Progettazione di una Base di Dati

Sistema informativo per la gestione delle scorte di vini di un'enoteca

Basi di Dati
Giovanni Prisco
Anno Accademico 2020/2021

Raccolta dati sulla realtà d'interesse

1.1 **Introduzione**

Come si evince dal titolo si vuole progettare una base di dati per gestire le scorte di vino di un'enoteca. I dati di cui ci serviremo riguarderanno appunto i vini, le loro caratteristiche quali colore, tipologia, famiglia di appartenenza, denominazione, azienda produttrice e altro ancora.

1.2 **Specifiche**

La cantina di un'enoteca comprende una serie di vini di cui si vogliono conoscere le specifiche.

Possiamo dire che una bottiglia di vino è individuata univocamente attraverso il nome della bottiglia e la sua annata ed è caratterizzata da: colore del vino (rosso, bianco, rosato...), tipologia del vino (Spumante, Passito, Liquoroso, Fermo...), denominazione del vino (DOCG...), azienda produttrice e regione (e la nazione) di origine.

Dell'azienda produttrice ci interessa sapere il nome e l'indirizzo.

Un vino è anche caratterizzato da una mescolanza di uve che lo compongono, quindi ci interessa sapere per ognuna di queste uve, il nome e la percentuale con la quale compare nel vino.

Le caratteristiche del vino sono sempre comuni ad un insieme più o meno piccolo di vini, quest'insieme è chiamato famiglia, e la famiglia di un vino ne determina colore, tipologia denominazione e regione d'origine.

Un'enoteca avrà bisogno di svolgere diverse operazioni che il nostro applicativo dovrà garantire:

1. Inserire una famiglia di vini;

2. Inserire una denominazione;
3. Inserire un tipo di uva;
4. Inserire un vino;
5. Inserire l'uvaggio per un vino;
6. Ottenere una lista dei vini disponibili in cantina;
7. Modificare il prezzo di una bottiglia;
8. Modificare il numero di pezzi disponibili per una certa bottiglia;
9. Eliminare definitivamente un vino dalle proprie scorte;

1.3 **Glossario**

Di seguito un piccolo glossario per meglio spiegare le terminologie che verranno utilizzate di seguito:

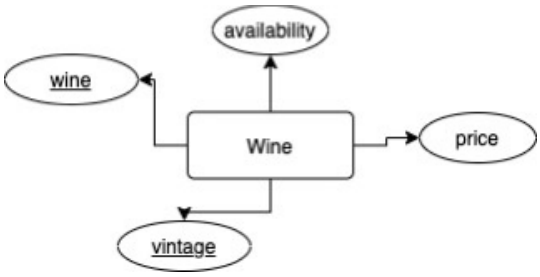
- **Vino:** entità principale contenente le informazioni principali sul vino;
- **Colore:** il colore caratteristico del vino (rosso, bianco...);
- **Tipologia:** la tipologia del vino (Spumante, Fermo, Passito...);
- **Uvaggi:** letteralmente la mescolanza di uve utilizzata per ottenere quello specifico vino;
- **Famiglia:** una famiglia di vini è un'insieme di caratteristiche, quali colore, tipologia e origine) condivise da tutti i vini di quella famiglia, tutti i vini ne hanno una;
- **Denominazione:** la denominazione della famiglia di vini (es. DOCG);
- **Produttore:** l'azienda che ha prodotto il vino in questione;
- **Identificativo:** per il resto della documentazione, in assenza di ulteriori spiegazioni, sarà utilizzato il termine identificativo per indicare un numero intero positivo che identifica univocamente un record di un'entità nella base di dati.

N.B. Da ora in pi sarà utilizzata la seguente convenzione: l'attributo che porta il nome della sua entità indica il nome di quest'ultima, es. l'attributo wine di Wine è il nome del vino, così come winery è il nome di una Winery.

Produzione del modello Entity-Relationship

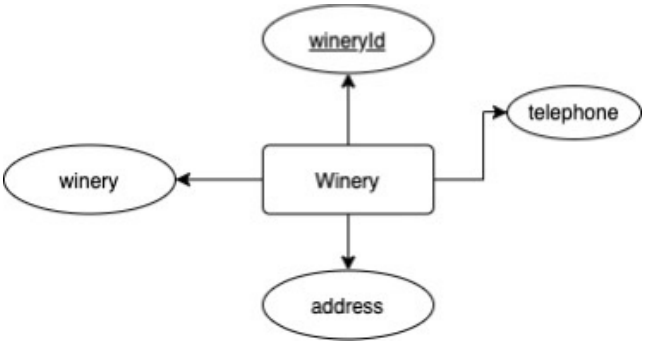
2.1 **Dal minimondo allo schema ER**

Per la produzione dello schema ER cominciamo a distinguere le varie entità, di certo avremo la prima chiamata Vino:



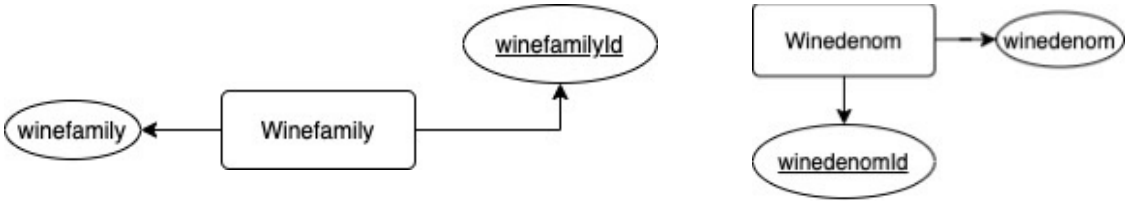
Un vino, come già detto in precedenza, sarà identificato univocamente dal nome della bottiglia (wine) e dalla sua annata, il che implica che di una stessa bottiglia possono esistere diverse annate, un numero positivo a virgola mobile che indica il prezzo in euro e un intero positivo che rappresenta il numero di bottiglie disponibili.

Come detto in precedenza, di un vino ci interessa anche l'azienda che l'ha prodotto:



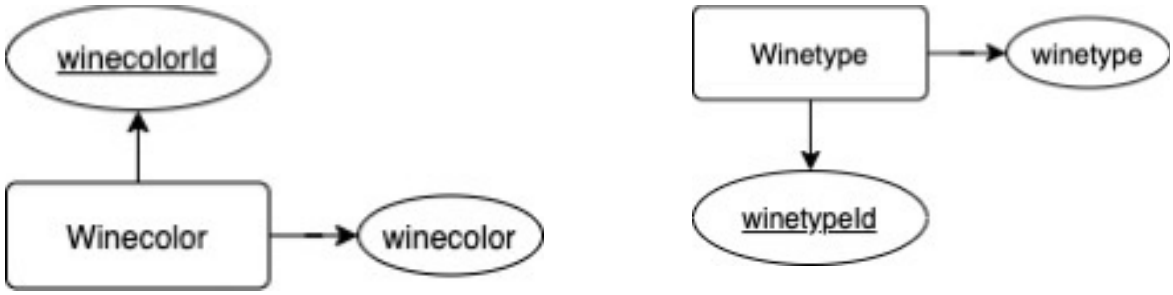
Dell'azienda avremo chiaramente un identificativo, una stringa nome (winery), l'indirizzo della sede (stringa) e il numero di telefono.

Abbiamo parlato anche di *famiglia* di vini, l'entità che determina le caratteristiche del vino in termini di colore e tipologia, oltre alla sua denominazione (es. DOCG):



Come si vede, della famiglia di vini ci interessa solo il nome (oltre all'identificativo), la cosa importante di questa entità è l'associazione con le entità Colore e Tipologia:

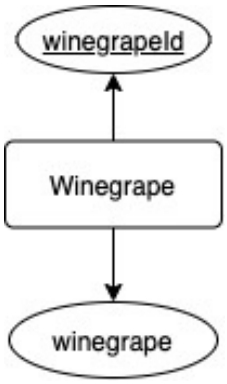
Identifichiamo Colore e Tipologia e diamo loro un nome (rispettivamente winecolor e winetype).



La famiglia di un vino ci dà informazioni anche sull'origine del vino, per questo motivo aggiungeremo altre due entità, **Region** e **Country:**

Queste due entità saranno associate a Winefamily e avranno (come da immagine) un identificativo e rispettivamente il nome della regione e il nome della nazione.

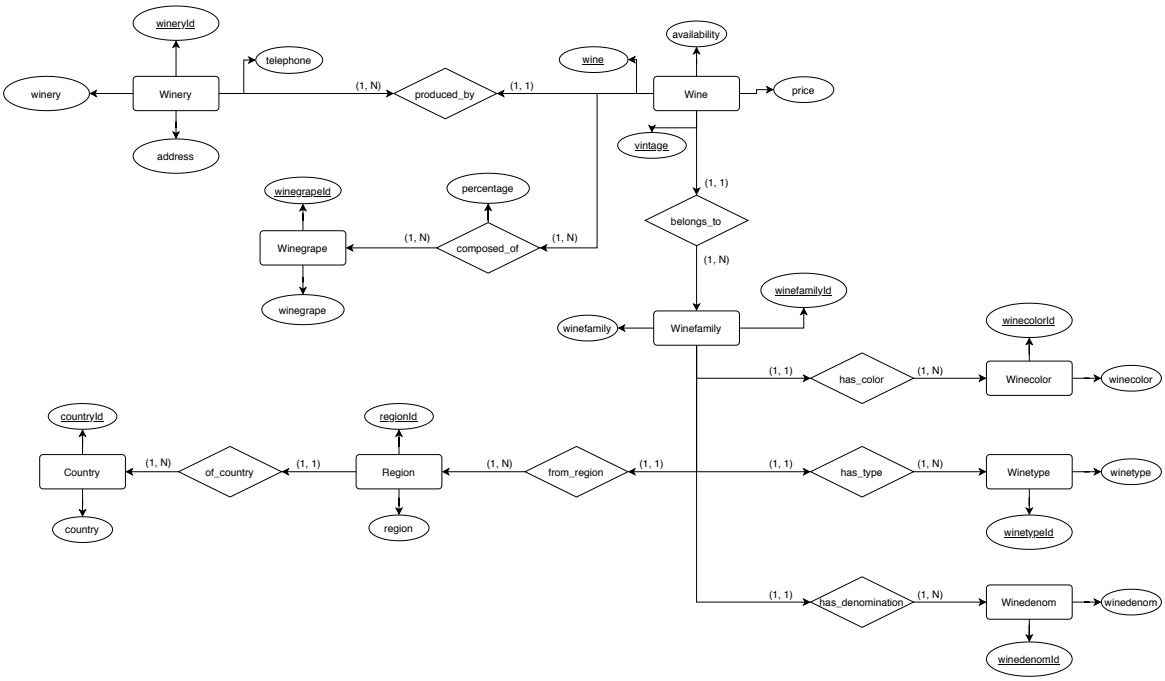
Le ultime informazioni che vogliamo salvare di un vino sono gli uvaggi, per fare questo ci serviremo della nostra ultima entità:



Un'uva è identificata da winegrapeId e ha un nome che la caratterizza.

2.2 **Schema ER Completo**

Mettendo insieme tutte le informazioni raccolte fino ad ora riusciamo ad ottenere lo schema completo di relazioni tra le entità:



Analizzandole una ad una:

- **produced_by:** associa un Vino ad un'azienda, indica il produttore del vino;
- **belongs_to:** associa un Vino ad una Famiglia, indica la famiglia di appartenenza del vino;
- **has_color:** associa la Famiglia ad un Colore, indica il colore caratteristico di quella data Famiglia;
- **has_type:** associa la Famiglia ad un Tipo (Winetype), indica la tipologia di quella Famiglia;
- **has_denomination:** associa la Famiglia ad una Denominazione (Winedenom), indica la denominazione di quella Famiglia;
- **composed_of:** è un'associazione molti a molti, associa N Vini ad N Uve (Winegrape), la lista di associazioni di un singolo vino (insieme alla percentuale, un intero da 0 a 100) dà come risultato l'uvaggio del Vino;
- **from_region:** associa una Famiglia ad una Regione, indica la regione a cui appartengono i Vini di quella Famiglia;
- **of_country:** associa una Regione ad una Nazione, indica a quale Nazione appartiene quella data Regione;

2.3 **Vincoli**

Come si evince dallo schema ER, **Winefamily** non può non avere **Winecolor**, **Winetype**, **Winedenom** e **Region** per essere considerata integra.

Tutti i vini di **Wine** appartengono ad una **Winefamily** e sono stati prodotti da una **Winery**, nessun campo può quindi essere *null*.

2.4 **Operazioni**

Cominciamo col considerare il volume dei dati immagazzinati, al fine di semplificare il processo, assumiamo di avere solo vini italiani:

Concetto	Tipo	Volume
Wine	E	1000
Winery	E	750
Winefamily	E	500
Winecolor	E	4
Winetype	E	6
Winedenom	E	10
Winegrape	E	2000
Region	E	21
Country	E	1
composed_of	R	2000
produced_by	R	1000
belongs_to	R	1000
has_color	R	500
has_type	R	500
has_denom	R	500
from_region	R	500
of_country	R	21

Ricordiamo le operazioni richieste e per questo caso:

1. Inserire una famiglia di vini;
2. Inserire un'azienda;
3. Inserire un tipo di uva;
4. Inserire un vino;
5. Ottenere dettagli su un vino, comprese azienda, uvaggio, famiglia, colore, tipologia, denominazione, regione e nazione;
6. Modificare il prezzo di una bottiglia;

7. Modificare il numero di pezzi disponibili per una certa bottiglia;
8. Eliminare definitivamente un vino dalle proprie scorte.

Operazione	Tipo	Frequenza	Descrizione
Op1	I	2/mese	L'utente si rifornisce di un nuovo vino di una nuova famiglia, questa viene registrata
Op2	B	2/mese	L'utente si rifornisce di un nuovo vino di una nuova azienda, questa viene registrata
Op3	B	1/mese	L'utente scopre una nuova uva e decide di arricchire la propria base di dati
Op4	I	4/mese	L'utente acquista un nuovo tipo di vino per la propria cantina e lo registra associandolo alla relativa famiglia e registrando l'uvaggio
Op5	B	30/mese	L'utente mostra un proprio vino ad un cliente
Op6	B	2/mese	L'utente decide di modificare il prezzo di un vino
Op7	B	1000/mese	L'utente aggiorna le scorte ogni qualvolta vende un vino
Op8	B	1/mese	L'utente decide di non rifornirsi più di un certo tipo di vino

2.5 **Analisi del Carico Applicativo**

Analizziamo le operazioni una ad una calcolando il costo di ognuna:

**le seguenti descrizioni seguiranno la convenzione <concetto> <tipo_entità> <accessi> <tipo_accesso>, inoltre il costo di una scrittura (2S) è doppio rispetto a quello di una lettura (1L)

Op1: Inserire una famiglia di vini

Winefamily	E	1	S
has_color	R	1	S
has_type	R	1	S
has_denom	R	1	S
from_region	R	1	S

Totale: (2S + 2S + 2S + 2S + 2S) * 2/mese = **40/mese**

Op2: Inserire un'azienda

Winery	E	1	S

Totale: 2S * 2/mese = **8/mese**

Op3: Inserire un tipo di uva

Winegrape	E	1	S

Totale: 2S * 1/mese = **4/mese**

Op4: Inserire un vino

Wine	E	1	S
belongs_to	R	1	S
produced_by	R	1	S

Totale: (2S + 2S + 2S) * 4/mese = **48/mese**

Op5: Ottenere dettagli su un vini in cantina e la relativa famiglia e colore

Wine	E	1	L
produced_by	R	1	L
Winery	E	1	L
belongs_to	R	1	L
Winefamily	E	1	L
has_color	R	1	L
Winecolor	E	1	L
has_type	R	1	L
Winetype	E	1	L
has_denom	R	1	L
Winedenom	E	1	L
from_region	R	1	L
Region	E	1	L
of_country	R	1	L
Country	E	1	L
composed_of	R	2	L
Winegrape	E	2	L

Totale: (15 + 4)L * 30/mese = **570/mese**

Op6: Modificare il prezzo di una bottiglia

Wine E 1 L

Wine E 1 S

Totale: (1L + 2S) * 2/mese = **6/mese**

Op7: Modificare il numero di pezzi disponibili per una certa bottiglia

Wine E 1 L

Wine E 1 S

Totale: (1L + 2S) * 1000/mese = **5000/mese**

Op8: Eliminare definitivamente un vino dalle proprie scorte

Wine E 1 L

Wine E 1 S

Totale: (1L + 2S) * 1/mese = **5/mese**

Passaggio al Modello Relazionale

3.1 **Traduzione delle entità**

Cominciamo col tradurre l'entità principale, **Wine**.

Wine(wine, vintage, wineryId↑, winefamilyId↑, availability, price)

Winery resta, invece, invariata:

Winery(wineryId, winery, address, telephone)

Ora **Wine** ha 2 foreign keys wineryId e winefamilyId che descrivono la relazione 1:1 rispettivamente con **Winery** e **Winefamily**.

Seguiamo lo stesso processo per l'entità **Winefamily**, rispettiamo l'associazione 1:1 con **Winecolor**, **Winetype** e **Winedenom**.

Winefamily(winefamilyId, winefamily, winecolorId↑, winetypeId↑, winedenomId↑, regionId↑)

Le entità correlate restano invece invariate:

Winetype(winetypeId, winetype)
Winecolor(winecolorId, winecolor)
Winedenom(winedenomId, winedenom)

Traduciamo ora la relazione **M:M** tra **Wine** e **Winegrape**, per fare ciò introduciamo una nuova entità **WineWinegrape** che si occuperà di gestire l'associazione e gli attributi:

Winegrape(winegrapeId, winegrape)

WineWinegrape(winegrapeId↑, wine↑, vintage↑, percentage)

Come ultime entità da tradurre restano **Region** e **Country**:

Region(regionId, region, countryId↑)

Country(countryId, country)

Abbiamo quindi aggiunto a **Region** un nuovo campo countryId, Foreign Key che fa riferimento a **Country**, passiamo ora alla **Normalizzazione** del nostro nuovo schema.

3.2 **Normalizzazione dello schema**

Prima di implementare la base di dati, dobbiamo assicurarci che sia nella 3ª forma normale (3NF) e se non lo è, decomporre i vincoli e normalizzarla.

1NF - La prima forma normale richiede che il dominio di un attributo di uno schema di relazione sia composto di **soli valori atomici** e che il valore di qualsiasi attributo di una tupla sia un valore singolo del dominio corrispettivo e **nel nostro caso è già soddisfatta**.

2NF - La seconda forma normale si basa sul concetto di dipendenza funzionale completa. Una DF X → Y di uno schema di relazione è una dipendenza funzionale completa (DFC) se la **rimozione di qualsiasi attributo A da X comporta che la DF non sussiste più**, nel caso in cui ciò non sia verificato si parla di dipendenza funzionale parziale (DFP). Uno schema di relazione si dice in 2NF se **ogni attributo non-primo** (un attributo non facente parte di nessuna chiave) dello schema **dipende in modo funzionale e**

completo da ogni chiave dello schema. Analizziamo il nostro schema allo stato attuale:

Wine(wine, vintage, wineryId↑, winefamilyId↑, availability, price)

Notiamo che tutti gli attributi di **Wine** dipendono solo dalle chiavi primarie wine e vintage, ovvero non possono esistere due vini aventi stesso nome e annata che sono stati prodotti da due aziende diverse, che sono di due famiglie diverse e di conseguenza prezzo e disponibilità diverse non ci sono dipendenze funzionali parziali.

Reiterando il processo tante volte quante sono le entità appena definite giungiamo facilmente alla conclusione che il nostro schema è in forma normale **2NF**.

3NF - La terza forma normale, a differenza della 2NF, si basa anche sulla dipendenza funzionale transitiva, andando ad eliminare ulteriori ridondanze. Una DF X → Y di uno schema di relazione è una dipendenza funzionale transitiva (DFT) se **esiste Z**, sottoinsieme di attributi dello schema, **che non è né chiave candidata né sottoinsieme di una chiave dello schema, tale per cui valgono contemporaneamente X → Z ed Z → Y.**

Data questa definizione e il nostro schema, non essendoci dipendenze funzionali transitive, possiamo benissimo affermare che il nostro schema è anche in forma normale **3NF**. Non è necessario quindi un ulteriore processo di normalizzazione.

Implementazione in MySQL

4.1 **Generazione della base di dati**

Il nostro schema definitivo (in forma relazionale) è il seguente:

Wine(wine, vintage, wineryId↑, winefamilyId↑, availability, price)

Winery(wineryId, winery, address, telephone*)*

Winefamily(winefamilyId, winefamily, winecolorId↑, winetypeId↑, winedenomId↑, regionId↑)

Winetype(winetypeId, winetype)

Winecolor(winecolorId, winecolor)

Winedenom(winedenomId, winedenom)

Winegrape(winegrapeId, winegrape)

WineWinegrape(winegrapeId↑, wine↑, vintage↑, percentage)

Region(regionId, region, countryId↑)

Country(countryId, country)

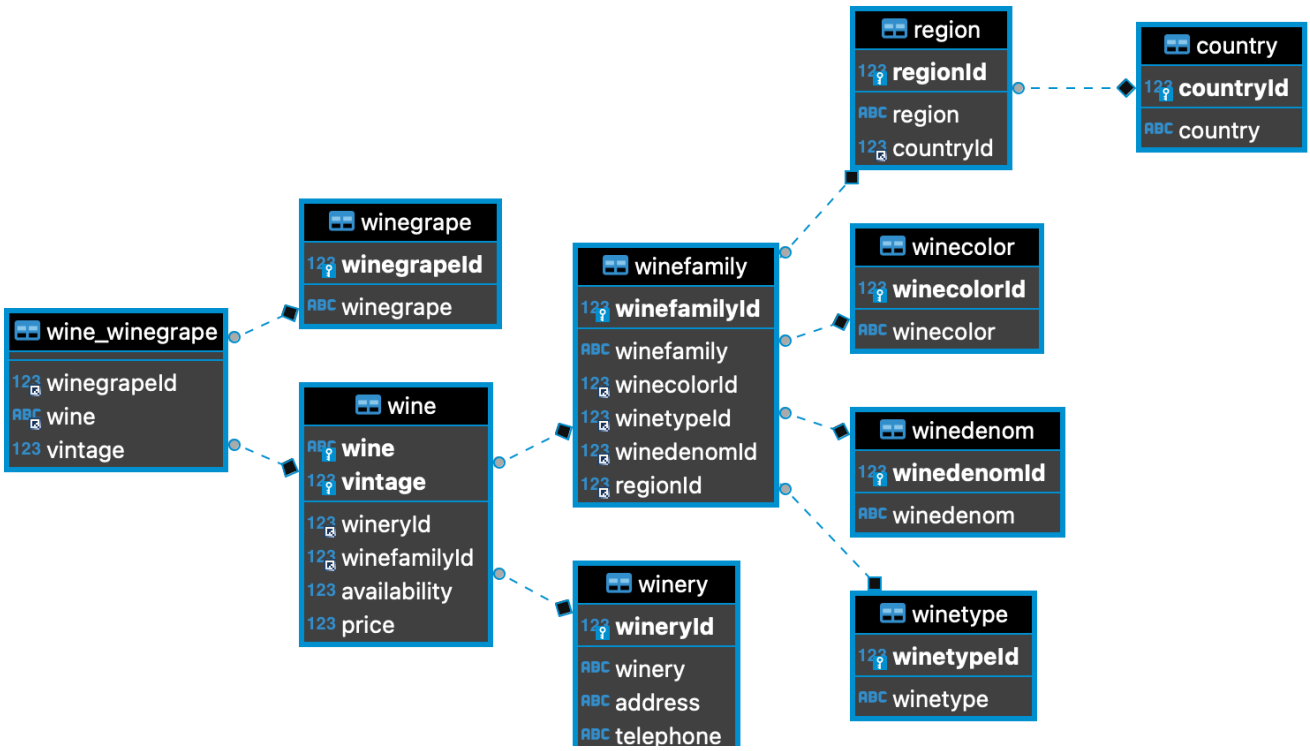
Di seguito tutte le istruzioni sul utilizzate per creare la base di dati, insieme a questo documento, sono forniti anche i file **.sql**.

```
CREATE DATABASE IF NOT EXISTS `enoteca`;
USE enoteca;

DROP USER IF EXISTS 'gestore_enoteca';
CREATE USER 'gestore_enoteca' IDENTIFIED BY
'Password-Molto-Sicura';
GRANT ALL PRIVILEGES ON enoteca.* TO
'gestore_enoteca';

DROP TABLE IF EXISTS wine_winegrape;
DROP TABLE IF EXISTS wine;
DROP TABLE IF EXISTS winefamily;
DROP TABLE IF EXISTS winegrape;
DROP TABLE IF EXISTS winery;
DROP TABLE IF EXISTS winedenom;
```

```sql
DROP TABLE IF EXISTS winetype;
DROP TABLE IF EXISTS winecolor;
DROP TABLE IF EXISTS region;
DROP TABLE IF EXISTS country;


CREATE TABLE winery (
    wineryId INT NOT NULL AUTO_INCREMENT,
    winery VARCHAR(100) NOT NULL,
    address VARCHAR(100),
    telephone VARCHAR(30),
    PRIMARY KEY(wineryId)
);

CREATE TABLE winecolor (
    winecolorId INT NOT NULL AUTO_INCREMENT,
    winecolor VARCHAR(10) NOT NULL,
    PRIMARY KEY(winecolorId)
);

CREATE TABLE winetype (
    winetypeId INT NOT NULL AUTO_INCREMENT,
    winetype VARCHAR(30) NOT NULL,
    PRIMARY KEY(winetypeId)
);

CREATE TABLE winedenom (
    winedenomId INT NOT NULL AUTO_INCREMENT,
    winedenom VARCHAR(10) NOT NULL,
    PRIMARY KEY(winedenomId)
);

CREATE TABLE country (
    countryId INT NOT NULL AUTO_INCREMENT,
    country VARCHAR(30) NOT NULL,
    PRIMARY KEY(countryId)
);

CREATE TABLE region (
    regionId INT NOT NULL AUTO_INCREMENT,
    region VARCHAR(30) NOT NULL,
    countryId INT NOT NULL,
    PRIMARY KEY(regionId),
```

```
    FOREIGN KEY(countryId) REFERENCES
country(countryId) ON DELETE CASCADE
);

CREATE TABLE winefamily (
    winefamilyId INT NOT NULL AUTO_INCREMENT,
    winefamily VARCHAR(50) NOT NULL,
    winecolorId INT NOT NULL,
    winetypeId INT NOT NULL,
    winedenomId INT NOT NULL,
    regionId INT NOT NULL,
    PRIMARY KEY(winefamilyId),
    FOREIGN KEY(winecolorId) REFERENCES
winecolor(winecolorId) ON DELETE CASCADE,
    FOREIGN KEY(winetypeId) REFERENCES
winetype(winetypeId) ON DELETE CASCADE,
    FOREIGN KEY(winedenomId) REFERENCES
winedenom(winedenomId) ON DELETE CASCADE,
    FOREIGN KEY(regionId) REFERENCES
region(regionId) ON DELETE CASCADE
);

CREATE TABLE winegrape (
    winegrapeId INT NOT NULL AUTO_INCREMENT,
    winegrape VARCHAR(30) NOT NULL,
    PRIMARY KEY(winegrapeId)
);

CREATE TABLE wine (
    wine VARCHAR(100) NOT NULL,
    vintage INT UNSIGNED NOT NULL,
    wineryId INT NOT NULL,
    winefamilyId INT NOT NULL,
    availability INT NOT NULL,
    price INT UNSIGNED NOT NULL,
    PRIMARY KEY(wine, vintage),
    FOREIGN KEY(wineryId) REFERENCES
winery(wineryId) ON DELETE CASCADE,
    FOREIGN KEY(winefamilyId) REFERENCES
winefamily(winefamilyId) ON DELETE CASCADE
);
```

```
CREATE TABLE wine_winegrape (
    winegrapeId INT NOT NULL,
    wine VARCHAR(100) NOT NULL,
    vintage INT UNSIGNED NOT NULL,
    percentage INT UNSIGNED NOT NULL,
    FOREIGN KEY(wine, vintage) REFERENCES
wine(wine, vintage) ON DELETE CASCADE,
    FOREIGN KEY(winegrapeId) REFERENCES
winegrape(winegrapeId) ON DELETE CASCADE
);
```



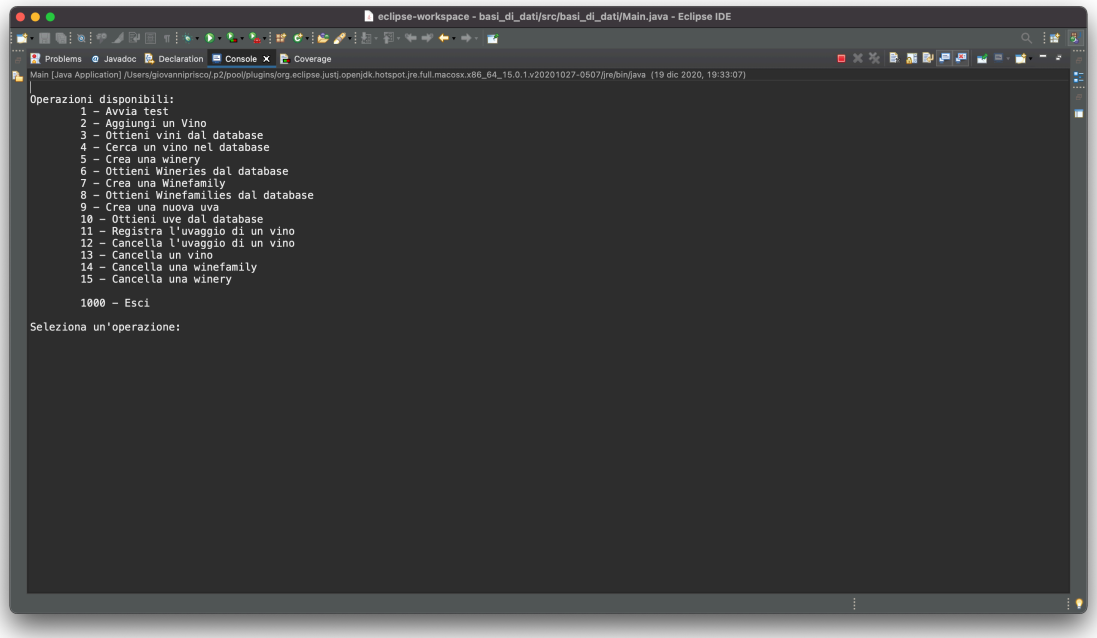
L'esecuzione di queste istruzioni porta alla generazione di una base di dati avente lo schema di cui sopra.

Insieme a questo documento, oltre ai file **.sql** per la generazione del database e il suo popolamento, è stata fornita un'applicazione da linea di comando sviluppata in Java che permette di interagire con la nostra base di dati.

Applicazione Java

5.1 **Quick tour**

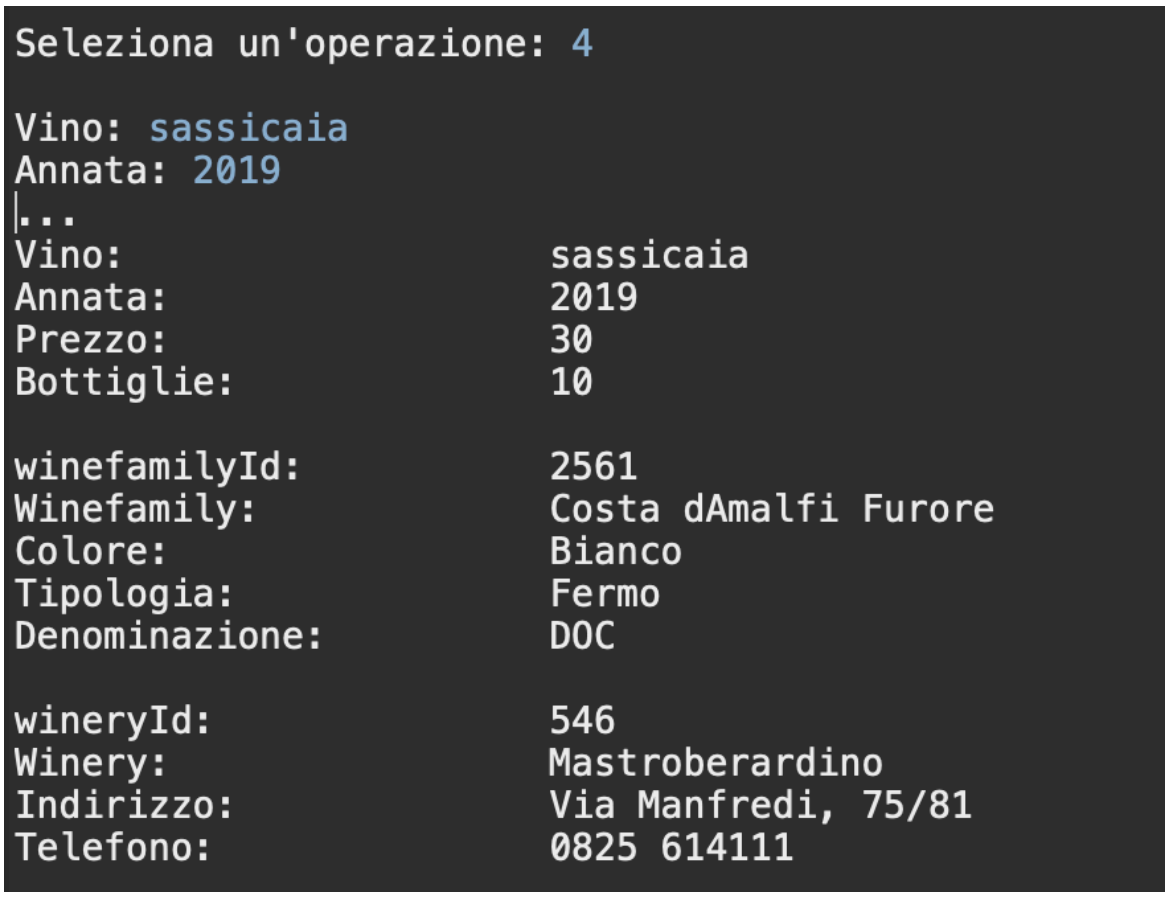
All'avvio l'applicazione si presenterà con una serie di operazioni che è possibile eseguire sulle varie tabelle del database implementato nel capitolo precedente.



Di seguito gli screenshot di alcune delle 15 operazioni per dare un'idea del flusso:

```
Seleziona un'operazione: 2

Nome del vino da aggiungere: sassicaia

Annata: 2019

Quante bottiglie? 10

Prezzo: 30

Winefamily ID: 2561

Winery ID: 546
...

[*] Vino creato con successo
```

Si è scelta l'operazione 2, aggiungere un vino. L'applicazione chiederà quindi i dati necessari alla creazione del vino.



In questo caso si è scelta l'operazione n.4, cercare un vino nel database.

Esattamente come prima, verrà chiesto all'utente di specificare il nome del vino e la sua annata per cercare il vino avente come chiave composta nome-annata. Una volta ottenuto il vino, esso viene stampato con tutte le sue caratteristiche, e se ha un uvaggio registrato, verrà stampato anche quello, come si può vedere nello screenshot a seguire:

```
Seleziona un'operazione: 4

Vino: fiano di avellino "béchar"
Annata: 2019
...
Vino:                   fiano di avellino "béchar"
Annata:                 2019
Prezzo:                 10
Bottiglie:              1

winefamilyId:           121
Winefamily:             Fiano di Avellino
Colore:                 Bianco
Tipologia:              Fermo
Denominazione:          DOCG

wineryId:               953
Winery:                 Antonio Caggiano
Indirizzo:              Contrada Sala
Telefono:               0827 74723

Uvaggio:
        #96 - Fiano (100%)
```