Documentazione EnoMarketDB

Progetto EnoMarket
Autori: La Rovere Andrea, Noce Davide, Zheng Simone
Corso di laurea: Informatica
Data: 11/07/2024

Indice

1. Generalità

1.1 Descrizione generale del prodotto

Il progetto mira alla creazione di un database per la gestione di un sito e-commerce dedicato alla vendita di vini. Questo database sarà il nucleo centrale del sistema informativo, supportando le operazioni quotidiane dell'e-commerce, dall'ordine da parte degli utenti alla spedizione del prodotto stesso. Il database si occuperà della gestione degli ordini, delle specifiche dei vini, della consegna dei prodotti e della raccolta delle recensioni da parte dei clienti riguardo la consegna e la qualità dei prodotti.

1.2 Obiettivi del Progetto:

1. **Organizzazione Efficiente dell'Inventario**:
    - Archiviazione dettagliata delle informazioni sui prodotti, inclusi nomi, descrizioni, varietà di uve, annate, regioni di produzione, cantine, e immagini.
2. **Gestione degli Utenti**:
    - Registrazione degli utenti con dati personali, credenziali di accesso e preferenze.
    - Registrazione dei metodi di pagamento dei clienti
    - Storico degli acquisti e tracciamento degli ordini.
3. **Processamento degli Ordini**:
    - Sistema per la creazione, la modifica e la cancellazione degli ordini.
    - Tracciabilità degli ordini dal momento della creazione fino alla consegna.
    - Gestione dei pagamenti e delle transazioni finanziarie.
4. **Integrazione con Sistemi Esterni**:
    - Integrazione con corrieri per la gestione delle spedizioni.
5. **Analisi e Reportistica**:
    - Generazione di recensioni sulle vendite e sulle performance dei prodotti.

- Analisi dei comportamenti degli utenti per migliorare l'esperienza di acquisto e le strategie di marketing.

1.3 Utenti

Gli utenti principali del DB sono i seguenti:

- *Clienti*;
- *Produttori*;
- *Corrieri*.

1.3.1 Clienti

Profilo: I clienti sono gli utenti finali che visitano il sito per acquistare vini. Possono essere appassionati di vini, collezionisti, ristoratori o semplici consumatori occasionali.

Obiettivi/Bisogni:

- **Ricerca e Scoperta**: Accesso facile e intuitivo per esplorare i prodotti disponibili, con filtri per regione, varietà di uva, annata, prezzo, ecc.
- **Acquisto Semplice e Sicuro**: Processo di acquisto fluido, supporto per vari metodi di pagamento, e sicurezza nelle transazioni.
- **Informazioni Dettagliate**: Descrizioni complete dei prodotti, recensioni, e consigli sull'abbinamento cibo-vino.
- **Gestione degli Ordini**: Visualizzazione dello stato degli ordini, storico degli acquisti, e aggiornamenti sulle spedizioni.
- **Personalizzazione**: Raccomandazioni personalizzate basate sui precedenti acquisti e preferenze.

1.3.2 Produttori

Profilo: I produttori gestiscono l'inventario fisico, preparando gli ordini per la spedizione e monitorando le scorte.

Obiettivi/Bisogni:

- **Gestione delle Scorte**: Monitoraggio accurato delle quantità in magazzino e notifica automatica per il riordino dei prodotti.
- **Processo degli Ordini**: Accesso agli ordini in tempo reale per preparare le spedizioni e aggiornare lo stato degli ordini.
- **Logistica e Spedizioni**: Coordinamento con i corrieri e generazione di etichette di spedizione.

1.3.3 Corrieri

Profilo: I corrieri sono i partner logistici responsabili della consegna dei prodotti acquistati dai clienti sul sito e-commerce. Possono essere aziende di spedizioni esterne contrattate per gestire le consegne.

Obiettivi/Bisogni:

- **Accesso agli Ordini di Spedizione**: Visualizzazione e gestione degli ordini pronti per la spedizione, con dettagli sui pacchi da consegnare.
- **Aggiornamento dello Stato di Spedizione**: Capacità di aggiornare in tempo reale lo stato della spedizione, inclusi i passaggi intermedi (in transito, consegnato, ritardo).
- **Generazione di Etichette di Spedizione**: Strumenti per generare e stampare etichette di spedizione con tutte le informazioni necessarie (indirizzo di destinazione, numero di tracciamento, codice a barre).
- **Tracciabilità delle Spedizioni**: Sistema di tracciamento per monitorare il percorso delle spedizioni e fornire aggiornamenti ai clienti e agli operatori del magazzino.

2. Raccolta e analisi dei Requisiti

2.1 Glossario dei termini

Entità	Descrizione	Sinonimi
Clienti	Comprano il prodotto	Compratori
Vini	Prodotti in vendita	Prodotti
Carte di Credito	Metodo di pagamento	Carte

Entità	Descrizione	Sinonimi
Spedizioni	Consegna dei prodotti	Invio
Ordini	Acquisto dei prodotti	Richieste
Recensioni	Feedback dell'utente	Feedback
Indirizzi	Indirizzi fisici	Recapiti
Corrieri	Gestore consegne	Trasportatori
Magazzini	Depositi dei vini	Riserve
Tipi di Vini	Varianti del vino	Tipologie
Uve	Ingrediente per i vini	Materie Prime
Produttori	Produttori dei vini	Realizzatori
Premi	Nome premio vinto	Riconoscimenti
Lista Prodotti	Prodotti dentro un certo ordine	Elenco

2.2 Specifiche, assunzioni e vincoli d'integrità

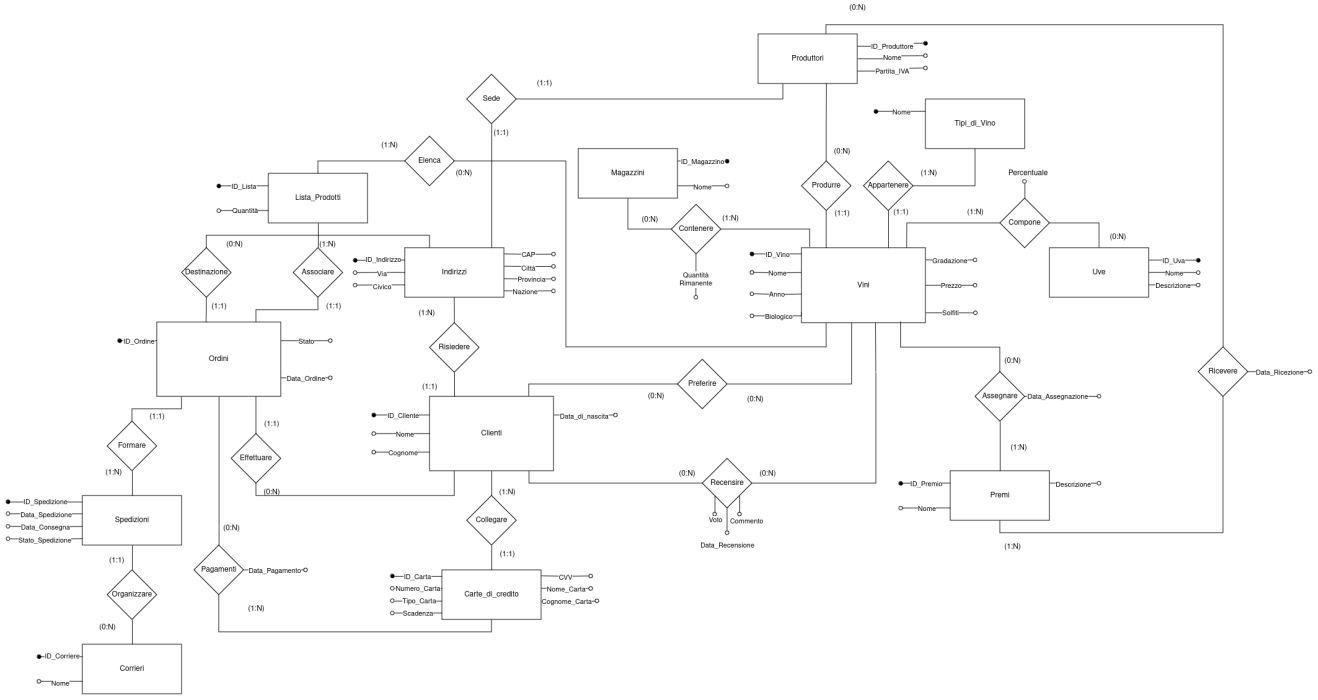
2.2.1 Vincoli

- **Sicurezza**: Il cliente viene inserito solo se maggiorenne.
- **Coerenza**: I voti delle recensioni vanno da 0 a 10.
- **Correttezza**: Un premio non può essere assegnato prima che il vino venga prodotto.
- **Quantità dei Prodotti**: La quantità di vini in `Lista_Prodotti` non può essere minore di 1.
- **Data di Scadenza della Carta di Credito**: La data di scadenza della carta di credito deve essere successiva alla data corrente.
- **Email Unica**: Ogni cliente deve avere un'email unica, senza duplicati (se esiste un campo email per i clienti).
- **Validità della Data di Nascita**: La data di nascita del cliente non può essere nel futuro.
- **Data di Spedizione e Consegna**: La data di consegna non può essere precedente alla data di spedizione.
- **Produttore Esistente**: Un vino può essere inserito solo se il produttore specificato esiste.

- **Quantità Rimasta in Magazzino**: La quantità rimanente di un vino in un magazzino non può essere negativa.
- **Percentuale di Composizione**: La somma delle percentuali delle uve per ogni vino deve essere pari a 100%.
- **Pagamento Unico per Ordine**: Ogni ordine spedito deve avere almeno un pagamento associato.
- **Ordini Validi**: Ogni ordine deve essere associato a un cliente valido e un indirizzo esistente.
- **Stato dell'Ordine**: Lo stato dell'ordine può essere solo "In Preparazione", "Spedito" o "Consegnato".

3. Progettazione concettuale

3.1 Diagramma E-R



3.2 Schema scheletro

Le entità principali del sistema sono le seguenti:

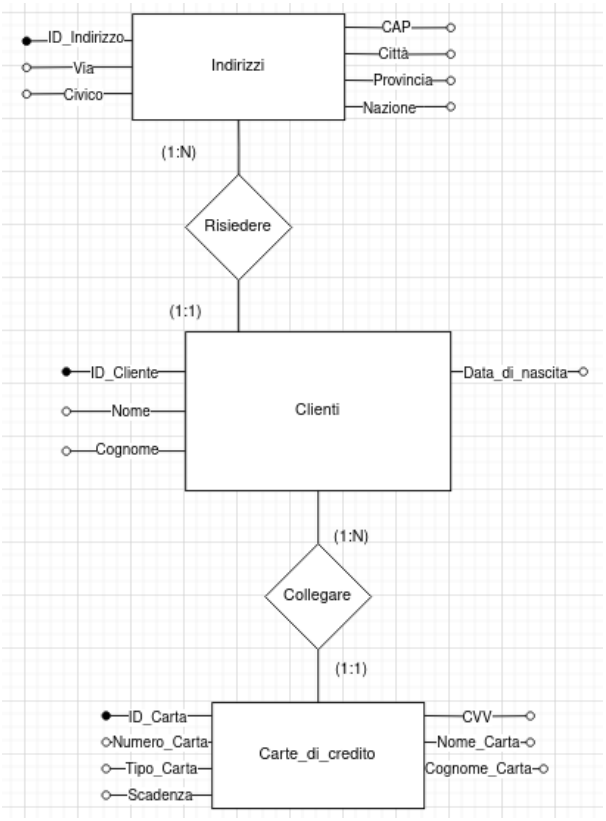
- Cliente

- Vino
- Corriere

  Le relazioni presenti permettono di affermare che un *Cliente* può comprare un *Vino* che viene spedito dal *Corriere*.

3.3 Raffinamenti

1. Raffinazione *Clienti*:

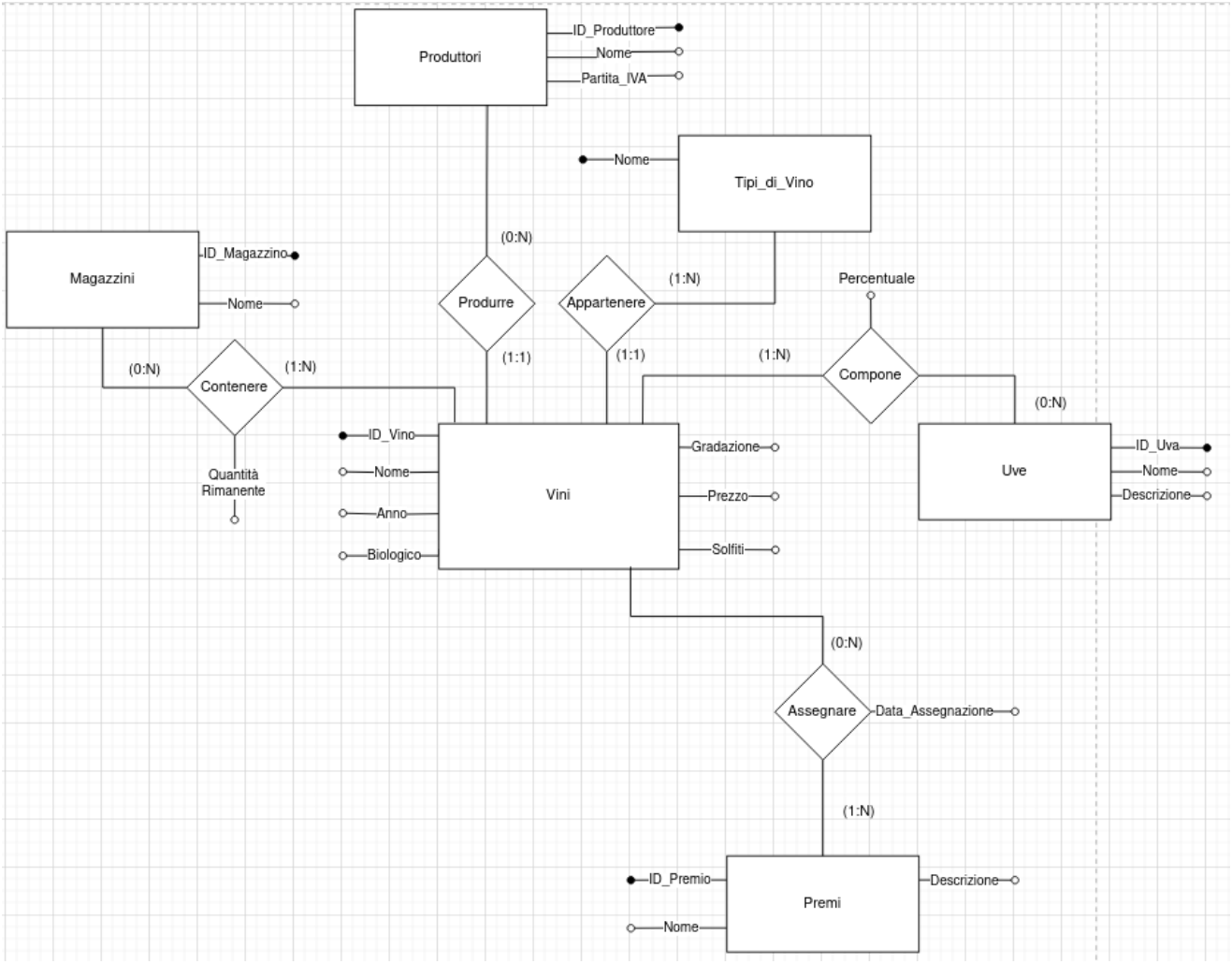


Qui si raffina l'entità *Cliente*:

- Ad ogni *Cliente* è collegata una o più *Carte di Credito* che gli permetterà di acquistare i prodotti;
- Il *Cliente*, al termine dell'acquisto e della consegna, può lasciare una *Recensione* riguardo il servizio di consegna e riguardo la qualità del prodotto acquistato.

---

2. Raffinazione *Vini*:

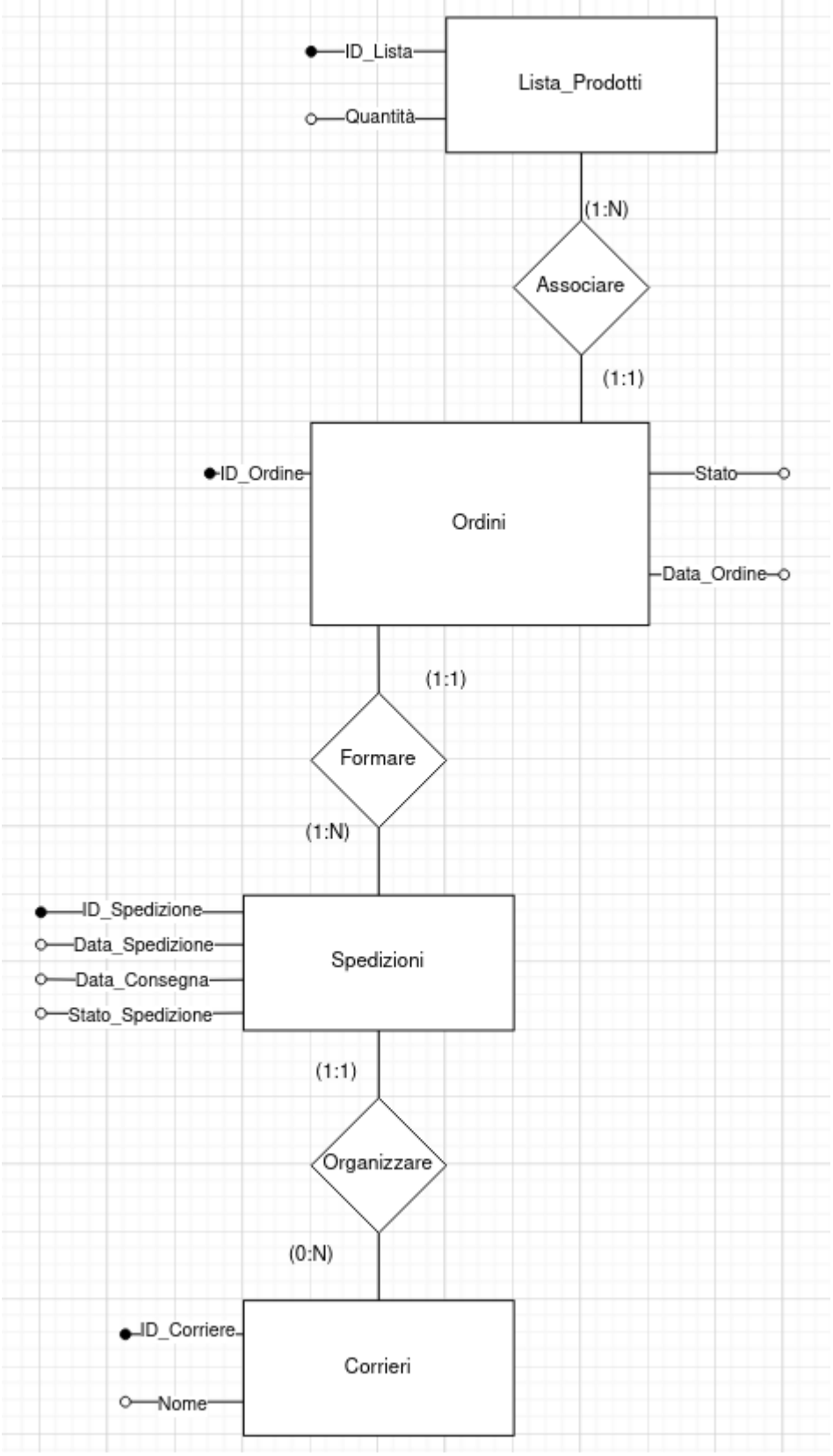


Qui si raffina l'entità *Vini*:

- I *Vini* saranno composti da *Uve* e creeranno diversi *Tipi di vino*;
- Essi verranno prodotti dai *Produttori* e conservate in dei *Magazzini* in attesa di un ordine e di una seguente spedizione.
- Ai vini migliori verranno assegnati dei *Premi* in base alle loro *Recensioni* ed al loro numero di acquisti

---

3. Raffinazione *Ordini*:

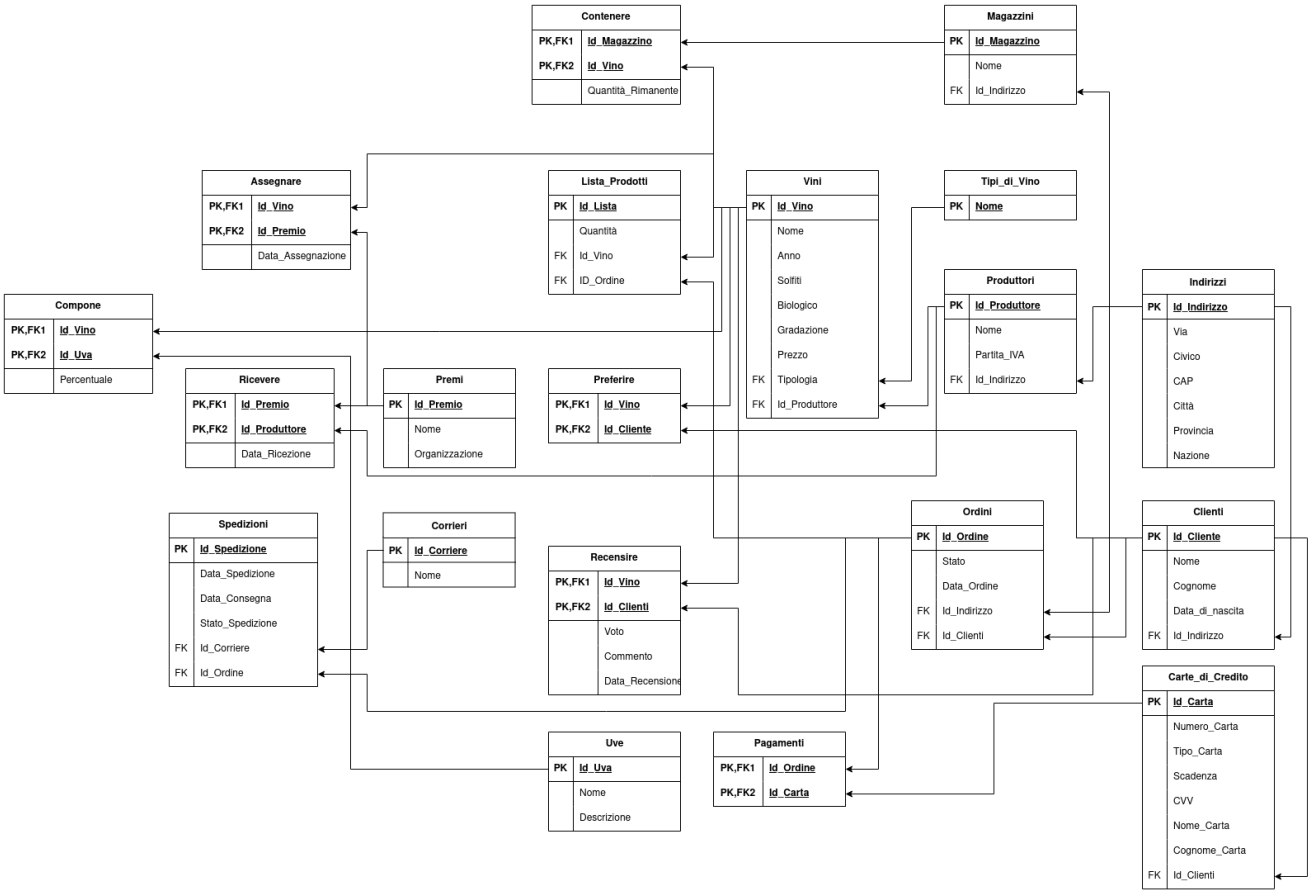


Qui si raffina l'entità *Ordini*:

- Gli ordini sono associati a delle *liste dei prodotti*;
- Successivamente gli ordini andranno a formare delle *Spedizioni* che verranno organizzate e consegnate dai *Corrieri*

4. Progettazione Logica

4.1 Schema Logico



Le chiavi primarie sono identificate in **grassetto**, mentre le chiavi secondarie in *corsivo*:

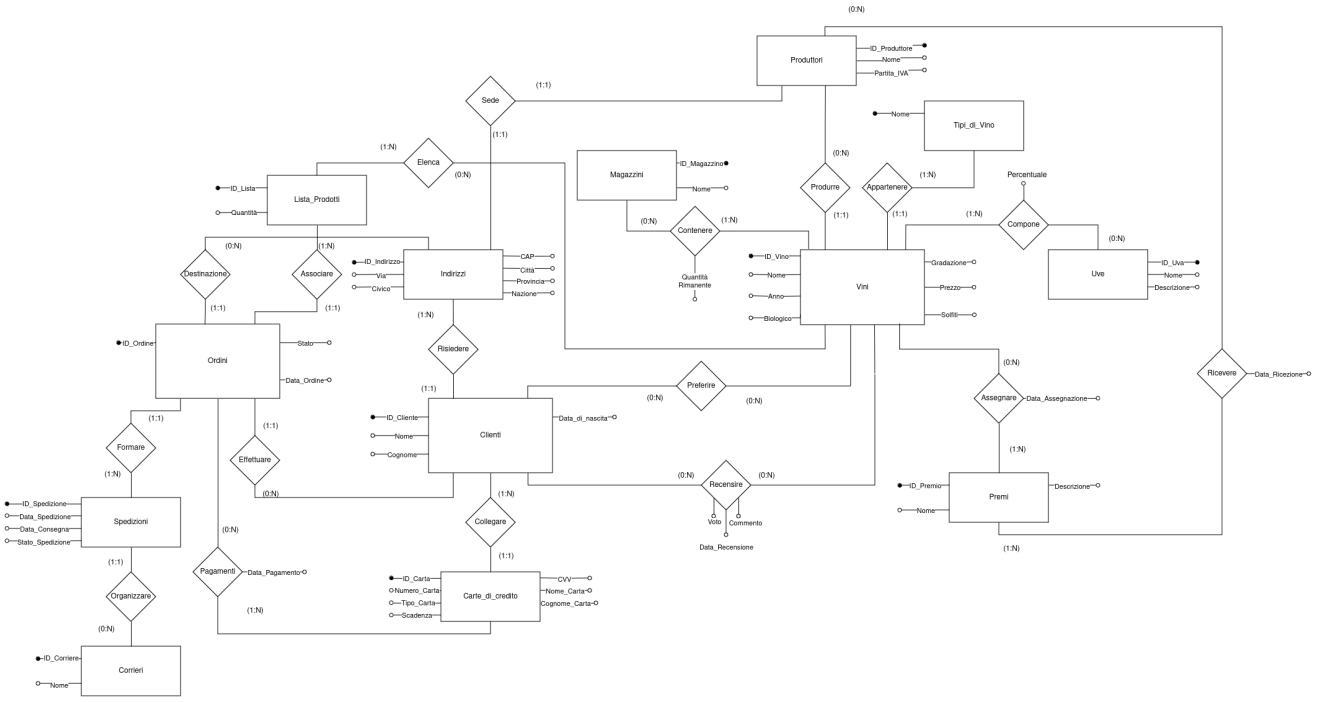
- **Magazzini** (**Id_Magazzino**, Nome, *Id_Indirizzo*)
- **Contenere** (**Id_Magazzino**, **Id_Vino**, Quantità_Rimanente)
- **Indirizzi** (**Id_Indirizzo**, Via, Civico, CAP, Città, Provincia, Nazione)
- **Produttori** (**Id_Produttore**, Nome, Partita_IVA, *Id_Indirizzo*)
- **Vini** (**Id_Vino**, Nome, Anno, Soffitti, Biologico, Gradazione, Prezzo, *Tipologia*, *Id_Produttore*)
- **Tipi_di_Vino** (**Nome**)
- **Preferire** (**Id_Vino**, **Id_Cliente**)
- **Assegnare** (**Id_Vino**, **Id_Premio**, Data_Assegnazione)
- **Premi** (**Id_Premio**, Nome, Organizzazione)
- **Ricevere** (**Id_Premio**, **Id_Produttore**, Data_Ricezione)
- **Recensire** (**Id_Vino**, **Id_Cliente**, Voto, Commento, Data_Recensione)

- **Clienti** (**Id_Cliente**, Nome, Cognome, Data_di_nascita, *Id_Indirizzo*)
- **Ordini** (**Id_Ordine**, Stato, Data_Ordine, *Id_Indirizzo*, *Id_Clienti*)
- **Pagamenti** (**Id_Ordine**, **Id_Carta**)
- **Carte_di_Credito** (**Id_Carta**, Numero_Carta, Tipo_Carta, Scadenza, CVV, Nome_Carta, Cognome_Carta, *Id_Clienti*)
- **Spedizioni** (**Id_Spedizione**, Data_Spedizione, Data_Consegna, Stato_Spedizione, *Id_Corriere*, *Id_Ordine*)
- **Corrieri** (**Id_Corriere**, Nome)
- **Uve** (**Id_Uva**, Nome, Descrizione)
- **Compone** (**Id_Vino**, **Id_Uva**, Percentuale)
- **Lista_Prodotti** (**Id_Lista**, Quantità, *Id_Vino*, *ID_Ordine*)

4.2 Normalizzazione

1. Le tuple del diagramma possiedono solo attributi semplici $\implies$ 1NF è soddisfatta.
2. Gli attributi non primi delle nostre entità dipendono unicamente dalla chiave primaria completa $\implies$ 2NF è soddisfatta.
3. Gli attributi non primi delle nostre entità dipendono in maniera non transitiva dalla chiave primaria completa $\implies$ 3NF è soddisfatta.
4. Le entità sono state decomposte in maniera da ottenere il massimo numero possibile di BCNF.

4.3 Schema E-R concettuale ristrutturato



4.4 Dizionario Entità e Relazioni

4.4.1 Glossario delle Entità

Entità	Descrizione	Attributi	Relazioni coinvolte
Clienti	Utente che intende acquistare il prodotto	*ID_Cliente*, Nome, Cognome, Data_Nascita	Risiedere, Effettuare, Collegare, Scrivere, Preferire
Vini	Prodotto in vendita	*ID_Vino*, Anno, Nome, Biologico, Gradazione, Prezzo, Solfiti	Contenere, Produrre, Valutare, Elenca, Appartenere, Assegnare, Preferire, Compone
Carte di credito	Metodo di pagamento	*ID_Carta*, CVV, Scadenza, Nome_Carta, Cognome_Carta, Tipo_Carta, Numero_Carta	Collegare, Pagamenti

Entità	Descrizione	Attributi	Relazioni coinvolte
Spedizioni	Metodo di consegna del prodotto	*ID_Spedizione*, Data_Spedizione, Data_Consegna, Stato_Spedizione	Organizzare, Formare
Ordini	Acquisto effettivo del prodotto	*ID_Ordine*, Data_Ordine, Stato	Formare, Effettuare, Pagamenti, Associare, Descrizione
Recensioni	Feedback dell'utente	*ID_Recensione*, Data_Recensione, Voto, Commento	Scrivere, Valutare
Produttori	Colui che produce il vino	*ID_Produttore*, Nome, Partita_IVA	Produrre, Sede, Ricevere
Corrieri	Azienda esterna che gestisce le consegne	*ID_Corriere*, Nome	Organizzare
Magazzini	Luogo in cui viene gestita la conservazione dei vini	*ID_Magazzino*, Nome	Contenere
Tipi di Vino	Informazioni aggiuntive del vino in base al tipo	*Nome*	Appartenere
Lista Prodotti	Informazione dei prodotti	*ID_Lista*, Quantità	Associare, Elenca
Uve	Ingredienti per la produzione dei vini	*ID_Uva*, Nome, Descrizione	Comporre
Premi	Premio assegnati	*ID_Premio*, Nome, Descrizione	Assegnare, Ricevere
Indirizzi	Indirizzo della sede e del cliente	*ID_Indirizzo*, Via, Civico, CAP, Città, Provincia, Nazione	Risiedere, Sede, Destinazione

4.4.2 Glossario delle Relazioni

Relazione	Descrizione	Entità
Associare	L'associazione di una lista dei prodotti ad un ordine	Ordini (1:1), Lista_Prodotti(1:N),
Effettuare	Effettuare l'ordine da parte dei clienti	Ordini (1:1), Clienti(0:N)
Produrre	Indica chi ha prodotto il vino	Vini (1:1), Produttori (0:N)
Compone	Indica cosa c'è all'interno del vino	Vini (1:N), Uve (0:N)
Scrivere	Commento lasciato dal cliente	Clienti (0:N), Recensione (1:1)
Valutare	Modo per valutare la qualitò del vino	Recensione (1:1), Vini (0:N)
Contenere	Luogo dove viene conservato il vino	Magazzini (0:N), Vini (1:N)
Formare	Preparare l'ordine da spedire	Spedizioni(1:N), Ordini(1:1)
Preferire	Indica la preferenza dei clienti sui vini	Clienti(0:N), Vini(0:N)
Assegnare	Assegnare il premio vinto al vino	Vini(0:N), Premi(1:N)
Ricevere	Ricevere il premio al produttore	Premio(1:N), Produttori(0:N)
Sede	Luogo dove si trova la sede	Produttori(1:1), Indirizzi(1:1)
Elenca	Elenca i vini presenti nella lista	Lista_Prodotti(1:N), Vini(0:N)
Destinazione	Luogo di consegna dell'ordine	Ordini(1:1), Indirizzi(0:N)
Organizzare	Corriere organizza la spedizione	Corrieri(0:N), Spedizioni(1:1)
Pagamenti	Pagamento degli ordini tramite carta	Carta di Credito(1:N), Ordini(0:N)

Relazione	Descrizione	Entità
Collegare	Carta collegata ad un cliente	Carta di Credito(1:1), Clienti(1:N)
Appartenere	Un vino appartiene ad un tipo	Vini(1:1), Tipi di Vino(1:N)
Risiedere	Luogo in cui un cliente abita	Clienti(1:1), Indirizzi(1:N)

4.5 Carico Applicativo

4.5.1 Volume dei dati

Concetto	Tipologia	Occorrenze	Dimensione Record	Volume Totale in Bytes
Corrieri	Entità	15	(4+101)=105	225
Indirizzi	Entità	1000	(4+64+4+4+51+51+51)=229	229000
Clienti	Entità	500	(4+51+51+3+4)=113	56500
Carte di Credito	Entità	550	(4+21+51+3+5+51+4)=139	76450
Produttori	Entità	300	(4+51+8+4)=67	20100
Tipi di Vino	Entità	10	(51)=51	510
Vini	Entità	3000	(4+51+4+1+1+51+4+4)=120	360000
Ordini	Entità	10000	(4+4+51+4+8)=71	710000
Spedizioni	Entità	1000	(4+4+4+8+8+51)=79	79000
Lista Prodotti	Entità	30000	(4+4+4+4)=16	480000
Pagamenti	Relazione	9000	(4+4+8)=16	144000
Preferire	Relazione	100	(4+4)=8	800
Recensire	Relazione	2000	(4+4+4+64+8)=84	168000
Premi	Entità	50	(4+51+51)=106	5300
Assegnare	Relazione	200	(4+4+3)=11	2200
Ricevere	Relazione	200	(4+4+3)=11	2200

Concetto	Tipologia	Occorrenze	Dimensione Record	Volume Totale in Bytes
Uve	Entità	300	(4+51+64)=119	35700
Compone	Relazione	3500	(4+4+4)=12	42000
Magazzini	Entità	5	(4+4+51)=59	295
Contenere	Relazione	30000	(4+4+4)=12	360000

4.5.2 Script SQL per il calcolo del Carico Applicativo

```
SELECT
    table_schema,
    table_name,
    ROUND(((data_length + index_length) / 1024 / 1024), 2)
AS "Size (MB)"
FROM
    information_schema.TABLES
WHERE
    table_schema = "EnoMarketDB"
ORDER BY
    (data_length + index_length) DESC;
```

Il seguente codice restituisce una tabella con le varie entità e relazioni con lo spazio che occupano sulla memoria secondaria.

5. Implementazione Database - MySQL

5.1 Creazione delle Tabelle

```
CREATE TABLE Corrieri (
Id_Corriere INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(100) NOT NULL,
PRIMARY KEY (Id_Corriere)
```

```
);

CREATE TABLE Indirizzi (
Id_Indirizzo INT NOT NULL AUTO_INCREMENT,
Via TEXT NOT NULL,
Civico INT NOT NULL,
CAP INT NOT NULL,
Città VARCHAR(50) NOT NULL,
Provincia VARCHAR(50) NOT NULL,
Nazione VARCHAR(50) NOT NULL,
PRIMARY KEY (Id_Indirizzo)
);

CREATE TABLE Clienti (
Id_Cliente INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50) NOT NULL,
Cognome VARCHAR(50) NOT NULL,
Data_di_nascita DATE NOT NULL,
Id_Indirizzo INT,
PRIMARY KEY (Id_Cliente),
FOREIGN KEY (Id_Indirizzo) REFERENCES
Indirizzi(Id_Indirizzo)
);

CREATE TABLE Carte_di_Credito (
Id_Carta INT NOT NULL AUTO_INCREMENT,
Numero_Carta VARCHAR(20) NOT NULL,
Tipo_Carta VARCHAR(50) NOT NULL,
Scadenza DATE NOT NULL,
CVV VARCHAR(4) NOT NULL,
Nome_Carta VARCHAR(50) NOT NULL,
Cognome_Carta VARCHAR(50) NOT NULL,
Id_Cliente INT NOT NULL,
PRIMARY KEY (Id_Carta),
FOREIGN KEY (Id_Cliente) REFERENCES Clienti(Id_Cliente)
);

CREATE TABLE Produttori (
Id_Produttore INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50) NOT NULL,
```

```
Partita_IVA BIGINT NOT NULL,
Id_Indirizzo INT,
PRIMARY KEY (Id_Produttore),
FOREIGN KEY (Id_Indirizzo) REFERENCES
Indirizzi(Id_Indirizzo)
);

CREATE TABLE Tipi_di_Vino (
Nome VARCHAR(50) NOT NULL,
PRIMARY KEY (Nome)
);

CREATE TABLE Vini (
Id_Vino INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50) NOT NULL,
Anno INT NOT NULL,
Solfiti BOOLEAN,
Biologico BOOLEAN,
Tipologia VARCHAR(50),
Gradazione DECIMAL(4,2) NOT NULL,
Id_Produttore INT,
Prezzo DECIMAL(10,2) NOT NULL,
PRIMARY KEY (Id_Vino),
FOREIGN KEY (Id_Produttore) REFERENCES
Produttori(Id_Produttore),
FOREIGN KEY (Tipologia) REFERENCES Tipi_di_Vino(Nome)
);

CREATE TABLE Ordini (
Id_Ordine INT NOT NULL AUTO_INCREMENT,
Id_Cliente INT,
Stato VARCHAR(50),
Id_Indirizzo INT NOT NULL,
Data_Ordine DATETIME NOT NULL,
PRIMARY KEY (Id_Ordine),
FOREIGN KEY (Id_Cliente) REFERENCES Clienti(Id_Cliente),
FOREIGN KEY (Id_Indirizzo) REFERENCES
Indirizzi(Id_Indirizzo)
);
```

```
CREATE TABLE Lista_Prodotti (
Id_Lista INT NOT NULL AUTO_INCREMENT,
Id_Vino INT,
Id_Ordine INT,
Quantità INT NOT NULL,
PRIMARY KEY (Id_Lista),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino),
FOREIGN KEY (Id_Ordine) REFERENCES Ordini(Id_Ordine)
);

CREATE TABLE Pagamenti (
Id_Ordine INT,
Id_Carta INT,
Data_Pagamento DATETIME NOT NULL,
PRIMARY KEY (Id_Ordine, Id_Carta),
FOREIGN KEY (Id_Carta) REFERENCES
Carte_di_Credito(Id_Carta),
FOREIGN KEY (Id_Ordine) REFERENCES Ordini(Id_Ordine)
);

CREATE TABLE Preferire (
Id_Cliente INT,
Id_Vino INT,
PRIMARY KEY (Id_Cliente, Id_Vino),
FOREIGN KEY (Id_Cliente) REFERENCES Clienti(Id_Cliente),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino)
);

CREATE TABLE Recensire (
Id_Cliente INT,
Id_Vino INT,
Voto INT NOT NULL,
Commento TEXT,
Data_recensione DATETIME NOT NULL,
PRIMARY KEY (Id_Cliente, Id_Vino),
FOREIGN KEY (Id_Cliente) REFERENCES Clienti(Id_Cliente),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino)
);

CREATE TABLE Premi (
```

```
Id_Premio INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50),
Organizzazione VARCHAR(50),
PRIMARY KEY (Id_Premio)
);

CREATE TABLE Assegnare (
Id_Premio INT,
Id_Vino INT,
Data_Assegnazione DATE NOT NULL,
PRIMARY KEY (Id_Premio, Id_Vino),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino),
FOREIGN KEY (Id_Premio) REFERENCES Premi(Id_Premio)
);

CREATE TABLE Ricevere (
Id_Premio INT,
Id_Produttore INT,
Data_Ricezione DATE NOT NULL,
PRIMARY KEY (Id_Premio, Id_Produttore),
FOREIGN KEY (Id_Produttore) REFERENCES
Produttori(Id_Produttore),
FOREIGN KEY (Id_Premio) REFERENCES Premi(Id_Premio)
);

CREATE TABLE Uve (
Id_Uva INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50),
Descrizione TEXT,
PRIMARY KEY (Id_Uva)
);

CREATE TABLE Compone (
Id_Vino INT,
Id_Uva INT,
Percentuale INT NOT NULL,
PRIMARY KEY (Id_Vino, Id_Uva),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino),
FOREIGN KEY (Id_Uva) REFERENCES Uve(Id_Uva)
);
```

```
CREATE TABLE Magazzini (
Id_Magazzino INT NOT NULL AUTO_INCREMENT,
Id_Indirizzo INT,
Nome VARCHAR(50),
PRIMARY KEY (Id_Magazzino),
FOREIGN KEY (Id_Indirizzo) REFERENCES
Indirizzi(Id_Indirizzo)
);

CREATE TABLE Contenere (
Id_Magazzino INT NOT NULL,
Id_Vino INT NOT NULL,
Quantità_Rimanente INT NOT NULL,
PRIMARY KEY (Id_Magazzino, Id_Vino),
FOREIGN KEY (Id_Magazzino) REFERENCES
Magazzini(Id_Magazzino),
FOREIGN KEY (Id_Vino) REFERENCES Vini(Id_Vino)
);

CREATE TABLE Spedizioni (
Id_Spedizione INT NOT NULL AUTO_INCREMENT,
Id_Ordine INT NOT NULL,
Id_Corriere INT NOT NULL,
Data_Spedizione DATETIME NOT NULL,
Data_Consegna DATETIME,
Stato_Spedizione VARCHAR(50) NOT NULL,
PRIMARY KEY (Id_Spedizione),
FOREIGN KEY (Id_Ordine) REFERENCES Ordini(Id_Ordine),
FOREIGN KEY (Id_Corriere) REFERENCES Corrieri(Id_Corriere)
);
```

5.2 Triggers

I trigger fanno parte del DDL (Data Definition Manipulation), essi seguono il principio ECA, ovvero Event-Condition-Action. Solitamente, un trigger si può attivare prima o dopo un inserimento e hanno 2 livelli di granularità:

1. Attivarsi per ogni tupla

2. Ativarsi per ogni istruzione DML
   Nello specifico in MySQL i trigger operano a livello di riga e si ammette un solo trigger per tabella. Osserviamo inoltre che questi vengono usati per mantenere constraint di ogni tipo, in primis il vincolo di integrità referenziale. Quelli di seguito sono una serie di trigger di esempio necessari per mantenere una serie di vincoli nel nostro database.

```
-- Permette l'inserimento di un cliente solo se è
maggiorenne.

CREATE TRIGGER VerificaMaggiorenne
BEFORE INSERT ON Clienti
FOR EACH ROW
BEGIN
DECLARE msg VARCHAR(255);
IF TIMESTAMPDIFF(YEAR, NEW.Data_di_nascita, CURDATE()) < 18
THEN
SET msg = CONCAT('Inserimento negato, il cliente ',
NEW.Nome, ' ', NEW.Cognome, ' non è maggiorenne.');
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
END IF;
END //

-- Riporta in range i voti minori di 0 o maggiori di 10

CREATE TRIGGER VerificaVoto
BEFORE INSERT ON Recensire
FOR EACH ROW
BEGIN
IF NEW.Voto < 0 THEN
SET NEW.Voto = 0;
ELSEIF NEW.Voto > 10 THEN
SET NEW.Voto = 10;
END IF;
END //

-- Evita che un premio possa essere assegnato prima che il
```

```
vino sia prodotto

CREATE TRIGGER VerificaDataPremio
BEFORE INSERT ON Assegnare
FOR EACH ROW
BEGIN
DECLARE AnnoVino INT;
SELECT Anno INTO AnnoVino
FROM Vini
WHERE Id_Vino = NEW.Id_Vino;
IF YEAR(NEW.Data_Assegnazione) < AnnoVino THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "La data di assegnazione del premio non
può precedere la data di produzione del vino.";
END IF;
END //
```

5.3 Inserimenti

Di seguito vengono riportati gli inserimenti per popolare il DB

5.3.1 Indirizzi

```
INSERT INTO Indirizzi (Via, Civico, CAP, Città, Provincia,
Nazione) VALUES

('Via Roma', 1, 10100, 'Torino', 'TO', 'Italia'),

('Via Milano', 2, 20100, 'Milano', 'MI', 'Italia'),

('Via Napoli', 3, 80100, 'Napoli', 'NA', 'Italia'),

('Via Firenze', 4, 50100, 'Firenze', 'FI', 'Italia'),

('Via Venezia', 5, 30100, 'Venezia', 'VE', 'Italia'),
```

```
('Via Bari', 6, 70100, 'Bari', 'BA', 'Italia'),

('Via Bologna', 7, 40100, 'Bologna', 'BO', 'Italia'),

('Via Genova', 8, 16100, 'Genova', 'GE', 'Italia'),

('Via Palermo', 9, 90100, 'Palermo', 'PA', 'Italia'),

('Via Cagliari', 10, 09100, 'Cagliari', 'CA', 'Italia'),

('Via Trieste', 11, 60100, 'Ancona', 'AN', 'Italia'),

('Via Torino', 12, 10100, 'Torino', 'TO', 'Italia'),

('Via Verona', 13, 37100, 'Verona', 'VR', 'Italia'),

('Via Padova', 14, 35100, 'Padova', 'PD', 'Italia'),

('Via Perugia', 15, 06100, 'Perugia', 'PG', 'Italia'),

('Via Potenza', 16, 85100, 'Potenza', 'PZ', 'Italia'),

('Via Catanzaro', 17, 88100, 'Catanzaro', 'CZ', 'Italia'),

('Via Reggio Calabria', 18, 89100, 'Reggio Calabria', 'RC',
'Italia'),

("Via L'Aquila", 19, 67100, "L'Aquila", 'AQ', 'Italia'),

('Via Campobasso', 20, 86100, 'Campobasso', 'CB',
'Italia');
```

5.3.2 Corrieri

```
INSERT INTO Corrieri (Nome) VALUES

('DHL'),

('FedEx'),
```

```
('UPS'),

('TNT'),

('GLS'),

('SDA'),

('Nexive'),

('Bartolini'),

('Poste Italiane'),

('TNT Express');
```

5.3.3 Clienti

```
INSERT INTO Clienti (Nome, Cognome, Data_di_nascita,
Id_Indirizzo) VALUES

('Mario', 'Rossi', '1980-01-01', 1),

('Luigi', 'Verdi', '1990-02-02', 2),

('Anna', 'Bianchi', '2000-03-03', 3),

('Carla', 'Neri', '1975-04-04', 4),

('Giovanni', 'Gialli', '1985-05-05', 5),

('Paolo', 'Blu', '1995-06-06', 6),

('Sara', 'Rosa', '2001-07-07', 7),

('Luca', 'Marrone', '1988-08-08', 8),

('Elena', 'Arancio', '1999-09-09', 9),
```

```
('Davide', 'Viola', '1992-10-10', 10);
```

5.3.4 Carte di Credito

```
INSERT INTO Carte_di_Credito (Numero_Carta, Tipo_Carta,
Scadenza, CVV, Nome_Carta, Cognome_Carta, Id_Cliente)
VALUES

('1234567890123456', 'Visa', '2025-01-01', '123', 'Mario',
'Rossi', 1),

('2345678901234567', 'Mastercard', '2026-02-02', '234',
'Luigi', 'Verdi', 2),

('3456789012345678', 'American Express', '2027-03-03',
'345', 'Anna', 'Bianchi', 3),

('4567890123456789', 'Discover', '2028-04-04', '456',
'Carla', 'Neri', 4),

('5678901234567890', 'Visa', '2029-05-05', '567',
'Giovanni', 'Gialli', 5),

('6789012345678901', 'Mastercard', '2030-06-06', '678',
'Paolo', 'Blu', 6),

('7890123456789012', 'American Express', '2031-07-07',
'789', 'Sara', 'Rosa', 7),

('8901234567890123', 'Discover', '2032-08-08', '890',
'Luca', 'Marrone', 8),

('9012345678901234', 'Visa', '2033-09-09', '901', 'Elena',
'Arancio', 9),

('0123456789012345', 'Mastercard', '2034-10-10', '012',
'Davide', 'Viola', 10);
```

5.3.5 Produttori

```
INSERT INTO Produttori (Nome, Partita_IVA, Id_Indirizzo)
VALUES

('Cantina Sociale', 12345678901, 11),

('Vigneti Rossi', 23456789012, 12),

('Azienda Agricola Bianchi', 34567890123, 13),

('Vinicola Neri', 45678901234, 14),

('Enoteca Gialli', 56789012345, 15),

('Cantine Blu', 67890123456, 16),

('Viticoltori Rosa', 78901234567, 17),

('Azienda Marrone', 89012345678, 18),

('Vini Arancio', 90123456789, 19),

('Vinicola Viola', 10123456780, 20);
```

5.3.6 Tipi di Vino

```
INSERT INTO Tipi_di_Vino (Nome) VALUES

('Rosso'),

('Bianco'),

('Rosato'),

('Spumante');
```

5.3.7 Vini

```
INSERT INTO Vini (Nome, Anno, Solfiti, Biologico,
Tipologia, Gradazione, Id_Produttore, Prezzo) VALUES

('Chianti Classico', 2018, TRUE, FALSE, 'Rosso', 13.5, 1,
12.50),

('Barolo Riserva', 2016, TRUE, TRUE, 'Rosso', 14.0, 2,
25.00),

('Merlot Veneto', 2019, FALSE, FALSE, 'Rosso', 12.0, 3,
8.00),

('Pinot Grigio delle Venezie', 2020, TRUE, TRUE, 'Bianco',
11.5, 4, 10.00),

('Sauvignon Blanc', 2017, TRUE, FALSE, 'Bianco', 13.0, 5,
15.00),

('Rosso di Montalcino', 2015, TRUE, TRUE, 'Rosso', 13.5, 6,
18.00),

('Prosecco DOC', 2021, FALSE, FALSE, 'Spumante', 11.0, 7,
9.00),

('Spumante Brut', 2019, TRUE, FALSE, 'Spumante', 12.5, 8,
20.00),

('Rosato Toscana', 2020, TRUE, TRUE, 'Rosato', 12.0, 9,
7.50),

('Bianco di Custoza', 2021, FALSE, FALSE, 'Bianco', 11.5,
10, 6.50),

('Cabernet Sauvignon', 2018, TRUE, FALSE, 'Rosso', 13.8,
11, 14.00),

('Nero di Avola', 2017, TRUE, TRUE, 'Rosso', 14.5, 12,
```

```
20.00),

('Chardonnay', 2020, FALSE, FALSE, 'Bianco', 12.5, 13,
11.00),

('Vermentino', 2019, TRUE, FALSE, 'Bianco', 13.0, 14,
13.50),

('Franciacorta', 2018, FALSE, TRUE, 'Spumante', 12.0, 15,
22.00),

('Valpolicella Ripasso', 2016, TRUE, TRUE, 'Rosso', 13.5,
16, 19.00),

('Lambrusco', 2021, FALSE, FALSE, 'Rosso', 10.5, 17, 8.50),

('Gewurztraminer', 2019, TRUE, TRUE, 'Bianco', 14.0, 18,
18.50),

('Brunello di Montalcino', 2015, TRUE, FALSE, 'Rosso',
14.5, 19, 30.00),

('Orvieto Classico', 2020, FALSE, FALSE, 'Bianco', 12.0,
20, 9.50),

('Sangiovese', 2019, TRUE, FALSE, 'Rosso', 13.2, 21,
12.00),

('Zinfandel', 2018, TRUE, TRUE, 'Rosso', 14.3, 22, 21.00),

('Riesling', 2021, FALSE, FALSE, 'Bianco', 11.8, 23,
10.50),

('Grillo', 2020, TRUE, FALSE, 'Bianco', 12.9, 24, 13.00),

('Moscato di Asti', 2019, FALSE, TRUE, 'Spumante', 7.5, 25,
15.50),

('Primitivo', 2017, TRUE, TRUE, 'Rosso', 14.7, 26, 17.00),
```

```
('Trebbiano', 2021, FALSE, FALSE, 'Bianco', 11.2, 27,
7.50),

('Syrah', 2019, TRUE, TRUE, 'Rosso', 13.8, 28, 19.50),

('Amarone della Valpolicella', 2015, TRUE, FALSE, 'Rosso',
15.0, 29, 35.00),

('Pinot Nero', 2020, FALSE, FALSE, 'Rosso', 12.7, 30,
11.00);
```

5.3.8 Ordini

```
INSERT INTO Ordini (Id_Cliente, Stato, Id_Indirizzo,
Data_Ordine) VALUES

(1, 'In Preparazione', 1, '2023-01-01 10:00:00'),

(2, 'Spedito', 2, '2023-02-02 11:00:00'),

(3, 'Consegnato', 3, '2023-03-03 12:00:00'),

(4, 'In Preparazione', 4, '2023-04-04 13:00:00'),

(5, 'Spedito', 5, '2023-05-05 14:00:00'),

(6, 'Consegnato', 6, '2023-06-06 15:00:00'),

(7, 'In Preparazione', 7, '2023-07-07 16:00:00'),

(8, 'Spedito', 8, '2023-08-08 17:00:00'),

(9, 'Consegnato', 9, '2023-09-09 18:00:00'),

(10, 'In Preparazione', 10, '2023-10-10 19:00:00');
```

5.3.9 Lista Prodotti

```
INSERT INTO Lista_Prodotti (Id_Vino, Id_Ordine, Quantità)
VALUES

(1, 1, 2),

(2, 2, 1),

(3, 3, 3),

(4, 4, 1),

(5, 5, 2),

(6, 6, 1),

(7, 7, 4),

(8, 8, 1),

(9, 9, 2),

(10, 10, 1);
```

5.3.10 Pagamenti

```
INSERT INTO Pagamenti (Id_Ordine, Id_Carta, Data_Pagamento)
VALUES

(1, 1, '2023-01-01 10:30:00'),

(2, 2, '2023-02-02 11:30:00'),

(3, 3, '2023-03-03 12:30:00'),

(4, 4, '2023-04-04 13:30:00'),

(5, 5, '2023-05-05 14:30:00'),
```

```
(6, 6, '2023-06-06 15:30:00'),

(7, 7, '2023-07-07 16:30:00'),

(8, 8, '2023-08-08 17:30:00'),

(9, 9, '2023-09-09 18:30:00'),

(10, 10, '2023-10-10 19:30:00');
```

5.3.11 Preferire

```
INSERT INTO Preferire (Id_Cliente, Id_Vino) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6),

(7, 7),

(8, 8),

(9, 9),

(10, 10);
```

5.3.12 Recensire

```
INSERT INTO Recensire (Id_Cliente, Id_Vino, Voto, Commento,
Data_recensione) VALUES
```

```
(1, 1, 8, 'Ottimo vino!', '2023-01-10 10:00:00'),

(2, 2, 9, 'Fantastico!', '2023-02-10 11:00:00'),

(3, 3, 7, 'Buono.', '2023-03-10 12:00:00'),

(4, 4, 6, 'Niente di speciale.', '2023-04-10 13:00:00'),

(5, 5, 10, 'Eccellente!', '2023-05-10 14:00:00'),

(6, 6, 5, 'Non male.', '2023-06-10 15:00:00'),

(7, 7, 8, 'Molto buono.', '2023-07-10 16:00:00'),

(8, 8, 9, 'Fantastico!', '2023-08-10 17:00:00'),

(9, 9, 7, 'Buono.', '2023-09-10 18:00:00'),

(10, 10, 6, 'Niente di speciale.', '2023-10-10 19:00:00');
```

5.3.13 Premi

```
INSERT INTO Premi (Nome, Organizzazione) VALUES

('Premio Eccellenza', 'Organizzazione 1'),

('Premio Qualità', 'Organizzazione 2'),

('Premio Innovazione', 'Organizzazione 3'),

('Premio Sostenibilità', 'Organizzazione 4'),

('Premio Tradizione', 'Organizzazione 5'),

('Premio Miglior Vino', 'Organizzazione 6'),

('Premio Miglior Produttore', 'Organizzazione 7'),
```

```
('Premio Miglior Design', 'Organizzazione 8'),

('Premio Miglior Gusto', 'Organizzazione 9'),

('Premio Miglior Annata', 'Organizzazione 10');
```

5.3.14 Assegnare

```
INSERT INTO Assegnare (Id_Premio, Id_Vino,
Data_Assegnazione) VALUES

(1, 1, '2024-01-01'),

(2, 2, '2024-02-02'),

(3, 3, '2024-03-03'),

(4, 4, '2021-04-04'),

(5, 5, '2024-05-05'),

(6, 6, '2024-06-06'),

(7, 7, '2024-07-07'),

(8, 8, '2024-08-08'),

(9, 9, '2024-09-09'),

(10, 10, '2022-10-10');
```

5.3.15 Ricevere

```
INSERT INTO Ricevere (Id_Premio, Id_Produttore,
Data_Ricezione) VALUES

(1, 1, '2024-01-01'),
```

```
(2, 2, '2024-02-02'),

(3, 3, '2024-03-03'),

(4, 4, '2024-04-04'),

(5, 5, '2024-05-05'),

(6, 6, '2024-06-06'),

(7, 7, '2024-07-07'),

(8, 8, '2024-08-08'),

(9, 9, '2024-09-09'),

(10, 10, '2024-10-10');
```

5.3.16 Uve

```
INSERT INTO Uve (Nome, Descrizione) VALUES

('Cabernet Sauvignon', 'Descrizione Cabernet Sauvignon'),

('Merlot', 'Descrizione Merlot'),

('Sangiovese', 'Descrizione Sangiovese'),

('Nebbiolo', 'Descrizione Nebbiolo'),

('Pinot Nero', 'Descrizione Pinot Nero'),

('Chardonnay', 'Descrizione Chardonnay'),

('Sauvignon Blanc', 'Descrizione Sauvignon Blanc'),

('Trebbiano', 'Descrizione Trebbiano'),

('Barbera', 'Descrizione Barbera'),
```

```
('Malvasia', 'Descrizione Malvasia');
```

5.3.17 Compone

```
INSERT INTO Compone (Id_Vino, Id_Uva, Percentuale) VALUES
(1, 1, 60),
(1, 3, 40),
(2, 4, 100),
(3, 2, 100),
(4, 6, 100),
(5, 7, 100),
(6, 3, 80),
(6, 9, 20),
(7, 8, 100),
(8, 6, 50),
(8, 7, 50),
(9, 5, 100),
(10, 6, 70),
(10, 10, 30);
```

5.3.18 Magazzini

```
INSERT INTO Magazzini (Id_Indirizzo, Nome) VALUES

(1, 'Magazzino Torino'),

(2, 'Magazzino Milano'),

(3, 'Magazzino Napoli'),

(4, 'Magazzino Firenze'),

(5, 'Magazzino Venezia'),

(6, 'Magazzino Bari'),

(7, 'Magazzino Bologna'),

(8, 'Magazzino Genova'),

(9, 'Magazzino Palermo'),

(10, 'Magazzino Cagliari');
```

5.3.19 Contenere

```
INSERT INTO Contenere (Id_Magazzino, Id_Vino,
Quantità_Rimanente) VALUES

(1, 1, 100),

(2, 2, 50),

(3, 3, 75),

(4, 4, 100),

(5, 5, 50),

(6, 6, 60),
```

```
(7, 7, 30),

(8, 8, 40),

(9, 9, 20),

(10, 10, 10);
```

5.3.20 Spedizioni

```
INSERT INTO Spedizioni (Id_Ordine, Id_Corriere,
Data_Spedizione, Data_Consegna, Stato_Spedizione) VALUES

(1, 1, '2023-01-01 09:00:00', '2023-01-02 09:00:00', 'In
Viaggio'),

(2, 2, '2023-02-01 10:00:00', '2023-02-03 10:00:00', 'In
Viaggio'),

(3, 3, '2023-03-01 11:00:00', '2023-03-05 11:00:00',
'Consegnato'),

(4, 4, '2023-04-01 12:00:00', '2023-04-04 12:00:00',
'Consegnato'),

(5, 5, '2023-05-01 13:00:00', '2023-05-02 13:00:00',
'Consegnato'),

(6, 6, '2023-06-01 14:00:00', '2023-06-03 14:00:00',
'Consegnato'),

(7, 7, '2023-07-01 15:00:00', '2023-07-04 15:00:00',
'Consegnato'),

(8, 8, '2023-08-01 16:00:00', '2023-08-05 16:00:00',
'Consegnato'),

(9, 9, '2023-09-01 17:00:00', '2023-09-06 17:00:00',
```

```
'Consegnato'),

(10, 10, '2023-10-01 18:00:00', '2023-10-04 18:00:00',
'Consegnato');
```

5.4 Query

1. Conta il numero di vini di una certa tipologia (es: rosso)

```
SELECT
        COUNT(vini.Id_Vino) AS Num_Vini_Rossi
FROM
        vini
WHERE
        vini.Tipologia = "Rosso";
```

2. Restituisce le 10 città con più clienti

```
SELECT
        indirizzi.Città,
        COUNT(clienti.Id_Cliente) AS Numero_Clienti
FROM
        clienti
INNER JOIN indirizzi ON clienti.Id_Indirizzo =
indirizzi.Id_Indirizzo
GROUP BY
        Città
ORDER BY
        Numero_Clienti
DESC
;
```

3. Restituisce i vini che rispettano alcuni parametri presi in input (annata, prezzo, tipologia)

```
SELECT
        *
FROM
        vini
WHERE
        vini.Tipologia = "Rosso" AND vini.Prezzo <= 20 AND
vini.Anno > 2015;
```

4. Trova i clienti che hanno fatto i 10 ordini più costosi.

```
SELECT
        clienti.Nome, clienti.Cognome, ordini.Id_Ordine,
        SUM(
                Lista_Prodotti.Quantità * vini.Prezzo
        ) AS Totale_Ordine
FROM clienti INNER JOIN
        ordini ON clienti.Id_Cliente = ordini.Id_Cliente
INNER JOIN Lista_Prodotti ON ordini.Id_Ordine =
Lista_Prodotti.Id_Ordine
INNER JOIN vini ON Lista_Prodotti.Id_Vino = vini.Id_Vino
GROUP BY
        ordini.Id_Ordine
ORDER BY
        Totale_Ordine
DESC
LIMIT 10;
```

5. Trova i 10 vini più costosi.

```
SELECT
        *
FROM
        vini
ORDER BY
        vini.Prezzo
DESC
LIMIT 10;
```

6. Restituisce quanti ordini hanno fatto i clienti

```
SELECT
        clienti.Nome,
        clienti.Cognome,
COUNT(ordini.Id_Ordine) AS Num_Ordini
FROM
        clienti
INNER JOIN ordini ON clienti.Id_Cliente = ordini.Id_Cliente
GROUP BY
        ordini.Id_Cliente;
```

7. Restituisce i vini col maggior numero di preferiti

```
SELECT
        vini.Id_Vino,
        vini.Nome,
        COUNT(preferire.Id_Cliente) AS Num_Preferiti
FROM
        vini
INNER JOIN preferire ON vini.Id_Vino = preferire.Id_Vino
GROUP BY
        vini.Id_Vino;
```

8. Restituisce i 10 vini più venduti

```
SELECT
        vini.Id_Vino,
        vini.Nome,
        SUM(
                vini.Prezzo * Lista_Prodotti.Quantità
        ) AS Totale_Venduto
FROM
        vini
INNER JOIN Lista_Prodotti ON vini.Id_Vino =
Lista_Prodotti.Id_Vino
INNER JOIN ordini ON Lista_Prodotti.Id_Ordine =
ordini.Id_Ordine
```

```
GROUP BY
        vini.Id_Vino
ORDER BY
        Totale_Venduto
DESC
LIMIT 10;
```

9. Restituisce i 10 vini meglio recensiti ed il loro prezzo

```
SELECT
        vini.Id_Vino,
        vini.Nome,
        vini.Prezzo,
        ROUND(AVG(recensire.Voto),
        2) AS Voto_Medio
FROM
        vini
INNER JOIN recensire ON vini.Id_Vino = recensire.Id_Vino
GROUP BY
        vini.Id_Vino
ORDER BY
        Voto_Medio
DESC
LIMIT 10;
```

10. Restituisce i 10 vini più premiati dalla critica

```
SELECT
        vini.Id_Vino,
        vini.Nome,
        COUNT(assegnare.Id_Premio) AS Num_Premi
FROM
        vini
INNER JOIN assegnare ON vini.Id_Vino = assegnare.Id_Vino
GROUP BY
        vini.Id_Vino
ORDER BY
        Num_Premi
```

```
DESC
LIMIT 10;
```

11. Restituisce i 10 produttori di vino più premiati dalla critica

```
SELECT
        produttori.Nome,
        COUNT(ricevere.Id_Premio) AS Num_Premi
FROM
        produttori
INNER JOIN ricevere ON produttori.Id_Produttore =
ricevere.Id_Produttore
GROUP BY
        produttori.Id_Produttore
ORDER BY
        Num_Premi
DESC;
```

12. Trova i clienti che hanno ordinato di più

```
SELECT
        clienti.Id_Cliente, clienti.Nome, clienti.Cognome
FROM
        clienti
INNER JOIN ordini ON clienti.Id_Cliente = ordini.Id_Cliente
INNER JOIN Lista_Prodotti ON ordini.Id_Ordine =
Lista_Prodotti.Id_Ordine
INNER JOIN vini ON Lista_Prodotti.Id_Vino = vini.Id_Vino
GROUP BY
        clienti.Id_Cliente,
        Lista_Prodotti.Id_Lista;
```

13. Restituisce il fatturato totale in un certo periodo

```
SELECT
    o.Id_Ordine,
    SUM(l.Quantità * v.Prezzo) AS FatturatoTot
FROM
```

```
    Ordini o
JOIN
    Lista_Prodotti l ON o.Id_Ordine = l.Id_Ordine
JOIN
    Vini v ON l.Id_Vino = v.Id_Vino
WHERE
    o.Data_Ordine BETWEEN '2023-01-01' AND '2023-12-31'
GROUP BY
    o.Id_Ordine
ORDER BY
    FatturatoTot;
```

14. Restituisce la lista dei vini novelli

```
SELECT
        *
FROM
        vini
WHERE
        vini.Anno = YEAR(CURDATE()) AND vini.gradazione >=
11;
```

15. Restituisce tutti i vini fatti con una percentuale arbitraria di tipo di uva

```
SELECT
    v.Id_Vino,
    v.Nome,
    c.Percentuale,
    u.Tipo
FROM
    Vini v
JOIN
    Compone c ON v.Id_Vino = c.Id_Vino
JOIN
    Uva u ON c.Id_Uva = u.Id_Uva
WHERE
    c.Percentuale >= 120
    AND u.Tipo = 'Chardonay'
```

16. Elenca tutti i vini prodotti da un particolare produttore

```
SELECT
        Nome,
        Anno,
        Prezzo
FROM
        Vini
WHERE
        Id_Produttore = 1; -- Sostituire 1 con
l'Id_Produttore desiderato
```

17. Visualizza tutti gli ordini effettuati da un cliente specifico

```
SELECT
        Id_Ordine,
        Stato,
        Data_Ordine
FROM
        Ordini
WHERE
        Id_Cliente = 1; -- Sostituire 1 con l'Id_Cliente
desiderato
```

18. Trova tutti i clienti che hanno acquistato un vino specifico

```
SELECT DISTINCT
        Clienti.Id_Cliente,
        Clienti.Nome,
        Clienti.Cognome
FROM
        Clienti
JOIN Ordini ON Clienti.Id_Cliente = Ordini.Id_Cliente.
JOIN Lista_Prodotti L ON Ordini.Id_Ordine =
Lista_Prodotti.Id_Ordine
WHERE
        Lista_Prodotti.Id_Vino = 1; -- Sostituire 1 con
l'Id_Vino desiderato
```

19. Trova i vini con una media di voti superiore a 8

```
SELECT
        V.Nome,
        AVG(R.Voto) AS MediaVoti
FROM
        Vini V
JOIN Recensire R ON V.Id_Vino = R.Id_Vino
GROUP BY
        V.Nome
HAVING
        AVG(R.Voto) > 8;
```

5.4.1 Ottimizzazione query

Riportiamo un esempio di ottimizzazione delle Query tramite Index: Utilizzeremo il seguente Snippet di codice che restituisce tutti i clienti di una data nazione:

```
SELECT
        Clienti.Nome , Clienti.Cognome,Indirizzi.CAP
,Indirizzi.Citta , Indirizzi.Provincia
FROM
        Indirizzi
INNER JOIN Clienti ON Indirizzi.Id_Indirizzo =
Clienti.Id_Indirizzo
WHERE
        Indirizzi.Nazione = 'Italia' ;
```

- Tempo richiesto dalla Query prima dell'ottimizzazione tramite Index:

  200 row(s) fetched - 0,015s (0,001s fetch), on 2024-0

- Tempo richiesto dalla Query dopo l'ottimizzazione tramite Index:

  200 row(s) fetched - 0,009s (0,001s fetch), on 2024-0

  Il miglioramento ottenuto risulta dalla formula seguente:

$$100 \cdot \frac{0,015 - 0,009}{0,015} \approx 40\%$$

5.4.2 Index

```
CREATE INDEX idx_indirizzi_cap ON Indirizzi (CAP);
CREATE INDEX idx_indirizzi_citta ON Indirizzi (Città);
CREATE INDEX idx_indirizzi_provincia ON Indirizzi
(Provincia);

CREATE INDEX idx_clienti_nome ON Clienti (Nome);
CREATE INDEX idx_clienti_cognome ON Clienti (Cognome);

CREATE INDEX idx_vini_nome ON Vini (Nome);
CREATE INDEX idx_vini_anno ON Vini (Anno);
CREATE INDEX idx_vini_tipologia ON Vini (Tipologia);

CREATE INDEX idx_ordini_id_cliente ON Ordini (Id_Cliente);
CREATE INDEX idx_ordini_data_ordine ON Ordini
(Data_Ordine);

CREATE INDEX idx_spedizioni_id_ordine ON Spedizioni
(Id_Ordine);
CREATE INDEX idx_spedizioni_data_spedizione ON Spedizioni
(Data_Spedizione);

CREATE INDEX idx_recensioni_id_vino ON Recensioni
(Id_Vino);
CREATE INDEX idx_recensioni_id_cliente ON Recensioni
(Id_Cliente);

CREATE INDEX idx_compone_id_vino ON Compone (Id_Vino);
CREATE INDEX idx_compone_id_uva ON Compone (Id_Uva);

CREATE INDEX idx_lista_prodotti_id_ordine ON Lista_Prodotti
(Id_Ordine);
CREATE INDEX idx_lista_prodotti_id_vino ON Lista_Prodotti
(Id_Vino);
```

5.5 Algebra Relazionale

L'algebra relazionale è un linguaggio query procedurale in notazione algebrica. In una query, si applicano sequenzialmente le operazioni alle relazioni. Ogni operazione (unaria o binaria) riceve in input una relazione e ne produce un'altra in output.

Le operazioni primitive sono:

- Selezione (σ)
- Proiezione (π)
- Unione (⋃)
- Differenza Insiemistica (-)
- Prodotto Cartesiano (X)
- Ridenominazione (ρ)

Esistono altre operazioni da esse derivabili, tra cui l'intersezione insiemistica (⋂).
Di seguito troviamo alcune query sul nostro database scritte in Algebra Relazionale:

- *Restituisce i vini che rispettano alcuni parametri presi in input (annata, prezzo, tipologia)*
  In algebra relazionale sarà:

$$\sigma_{vini.Tipologia='Rosso' \wedge vini.Prezzo \leq 20 \wedge vini.Anno > 2015}(Vini)$$

- *Visualizza tutti gli ordini effettuati da un cliente specifico*
  In algebra relazionale sarà:

$$\pi_{IdOrdine,Stato,Data\_Ordine}(\sigma_{IdCliente=1}(Ordini))$$

- *Trova i vini con una media di voti superiore a 8*
  In algebra relazionale sarà:

$$\sigma_{MediaVoti>8}(\gamma_{Vini.Nome;AVG(Recensioni.Voto) \rightarrow MediaVoti}(\pi_{Vini.Nome,Recensioni.Voto}(Vini \bowtie_{Vini.Id\_Vino=Recensioni.Id\_Vino} Recensioni)))$$

5.6 Calcolo Relazionale

Il calcolo relazionale è un linguaggio query non procedurale ma dichiarativo. Invece dell'algebra, utilizza il calcolo dei predicati matematici del primo ordine in notazione logica. L'output di una query è una relazione che contiene solo tuple che soddisfano le formule logiche espresse. Il potere espressivo del calcolo relazionale è dunque equivalente a quello dell'algebra relazionale. Versioni:

1. Calcolo relazionale sui domini
2. Calcolo relazionale sulle tuple con dichiarazione di range

Di seguito sono alcune query espresse tramite il calcolo relazionale sulle tuple con dichiarazione di range:

- *Restituisce i vini che rispettano alcuni parametri presi in input (annata, prezzo, tipologia)*

$$P = \{(v \mid v \in Vini, v.Tipologia =' Rosso', v.Prezzo \leq 20, v.Anno > 2015)\}$$

- *Restituisce i 10 vini più premiati dalla critica*

$$P = \{(v.Id\_Vino, v.Nome, COUNT(a.Id\_Premio) \rightarrow NumPremi) \mid v \in vini, a \in assegnare, v.Id\_Vino = a.Id\_Vino\}$$

- *Visualizza tutti gli ordini effettuati da un cliente specifico*

$$P = \{(o.Id\_Ordine, o.Stato, o.Data\_Ordine) \mid o \in Ordine, o.Id\_Cliente = 1\}$$

- *Trova i vini con una media di voti superiore a 8*

$$P = \{(v.Nome, AVG(r.Voto) \rightarrow MediaVoti) \mid v \in Vini, r \in Recensioni, v.Id\_Vino = r.Id\_Vino, AVG(r.Voto) > 8\}$$

5.7 Views

```
CREATE VIEW Dettagli_Cliente AS SELECT
c.Id_Cliente,
c.Nome,
c.Cognome,
c.Data_di_nascita,
i.Via,
i.CAP,
i.Città,
i.Provincia,
i.Nazione
FROM
Clienti c
JOIN Indirizzi i ON
c.Id_Indirizzo = i.Id_Indirizzo;

-- Fornisce una vista completa dei dettagli contenente solo
i vini rossi

CREATE VIEW Vini_Rossi AS SELECT
vini.Id_Vino,
vini.Nome AS Nome_Vino,
produttori.Nome AS Nome_Produttore,
vini.Anno,
vini.Solfiti,
vini.Biologico,
vini.Prezzo
FROM
Vini
INNER JOIN Produttori ON vini.Id_Produttore =
produttori.Id_Produttore
WHERE
tipologia = "Rosso";

-- Fornisce una vista completa dei dettagli contenente solo
```

```
i vini bianchi


CREATE VIEW Vini_Bianchi AS SELECT
vini.Id_Vino,
vini.Nome AS Nome_Vino,
produttori.Nome AS Nome_Produttore,
vini.Anno,
vini.Solfiti,
vini.Biologico,
vini.Prezzo
FROM
Vini
INNER JOIN Produttori ON vini.Id_Produttore =
produttori.Id_Produttore
WHERE tipologia = "Bianco";

-- Fornisce una vista dei 10 vini più venduti

CREATE VIEW Vini_più_venduti AS
SELECT vini.Id_Vino,
vini.Nome,
SUM( vini.Prezzo * Lista_Prodotti.Quantità) AS
Totale_Venduto
FROM vini
INNER JOIN Lista_Prodotti ON vini.Id_Vino =
Lista_Prodotti.Id_Vino
INNER JOIN ordini ON Lista_Prodotti.Id_Ordine =
ordini.Id_Ordine
GROUP BY vini.Id_Vino
ORDER BY Totale_Venduto DESC
LIMIT 10;
```

5.8 Procedure

```
DELIMITER //

CREATE PROCEDURE AggiungiVinoOrdine (
```

```
    IN p_Id_Ordine INT,
    IN p_Id_Vino INT,
    IN p_Quantità INT
)
BEGIN
    DECLARE v_QuantitàRimanente INT;

    -- Controlla la disponibilità nel magazzino
    SELECT Quantità_Rimanente INTO v_QuantitàRimanente
    FROM Contenere
    WHERE Id_Vino = p_Id_Vino;

    IF v_QuantitàRimanente < p_Quantità THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'Quantità non disponibile nel magazzino.';
    END IF;

    -- Aggiunge il vino all'ordine
    INSERT INTO Lista_Prodotti (Id_Ordine, Id_Vino,
Quantità)
    VALUES (p_Id_Ordine, p_Id_Vino, p_Quantità);

    -- Aggiorna la quantità rimanente nel magazzino
    UPDATE Contenere
    SET Quantità_Rimanente = Quantità_Rimanente -
p_Quantità
    WHERE Id_Vino = p_Id_Vino;
END //

DELIMITER ;
```

6. Sicurezza

In questa sezione mostriamo il codice necessario alla creazione di profili diversi i quali possono essere limitati nei comandi che possono eseguire sul nostro database.

6.1 Utenti

```
CREATE USER 'admin'@'%' IDENTIFIED BY 'strong_password';
GRANT ALL PRIVILEGES ON EnoMarketDB.* TO 'admin'@'%';

CREATE USER 'data_analyst'@'%' IDENTIFIED BY
'strong_password';
GRANT SELECT ON EnoMarketDB.* TO 'data_analyst'@'%';

CREATE USER 'app_user'@'%' IDENTIFIED BY 'strong_password';
GRANT SELECT, INSERT, UPDATE, DELETE ON EnoMarketDB.* TO
'app_user'@'%';

-- È possibile anche garantire permessi per singole tabelle

GRANT SELECT, INSERT ON EnoMarketDB.Ordini TO
'app_user'@'%';
GRANT SELECT, UPDATE ON EnoMarketDB.Spedizioni TO
'app_user'@'%';

CREATE USER 'read_only'@'%' IDENTIFIED BY
'strong_password';
GRANT SELECT ON EnoMarketDB.* TO 'read_only'@'%';
```