Laboratorio Basi di Dati 2023/2024
Progetto di piattaforma di food delivery

Ahed M A ALHijjawi

Table of Content

1. Progettazione Concettuale

1.1 Requisiti Iniziali

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

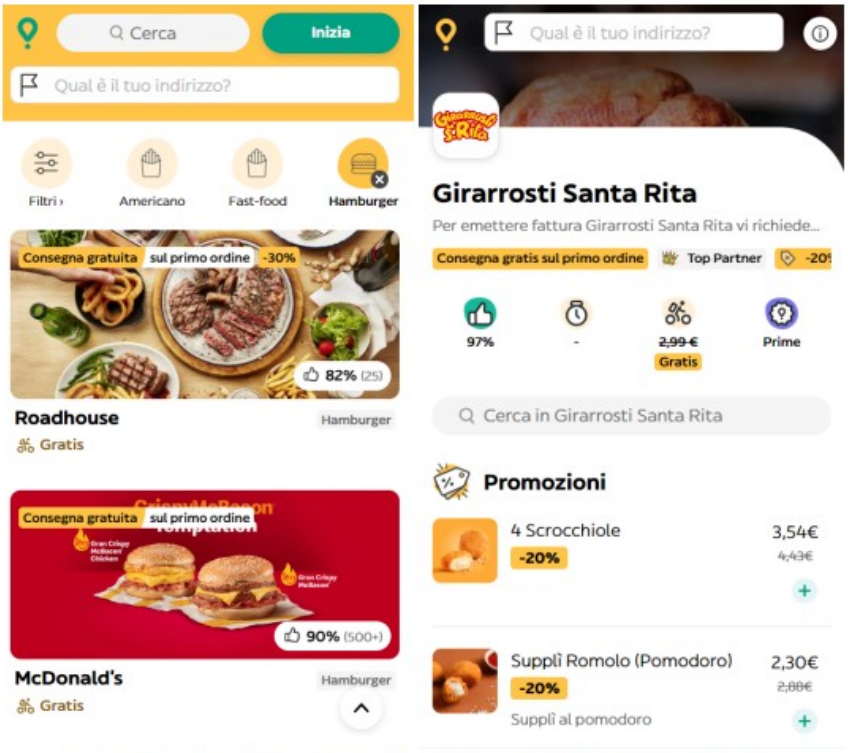


Figura 1 (a) La lista dei ristoranti con filtro "Hamburger". (b) I dettagli di un ristorante.

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, 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.

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 più popolari
- Ristoranti con più recensioni positive
- Clienti che hanno speso di più

1.2 Glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Utente	Individuo registrato al servizio per ordinare	Cliente	Ristorante, Ordine, Rider, Recensione, Valutazione, Pagamento
Ristorante	Ente che si occupa della preparazione degli ordini	Ristoratori	Utente, Ordine, Rider, Recensione, Valutazione, Pagamento
Ordine	Bene richiesto dal cliente	Ordinazione	Utente,Ristorante, Rider
Rider	Individuo che si occupa della consegna dell'ordine	Ciclofattorino, Fattorino	Utente, Ordine, Ristorante, Recensione, Valutazione
Recensione	Commento testuale inteso a giudicare un alloggio o un utente	/	Utente, Ristorante, Rider, ordine
Valutazione	Serie di punteggi su una scala da 1 a 5 riguardanti alcune dimensioni	/	Utente, Ristorante, Rider, ordine
Portata	Elemento che compone l'ordine	Cibo, Pietanze, Piatto	Ordine
Pagamento(?????)	Trasferimento di una somma di denaro in cambio di un servizio/prestazione	/	Utente, Ristorante, Ordine
Posizione (?)	Collocazione spaziale di un oggetto/soggetto	/	Utente, Ristorante Rider,
Top Partner	Categoria speciale di Ristorante	Ristorante	Utente, Ristorante, Valutazione

1.3 Requisiti rivisti e strutturati in gruppi di frasi omogenee

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~~ riders 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~~ ordine e l'utente può ricaricare il proprio borsellino in qualsiasi momento. Inoltre, gli utenti si possono ~~sottoscrivere~~ iscrivere alla 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 da 1 a 5. Il numero di stelle viene 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~~ garantiscono un ottimo servizio (almeno 20 ordini consegnati correttamente, una valutazione ~~clienti~~ utenti 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 ristoranti Top Partner si vuole tenere traccia della data in cui sono entrati a far parte della categoria speciale Top Partner. 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~~ ristoranti . Nel profilo dell'utente si possono ispezionare gli ordini passati ed eventualmente effettuare dei reclami inviando un messaggio al ristorante.

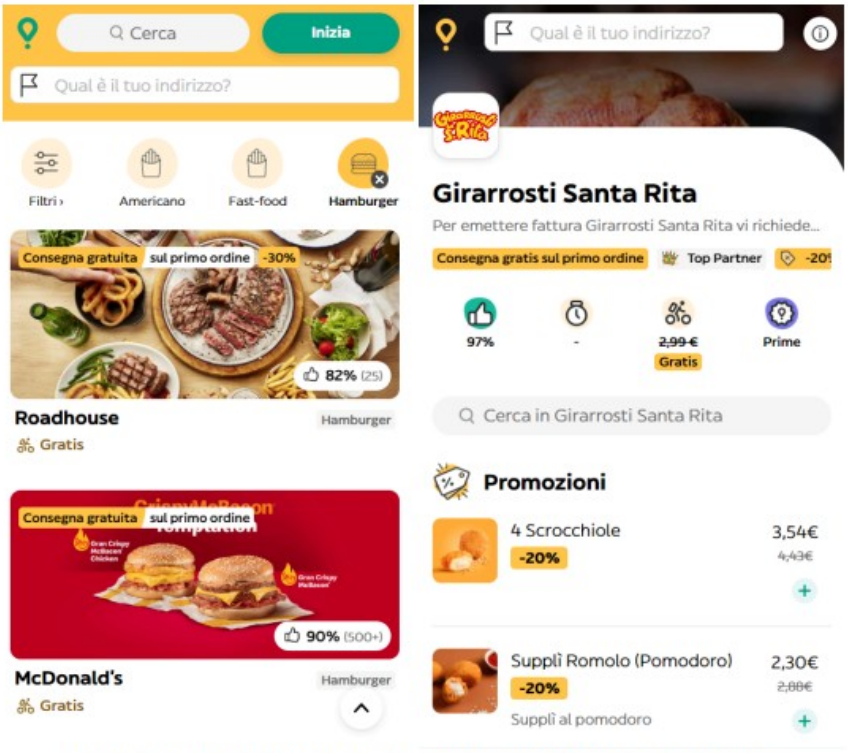


Figura 1 (a) La lista dei ristoranti con filtro "Hamburger". (b) I dettagli di un ristorante.

~~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~~ utente" 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.

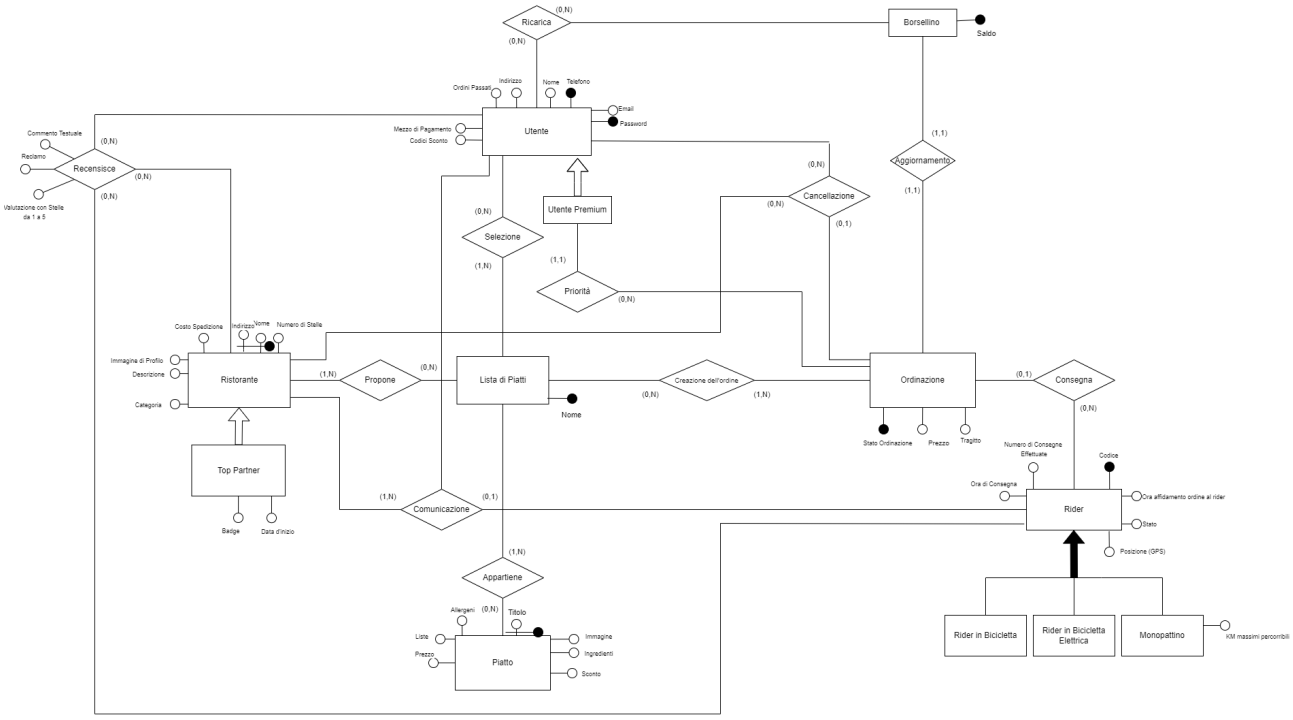
Quando l'ordine è consegnato l'utente può recensire il ristorante e il rider con una valutazione da 1 a 5 e un commento testuale. Il commento testuale è facoltativo. Inoltre è anche presente la possibilità di dare una mancia al rider per la consegna.

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

1.4 Schema E-R + business rules



2.0 Progettazione Logica

2.1 Tavola dei volumi:

Concetto	Tipo	Volume
Utente	E	7 000 000
Utente Premium	E	1 000 000
Borsellino	E	7 000 000
Ristorante	E	115 000
Top Partner	E	11 000
Valutazione	E	2 000 000
Commento Testuale	E	800 000

Valutazione a Stelle	E	1 200 000
Lista di Piatti	E	230 000
Piatto	E	1 150 000
Ordinazione	E	70 000 000
Rider	E	50 000
Bicicletta	E	30 000
Bicicletta Elettrica	E	12 000
Monopattino	E	8 000
Cancellazione	R	5 600 000
Aggiornamento	R	21 000 000
Comunicazione	R	40 000 000
Ricarica	R	21 000 000

Motivazioni:

Abbiamo ipotizzato che un valore stimato per il numero di Utenti del servizio potesse essere 7 milioni.

Abbiamo ipotizzato che un valore stimato per il numero di Utenti Premium del servizio potesse essere circa il 14% del numero di utenti totali (7 000 000) ovvero 1 milione.

Abbiamo ipotizzato che il volume del borsellino deve per forza essere 7 milioni essendo che ogni Utente deve ricaricare il proprio borsellino prima di fare un ordine.

Abbiamo ipotizzato che un valore stimato per il numero di Ristoranti aderenti al servizio potesse essere 115 mila, di cui 11 mila entranti nella categoria speciale Top Partner.

Abbiamo ipotizzato che un valore stimato per il numero di utenti che facciano una valutazione sia 2 milioni. Non tutti gli utenti fanno una valutazione per cui il numero sarà sicuramente molto più piccolo del totale di utenti registrati.

Abbiamo ipotizzato che un valore stimato per il numero di utenti che facciano una valutazione sotto forma di commento testuale sia 800 mila ed è più piccolo del numero di utenti che fanno la valutazione sotto forma di stelle da 1 a 5 essendo che è molto meno impegnativo sia a livello di tempo sia di energie fare una valutazione usando delle stelle (1 200 000).

Abbiamo ipotizzato che un valore stimato per il numero di Liste di Piatti potesse essere 230 mila. Considerando che ogni Ristorante avrà almeno 2 liste di piatti ed è molto difficile stimare una media dato che i ristoranti sono diversi e possono offrire un numero di liste di piatti diversi.

Abbiamo ipotizzato che ogni lista di piatti potesse contenere almeno 7 o 8 piatti per cui il totale dei piatti sarà 1 150 000.

Abbiamo ipotizzato che in un servizio simile al nostro circa 70 000 000 ordinazioni vengono fatte.

Abbiamo ipotizzato che il numero di rider del servizio potesse essere 50 mila. Il mezzo più usato è la bicicletta normale essendo la più economica e disponibile per cui il valore di riders che usano la bici normale è 30 mila. La bici elettrica 12 mila e il monopattino 8 mila per un totale di 50 mila.

Abbiamo ipotizzato che circa il 8% delle ordinazioni totali vengono cancellate per cui il 8% di 70 000 000 è 5 600 000.

Abbiamo ipotizzato che come minimo il volume di aggiornamento deve essere superiore a 7 milioni perchè ogni volta che avviene la ricarica del borsellino il sistema si deve aggiornare per farsi che i soldi caricati si possano usare. Per cui esiste un legame tra Aggiornamento e Ricarica.

Abbiamo ipotizzato che solo una parte degli utenti avrebbe sfruttato il servizio di comunicazione per parlare con i ristoranti o rider.

2.2 Tavola delle Operazioni

N°	Operazione	Tipo	Frequenza
1	Inserimento di una nuova ordinazione	I	15 000/giorno
2	Registrazione nuovo utente	I	10.000/giorno
3	Avviene la scrittura di una recensione	I	2000/giorno
4	Viene assegnata una valutazione	I	5000/giorno
5	Elenco delle liste di piatti prenotabili in un certo periodo di tempo	I	25.000/giorno
6	Gestione sistema di riders	I	20.000/giorno
7	Calcola la classifica dei cibi più popolari	B	4/mese
8	Calcola ristoranti con più recensioni positive	B	4/mese
9	Aggiorna la classifica dei clienti che hanno speso di più	B	1/mese
10	Calcola la classifica dei riders più veloci nelle consegne	B	1/mese

Motivazioni:

Dopo varie ricerche abbiamo ipotizzato che ogni giorno vengono inseriti circa 15.000 ordini al giorno

Il numero di nuovi utenti registrati è in media 10.000 al giorno.

Si suppone che una recensione venga scritta 2000 volte al giorno.

Si suppone che il numero di valutazioni fatte al giorno siano 5000.

In un certo periodo di tempo le liste di piatti prenotabili sono 25.000 al giorno.

I riders vengono gestiti 20.000 volte al giorno.

La classifica dei cibi più popolari viene aggiornata 4 volte al mese.

La classifica dei ristoranti con più recensioni positive viene aggiornata 4 volte al mese.

La classifica dei clienti che hanno speso di più viene aggiornata 1 volta al mese.

La classifica dei riders più veloci nelle consegne viene aggiornata 1 volta al mese.

2.3 Ristrutturazione schema E-R:

2.3.1 Analisi delle ridondanze

Le ridondanze individuate sono:

- Numero di Stelle (RISTORANTE): attributo derivabile da altre entità
- Prezzo (PIATTO): derivabile da altre entità
- Numero recensione(RISTORANTE): attributo derivabile da un conteggio
- Lista (PIATTO): attributo derivabile da altre entità
- Stato (RIDER): attributo derivabile da altre entità

Svolgimento analisi ridondanza “Numero di stelle”.

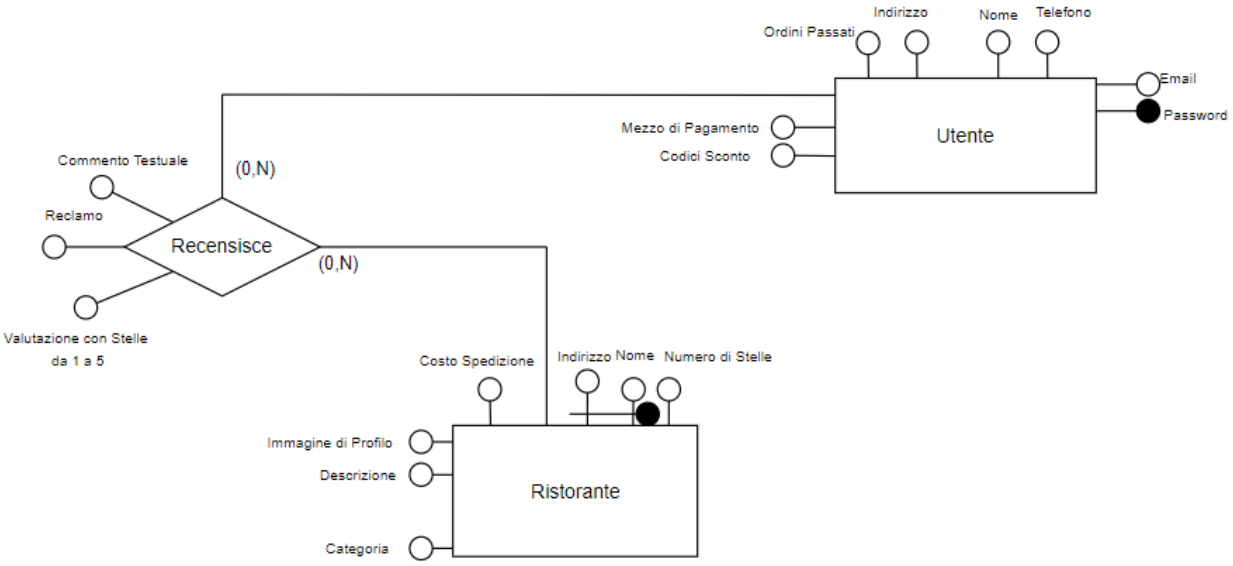
Operazione	**Frequenza**
Viene assegnata una valutazione	5000/giorno

Analisi con ridondanza

1) Tavola dei volumi

Concetto	**Tipo**	**Volume**
Utente	E	7 000 000
Valutazione	R	2 000 000
Ristorante	E	115 000

2) Schema di navigazione



3) Tavola degli accessi

Concetto	**Costrutto**	**Accessi**	**Tipo**
Valutazione	R	1	S
Utente	E	1	L
Ristorante	E	1	L

1. Accesso in scrittura a Valutazione

2. Accesso in lettura a Utente

3. Accesso in lettura a Ristorante

Totale accessi operazione: 2 accessi in lettura, 1 in scrittura

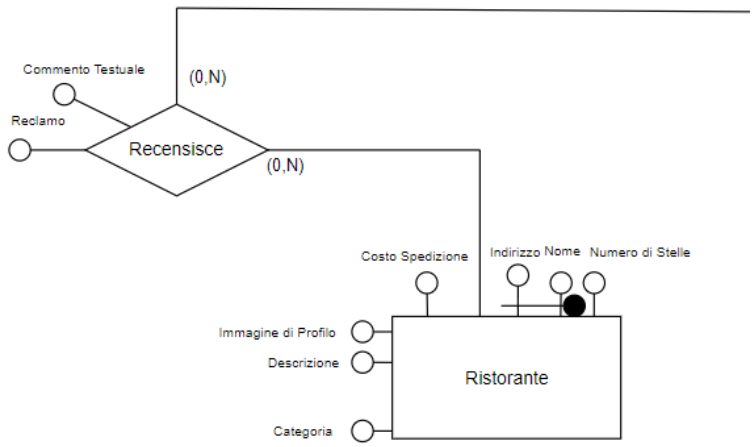
Totale accessi per giorno: 2 * 5000 = 10 000 in lettura, 1 * 5000 = 5000 in scrittura

Analisi senza ridondanza

1) Tavola dei volumi

Concetto	**Tipo**	**Volume**
Valutazione	R	2 000 000
Ristorante	E	115 000

2) Schema di navigazione



3) Tavola degli accessi

Concetto	**Costrutto**	**Accessi**	**Tipo**
Valutazione	R	1	S
Ristorante	E	1	L

1. Accesso in scrittura a Valutazione

2. Accesso in lettura a Ristorante

Totale accessi operazione: 1 accesso in lettura, 1 in scrittura

Totale accessi per giorno: 1 * 5000 = 5000 in lettura, 1 * 5000 = 5000 in scrittura

Analisi Tempo

- Con ridondanza: 10.000 accessi in lettura e 5000 accessi in scrittura (valgono come 10.000)

  Totale = 20.000 accessi al giorno

- Senza ridondanza: 5000 accessi in lettura e 5000 accessi in scrittura (valgono come 10.000)

  Totale = 15.000 accessi al giorno

Analisi Spazio

- Con ridondanza: 4 byte per memorizzare

  Spazio totale = 4 * 2 000 000 = 8 000 000 (≈ 7,63 MB)

- Senza ridondanza: Non necessita spazio aggiuntivo

Analisi Tempo-Spazio

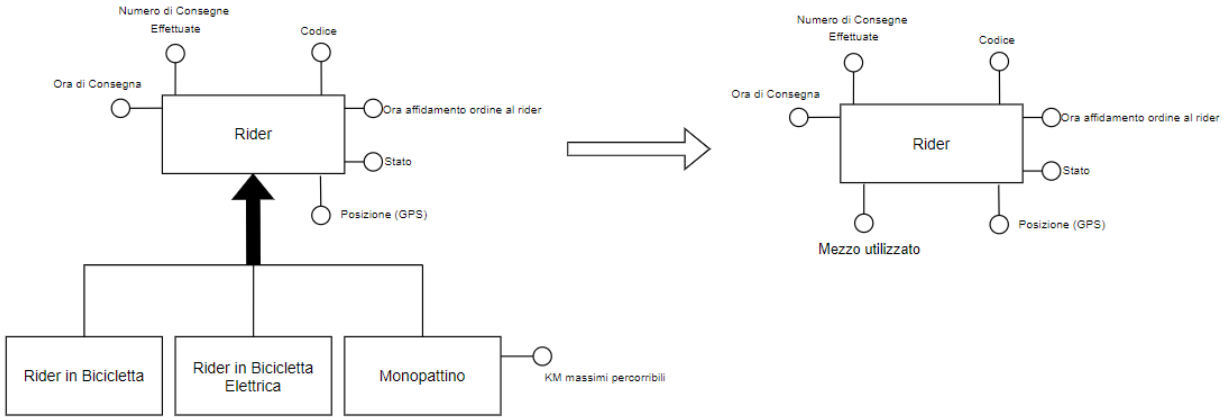
Con Ridondanza	Senza Ridondanza
20.000 accessi al giorno	15.000 accessi al giorno
7,63 MB di spazio aggiuntivo	0 MB di spazio aggiuntivo

2.3.2 Eliminazione delle generalizzazioni

1. Entità padre: Rider

   Entità figlie: Rider in bicicletta, Rider in bicicletta elettrica, Monopattino

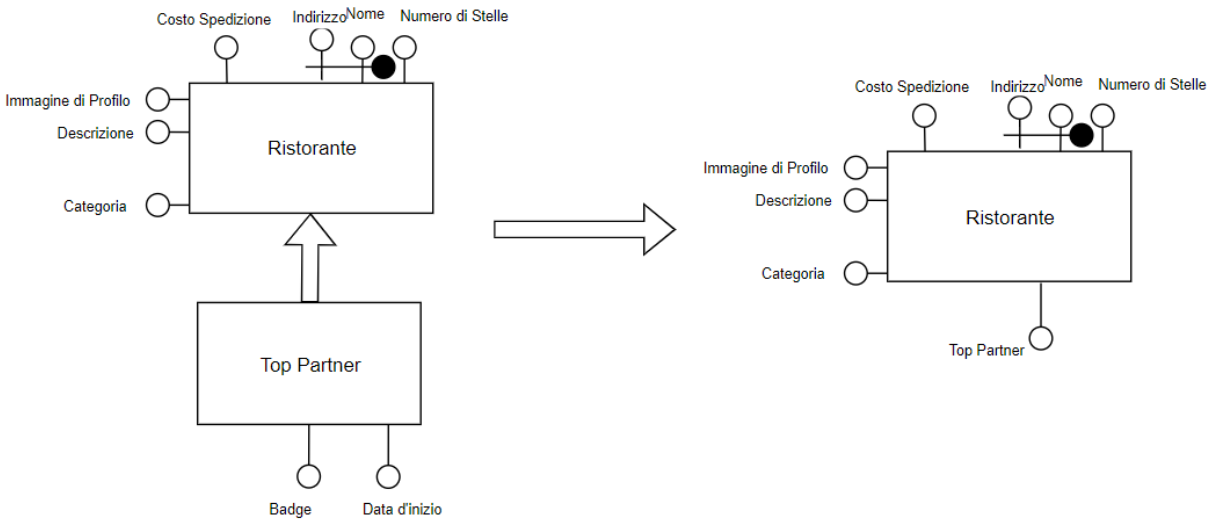
   Primo tipo: accorpamento delle entità figlie della generalizzazione nell'entità genitore.



2. Entità padre: Ristorante

   Entità figlia: Top Partner

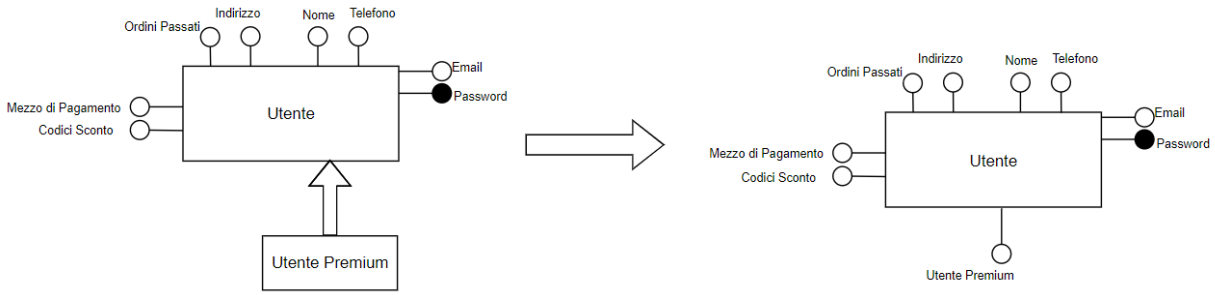
   Primo tipo: accorpamento delle entità figlie della generalizzazione nell'entità genitore.



3. Entità padre: Utente

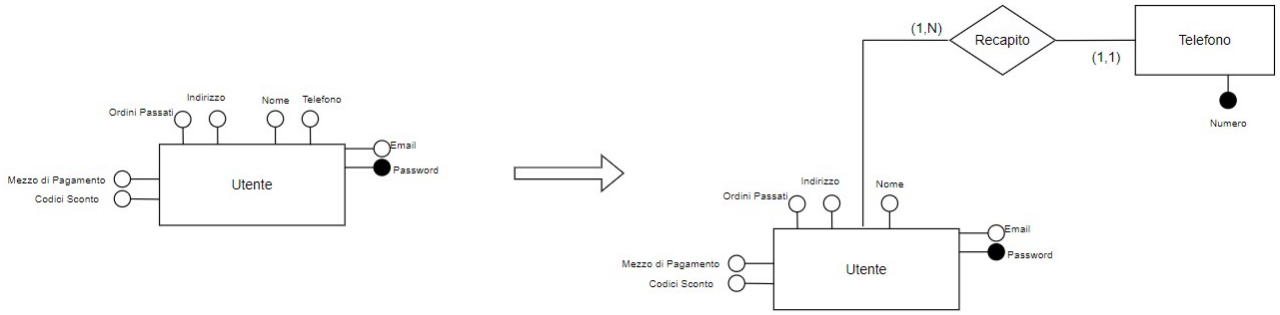
   Entità figlie: Utente Premium

   Primo tipo: accorpamento delle entità figlie della generalizzazione nell'entità genitore

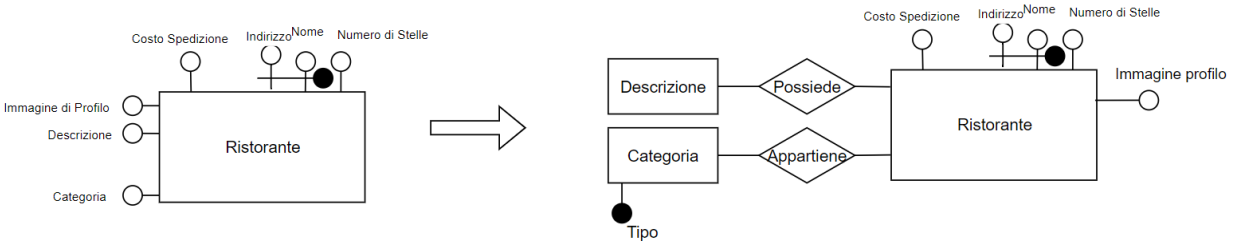


2.3.3 Eventuale partizionamento/accorpamento di entità e associazioni

- Telefono in Utente

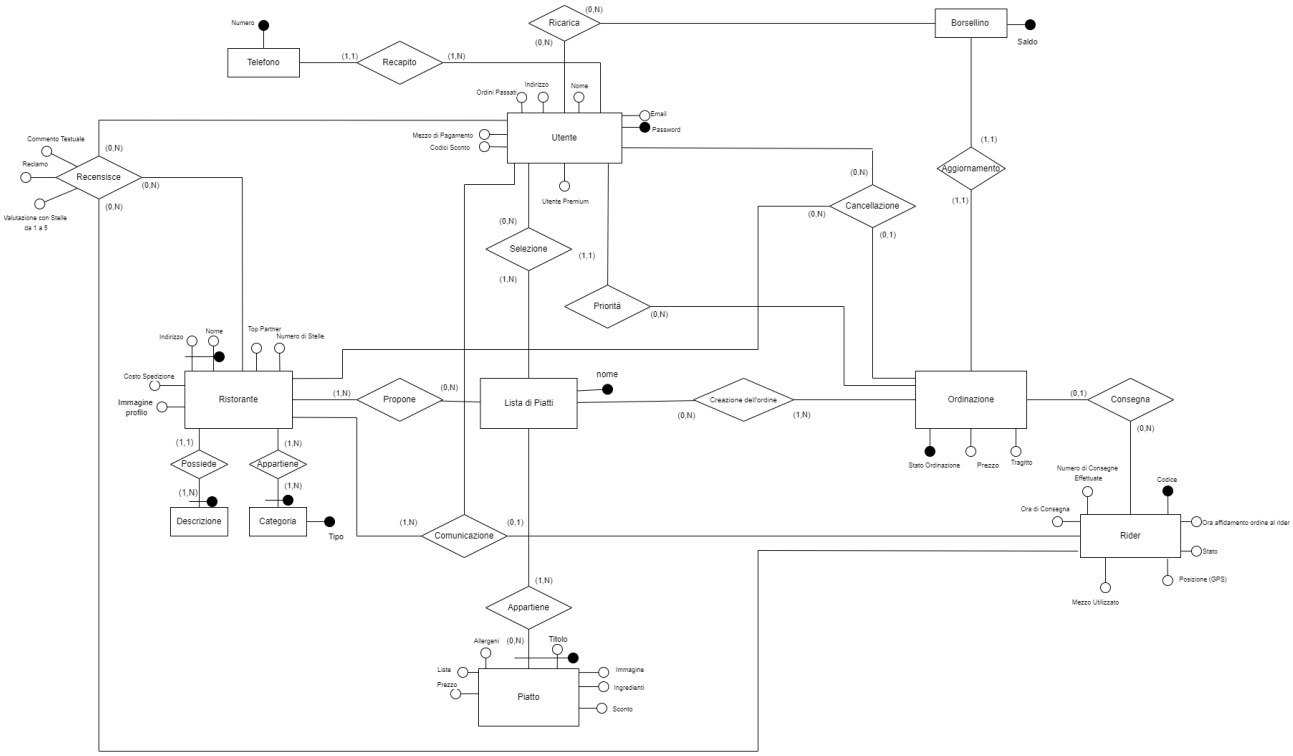


- Descrizione e categoria in Ristorante



2.4 Schema E-R ristrutturato + business rules

2.4.1 Schema E-R ristrutturato



2.5 Schema relazionale + vincoli di integrità referenziale

2.5.1 Schema relazionale

UTENTE(<u>Password</u>, Nome, Email, Indirizzo, Ordini passati, Mezzo di Pagamento, Codici Sconto, Utente Premium)

RISTORANTE(<u>Nome, Indirizzo</u>, Costo Spedizione, Immagine profilo, Numero Stelle, Top Partner)

RIDER(<u>Codice</u>, Ora affidamento ordine, Stato, Posizione, Mezzo Utilizzato, Ora Consegna, Numero consegne effettuate)

PIATTO(<u>Titolo, Lista di Piatti</u>, Immagine, Ingredienti, Sconto, Allergeni, Liste, Prezzo)

ORDINAZIONE(<u>Stato</u>, Prezzo, Tragitto)

RECENSIONE(<u>Utente, Ristorante, Rider</u>, Commento Testuale, Reclamo, Valutazione)

LISTA PIATTI(<u>Nome</u>)

CANCELLAZIONE(<u>Utente, Ordinazione, Ristorante)</u>

CONSEGNA(<u>Ordinazione, Rider</u>)

2.5.2 Vincoli di integrità referenziale

L'attributo Nome in UTENTE e la chiave di RECENSIONE

L'attributo Nome in UTENTE e la chiave di CANCELLAZIONE

L'attributo Nome in RISTORANTE e la chiave di RECENSIONE

L'attributo Codice in RIDER e la chiave di RECENSIONE

L'attributo Stato in ORDINAZIONE e la chiave di CONSEGNA

L'attributo Nome in RISTORANTE e la chiave di CANCELLAZIONE

L'attributo Codice in RIDER e la chiave di CONSEGNA

3. Implementazione

3.1 DDL di creazione del database

```
create table Utente (
    password varchar(20) not null,
    email varchar(50) check (email like '%@%.%'),
    nome varchar(20) not null,
    indirizzo varchar(50) not null,
    ordini_passati integer not null,
    mezzo_pagamento varchar(20) not null
    codici_sconto integer,
    utente_premium boolean not null,
    constraint Utente_PK primary key(password)
);

create table Ristorante (
    nome varchar(20) not null,
    indirizzo varchar(50) not null,
    costo_spedizione integer not null,
    immagine_profilo bytea,
```

```
    top_partner boolean not null,
    numero_stelle integer not null,
    constraint Ristorante_PK primary key(nome, indirizzo)
);

create table Rider (
    codice integer nbot null,
    ora_affidamento_ordine integer,
    stato varchar(20),
    posizione varchar(50) not null,
    mezzo_utilizzato varchar(50) not null,
    ora_consegna integer,
    numero_consegne_effettuate integer not null,
    constraint Rider_PK primary key(codice)
);

create table Ordinazione (
    stato_ordinazione vrchar(20) not null,
    prezzo integer not null,
    tragitto varchar(50) not null,
    constraint Ordinazione_PK primary key(stato_ordinazione)
);

create table Piatto (
    titolo varchar(50) not null,
    allergeni varchar(50) not null,
    liste varchar(50) not null,
    prezzo integer not null,
    sconto integer,
    ingredienti varchar(100) not null,
```

```
    immagine bytea,
    constraint Piatto_PK primary key(titolo)
    constraint Piatto_FK_Lista_Piatti foreign key(titolo)

);

create table Telefono (
    numero integer not null;
    constraint Telefono_PK primary key(numero)

);

create table Borsellino (
    saldo integer not null,
    constraint Borsellino_PK primary key(saldo)
);

create table Lista_Piatti (
    nome varchar(50) not null,
    constraint Lista_Piatti_PK primary key(nome)
);

create table Descrizione (
    ristorante varchar(50) not null,
    constraint Descrizione_PK primary key(ristorante),
    constraint Descrizione_FK_Ristorante foreign key(ristorante)
          references Ristorante(nome, indirizzo) on update cascade on delete
cascade
);
```

```
create table Categoria (
    ristorante varchar(50) not null,
    constraint Descrizione_PK primary key(ristorante),
    constraint Descrizione_FK_Ristorante foreign key(ristorante)
        references Ristorante(nome, indirizzo) on update cascade on delete
cascade
);

create table Recensione (
    ristorante varchar(50) not null,
    rider varchar(50) null not,
    utente varchar(50) not null,
    commento_testuale varchar(500),
    reclamo varchar(500),
    valutazione_stelle integer not null,
    constraint Recensione_PK primary key(nome)
      constraint Recensione_FK foreign key (nome, indirizzo) references
Ristorante(nome, indirizzo) on update cascade on delete cascade
         constraint Recensione_FK foreign key (password) references
Utente(password) on update cascade on delete cascade
    constraint Recensione_FK foreign key (codice) references Rider(codice) on
update cascade on delete cascade
);

create table Recapito (
    telefono integer not null,
    utente varchar(50) not null,
    constraint Recapito_PK primary key(numero)
    constraint Recensione_FK foreign key (numero) references Telefono(numero)
on update cascade on delete cascade
         constraint Recensione_FK foreign key (password) references
Utente(password) on update cascade on delete cascade
```

```
);

create table Ricarica (
    borsellino varchar(50) not null,
    utente varchar(50) not null,
    constraint Ricarica_PK primary key(saldo)
   constraint Ricarica_FK foreign key (saldo) references Borsellino(saldo) on update cascade on delete cascade
    constraint Ricarica_FK foreign key (password) references Utente(password) on update cascade on delete cascade
);

create table Propone (
    ristorante varchar(50) not null,
    lista_piatti varchar(50) not null,
    constraint Propone_PK primary key(nome)
    constraint Propone_FK foreign key (nome) references Lista_Piatti(nome) on update cascade on delete cascade
        constraint Propone_FK foreign key (nome, indirizzo) references Ristorante(nome, indirizzp) on update cascade on delete cascade
);

create table Possiede (
    ristorante varchar(50) not null,
    descrizione varchar(500) not null,
    constraint Possiede_PK primary key(nome, indirizzo)
        constraint Possiede_FK foreign key (nome, indirizzo) references Ristorante(nome, idirizzo) on update cascade on delete cascade
);

create table Appartiene (
```

```
    ristorante varchar(50) not null,
    descrizione varchar(500) not null,
    constraint Appartiene_PK primary key(nome, indirizzo)
       constraint Appartiene_FK foreign key (nome, indirizzo) references Ristorante(nome, indirizzo) on update cascade on delete cascade
);

create table Selezione (
    utente varchar(50) not null,
    lista_piatti varchar(50) not null,
    constraint Selezione_PK primary key (nome)
   constraint Selezione_FK foreign key (nome) references Lista_piatti(nome) on update cascade on delete cascade
    constraint Selezione_FK foreign key (password) references Utente(password) on update cascade on delete cascade
);

create table Priorita (
    utente varchar(50) not null,
    ordinazione varchar(20) not null,
    constraint Priorita_PK primary key (stato_ordinazione)
       constraint Priorita_FK foreign key (stato_ordinazione) references Ordinazione(stato_ordinazione) on update cascade on delete cascade
    constraint Priorita_FK foreign key (password) references Utente(password) on update cascade on delete cascade
);

create table Comunicazione (
    ristorante varchar(50) not null,
    rider varchar(50) null not,
    utente varchar(50) not null,
```

```
    constraint Comunicazione_PK primary key(nome)

      constraint Comunicazione_FK foreign key (nome, indirizzo) references Ristorante(nome, indirizzo) on update cascade on delete cascade

         constraint Comunicazione_FK foreign key (password) references Utente(password) on update cascade on delete cascade

    constraint Comunicazione_FK foreign key (codice) references Rider(codice) on update cascade on delete cascade

);


create table Creazione_Ordine (

    lista_piatti varchar(50) not null,

    ordinazione varchar(20) not null,

    constraint Creazione_Ordine_PK primary key (nome)

    constraint Creazione_Ordine_FK foreign key (stato_ordinazione) references Ordinazione(stato_ordinazione) on update cascade on delete cascade

         constraint Creazione_Ordine_FK foreign key (nome) references Lista_Piatti(nome) on update cascade on delete cascade

);


create table Cancellazione (

    ristorante varchar(50) not null,

    utente varchar(50) not null,

    ordinazione varchar(20) not null,

    constraint Creazione_Ordine_PK primary key (stato_ordinazione)

       constraint Priorita_FK foreign key (stato_ordinazione) references Ordinazione(stato_ordinazione) on update cascade on delete cascade

    constraint Priorita_FK foreign key (password) references Utente(password) on update cascade on delete cascade

      constraint Comunicazione_FK foreign key (nome, indirizzo) references Ristorante(nome, indirizzo) on update cascade on delete cascade

);
```

```
create table Aggiornamento (

    borsellino varchar(20) not null,

    ordinazione varchar(20) not null,

    constraint Creazione_Ordine_PK primary key (stato_ordinazione)

       constraint Priorita_FK foreign key (stato_ordinazione) references Ordinazione(stato_ordinazione) on update cascade on delete cascade

   constraint Priorita_FK foreign key (saldo) references Borsellino(saldo) on update cascade on delete cascade

);


create table Consegna (

    rider varchar(50) null not,

    ordinazione varchar(20) not null,

    constraint Creazione_Ordine_PK primary key (stato_ordinazione)

       constraint Priorita_FK foreign key (stato_ordinazione) references Ordinazione(stato_ordinazione) on update cascade on delete cascade

    constraint Comunicazione_FK foreign key (codice) references Rider(codice) on update cascade on delete cascade

);
```

3.2 DML di popolamento di tutte le tabelle del database

```
insert into Utente values ('asajdjajD324', 'marco@email.com', 'Marco', 'Via Roma 12', 49, 'carta di credito', 263328, true);

insert into Utente values ('326%&fGjilds', 'paolo@gmail.com', 'Paolo', 'Via Potenza 56', 167, 'satispay', 854930, false);

insert into Utente values ('768DF$rg26hk', 'luca@email.com', 'Luca', 'Via Tegas 24', 89, 'carta di credito', null, false);

insert into Utente values ('djisjjs483)K', 'maria@email.com', 'Maria', 'Via Lanza 90', 43, 'bancomat', null, false);

insert into Utente values ('hhsUIHjkj8&%', 'lucia@gmail.com', 'Lucia', 'Via Milano 12', 126, 'bancomat', 128063, true);


insert into Ristorante values ('La Rosa in Fiore', 'Viale Amicis 23', 10, 'file.rosa_in_fiore.jpg', true, 5);
```

```
insert into Ristorante values ('La Volpe Addormentata', 'Via Roma 25', 12, 'file.volpe_addormentata.jpg', true, 4);

insert into Ristorante values ('Fujot', 'Viale Leopardi 14', 8, 'file.fujot.jpg', false, 3);


insert into Rider values (452695, 15:30, 'spedito', 'via tesla', 'bicicletta', 16:00, 19);

insert into Rider values (781340, 19:20, 'consegnato', 'via torino', 'monopattino', 20:00, 71);

insert into Rider values (854714, 20:00, 'preparazione', 'via milano', 'monopattino', 20:20, 54);

insert into Rider values (878543, 18:45, 'spedito', 'via tegas', 'bicicletta elettrica', 19:00, 27);

insert into Rider values (878545, null, 'annullato', 'via lamarmora', 'biciletta', null, 45);


insert into Ordinazione values ('spedito', 14, 'via mazzini');

insert into Ordinazione values ('annullato', 16, 'via milano');

insert into Ordinazione values ('consegnato', 10, 'piazza darmi');

insert into Ordinazione values ('spedito', 12, 'via rossini');


insert into Piatto values ('pasta al pomodoro', 'glutine', 'primi', 9, null, 'farina,pomodoro', 'file.pasta_pomodoro.jpg');

insert into Piatto values ('gnocchi alla romana', 'lattosio, uova, glutine', 'primi', 8, 2, 'formaggio, farina', 'file.gnocchi_romana.jpg');

insert into Piatto values ('gelato', 'lattosio', 'dolci', 3, null, 'latte, frutta', 'file.gelato.jpg');


insert into Telefono values ('340 1587439');

insert into Telefono values ('334 8587413');

insert into Telefono values ('340 8755674');

insert into Telefono values ('348 8544355');

insert into Telefono values ('333 4857854');
```

```
insert into Borsellino values (154);

insert into Borsellino values (74);

insert into Borsellino values (85);

insert into Borsellino values (20);


insert into Lista_Piatti values ('primi');

insert into Lista_Piatti values ('secondi');

insert into Lista_Piatti values ('contorni');

insert into Lista_Piatti values ('dolci');


insert into Descrizione values ('La Rosa in Fiore');

insert into Descrizione values ('La Volpe Addormentata');

insert into Descrizione values ('Il Fujot');

insert into Descrizione values ('La Sirena');


insert into Categoria values ('La Rosa in Fiore');

insert into Categoria values ('La Volpe Addormentata');

insert into Categoria values ('Il Fujot');


insert into Recensione values ('La Sirena', 'Paolo', 'Marco', 'Cibo eccellente, grazie', null, 5);

insert into Recensione values ('La Volpe Addormentata', 'Luca', 'Marta', 'Cibo scadente', 'Non compreprò più da questo ristorante', 2);

insert into Recensione values ('La Rosa in Fiore', 'Raffaele', 'Mario', 'Ottimo cibo', null, 4);

insert into Recensione values ('Il Fujot', 'Paolo', 'Francesca', 'Nulla da dire, perfetto', null, 5);


insert into Recapito values ('340 4589712', 'Paolo');

insert into Recapito values ('338 5454566', 'Mario');
```

```
insert into Recapito values ('348 5689656', 'Vittoria');

insert into Ricarica values (20);
insert into Ricarica values (50);
insert into Ricarica values (100);

insert into Propone values ('La Rosa in Fiore', 'primi');
insert into Propone values ('La Rosa in Fiore', 'dolci');
insert into Propone values ('La Volpe Addormentata', 'contorni');
insert into Propone values ('Il Fujot', 'dolci');

insert into Selezione values ('Paolo', 'primi');
insert into Selezione values ('Marco', 'dolci');
insert into Selezione values ('Lucia', 'contorni');

insert into Priorita values ('Luca', 'spedito');
insert into Priorita values ('Marta', 'consegnato');

insert into Comunicazione values ('La Rosa in Fiore', 'Luca', 'Marco');
insert into Comunicazione values ('La Volpe Addormentata', 'Francesco',
'Raffaele');
insert into Comunicazione values ('Il Fujot', 'Manuel', 'Giuseppe');

insert into Creazione_Ordine values ('primi', 'preparazione');
insert into Creazione_Ordine values ('primi', 'spedito');
insert into Creazione_Ordine values ('dolci', 'spedito');
insert into Creazione_Ordine values ('contorni', 'consegnato');

insert into Cancellazione values ('La Rosa in Fiore', 'Luca', 'preparazione');
```

```
insert into Cancellazione values ('La Volpe Addormentata', 'Marlo', 'preparazione');

insert into Cancellazione values ('Il Fujot', 'Niccolo', 'spedito');


insert into Aggiornamento values (20, 'annullata');

insert into Aggiornamento values (59, 'consegnata');

insert into Aggiornamento values (89, 'spedita');

insert into Aggiornamento values (137, 'consegnata');


insert into Consegna values ('Paolo', 'spedito');

insert into Consegna values ('Marco', 'annullato');

insert into Consegna values ('Maria', 'consegnato');
```

3.3 Qualche operazione di cancellazione e modifica per verificare i vincoli e gli effetti causati da operazioni su chiavi esterne

3.3.1 Operazioni di cancellazione

- Cancellazione delle recensioni il cui Utente è 'Lucia':

  ```
  delete from Recensione where utente like 'Lucia';
  ```

  Cancellando la tupla nella tabella Recensione, che ha come valore dell'attributo utente 'Lucia', vengono cancellate di conseguenza tutte le tuple nel database ad esso collegate con vincoli d'integrità referenziale perché abbiamo aggiunto la clausola on delete cascade a tutti i vincoli di chiave esterna.

- Cancellazione dei piatti il cui titolo è Carbonara:

  ```
  delete from Piatto where titolo like 'pasta al pomodoro';
  ```

  Cancellando la tupla nella tabella Piatto, che ha come valore dell'attributo titolo 'pasta al pomodoro', non vengono cancellate altre tuple nel database perchè nessun attributo ha un vincolo d'integrità referenziale con la tabella Piatto.

3.3.2 Operazioni di modifica

• Modifica dell'attributo telefono nella tabella Recapito:

```
update Recapito

set telefono = '392 8563735'

where utente like 'Paolo';
```

Aggiornando l'attributo telefono nella tabella Recapito, vengono aggiornate di conseguenza tutte le tuple nel database che hanno un vincolo d'integrità referenziale con l'attributo telefono in Recapito per via della clausola on update cascade.

• Modifica dell'attributo ordini_passati nella tabella Utente:

```
update Utente

set ordini_passati = 5

where email like 'marco@email.com';
```

Aggiornando l'attributo ordini_passati nella tabella Utente, non vengono aggiornati altri valori perché nessun'altra tabella ha un vincolo d'integrità referenziale con un attributo di Utente.