UNIVERSITÀ
degli STUDI
di CATANIA

Progetto di basi di dati
Gestione della prenotazione di un Ristorante

Nome e Cognome: Andrea Filippo Salemi

Matricola: 1000012617

Anno accademico: 2021/2022

Indice

1. Presentazione Progetto

In questo progetto bisognerà realizzare una base di dati in cui si dovranno gestire le prenotazioni di un ristorante in tutte le sue sfaccettature.

Bisognerà realizzare diverse entità e relazioni su cui si possa creare una prenotazione per il cliente.

La progettazione sarà effettuata tenendo in considerazione che si avranno dei volumi e un diagramma E-R con l'equivalente schema relazionale in modo vero-simile a ciò che potrebbe rappresentare la realtà in un ipotetico ristorante.

1.1. Software Usati

I software scelti per realizzare la base di dati sono:
phpMyAdmin: Software utilizzato per costruire la basi di dati utilizzando il linguaggio SQL (mySql o MariaDB) .
DRAW.IO: un software usato per progettare il diagramma E-R.

1.2. Strategia D'uso

La strategia da usare per la progettazione della base di dati utilizzando il diagramma E-R è quella della strategia Top-Down :
è un tipo di strategia in cui si parte da uno scheletro composto dalle opportune entità collegate attraverso le relazioni, si aggiungono le opportune cardinalità e gli opportuni attributi.

2. Specifiche sui Dati

I dati hanno diversi termini rappresentate da delle relazioni e da delle entità identificate:
Il **cliente** richiede di prenotare un **tavolo**:
Ogni cliente ha un codice fiscale (CF) che identifica in maniera univoca la persona così da non creare delle ambiguità tra gli omonimi, ogni cliente ha inoltre un nome e un cognome e anche il gruppo;
Ogni **prenotazione** può essere effettuata da un singolo **cliente** ma ogni ristorante può avere diverse prenotazioni (in tavoli diversi e da clienti diversi), ogni prenotazione ha inoltre una data, un cliente e un tavolo.

Ogni **Piatto** può essere **servito** in nessuno o più **tavoli** e **ordinato** da più nessuno o più **clienti,** il piatto è identificato da un codice, da un nome e dalla descrizione del prodotto, e l'id dello che ha fatto questo piatto.

In ogni **tavolo** possono essere seduti o nessuno o più **clienti,** ogni tavolo è identificato posti(disponibili), descrizione.

Ogni **Ordinazione** può essere effettuata per uno o più piatti ma tutti i **clienti** possono ordinare lo stesso **piatto**, l'ordinazione contiene più piatti:

piatto che è stato ordinato;

dataP contiene la data della prenotazione;

oraP contiene l'ora della prenotazione;

il tavolo in cui il piatto deve essere portato;

Ogni **chef** può avere preparato uno o più **piatti**, ma chiaramente un piatto deve essere fatto da uno e un solo **chef:**

Ogni chef deve aver un id, un nome, un cognome e la specializzazione in cui lui si è specializzato (per esempio: Dolci, tavola calda, torte e così via).

Gruppo di persone con la quale possono contenere 1 o più clienti, ma una persona non può fare parte di più un gruppo; quindi, un gruppo può essere formato da più persone(clienti), ogni gruppo ha un id del gruppo ovvero un codice identificativo univoco, il capo gruppo(prenotante) e il nome del gruppo.

2.1 Glossario Dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Cliente	Cliente che prenota e che si siede al tavolo	Persona	Gruppo
Tavolo	Tavolo in cui i clienti si devono sedere	Posti a sedere	Prenotazione, Ordinazione
Piatto	I piatti che devono essere preparati e serviti	Delizia, cibo, mangiare	Ordinazione, Chef
Prenotazione	Prenotazione posti	Riserva, impegno	Tavolo, gruppo
Ordinazione	Ordinare il cibo, da mangiare	Scegliere	Piatto, Tavolo
Chef	Colui/colei che cucina il cibo	Cuoco	Piatto
Gruppo	Gruppo Di Clienti	Famiglia, Amici, Parenti, Colleghi	Cliente, prenotazione

2.2 Altre informazioni (frasi a caratteri generali)

Un cliente prenota un tavolo di solito per i propri amici o i propri famigliari quindi in ogni tavolo c'è una sola persona che prenota e tutti gli altri vanno a sedersi;

quindi, un cliente "prenotante" è un cliente ma non tutti i clienti sono prenotanti;

Invece per convenzione tutti i tavoli hanno lo stesso numero di posti e quindi qualora dovesse esserci un gruppo di persone superiore al numero di posti si aggiunge un secondo tavolo.

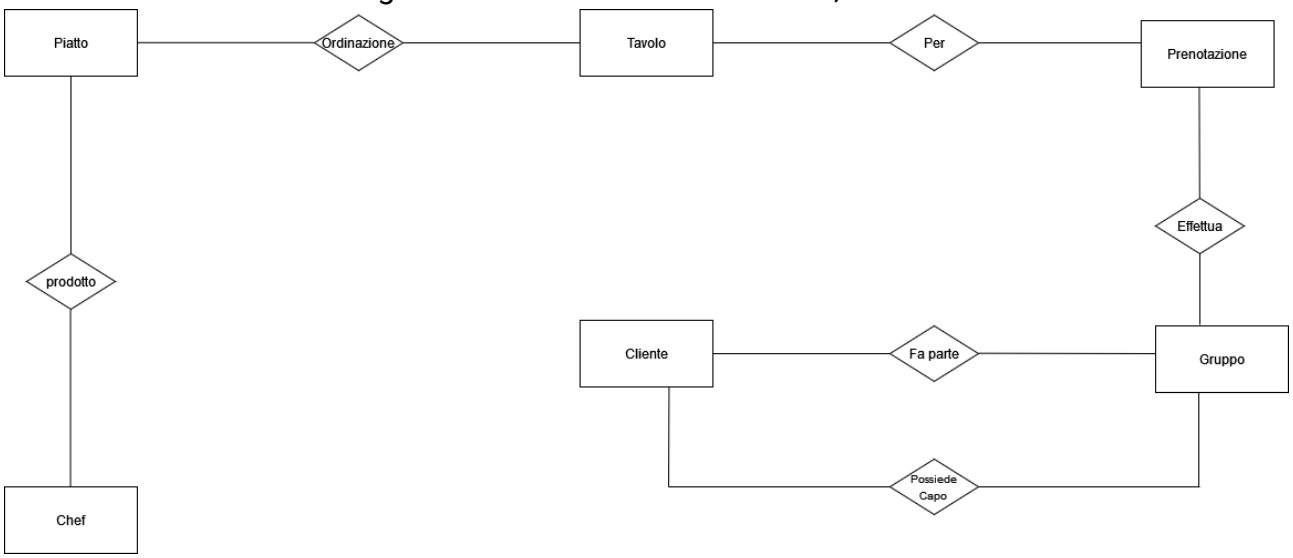
C'è da considerare, inoltre, che i clienti possono effettuare solo una prenotazione per sera.

Alla fine di ogni mangiata si decide di pagare insieme, ogni gruppo paga il suo conto in cassa , quindi il prenotante va anche a pagare il conto per gli altri (si suppone per comodità che ogni gruppo dà i soldi al prenotante), l'operatore che sta in cassa va a calcolarsi quello che è stato preso per ogni gruppo di cliente di ogni tavolo, quindi viene calcolato ogni qual volta in cassa, in maniera autonoma (facendo la somma dei piatti), ovviamente ogni piatto a un proprio prezzo e può essere ordinato più volte (calcolo può essere gestita con una procedura).

3. Schema Concettuale

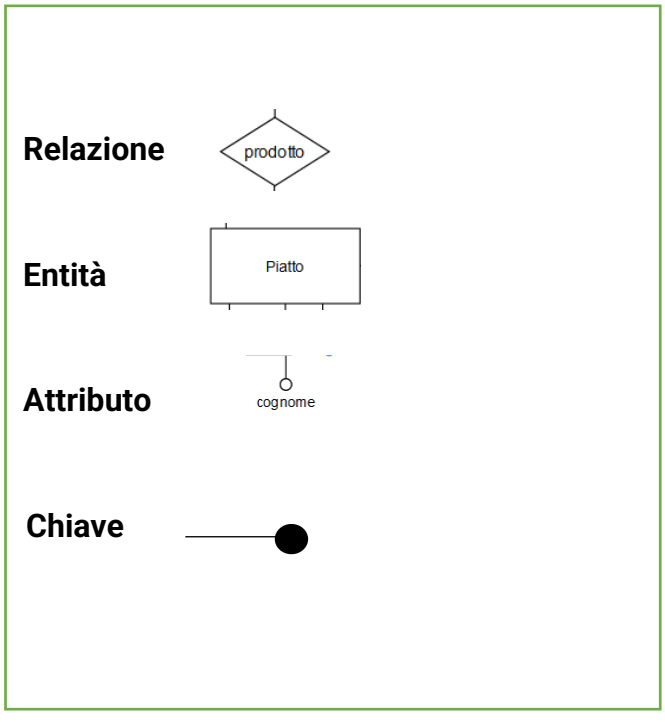
3.1 Schema Scheletro

Lo schema scheletro è diagramma E-R senza le cardinalità, senza attributi e senza chiavi.

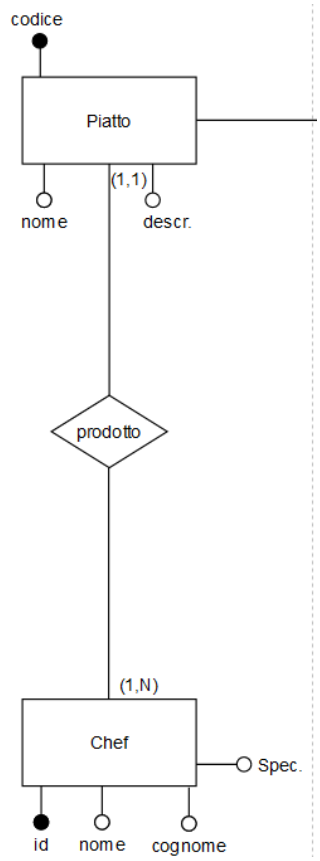


3.2 Raffinamenti

Seguendo la strategia Top-Down bisogna inserire gli opportuni raffinamenti, ovvero, come già detto durante la spiegazione della strategia Top-Down, saranno inserite gli attributi, le cardinalità e per non creare confusione durante la lettura verrà illustrata una piccola legenda così da favorire agilmente la lettura dei diagrammi durante i raffinamenti.



3.2.1 Raffinamento Attributi tra Piatto e Chef



Si definisce gli attributi nelle due relazioni:

piatto (**Codice,** nome e descrizione);

Chef (**id**, nome, cognome e specializzazione);

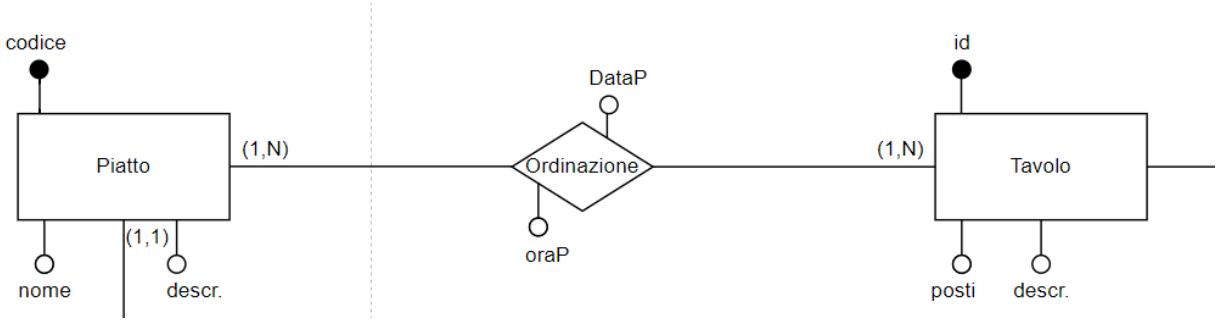
Note

Il ragionamento per la cardinalità è il seguente:

un piatto può essere prodotto esclusivamente da un SOLO chef,

ogni chef può preparare almeno uno o più piatti (si suppone che tutti gli chef preparino sempre qualcosa);

3.2.2 Raffinamento Attributi tra Piatto e Tavolo



Si definisce l'entità tavolo la quale possiederà 3 attributi (id, posti e descrizione).

Si definiscono degli attributi anche nella relazione Ordinazione (dataP e oraP).

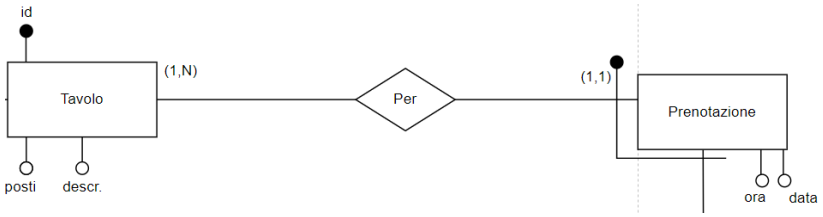
NOTE

Il ragionamento delle cardinalità è il seguente:

1 o più piatti possono essere ordinati per uno o più tavoli, ricordando che uno o più piatti, si può omettere la dataP in Ordinazione poiché risolta ridondante.

Un altro attributo la quale può essere considerato ridondante è posto che si calcola con un trigger a ogni inserimento si decrementa di uno, si può, per comodità imporre che i tavoli hanno tutti lo stesso numero, quando il numero di posti raggiunge lo zero, bisognerà cambiare e mandare i nuovi clienti all'altro tavolo per comodità si inserirà che ogni tavolo abbia 8 posti così da poter ospitare un gruppo medio di persone.

3.2.3 Raffinamento Attributi tra Tavolo e Prenotazione

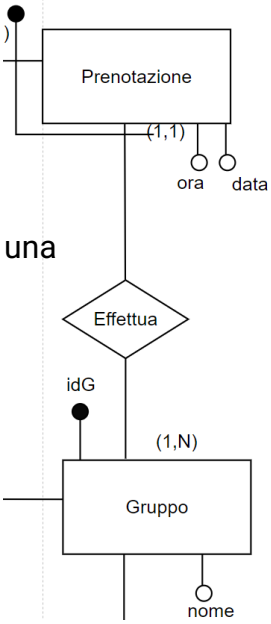


Si definiscono gli attributi in prenotazione, due dei quali sono delle chiavi esterne le quali coincidono con le chiavi esterne e gli altri due attributi (ora e data).

Note

Tavolo e prenotazioni hanno una relazione di tipo 1 a N, ovvero, per ogni prenotazione possono essere richieste (almeno) uno o più tavoli, poiché si suppone che se c'è un gruppo di persone superiore a 8 può prendere più di un tavolo ma un tavolo può essere prenotato solo da un gruppo.

3.2.4 Raffinamento Attributi tra Prenotazione e Gruppo



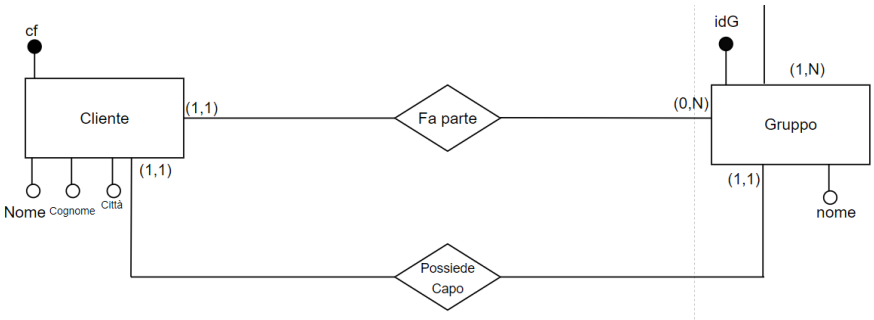
L’entità Gruppo ha due attributi (idG e nome).

Note

Le cardinalità sono di tipo 1 a N ovvero un o più gruppi possono fare una

Prenotazione, mentre una prenotazione può essere fatta da più gruppi.

3.2.5 Raffinamento Attributi tra Cliente e Gruppo



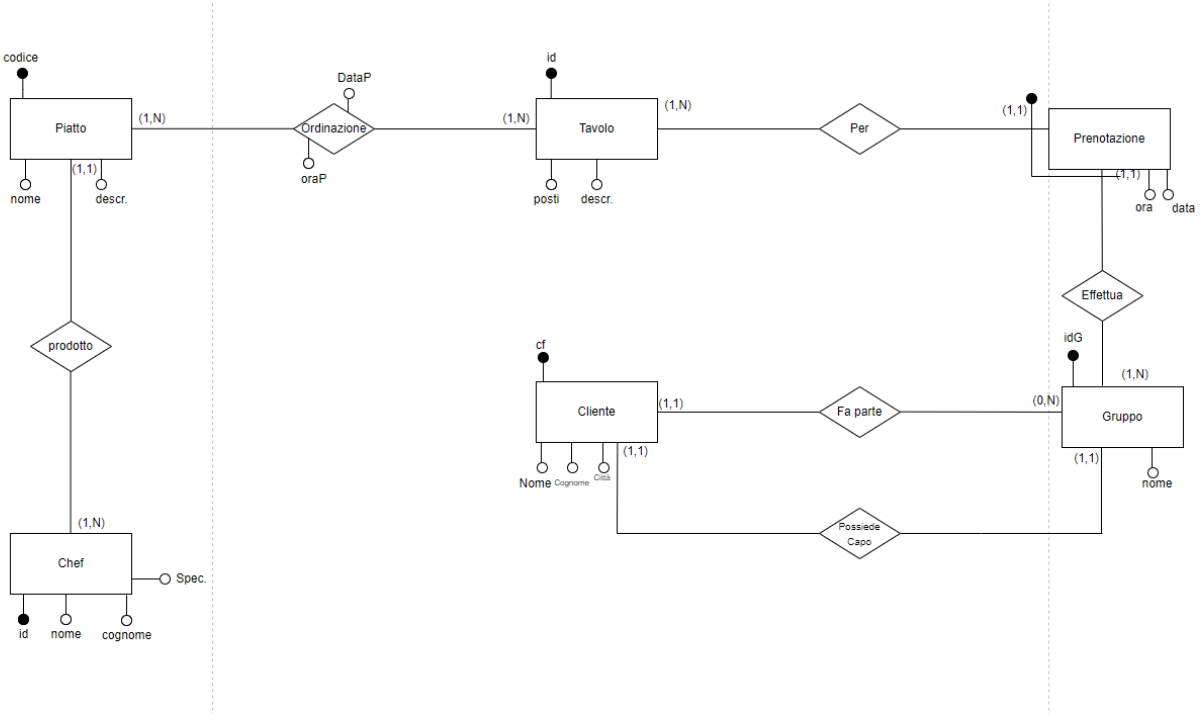
Si inseriscono i vari attributi (cf, nome, cognome, città), le due associazioni e le cardinalità

Note

Tra Gruppo (di clienti) e Cliente ci sono due associazioni, uno è “fa parte” e l’altra “possiede capo (Gruppo)” queste due relazioni consentono di gestire le relazioni tra i due entità

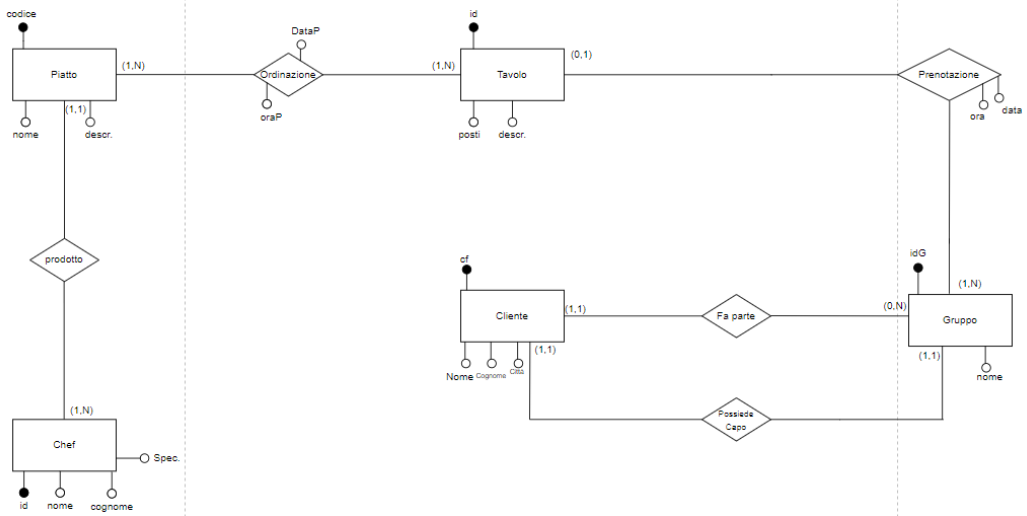
3.3 Schema Finale (senza ristrutturazione)

questo è il risultato di tutti i vari raffinamenti tra le varie entità, chiaramente, è possibile effettuare una ristrutturazione.



3.4 Schema Finale (con ristrutturazione)

Nello schema ristrutturato è stato semplicemente eliminato due relazioni e sono state in un certo senso assorbito dall'entità prenotazione trasformandosi esso in una relazione che unisce il Gruppo col Tavolo.



4. Vincoli non esprimibili

I vincoli non esprimibili sono dal diagramma E-R:

Numero Totale di persone
Non è possibile sapere qual è il numero totale di posti che verranno occupati

Costo Totale
Non abbiamo accesso diretto al costo totale per ciascun gruppo, ma sarà una somma di ciascun piatto.

Attributi che potranno essere calcolati:

Costo Totale Gruppo
Facendo la somma è possibile calcolare il totale speso di ogni gruppo.

5. Schema Relazionale e Dizionario dei dati

Dal diagramma E-R si può ricavare lo schema relazionale, e di conseguenza, dallo schema relazionale si può ricavare il diagramma dei dati e la tavola dei volumi.

5.1 Schema Relazionale

Nello schema relazionale con il **grassetto** saranno indicate le chiavi primarie e con le linee tratteggiate saranno indicate le chiave esterne. Invece gli attributi che sono sia chiavi interne che esterne avranno sia il tratteggiato che il grassetto.

Cliente (**cf**, cognome, nome, città, gruppo)
Tavolo (**id**, posti, descrizione)
Prenotazione (**tavolo, prenotante**, data, ora)
Piatto (**codice**, nome, prezzo, chef)
Ordinazione (**piatto, tavolo, dataP, oraP**)
Chef (**id**, nome, cognome, specializzazione)
Gruppo (**idG**, nome, prenotante)

5.2 Dizionario dei Dati

Entità	Descrizione	Attributi	Identificatori
Cliente	Colui che comprerà	cf, cognome, nome , città, gruppo	CF
Tavolo	Il posto in cui ci si siedono le persone	id, posti, descrizione	id
Prenotazione	Prenota un tavolo	data, ora, tavolo, prenotante	Tavolo, prenotante
Piatto	Pietanza cucinata	codice, nome, prezzo, chef	codice
Ordinazione	Cibo che viene ordinato	piatto, dataP, oraP , tavolo	piatto, tavolo, dataP, oraP
Chef	Cuoco che cucina i piatti	id, nome, cognome, specializzazione	id
Gruppo	Insieme di clienti/Persone	idG, nome,prenotante	idG

6. Progettazione logica

Durante la progettazione logica verranno mostrate le operazioni da effettuare, l'analisi delle ridondanze e infine si provvederà alla progettazione delle tabelle con codice sql.

6.1 Operazioni

Operazioni
1.Tutte le date delle prenotazioni fatte dai clienti di una determinata città.
2. Piatti ordinati da un gruppo.
3. Conteggio dei gruppi che hanno ordinato un certo tipo di piatto.
4. I nomi dei piatti preparati da uno chef.
5. Stampa per ogni tavolo i piatti ordinati.
6. Per ogni prenotante stampa il nome e cognome.
7. Stampa tutti i gruppi che NON hanno alcun ordinato un tipo di piatto.
8. Inserimento di un nuovo chef.
9. Per ogni gruppo stampa il totale da pagare.

6.2 Tabella dei volumi

Si può fare una stima approssimativa della tavola dei volumi facendo delle stime

Concetto	Tipo	Volumi
Cliente	**E**	80
Tavolo	**E**	15
Prenotazione	**R**	10
Piatto	**E**	20
Ordinazione	**R**	200
Chef	**E**	8
Gruppo	**E**	10

6.3 Tabella delle operazioni

Indice	Tipo	Frequenza giornaliera
Op. 1	**I**	**5**
Op. 2	**I**	**100**
Op. 3	**I**	**1**
Op. 4	**I**	**20**
Op. 5	**B**	**3**
Op. 6	**B**	**20**
Op. 7	**I**	**2**
Op. 8	**I**	**1**
Op. 9	**B**	**5**

Note

Ci sono due tipi di operazioni ovvero
-“I” sta per interattivo dove l’utente finale interagisce inserendo dei dati all’interno dei parametri;

-“B” sta per Batch il quale indica un tipo di operazione che non ha bisogno di alcun dato da inserire da parte dell’utente finale.

6.4 Analisi delle ridondanze

L'unica ridondanza che lo schema possiede è l'attributo **posti**, ma essendo che non c'è alcun calcolo di ciò non è necessaria eliminarla poiché i calcoli delle operazioni totali rimarrebbero gli stessi con o senza l'attributo "posti".
Invece nell'ultima operazione sembrerebbe che lo schema abbia bisogno di un attributo ridondante in cui si effettua il calcolo del costo totale.

Si eseguono i calcoli:

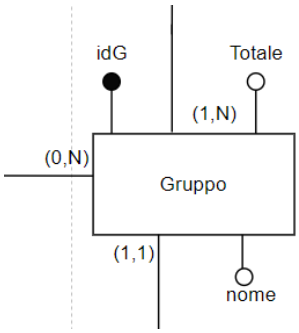
Senza Ridondanza in gruppo (senza l'attributo totale):
1 Lettura in gruppo
1 Lettura in prenotazione
1 Lettura in tavolo
1 Lettura in piatto
1 Lettura in clienti
Tot : (4 Letture + (1 Lettura)*80)*5=420L/gg

Con Ridondanza in gruppo (con l'attributo totale):
1 Lettura in gruppo
1 Lettura
1 Lettura in prenotazione
1 Lettura in tavolo
1 Lettura in piatto
1 Scrittura in gruppo
Tot: (4 Letture + 1 Scritture) *5 = 30Lettute

Quindi conviene effettivamente introdurre nella tabella gruppo un attributo che andrà ad indicare il "totale" speso.

6.5 Aggiunta attributo Totale all'entità gruppo

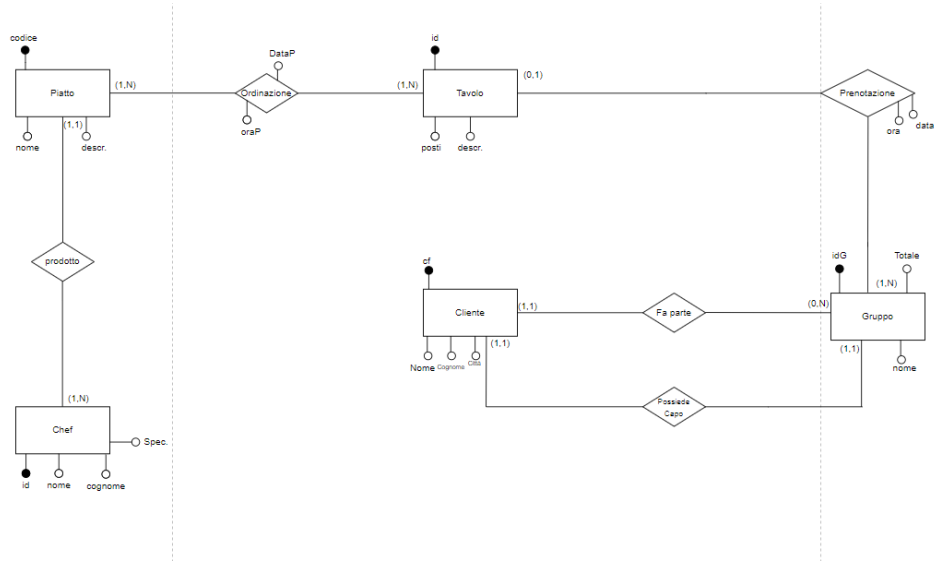
Diagramma E-R dell'entità dopo l'aggiunta dell'attributo "totale"



Note

Un trigger andrà ad aggiornare il valore pian piano che i clienti ordineranno dei piatti.

Lo Schema E-R con l'aggiunta della ridondanza :



Quindi bisognerà, di conseguenza aggiungere anche il l'attributo allo schema relazionale che quindi diventa:

Cliente (**cf**, cognome, nome, città, gruppo)
Tavolo (**id**, posti, descrizione)
Prenotazione (**tavolo, prenotante**, data, ora)
Piatto (**codice**, nome, prezzo, chef)
Ordinazione (**piatto, tavolo, dataP, oraP**)
Chef (**id**, nome, cognome, specializzazione)
Gruppo (**idG**, nome, totale, prenotante)

Nota

Il prenotante è il capogruppo.

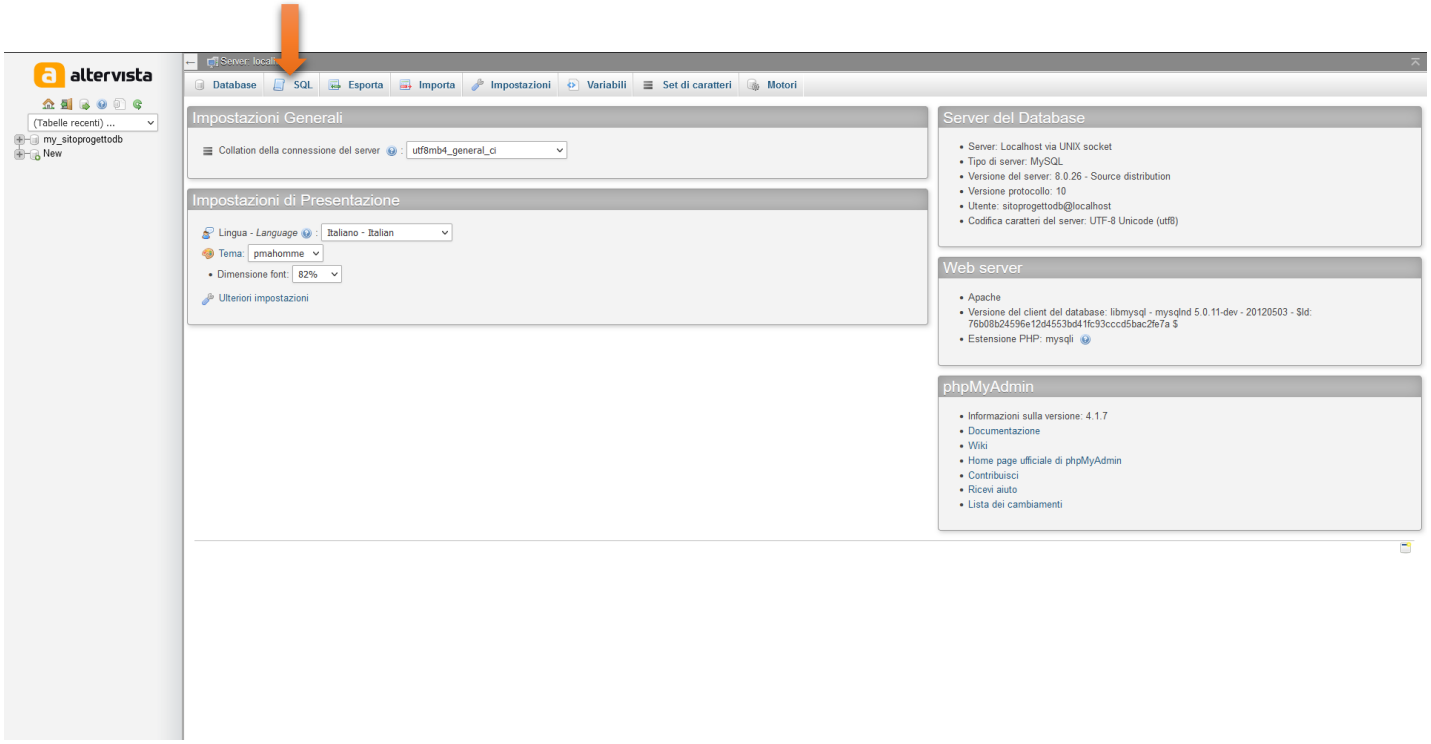
7.Progettazione fisica

La progettazione fisica di una base di dati consiste nel memorizzare delle strutture dati ed effettuare delle operazioni utilizzando un determinato DBMS.

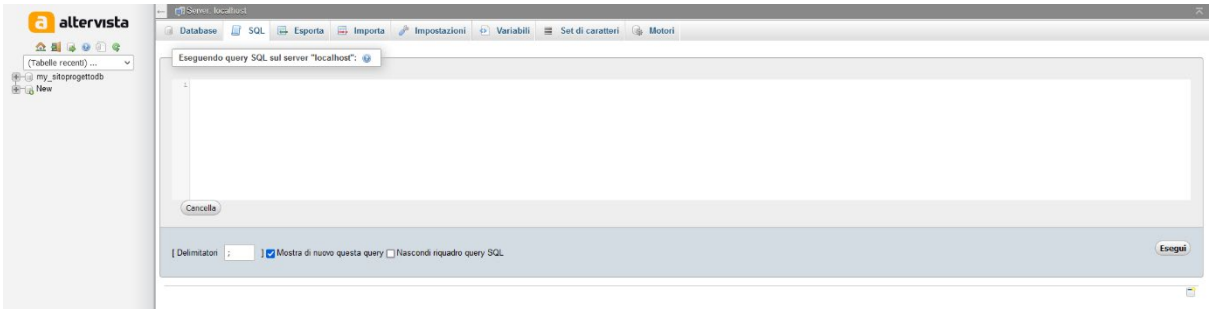
In questo caso verrà utilizzato come linguaggio di interrogazione un linguaggio chiamato SQL (mySQL/MariaDB) attraverso il Phpmyadmin.

Andando su Phpmyadmin accendendo al sito il quale può essere hostato localmente o da un sito di hosting (es. Altervista).

L'ambiente di PhpMyadmin si presenta con questa interfaccia grafica web, la quale si andrà a scrivere le varie interrogazioni in SQL ed andando a cliccare anche nella sezione SQL (indicata con la freccia in arancione) aprirà un'interfaccia in cui si potrà scrivere il codice desiderato.



Dopo aver cliccato SQL, qui apparirà una sezione in cui si potrà sia scrivere il codice e sia eseguirlo (interrogazioni, procedure e trigger, ecc).



7.1 Codice SQL

Si Seleziona il database in cui bisognerà lavorare utilizzando il comando:

```
USE DATABASE  <nome_database>;
```

Si parte dalla creazione della tabella **Cliente:**

```
CREATE Table Cliente
(
    cf varchar(8) NOT NULL ,
    cognome varchar(30) NOT NULL,
    nome varchar(30) NOT NULL,
    città VARCHAR(20),
    gruppo int NOT NULL,
    PRIMARY KEY(cf)
);
```

Per adesso essendo che la tabella “Gruppo” è ancora inesistente si andrà a inserire la chiave esterna dopo aver creato la tabella Gruppo in modo tale che non va a generare degli errori. Commentando quello che è stato scritto nel codice, si può esprimere del codice che descrive la tabella Cliente sono stati scelti che gli attributi CF, cognome, nome e gruppo come **NOT NULL** poiché sono degli attributi che non possono essere nulli.

Creazione tabella **Gruppo:**

```
CREATE TABLE Gruppo
(
    idG INT AUTO_INCREMENT ,
    nome VARCHAR(10),
    totale FLOAT,
    prenotante VARCHAR(8) NOT NULL,
    PRIMARY KEY(idG),
    FOREIGN KEY (prenotante) REFERENCES Cliente(cf)
);
```

Essendo che la tabella Cliente è già stata creata prima, possiamo fin da subito inserire una chiave esterna.

Dopo aver creato la tabella **Gruppo** possiamo andare a modificare la tabella **Cliente** per andare a inserire la chiave esterna utilizzando il comando **ALTER TABLE**:

```
ALTER TABLE Cliente
ADD FOREIGN KEY (gruppo) REFERENCES Gruppo(prenotante);
```

Creazione tabella **Tavolo**

```
CREATE TABLE Tavolo
(
    id INT AUTO_INCREMENT,
    posti INT,
    descrizione VARCHAR(100),
    PRIMARY KEY (id)
);
```

Questa è la creazione della tabella **Tavolo,** i posti non sono mai nulli poiché parte da 8 e vengono decrementati fino ad arrivare a 0 (sarà gestita da un trigger).

La descrizione è opzionale, quindi può rimanere anche vuoto (NULL).

Creazione Tabella **Prenotazione**:

```
CREATE TABLE Prenotazione
(
    tavolo INT,
    prenotante VARCHAR(8),
    data DATE,
    ora TIME,
    PRIMARY KEY (tavolo,prenotante),
    FOREIGN KEY (prenotante) REFERENCES gruppo(prenotante),
    FOREIGN KEY (tavolo) REFERENCES Tavolo(id)
);
```

Creazione tabella **Piatto**:

```
CREATE TABLE Piatto (
    codice int AUTO_INCREMENT,
    nome varchar(50),
    prezzo FLOAT,
    chef int,
    PRIMARY KEY (codice)
)
```

Come descritto in precedenza il codice ha una chiave esterna ma che verrà definita appena l'altra tabella (Chef) verrà creata.

Creazione tabella **Chef**:

```
CREATE TABLE Chef
(
    id INT AUTO_INCREMENT,
    nome varchar(30) NOT NULL,
    cognome varchar(30) NOT NULL,
    specializzazione varchar(40),
    PRIMARY KEY (id)
);
```

Dopo aver creato la tabella **Chef**, dovremo inserire in Piatto la chiave esterna:

```
ALTER TABLE Piatto
ADD FOREIGN KEY (chef) REFERENCES Chef(id);
```

la quale andrà ad inserire la chiave esterna richiesta.

Si effettua anche la creazione della tabella **Ordinazione**:

```
CREATE TABLE Ordinazione
(
    piatto int,
    tavolo int,
    dataP date,
    oraP time,
    PRIMARY KEY (piatto,tavolo,dataP,oraP),
    FOREIGN KEY (piatto) REFERENCES Piatto(id),
    FOREIGN KEY (tavolo) REFERENCES Tavolo(idG),
    FOREIGN KEY (dataP) REFERENCES prenotazione(data),
    FOREIGN KEY (oraP) REFERENCES prenotazione(ora)
)
```

Ci sono 4 campi la quale sono sia chiavi primarie che chiavi esterne .

Il diagramma E-R è stato trasposto in SQL.

Si passa all'inserimento dei dati attraverso il comando:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

(Tutti gli inserimenti effettuati nella base di dati si trovano in **Inserimento.sql**)

Creazione Trigger

Per gestire ad ogni inserimento la modifica dei posti rimanenti si devono gestire con dei Trigger. I trigger sono degli eventi, ovvero delle procedure che vengono eseguite non dall'utente ma ben sì da dei determinati **eventi**, in questo caso se c'è una prenotazione, il tavolo andrà a decrementare il numero di posti disponibili per quel tavolo, così nel caso scenda a 0, il trigger andrà a creare un tavolo inserendo le persone in un altro tavolo.

```
#trigger per gestire la prenotazione
DELIMITER //
CREATE TRIGGER T_Prenotazione
AFTER INSERT ON Prenotazione
FOR EACH ROW
BEGIN
DECLARE X INT ;
DECLARE Y INT;

SELECT count(*) into X
FROM Prenotazione p, Gruppo g, cliente C
WHERE new.tavolo=p.tavolo and p.prenotante=g.prenotante and C.gruppo=g.idG;
IF X > 8
then
    UPDATE Tavolo
    SET posti=0
    WHERE new.tavolo=Tavolo.id;
    SET Y= 8 - X;
    INSERT INTO Tavolo(posti)
    VALUES(8+Y);
ELSE
    UPDATE Tavolo
    SET posti=posti-X
    WHERE new.tavolo=Tavolo.id;
END IF;

END //
DELIMITER ;
```

Il secondo trigger che verrà creato sarà per la gestione di ogni inserimento di ogni ordinazione, questo trigger, garantirà, l'aggiornamento "automatico" dell'aggiornamento totale del conto.

```
#trigger per gestire l'ordinazione : ad ogni ordinazione si calcola il totale

DELIMITER //
CREATE TRIGGER  AggiornaTotale
AFTER INSERT ON Ordinazione
FOR EACH ROW
BEGIN
    DECLARE X FLOAT;
    SELECT  p.prezzo INTO X
    FROM Ordinazione o NATURAL JOIN Prenotazione pr, Piatto p
    WHERE new.piatto=o.piatto and o.piatto=p.codice  ;
    UPDATE Gruppo
    SET totale=totale+X
    WHERE Gruppo.prenotante=Prenotazione.prenotante;
END //
DELIMITER ;
```

Come ultimo trigger è quello dell'inizializzazione del numero di posti ad 8 all'interno della tabella Tavolo poiché nella progettazione è stata specificata:

```
      DELIMITER //

CREATE TRIGGER CreazioneTavolo
AFTER INSERT ON Tavolo
FOR EACH ROW
    UPDATE Tavolo
    SET posti=8
    WHERE new.id=id;
DELIMITER ;
```

Tutti i trigger sono salvati in un apposito file chiamato **trigger.sql .**

Si passa adesso alla realizzazione delle **procedure**, le procedure sono delle **"operazioni"** che sono state proposte in precedenza; quindi, bisognerà creare delle procedura in cui si dovrà chiamare l'operazione richiesta.

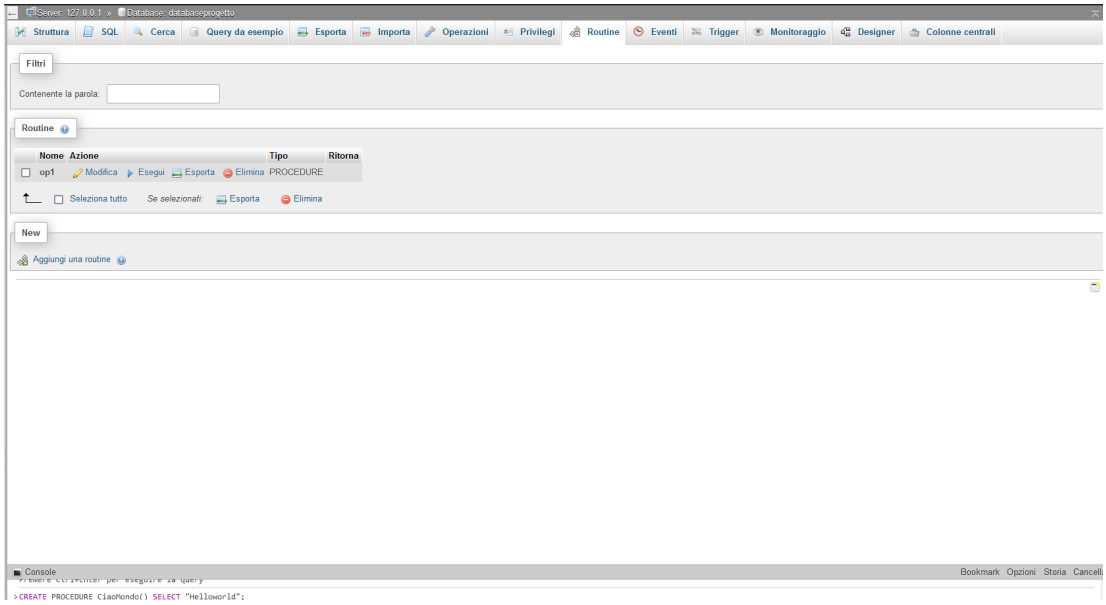
Utilizzando la parola chiave **CREATE PROCEDURE** si andrà a creare delle procedure, la prima operazione può essere creata in questo modo:

OP1 (Tutte le date delle prenotazioni fatte dai clienti di una determinata città)

```
DELIMITER //
CREATE PROCEDURE op1(IN  cit VARCHAR(20))
BEGIN
    SELECT data
    FROM Prenotazione P , Cliente C
    WHERE P.prenotante=C.cf and città=cit ;
END //
DELIMITER ;
```

Note

Per vedere tutte le procedure che sono state create in precedenza bisogna andare in routine e lì ci sono tutte le procedure salvate la quale potranno essere eseguite o cliccando su esegui o utilizzando il comando **CALL <nome_procedura>(<param>) ;**



Op2 (Piatti ordinati da un gruppo):

```
#operazione 2
DELIMITER //
CREATE PROCEDURE op2 (IN gr VARCHAR(8))
BEGIN
    SELECT piatto
    FROM Ordinazione O,Prenotazione P
    WHERE O.tavolo=P.tavolo and prenotante=gr;

END //
DELIMITER ;
```

Op3 (Conteggio dei gruppi che hanno ordinato un certo tipo di piatto):

```
#operazione 3
DELIMITER //
CREATE PROCEDURE op3 (IN piat INT)
BEGIN
   SELECT count(*)
   FROM Ordinazione O, Prenotazione P, Gruppo G
   WHERE Piatto=piat and O.tavolo=P.tavolo and G.prenotante=P.prenotante
   GROUP BY idG;

END //
DELIMITER ;
```

Op4 (I nomi dei piatti preparati da uno chef):

```
#operazione 4
DELIMITER //
CREATE PROCEDURE op4(IN cuoco INT)
BEGIN
    SELECT p.nome
    FROM Chef c,Piatto p
    WHERE c.chef=id and c.chef=cuoco;
END //

DELIMITER ;
```

Op5 (Stampa per ogni tavolo i piatti ordinati):

```
CREATE PROCEDURE op5()
BEGIN
    SELECT O.tavolo,P.codice,P.nome
    FROM Ordinazione O, Piatto P
    WHERE  O.piatto=P.codice
    GROUP BY O.tavolo;
END//
DELIMITER ;
```

Note

Essendo una batch non avrà bisogno di alcun parametro passato.

Op6 (Per ogni prenotante stampa il nome e cognome):

```
DELIMITER //
CREATE PROCEDURE op6()
BEGIN
    SELECT prenotante , nome, cognome
    FROM Gruppo G, Cliente C
    WHERE prenotante=cf
    GROUP BY prenotante;
END//
DELIMITER ;
```

Op7 (Stampa tutti i gruppi che NON hanno alcun ordinato un tipo di piatto):

```
#operazione 7
DELIMITER //
CREATE PROCEDURE op7()
BEGIN
    SELECT G.idG
    FROM Ordinazione O, Gruppo G, Prenotazione P
    WHERE O.tavolo=P.prenotante and NOT EXISTS (
        SELECT *
        FROM Piatto
        WHERE O.piatto=codice
    );


END //
DELIMITER;
```

Op8 (Inserimento di un nuovo chef):

```
# Operazione 8
DELIMITER //
CREATE PROCEDURE Op8(IN cuoco INT, IN nome varchar (20) , IN cognome varchar(20) ,
IN specializzazione INT )
BEGIN
        INSERT INTO Chef VALUES (cuoco,nome,cognome,specializzazione);
END //
DELIMITER ;
```

Op9 (Per ogni gruppo stampa il totale da pagare):

```
#op9
DELIMITER //
CREATE PROCEDURE Op9()
BEGIN
    SELECT idG,totale
    FROM gruppo
    GROUP BY idG;
END //
DELIMITER;
```

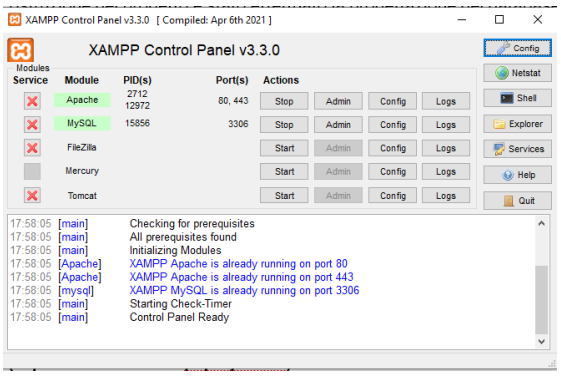
Note

Grazie al trigger che gestisce il totale e quindi mantenendo tale ridondanza ha concesso di poter semplificare anche questa query. Questo fa in modo che **l'analisi della ridondanza** possa effettivamente migliorare l'efficienza in tutto e per tutto, riducendo anche le problematiche legate ed essi se essa è fatta correttamente.

8. Conclusione & File

Durante la costruzione del progetto è stato effettuato la progettazione del database:

Inizialmente il database era stato pensato per girare su Altervista (un noto sito di host) che consentiva l'host del database in modo gratuito utilizzando PhpMyAdmin con il Mysql versione 8.0.26, ma durante l'implementazione della base di dati sono stati riscontrati dei problemi, un problema noto era l'impossibilità di gestire le procedure poiché l'Host non consente l'abilitazione di essi. La base di dati è stata quindi esportata per poi essere eseguita localmente su MariaDB ver. 10.4.22 (un fork di mysql) utilizzando un software chiamato XAMPP (una piattaforma costituita da diversi software per la gestione di siti internet e Database).



FILE

Ci sono diversi File con i codici in linguaggio SQL:

tabelle.sql (strutture delle tabelle)
procedure.sql (codici procedure)
inserimento_dati.sql (dati d'esempio)
trigger.sql (trigger)
Export.sql (esportazione fatta da phpmyadmin)