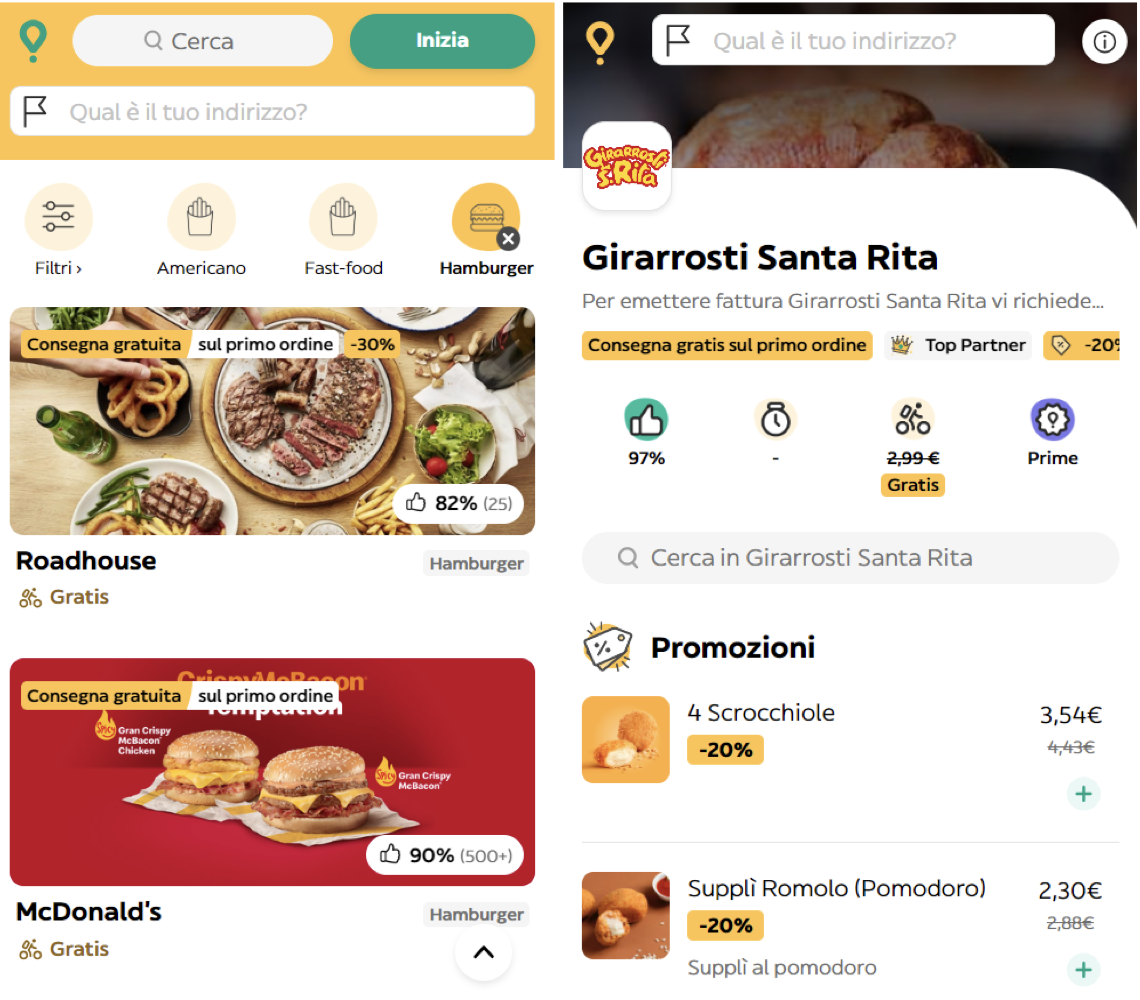
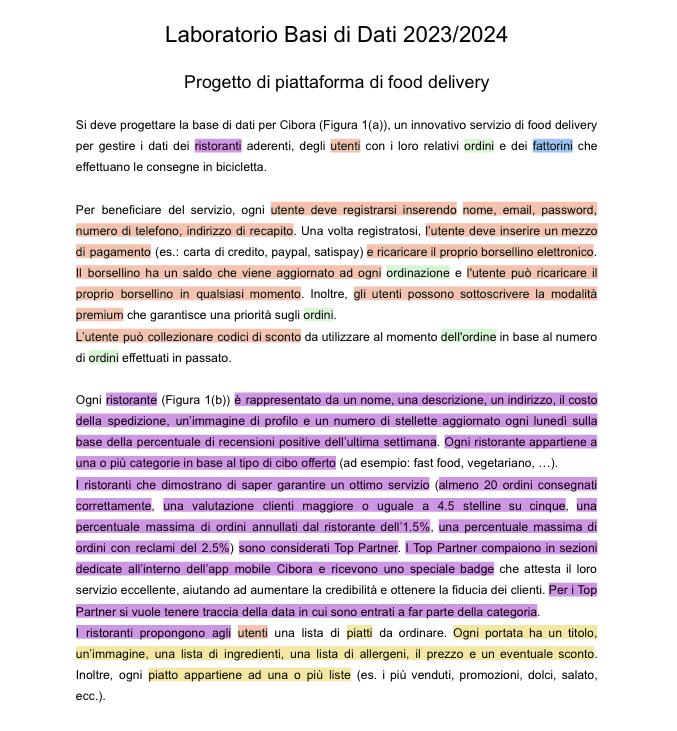
RELAZIONE PROGETTO BASE DI DATI 2023/24

Abourida Zakaria - 950120 - T2

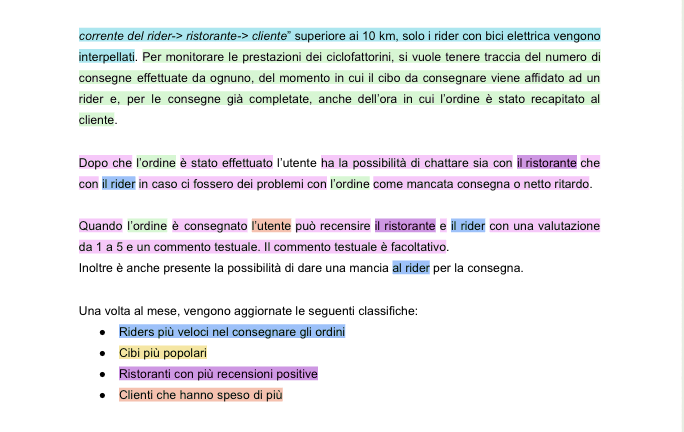


# PROGETTAZIONE CONCETTUALE

## 1.1) Requisiti iniziali







## 1.2) Glossario dei termini

|  |  |  |  |
| --- | --- | --- | --- |
| **TERMINE** | **DESCRIZIONE** | **SINONIMI** | **COLLEGAMENTI** |
| RISTORANTE | Locale pubblico dove vengono serviti piatti e bevande ai clienti. |  | UTENTE,  PIATTO,  ORDINE,  RECENSIONE |
| UTENTE | Cliente che decide di fare gli ordini | Clienti | ORDINE,  PIATTO,  RISTORANTE,  RECENSIONE |
| RIDER | Persona incaricata a portare l’ordine del ristorante al proprio cliente | Fattorino | MEZZO,  ORDINE,  UTENTE,  RECENSIONE |
| ORDINE | Richiesta di determinate pietanze da asporto | Consegna | UTENTE,  RISTORANTE,  PIATTO,  RIDER |
| PIATTO | Il cibo preparato dal ristorante | Portata, Pietanza | UTENTE,  ORDINE,  RISTORANTE |
| RECENSIONE | Messaggio da parte dell’utente per il ristorante e/o il rider | Reclamo | UTENTE,  RISTORANTE,  RIDER, |
| MEZZO | Veicolo utilizzato dal rider per fare le consegne |  | RIDER |

## 

## 1.3) Requisiti rivisti e strutturati in gruppi di frasi omogenee

#### **Requisiti rivisti:**

Si deve progettare la base di dati per Cibora (Figura 1(a)), un innovativo servizio di food delivery per gestire i dati dei **ristoranti** aderenti, degli **utenti** con i loro relativi **ordini** e dei ~~fattorini~~ **rider** che effettuano le consegne in bicicletta.

Per beneficiare del servizio, ogni **utente** deve registrarsi inserendo nome, email, password, numero di telefono, indirizzo di recapito. Una volta registratosi, l’**utente** deve inserire un mezzo di pagamento ~~(es.: carta di credito, paypal, satispay)~~ e ricaricare il proprio borsellino elettronico. Il borsellino ha un saldo che viene aggiornato ad ogni ordinazione e l'**utente** può ricaricare il proprio borsellino in qualsiasi momento. Inoltre, gli **utenti** possono sottoscrivere la modalità premium che garantisce una priorità sugli ordini. L’utente può collezionare codici di sconto da utilizzare al momento dell'ordine in base al numero di ordini effettuati in passato.

Ogni **ristorante** ~~(Figura 1(b))~~ è rappresentato da un nome, una descrizione, un indirizzo, il costo della spedizione, un’immagine di profilo e un numero di stellette aggiornato ogni lunedì sulla base della percentuale di recensioni positive dell’ultima settimana. Ogni **ristorante** appartiene a una o più categorie in base al tipo di cibo offerto ~~(ad esempio: fast food, vegetariano, …).~~ I **ristoranti** che dimostrano di saper garantire un ottimo servizio (almeno 20 ordini consegnati correttamente, una valutazione clienti maggiore o uguale a 4.5 stelline su cinque, una percentuale massima di ordini annullati dal ristorante dell’1.5%, una percentuale massima di ordini con reclami del 2.5%) sono considerati Top Partner. I Top Partner compaiono in sezioni dedicate all’interno dell’app mobile Cibora e ricevono uno speciale badge che attesta il loro servizio eccellente, aiutando ad aumentare la credibilità e ottenere la fiducia dei ~~clienti~~ **utenti**. Per i Top Partner si vuole tenere traccia della data in cui sono entrati a far parte della categoria. I ristoranti propongono agli utenti una lista di **piatti** da ordinare. Ogni ~~portata~~ **piatto** ha un titolo, un’immagine, una lista di ingredienti, una lista di allergeni, il prezzo e un eventuale sconto. Inoltre, ogni **piatto** appartiene ad una o più liste ~~(es. i più venduti, promozioni, dolci, salato, ecc.)~~.

Ogni **utente** può selezionare una lista di ~~pietanze~~ **piatti** ed effettuare l’**ordine**. Finché non sono affidati ad un **rider** per la consegna, gli **ordini** possono essere annullati sia dai ~~clienti~~ **utenti**, sia dai **ristoratori**. Nel profilo dell’**utente** si possono ispezionare gli **ordini** passati ed eventualmente effettuare dei ~~reclami~~ **recensioni** inviando un messaggio al **ristorante**.

Il sistema gestisce un numero arbitrario di **riders** dove ogni **rider** è identificato da un codice, dallo stato (occupato/disponibile/fuori servizio), dalla posizione aggiornata in tempo reale tramite GPS. I riders sono classificati in base al tipo di mezzo che utilizzano (bicicletta normale, bicicletta elettrica, monopattino). I **riders** che utilizzano il monopattino devono indicare quanti km possono effettuare prima che si scarichi la batteria.

Al momento dell'**ordine**, il sistema trova il **rider** libero con la somma minima della distanza dal ristorante più la distanza dall’utente. Tuttavia, per **ordini** che prevedano un tragitto “posizione corrente del rider-> ristorante-> cliente” superiore ai 10 km, solo i **rider** con bici elettrica vengono interpellati. Per monitorare le prestazioni dei ~~ciclofattorini~~ **riders**, si vuole tenere traccia del numero di consegne effettuate da ognuno, del momento in cui il cibo da consegnare viene affidato ad un **rider** e, per le consegne già completate, anche dell’ora in cui l’**ordine** è stato recapitato al ~~cliente~~ **utente**.

Dopo che l’**ordine** è stato effettuato l’**utente** ha la possibilità di chattare sia con il ristorante che con il **rider** in caso ci fossero dei problemi con l’**ordine** come mancata consegna o netto ritardo.

Quando l’**ordine** è consegnato l’**utente** può recensire il **ristorante** e il **rider** con una valutazione da 1 a 5 e un commento testuale. Il commento testuale è facoltativo. Inoltre è anche presente la possibilità di dare una mancia al **rider** per la consegna.

Una volta al mese, vengono aggiornate le seguenti classifiche:

● **Riders** più veloci nel consegnare gli ordini

● ~~Cibi~~ **Piatti** più popolari

● **Ristoranti** con più recensioni positive

● ~~Clienti~~ **Utenti** che hanno speso di più

#### 

#### **Requisiti strutturati in gruppi di frasi omogenee:**

* Frasi per RISTORANTE:

Ogni ristorante è rappresentato da un nome, una descrizione, un indirizzo, il costo della spedizione, un’immagine di profilo e un numero di stellette aggiornato ogni lunedì sulla base della percentuale di recensioni positive dell’ultima settimana. Ogni ristorante appartiene a una o più categorie in base al tipo di cibo offerto. I ristoranti che dimostrano di saper garantire un ottimo servizio (almeno 20 ordini consegnati correttamente, una valutazione clienti maggiore o uguale a 4.5 stelline su cinque, una percentuale massima di ordini annullati dal ristorante dell’1.5%, una percentuale massima di ordini con reclami del 2.5%) sono considerati Top Partner. I Top Partner compaiono in sezioni dedicate all’interno dell’app mobile Cibora e ricevono uno speciale badge che attesta il loro servizio eccellente, aiutando ad aumentare la credibilità e ottenere la fiducia degli utenti. Per i Top Partner si vuole tenere traccia della data in cui sono entrati a far parte della categoria. I ristoranti propongono agli utenti una lista di piatti da ordinare.

* Frasi per UTENTE:

Ogni utente deve registrarsi inserendo nome, email, password, numero di telefono, indirizzo di recapito. Una volta registratosi, l’utente deve inserire un mezzo di pagamento (es.: carta di credito, paypal, satispay) e ricaricare il proprio borsellino elettronico. Il borsellino ha un saldo che viene aggiornato ad ogni ordinazione e l'utente può ricaricare il proprio borsellino in qualsiasi momento. Inoltre, gli utenti possono sottoscrivere la modalità premium che garantisce una priorità sugli ordini. L’utente può collezionare codici di sconto da utilizzare al momento dell'ordine in base al numero di ordini effettuati in passato.

Finché non sono affidati ad un rider per la consegna, gli ordini possono essere annullati sia dagli utenti, sia dai ristoratori. Nel profilo dell’utente si possono ispezionare gli ordini passati ed eventualmente effettuare dei reclami inviando un messaggio al ristorante.

Dopo che l’ordine è stato effettuato l’utente ha la possibilità di chattare sia con il ristorante che con il rider in caso ci fossero dei problemi con l’ordine come mancata consegna o netto ritardo.

Quando l’ordine è consegnato l’utente può recensire il ristorante e il rider con una valutazione da 1 a 5 e un commento testuale. Il commento testuale è facoltativo. Inoltre è anche presente la possibilità di dare una mancia al rider per la consegna.

* Frasi per RIDER:

Il sistema gestisce un numero arbitrario di riders dove ogni rider è identificato da un codice, dallo stato, dalla posizione aggiornata in tempo reale tramite GPS. I riders sono classificati in base al tipo di mezzo che utilizzano. I riders che utilizzano il monopattino devono indicare quanti km possono effettuare prima che si scarichi la batteria.

Per monitorare le prestazioni dei riders, si vuole tenere traccia del numero di consegne effettuate da ognuno, del momento in cui il cibo da consegnare viene affidato ad un rider e, per le consegne già completate, anche dell’ora in cui l’ordine è stato recapitato all’utente.

* Frasi per ORDINE:

Ogni utente può selezionare una lista di piatti ed effettuare l’ordine.

Gli ordini possono essere annullati sia dagli utenti, sia dai ristoratori.

Al momento dell'ordine, il sistema trova il rider libero con la somma minima della distanza dal ristorante più la distanza dall’utente. Tuttavia, per ordini che prevedano un tragitto “posizione corrente del rider-> ristorante-> cliente” superiore ai 10 km, solo i rider con bici elettrica vengono interpellati.

* Frasi per PIATTO:

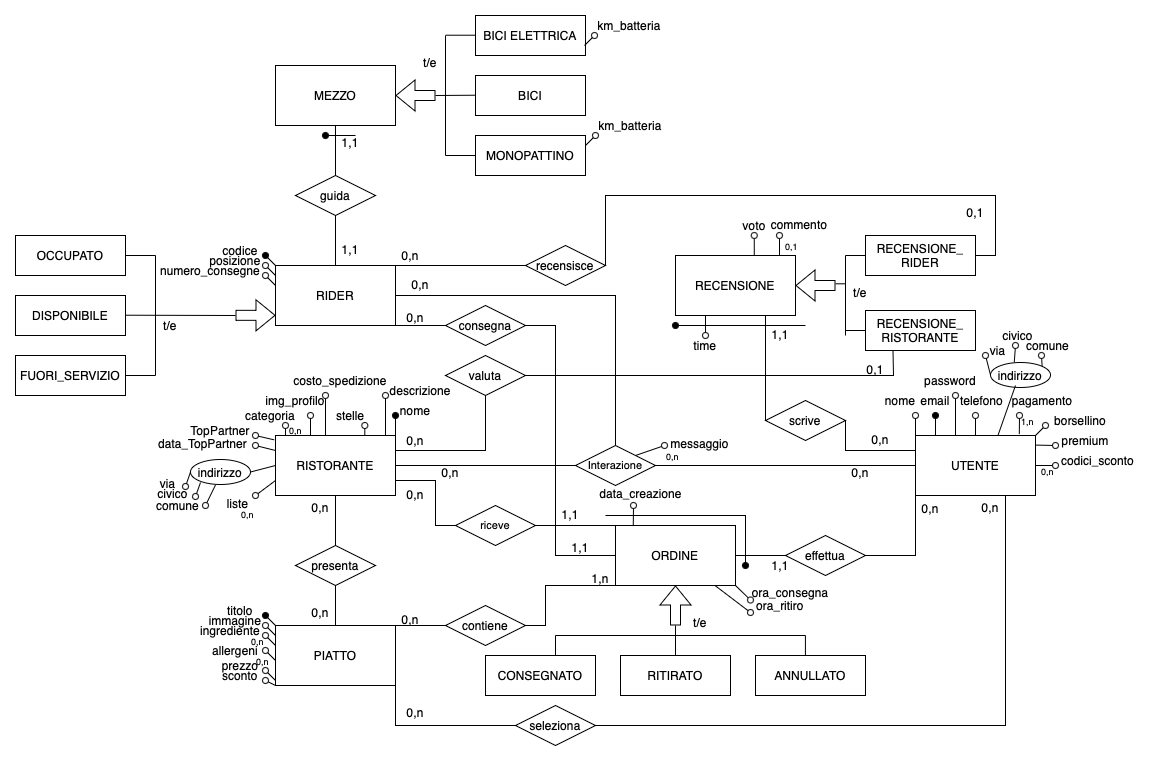
Ogni piattoha un titolo, un’immagine, una lista di ingredienti, una lista di allergeni, il prezzo e un eventuale sconto. Inoltre, ogni piatto appartiene ad una o più liste.

* Frasi per RECENSIONE:

Nel profilo dell’utente si possono ispezionare gli ordini passati ed eventualmente effettuare delle recensioni inviando un messaggio al ristorante.

Quando l’ordine è consegnato l’utente può recensire il ristorante e il rider con una valutazione da 1 a 5 e un commento testuale. Il commento testuale è facoltativo. Inoltre è anche presente la possibilità di dare una mancia al rider per la consegna.

## 1.4) Schema E-R principale + business rule



BUSINESS RULES:

* L’utente deve inserire un mezzo di pagamento per poter ricaricare il borsellino
* Il Borsellino ha un saldo che viene aggiornato ad ogni ordinazione
* L’utente può ricaricare il proprio borsellino in qualsiasi momento
* Se un utente ha la sottoscrizione alla modalità premium, garantisce la priorità sugli ordini
* I ristoranti possono diventare Top Partner se possiedono le seguenti valutazioni:
  + almeno 20 ordini consegnati correttamente
  + valutazione clienti maggiore o uguale di 4.5/5 stelline
  + percentuale massima ordini annullati di 1.5%
  + percentuale massima reclami del 2.5%
* I ristoranti Top Partner hanno uno speciale badge, aumenta la credibilità
* Gli ordini possono essere annullati sia dagli utenti che dai ristoratori
* L’utente può mandare reclami di ordini mandando un messaggio
* Se il tragitto “posizione corrente del rider-> ristorante-> cliente” è superiore ai 10 km, solo i rider con bici elettrica vengono interpellati
* L’utente può dare la mancia al rider, se desidera
* L’utente può interagire mandare messaggi sia al ristorante che al rider
* Il ristorante ed il rider non possono mandare messaggi tra loro
* Le stelle di un ristorante si calcolano facendo la media dei voti delle recensioni degli utenti
* Il numero di consegne di un rider è la somma degli ordini consegnati da essi
* Un ordine non può essere vuoto, deve contenere almeno un piatto

# PROGETTAZIONE LOGICA

## 2.1) Tavola dei volumi

|  |  |  |  |
| --- | --- | --- | --- |
| **ENTITA’/**  **ASSOCIAZIONE** | **TIPO** | **VOLUME** | **NOTE** |
| Utente | E | 100000 | Numero stimato clienti registrati al servizio |
| Ristorante | E | 10000 | Numero stimato di ristoranti aderenti |
| Piatto | E | 100000 | Numero stimato di piatti offerti da tutti i ristoranti |
| Ordine | E | 1000000 | Numero stimato di ordini effettuati |
| Recensione | E | 500000 | Numero stimato di recensioni su ordini |
| Rider | E | 5000 | Numero stimato di riders |
| Mezzo | E | 5000 | Numero stimato di mezzi per i riders |
| Bici Elettrica | E | 2000 | Numero stimato di riders con bici elettriche |
| Bici | E | 2500 | Numero stimato di riders con la bici |
| Monopattino | E | 500 | Numero stimato di riders con il monopattino |
| Recensione\_Ristorante | E | 250000 | Numero stimato di recensioni sui ristoranti |
| Recensione\_Rider | E | 250000 | Numero stimato di recensioni sui riders |
| Consegnato | E | 900000 | Stima degli ordini consegnati |
| Ritirato | E | 900000 | Stima degli ordini ritirati dai riders |
| In\_Consegna | E | 900000 | Stima degli ordini in attesa di essere consegnati |
| Annullato | E | 100000 | Stima degli ordini annullati da utenti o ristoratori |
| Contiene | A | 1000000 | Ogni ordine può contenere più piatti |
| Effettua | A | 1000000 | Numero di ordini effettuati dagli utenti |
| Riceve | A | 1000000 | Numero di ordini ricevuti dai ristoranti |
| Consegna | A | 1000000 | Numero di ordini consegnati dai riders |
| Recensisce | A | 250000 | Recensioni lasciate ai ristoranti |
| Valuta | A | 250000 | Recensioni lasciate ai riders |
| Presenta | A | 100000 | Piatti presentati dai ristoranti |
| Interazione | A | 2000000 | Numero di chat tra utenti e ristoranti/riders |
| Guida | A | 5000 | Ogni rider guida un mezzo |
| Seleziona | A | 300000 | Stima piatti selezionati dagli utenti |
| Scrive | A | 500000 | Stima delle recensioni scritte dagli utenti |

**Motivazioni delle Stime**

1. **Utente:** Considerando il mercato e la diffusione di servizi di consegna a domicilio, stimiamo che il numero di clienti registrati al servizio possa raggiungere i 100,000 in un tempo ragionevole.
2. **Ristorante:** La stima di 10,000 ristoranti aderenti è basata sull'attrattività del servizio per i ristoranti e il potenziale mercato, considerando sia le grandi città che i centri minori.
3. **Piatto:** Ogni ristorante potrebbe offrire in media 10 piatti. Con 10,000 ristoranti, il numero di piatti offerti potrebbe facilmente arrivare a 100,000.
4. **Ordine:** Se ogni utente effettua in media 10 ordini all’anno, con 100,000 utenti, il numero totale di ordini potrebbe raggiungere 1,000,000.
5. **Recensione:** Stimando che metà degli ordini ricevono una recensione, si ottiene un volume di 500,000 recensioni per 1,000,000 di ordini.
6. **Rider:** Basato sulla necessità di riders per coprire un grande numero di ordini giornalieri, stimiamo un numero di 5,000 riders attivi.
7. **Mezzo:** Presumendo che ogni rider disponga di un mezzo, il volume stimato è di 5,000 mezzi.
8. **Bici Elettrica:** Si stima che il 40% dei riders usi biciclette elettriche, basandosi sulla crescente diffusione di queste per la loro efficienza.
9. **Bici**: Si stima che il 50% dei riders utilizzi biciclette tradizionali.
10. **Monopattino**: Il restante 10% dei riders potrebbe utilizzare monopattini.
11. **Recensione\_Ristorante**: Circa il 50% delle recensioni potrebbe riguardare i ristoranti specificamente.
12. **Recensione\_Rider:**: L’altra metà delle recensioni potrebbe riguardare i riders.
13. **Consegnato**: La maggior parte degli ordini verrà consegnata con successo.
14. **Ritirato**: La maggior parte degli ordini sarà ritirata dai riders per la consegna.
15. **In\_Consegna**: La maggior parte degli ordini passerà attraverso lo stato “in consegna”.
16. **Annullato**: Una piccola percentuale degli ordini potrebbe essere annullata da utenti o ristoratori.
17. **Contiene**: Ogni ordine contiene almeno un piatto, quindi il numero di ordini corrisponde al volume dell’associazione.
18. 2. **Effettua**: Ogni ordine è effettuato da un utente, quindi il volume dell’associazione corrisponde al numero di ordini.
19. **Riceve**: Ogni ordine è ricevuto da un ristorante, quindi il volume dell’associazione corrisponde al numero di ordini.
20. **Consegna**: Ogni ordine è consegnato da un rider, quindi il volume dell’associazione corrisponde al numero di ordini.
21. **Recensisce**: Numero stimato di recensioni sui ristoranti.
22. **Valuta**: Numero stimato di recensioni sui riders.
23. **Presenta**: Ogni piatto è presentato da un ristorante.
24. **Interazione**: Ogni ordine può generare diverse interazioni, quindi il volume è stimato in modo conservativo.
25. **Guida**: Ogni rider guida un mezzo.
26. **Seleziona**: Numero stimato di piatti selezionati dagli utenti.
27. **Scrive:** Numero stimato di recensioni scritte dagli utenti

## 2.2) Tavola delle operazioni

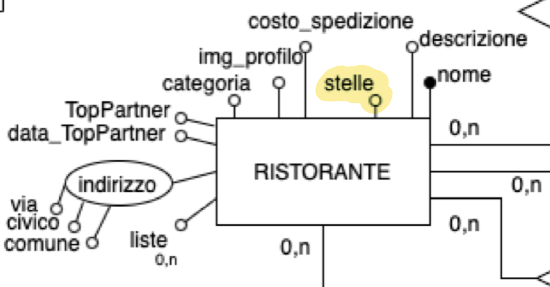
|  |  |  |  |
| --- | --- | --- | --- |
| NUMERO | OPERAZIONE | TIPO | FREQUENZA |
| 1 | Registrazione utente al sito | I | 500/settimana |
| 2 | Aggiorna credenziali utente | I | 200/settimana |
| 3 | Aggiorna metodo di pagamento di un utente | I | 200/settimana |
| 4 | Creazione di un ordine da parte di un utente | I | 20000/giorno |
| 5 | Cancellazione ordine da parte di un utente | I | 1000/giorno |
| 6 | Aggiunta ristorante al sito | I | 20/settimana |
| 7 | Rimozione ristorante dal sito | I | 10/mese |
| 8 | Modifica informazioni ristorante | I | 50/settimana |
| 9 | Aggiunta piatto da parte di un ristorante | I | 200/giorno |
| 10 | Rimozione di un piatto da parte di un ristorante | I | 100/giorno |
| 11 | Modifica le informazioni di un piatto | I | 300/giorno |
| 12 | Inserimento recensione ad un ristorante | I | 5000/giorno |
| 13 | Inserimento recensione ad un rider | I | 5000/giorno |
| 14 | Elenca tutti i piatti di un ristorante | B | 50000/giorno |
| 15 | Elenca tutte le recensioni di un ristorante | B | 10000/giorno |
| 16 | Elenca tutte le recensioni di un rider | B | 10000/giorno |
| 17 | Aggiungi un piatto alla lista | I | 20000/giorno |
| 18 | Rimuovi un piatto dalla lista | I | 10000/giorno |
| 19 | Aggiorna lo stato di un ordine | I | 20000/giorno |
| 20 | Verifica le condizioni per la qualifica di Top Partner | B | 1/settimana |
| 21 | Calcolo classifiche ristoranti con recensioni più positive | B | 1/mese |
| 22 | Calcolo riders più veloci nel consegnare gli ordini | B | 1/mese |
| 23 | Calcolo piatti più popolari | B | 1/mese |
| 24 | Calcolo utenti che hanno speso di più | B | 1/mese |
| 25 | Valutazione assegnamento ordine al rider | B | 5000/giorno |
| 26 | Elenca tutti gli ordini di un utente | B | 50000/giorno |
| 27 | Elenca tutti gli ordini di un ristorante | B | 20000/giorno |
| 28 | Elenca tutti gli ordini di un rider | B | 20000/giorno |

\*Le stime sono basate su 100000 utenti attivi, 10000 ristoranti aderenti, e un alto volume di attività giornaliera.

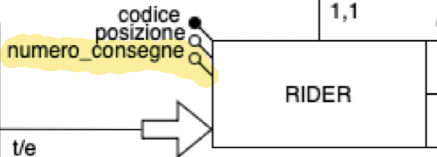
## 2.3) Ristrutturazione dello schema E-R:

#### 2.3.1) Analisi delle ridondanze:

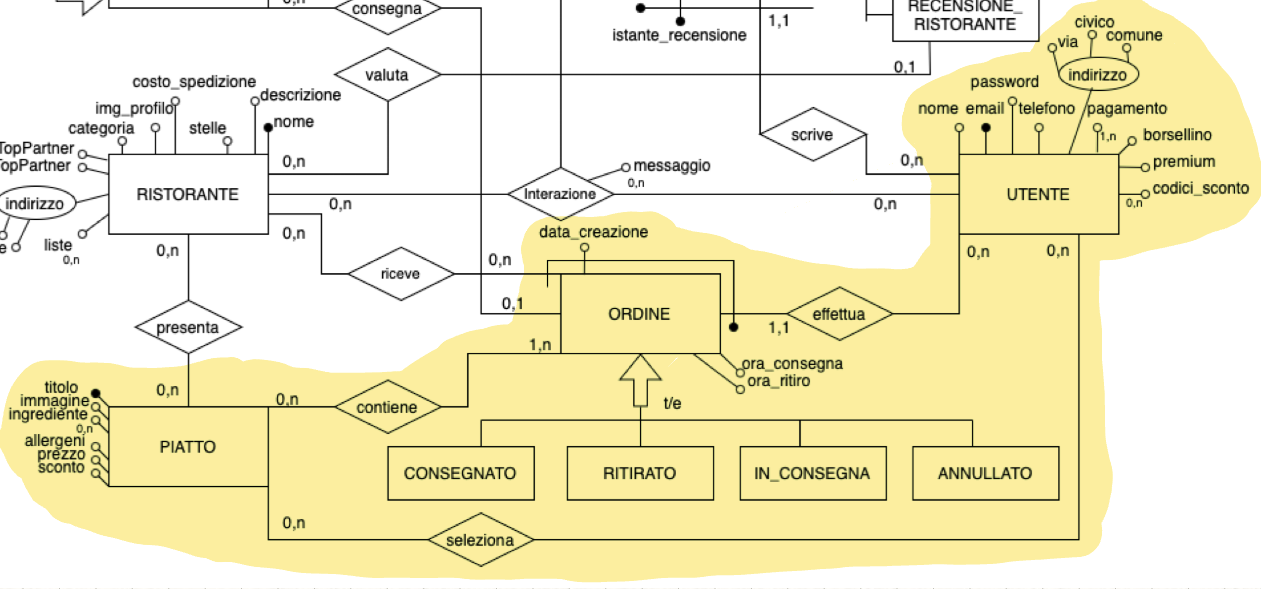
* Stelle: questo attributo fa parte dell’entità RISTORANTE e fa riferimento alla media delle valutazioni dei clienti riguardo ad esso. La reputo ridondante perchè si potrebbe facilmente ottenere facendo la media con i voti presi da RECENSIONE\_RISTORANTE.



* per quanto riguarda RISTORANTE anche TopPartner il quale fa riferimento alla condizione di un Ristorante se appartiene alla categoria stessa o meno.
* numero\_consegne: questo attributo fa parte dell’entità RIDER e fa riferimento al numero di consegne effettuate (completate) da parte del rider. La reputo una ridondanza perché derivabile per conteggio del numero di occorrenze di Ordini consegnati ed il relativo rider.



* Seleziona: questa associazione tra PIATTO ed UTENTE fa riferimento ai piatti che gli utenti selezionano per poter completare gli ordini. La ritengo una ridondanza perché i piatti selezionati sono derivabili attraverso l’ordine. L’utente crea l’ordine e poi seleziona i piatti desiderati.



##### 

##### Analisi prima ridondanza:

##### **TABELLA DEI VOLUMI D’INTERESSE:**

|  |  |  |
| --- | --- | --- |
| **CONCETTO** | **TIPO** | **VOLUME** |
| RISTORANTE | E | 10000 |
| RECENSIONE\_RISTORANTE | E | 250000 |
| VALUTA | A | 250000 |

**TABELLA DELLE OPERAZIONI D’INTERESSE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **OPERAZIONE** | **DESCRIZIONE** | **TIPO** | **FREQUENZA** |
| 12 | Inserimento recensione ad un ristorante | I | 5000/giorno |
| 20 | Verifica le condizioni per la qualifica di Top Partner | B | 1/settimana |
| 21 | Calcolo classifiche ristoranti con più recensioni positive | B | 1/mese |

SCENARIO A: con ridondanza

Tavola accessi operazione n’12:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RISTORANTE | E | 1 | S |
| RECENSIONE\_  RISTORANTE | E | 1 | S |
| VALUTA | A | 1 | S |

Tavola accessi operazione n’20:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RISTORANTE | E | 1 | L |

Tavola accessi operazione n’21:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RISTORANTE | E | 1 | L |

SCENARIO B: senza ridondanza

Tavola accessi operazione n’12:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RECENSIONE\_  RISTORANTE | E | 1 | S |

Tavola accessi operazione n’20:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RECENSIONE\_  RISTORANTE | E | 1 | L |

Tavola accessi operazione n’21:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| RECENSIONE\_  RISTORANTE | E | 1 | L |

***COSTI:***

Scenario A:

-Spazio: contando all’incirca 4 byte per memorizzare le stelle, per ogni ristorante si conta 4\*10000 = 40000 byte (40 Kilobyte)

-Tempo:

-op.12: 3\*35000 \*1 (3 scritture \* 35000 accessi/settimana(=5000 accessi al giorno))

-op.20: 1\*1\*10000 (1 lettura \* 1 accesso/settimana \* 10000 ristoranti)

-op.21: 1\*0.13\*10000 (1 lettura \* 0.13 accessi/settimana(=1 accessi/mese) \* 10000 ristoranti)

->totale: 105000 + 10000 + 1300= 116300 accessi a settimana

Scenario B:

-Spazio:0 byte perché non essendoci l’attributo non occupa spazio

-Tempo:

-op.12: 1\*35000\*1 (1 scrittura \* 35000 accessi/settimana(=5000 accessi al giorno))

-op.20: 1\*1\*250000 (1 lettura \* 1 accesso/settimana \* 250000 volume Recensioni Ristoranti)

-op.21: 1\*0.13\*250000(1 lettura \* 1 accesso/mese \* 250000 volume Recensioni Ristoranti)

->totale: 35000 + 250000 + 32500 = 317500 accessi a settimana

In conclusione, mantenere la ridondanza comporta una lieve occupazione di spazio in più, ma offre un notevole risparmio in termini di accessi settimanali.

Anche la seconda e la terza ridondanza presentano un risultato simile, rivelandosi quindi più efficiente da mantenere.

##### Analisi quarta ridondanza:

##### **TABELLA DEI VOLUMI D’INTERESSE:**

|  |  |  |
| --- | --- | --- |
| **CONCETTO** | **TIPO** | **VOLUME** |
| UTENTE | E | 100000 |
| ORDINE | E | 100000 |
| PIATTO | E | 100000 |
| EFFETTUA | A | 100000 |
| CONTIENE | A | 100000 |
| SELEZIONA | A | 300000 |

**TABELLA DELLE OPERAZIONI D’INTERESSE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **OPERAZIONE** | **DESCRIZIONE** | **TIPO** | **FREQUENZA** |
| 11 | Modifica le informazioni di un piatto | I | 300/giorno |
| 14 | Elenca tutti i piatti di un ristorante | B | 50000/giorno |

SCENARIO A: con ridondanza

Tavola accessi operazione n’11:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| PIATTO | E | 1 | S |

Tavola accessi operazione n’14:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| PIATTO | E | 100000 | L |

SCENARIO B: senza ridondanza

Tavola accessi operazione n’11:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| PIATTO | E | 1 | S |

Tavola accessi operazione n’14:

|  |  |  |  |
| --- | --- | --- | --- |
| **CONCETTO** | **COSTRUTTO** | **ACCESSI** | **TIPO** |
| PIATTO | E | 100000 | L |

***COSTI:***

Scenario A:

-Spazio: Assumiamo:

- gli identificatori sia dell’utente che del piatto siano lunghi 10 caratteri.

- ogni carattere = 1 byte

- Spazio per una tupla di “Seleziona” = 10 + 10 ⇒ 20 byte

- Spazio totale stimato: 20 \* 300000 ⇒ 6000000 byte (6000 Kilobyte)

-Tempo:

-op.11: 1\*2100\*1 (1 scrittura \* 2100 accesso/settimana(=300 accessi/ giorno) \* 1 Piatto)

-op.14: 1\*350000\*100000(1 lettura \* 350000 accesso/settimana(=50000 accessi/ giorno) \* 100000 volume di Piatto)

->totale: 2100 + 35.000.000.000 = 35.000.002.100 accessi a settimana

Scenario B:

-Spazio:0 byte perché non essendoci l’attributo non occupa spazio

-Tempo:

-op.11: 1\*2100\*1 (1 scrittura \* 2100 accesso/settimana(=300 accessi/ giorno) \* 1 Piatto)

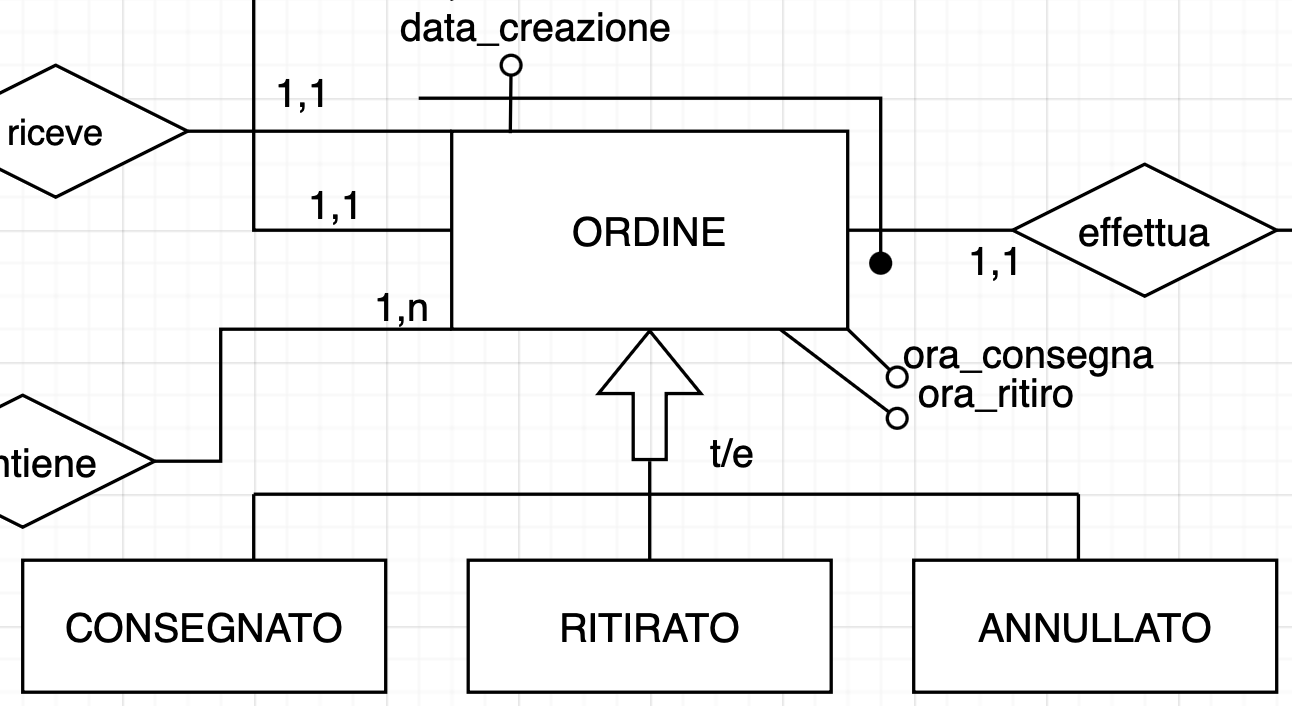
-op.14: 1\*350000\*100000(1 lettura \* 350000 accesso/settimana(=50000 accessi/ giorno) \* 100000 volume di Piatto)

->totale: 2100 + 35.000.000.000 = 35.000.002.100 accessi a settimana

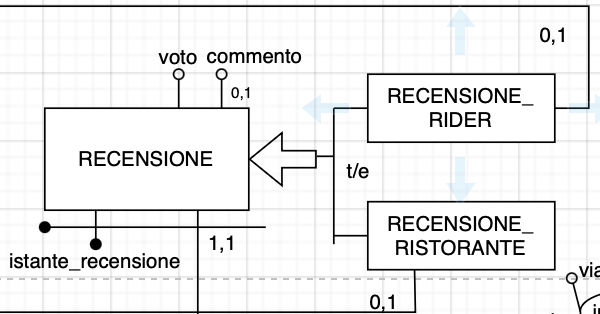
In conclusione, mantenere la ridondanza comporta un aumento dello spazio occupato senza ridurre il numero di accessi settimanali. Pertanto, si è deciso di eliminare la ridondanza per semplificare e ottimizzare il sistema.

#### 2.3.2) Eliminazione delle generalizzazioni

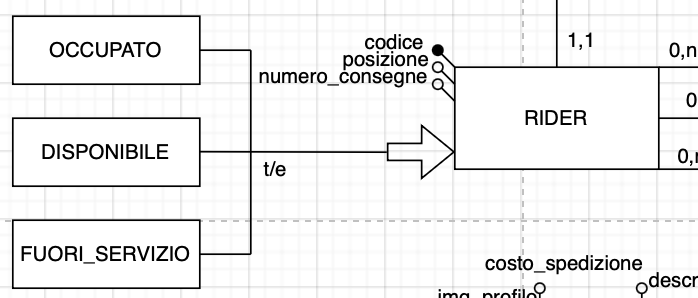
1. ORDINE -> CONSEGNATA / RITIRATO / IN\_CONSEGNA / ANNULLATO



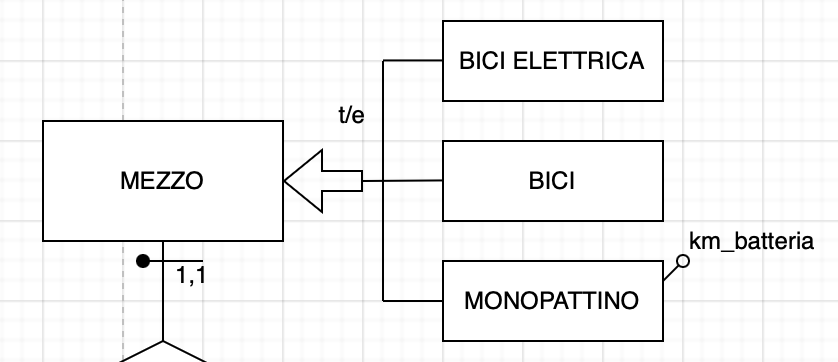
1. RECENSIONE -> RECENSIONE\_RISTORANTE / RECENSIONE\_RIDER



1. RIDER -> OCCUPATO / DISPONIBILE / FUORI\_SERVIZIO



1. MEZZO -> BICI / BICI\_ELETTRICA / MONOPATTINO



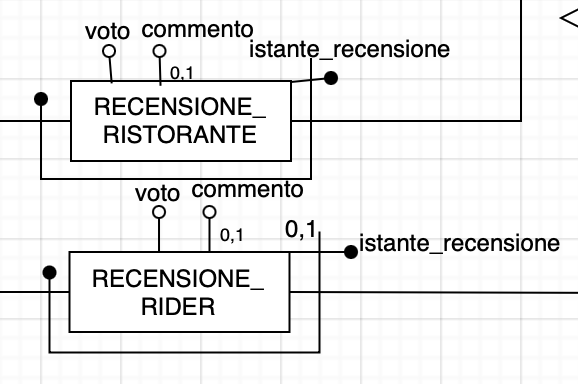
1) E’ stato deciso di accorpare le entità figlie nel genitore poiché le operazioni non fanno troppa distinzioni tra le istanze delle varie entità.

Verranno aggiunte le seguenti Business Rules:

* L’attributo “annullato” di ORDINE ha cardinalità (0,1)
* L’attributo “ritirato” di ORDINE ha cardinalità (0,1)
* L’attributo “consegnato” di ORDINE ha cardinalità (0,1)
* Se l’attributo “annullato” è true (=1) allora gli attributi “ritirato” e “consegnato” sono false (=0)
* Se l’attributo “ritirato” è true (=1) allora gli attributi “annullato” e “consegnato” sono false (=0)
* Se l’attributo “consegnato” è true (=1) allora gli attributi “ritirato” e “annullato” sono false (=0)
* Gli attributi “annullato”, “ritirato” e “consegnato” non possono essere tutti true o false



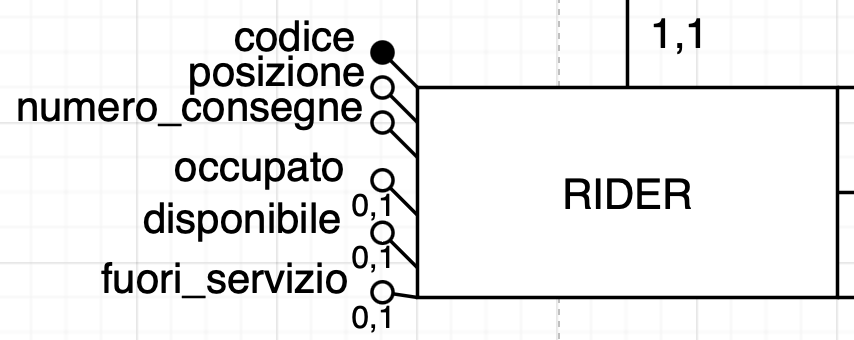
2) E’ stato deciso di accorpare l’entità padre all’interno delle entità figlie perché si fa distinzione tra le entità figlie e ciascuna ha operazioni e relazioni distinte fra loro.



3) E’ stato deciso di accorpare le entità figlie nel genitore poiché le operazioni non fanno troppa distinzioni tra le istanze delle varie entità.

Verranno aggiunte le seguenti Business Rules:

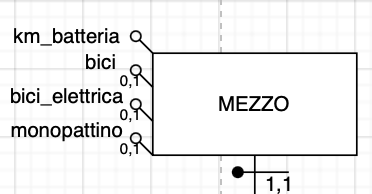
* L’attributo “occupato” di RIDER ha cardinalità (0,1)
* L’attributo “disponibile” di RIDER ha cardinalità (0,1)
* L’attributo “fuori\_servizio” di RIDER ha cardinalità (0,1)
* Se l’attributo “occupato” è true (=1) allora gli attributi “disponibile” e “fuori\_servizio” sono false (=0)
* Se l’attributo “disponibile” è true (=1) allora gli attributi “occupato” e “fuori\_servizio” sono false (=0)
* Se l’attributo “fuori\_servizio” è true (=1) allora gli attributi “disponibile” e “occupato” sono false (=0)
* Gli attributi “occupato”, “disponibile” e “fuori\_servizio” non possono essere tutti true o false



4) E’ stato deciso di accorpare le entità figlie nel genitore poiché le operazioni non fanno troppa distinzioni tra le istanze delle varie entità.

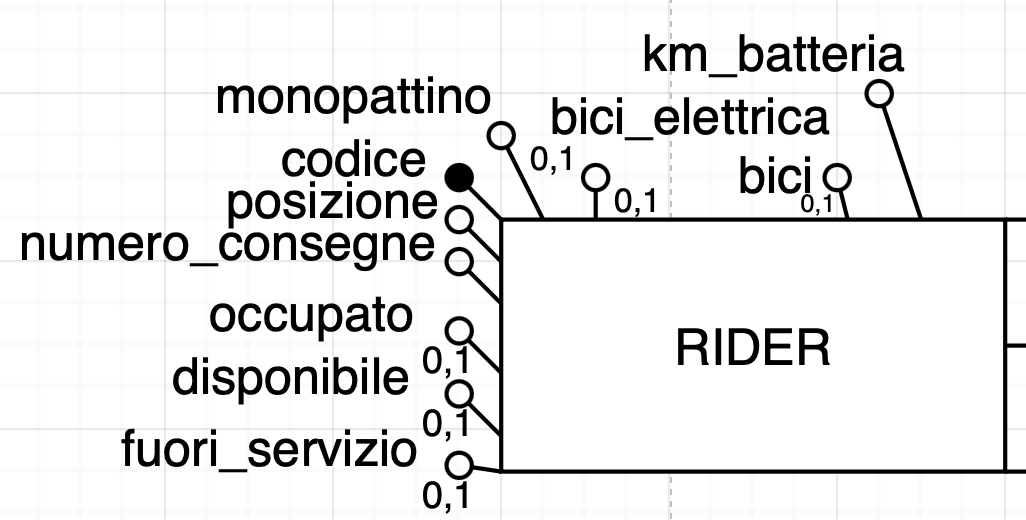
Verranno aggiunte le seguenti Business Rules:

* L’attributo “bici” di MEZZO ha cardinalità (0,1)
* L’attributo “bici\_elettrica” di MEZZO ha cardinalità (0,1)
* L’attributo “monopattino” di MEZZO ha cardinalità (0,1)
* Se l’attributo “bici” è true (=1) allora gli attributi “bici\_elettrica” e “monopattino” sono false (=0)
* Se l’attributo “bici\_elettrica” è true (=1) allora gli attributi “bici” e “monopattino” sono false (=0)
* Se l’attributo “monopattino” è true (=1) allora gli attributi “bici” e “bici\_elettrica” sono false (=0)
* Gli attributi “bici”, “elettrico” e “monopattino” non possono essere tutti true o false

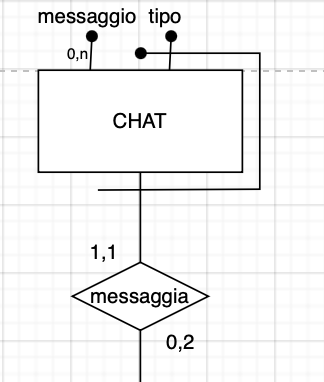


#### 2.3.4) Eventuale partizionamento/accorpamento di entità o associazioni

1)Ho deciso di accorpare l’entità MEZZO in RIDER perché ritengo che possa essere più semplice e veloce accedere alle informazioni.

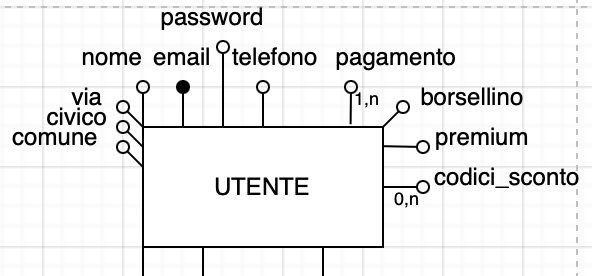


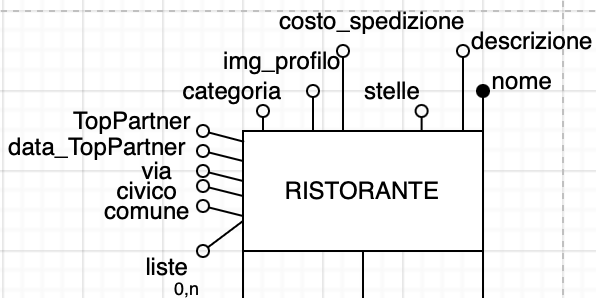
2) Ho deciso di trasformare l’associazione “Interazione” in un’entità. Questa scelta è stata fatta per evitare complicazioni derivanti dalla relazione ternaria, migliorando così la chiarezza e la semplicità del sistema.



#### 2.3.4) Eventuale eliminazione degli attributi composti e degli attributi multivalore

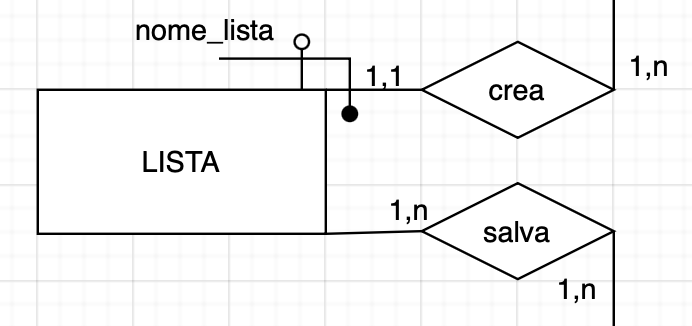
-Attributi composti:  
 -Indirizzo sia in UTENTE che in RISTORANTE





-Attributi multivalore:

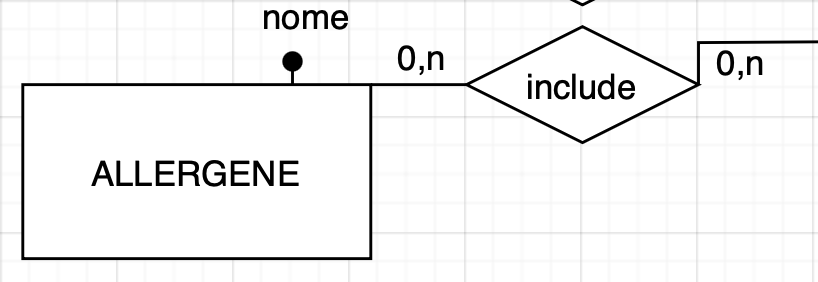
1. liste(RISTORANTE)



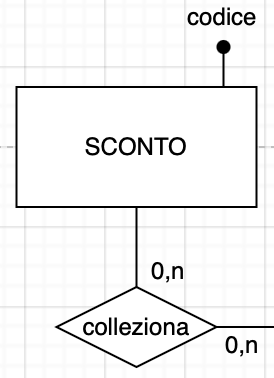
1. ingrediente(PIATTO)



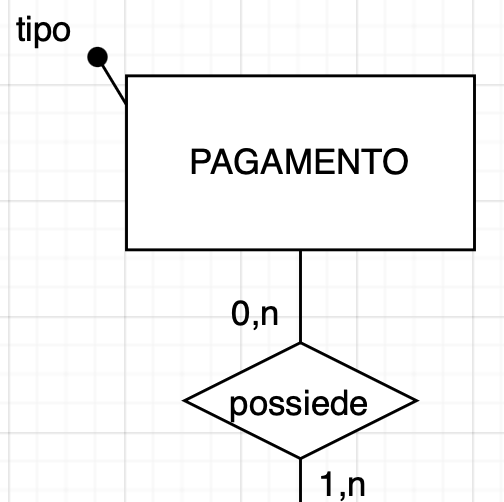
1. allergeni(PIATTO)



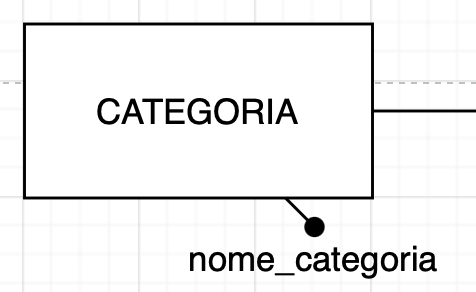
1. codice\_sconto(UTENTE)



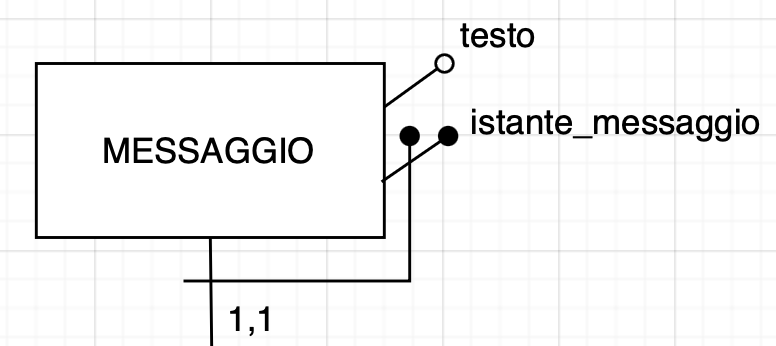
1. pagamento(UTENTE)



1. categoria(RISTORANTE)



1. messaggio(CHAT)



#### 2.3.5) Scelta degli identificatori principali

|  |  |
| --- | --- |
| ***ENTITA’*** | ***IDENTIFICATORE*** |
| UTENTE | email |
| PAGAMENTO | tipo |
| SCONTO | codice |
| RECENSIONE\_  RISTORANTE | istante\_recensione - email\_utente-  nome\_ristorante |
| RECENSIONE\_  RIDER | istante\_recensione - email\_utente -  codice\_rider |
| PIATTO | titolo - nome\_lista |
| INGREDIENTE | nome |
| ALLERGENE | nome |
| LISTA | id\_lista |
| RISTORANTE | nome |
| CATEGORIA | nome |
| ORDINE | id\_ordine |
| CHAT | id\_chat |
| RIDER | codice |
| MESSAGGIO | istante\_messaggio - |

* Utente viene identificato tramite la propria email
* Pagamento viene identificato tramite il proprio tipo
* Sconto viene identificato tramite il proprio codice
* Recensione Ristorante viene identificato tramite l’istante recensione, l’email dell’utente la scrive e il nome del ristorante
* Recensione Rider viene identificato tramite l’istante recensione, l’email dell’utente che la scrive ed il codice del rider
* Piatto viene identificato dal titolo del piatto e dal nome della lista a cui appartiene
* Ingrediente viene identificato dal proprio nome
* Allergene viene identificato dal proprio nome
* Lista viene identificata dal nome delle lista e dal nome del ristorante che la crea
* Ristorante viene identificato dal proprio nome
* Categoria viene identificata dal proprio nome
* Ordine viene identificato da un ID (è stato deciso di aggiungere un ID in quanto ne semplifica e aumenta la chiarezza)
* Chat viene identificata tramite un ID (è stato deciso di aggiungere un ID in quanto ne semplifica e aumenta la chiarezza)
* Rider viene identificato tramite il proprio codice
* Messaggio viene identificato tramite l’istante in cui viene scritto e l’ID della chat a cui appartiene

## 

## 2.4) Schema relazionale

BUSINESS RULES:

* L’utente deve inserire un mezzo di pagamento per poter ricaricare il borsellino
* Il Borsellino ha un saldo che viene aggiornato ad ogni ordinazione
* L’utente può ricaricare il proprio borsellino in qualsiasi momento
* Se un utente ha la sottoscrizione alla modalità premium, garantisce la priorità sugli ordini
* I ristoranti possono diventare Top Partner se possiedono le seguenti valutazioni:
  + almeno 20 ordini consegnati correttamente
  + valutazione clienti maggiore o uguale di 4.5/5 stelline
  + percentuale massima ordini annullati di 1.5%
  + percentuale massima reclami del 2.5%
* I ristoranti Top Partner hanno uno speciale badge, aumenta la credibilità
* Gli ordini possono essere annullati sia dagli utenti che dai ristoratori
* L’utente può mandare reclami di ordini mandando un messaggio
* Se il tragitto “posizione corrente del rider-> ristorante-> cliente” è superiore ai 10 km, solo i rider con bici elettrica vengono interpellati
* L’utente può dare la mancia al rider, se desidera
* L’utente può interagire mandare messaggi sia al ristorante che al rider
* Il ristorante ed il rider non possono mandare messaggi tra loro
* Le stelle di un ristorante si calcolano facendo la media dei voti delle recensioni degli utenti
* Il numero di consegne di un rider è la somma degli ordini consegnati da essi
* Un ordine non può essere vuoto, deve contenere almeno un piatto

NUOVE REGOLE AGGIUNTE:

* L’attributo “annullato” di ORDINE ha cardinalità (0,1)
* L’attributo “ritirato” di ORDINE ha cardinalità (0,1)
* L’attributo “consegnato” di ORDINE ha cardinalità (0,1)
* Se l’attributo “annullato” è true (=1) allora gli attributi “ritirato” e “consegnato” sono false (=0)
* Se l’attributo “ritirato” è true (=1) allora gli attributi “annullato” e “consegnato” sono false (=0)
* Se l’attributo “consegnato” è true (=1) allora gli attributi “ritirato” e “annullato” sono false (=0)
* Gli attributi “annullato”, “ritirato” e “consegnato” non possono essere tutti true o false
* L’attributo “occupato” di RIDER ha cardinalità (0,1)
* L’attributo “disponibile” di RIDER ha cardinalità (0,1)
* L’attributo “fuori\_servizio” di RIDER ha cardinalità (0,1)
* Se l’attributo “occupato” è true (=1) allora gli attributi “disponibile” e “fuori\_servizio” sono false (=0)
* Se l’attributo “disponibile” è true (=1) allora gli attributi “occupato” e “fuori\_servizio” sono false (=0)
* Se l’attributo “fuori\_servizio” è true (=1) allora gli attributi “disponibile” e “occupato” sono false (=0)
* Gli attributi “occupato”, “disponibile” e “fuori\_servizio” non possono essere tutti true o false
* L’attributo “bici” di MEZZO ha cardinalità (0,1)
* L’attributo “bici\_elettrica” di MEZZO ha cardinalità (0,1)
* L’attributo “monopattino” di MEZZO ha cardinalità (0,1)
* Se l’attributo “bici” è true (=1) allora gli attributi “bici\_elettrica” e “monopattino” sono false (=0)
* Se l’attributo “bici\_elettrica” è true (=1) allora gli attributi “bici” e “monopattino” sono false (=0)
* Se l’attributo “monopattino” è true (=1) allora gli attributi “bici” e “bici\_elettrica” sono false (=0)
* Gli attributi “bici”, “elettrico” e “monopattino” non possono essere tutti true o false

**REGOLE DI DERIVAZIONE:**

* L’attributo stelle in RISTORANTE viene calcolato facendo la media dei voti delle recensioni da parte degli utenti
* Il numero di consegne dei RIDER viene calcolato contando il numero di ordini da loro ritirati e consegnati.

## 2.5) Schema relazionale

**VINCOLI D’INTEGRITA’ RELAZIONALE:**

* vincolo UNIQUE su UTENTE(telefono)
* LISTA(nome\_ristorante) referenzia RISTORANTE(nome)
* vincolo UNIQUE su PIATTO(id\_lista)
* LISTA(nome\_ristorante) referenzia RISTORANTE(nome)
* PIATTO(id\_lista) referenzia LISTA(id\_lista)
* PAGAMENTO(email\_utente) referenzia UTENTE(email)
* COLLEZIONA(email\_utente) referenzia UTENTE(email)
* COLLEZIONA(codice\_sconto) referenzia SCONTO(codice)
* RECENSIONE\_RISTORANTE(email\_utente) referenzia UTENTE(email)
* RECENSIONE\_RISTORANTE(nome\_ristorante) referenzia RISTORANTE(nome)
* RECENSIONE\_RIDER(email\_utente) referenzia UTENTE(email)
* RECENSIONE\_RIDER(codice\_rider) referenzia RIDER(codice)
* ORDINE(email\_utente) referenzia UTENTE(email)
* ORDINE(nome\_ristorante) referenzia RISTORANTE(nome)
* ORDINE(codice\_rider) referenzia RIDER(codice)
* CHAT(id\_ordine) referenzia ORDINE(id\_ordine)
* CHAT(email\_utente) referenzia UTENTE(email)
* MESSAGGIO(id\_chat) referenzia CHAT(id\_chat)
* CONTIENE(id\_ordine) referenzia ORDINE(id\_ordine)
* CONTIENE(titolo\_piatto) referenzia PIATTO(titolo)
* INCLUDE(nome\_allergene) referenzia ALLERGENE(nome)
* INCLUDE(titolo\_piatto) referenzia PIATTO(titolo)
* APPARTIENE(nome\_categoria) referenzia CATEGORIA(nome)
* APPARTIENE(nome\_ristorante) referenzia RISTORANTE(nome)
* COMPOSTO(titolo\_piatto) referenzia PIATTO(titolo)
* COMPOSTO(nome\_ingrediente) referenzia INGREDIENTE(nome)

# IMPLEMENTAZIONE

## 3.1) DDL di creazione del database

**-- UTENTE**

CREATE TABLE UTENTE (

email VARCHAR(255) PRIMARY KEY,

nome VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

telefono NUMERIC(10) unique CHECK (telefono > 999999999) NOT NULL,

via VARCHAR(255) NOT NULL,

civico NUMERIC(3),

comune VARCHAR(100) NOT NULL,

borsellino NUMERIC(5),

premium BOOLEAN

);

-- **RISTORANTE**

CREATE TABLE RISTORANTE (

nome VARCHAR(255) PRIMARY KEY,

descrizione TEXT NOT NULL,

stelle DECIMAL(2,1) NOT NULL CHECK(stelle > 1 AND stelle <= 5),

costo\_spedizione DECIMAL(5, 2) NOT NULL,

img\_profilo TEXT,

TopPartner BOOLEAN NOT NULL,

data\_TopPartner TIMESTAMP,

via VARCHAR(255) NOT NULL,

civico NUMERIC(3),

comune VARCHAR(100) NOT NULL

);

-- **RIDER**

CREATE TABLE RIDER (

codice VARCHAR(50) PRIMARY KEY,

posizione VARCHAR(255) NOT NULL,

numero\_consegne NUMERIC(5) NOT NULL,

occupato BOOLEAN,

disponibile BOOLEAN,

fuori\_servizio BOOLEAN,

bici BOOLEAN,

bici\_elettrica BOOLEAN,

monopattino BOOLEAN,

km\_batteria DECIMAL(10, 2)

);

-- **LISTA**

CREATE TABLE LISTA (

id\_lista VARCHAR(50) UNIQUE,

nome VARCHAR(255),

nome\_ristorante VARCHAR(255),

PRIMARY KEY (id\_lista, nome\_ristorante),

FOREIGN KEY (nome\_ristorante) REFERENCES RISTORANTE(nome) ON DELETE CASCADE ON UPDATE CASCADE

-- **PIATTO**

CREATE TABLE PIATTO (

titolo VARCHAR(255) UNIQUE,

id\_lista VARCHAR(50) UNIQUE,

immagine TEXT,

prezzo DECIMAL(5, 2) NOT NULL,

sconto DECIMAL(4, 2) NOT NULL,

PRIMARY KEY (titolo, id\_lista),

FOREIGN KEY (id\_lista) REFERENCES LISTA(id\_lista) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **INGREDIENTE**

CREATE TABLE INGREDIENTE (

nome VARCHAR(255) PRIMARY KEY

);

-- **PAGAMENTO**

CREATE TABLE PAGAMENTO (

tipo INT PRIMARY KEY,

email\_utente VARCHAR(255),

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **SCONTO**

CREATE TABLE SCONTO (

codice VARCHAR(10) PRIMARY KEY

);

-- **COLLEZIONA**

CREATE TABLE COLLEZIONA (

email\_utente VARCHAR(255),

codice\_sconto VARCHAR(50),

PRIMARY KEY (email\_utente, codice\_sconto),

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (codice\_sconto) REFERENCES SCONTO(codice) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **RECENSIONE\_RISTORANTE**

CREATE TABLE RECENSIONE\_RISTORANTE (

istante\_recensione TIMESTAMP NOT NULL,

email\_utente VARCHAR(255),

nome\_ristorante VARCHAR(255),

voto NUMERIC(1) NOT NULL CHECK(voto > 1 AND voto < 5),

commento TEXT,

PRIMARY KEY (istante\_recensione, email\_utente, nome\_ristorante),

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (nome\_ristorante) REFERENCES RISTORANTE(nome) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **RECENSIONE\_RIDER**

CREATE TABLE RECENSIONE\_RIDER (

istante\_recensione TIMESTAMP NOT NULL,

email\_utente VARCHAR(255),

codice\_rider VARCHAR(50),

voto NUMERIC(1) NOT NULL CHECK(voto > 1 AND voto < 5),

commento TEXT,

PRIMARY KEY (istante\_recensione, email\_utente, codice\_rider),

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (codice\_rider) REFERENCES RIDER(codice) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **ORDINE**

CREATE TABLE ORDINE (

id\_ordine VARCHAR(50) PRIMARY KEY,

email\_utente VARCHAR(255),

nome\_ristorante VARCHAR(255),

codice\_rider VARCHAR(50),

data\_creazione TIMESTAMP NOT NULL,

ora\_consegna TIMESTAMP,

ora\_ritiro TIMESTAMP,

consegnato BOOLEAN,

ritirato BOOLEAN,

annullato BOOLEAN,

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (nome\_ristorante) REFERENCES RISTORANTE(nome) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (codice\_rider) REFERENCES RIDER(codice) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **CHAT**

CREATE TABLE CHAT (

id\_chat VARCHAR(50) PRIMARY KEY,

id\_ordine VARCHAR(50),

email\_utente VARCHAR(255),

tipo INT NOT NULL CHECK (tipo > 0 AND tipo < 3),

FOREIGN KEY (id\_ordine) REFERENCES ORDINE(id\_ordine) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **MESSAGGIO**

CREATE TABLE MESSAGGIO (

istante\_messaggio TIMESTAMP,

id\_chat VARCHAR(50),

testo TEXT NOT NULL,

PRIMARY KEY (istante\_messaggio, id\_chat),

FOREIGN KEY (id\_chat) REFERENCES CHAT(id\_chat) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **SALVA**

CREATE TABLE SALVA (

id\_lista VARCHAR(255),

titolo\_piatto VARCHAR(255),

PRIMARY KEY (id\_lista, titolo\_piatto)

);

-- **CONTIENE**

CREATE TABLE CONTIENE (

titolo\_piatto VARCHAR(255),

id\_ordine VARCHAR(50),

PRIMARY KEY (titolo\_piatto, id\_ordine),

FOREIGN KEY (id\_ordine) REFERENCES ORDINE(id\_ordine) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (titolo\_piatto) REFERENCES PIATTO(titolo) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **ALLERGENE**

CREATE TABLE ALLERGENE (

nome VARCHAR(255) PRIMARY KEY

);

-- **INCLUDE**

CREATE TABLE INCLUDE (

nome\_allergene VARCHAR(255),

titolo\_piatto VARCHAR(255),

PRIMARY KEY (nome\_allergene, titolo\_piatto),

FOREIGN KEY (nome\_allergene) REFERENCES ALLERGENE(nome) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (titolo\_piatto) REFERENCES PIATTO(titolo) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **CATEGORIA**

CREATE TABLE CATEGORIA (

nome VARCHAR(255) PRIMARY KEY

);

-- **APPARTIENE**

CREATE TABLE APPARTIENE (

nome\_categoria VARCHAR(255),

nome\_ristorante VARCHAR(255),

PRIMARY KEY (nome\_categoria, nome\_ristorante),

FOREIGN KEY (nome\_categoria) REFERENCES CATEGORIA(nome) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (nome\_ristorante) REFERENCES RISTORANTE(nome) ON DELETE CASCADE ON UPDATE CASCADE

);

-- **COMPOSTO**

CREATE TABLE COMPOSTO (

nome\_ingrediente VARCHAR(255),

titolo\_piatto VARCHAR(255),

PRIMARY KEY (nome\_ingrediente, titolo\_piatto),

FOREIGN KEY (titolo\_piatto) REFERENCES PIATTO(titolo) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (nome\_ingrediente) REFERENCES INGREDIENTE(nome) ON DELETE CASCADE ON UPDATE CASCADE

);

## 3.2) DML di popolamento di tutte le tabelle del database

**-- UTENTE**

INSERT INTO UTENTE (email, nome, password, telefono, via, civico, comune, borsellino, premium) VALUES

('utente1@example.com', 'Mario Rossi', 'password123', 1234567890, 'Via Roma', 1, 'Roma', 100, TRUE);

INSERT INTO UTENTE (email, nome, password, telefono, via, civico, comune, borsellino, premium) VALUES

('utente2@example.com', 'Luca Bianchi', 'password456', 2345678901, 'Via Milano', 2, 'Milano', 200, FALSE);

INSERT INTO UTENTE (email, nome, password, telefono, via, civico, comune, borsellino, premium) VALUES

('utente3@example.com', 'Giulia Verdi', 'password789', 3456789012, 'Via Napoli', 3, 'Napoli', 150, TRUE);

INSERT INTO UTENTE (email, nome, password, telefono, via, civico, comune, borsellino, premium) VALUES

('utente4@example.com', 'Sara Gialli', 'password012', 4567890123, 'Via Torino', 4, 'Torino', 120, FALSE);

INSERT INTO UTENTE (email, nome, password, telefono, via, civico, comune, borsellino, premium) VALUES

('utente5@example.com', 'Marco Neri', 'password345', 5678901234, 'Via Palermo', 5, 'Palermo', 110, TRUE);

**--RISTORANTE**

INSERT INTO RISTORANTE (nome, descrizione, stelle, costo\_spedizione, img\_profilo, TopPartner, data\_TopPartner, via, civico, comune) VALUES

('Ristorante A', 'Cucina Italiana', 4.5, 5.00, 'img\_a.jpg', TRUE, '2023-01-01 12:00:00', 'Via Roma', 1, 'Roma');

INSERT INTO RISTORANTE (nome, descrizione, stelle, costo\_spedizione, img\_profilo, TopPartner, data\_TopPartner, via, civico, comune) VALUES

('Ristorante B', 'Cucina Cinese', 3.8, 4.50, 'img\_b.jpg', FALSE, NULL, 'Via Milano', 2, 'Milano');

INSERT INTO RISTORANTE (nome, descrizione, stelle, costo\_spedizione, img\_profilo, TopPartner, data\_TopPartner, via, civico, comune) VALUES

('Ristorante C', 'Cucina Giapponese', 4.2, 6.00, 'img\_c.jpg', TRUE, '2023-03-15 18:00:00', 'Via Napoli', 3, 'Napoli');

INSERT INTO RISTORANTE (nome, descrizione, stelle, costo\_spedizione, img\_profilo, TopPartner, data\_TopPartner, via, civico, comune) VALUES

('Ristorante D', 'Cucina Messicana', 4.0, 5.50, 'img\_d.jpg', FALSE, NULL, 'Via Torino', 4, 'Torino');

INSERT INTO RISTORANTE (nome, descrizione, stelle, costo\_spedizione, img\_profilo, TopPartner, data\_TopPartner, via, civico, comune) VALUES

('Ristorante E', 'Cucina Indiana', 4.7, 6.50, 'img\_e.jpg', TRUE, '2023-06-01 20:00:00', 'Via Palermo', 5, 'Palermo');

**--RIDER**

INSERT INTO RIDER (codice, posizione, numero\_consegne, occupato, disponibile, fuori\_servizio, bici, bici\_elettrica, monopattino, km\_batteria) VALUES

('R001', 'Roma', 100, FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, NULL);

INSERT INTO RIDER (codice, posizione, numero\_consegne, occupato, disponibile, fuori\_servizio, bici, bici\_elettrica, monopattino, km\_batteria) VALUES

('R002', 'Milano', 200, TRUE, FALSE, FALSE, FALSE, TRUE, TRUE, 50.00);

INSERT INTO RIDER (codice, posizione, numero\_consegne, occupato, disponibile, fuori\_servizio, bici, bici\_elettrica, monopattino, km\_batteria) VALUES

('R003', 'Napoli', 150, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, 30.00);

INSERT INTO RIDER (codice, posizione, numero\_consegne, occupato, disponibile, fuori\_servizio, bici, bici\_elettrica, monopattino, km\_batteria) VALUES

('R004', 'Torino', 120, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, 40.00);

INSERT INTO RIDER (codice, posizione, numero\_consegne, occupato, disponibile, fuori\_servizio, bici, bici\_elettrica, monopattino, km\_batteria) VALUES

('R005', 'Palermo', 180, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, 60.00);

**--LISTA**

INSERT INTO LISTA (id\_lista, nome, nome\_ristorante) VALUES

('L001', 'Antipasti', 'Ristorante A');

INSERT INTO LISTA (id\_lista, nome, nome\_ristorante) VALUES

('L002', 'Primi Piatti', 'Ristorante B');

INSERT INTO LISTA (id\_lista, nome, nome\_ristorante) VALUES

('L003', 'Secondi Piatti', 'Ristorante C');

INSERT INTO LISTA (id\_lista, nome, nome\_ristorante) VALUES

('L004', 'Dolci', 'Ristorante D');

INSERT INTO LISTA (id\_lista, nome, nome\_ristorante) VALUES

('L005', 'Bevande', 'Ristorante E');

**--PIATTO**

INSERT INTO PIATTO (titolo, id\_lista, immagine, prezzo, sconto) VALUES

('Bruschetta', 'L001', 'img\_bruschetta.jpg', 5.00, 0.00);

INSERT INTO PIATTO (titolo, id\_lista, immagine, prezzo, sconto) VALUES

('Spaghetti Carbonara', 'L002', 'img\_carbonara.jpg', 12.00, 2.00);

INSERT INTO PIATTO (titolo, id\_lista, immagine, prezzo, sconto) VALUES

('Pollo alla Cacciatora', 'L003', 'img\_pollo.jpg', 15.00, 1.50);

INSERT INTO PIATTO (titolo, id\_lista, immagine, prezzo, sconto) VALUES

('Tiramisù', 'L004', 'img\_tiramisu.jpg', 6.00, 0.50);

INSERT INTO PIATTO (titolo, id\_lista, immagine, prezzo, sconto) VALUES

('Vino Rosso', 'L005', 'img\_vino.jpg', 20.00, 3.00);

**--INGREDIENTE**

INSERT INTO INGREDIENTE (nome) VALUES

('Pomodoro');

INSERT INTO INGREDIENTE (nome) VALUES

('Mozzarella');

INSERT INTO INGREDIENTE (nome) VALUES

('Basilico');

INSERT INTO INGREDIENTE (nome) VALUES

('Olio di oliva');

INSERT INTO INGREDIENTE (nome) VALUES

('Sale');

**--PAGAMENTO**

INSERT INTO PAGAMENTO (tipo, email\_utente) VALUES

(1, 'utente1@example.com');

INSERT INTO PAGAMENTO (tipo, email\_utente) VALUES

(2, 'utente2@example.com');

INSERT INTO PAGAMENTO (tipo, email\_utente) VALUES

(3, 'utente3@example.com');

INSERT INTO PAGAMENTO (tipo, email\_utente) VALUES

(4, 'utente4@example.com');

INSERT INTO PAGAMENTO (tipo, email\_utente) VALUES

(5, 'utente5@example.com');

**--SCONTO**

INSERT INTO SCONTO (codice) VALUES

('SCONT01');

INSERT INTO SCONTO (codice) VALUES

('SCONT02');

INSERT INTO SCONTO (codice) VALUES

('SCONT03');

INSERT INTO SCONTO (codice) VALUES

('SCONT04');

INSERT INTO SCONTO (codice) VALUES

('SCONT05');

**--COLLEZIONA**

INSERT INTO COLLEZIONA (email\_utente, codice\_sconto) VALUES

('utente1@example.com', 'SCONT01');

INSERT INTO COLLEZIONA (email\_utente, codice\_sconto) VALUES

('utente2@example.com', 'SCONT02');

INSERT INTO COLLEZIONA (email\_utente, codice\_sconto) VALUES

('utente3@example.com', 'SCONT03');

INSERT INTO COLLEZIONA (email\_utente, codice\_sconto) VALUES

('utente4@example.com', 'SCONT04');

INSERT INTO COLLEZIONA (email\_utente, codice\_sconto) VALUES

('utente5@example.com', 'SCONT05');

**--RECENSIONE\_RISTORANTE**

INSERT INTO RECENSIONE\_RISTORANTE (istante\_recensione, email\_utente, nome\_ristorante, voto, commento) VALUES

('2023-06-01 12:00:00', 'utente1@example.com', 'Ristorante A', 4, 'Ottimo cibo!');

INSERT INTO RECENSIONE\_RISTORANTE (istante\_recensione, email\_utente, nome\_ristorante, voto, commento) VALUES

('2023-06-02 13:00:00', 'utente2@example.com', 'Ristorante B', 3, 'Buon servizio ma cibo mediocre.');

INSERT INTO RECENSIONE\_RISTORANTE (istante\_recensione, email\_utente, nome\_ristorante, voto, commento) VALUES

('2023-06-03 14:00:00', 'utente3@example.com', 'Ristorante C', 5, 'Fantastico!');

INSERT INTO RECENSIONE\_RISTORANTE (istante\_recensione, email\_utente, nome\_ristorante, voto, commento) VALUES

('2023-06-04 15:00:00', 'utente4@example.com', 'Ristorante D', 2, 'Non mi è piaciuto.');

INSERT INTO RECENSIONE\_RISTORANTE (istante\_recensione, email\_utente, nome\_ristorante, voto, commento) VALUES

('2023-06-05 16:00:00', 'utente5@example.com', 'Ristorante E', 4, 'Molto buono, tornerò.');

**--RECENSIONE\_RIDER**

INSERT INTO RECENSIONE\_RIDER (istante\_recensione, email\_utente, codice\_rider, voto, commento) VALUES

('2023-06-01 17:00:00', 'utente1@example.com', 'R001', 4, 'Consegna rapida.');

INSERT INTO RECENSIONE\_RIDER (istante\_recensione, email\_utente, codice\_rider, voto, commento) VALUES

('2023-06-02 18:00:00', 'utente2@example.com', 'R002', 3, 'Puntuale ma scortese.');

INSERT INTO RECENSIONE\_RIDER (istante\_recensione, email\_utente, codice\_rider, voto, commento) VALUES

('2023-06-03 19:00:00', 'utente3@example.com', 'R003', 5, 'Servizio eccellente!');

INSERT INTO RECENSIONE\_RIDER (istante\_recensione, email\_utente, codice\_rider, voto, commento) VALUES

('2023-06-04 20:00:00', 'utente4@example.com', 'R004', 2, 'Molto lento.');

INSERT INTO RECENSIONE\_RIDER (istante\_recensione, email\_utente, codice\_rider, voto, commento) VALUES

('2023-06-05 21:00:00', 'utente5@example.com', 'R005', 4, 'Gentile e puntuale.');

**--ORDINE**

INSERT INTO ORDINE (id\_ordine, email\_utente, nome\_ristorante, codice\_rider, data\_creazione, ora\_consegna, ora\_ritiro, consegnato, ritirato, annullato) VALUES

('O001', 'utente1@example.com', 'Ristorante A', 'R001', '2023-06-01 12:00:00', '2023-06-01 12:30:00', '2023-06-01 12:15:00', TRUE, FALSE, FALSE);

INSERT INTO ORDINE (id\_ordine, email\_utente, nome\_ristorante, codice\_rider, data\_creazione, ora\_consegna, ora\_ritiro, consegnato, ritirato, annullato) VALUES

('O002', 'utente2@example.com', 'Ristorante B', 'R002', '2023-06-02 13:00:00', '2023-06-02 13:45:00', '2023-06-02 13:30:00', TRUE, FALSE, FALSE);

INSERT INTO ORDINE (id\_ordine, email\_utente, nome\_ristorante, codice\_rider, data\_creazione, ora\_consegna, ora\_ritiro, consegnato, ritirato, annullato) VALUES

('O003', 'utente3@example.com', 'Ristorante C', 'R003', '2023-06-03 14:00:00', '2023-06-03 14:20:00', '2023-06-03 14:10:00', TRUE, FALSE, FALSE);

INSERT INTO ORDINE (id\_ordine, email\_utente, nome\_ristorante, codice\_rider, data\_creazione, ora\_consegna, ora\_ritiro, consegnato, ritirato, annullato) VALUES

('O004', 'utente4@example.com', 'Ristorante D', 'R004', '2023-06-04 15:00:00', '2023-06-04 15:50:00', '2023-06-04 15:30:00', FALSE, TRUE, FALSE);

INSERT INTO ORDINE (id\_ordine, email\_utente, nome\_ristorante, codice\_rider, data\_creazione, ora\_consegna, ora\_ritiro, consegnato, ritirato, annullato) VALUES

('O005', 'utente5@example.com', 'Ristorante E', 'R005', '2023-06-05 16:00:00', '2023-06-05 16:40:00', '2023-06-05 16:20:00', TRUE, FALSE, FALSE);

**--CHAT**

INSERT INTO CHAT (id\_chat, id\_ordine, email\_utente, tipo) VALUES

('C001', 'O001', 'utente1@example.com', 1);

INSERT INTO CHAT (id\_chat, id\_ordine, email\_utente, tipo) VALUES

('C002', 'O002', 'utente2@example.com', 2);

INSERT INTO CHAT (id\_chat, id\_ordine, email\_utente, tipo) VALUES

('C003', 'O003', 'utente3@example.com', 1);

INSERT INTO CHAT (id\_chat, id\_ordine, email\_utente, tipo) VALUES

('C004', 'O004', 'utente4@example.com', 2);

INSERT INTO CHAT (id\_chat, id\_ordine, email\_utente, tipo) VALUES

('C005', 'O005', 'utente5@example.com', 1);

**--MESSAGGIO**

INSERT INTO MESSAGGIO (istante\_messaggio, id\_chat, testo) VALUES

('2023-06-01 12:05:00', 'C001', 'Grazie per il vostro ordine!');

INSERT INTO MESSAGGIO (istante\_messaggio, id\_chat, testo) VALUES

('2023-06-02 13:10:00', 'C002', 'Il vostro ordine è in preparazione.');

INSERT INTO MESSAGGIO (istante\_messaggio, id\_chat, testo) VALUES

('2023-06-03 14:15:00', 'C003', 'Il rider sta arrivando.');

INSERT INTO MESSAGGIO (istante\_messaggio, id\_chat, testo) VALUES

('2023-06-04 15:20:00', 'C004', 'Il vostro ordine è stato ritirato.');

INSERT INTO MESSAGGIO (istante\_messaggio, id\_chat, testo) VALUES

('2023-06-05 16:10:00', 'C005', 'Grazie per aver scelto il nostro servizio!');

**--SALVA**

INSERT INTO SALVA (id\_lista, titolo\_piatto) VALUES

('L001', 'Bruschetta');

INSERT INTO SALVA (id\_lista, titolo\_piatto) VALUES

('L002', 'Spaghetti Carbonara');

INSERT INTO SALVA (id\_lista, titolo\_piatto) VALUES

('L003', 'Pollo alla Cacciatora');

INSERT INTO SALVA (id\_lista, titolo\_piatto) VALUES

('L004', 'Tiramisù');

INSERT INTO SALVA (id\_lista, titolo\_piatto) VALUES

('L005', 'Vino Rosso');

**--CONTIENE**

INSERT INTO CONTIENE (titolo\_piatto, id\_ordine) VALUES

('Bruschetta', 'O001');

INSERT INTO CONTIENE (titolo\_piatto, id\_ordine) VALUES

('Spaghetti Carbonara', 'O002');

INSERT INTO CONTIENE (titolo\_piatto, id\_ordine) VALUES

('Pollo alla Cacciatora', 'O003');

INSERT INTO CONTIENE (titolo\_piatto, id\_ordine) VALUES

('Tiramisù', 'O004');

INSERT INTO CONTIENE (titolo\_piatto, id\_ordine) VALUES

('Vino Rosso', 'O005');

**--ALLERGENE**

INSERT INTO ALLERGENE (nome) VALUES ('Glutine');

INSERT INTO ALLERGENE (nome) VALUES ('Latte');

INSERT INTO ALLERGENE (nome) VALUES ('Uova');

INSERT INTO ALLERGENE (nome) VALUES ('Soia');

INSERT INTO ALLERGENE (nome) VALUES ('Arachidi');

**--INCLUDE**

INSERT INTO INCLUDE (nome\_allergene, titolo\_piatto) VALUES ('Glutine', 'Bruschetta');

INSERT INTO INCLUDE (nome\_allergene, titolo\_piatto) VALUES ('Latte', 'Spaghetti Carbonara');

INSERT INTO INCLUDE (nome\_allergene, titolo\_piatto) VALUES ('Uova', 'Pollo alla Cacciatora');

INSERT INTO INCLUDE (nome\_allergene, titolo\_piatto) VALUES ('Soia', 'Tiramisù');

INSERT INTO INCLUDE (nome\_allergene, titolo\_piatto) VALUES ('Arachidi', 'Vino Rosso');

**--CATEGORIA**

INSERT INTO CATEGORIA (nome) VALUES ('Italiano');

INSERT INTO CATEGORIA (nome) VALUES ('Cinese');

INSERT INTO CATEGORIA (nome) VALUES ('Messicano');

INSERT INTO CATEGORIA (nome) VALUES ('Indiano');

INSERT INTO CATEGORIA (nome) VALUES ('Giapponese');

**--APPARTIENE**

INSERT INTO APPARTIENE (nome\_categoria, nome\_ristorante) VALUES ('Italiano', 'Ristorante A');

INSERT INTO APPARTIENE (nome\_categoria, nome\_ristorante) VALUES ('Cinese', 'Ristorante B');

INSERT INTO APPARTIENE (nome\_categoria, nome\_ristorante) VALUES ('Messicano', 'Ristorante D');

INSERT INTO APPARTIENE (nome\_categoria, nome\_ristorante) VALUES ('Indiano', 'Ristorante E');

INSERT INTO APPARTIENE (nome\_categoria, nome\_ristorante) VALUES ('Giapponese', 'Ristorante C');

**--COMPOSTO**

INSERT INTO COMPOSTO (nome\_ingrediente, titolo\_piatto) VALUES ('Pomodoro', 'Bruschetta');

INSERT INTO COMPOSTO (nome\_ingrediente, titolo\_piatto) VALUES ('Basilico', 'Spaghetti Carbonara');

INSERT INTO COMPOSTO (nome\_ingrediente, titolo\_piatto) VALUES ('Olio di oliva', 'Pollo alla Cacciatora');

INSERT INTO COMPOSTO (nome\_ingrediente, titolo\_piatto) VALUES ('Mozzarella', 'Tiramisù');

INSERT INTO COMPOSTO (nome\_ingrediente, titolo\_piatto) VALUES ('Sale', 'Vino Rosso');

## 3.3) Qualche operazione di cancellazione e modifica per verificare i vincoli

* Cancellazione con vincoli di chiave esterna
  + 1) DELETE FROM UTENTE WHERE email = '[utente1@example.com](mailto:utente1@example.com)';
    - L’operazione viene eseguita correttamente, elimina tutto ciò che è annesso all’utente selezionato
  + 2) DELETE FROM PIATTO WHERE titolo = 'Bruschetta';
* Modifica con vincoli di chiave esterna
  + 1) UPDATE UTENTE SET email = 'nuovo\_utente@example.com' WHERE email = '[utente3@example.com](mailto:utente3@example.com)';
    - L’operazione viene eseguita correttamente, la nuova email viene correttamente cambiata sia all’utente che agli ordini che al resto delle tabelle
  + 2) UPDATE INGREDIENTE SET nome = 'Pomodoro Fresco' WHERE nome = 'Pomodoro';
* Aggiungere un vincolo di chiave esterna con ON DELETE CASCADE
  + ALTER TABLE ORDINE

ADD CONSTRAINT fk\_utente

FOREIGN KEY (email\_utente) REFERENCES UTENTE(email)

ON DELETE CASCADE;