Just Eat

Davide Zilio
Francesco Magarotto
Written in LaTeX

21 Novembre 2017

Abstract

I Just Eat è un servizio per la spedizione pasti fondato nel 2005 da Jesper Buch in Regno Unito.
II Il sito permette al cliente di ordinare comodamente le pietanze da lui preferite e farsele recapitare
III ovunque lui voglia: al lavoro o a casa. Oggigiorno, l'azienda è presente in 13 paesi diversi, e a
IV partire dal 2011 il servizio è attivo anche in Italia. Just Eat propone diversi tipi di cucine a seconda
V della città in cui si trova il cliente, e delle attività ristorative nelle vicinanze con cui la società inglese ha
VI stretto una partnership. Ad esempio, nella città di Padova è possibile spaziare da un menù classico,
VII come la pizza, fino all'innovativa cucina Asianfusion. Il servizio è in continua espansione, elaborando
VIII migliaia di ordini al mese, e cerca di essere alla portata di tutti: permette ordinazioni telefoniche e
IX pagamento alla consegna, consentendo così l'utilizzo dei servizio anche a coloro che non sono "nativi
X digitali". A quest'ultimi, invece, viene messa a disposizione una pratica applicazione, disponibile per
XI le piattaforme Android™ e iOS™.

Indice

1 Analisi dei requisiti

Si vuole realizzare una base di dati che contenga e gestisca gli ordini Just Eat™, effettuati dai clienti presso i vari ristoranti localizzati nelle diverse città, individuate tramite il CAP. In particolare, per quanto riguarda i clienti è fondamentale che questi siano registrati sul sistema per permettere loro di lasciare un *feedback*, cioè un commento relativo alla qualità delle pietanze ricevute, ed effettuare gli ordini presso i vari ristoranti. Quest'ultimi si servono di fattorini[1] dipendenti di Just Eat™ che una volta consegnato l'ordine ricevono un punteggio (per ogni ordine completato correttamente, cioè completato senza danneggiamenti dovuti al trasporto, l'impiegato riceve un punto in più. In caso contrario, i danneggiamenti vengono segnalati, specificandone la natura, direttamente dall'utente attraverso l'apertura di un *ticket*. Ogni mese il fattorino più giovane con il punteggio più alto riceve un incremento stipendiale di 0,20 € all'ora. È quindi necessario tener conto dei dati anagrafici del fattorino (nome, cognome, codice fiscale, recapito telefonico) e del suo **stipendio** (composto dalle ore lavorate mensilmente e dallo stipendio all'ora). I clienti sono identificati all'interno della piattaforma attraverso l'indirizzo email che dev'essere confermato, se la conferma non dovesse avvenire, l'utente non può effettuare ordini. Inoltre, per i clienti si ritiene opportuno memorizzare i dati anagrafici come data di nascita, nome, cognome, indirizzo, città di residenza ed eventuali allergie. I ristoranti mettono a disposizione del cliente un menù composto da diverse *pietanze* caratterizzate da un codice univoco, una descrizione, la disponibilità e il costo; la pietanza può essere vegetariana o di altre tipologie, per quanto riguarda le prime si vuole sapere il *metodo di cottura* (al vapore, sottovuoto, al forno). Il cliente nel proprio ordine indica per ogni pietanza desiderata la quantità (espressa in porzioni) di cui necessita, e il metodo di pagamento preferito. Ogni ordine è indentificabile in tutta la piattaforma Just Eat™ tramite un codice univoco e contiene l'orario di ricezione dell'ordine e l'orario indicativo di consegna. Per i ristoranti è d'interesse sapere la partita iva, il nome, la locazione, il genere culinario, il titolare, un eventuale sito web e il numero di telefono.

1.1 Glossario dei termini

Termine	Descrizione	Attributi	Identificatore
Cliente	Persona fisica che attraverso telefono o applicazione web ordina pietanze.	Nome, Cognome, CF, Indirizzo, Email, Allergia	CF
Ordine	Ordine eseguito da un cliente.	Codice, Ora ordinazione, metodo di pagamento	Codice
Fattorino	Persona fisica che consegna l'ordine a casa di un cliente.	Nome, Cognome, CF, Data di nascita, Telefono, Stipendio	CF
Ristorante	Attività imprenditoriale munita di partita iva	Nome, Partita IVA, genere culinario, titolare, indirizzo, sito web e numero di telefono	Partita iva
Città	Luogo di appartenenza del ristorante e città di residenza del cliente	CAP, Nome, Nazione	CAP
Ticket	Ticket virtuale aperto dal cliente per segnalare a JustEat una problematica relativa alla consegna.	Codice, descrizione	Codice
Pietanza	Pietanza realizzata da un ristorante	Codice, nome, prezzo, disponibilità, Descrizione	Codice

[1] Si analizza un contesto internazionale dove i fattorini sono dipendenti Just Eat™. In Italia invece il 90% di questi sono dipendenti del ristorante dove lavorano

2 Progettazione concettuale

2.1 Descrizione testuale delle classi

2.1.1 Città

La classe città rappresenta la città e la nazione del ristorante.

Attributi

- CAP: *intero* ≪*PK*≫ - codice identificativo della città
- Nome: *stringa* - nome della città.
- Nazione: *stringa* - nazione della città.

2.1.2 Cliente

La classe cliente contiene tutte le informazioni di un cliente.

Attributi

- CF: *stringa* ≪*PK*≫ - codice fiscale del cliente.
- Nome: *stringa* - nome del cliente.
- Cognome: *stringa* - cognome del cliente.
- Indirizzo: *stringa* - indirizzo di casa del cliente.

2.1.2.1 Cliente registrato

La classe cliente registrato contiene l'email del cliente registrato a JustEat, la data di attivazione dell'account e le sue eventuali allergie.

Attributi

- Email: *stringa* ≪*PK*≫ - email con la quale il cliente si è registrato.
- Data attivazione: *date* - data di attivazione dell'account del cliente.
- Allergia: *stringa* - possibili allergie di un cliente (attributo multivalore 0:N).

2.1.3 Ristorante

La classe ristorante contiene le informazioni di un ristorante.

Attributi

- PIVA: *stringa* ≪*PK*≫ - partita iva del ristorante.
- Nome: *stringa* - nome del ristorante.
- Titolare: *stringa* - nome del titotare del ristorante.
- Indirizzo: *stringa* - indirizzo del ristorante.
- Sito web: *stringa* - sito web del ristorante.
- Telefono: *stringa* - recapito telefonico ristorante.
- Genere culinario: *stringa* - genere di cucina realizzata nel ristorante.

2.1.4 Pietanza

La classe pietanza rappresenta il cibo che il ristorante propone e che il cliente registrato ordina.

Attributi

- Codice: *stringa* ≪*PK*≫ - codice identificativo della pietanza.
- Nome: *stringa* - nome della pietanza.
- Descrizione: *string* - descrizione della pietanza.
- Costo: *intero* - costo della pietanza.
- Disponibilità: *booleano* - presenza o meno della pietanza.

La pietanza può essere di 2 tipi:

- Altre: pietanze non vegetariane.
- Vegetariana: verdura, legumi, [...].

2.1.5 Fattorino

La classe fattorino contiene le informazioni del fattorino e il suo codice identificativo.

Attributi

- CF: *stringa* ≪*PK*≫ - codice fiscale della fattorino.
- Nome: *stringa* - nome del fattorino.
- Cognome: *stringa* - cognome del fattorino.
- Numero di telefono: *intero* - numero telefonico del fattorino.
- Stipendio: *stringa* - stipendio del fattorino (ore lavorate + stipendio all'ora).

2.1.6 Ordine

La classe ordine è composta da un codice identificativo e dall'ora in cui l'ordine è stato effettuato.

Attributi

- Codice: *stringa* ≪*PK*≫ - codice dell'ordinazione.
- Orario ordine: *time* - ora dell'ordinazione.
- Orario consegna: *time* - ora della consegna dell'ordine.

2.1.7 Ticket

La classe ticket è identificata da un codice e contiene il commento relativo al danneggiamento della pietanza ordinata.

Attributi

- Codice: *stringa* ≪*PK*≫ - codice del ticket.
- Commento: *stringa* - commento relativo al ticket.

2.2 Descrizione testuale delle associazioni

2.2.1 Città - Cliente: “Residenza”

Molteplicità N:1 Una città può essere città di residenza per più clienti; ogni cliente risiede in una ed una sola città.

Totalità: parziale verso Cliente / totale verso Città Una città può non avere clienti, ogni cliente deve risiedere in una e una sola città.

2.2.2 Città-Ristorante: “Appartenenza”

Molteplicità N:1 Una città può avere più ristoranti, il ristorante appartiene ad una e una sola città.

Totalità: parziale verso Ristorante / totale verso Città Una città può non avere un ristorante, il ristorante deve appartenere ad una e una sola città.

2.2.3 Ristorante-Pietanza: “Proposta”

Molteplicità N:1 Un ristorante propone più pietanze, la pietanza viene proposta solo da un ristorante (quello di riferimento).

Totalità: totale verso Pietanza / totale verso Ristorante Un ristorante propone almeno una pietanza, la pietanza viene proposta da un solo ristorante.

2.2.4 Pietanza-Ordine: “Composizione”

Molteplicità N:N Una pietanza può comporre un ordine, un ordine è composto da una o più pietanze.

Totalità: totale verso Pietanza / parziale verso Ordine Una pietanza può non comporre un ordine, un ordine deve essere composto da almeno una pietanza.

NB: L’associazione *Composizione* verrà chiamata *dettagli_ordine* nel codice sql per comodità e praticità.

2.2.5 Ordine-Fattorino: “Trasporto”

Molteplicità 1:N Un ordine viene trasportato da un solo fattorino, il fattorino può trasportare più ordini.

Totalità: totale verso Fattorino / parziale verso Ordine Un ordine deve essere trasportato da un solo fattorino, il fattorino può non avere ordini da trasportare.

2.2.6 Cliente registrato-Ordine: “Effettua”

Molteplicità N:1 Il cliente registrato può effettuare più ordini, un ordine viene effettuato da un cliente.

Totalità: parziale verso Ordine / totale verso Cliente registrato Un cliente registrato può non effettuare alcun ordine, l’ordine deve essere effettuato da un solo cliente registrato.

2.2.7 Cliente registrato-Ristorante: “Feedback”

L’associazione feedback conterrà un attributo **commento**, che sarà la descrizione del feedback.

Molteplicità N:N Un cliente registrato può rilasciare più feedback al ristorante, il ristorante può ottenere più feedback dai suoi clienti registrati.

Totalità: parziale verso Ristorante / parziale verso Cliente registrato Un cliente registrato può non rilasciare alcun feedback al ristorante, il ristorante può non ottenere nessun feedback dai suoi clienti registrati.

2.2.8 Ticket-Ordine: “Relativo”

Molteplicità 1:1 Un ticket si riferisce ad un solo ordine, un ordine può avere al massimo un ticket.

Totalità: totale verso Ordine / parziale verso Ticket Un ticket è riferito ad un solo ordine, un ordine può non avere un ticket.

2.2.9 Ticket-Cliente registrato: “Apertura”

Molteplicità 1:N Il ticket può essere aperto dal solo cliente interessato, relativo all’ordine effettuato; Il cliente registrato può aprire più ticket.

Totalità: parziale verso Ticket / totale verso Cliente registrato Il cliente può non aprire un ticket, il ticket deve essere aperto da un cliente registrato.

2.3 Output: Modello E/R

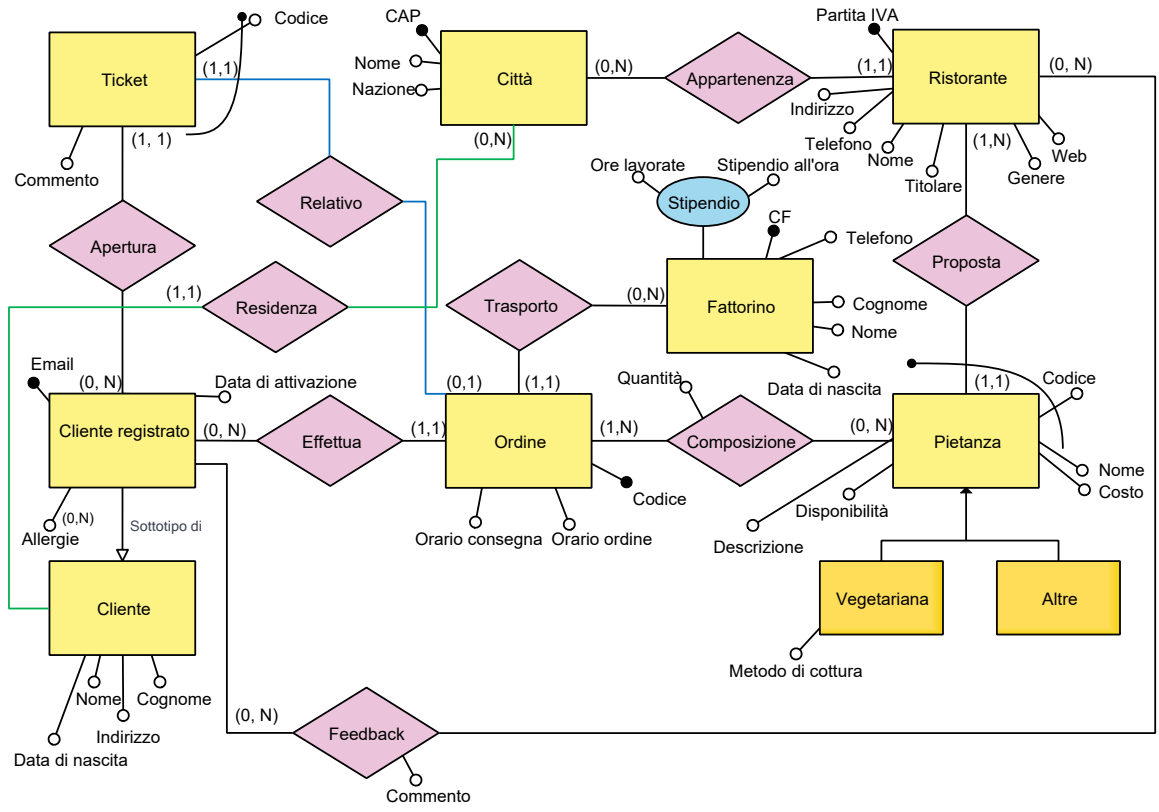


Figura 1: Modello ER rappresentativo della base di dati

3 Progettazione logica

3.1 Descrizione testuale dello schema relazionale

Partendo dallo schema Entità/Relazioni (Figura 1) non ristrutturato possiamo fare alcune osservazioni relativamente alle entità `Utente` e `Pietanza`. In particolare, nell’analisi dei requisiti è specificato che un utente registrato può soffrire di una o più allergie, è pertanto necessario trasformare questo attributo `multivalore` *Allergia* in un’entità indipendente legata tramite una relazione `molti a molti`. Infatti, ogni utente può soffrire di una o più allergie e la stessa allergia potrebbe essere la medesima per più clienti. Per quanto riguarda l’entità *Pietanza* invece, introduciamo un attributo `Tipologia` che indica il tipo di pietanza (vegana, composta da carne, ecc), se questa pientanza apparterrà alle tipologia vegetariana

allora sarà necessario conoscere il metodo di cottura della pietanza. Pertanto, la relazione `Pietanza` avrà l'attributo *metodo di cottura* che eventualmente avrà valore nullo.

3.2 Ristrutturazione schema E/R

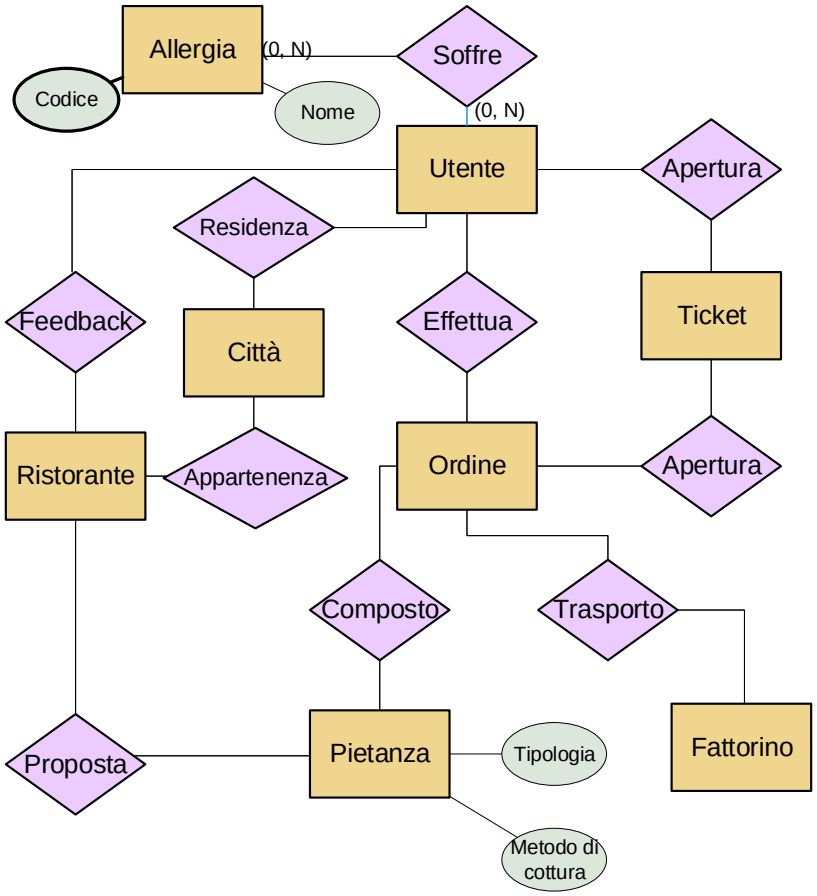


Figura 2: Modello E/R ristrutturato, si omettono gli attributi rispetto al modello precedente poichè sono gli stessi mentre si introducono le nuove entità

3.2.1 Cliente

L'attributo multivalore Allergie[2] diventerà una nuova entità legata a cliente.

Attributi aggiornati

- Email: *stringa* ≪*PK*≫ - email con la quale il cliente si è registrato.

3.2.2 Allergia

L'entità Allergia conterrà il nome dell'allergia e il relativo codice identificativo.

Attributi

- Codice: *intero* ≪*PK*≫ - codice dell'allergia.
- Nome: *stringa* - nome dell'allergia.

Cliente-Allergia: "Soffre"

[2]Le allergie non sono strettamente dipendenti dai clienti.

Molteplicità N:N Il cliente può soffrire di più allergie, un'allergia può appartenere a più clienti.

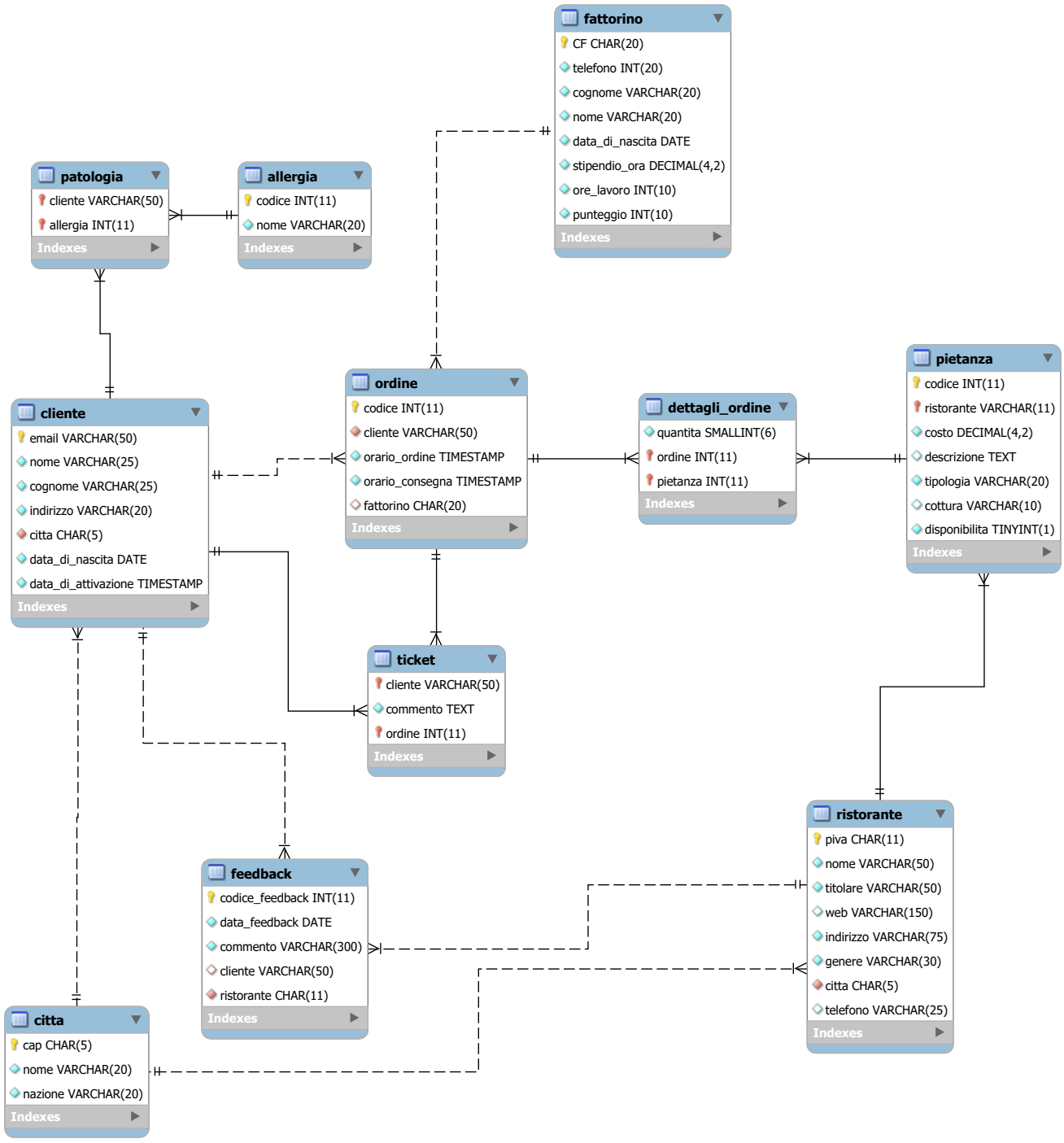
Totalità: parziale verso Allergia/ Totale verso Cliente Un cliente può non avere allergie, un'allergia appartiene ad almeno un cliente.

3.2.3 Pietanza

L'entità Pietanza rappresenta l'insieme di pietanze che l'utente può ordinare. Rispetto alla precedente entità aggiungiamo gli attributi:

- Tipologia: *stringa* - tipologia del cibo (i.e. vegana, vegetariana, [...])
- Metodo di cottura: *stringa* - indica il metodo di cottura della pietanza vegetariana/vegana.

3.3 Schema logico



3.4 Modello relazionale

Gli attributi marcati con * sono attrbuti facoltativi che possono avere valore `NULL`.
Gli attributi (o l'attributo) sottolineati/o indicano una `CHIAVE`.
Gli attributi marcati in *corsivo* indicano una `CHIAVE ESTERNA`.

Allergia(Codice, Nome)
Città(CAP, Nome, Nazione)
Cliente(Email, Nome, Cognome, Indirizzo, *Città*, Data di nascita, Data attivazione account, Allergia*)
Ordine(Codice, Orario ordine, Orario Consegna, *Cliente*)
Ristorante(PIVA, Nome, Titolare, Web, Indirizzo, Telefono, Genere, *Città*)
Fattorino(CF, Nome, Cognome, Data di nascita, Telefono, Stipendio/h, Ore lavorate)
Pietanza(Codice, *Ristorante*, Costo, Disponibilità, Descrizione, Tipologia, Metodo di cottura*)
Dettagli Ordine(Ordine, Pietanza, Quantità)
Ticket(Codice, Ordine, Commento)
Feedback(Codice, Cliente, Ristorante, Data commento, Commento)

4 Query

1. Query che ritorna tutti i codici degli ordini (conclusi e non) e le email dei clienti che li hanno effettuati, nei quali sono stati ordinate solamente pietanze vegetariane cotte a vapore.

```
1  SELECT o.codice, o.cliente
2  FROM ordine o JOIN cliente c ON o.cliente = c.email
3  WHERE NOT EXISTS (
4      SELECT *
5      FROM ordine o JOIN dettagli_ordine d_o ON o.codice = d_o.ordine JOIN pietanza p ON d_o.pietanza = p.
       codice
6      WHERE cottura <> 'VAPORE' AND tipologia <> 'VEGETARIANA' )
```

MySQL ha restituito un insieme vuoto (i.e. zero righe). (La query ha impiegato 0.0029 secondi.)

2. Query che ritorna i codici fiscali di tutti i fattorini che hanno portato a termine tutti gli ordini senza nessun ticket aperto ed hanno ottenuto un punteggio superiore a 10.

```
1  SELECT f.CF
2  FROM fattorino f LEFT JOIN ordine o ON f.cf = o.codice
3  WHERE punteggio > 10 AND f.CF NOT IN (
4          SELECT o.fattorino
5          FROM ordine o JOIN ticket t ON o.codice = t.ordine
6      )
```

CF
ZKYDEV87A45T560I

3. Query che ritorna i cognomi dei clienti che hanno effettuato un ordine e soffrono di particolari allergie (realizzata con viste).

```
1  SELECT cognome
2  FROM cliente AS c, o1, a1
3  WHERE c.cognome = o1.ordini AND  c.cognome = a1.allergie AND o1.ordini = a1.allergie
```

cognome
Rowling
Rowling
Auditore

4. Query che ritorna l'email del cliente e il codice della pietanza che ha ordinato più volte in ordini diversi (realizzata con vista).

```
SELECT cliente, occorrenze as numero_pietanza
FROM occ JOIN ( SELECT MAX(occorrenze) as oc FROM occ ) AS Mocc ON Mocc.oc = occorrenze
```

email	numero_pietanza
timoty96@yahoo.com	4

5. Query che ritorna l'email clienti che non hanno mai effettuato un ordine nonostante possano farlo (data_di_attivazione non nulla). Realizziamo questa query anche in algebra relazionale per completezza.
   $R1 := \pi_{email}((cliente \bowtie_{email = cliente \wedge data\_di\_attivazione \neg NULL} ordine) \bowtie_{codice = ordine} dettagli\_ordine)$
   $R2 := \pi_{email}(cliente) - R1$
   Siccome il DBMS MariaDB/MySQL non mette a disposizione l'operazione di intersezione (keyword `INTERSECT`), faremo uso della keyword `IN` preceduta dalla negazione `NOT`.

```
SELECT email
FROM cliente
WHERE email NOT IN (
    SELECT email
    FROM cliente c JOIN ordine o ON c.email = o.cliente JOIN dettagli_ordine d_o ON o.codice = d_o.
    ordine
    WHERE c.data_di_attivazione IS NOT NULL
)
```

email
caty65@libero.com
timoty96@yahoo.com

6. Query che ritorna il nome della pietanza ordinata più volte e il suo numero di occorrenze (realizzata con vista).

```
SELECT nome, maxP as occorrenze
FROM pmax JOIN ( SELECT MAX(maxP) as maP FROM pmax) AS mocc ON mocc.maP = maxP
```

nome	occorrenze
Risotto ai frutti di mare	2

5 Eventi, viste, trigger e funzioni

5.1 Eventi

Nell'analisi dei requisiti viene espressamente richiesto che ogni mese il fattorino più giovane con punteggio più alto riceva un incremento stipendiale di 0,20 € all'ora. Le operazioni sulla tabella fattorino vengono delegate alla procedure `AumentoStip`.

```
DELIMITER $$
CREATE EVENT `MigliorImpiegato`
ON SCHEDULE EVERY 1 MONTH STARTS '2018-03-01 00:00:00'
DO BEGIN
UPDATE fattorino SET stipendio_ora = stipendio_ora + 0.20 WHERE CF = (SELECT CF FROM fattorino WHERE CF NOT IN
     (SELECT f1.CF FROM fattorino f1 CROSS JOIN fattorino f2 WHERE f1.data_di_nascita < f2.data_di_nascita AND
     f1.punteggio < f2.punteggio AND f1.CF != f2.CF));
UPDATE fattorino SET punteggio = 0;
END $$
DELIMITER ;
```

5.2 Viste

```
CREATE OR REPLACE VIEW o1 AS
SELECT cognome AS ordini
FROM cliente c JOIN ordine o ON c.email = o.cliente JOIN dettagli_ordine as do ON o.codice = do.ordine
```

```
CREATE OR REPLACE VIEW a1 AS
SELECT cognome AS allergie
FROM cliente c JOIN patologia p ON c.email = p.cliente JOIN allergia a ON p.allergia = a.codice
```

```
CREATE OR REPLACE VIEW occ AS
SELECT COUNT(o.cliente) AS occorrenze, o.cliente AS cliente
FROM dettagli_ordine d_o JOIN dettagli_ordine d_or ON (d_o.ordine <> d_or.ordine AND d_o.pietanza = d_or.
    pietanza) JOIN ordine o ON d_o.ordine = o.codice JOIN ordine o2 ON (o.codice <> o2.codice AND o.cliente =
    o2.cliente)
WHERE d_o.quantita >= 1
```

```
CREATE OR REPLACE VIEW pmax AS
SELECT p.nome AS nome, COUNT(d_o.pietanza) AS maxP
FROM dettagli_ordine d_o JOIN pietanza p ON d_o.pietanza = p.codice
GROUP BY p.nome
```

5.3 Trigger

1. Un problema che si viene a creare nella tabella `Pietanza` con l'impiego di una chiave primaria composta è il seguente: Ogni pietanza ha un codice che associato al risorante mi permette di identificare la pientanza. Questo codice però non può essere settato come `AUTO_INCREMENT` perchè MySQL non supporta questa funzione con chiavi primarie composte da più attributi.

```
DELIMITER $$
    CREATE TRIGGER `MANUAL_AUTOINCREMENT` BEFORE INSERT ON `pietanza`
        FOR EACH ROW BEGIN
        SET NEW.codice = (
            SELECT MAX(codice) + 1
            FROM pietanza WHERE ristorante = NEW.ristorante
        );
    END $$
DELIMITER ;
```

Il trigger `MANUAL_AUTOINCREMENT` é stato creato.

2. Trigger che verifica che il cliente abbia attivato il suo account (`data di attivazione` diversa da `null`)

```
DELIMITER $$
CREATE TRIGGER ‘ClienteAttivato‘ BEFORE INSERT ON ‘ordine‘
 FOR EACH ROW BEGIN
   IF((SELECT cliente.data_di_attivazione FROM cliente = cliente) IS NULL)
    THEN
     SIGNAL SQLSTATE ’45000’ SET message_text = ’Cliente non puo’’  effettuare ordini se non ha attivato
      il suo account’;
  END IF;
END $$
DELIMITER ;
```

Il trigger `ClienteAttivato` é stato creato.

5.4 Funzioni

1. Funzione che restituisce il prezzo minimo o massimo delle pietanze ordinate.

```
CREATE FUNCTION ‘prezzoMinMax‘(‘cos‘ DECIMAL(4,2), ‘MinMax‘ ENUM(’min’, ’max’)) RETURNS DECIMAL(4,2)
DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
IF(MinMax = ’min’) THEN
    SELECT MIN(p.costo) INTO cos
    FROM dettagli_ordine dor JOIN pietanza p ON dor.pietanza = p.codice;
    RETURN cos;
ELSE
    SELECT MAX(p.costo) INTO cos
    FROM dettagli_ordine dor JOIN pietanza p ON dor.pietanza = p.codice;
    RETURN cos;
END IF;
```

prezzoMinMax
10.00

2. Funzione che restituisce il costo totale dell’ordine di un cliente.

```
DELIMITER $$
FUNCTION ‘costoTotale‘(‘tot‘ DECIMAL(4,2), ‘email‘ VARCHAR(30)) RETURNS decimal(4,2)
COMMENT ’Il cliente è passato per parametro (email).’
BEGIN
SELECT SUM(p.costo * d_o.quantita) INTO tot
FROM ordine o JOIN dettagli_ordine d_o on o.codice = d_o.ordine JOIN pietanza p on d_o.pietanza = p.
    codice
WHERE o.cliente = email;
RETURN tot;
END$$
DELIMITER ;
```

```
SET @p0='0.0'; SET @p1='marcocostantino@libero.it';
SELECT 'costoTotale'(@p0, @p1) AS 'costoTotale';
```

costoTotale
37.98