PROGETTO BASI DI DATI
iTunes

Saverio Follador
account: sfollado
matricola: 1096984

Omid Taha
account: otaha
matricola: 1126078

Il progetto è stato caricato sull'account di laboratorio otaha.

ABSTRACT

iTunes è un'applicazione sviluppata e distribuita da Apple Inc. per riprodurre e organizzare brani musicali, permettendone l'acquisto online attraverso un servizio di distribuzione digitale (iTunes Store). Una volta effettuato l'acquisto, il contenuto viene aggiunto alla libreria personale dell'utente, la quale poi può essere sincronizzata sui dispositivi di riproduzione digitale di proprietà. iTunes supporta nativamente il lettore di musica digitale di Apple, l'iPod, oltre che l'iPhone e l'iPad. La distribuzione di iTunes come freeware, la sua semplicità di utilizzo e la sua capacità di mantenere una buona organizzazione della libreria audio hanno contribuito al suo successo, rendendolo standard sotto macOS. L'applicazione è disponibile in 120 paesi nel mondo; a maggio 2014 risultano oltre 800 milioni di accounts, 37 milioni di brani musicali in vendita e 35 miliardi di brani venduti.

ANALISI DEI REQUISITI

Si vuole realizzare una base di dati che contenga e gestisca le informazioni relative all'applicazione di organizzazione e distribuzione digitale di brani musicali iTunes. In particolare si vogliono conoscere le informazioni riguardo agli utenti, i brani, gli acquisti - ascolti - valutazioni dei brani effettuati dagli utenti.

Ciascun utente di iTunes è identificato dalle seguenti informazioni:

- un ID che lo identifica univocamente;
- varie informazioni personali quali nome, cognome, data di nascita, nazionalità;
- le credenziali usate per l'autenticazione (e-mail e password);
- il credito disponibile per effettuare acquisti;
- una libreria, contenente i brani acquistati;

Ciascun brano è identificato dalle seguenti informazioni:

- un ID che lo identifica univocamente;
- il suo titolo;
- la sua durata
- la data di pubblicazione;
- la sua dimensione (in byte);
- il prezzo a cui viene venduto (in Euro);
- l'artista (e relative informazioni personali) che lo esegue;
- il genere musicale di appartenenza;
- l'album musicale (qualora sia contenuto in alcuno);

Ciascun acquisto è identificato dalle seguenti informazioni:

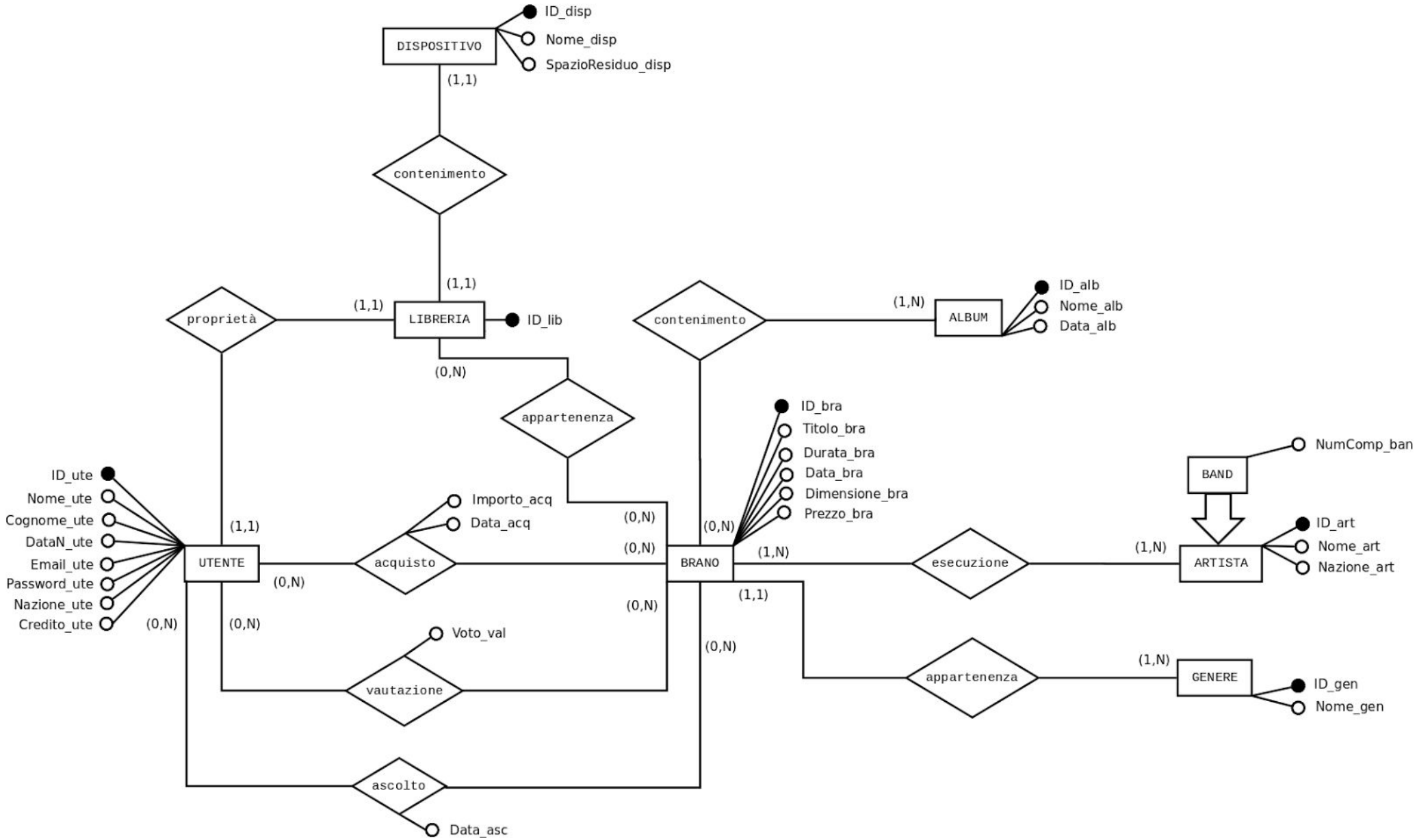
- gli ID dell'utente e del brano, che insieme identificano univocamente l'acquisto;
- la data in cui viene effettuato l'acquisto;
- il suo importo;

Ciascun ascolto è identificato dalle seguenti informazioni:

- l'ID dell'utente, l'ID del brano e la data (data e ora) in cui viene effettuato l'ascolto. Insieme, queste informazioni lo identificano univocamente;

Ciascuna valutazione è identificata dalle seguenti informazioni:

- gli ID dell'utente e del brano, che insieme identificano univocamente la valutazione;
- il voto assegnato al brano dall'utente (da 0 a 5)


DISPOSITIVO
ID_disp
Nome_disp
SpazioResiduo_disp
(1,1)
contenimento
(1,1)
LIBRERIA
ID_lib
(1,1)
proprietà
(0,N)
appartenenza
contenimento
(1,N)
ALBUM
ID_alb
Nome_alb
Data_alb
ID_ute
Nome_ute
Cognome_ute
DataN_ute
Email_ute
Password_ute
Nazione_ute
Credito_ute
(1,1)
UTENTE
(0,N)
(0,N)
acquisto
Importo_acq
Data_acq
(0,N)
(0,N)
BRANO
(0,N)
(0,N)
(0,N)
(0,N)
vautazione
Voto_val
(0,N)
ascolto
Data_asc
ID_bra
Titolo_bra
Durata_bra
Data_bra
Dimensione_bra
Prezzo_bra
(0,N)
(1,N)
esecuzione
(1,N)
ARTISTA
BAND
NumComp_ban
ID_art
Nome_art
Nazione_art
(1,1)
appartenenza
(1,N)
GENERE
ID_gen
Nome_gen

Vincoli non esprimibili nel diagramma E-R:

- L'utente può ascoltare e valutare un brano solo se questo è presente nella sua libreria
- L'utente può valutare un brano solo se questo è stato ascoltato almeno una volta

I vincoli vengono implementati nel database tramite triggers.

PROGETTAZIONE CONCETTUALE

1) Entità e Attributi

○ Utente
- ID_ute : int
- Nome_ute : string
- Cognome_ute : string
- DataN_ute : date
- Email_ute : string
- Password_ute : string
- Nazione_ute : string
- Credito_ute : float

○ Brano
- ID_bra : int
- Titolo_bra : string
- Durata_bra : time
- Data_bra : date
- Dimensione_bra : int
- Prezzo_bra : float

○ Libreria
- ID_lib : int

○ Dispositivo
- ID_disp : int
- Nome_disp : string
- SpazioResiduo_disp : int

○ Genere
- ID_gen : int
- Nome_gen : string

○ Album

- ID_alb : int
- Nome_alb : string
- Data_alb : date

○ Artista
- ID_art : int
- Nome_art : string
- Nazione_art : string

Viene definita la seguente entità figlia di Artista:

○ Band
- NumComp_ban: int

2) Relazioni

Brano-Genere: appartenenza.
- Un brano appartiene ad uno solo genere.
- Ad un genere appartengono uno o più di uno brani.

Brano-Artista: esecuzione.
- Un brano è eseguito da uno o più di uno artisti.
- Un artista esegue uno o più di uno brani.

Brano-Album: contenimento.
- Un brano può essere contenuto in nessuno, uno o più di un album.
- Un album contiene uno o più di uno brani.

Brano-Libreria: appartenenza.
- Un brano può appartenere a nessuna, una o più di una librerie.
- Ad una libreria possono appartenere nessuno, uno o più di uno brani.

Brano-Utente: acquisto.
- Un brano può essere acquistato da nessuno, uno o più di uno utenti.
- Un utente può acquistare nessuno, uno o più di uno brani.

Brano-Utente: valutazione.
- Un brano può essere valutato da nessuno, uno o più di uno utenti.
- Un utente può valutare nessuno, uno o più di uno brani.

Brano-Utente: ascolto.
- Un brano può essere ascoltato da nessuno, uno o più di uno utenti.

- Un utente può ascoltare nessuno, uno o più di uno brani.

Utente-Libreria: proprietà.
- Un utente possiede una e una sola libreria.
- Una libreria è posseduta da uno e un solo utente.

Dispositivo-Libreria: contenimento.
- Un dispositivo può contenere una e una sola libreria.
- Una libreria può essere contenuta in uno ed un solo dispositivo.

PROGETTAZIONE LOGICA

1) Generalizzazioni

La generalizzazione nell'entità "Artista" viene tradotta accorpando l'entità figlia "Band" nell'entità padre "Artista". Ciò è possibile poiché l'entità figlia "Band" non ha associazioni proprie. Come conseguenza di questa scelta si ottengono un minore numero di accessi e tuttavia un consumo di memoria maggiore (si avranno alcuni casi con valore NULL).
La generalizzazione diventa quindi:

○ Artista
- ID_art : int <<PK>>
- Nome_art : string
- Nazione_art : string
- NumComp_ban: int

2) Associazioni

Brano-Genere: appartenenza.
Essendo una relazione uno a molti viene aggiunta una chiave esterna nella relazione Brano:
Brano (ID_bra, Titolo_bra, Durata_bra, Data_bra, Dimensione_bra, Prezzo_bra, Genere_bra)

Esiste il vincolo di integrità referenziale tra l'attributo "Genere_bra" della relazione "Brano" e l'attributo "ID_gen" della relazione "Genere".

Brano-Artista: esecuzione.
Essendo una relazione molti a molti viene tradotta con questa relazione:

Esecuzione (Brano_ese, Artista_ese)

Esiste il vincolo di integrità referenziale tra gli attributi “Brano_ese” e “Artista_ese” della relazione “Esecuzione” e gli attributi “ID_bra” della relazione “Brano” e “ID_art” della relazione “Artista”.

Brano-Album: contenimento.
Essendo una relazione molti a molti viene tradotta con questa relazione:

BranoAlbum (Brano_ba, Album_ba)

Esiste il vincolo di integrità referenziale tra gli attributi “Brano_ba” e “Album_ba” della relazione “BranoAlbum” e gli attributi “ID_bra” della relazione “Brano” e “ID_alb” della relazione “Album”.

Brano-Libreria: appartenenza.
Essendo una relazione molti a molti viene tradotta con questa relazione:

BranoLibreria (Brano_lb,Libreria_lb)

Esiste il vincolo di integrità referenziale tra gli attributi “Brano_lb” e “Libreria_lb” della relazione “BranoLibreria” e gli attributi “ID_bra” della relazione “Brano” e “ID_lib” della relazione “Libreria”.

Brano-Utente: acquisto.
Essendo una relazione molti a molti viene tradotta con questa relazione:

Acquisto (Data_acq, Importo_acq, Utente_acq, Brano_acq)

Esiste il vincolo di integrità referenziale tra gli attributi “Utente_acq” e “Brano_acq” della relazione “Acquisto” e gli attributi “ID_ute” della relazione “Utente” e “ID_bra” della relazione “Brano”.

Brano-Utente: valutazione.
Essendo una relazione molti a molti viene tradotta con questa relazione:

Valutazione (Voto_val, Utente_val, Brano_val)

Esiste il vincolo di integrità referenziale tra gli attributi “Utente_val” e “Brano_val” della relazione “Valutazione” e gli attributi “ID_ute” della relazione “Utente” e “ID_bra” della relazione “Brano”.

Brano-Utente: ascolto.

Essendo una relazione molti a molti viene tradotta con questa relazione:

Ascolto (Data_asc, Utente_asc, Brano_asc)

Esiste il vincolo di integrità referenziale tra gli attributi “Utente_asc” e “Brano_asc” della relazione “Ascolto” e gli attributi “ID_ute” della relazione “Utente” e “ID_bra” della relazione “Brano”.

Utente-Libreria: proprietà.
Essendo una relazione uno a uno, viene aggiunta una chiave esterna nella relazione Libreria:

Libreria (ID_lib, Utente_lib)

Esiste il vincolo di integrità referenziale tra l'attributo “Utente_lib” della relazione “Libreria” e l'attributo “ID_ute” della relazione “Utente”.

Dispositivo-Libreria: contenimento.
Essendo una relazione uno a uno, viene aggiunta una chiave esterna nella relazione Dispositivo:

Dispositivo (ID_disp, Nome_disp, SpazioResiduo_disp, Libreria_disp)

Esiste il vincolo di integrità referenziale tra l'attributo “Libreria_disp” della relazione “Dispositivo” e l'attributo “ID_lib” della relazione “Libreria”.

3) Schema relazionale

Utente (ID_ute, Nome_ute, Cognome_ute, DataN_ute, Email_ute, Password_ute, Nazione_ute, Credito_ute)

Genere (ID_gen, Nome_gen)

Brano (ID_bra, Titolo_bra, Durata_bra, Data_bra, Dimensione_bra, Prezzo_bra, Genere_bra)

Libreria (ID_lib, Utente_lib)

Dispositivo (ID_disp, Nome_disp, SpazioResiduo_disp, Libreria_disp)

Album (ID_alb, Nome_alb, Data_alb)

Artista (ID_art, Nome_art, Nazione_art, NumComp_ban)

Valutazione (Voto_val, Utente_val, Brano_val)

Acquisto (Data_acq, Importo_acq, Utente_acq, Brano_acq)

Ascolto (Data_asc, Utente_asc, Brano_asc)

BranoLibreria (Brano_lb,Libreria_lb)

BranoAlbum (Brano_ba, Album_ba)

Esecuzione (Brano_ese, Artista_ese)

IMPLEMENTAZIONE DELLO SCHEMA RELAZIONALE

Di seguito viene riportato il codice relativo all'implementazione delle tabelle sopra indicate nella base di dati.
Viene aggiunta anche una tabella Errori che conterrà eventuali errori sollevati da trigger e procedure.

```
SET FOREIGN_KEY_CHECKS=0;

CREATE TABLE IF NOT EXISTS Utente(
   ID_ute int(11) auto_increment primary key,
   Nome_ute varchar(50) not null,
   Cognome_ute varchar(50) not null,
   DataN_ute date not null,
   Email_ute varchar(50) not null,
   Password_ute varchar(50) not null,
   Nazione_ute varchar(50) not null,
   Credito_ute float not null
);

CREATE TABLE IF NOT EXISTS Genere(
   ID_gen int(11) auto_increment primary key,
   Nome_gen varchar(50) not null
);

CREATE TABLE IF NOT EXISTS Brano(
   ID_bra int(11) auto_increment primary key,
   Titolo_bra varchar(50) not null,
   Durata_bra time not null,
   Data_bra date not null,
   Dimensione_bra int(11) not null,
   Prezzo_bra float not null,
   Prezzo_bra float not null,
   Genere_bra int(11),
   FOREIGN KEY(Genere_bra) REFERENCES Genere(ID_gen)
);

CREATE TABLE IF NOT EXISTS Libreria(
```

```
    ID_lib int(11) auto_increment primary key,
    Utente_lib int(11) not null,
    FOREIGN KEY(Utente_lib) REFERENCES Utente(ID_ute)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Dispositivo(
    ID_disp int(11) auto_increment primary key,
    Nome_disp varchar(50),
    SpazioResiduo_disp bigint(12) not null,
    Libreria_disp int(11) not null,
    FOREIGN KEY(Libreria_disp) REFERENCES Libreria(ID_lib)
);

CREATE TABLE IF NOT EXISTS Album(
    ID_alb int(11) auto_increment primary key,
    Nome_alb varchar(50) not null,
    Data_alb date not null
);

CREATE TABLE IF NOT EXISTS Artista(
    ID_art int(11) auto_increment primary key,
    Nome_art varchar(50) not null,
    Nazione_art varchar(50) not null,
    NumComp_ban int(11)
);

CREATE TABLE IF NOT EXISTS Esecuzione(
    Brano_ese int(11),
    Artista_ese int(11),
    PRIMARY KEY(Brano_ese,Artista_ese),
    FOREIGN KEY(Brano_ese) REFERENCES Brano(ID_bra)
        ON DELETE CASCADE,
    FOREIGN KEY(Artista_ese) REFERENCES Artista(ID_art)
        ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS BranoAlbum(
    Brano_ba int(11),
    Album_ba int(11),
    PRIMARY KEY(Brano_ba,Album_ba),
    FOREIGN KEY(Brano_ba) REFERENCES Brano(ID_bra)
        ON DELETE CASCADE,
    FOREIGN KEY(Album_ba) REFERENCES Album(ID_alb)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS BranoLibreria(
    Brano_lb int(11),
    Libreria_lb int(11),
    PRIMARY KEY(Brano_lb,Libreria_lb),
    FOREIGN KEY(Brano_lb) REFERENCES Brano(ID_bra)
        ON DELETE CASCADE,
    FOREIGN KEY(Libreria_lb) REFERENCES Libreria(ID_lib)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Acquisto(
    Data_acq date not null,
    Importo_acq float not null,
    Utente_acq int(11),
    Brano_acq int(11),
    PRIMARY KEY(Utente_acq,Brano_acq),
    FOREIGN KEY(Utente_acq) REFERENCES Utente(ID_ute),
    FOREIGN KEY(Brano_acq) references Brano(ID_bra)
);

CREATE TABLE IF NOT EXISTS Valutazione(
    Voto_val int(1),
    Utente_val int(11),
    Brano_val int(11),
    PRIMARY KEY(Utente_val,Brano_val),
    FOREIGN KEY(Utente_val) REFERENCES Utente(ID_ute),
    FOREIGN KEY(Brano_val) REFERENCES Brano(ID_bra)
```

```
);

CREATE TABLE IF NOT EXISTS Ascolto(
   Data_asc datetime,
   Utente_asc int(11),
   Brano_asc int(11),
   PRIMARY KEY(Data_asc,Utente_asc,Brano_asc),
   FOREIGN KEY(Utente_asc) REFERENCES Utente(ID_ute),
   FOREIGN KEY(Brano_asc) REFERENCES Brano(ID_bra)
);

CREATE TABLE IF NOT EXISTS Errori(
   ID_err int(11) auto_increment primary key,
   Descrizione_err varchar(50) not null
);

SET FOREIGN_KEY_CHECKS=1;
```

QUERIES

1) Mostra ID, Titolo, Artista e numero di ascolti dei 5 brani più ascoltati in Italia

```
CREATE VIEW Ascolti AS
SELECT Brano.ID_bra, Brano.Titolo_bra, COUNT(*) AS Numero_Ascolti
FROM Brano, Ascolto, Utente
WHERE Brano.ID_bra=Ascolto.Brano_asc AND Ascolto.Utente_asc=Utente.ID_ute
AND Utente.Nazione_ute='Italia'
GROUP BY Brano.ID_bra, Brano.Titolo_bra

SELECT Ascolti.ID_Bra AS ID, Ascolti.Titolo_bra AS Titolo,
GROUP_CONCAT(Artista.Nome_art SEPARATOR ', ') AS Artista,
Ascolti.Numero_Ascolti
FROM Ascolti, Esecuzione, Artista
```

```
WHERE Ascolti.ID_bra=Esecuzione.Brano_ese and
Esecuzione.Artista_ese=Artista.ID_art
GROUP BY Ascolti.ID_Bra, Ascolti.Titolo_bra
ORDER BY Numero_ascolti DESC
LIMIT 5
```

ID	Titolo	Artista	Numero_Ascolti
49	Partigiano Reggiano	Zucchero	7
36	Feel So Close	Calvin Harris	5
108	Alive - Single	Sia	5
92	Te Amo	Rihanna	4
12	Thunderstruck	AC/DC	3

2) Totale speso per utente in acquisti

```
SELECT Utente.ID_ute AS ID, Utente.Nome_ute AS Nome, Utente.Cognome_ute
AS Cognome, Utente.Nazione_ute AS Nazione, round(Sum(Importo_acq),2) AS
Totale_Speso
FROM Acquisto JOIN Utente ON Acquisto.Utente_acq=Utente.ID_ute
GROUP BY Utente.ID_ute, Utente.Nome_ute, Utente.Cognome_ute,
Utente.Nazione_ute
```

ID	Nome	Cognome	Nazione	Totale_Speso
1	Mario	Rossi	Italia	19.35
2	John	Smith	UK	19.35
3	Oliver	Brown	USA	38.70
4	Charlotte	Petit	Francia	21.93
5	Lucas	Bauer	Germania	28.38
6	Lucia	Esposito	Italia	12.90

3) Mostra ID, Titolo, Artista e numero di acquisti dei 5 brani più venduti del 2016

```
CREATE VIEW Acquisti_2016 AS
SELECT Brano.ID_bra, Brano.Titolo_bra, COUNT(*) AS Numero_Acquisti
FROM Acquisto JOIN Brano ON Acquisto.Brano_acq=Brano.ID_bra
WHERE Acquisto.Data_acq BETWEEN '2016-01-01' AND '2016-12-31'
GROUP BY Brano.ID_bra, Brano.Titolo_bra
```

```
SELECT Acquisti_2016.ID_Bra AS ID, Acquisti_2016.Titolo_bra AS Titolo,
GROUP_CONCAT(Artista.Nome_art SEPARATOR ', ') AS Artista,
Acquisti_2016.Numero_Acquisti
FROM Acquisti_2016, Esecuzione, Artista
WHERE Acquisti_2016.ID_bra=Esecuzione.Brano_ese and
Esecuzione.Artista_ese=Artista.ID_art
GROUP BY Acquisti_2016.ID_bra, Acquisti_2016.Titolo_bra
ORDER BY Numero_Acquisti DESC
LIMIT 5
```

ID	Titolo	Artista	Numero_Acquisti
1	We Will Rock You	Queen	5
12	Thunderstruck	AC/DC	4
109	Vertigo - Single	U2	4
49	Partigiano Reggiano	Zucchero	3
97	Hey Brother	Avicii	3

4) Mostra Nome e Data d'esordio dell'artista da più tempo in attività

```
SELECT Artista.Nome_art AS Nome_Artista, Brano.Data_bra AS Data
FROM Artista, Esecuzione, Brano
WHERE Artista.ID_art=Esecuzione.Artista_ese AND
Esecuzione.Brano_ese=Brano.ID_bra
ORDER BY Brano.Data_bra ASC
LIMIT 1
```

Nome_Artista	Data
Pink Floyd	1973-03-01

5) Mostra ID e Nome degli artisti che sono band

```
SELECT ID_art AS ID, Nome_art AS Nome_Artista
FROM Artista
WHERE NumComp_ban IS NOT NULL
```

ID	Nome_Artista
1	Queen
2	AC/DC
3	Pink Floyd
11	U2

6) Mostra ID, Nome e relativo Genere musicale degli album (ottenuto dai brani ad esso appartenenti)

```
SELECT Album.ID_alb AS ID, Album.Nome_alb AS Nome_Album,
GROUP_CONCAT(DISTINCT Genere.Nome_gen SEPARATOR ', ') AS
Genere_Album
FROM Genere, Brano, BranoAlbum, Album
WHERE Genere.ID_gen=Brano.Genere_bra AND
Brano.ID_bra=BranoAlbum.Brano_ba AND BranoAlbum.Album_ba=Album.ID_alb
GROUP BY Album.ID_alb, Album.Nome_alb
```

ID	Nome_Album	Genere_Album
1	New Of The World	Rock
2	The Razors Edge	Rock
3	The Dark Side Of The Moon	Rock
4	18 Months	Dance
5	Black Cat	Pop
6	Encore	Hip-Hop/Rap
7	Rated R	Pop
8	True	Dance

7) ID e Titolo dei brani “single”, ovvero non appartenenti ad alcun album

```
SELECT ID_bra AS ID, Titolo_bra AS Titolo
FROM Brano
WHERE Brano.ID_bra NOT IN (SELECT Brano.Id_bra AS Brano
FROM Brano JOIN BranoAlbum ON Brano.ID_bra=BranoAlbum.Brano_ba)
```

ID	Titolo
107	This One's For You - Single
108	Alive - Single
109	Vertigo - Single

TRIGGERS

1) Trigger che, prima di inserire un nuovo brano nella libreria dell'utente, verifica che nel dispositivo ad essa associato vi sia sufficiente spazio nella memoria

```
DELIMITER ///
CREATE TRIGGER Check_Spazio
BEFORE INSERT ON BranoLibreria
FOR EACH ROW
BEGIN
DECLARE branosize double;
DECLARE spazioresiduo double;
SELECT Dimensione_bra INTO branosize
FROM Brano WHERE ID_bra=new.Brano_lb
LIMIT 1;
SELECT SpazioResiduo_disp INTO spazioresiduo
FROM Dispositivo WHERE Dispositivo.Libreria_disp=new.Libreria_lb
LIMIT 1;
IF (branosize>spazioresiduo) THEN
INSERT INTO Errori (Descrizione_err) VALUES ('Spazio esaurito');
END IF;
END ///
```

2) Trigger che, prima di inserire una tupla in Ascolto, verifica che il brano sia contenuto nella libreria dell'utente

```
DELIMITER ///
CREATE TRIGGER Check_Libreria_Ascolto
BEFORE INSERT ON Ascolto
FOR EACH ROW
BEGIN
DECLARE n integer;
SELECT COUNT(*) INTO n
FROM Libreria JOIN BranoLibreria ON Libreria.ID_lib=BranoLibreria.Libreria_lb
WHERE BranoLibreria.Brano_lb=new.Brano_asc AND
Libreria.Utente_lib=new.Utente_asc;
IF (n=0) THEN
```

```
INSERT INTO Errori (Descrizione_err) VALUES ('Ascolto di un brano non presente nella libreria');
END IF;
END ///
```

3) Trigger che, prima di inserire una tupla in Valutazione, verifica che il brano sia stato ascoltato almeno una volta dall'utente

```
DELIMITER ///
CREATE TRIGGER Check_Ascolto_Valutazione
BEFORE INSERT ON Valutazione
FOR EACH ROW
BEGIN
DECLARE n integer;
SELECT COUNT(*) INTO n
FROM Ascolto
WHERE Ascolto.Brano_asc=new.Brano_val;
IF (n=0) THEN
INSERT INTO Errori (Descrizione_err) VALUES ('Valutazione di un brano mai ascoltato');
END IF;
END ///
```

PROCEDURE

1) Procedura che ritorna la lista dei brani presenti nella libreria di un determinato utente il cui ID viene assunto in input

```
DELIMITER ///
CREATE PROCEDURE MostraLibreria(IN user INT(11))
BEGIN
SELECT Brano.ID_bra, Brano.Titolo_bra
FROM Brano, BranoLibreria, Libreria, Utente
```

```
WHERE Brano.ID_bra=BranoLibreria.Brano_lb AND
BranoLibreria.Libreria_lb=Libreria.ID_lib AND Libreria.Utente_lib=Utente.ID_ute
AND Utente.ID_ute=user;
END ///
```

2) Procedura che scala dal credito di un utente un importo; ID utente e importo da scalare dati in input

```
DELIMITER ///
CREATE PROCEDURE ScalaImporto(IN importo FLOAT,IN utente INT(11))
BEGIN
UPDATE Utente SET Credito_ute=Credito_ute-importo WHERE ID_ute=utente;
END ///
```

FUNZIONI

1) Funzione che calcola il costo di un album

```
DELIMITER ///
CREATE FUNCTION prezzoAlbum(IDalbum INT(11)) RETURNS DECIMAL(5,2)
BEGIN
DECLARE totale DECIMAL(5,2);
SELECT SUM(Brano.Prezzo_bra) INTO totale
FROM BranoAlbum join Brano on BranoAlbum.Brano_ba=Brano.ID_bra
WHERE BranoAlbum.Album_ba=IDalbum;
RETURN totale;
END ///
```

2) Funzione che calcola la spesa di un utente in un determinato arco di tempo

```
DELIMITER ///
CREATE FUNCTION spesaUtenteData(IDute INT(11), dataMin DATE, dataMax
DATE) RETURNS DECIMAL(11,2)
BEGIN
```

```
DECLARE tot DECIMAL(11,2);
SELECT SUM(Brano.Prezzo_bra) INTO tot
FROM Acquisto JOIN Brano ON Acquisto.Brano_acq=Brano.ID_bra
WHERE Acquisto.Utente_acq=IDute AND Acquisto.Data_acq BETWEEN dataMin
AND dataMax;
RETURN tot;
END ///
```