Flash Delivering

1 Abstract

“Flash Delivering” è un’app di food delivering che conta più di 100 000 utenti e ha deciso di rinnovare il proprio database.
Con “Flash Delivering” è possibile ordinare cibo da qualsiasi ristorante, potendo scegliere se ricevere l’ordine a casa propria in brevissimo tempo oppure utilizzare l’applicazione solamente per ordinare il cibo desiderato e ritirarlo personalmente.
L’utente per registrarsi sull’applicazione dovrà inserire i suoi dati personali tra cui gli indirizzi per le spedizioni e i metodi di pagamento, una volta registrato potrà effettuare acquisti presso qualsiasi ristorante ed inviare delle recensioni successive agli acquisti.
Ogni ristorante interessato ad effettuare vendite su “Flash Delivering” rende disponibile nell’applicazione un menù e i suoi orari di apertura, alcuni ristoranti scelgono di affiliarsi a “Flash Delivering” fornendo un piccolo sconto a chi ordina tramite l’app e assumendo dei rider che svolgono le consegne solo per loro garantendo tempi di consegna più rapidi; degli altri ristoranti invece se ne occupano dei rider che effettuano consegne per qualsiasi attività ricevendo un compenso per ogni consegna effettuata.
Flash Delivering si prende anche cura della salute dei clienti tenendo traccia di ogni allergene presente nei piatti ordinabili tramite l’applicazione.

2 Analisi dei requisiti

2.1 Descrizione testuale

Nella base di dati sono presenti i dati degli **utenti** registrati all’applicazione, di ogni utente sono noti:

- Nome
- Cognome
- Indirizzo mail
- password
- Numero di telefono
- Indirizzi per le consegne
- Metodi di pagamento

Un utente può diventare “premium” tramite un abbonamento mensile. L’utente premium non paga i costi di spedizione, inoltre ogni utente può salvare i dati relativi alle sue allergie.
I **metodi di pagamento** utilizzati dall’utente devono memorizzare:

- Numero
- Circuito (Mastercard, Visa, …)
- Scadenza
- Intestatario

I pagamenti vengono autorizzati attraverso un sito esterno, rilevato automaticamente dal circuito.
Ogni **Ristorante** deve fornire le informazioni riguardo a:

- Nome
- Numero di telefono
- Indirizzo email
- Indirizzo: (Stato, Città, Via, numero civico, CAP)
- Giorni e orari di apertura
- Piatti ordinabili
- Recensione
- Valutazione media

I ristoranti possono anche essere affiliati all’applicazione, in quel caso devono dichiarare la percentuale di sconto che verrà applicata ad ogni ordine propria di quel ristorante (almeno del 5%).
Di ogni **Piatto**, inoltre, si vuole conoscere:

- Nome
- Prezzo in uno specifico ristorante
- Ristorante in cui viene preparato
- Allergeni presenti

La base di dati contiene anche le informazioni relative ai **rider** ovvero:

- Nome

- Cognome
- Codice fiscale
- Indirizzo mail
- Numero di telefono
- Codice IBAN

Alcuni rider lavorano come dipendenti dei ristoranti affiliati a Flash Delivery, di questi si vuole conoscere lo stipendio mensile e il ristorante per cui lavorano. I rider che non appartengono alla prima categoria invece effettuano consegne presso qualsiasi ristorante guadagnando dalle spese di spedizione.
Ogni **Ordine** può riferirsi ad un solo ristorante e deve memorizzare le informazioni riguardo:

- Cliente
- Ristorante
- Piatti ordinati
- Metodo di pagamento utilizzato
- Data e ora del pagamento
- Importo pagato
- Recensione del ristorante (opzionale)

Gli ordini possono essere ritirati dal cliente, in quel caso deve essergli fornita la data e l'ora del ritiro, oppure possono essere spediti da un rider.
Quando viene richiesta una spedizione si vogliono memorizzare le informazioni riguardo:

- Indirizzo di spedizione
- Rider assegnato alla consegna
- Costo di spedizione
- Fascia oraria di arrivo
- Orario effettivo di arrivo

Per ogni ordine effettuato il cliente può scrivere una **recensione** che verrà salvata nella base di dati, la recensione relativa al ristorante contiene un punteggio che va da 0 a 5. Se l'ordine è stato effettuato tramite consegna la recensione includerà anche un punteggio da 0 a 5 relativo al rider. Infine il cliente può scegliere se aggiungere un testo di massimo 500 caratteri per completare la sua recensione.

2.2 Glossario dei termini

Termine	**Descrizione**	**Collegamenti**
Cliente	Utente registrato nell'applicazione Sinonimi: utente	Indirizzo, Carta di credito, Allergene, Ordine
Premium	Utente che ha sottoscritto un abbonamento mensile per non pagare i costi di spedizione	Entità figlia di cliente
Carta di credito	Metodo di pagamento utilizzato per pagare gli ordini. Ne fanno parte anche carte di debito e carte prepagate	Cliente, Ordine
Circuito	Circuito di pagamento della carta di credito (Visa, Mastercard, Maestro, ecc…)	Attributo di Carta di Credito
Indirizzo	Indirizzo associato ad un cliente per le spedizioni	Cliente, Ordine
Importo	Spesa totale relativa ad un ordine (con eventuali sconti già applicati). Non comprende i costi di spedizione	Attributo di Ordine
Ritiro	Ordine in cui è previsto il ritiro in ristorante da parte dell'utente	Entità figlia di Ordine
Spedizione	Ordine in cui l'utente ha richiesto la spedizione da parte di un rider presso uno dei suoi indirizzi registrati	Indirizzo, Rider
CostoSpedizione	Costo della spedizione (viene registrato anche se il cliente è premium, per calcolare il guadagno del rider)	Attributo di Spedizione
Dataora_min/Dataora_max	Intervallo orario in cui si prevede l'arrivo dell'ordine effettuato dal cliente all'indirizzo specificato	Attributo di Spedizione
Ristorante	Locale in cui è possibile ordinare cibo da asporto	Ordine, Piatto, Giorno, Recensione

Giorno	Data e orario di apertura e chiusura di un ristorante	Ristorante
Ristorante Affiliato	Ristorante affiliato a “Flash Delivering”	Entità figlia di Ristorante. Dipendente
Percentuale_sconto	Percentuale di sconto applicata all’importo totale degli ordini (spedizione esclusa) dai ristoranti affiliati	Attributo di Ristorante Affiliato
Stelle	Punteggio espresso in numero di stelle (0-5) riferito ad una recensione	Attributi delle relazioni Voto
Recensione	Recensione relativa alla qualità del cibo ordinato e dell’ordine in generale, associata ad un ristorante	Ordine, Ristorante
Recensione Rider	Recensione relativa alla velocità e qualità della sola spedizione. Complementare alla recensione. Presente se e solo se l’ordine è una spedizione	Entità figlia di Recensione. Rider
Rider	Persona addetta alla consegna degli ordini in cui è stata richiesta la spedizione	Spedizione, Recensione Rider
Dipendente	Rider che lavora presso un ristorante affiliato a “Flash Delivering”. Può effettuare soltanto consegne relative al ristorante in cui lavora	Entità figlia di Rider. Ristorante affiliato
Libero	Rider che può effettuare consegne per qualsiasi ristorante e percepisce il costo della spedizione	Entità figlia di Rider

2.3 Operazioni

OPERAZIONE	**TIPO**	**FREQUENZA**
Ricerca del numero di ordini di un ristorante in una certa fascia oraria	L	10000 al giorno
Inserimento di un nuovo cliente	S	1000 al giorno
Inserimento nuovo ordine	S	10000 al giorno
Annullamento ordine	S	100 al giorno
Inserimento nuova recensione	S	100 al giorno
Stampare l’elenco delle recensioni di un ristorante	L	5000 al giorno
Visualizzare giorni di apertura di un ristorante	L	10000 al giorno
Aggiornare il menù di un ristorante	S	1000 al mese
Calcolare il punteggio medio di un rider	L	3000 al giorno
Viusalizzare il punteggio medio di un ristorante	L	25000 al giorno
Calcolare il guadagno mensile di un rider libero	L	1000 al mese
Calcolare il guadagno giornaliero di un ristorante	L	5000 al giorno

3 Progettazione Concettuale

3.1 Lista entità

Se non specificato l’attributo è NOT NULL

- Cliente:
  - E-mail: varchar (100) primary key
  - Nome: varchar (50)
  - Cognome: varchar (50)
  - Telefono: varchar (20)
  - Password: varchar (30)

  L’entità cliente si specializza in una sottocategoria con una generalizzazione parziale:
  - Cliente premium
- Indirizzo:
  - Stato: varchar (20)
  - Città: varchar (50)
  - CAP: char (5) primary key
  - Via: varchar (50) primary key

    - N_civico: varchar (10) primary key
    - Cliente: varchar (100) primary key
- CartaDiCredito:
    - Numero: char (16) primary key
    - Circuito: varchar (20)
    - Scadenza: date
    - Intestatario: varchar (50)
- Allergene:
    - Nome: varchar (50) primary key
- Ordine:
    - ID: int primary key
    - DataOraPagamento: datetime
    - Importo: decimal (7,2)

  L'entità ordine si specializza in due sottocategorie con una generalizzazione totale:
    - Ritiro:
        - DataOra_ritiro: datetime, > DataOraPagamento
    - Spedizione:
        - CostoSpedizione: decimal (5,2)
        - DataOra_min: datetime, > DataOraPagamento
        - DataOra_max: datetime, > DataOra_min
        - DataOraArrivo
- Ristorante:
    - Nome: varchar (50)
    - Telefono: varchar (20)
    - E-mail: varchar (100)
    - PuneggioMedio: decimal (2,1), >= 0, <= 5, può essere NULL
    - Indirizzo: attributo composto
        - Stato: varchar (20) primary key
        - Città: varchar (50) primary key
        - CAP: char (5) primary key
        - Via: varchar (50) primary key
        - N_civico: varchar (10) primary key

  L'entità ristorante si specializza in una sottocategoria con una generalizzazione parziale:
    - Ristorante affiliato:
        - PercenualeSconto: int, > 5, <= 100
- Piatto:
        - Nome: varchar (50) primary key
        - Id_ristorante: int primary key
        - Prezzo: decimal (7,2)

1. Giorni:
    - Data: date primary key
    - Ora_apertura: time primary key
    - Ora_chiusura: time, > Ora_apertura
    - Id_ristorante: int primary key
2. Rider:
    - CF: char (16) primary key
    - Cognome: varchar (50)
    - Nome: varchar (50)
    - Mail: varchar (100)
    - Telefono: varchar (20)
    - IBAN: char (27)

  L'entità rider si specializza in due sottocategorie con una generalizzazione totale:
    - Dipendente:
        - Stipendio: decimal (7,2)
    - Libero
- Recensione:
    - ID_ordine: int primary key

- Testo: varchar (500) può essere NULL

L'entità recensione si specializza in una sottocategoria con una generalizzazione parziale

- Recensione_rider

3.2 Tabella delle relazioni

Relazione	Entità coinvolte	Descrizione	Attributi
Abitazione	Cliente (0,N) Indirizzo (1,1)	Un cliente può registrare più indirizzi o non registrarne nessuno se non richiede spedizioni, un indirizzo si riferisce a un solo cliente	Nessuno
MetodoPagamento	Cliente (1,N) Carta di credito (1,N)	Un cliente deve registrare uno o più metodi di pagamento, una carta di credito può essere memorizzata da uno o più utenti	Nessuno
Allergia	Cliente (0,N) Allergene (0,N)	Un cliente può registrare zero o più allergie, un allergene può essere associato a zero o più clienti	Nessuno
Presente	Allergene (0,N) Piatto (0,N)	Un allergene può essere presente in zero o più piatti, un piatto può contenere zero o più allergeni	Nessuno
Acquisto	Cliente (0,N) Ordine (1,1)	Un cliente può effettuare zero o più ordini, un ordine è effettuato da un solo utente	Nessuno
Transazione	Carta di credito (0,N) Ordine (1,1)	Una carta di credito può essere utilizzata per zero o più ordini, un ordine è sempre associato ad un metodo di pagamento	Nessuno
Ordinato	Ordine (1,N) Piatto (0,N)	Un ordine deve contenere uno o più piatti, un piatto può essere stato ordinato zero o più volte. La quantità di uno stesso piatto ordinato può essere più di uno	Quantità: int > 0
Presso	Ordine (1,1) Ristorante (0,N)	Un ordine deve essere fatto presso uno specifico ristorante, presso un ristorante possono essere stati fatti zero o più ordini	Nessuno
Valutazione	Ordine (0,1) Recensione (1,1)	Un ordine può avere una recensione, una recensione deve riferirsi ad un solo ordine	Nessuno
IndirizzoSpedizione	Spedizione (1,1) Indirizzo (0,N)	Una spedizione deve essere fatta presso uno specifico indirizzo, ad un indirizzo possono essere stati spediti zero o più ordini	Nessuno
Incarico	Spedizione (1,1) Rider (0,N)	Una spedizione deve memorizzare il rider che l'ha effettuata, un rider può aver effettuato zero o più spedizioni	Nessuno
Menù	Piatto (1,1) Ristorante (1,N)	Un piatto deve registrare il ristorante in cui viene preparato, un ristorante ha almeno un piatto nel suo menù	Nessuno
Apertura	Ristorante (0,N) Giorni (1,1)	Un ristorante ha zero o più giorni di apertura, un giorno è di apertura per uno specifico ristorante	Nessuno
Voto	Ristorante (0,N) Recensione (1,1)	Un ristorante può avere zero o più recensioni, una recensione si riferisce ad uno specifico ristorante	Stelle: int >= 0, <= 5
Lavoro	Ristorante affiliato (1,N) Dipendente (1,1)	Un ristorante affiliato deve avere almeno un rider dipendente, un rider dipendente lavora in uno specifico ristorante affiliato	Nessuno
Voto	Rider (0,N) Recensione rider (1,1)	Un rider può avere zero o più recensioni, una recensione di un rider deve essere associata al rider da recensire	Stelle: int >= 0, <= 5

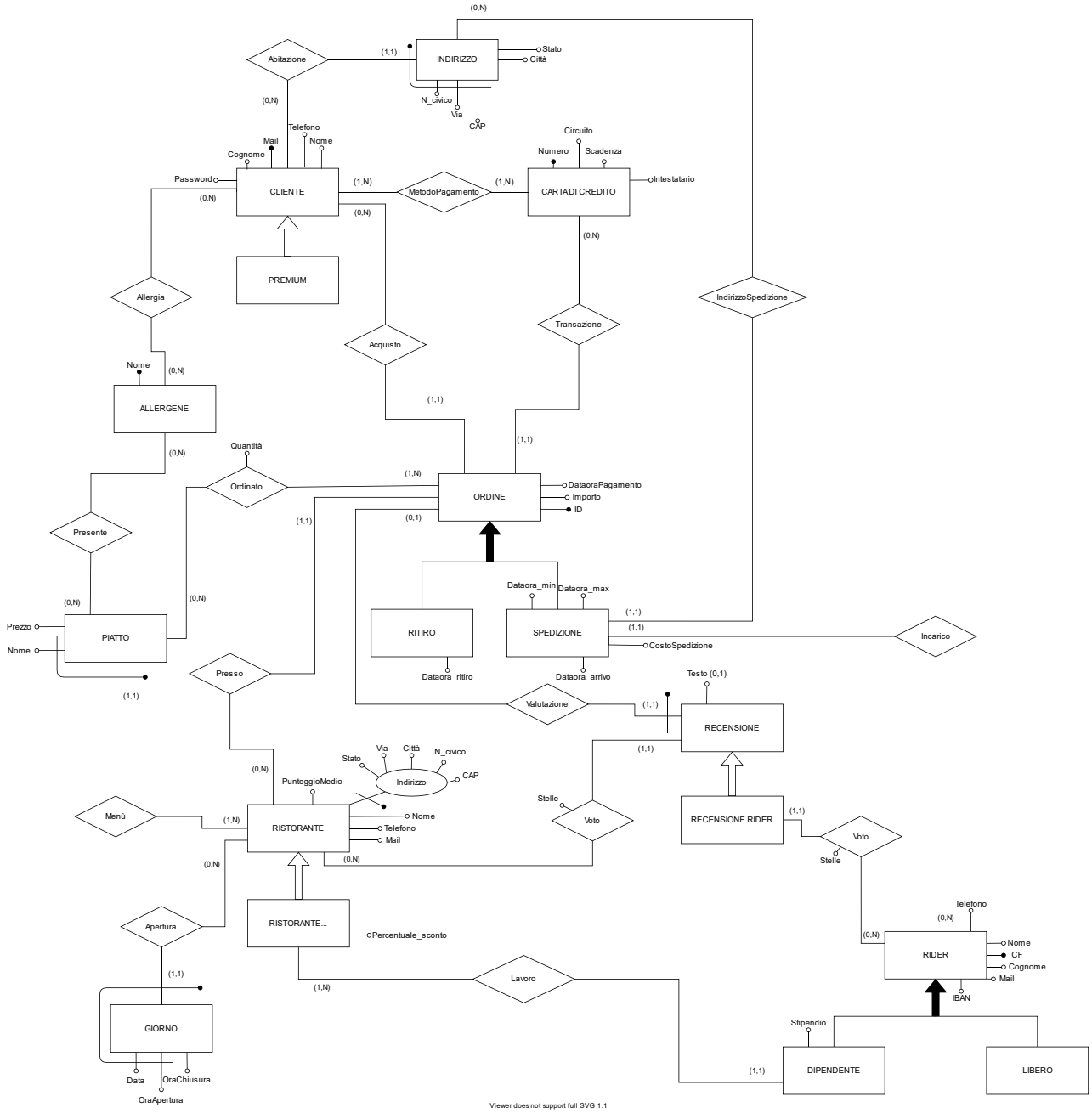
Vincoli non rappresentabili tramite schema E-R:

- Un rider dipendente effettua consegne solo per il ristorante presso cui lavora.
- Una recensione deve comprendere anche la recensione del rider se e solo se si riferisce a un ordine che è una spedizione.

Vincoli di derivazione:

- L’attributo PunteggioMedio dell’entità ristorante è uguale alla media delle stelle presenti nelle recensione relative a quel ristorante.
- L’attributo importo dell’entità ordine è uguale alla somma dei prezzi dei piatti presenti in quell’ordine moltiplicati per la loro quantità al quale viene sottratto l’eventuale sconto.

Schema Concettuale



4 Progettazione Logica

4.1 Ristrutturazione

4.1.1 Analisi delle ridondanze

L’attributo “**Importo**” dell’entità “Ordine” è ridondate in quanto può essere calcolato sommando i prezzi dei piatti ordinati moltiplicati per la quantità con cui sono stati ordinati, però verrebbe utilizzato in solamente tre

operazioni: 1. calcolo del guadagno giornaliero di un ristorante, 2. inserimento di un nuovo ordine e 3. visualizzazione dello storico degli ordini di un cliente.
Operazione 1: viene eseguita una sola volta al giorno per ogni ristorante, quindi utilizzerebbe l'attributo una e una sola volta
Operazione 2: viene eseguita una sola volta, ovvero quando viene eseguito un ordine specifico
Operazione 3: è un'operazione poco frequente, quindi non è conveniente introdurre ridondanza per questa operazione
In conclusione l'attributo ridondante occuperebbe memoria senza portare alcun beneficio, con l'aggiunta del rischio di inconsistenza, quindi verrà eliminato.

L'attributo "**PunteggioMedio**" dell'entità Ristorante è ridondante in quanto può essere calcolato facendo la media dei punteggi relativi al ristorante

Concetto	**Tipo**	**Volume**
Ristorante	E	5000
Recensione	E	250000
Voto	R	250000

- OPERAZIONE 1: inserimento di una nuova recensione (100 volte al giorno)
- OPERAZIONE 2: visualizzare punteggio medio di un ristorante (25000 volte al giorno)

PRESENZA RIDONDANZA:
operazione 1:
un ristorante ha in media 250000 / 5000 = 50 recensioni

Concetto	Costrutto	Accessi	Tipo	
Voto	Relazione	1	S	x 100
Voto	Relazione	50	L	x 100
Ristorante	Entità	1	S	x 100

operazione 2:

Concetto	Costrutto	Accessi	Tipo	
Ristorante	Entità	1	L	x25000

considerando che gli accessi in scrittura costano il doppio di quelli in lettura
costo giornaliero = 100*2+50*100+100*2+25000 = 30400
ASSENZA RIDONDANZA:
operazione 1

Concetto	Costrutto	Accessi	Tipo	
Voto	Relazione	1	S	x100

operazione 2

Concetto	Costrutto	Accessi	Tipo	
Ristorante	Entità	50	L	x25000

costo giornaliero = 100*2 + 50*25000 =1250200
In conclusione l'analisi della ridondanza dimostra che conviene mantenere l'attributo "PunteggioMedio" che verrà aggiornato con l'aggiunta di ogni recensione.

4.1.2 Eliminazione delle generalizzazioni

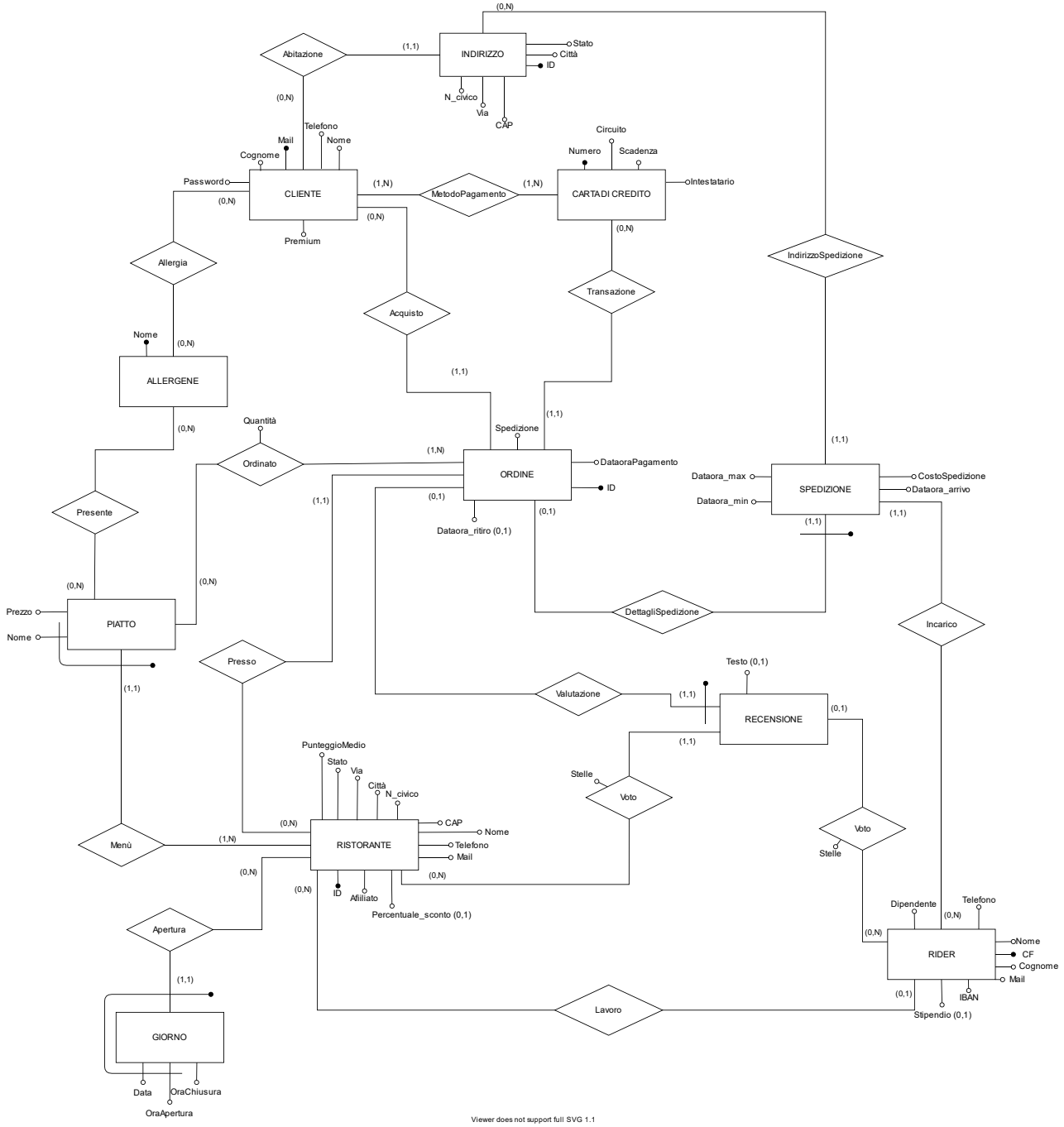
Generalizzazione	Risoluzione
Cliente ⇐ Premium	L'entità Premium viene accorpata in Utente, perché il concetto della generalizzazione può essere facilmente espresso con un attributo booleano Premium
Ristorante ⇐ Affiliato	L'entità Affiliato viene accorpata in Ristorante perché gli accessi alle due entità sono contestuali, viene aggiunto un attributo a Ristorante: • Affiliato: bool, NOT NULL L'attributo Percentuale_sconto diventa opzionale La relazione Lavoro ora coinvolge le entità Ristorante (0,N) e Dipendente (1,1)

Ordine ← Spedizione, Ritiro	Per eliminare questa generalizzazione si utilizza una soluzione “ibrida”: • L’entità Ritiro viene accorpata in Ordine in quanto gli accessi sono contestuali • La generalizzazione di Spedizione viene sostituita con una relazione “DettagliSpedizione” dato che gli accessi al padre non implicano accessi al figlio e viceversa, la relazione è chiave dell’entità Spedizione
Rider ← Dipendente, Libero	Le entità Dipendente e Libero vengono accorpate in Rider perché rappresentano lo stesso concetto, viene aggiunto un attributo a Rider: • Dipendente: bool, NOT NULL L’attributo stipendio diventa opzionale La relazione Lavoro ora coinvolge le entità Ristorante (0,N) e Rider (0,1)
Recensione ⇐ RecensioneRider	L’entità RecensioneRider viene accorpata in Recensione in quanto gli accessi alle due entità sono contestuali, la relazione Voto che prima collegava RecensioneRider (1,1) a Rider (0,N) ora collega le entità Recensione (0,1) a Rider (0,N) mantenendo sempre l’attributo stella

4.1.3 Scelta degli identificatori primari

Per l’entità Ristorante era stato scelto di utilizzare l’indirizzo, un attributo composto, come identificatore primario, ma in questo modo le tabelle Piatto, Giorno, Ordine, Rider e Recensione avrebbero dovuto memorizzare i cinque campi dell’indirizzo come chiave esterna, quindi è stato deciso di introdurre l’attributo ID come chiave primaria. È stato inoltre deciso di introdurre un altro attributo ID nell’entità indirizzo, dato che la tabella Spedizione avrebbe dovuto memorizzare quattro campi come chiave esterna, perché collegata a Indirizzo con una relazione 1 a N.

Schema E-R ristrutturato



4.2 Creazione delle tabelle (A→B indica che B è chiave esterna di A)
Cliente (Mail, Password, Nome, Cognome, Telefono, Premium)
Indirizzo (ID, Via, NCivico, CAP, Città, Stato, Cliente → Cliente.Mail)
Allergene (Nome)
Allergie_clienti (Allergene → Allergene.Nome, Cliente → Cliente.Mail)
Carta_di_credito (Numero, Circuito, Scadenza, Intestatario)
Metodi_pagamento (Cliente → Cliente.Mail, CartaDiCredito → Carta_di_credito.Numero)
Piatto (Nome, Ristorante → Ristorante.ID, Prezzo)
Allergie_piatti (Piatto → Piatto.Nome, Ristorante → Piatto.Ristorante, Allergene → Allergene.Nome)
Ordine (ID, DataoraPagamento, DataoraRitiro, Spedizione, Cliente → Cliente.Mail, CartaDiCredito → Carta_di_credito.Numero, Ristorante → Ristorante.ID)
Piatti_ordinati (Piatto → Piatto.Nome, Ristorante → Piatto.Ristorante, Ordine → Ordine.ID, Quantita)
Spedizione (Ordine → Ordine.ID, DataoraMax, DataoraMin, DataoraArrivo, CostoSpedizione, Indirizzo → Indirizzo.ID, Rider → Rider.CF)

Ristorante (ID, Nome, Affiliato, PercentualeSconto, Mail, Telefono, CAP, Via, NCivico, Città, Stato, PunteggioMedio)
Recensione (Ordine →Ordine.ID, Testo, Ristorante → Ristorante.ID, Rider → Rider.CF, StelleRistorante, StelleRider)
Giorno (Ristorante → Ristorante.ID, Data, OraApertura, OraChiusura)
Rider (CF, Nome, Cognome, Mail, Telefono, Dipendente, IBAN, Stipendio, Ristorante → Ristorante.ID)

5 Query e indici

5.1 Query

1. Mostrare tutte le recensioni di un ristorante, nell'esempio vengono visualizzate quelle del ristorante da Pino (ID=6).
La query può essere ottimizzata attraverso l'utilizzo dell'indice Ristoranti_recensioni.

```
select stelleristorante, testo, rider, stellerider
from recensione rec, ristorante ris
where rec.ristorante = ris.id and ris.Id = 6
order by stelleristorante desc;
```

stelleristorante	testo	rider	stellerider
5	Ristorante ottimo!		
5	Il cibo era buonissimo, consiglio questo ristorante a tutti!		
5	cibo ottimo, consegna lenta	GRGCLN87E34T365G	1
4			
4			
4		GRGCLN87E34T365G	2
4			
4			
3		GLARSA03B62J897Z	0
3		GLARSA03B62J897Z	5
1	Cibo scadente, sconsigliatissimo		
0			
0	Cibo andato a male, la consegna era decente	GRGCLN87E34T365G	3

2. Per ogni cliente visualizzare i piatti che ha ordinato (nome, ristorante) a cui è allergico mostrando l'allergene in questione.

```
select c.mail, p.nome as piatto, r.nome as ristorante,
    a.nome as allergene
from cliente c, allergene a, ordine o, allergie_clienti ac,
    piatti_ordinati po, piatto p, allergie_piatti ap, ristorante r
where c.mail = o.cliente and c.mail = ac.cliente and
    o.id = po.ordine and po.piatto = p.nome and
    po.ristorante = p.ristorante and ap.piatto = p.nome and
    ap.ristorante = p.ristorante and ac.allergene = a.nome and
    ap.allergene = a.nome and r.id = o.ristorante
group by c.mail, p.nome, a.nome, r.nome;
```

mail	piatto	ristorante	allergene
marco.berlusconi@hotmail.com	Pasta_al_pesto	da Mina	Glutine
marco.berlusconi@hotmail.com	Pasta_al_pomodoro	da Mina	Glutine
marco.neri@hotmail.com	Pasta_al_pomodoro	da Marco	Glutine
marco.neri@hotmail.com	Pasta_alla_carbonara	da Marco	Glutine
ugo.neri@hotmail.com	Pasta_alla_carbonara	da Pino	Glutine

3. Visualizzare il guadagno dei rider nel mese di maggio 2020, tenendo conto che quelli dipendenti hanno uno stipendio fisso mentre gli altri guadagnano in base alle consegne.

```
select cf, nome, cognome, dipendente,stipendio as guadagno_maggio2020
from rider
where dipendente=true
union
select cf, nome, cognome,dipendente, sum(costospedizione)
from rider r, spedizione s
where r.dipendente=false and s.rider=r.CF and s.DataoraArrivo >= '2020-05-
01' and s.DataoraArrivo <= '2020-05-31'
group by r.cf
order by guadagno_maggio2020 desc;
```

cf	nome	cognome	dipendente	guadagno_maggio2020
CRLCNT35T72C638R	Carlo	Conti	si	4000.00
UGOFSL64D65G537R	Ugo	Foscolo	si	3000.00
LTRLMB65F83T395O	Elettra	Lamborghini	si	2700.00
EZIADT45C10H374L	Ezio	Auditore	si	2500.00
MNLMOR96R27S309Y	Emanuela	Mori	si	2000.00
RBTRBT73P38P104S	Roberto	Ruberti	si	1700.00
MTTBNC34A66V527X	Matteo	Bianchi	si	1500.00
GRGCLN87E34T365G	Giorgio	Chiellini	no	32.18
GLARSA03B62J897Z	Giulia	Rosa	no	13.68
MRZRSS52R27T296A	Maurizio	Rossi	no	2.95

4. Stampare le mail dei clienti che hanno un determinato piatto come preferito e quante volte lo hanno ordinato (nell'esempio il kebab)

```
drop view if exists piatti_clienti;
create view piatti_clienti as
select c.mail,po.piatto,sum(quantita) as tot
from cliente c, ordine o, piatti_ordinati po
where c.mail=o.cliente and o.id=po.ordine
group by c.mail,po.piatto;

select p1.mail as cliente, p1.tot as n_ordinazioni
from piatti_clienti p1
where p1.piatto = 'Kebab'
except
select p1.mail, p1.tot
from piatti_clienti p1, piatti_clienti p2
where p1.mail=p2.mail and p1.tot<p2.tot
order by n_ordinazioni desc;
```

cliente	n_ordinazioni
gianni.bruni@hotmail.com	6
alberto.berlusconi@hotmail.com	5
massimiliano.totti@libero.com	5
massimiliano.berlusconi@gmail.com	5
francesco.cassano@hotmail.com	5
alberto.totti@icloud.com	4
massimiliano.garibaldi@libero.com	4
marco.berlusconi@hotmail.com	4
lucia.cassano@hotmail.com	4

5. Mostrare i dieci clienti che hanno speso più soldi su Flash Delivering (la query tiene conto delle spese relative al cibo calcolando gli sconti dei ristoranti affiliati e delle spese di spedizione che sono azzerate per i clienti premium)

```
drop view if exists importo_tot_clienti;
create view importo_tot_clienti as
select c.mail, sum(
    p.prezzo * po.quantita * coalesce(r.percentualesconto * 0.01, 1)
) as tot
from cliente c, ordine o, piatti_ordinati po, piatto p, ristorante r
where c.mail = o.cliente and po.ordine = o.id and
    po.piatto = p.nome and po.ristorante = p.ristorante and
    p.ristorante = r.id
group by c.mail;

drop view if exists spese_sped_tot_clienti;
create view spese_sped_tot_clienti as
select c.mail, coalesce(sum(s.costospedizione), 0) as tot
from cliente c, ordine o left join spedizione s on (s.ordine = o.id)
where o.cliente = c.mail and c.premium = false
group by c.mail;

select c.nome, c.cognome,
    round(i.tot, 2) as importo_pagato,
    coalesce(s.tot, 0) as tot_spedizione,
    round(i.tot + coalesce(s.tot, 0), 2) as totale_pagato
from cliente c,
    importo_tot_clienti i left join spese_sped_tot_clienti s
    on (i.mail = s.mail)
where c.mail = i.mail
order by totale_pagato desc
limit 10;
```

nome	cognome	importo_pagato	tot_spedizione	totale_pagato
francesco	bianco	355.03	14.13	369.16
francesco	cassano	251.80	0	251.80
ugo	neri	233.87	14.02	247.89
maria	rossi	229.16	3.22	232.38
maria	neri	222.40	0	222.40
mario	totti	211.70	0	211.70
carla	garibaldi	198.65	0	198.65
marco	totti	189.48	0	189.48
maria	berlusconi	189.01	0	189.01
francesco	rossi	179.60	1.61	181.21

6. Mostrare i ristoranti aperti in un determinato momento che preparano solo piatti a cui uno specifico cliente non è allergico (la query dell'esempio utilizza 2020-05-12 17:45 come data e ora attuale e Maria Garibaldi come cliente)

```
select r.nome as ristorante,
g.oraapertura as orario_di_apertura, g.orachiusura as orario_di_chiusura
from ristorante r, giorno g, (
    select c.mail, r.id
    from cliente c, ristorante r
    except (select ac.cliente, ap.ristorante
        from allergie_clienti ac, allergie_piatti ap
        where ac.allergene = ap.allergene)
) as cr
where r.id = g.ristorante and g.data = '2020-05-12' and
    g.oraapertura <= '17:45' and g.orachiusura >= '17:45' and
    r.id = cr.id and cr.mail = 'maria.garibaldi@icloud.com'
order by cr.mail, r.id;
```

ristorante	orario_di_apertura	orario_di_chiusura
da Piero	17:40:00	21:30:00
da Marco	17:10:00	21:40:00
da Paolo	15:00:00	18:20:00
da Giulia	14:00:00	17:50:00
da Mina	08:30:00	18:00:00

5.2 Indici

Le recensioni dei ristoranti sono un dato che viene utilizzato molto spesso in lettura, perché vengono visualizzate in automatico nella pagina del ristorante nell'applicazione, mentre l'inserimento di una recensione è molto meno frequente. Inoltre la visualizzazione di tutte le recensioni di un ristorante avviene in ordine di valutazione. Quindi si decide, ipotizzando uno sviluppo su larga scala delle recensioni, di indicizzare gli attributi Ristorante e StelleRistorante nell'indice Ristoranti_recensioni.

```
create index Ristoranti_recensioni on Recensione (Ristorante, StelleRistorante);
```

6 Codice C++

6.1 Descrizione dell'utilizzo del codice

Il codice C++ per l'esecuzione delle query consiste in un unico file .cpp, che va compilato attraverso il comando `g++ -L dependencies/lib -l pq "Codice C++.cpp" -o Queries`.
Prima di poter compilare il file è necessario copiare i file libpq.dll e libpq.lib in "./dependencies/lib" e copiare libpq-fe.h, pg_consig_ext.h e postgres_ext.h in "./dependencies/include", dove "./" è il percorso della directory che contiene il file .cpp.
Per eseguire il codice, basta avviare l'eseguibile "Queries". La prima cosa che il programma chiede è la password per accedere al database (il database deve chiamarsi esattamente "Flash Delivering"), successivamente, se riesce ad accedere correttamente al database, mostrerà a schermo la lista delle query, identificate da un numero da 1 a 6, eseguibili all'interno del programma. Per eseguire una query bisogna inserire da tastiera il numero della query scelta, mentre per terminare l'esecuzione del programma va inserito '0'. Alcune query (la numero 1, 4 e 6) richiedono l'inserimento di alcuni parametri da pate dell'utente:
1. Viene mostrata la lista di tutti i Ristoranti disponibili con il loro relativo ID. L'utente deve inserire l'identificativo del Ristorante scelto.
4. Il programma chiede di inserire il nome di un piatto (il primo carattere deve essere maiuscolo e gli spazi vanno sotituiti con underscore es. Pasta_alla_carbonara).
6. Questa query richiede 3 parametri: la data e l'ora del momento in cui si vuole effettuare la ricerca e la mail dell'utente con cui la si effettua. Il codice c++ di questa query inoltre richiede all'utente l'inserimento della propria password di Flash Delivering, verificandone la correttezza e mettendo a disposizione un massimo di 5 tentativi per l'inserimento. (esempio parametri per eseguire la query: 2020-05-12, 17:45, maria.garibaldi@icloud.com, teamdcjb)

6.2 Documentazione del codice

Funzioni utilizzate dal codice:

```
PGconn* connect(const char* host, const char* user, const char* db, const char* pas
s, const char* port)
```

Ritorna una connessione al database, utilizzando i paramteri passati come credenziali di accesso. Se la connessione non va a buon fine mostra un messaggio di errore e termina il programma.

```
PGresult* execute(PGconn* conn, const char* query)
```

Esegue una query passata come stringa ritornandone il risultato.
Se l'esecuzione non va a buon fine mostra un messaggio di errore e termina il programma.

```
void printQuery(PGresult* res)
```

Stampa a schermo sotto forma di tabella il risultato res di una query.
La funzione è in grado di gestire la dimensione delle colonne automaticamente.

```
void printLine(int campi, int* maxChar)
```

Funzione ausiliaria di printQuery che stampa una riga di separazione per la tabella

```
char* chooseParam(PGconn* conn, const char* query, const char* table)
```

Viene utilizzata per mostrare un elenco di valori da una query per poter sceglierne uno (viene usata per scegliere il ristorante nella query 1)