# Laboratorio di Basi di Dati: *Progetto “Collectors”*

**Gruppo di lavoro**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Matricola** | **Nome** | **Cognome** | **Contributo al progetto** |
| 278661 | Luca Francesco | Macera | Diagramma ER, diagramma ER ristrutturato, modello relazionale, realizzazione del database, query 1 3 4 5 6, diversi trigger |
| 279660 | Michael | Piccirilli | Diagramma ER, diagramma ER ristrutturato, realizzazione del database, documentazione, query 10 11,trigger |
| 278844 | Calogero | Carlino | Diagramma ER, diagramma ER ristrutturato, modello relazionale, realizzazione del database, query 2 7 8 9 12 |

**Data di consegna del progetto**: gg/mm/2023

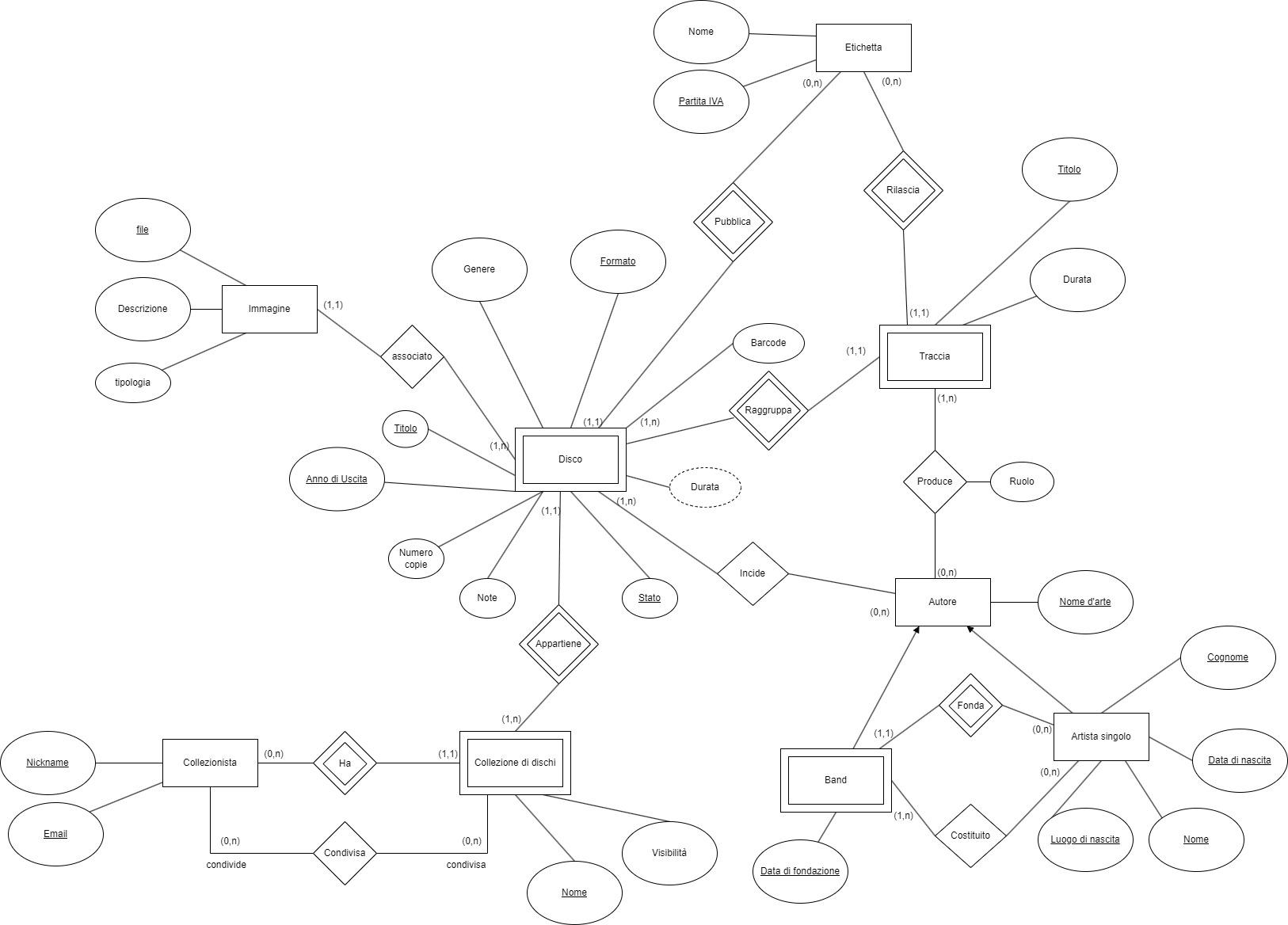
## Analisi dei requisiti

Il caso d’uso rappresenta una situazione in cui si vuole tenere traccia delle **collezioni** relative ad un **collezionista**. Tali collezioni rappresentano un gruppo di **dischi** , i quali presentano ognuno una tracklist formata di **tracce,** delle **immagini** e ulteriori informazioni quali, ad esempio, gli **autori**.

**Dominio:**

* **Collezionista**: persona identificata univocamente da un nickname e da un indirizzo email che crea delle collezioni che possono essere anche condivise con altri collezionisti.
* **Collezione:** gruppo di dischi che può, o meno, essere condivisa(visualizzabile) con altri collezionisti.
* **Disco**: rappresenta in tutto e per tutto il concetto di disco, con i suoi autori, un suo titolo, l’anno di uscita, l’etichetta, dei generi, lo stato di conservazione, il formato, il barcode (il quale potrebbe essere anche assente).
* **Immagine:** relativa ad un disco, può essere la copertina, il retro o ulteriori tipologie.
* **Etichetta**: rappresenta l’azienda che distribuisce e pubblicizza un disco, identificata da una partita IVA e da un nome
* **Traccia:** o canzone, presente all’interno del disco, con un suo titolo, la durata ed altre informazioni come gli autori e/o gli esecutori e/o i compositori
* **Autore:** rappresenta colui che compone e/o esegue una traccia o, per quanto riguarda il disco, colui che lo ha pubblicato

## Progettazione concettuale



* **Genere** rappresenta un attributo multivalore dal momento che un disco può avere più di un genere.
* **Durata** (di un disco) è un attributo calcolato sulla base della somma delle durate di ogni traccia presente nel disco stesso.
* La **visibilità** rappresenta i due stati in cui può trovarsi una collezione:

1. *privato* - ciò significa che una determinata collezione che appartiene ad un collezionista è visibile solo a lui o eventualmente anche agli altri collezionisti con cui è condivisa
2. *pubblica* - ciò significa che la collezione è visibile a tutti i collezionisti

### Formalizzazione dei vincoli non esprimibili nel modello ER

* **Collezionista:** *nickname* ed *email* sono chiavi primarie dunque devono essere **UNIQUE** e **NOT NULL.**
* **Collezioni di dischi:** *nome* e gli attributi della relazione *ha* (collezionista) sono chiave primaria dunque devono essere **UNIQUE** e **NOT NULL.** Il flag *visibilità* non può essere **NULL**, può avere solo *privata/pubblica (0 o 1)* come valori.
* **Disco:** *titolo*, *anno di uscita, formato, stato* e la relazione *appartiene*(collezione) e *pubblica* (etichetta) sono la chiave primaria dunque devono essere **UNIQUE** e **NOT NULL.** Barcode è **UNIQUE** dal momento che deve essere univoco. Il numero di copie non può essere **NULL**(di **DEFAULT** è 1).
* **Tracce:** *titolo* e gli attributi della relazione *rilascia*(etichetta) e *raggruppa*(disco) sono la chiave primaria dunque devono essere **UNIQUE** e **NOT NULL.** *Durata* non può essere **NULL.**
* **Immagine:** *file* è chiave primaria dunque deve essere **UNIQUE** e **NOT NULL.** L’attributo *tipologia* non può essere **NULL.**
* **Etichetta:** *Partita\_IVA* è chiave primaria dunque deve essere **UNIQUE** e **NOT NULL.**
* **Band:** *Data di fondazione* e gli attributi della relazione *fonda*(artista singolo) sono chiave primaria dunque devono essere **UNIQUE** e **NOT NULL.**
* **Artista Singolo:** *nome*, *cognome*, *luogo di nascita*, *data di nascita* sono chiave primaria dunque devono essere **UNIQUE** e **NOT NULL.**
* **Autore:** *nome d’arte* è chiave primaria, dunque deve essere **UNIQUE** e **NOT NULL.**

## Progettazione logica

### Ristrutturazione ed ottimizzazione del modello ER

Immagine che contiene disegno, schizzo, diagramma, modello

Descrizione generata automaticamente

Si è scelto di dividere l’entità Disco in due entità separate (**Disco** ed **Info\_Disco**) per fare in modo che questa non fosse troppo carica di attributi. Inoltre, tale divisione è avvenuta in virtù di quegli attributi meno importanti come, appunto, le *note***,** il *barcode* e il *numero di copie*.

Le generalizzazioni presenti nel modello ER sono state eliminate in favore di due nuove relazioni: **is\_band** e **is\_singolo** tra **Autore**, **Band** e **Artista Singolo** per fare in modo di tenere separate le tre entità senza sovraccaricare le entità figlie o padre. Inoltre, si è scelto di aggiungere un attributo *tipo* ad **Autore** in modo da ricavare subito se i dati di un autore salvato riferiscono ad un artista singolo o ad una band. Questo potrebbe anche essere esteso nel caso in cui si volessero modellare altri tipi di autore oltre ai due citati precedentemente.

Sono state aggiunte nuove entità: **Genere, Formato, Tipologia, Stato, Ruolo.** Queste nuove entità servono per poter salvare dati selezionabili da liste preimpostate , come i generi di un disco, il formato in cui essi sono salvati o il ruolo di un autore nella produzione di un brano. L’uso di queste entità permette maggiore flessibilità della base di dati nel caso in cui fosse necessaria la manipolazione di questi dati.

### Traduzione del modello ER nel modello relazionale

**Collezionista**(ID, nickname, email)

**Collezione\_di\_dischi**(ID , nome,visibilità, *ID\_collezionista*)

**Disco**(ID, titolo, anno\_di\_uscita, nome\_formato, nome\_stato, *ID\_etichetta*, *ID\_collezione\_di\_dischi*)

**Info\_disco**(*ID\_disco*, barcode, note, numero\_copie)

**Etichetta**(ID, nome, partita\_IVA)

**Genere**(nome)

**Formato**(nome)

**Ruolo**(nome)

**Stato**(nome)

**Tipologia**(nome)

**Immagine**(ID, file, descrizione, nome\_tipologia, *ID\_disco*)

**Traccia**(ID, titolo, durata, *ID\_etichetta*, *ID\_disco*)

**Autore**(ID, nome\_darte, tipo)

**Artista\_Singolo**(ID, nome, cognome, data\_di\_nascita, luogo\_di\_nascita, *ID\_autore*)

**Band**(ID, data\_di\_fondazione, *ID\_fondatore*, *ID\_autore*)  
**Condivisa**(*ID\_collezione\_di\_dischi*, *ID\_collezionista*)

**Classificazione**(*ID\_disco, nome\_genere*)

**Produce**(*ID\_traccia, ID\_autore, nome\_ruolo*)

**Costituito**(*ID\_band, ID\_membro*)

**Incide**(*ID\_disco, ID\_autore*)

**\***Le primary key sono marcate in underline mentre le foreign key in *italic*

## Progettazione fisica

### Implementazione del modello relazionale

*Script sql per la creazione della struttura della base di dati estratto dal file “DDL.sql”*

#Inizializzazione database

drop database if exists collectors;

create database collectors;

use collectors;

#Tabelle

drop table if exists incide;

drop table if exists classificazione;

drop table if exists produce;

drop table if exists costituito;

drop table if exists condivisa;

drop table if exists band;

drop table if exists artista\_singolo;

drop table if exists traccia;

drop table if exists immagine;

drop table if exists info\_disco;

drop table if exists disco;

drop table if exists collezione\_di\_dischi;

drop table if exists collezionista;

drop table if exists autore;

drop table if exists etichetta;

drop table if exists tipologia;

drop table if exists stato;

drop table if exists ruolo;

drop table if exists formato;

drop table if exists genere;

create table genere(

nome varchar(50),

constraint NOME\_genere primary key(nome)

);

create table formato(

nome varchar(50),

constraint NOME\_formato primary key(nome)

);

create table ruolo(

nome varchar(50),

constraint NOME\_ruolo primary key(nome)

);

create table stato(

nome varchar(50),

constraint NOME\_stato primary key(nome)

);

create table tipologia(

nome varchar(50),

constraint NOME\_tipologia primary key(nome)

);

create table etichetta(

id integer unsigned auto\_increment,

nome varchar(200) not null,

partitaIVA varchar(11) not null,

constraint ID\_etichetta primary key(id),

constraint unique\_partitaIVA unique(partitaIVA)

);

create table autore(

id integer unsigned auto\_increment,

nome\_darte varchar(100) not null,

tipo boolean not null,

constraint ID\_autore primary key(id),

constraint unique\_autore unique(nome\_darte)

);

create table collezionista(

id integer unsigned auto\_increment,

nickname varchar(100) not null,

email varchar(200) not null,

constraint ID\_collezionista primary key(id),

constraint unique\_collezionista unique(nickname, email)

);

create table collezione\_di\_dischi(

id integer unsigned auto\_increment,

nome varchar(200) not null,

visibilita boolean not null,

id\_collezionista integer unsigned not null,

constraint ID\_collezione primary key(id),

constraint collezione\_collezionista foreign key(id\_collezionista) references collezionista(id) on delete cascade,

constraint unique\_collezione unique(id\_collezionista,nome)

);

create table disco(

id integer unsigned auto\_increment,

titolo varchar(50) not null,

anno\_di\_uscita date not null,

nome\_formato varchar(50) not null,

nome\_stato varchar(50) not null,

id\_etichetta integer unsigned not null,

id\_collezione\_di\_dischi integer unsigned not null,

constraint ID\_disco primary key(id),

constraint disco\_formato foreign key(nome\_formato) references formato(nome) on delete restrict on update cascade,

constraint disco\_etichetta foreign key(id\_etichetta) references etichetta(id) on delete restrict,

constraint disco\_stato foreign key(nome\_stato) references stato(nome) on delete restrict on update cascade,

constraint disco\_collezione foreign key(id\_collezione\_di\_dischi) references collezione\_di\_dischi(id) on delete cascade,

constraint unique\_disco unique(titolo,anno\_di\_uscita,nome\_formato,nome\_stato, id\_etichetta,id\_collezione\_di\_dischi)

);

create table info\_disco(

id\_disco integer unsigned,

barcode varchar(200),

note text,

numero\_copie integer unsigned default 1,

constraint ID\_info\_disco primary key(id\_disco),

constraint unique\_barcode unique(barcode),

constraint info\_disco\_disco foreign key(id\_disco) references disco(id) on delete cascade

);

create table immagine(

id integer unsigned auto\_increment,

path varchar(100) not null,

descrizione text,

id\_disco integer unsigned not null,

nome\_tipologia varchar(50) not null,

constraint ID\_immagine primary key(id),

constraint unique\_immagine unique(id\_disco,path,nome\_tipologia),

constraint immagine\_disco foreign key(id\_disco) references disco(id) on delete cascade,

constraint immagine\_tipologia foreign key(nome\_tipologia) references tipologia(nome) on delete restrict on update cascade

);

create table traccia(

id integer unsigned auto\_increment,

titolo varchar(50) not null,

durata decimal(10,2) unsigned default 0.0,

id\_etichetta integer unsigned not null,

id\_disco integer unsigned not null,

constraint ID\_traccia primary key(id),

constraint traccia\_etichetta foreign key(id\_etichetta) references etichetta(id) on delete restrict,

constraint traccia\_disco foreign key(id\_disco) references disco(id) on delete cascade,

constraint unique\_traccia unique(titolo,id\_etichetta,id\_disco)

);

create table artista\_singolo(

id integer unsigned auto\_increment,

nome varchar(50) not null,

cognome varchar(50) not null,

data\_nascita date not null,

luogo\_nascita varchar(50) not null,

id\_autore integer unsigned not null,

constraint ID\_artista primary key(id),

constraint artista\_singolo\_autore foreign key(id\_autore) references autore(id) on delete cascade,

constraint unique\_artista\_singolo unique(nome,cognome,data\_nascita,luogo\_nascita),

constraint unique\_autore\_artista\_singolo unique(id\_autore)

);

create table band(

id integer unsigned auto\_increment,

id\_fondatore integer unsigned not null,

data\_fondazione date not null,

id\_autore integer unsigned not null,

constraint ID\_band primary key(id),

constraint band\_artista\_singolo foreign key(id\_fondatore) references artista\_singolo(id) on delete restrict,

constraint band\_autore foreign key(id\_autore) references autore(id) on delete cascade,

constraint unique\_band unique(id\_fondatore,data\_fondazione),

constraint unique\_autore\_band unique(id\_autore)

);

# Relazioni

create table condivisa(

id\_collezionista integer unsigned not null,

id\_collezione integer unsigned not null,

constraint primary\_condivisione primary key(id\_collezionista,id\_collezione),

constraint condivisa\_collezionista foreign key(id\_collezionista) references collezionista(id) on delete cascade,

constraint condivisa\_collezione foreign key(id\_collezione) references collezione\_di\_dischi(id) on delete cascade

);

create table costituito(

 id\_band integer unsigned not null,

 id\_artista integer unsigned not null,

 constraint primary\_costituito primary key(id\_band,id\_artista),

 constraint costituito\_band foreign key(id\_band) references band(id) on delete cascade,

 constraint costituito\_artista\_singolo foreign key(id\_artista) references artista\_singolo(id) on delete restrict

);

create table produce(

id\_traccia integer unsigned not null,

id\_autore integer unsigned not null,

nome\_ruolo varchar(50) not null,

constraint primary\_produce primary key(id\_traccia,id\_autore,nome\_ruolo),

constraint produce\_traccia foreign key(id\_traccia) references traccia(id) on delete cascade,

constraint produce\_autore foreign key(id\_autore) references autore(id) on delete restrict,

constraint produce\_ruolo foreign key(nome\_ruolo) references ruolo(nome) on delete restrict on update cascade

);

create table classificazione(

nome\_genere varchar(50) not null,

id\_disco integer unsigned not null,

constraint primary\_classificazione primary key(nome\_genere,id\_disco),

constraint classificazione\_genere foreign key(nome\_genere) references genere(nome) on delete restrict on update cascade,

constraint classificazione\_disco foreign key(id\_disco) references disco(id) on delete cascade

);

create table incide(

id\_disco integer unsigned not null,

id\_autore integer unsigned not null,

constraint primary\_raggruppamento primary key(id\_disco,id\_autore),

constraint incide\_disco foreign key(id\_disco) references disco(id) on delete cascade,

constraint incide\_autore foreign key(id\_autore) references autore(id) on delete restrict

);

*È presente anche uno script per il popolamento delle tabelle chiamato “data.sql”*

### Implementazione dei vincoli

*Trigger che controlla l’inserimento di un artista singolo.*

DROP TRIGGER IF EXISTS check\_artista;

DELIMITER $

CREATE TRIGGER check\_artista BEFORE INSERT ON artista\_singolo FOR EACH ROW

BEGIN

  DECLARE tipo\_autore boolean;

  SET tipo\_autore = (

      SELECT tipo

      FROM autore

      WHERE id = new.id\_autore);

/\* Controlla che il tipo dichiarato nell'inserimento dell’autore combaci con quello dell'artista che si sta inserendo (artista singolo o band)\*/

      IF(tipo\_autore = true) THEN

SIGNAL SQLSTATE "45000" SET MESSAGE\_TEXT="Il tipo dell'autore non combacia.";

      END IF;

      /\* Controllo sulla data di nascita affinchè non ecceda quella corrente\*/

      IF(new.data\_nascita > date(now())) THEN

        SIGNAL SQLSTATE "45000" SET MESSAGE\_TEXT="Errore nella data di nascita.";

      END IF;

END$

*Trigger che controlla l’inserimento di una band.*

DROP TRIGGER IF EXISTS check\_band;

DELIMITER $

CREATE TRIGGER check\_band BEFORE INSERT ON band FOR EACH ROW

BEGIN

  DECLARE tipo\_autore boolean;

  SET tipo\_autore = (

      SELECT tipo

      FROM autore

      WHERE id = new.id\_autore);

  /\* Controlla che il tipo dichiarato nell'inserimento dell’autore combaci con

  quello dell'artista che si sta inserendo (artista singolo o band)\*/

  IF(tipo\_autore = false) THEN

    SIGNAL SQLSTATE "45000" SET MESSAGE\_TEXT="Il tipo dell'autore non combacia";

  END IF;

  /\* Controllo sulla data di nascita affinchè non ecceda quella corrente\*/

  IF(new.data\_fondazione > date(now())) THEN

    SIGNAL SQLSTATE "45000" SET MESSAGE\_TEXT="Errore nella data di nascita.";

  END IF;

END$

DELIMITER ;

*Trigger che controlla che ogni disco abbia associato almeno un genere*

DROP TRIGGER IF EXISTS controlla\_generi;

DELIMITER $

CREATE TRIGGER controlla\_generi BEFORE DELETE ON classificazione FOR EACH ROW

BEGIN

    DECLARE numero\_generi integer;

    /\* Controlla che un disco non rimanga mai con meno di 1 genere \*/

    SET numero\_generi = (

        SELECT count(distinct c.nome\_genere)

        FROM classificazione c

        WHERE c.id\_disco=old.id\_disco);

    IF(numero\_generi <= 1) THEN

        SIGNAL SQLSTATE "45000" SET MESSAGE\_TEXT="Operazione vietata!";

    END IF;

END$

DELIMITER ;

*Trigger che controlla che la visibilità di una collezione non rimanga in uno stato inconsistente (non possono esistere collezioni pubbliche e condivise allo stesso momento)*

DROP TRIGGER IF EXISTS on\_update\_visibilita;

DELIMITER $

CREATE TRIGGER on\_update\_visibilita BEFORE UPDATE ON collezione\_di\_dischi FOR EACH ROW

BEGIN

    IF(old.visibilita = false AND new.visibilita = true) THEN

        DELETE FROM condivisa WHERE id\_collezione = old.id;

  END IF;

END$

DELIMITER ;

### Implementazione funzionalità richieste

* Riportate qui il **codice che implementa tutte le funzionalità richieste**, che si tratti di SQL o di pseudocodice o di entrambi. *Il codice di ciascuna funzionalità dovrà essere preceduto dal suo numero identificativo e dal testo della sua definizione*, come riportato nella specifica.
* Se necessario, riportate anche il codice delle procedure e/o viste di supporto.

#### Funzionalità 1

Definizione come da specifica

CODICE

#### Funzionalità 2

Definizione come da specifica

CODICE

## Interfaccia verso il database

* Opzionalmente, se avete deciso di realizzare anche una **(semplice) interfaccia** (a linea di comando o grafica) in un linguaggio di programmazione a voi noto (Java, PHP, …) che manipoli il vostro database , dichiaratelo in questa sezione, elencando le tecnologie utilizzate e le funzionalità invocabili dall’interfaccia.
* Il relativo codice sorgente dovrà essere *allegato* alla presente relazione.

**Raccomandazioni finali**

* Questo documento è un modello che spero possa esservi utile per scrivere la documentazione finale del vostro progetto di Laboratorio di Basi di Dati.
* Cercate di includere tutto il codice SQL nella documentazione, come indicato in questo modello, per facilitarne la correzione. Potete comunque allegare alla documentazione anche il *dump* del vostro database o qualsiasi altro elemento che ritenete utile ai fini della valutazione.
* Ricordate che la documentazione deve essere consegnata, anche per email, almeno *una settimana prima* della data prevista per l’appello d’esame. Eventuali eccezioni a questa regola potranno essere concordate col docente.