, 6, 40, '2021-05-23', 'A' ),
129. (3, 'book@ofsand.com', 2, 32, '20.9', '2022-01-01', 'N' );
130.
131. --Popolamento tabella composizionecarrello
132. INSERT INTO composizionecarrello(codicecarrello, codiceseriale,
133. quantità) VALUES
134. (2, 7, 1),
135. (4, 8, 1),
136. (5, 5, 1);
137.
138. --Popolamento tabella carrello
139. INSERT INTO carrello(DATA, codicecarrello) VALUES
140. ('2021-12-27', 1),
141. ('2021-12-27', 2),
142. ('2021-07-20', 3),
143. ('2021-12-27', 4),
144. ('2021-12-27', 5);
145.
146.
147. --Popolamento tabella fornitura

```

```
148.INSERT INTO fornitura(codiceRistorante, codiceSeriale,  
    quantitaProdotto) VALUES  
149.(1, 1, 120),  
150.(1, 2, 46),  
151.(1, 3, 19),  
152.(1, 4, 21),  
153.(1, 12, 70),  
154.(1, 17, 23),  
155.(1, 27, 37),  
156.(1, 22, 200),  
157.(1, 23, 160),  
158.(2, 5, 9),  
159.(2, 6, 16),  
160.(2, 7, 40),  
161.(2, 8, 12),  
162.(2, 9, 27),  
163.(2, 10, 33),  
164.(2, 11, 70),  
165.(2, 12, 14),  
166.(2, 13, 70),  
167.(2, 14, 48),  
168.(2, 20, 84),  
169.(2, 21, 68),  
170.(2, 19, 18),  
171.(2, 18, 81),  
172.(2, 16, 100),  
173.(3, 15, 32),  
174.(3, 24, 12),  
175.(3, 25, 72),  
176.(3, 26, 48),  
177.(3, 28, 87),  
178.(3, 27, 23),  
179.(3, 29, 31),  
180.(3, 30, 21);  
181.
```

Capitolo V – Glossario

Glossario

	Termine	Descrizione
1	Dati relazionali	Modello logico di rappresentazione o strutturazione dei dati di un database
2	DBMS	Il <i>DBMS</i> (Database Management System) è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database è ospitato su architettura hardware dedicata oppure su semplice computer. Esistono vari DBMS tra cui il <i>DBMS relazionale ad oggetti</i> : un modello di base di dati in cui l'informazione è rappresentata in forma di oggetti come nei linguaggi di programmazione ad oggetti.
3	Schema concettuale	È una tecnica molto nota di progettazione dati indipendente dai dettagli dell'implementazione. Ha lo scopo di esprimere il significato di termini e concetti usati dagli esperti del dominio per discutere il problema, e di trovare le giuste relazioni tra concetti differenti.
4	Class diagram UML	Sono tipi di diagrammi che consentono di descrivere i <i>tipi di entità</i> , con le loro caratteristiche e le eventuali relazioni fra questi tipi.
5	Schema logico	È una fase, della progettazione delle basi di dati, che si occupa di perfezionare lo schema concettuale andando a curare aspetti come: Riduzione delle ridondanze, Normalizzazione degli attributi, Eliminazione delle generalizzazioni.
6	Chiavi primarie	È un insieme di uno o più attributi che permette di individuare univocamente un record o tupla o ennupla in una tabella o relazione.
7	Chiavi esterne	È un insieme di uno o più attributi che fanno riferimento a una chiave di un'altra tabella, permettendo in tal modo di esplicitare relazioni di tipo.