**Progetto di Basi di Dati**

**-Piattaforma di prenotazione viaggi in Bus per eventi-**

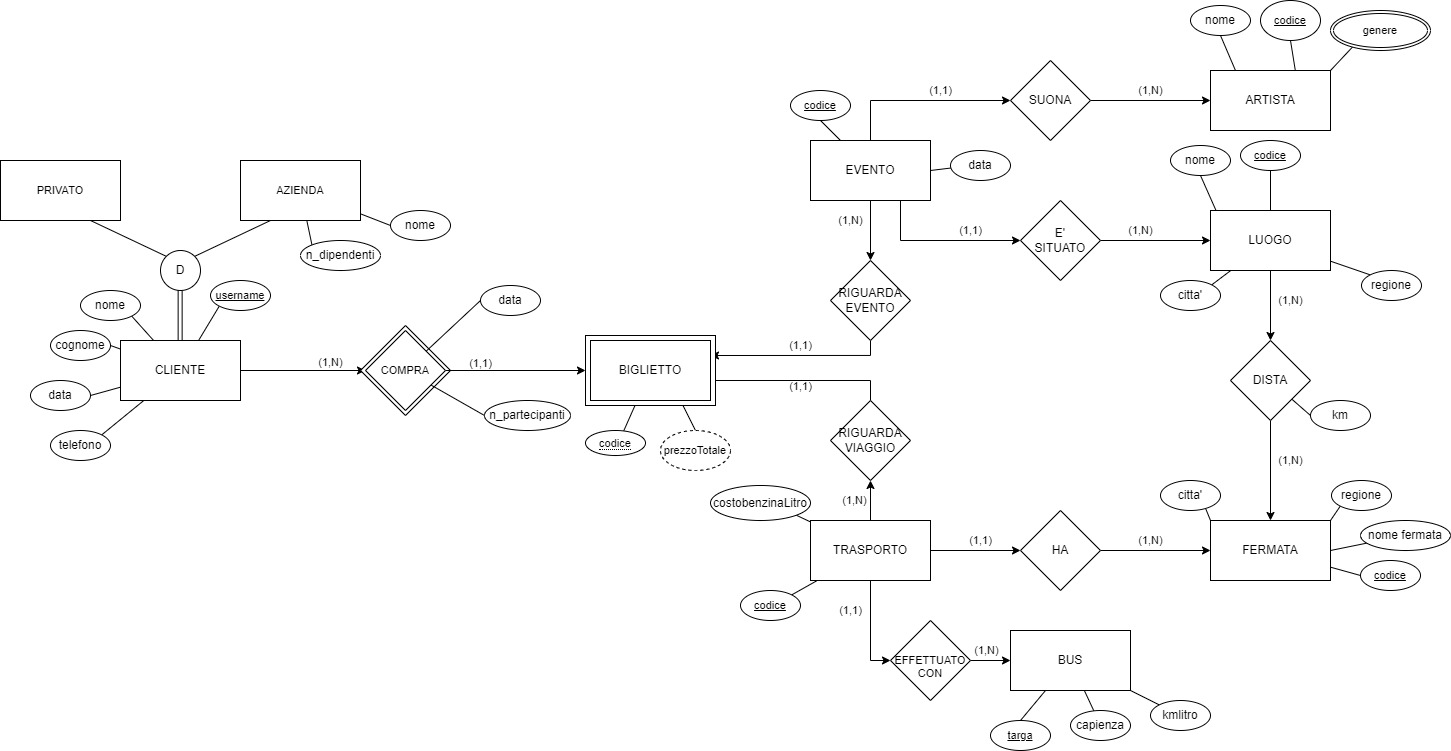
**1. Raccolta delle specifiche della realtà d’interesse  
Descrizione**  
Si vuole progettare una base di dati per la gestione di una **piattaforma di prenotazione viaggi in Bus per eventi.**  
L’accesso alla piattaforma si effettua tramite la creazione di un Account Cliente, identificato univocamente da un username. Inoltre, di un cliente vogliamo sapere anche il nome, cognome, data di nascita, numero telefonico.  
Un cliente può essere un privato o un’azienda. Di un’azienda teniamo registrati il numero dei dipendenti e il nome dell’azienda, nel caso voglia prenotare un viaggio in bus per il numero dei suoi dipendenti.  
Un cliente compra in una certa data e con un certo numero di partecipanti un biglietto, identificato univocamente da un codice, per il suo viaggio.  
Il biglietto riguarda sia un evento e sia il trasporto. L’evento, identificato univocamente da un codice, memorizza anche la data, l’artista e il luogo del suddetto evento. Di un’artista memorizziamo un codice identificativo, il nome e il proprio genere. Di un luogo memorizziamo un codice, il nome del luogo, la città e la regione.  
Di un trasporto memorizziamo un codice univoco e il costo della benzina al litro attuale in base al bus e la fermata. Della fermata vogliamo memorizzare un codice, la città, la regione e il nome. Mentre di un bus memorizziamo la targa che la identifica, la capienza dei posti e i km al litro che può percorrere.  
Memorizziamo in oltre la distanza in km tra il luogo del concerto e la fermata.

Specifiche della realtà d’interesse  
Questa piattaforma offre l’opportunità di raggiungere gli eventi senza doversi preoccupare del trasporto.   
Il servizio è rivolto a tutti i frequentatori di eventi che vivono distanti dalle città che ospitano concerti e che non vogliono mancare.  
Il cliente compra un biglietto del bus dove avrà la scelta tra i vari artisti nei vari luoghi e date, e sceglierà la fermata più consona per lui.   
Avremo la distanza fra ogni luogo e ogni fermata

**Glossario dei termini**

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| **Cliente** | Account Cliente che può comprare un biglietto per il bus per il raggiungimento di un evento a sua scelta |
| **Privato** | Una sottoclasse di cliente che acquista biglietti privatamente |
| **Azienda** | Una sottoclasse di cliente che acquista biglietti a nome di un’azienda |
| **Biglietto** | E’ il biglietto del bus, una ricevuta di pagamento che dimostra l’acquisto per il viaggio e ne da l’accesso |
| **Evento** | L’evento puo’ effettuarsi in qualsiasi tipo di spazio o struttura con gli adeguati requisiti per poterlo ospitare |
| **Trasporto** | Permette lo spostamento dei clienti alle loro destinazioni |
| **Artista** | Un creatore di opere dotate di valore estetico nei campi della musica collegato all’evento |
| **Luogo** | Sarà il posto in cui si esibira’ l’artista collegato all’evento |
| **Fermata** | La Fermata è il punto dove il bus si fermerà per far salire i clienti che avranno comprato il biglietto e li porteranno a destinazione |
| **Bus** | Un’autovettura portata per lunghi e piccole tratte e con un numero di persone e bagagli alto al servizio della piattaforma. |

**2. Progettazione concettuale della base di dati schema EER**

****Procedendo con la progettazione concettuale della base di dati, si ottiene il seguente schema EER

**Dizionario delle entità**

**Legenda:** sotto-entità, attributo multivalore, attributo ridondante, entità debole, chiave candidata

|  |  |  |  |
| --- | --- | --- | --- |
| Entità | Descrizione | Attributi | Identificatori |
| Cliente | Identità di un utente sulla piattaforma | * username * Nome * Cognome * data * telefono * dataNascita * password | username |
| Privato | Cliente privato che acquista un biglietto bus | \ | \ |
| Azienda | Cliente azienda che acquista un biglietto bus | * nome * n\_dipendenti | \ |
| Biglietto | Informazioni sul viaggio che il cliente ha bisogno | * Codice * prezzoTotale | codice |
| Evento | Costo dell’effettivo viaggio richiesto in base ai vari conducenti che possono effettuare il viaggio richiesto | * Codice * Data | Codice |
| Trasporto | Mappa dei luoghi dove sono possibili effettuare viaggi e dove si trovano i clienti e conducenti | * benzinaLitro * codice | codice |
| Artista | Sono i costi che deve affrontare un conducente in base alla sua autovettura | * codice * nome * genere | codice |
| Luogo | Autovettura che ha in possesso un conducente per effettuare i viaggi | * codice * nome * citta’ * regione | codice |
| Fermata | Tipo di Autovettura | * Citta * Regione * Nome fermata * codice | codice |
| Bus | Tipo di Autovettura | * targa * capienza * kmlitro | \ |

**Dizionario delle relazioni**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Relazione | | Descrizione | | Entità coinvolte | | Attributi | |
| compra | | Un cliente compra dei biglietti del bus, ed un biglietto è di un solo cliente | | Cliente(1,N)  biglietto(1,1) | | * data * n\_partecipanti | |
| riguarda evento | | Un biglietto riguarda un solo evento ed un evento può stare in piu’ biglietti | | biglietto(1,1)  evento(0,N) | | \ | |
| riguarda viaggio | | Un biglietto riguarda un solo viaggio e un viaggio puo’ stare in piu’ biglietti | | Trasporto(1,N)  biglietto(1,1) | | \ | |
| suona | | Ad un evento suona un solo artista ed un artista puo’ suonare a più eventi | | artista(1,N)  evento(1,1) | | \ | |
| e’ situato | | Un evento è situato in un solo luogo ed un luogo puo’ trovarsi in più eventi | | evento(1,1)  luogo(1,N) | | \ | |
| ha | | Un determinato trasporto ha una singola fermata ed una fermata sta in più trasporti | | trasporto(1,1)  fermata(1,N) | | \ | |
| Effettuato con | Un trasporto è effettuato con un singolo bus ed un bus puo’ trovarsi in più trasporti | | -trasporto(1,1)  -bus(1,N) | |  | |
| Dista | Un luogo dista da più fermata ed una fermata dista da più luoghi | | luogo(1,N)  Fermata(1,N) | | * km | |

**Vincoli non esprimibili nello schema**

Oltre ciò che è deducibile dallo schema EER, si tenga conto dei seguenti **vincoli**:

* L’attributo costobenzinalitro dell’entità trasporto deve essere il prezzo effettivo della benzina al litro che un determinato bus spende per una certa fermata
* L’attributo kmlitro dell’entità bus deve essere il noto dispendio del bus che spende per ogni km al litro

**3. Definizione delle procedure per la gestione della base di dati Tavola dei volumi**

Definiamo di seguito la tavola dei volumi della base di dati

SE=sub entity

|  |  |  |
| --- | --- | --- |
| Concetto | Tipo | Carico Applicativo |
| Cliente | E | 20 |
| Privato | SE | 15 |
| Azienda | SE | 5 |
| biglietto | E | 20 |
| evento | E | 10 |
| trasporto | E | 10 |
| artista | E | 5 |
| luogo | E | 5 |
| bus | E | 5 |
| fermata | E | 5 |
| compra | R | 20 |
| Riguarda evento | R | 10 |
| Riguarda viaggio | R | 10 |
| suona | R | 10 |
| E’ situato | R | 10 |
| Dista | R | 25 |
| ha | R | 10 |
| Effettuato con | R | 10 |

**Tavole delle operazioni**

Definisco di seguito la tavola delle operazioni per la gestione dei dati memorizzati nella base di dati

|  |  |  |  |
| --- | --- | --- | --- |
|  | Operazione | Tipo | Frequenza |
| 1 | Un nuovo cliente si registra | I | 5/mm |
| 2 | Un cliente compra un biglietto del bus e visualizza il prezzo | I | 10/mm |
| 3 | Dato un artista, le possibili date di concerti con città e regione e genere | I | 10/mm |
| 4 | Una azienda prenota e visualizza il prezzo del bus per il numero dei dipendenti | I | 5/aa |
| 5 | Visualizzazione per ogni cliente, il numero di ordini effettuati | B | 10/aa |
| 6 | Cancellazione di una prenotazione di un dato cliente | I | 2/mm |
| 7 | Modifica il costo della benzina al litro che paga una determinata macchina | I | 5/mm |

**4. Progettazione logica**

**Analisi delle ridondanze**

Il dato ridondante è l’attributo “PrezzoTotale” dell’entità Biglietto. Infatti,

sarebbe possibile ricavare il prezzo calcolando i dati del suo viaggio in bus.

Supponendo che l’attributo abbia un peso di 4 byte, essendo un normale

intero, e considerato che il volume dell’entità Biglietto è uguale a 20, il dato

andrebbe ad occupare uno spazio totale di circa 80 byte. Per decidere se

mantenere o meno il dato ridondante è necessario calcolare, per le operazioni che lo coinvolgono

**Tavola degli accessi**

**Operazione 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Con ridondanze** |  |  |  | **Senza ridonzanze** |  |  |
| **concetto** | **costrutto** | **tipo** | **accesso** | **concetto** | **costrutto** | **tipo** | **accesso** |
| evento | **E** | **L** | **1** | evento | **E** | **L** | **1** |
| trasporto | **E** | **L** | **1** | trasporto | **E** | **L** | **1** |
| cliente | **E** | **L** | **1** | cliente | **E** | **L** | **1** |
| biglietto | **E** | **L** | **1** | biglietto | **E** | **S** | **1** |
| biglietto | **E** | **S** | **1** | suona | **E** | **L** | **1** |
| suona | **R** | **L** | **1** | È situato | **R** | **L** | **1** |
| È situato | **R** | **L** | **1** | ha | **R** | **L** | **1** |
| ha | **R** | **L** | **1** | Effettuato  con | **R** | **L** | **1** |
| Effettuato con | **R** | **L** | **1** | dista | **R** | **L** | **1** |
| dista | **R** | **L** | **1** | Riguarda evento | **R** | **S** | **1** |
| Riguarda evento | **R** | **S** | **1** | Riguarda viaggio | **R** | **S** | **1** |
| Riguarda viaggio | **R** | **S** | **1** | Compra | **R** | **S** | **1** |
| Compra | **R** | **S** | **1** | \ | **\** | **\** | **\** |

TotaleConRidondanza=[1+1+1+1+1+1+1+1+1+(1+1+1+1)\*2)]\*10 = 170/mm

TotaleSenzaRidondanza=[1+1+1+1+1+1+1+1+((1+1+1+1)\*2)]\*10 =160a/mm

**Operazione 4**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Con ridondanze** |  |  |  | **Senza ridonzanze** |  |  |
| **concetto** | **costrutto** | **tipo** | **accesso** | **concetto** | **costrutto** | **tipo** | **accesso** |
| evento | **E** | **L** | **1** | evento | **E** | **L** | **1** |
| trasporto | **E** | **L** | **1** | trasporto | **E** | **L** | **1** |
| azienda | **E** | **L** | **1** | azienda | **E** | **L** | **1** |
| cliente | **E** | **L** | **1** | cliente | **E** | **L** | **1** |
| biglietto | **E** | **L** | **1** | biglietto | **E** | **S** | **1** |
| biglietto | **E** | **S** | **1** | suona | **E** | **L** | **1** |
| suona | **R** | **L** | **1** | È situato | **R** | **L** | **1** |
| È situato | **R** | **L** | **1** | ha | **R** | **L** | **1** |
| ha | **R** | **L** | **1** | Effettuato  con | **R** | **L** | **1** |
| Effettuato con | **R** | **L** | **1** | dista | **R** | **L** | **1** |
| dista | **R** | **L** | **1** | Riguarda evento | **R** | **S** | **1** |
| Riguarda evento | **R** | **S** | **1** | Riguarda viaggio | **R** | **S** | **1** |
| Riguarda viaggio | **R** | **S** | **1** | compra | **R** | **S** | **1** |
| compra | **R** | **S** | **1** | E’ | **R** | **L** | **1** |
| E’ | **R** | **L** | **1** | \ | **\** | **\** | **\** |

TotaleConRidondanza3=[1+1+1+1+1+1+1+1+1+1+1+((1+1+1+1)\*2)]\*10 = 190a/mm

TotaleSenzaRidondanza3=(1+1+1+1+1+1+1+1+1+1+((1+1+1+1)\*2))\*10 = 180a/mm

Somma con **ridondanze**=190+170=**260 a/mm + 70 byte**

Somma **senza ridondanza**= 180+160=**240 a/mm**

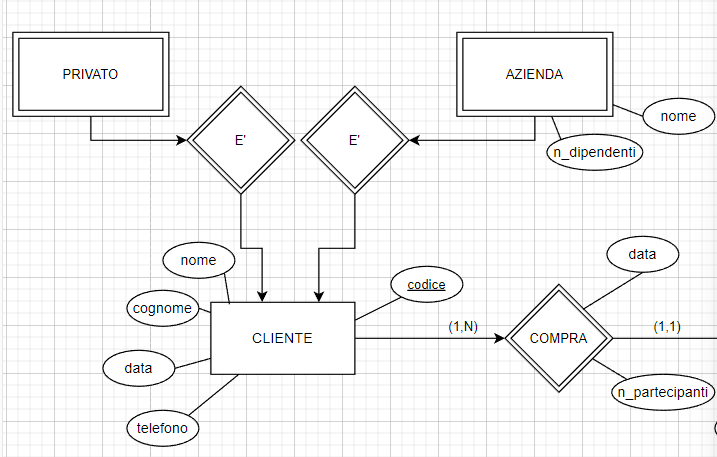
Ci conviene **eliminare la ridondanza**

**Eliminazione delle gerarchie**

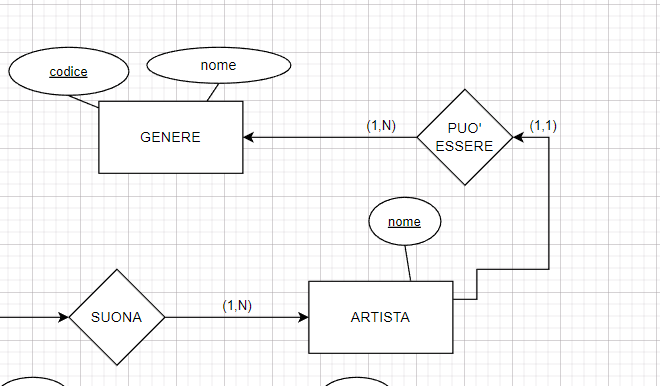
Nello schema inizialmente elaborato, è presente la seguente specializzazione dell’entità ‘cliente’:

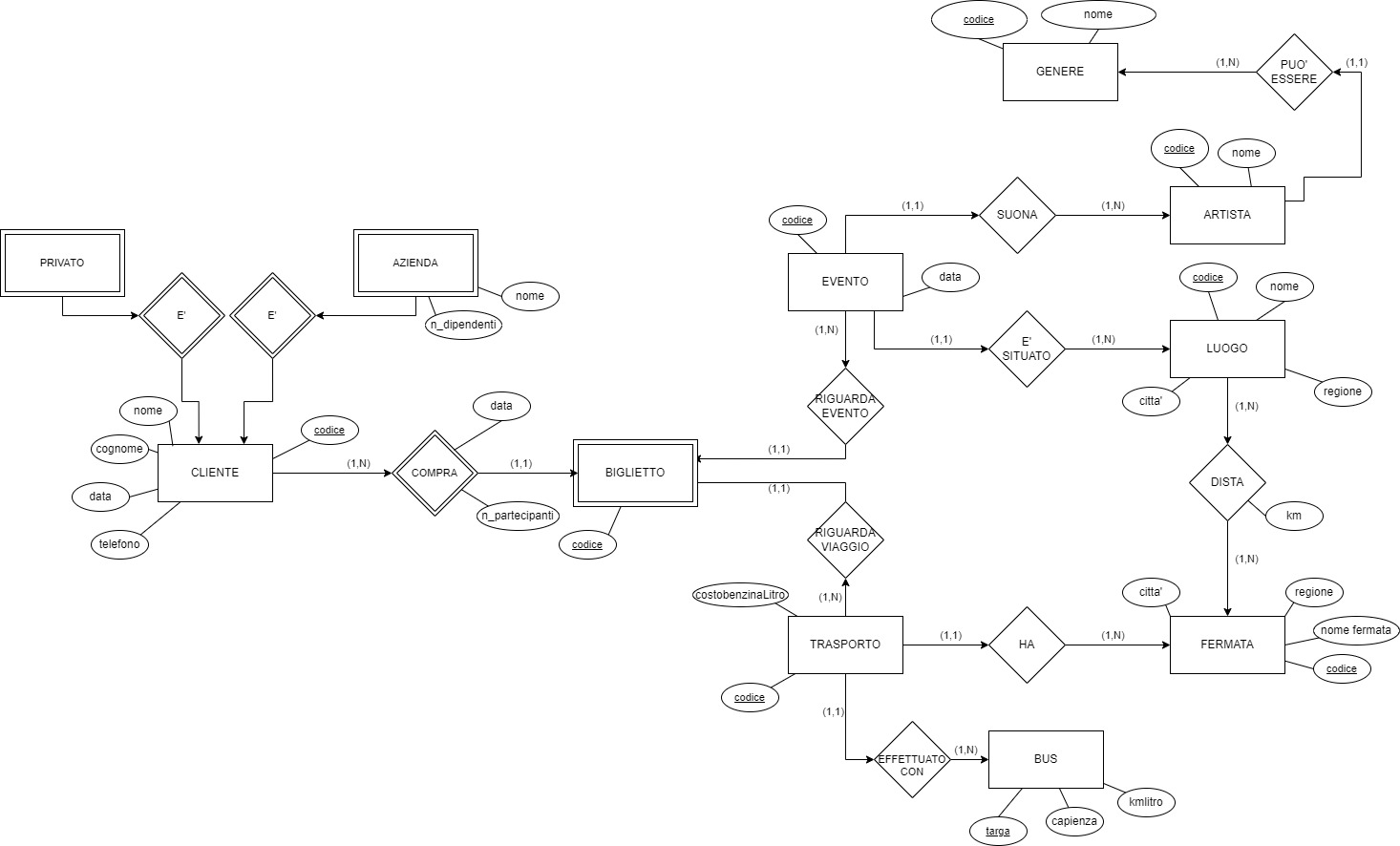
In questa fase di progettazione logica, è necessario individuare un metodo efficace di ristrutturazione che permetta l’eliminazione di questa gerarchia

La scelta effettuata è la creazione di due nuove relazioni, che rendono le entità figlie due nuove entità deboli. Ciò è necesario affinchè si preservi la separazione dei due concetti e si evitino attributi NULL.

Ristrutturiamo quindi come segue:

**Eliminazione dell’attributo multivalore**Nello schema inizialmente elaborato, compare un attributo multivalore. Si sceglie di definire una nuova entità debole ‘’genere’’, in relazione con l’entità ‘’artista’’



**Schema EER ristrutturato**

**Schema relazionale**

Si procede al mapping della base di dati

**Cliente** (username, dataNascita, nome, cognome, dataNascita,numTelefono)

**privato**(cliente.username↑, citta, via)

**azienda**(cliente.username↑, n\_dipendenti, nome)

**biglietto**(cliente.username↑, codice, data,n\_partecipanti, evento.codice↑, evento.codice↑,)

**evento**(codice, dataConcerto, artista.codice↑, luogo.codice↑)

**Artista**(codice, genere.codice↑, nome)

**genere**(codice,nome)

**Luogo**(luogo, nome, regione, città)

**bus**(targa, capienza, kmlitro)

**Trasporto**(codice, fermata.codice↑, bus.targa↑,CostobenzinaLitro)

**fermata**(codice, citta, regione, nome)

**dista**(fermata.codice↑, luogo.codice ↑, km)

**Normalizzazione**

Il database si presenta già normalizzato.

È infatti in prima forma normale in quanto tutti gli attributi sono atomici .

È in seconda forma normale perché, oltre ad essere già in

1NF, quando è presente una chiave primaria composta da più attributi tutte le

dipendenze funzionali che la riguardano sono piene e non parziali. È in terza forma

normale perché, oltre ad essere già in 2NF, in tutte le tabelle non sono presenti

dipendenze transitive fra attributi non chiave e la chiave primaria.

**creazione query**

DROP TABLE IF EXISTS genere;

CREATE TABLE genere

(

nome varchar(20) NOT NULL,

codice int PRIMARY KEY NOT NULL AUTO\_INCREMENT

);

DROP TABLE IF EXISTS artista;

CREATE TABLE artista

(

codice int NOT NULL AUTO\_INCREMENT primary key,

nome varchar(20) NOT NULL,

codiceGenere int NOT NULL,

foreign key(codiceGenere) references genere(codice)

);

DROP TABLE IF EXISTS luogo;

CREATE TABLE luogo

(

codice int NOT NULL AUTO\_INCREMENT primary key,

nome varchar(20) NOT NULL,

regione varchar(20) NOT NULL,

citta varchar(20) NOT NULL

);

DROP TABLE IF EXISTS bus;

CREATE TABLE bus

(

capienza int NOT NULL,

targa varchar(7) NOT NULL,

kmlitro int not null,

primary key(targa)

);

DROP TABLE IF EXISTS fermata;

CREATE TABLE fermata

(

codice int NOT NULL AUTO\_INCREMENT,

regione varchar(20) NOT NULL,

citta varchar(20) NOT NULL,

nome varchar(50) NOT NULL,

primary key(codice)

);

DROP TABLE IF EXISTS dista;

CREATE TABLE dista

(

codiceFermata int NOT NULL,

codiceLuogo int NOT NULL,

km int NOT NULL,

primary key(codiceFermata,codiceLuogo),

foreign key(codiceLuogo) references luogo(codice),

foreign key(codiceFermata) references fermata(codice)

);

DROP TABLE IF EXISTS trasporto;

CREATE TABLE trasporto

(

codice int NOT NULL AUTO\_INCREMENT,

codiceFermata int NOT NULL,

targa varchar(7) NOT NULL,

benzinaLitro float not null,

primary key(codice),

foreign key(codiceFermata) references fermata(codice),

foreign key(targa) references bus(targa)

);

DROP TABLE IF EXISTS evento;

CREATE TABLE evento

(

codice int NOT NULL AUTO\_INCREMENT,

codiceArtista int NOT NULL,

codiceLuogo int NOT NULL,

dataConcerto date not null,

primary key(codice),

foreign key(codiceArtista) references artista(codice),

foreign key(codiceLuogo) references luogo(codice)

);

DROP TABLE IF EXISTS cliente;

CREATE TABLE cliente

(

username varchar(20) NOT NULL,

userpass varchar(20) NOT NULL,

nome varchar(50) NOT NULL,

cognome varchar(50) NOT NULL,

telefono varchar(10) NOT NULL,

dataNascita date not null,

primary key(username)

);

DROP TABLE IF EXISTS privato;

CREATE TABLE privato

(

username varchar(20) NOT NULL,

primary key(username),

foreign key(username) references cliente(username)

);

DROP TABLE IF EXISTS azienda;

CREATE TABLE azienda

(

username varchar(20) NOT NULL,

primary key(username),

nome varchar(20) NOT NULL,

n\_dipendenti int NOT NULL,

foreign key(username) references cliente(username)

);

DROP TABLE IF EXISTS biglietto;

CREATE TABLE biglietto

(

codice int NOT NULL AUTO\_INCREMENT,

username varchar(20) NOT NULL,

codiceEvento int NOT NULL,

codiceTrasporto int NOT NULL,

dataAcquisto date not null,

n\_partecipanti int not null,

primary key(codice,username),

foreign key(username) references cliente(username),

foreign key(codiceEvento) references evento(codice),

foreign key(codiceTrasporto) references trasporto(codice)

);