

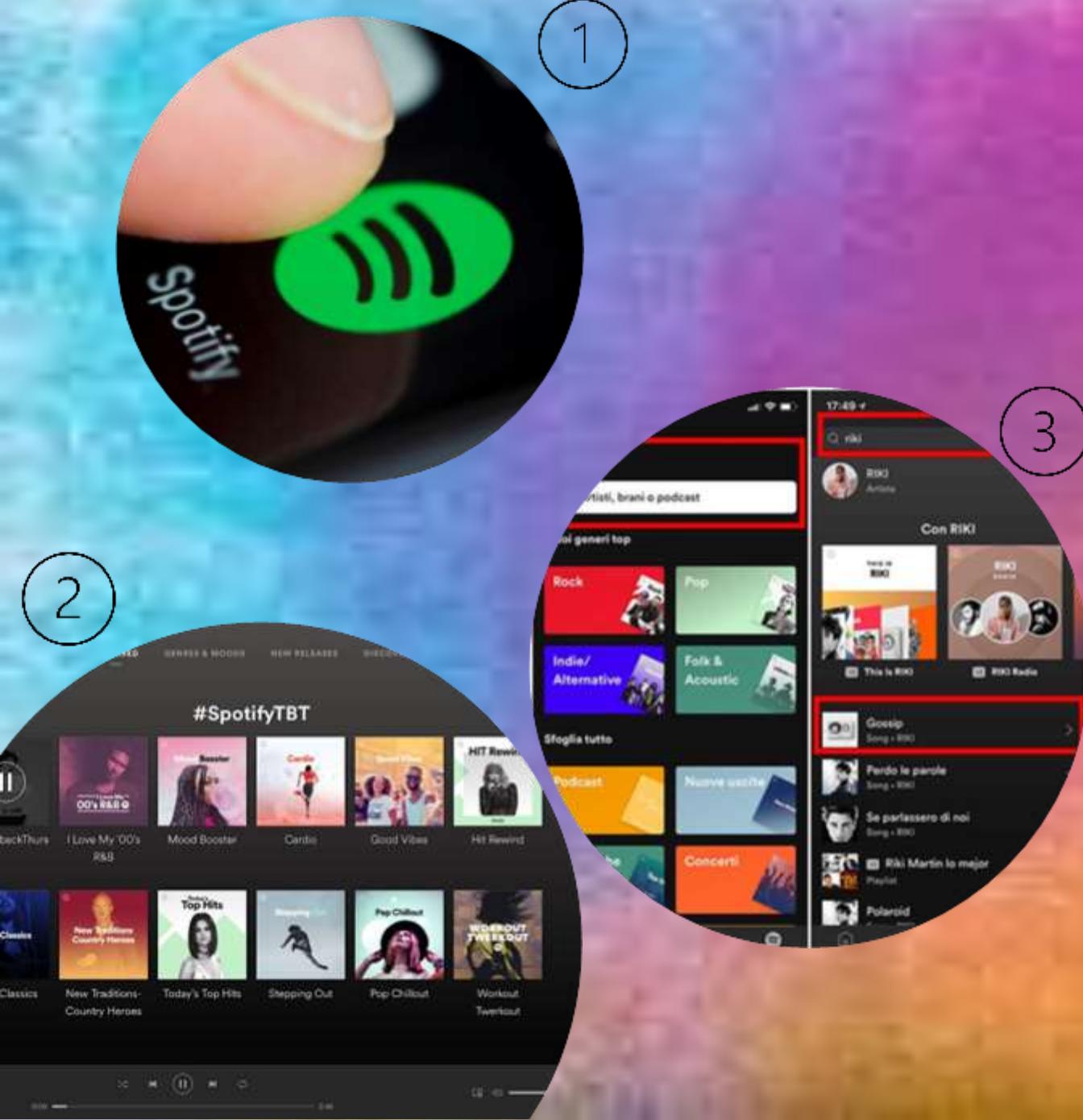
# **PROGETTO BASI DI DATI**

**M.COIRO-K.BUONOCORE-R.CUCCARO**

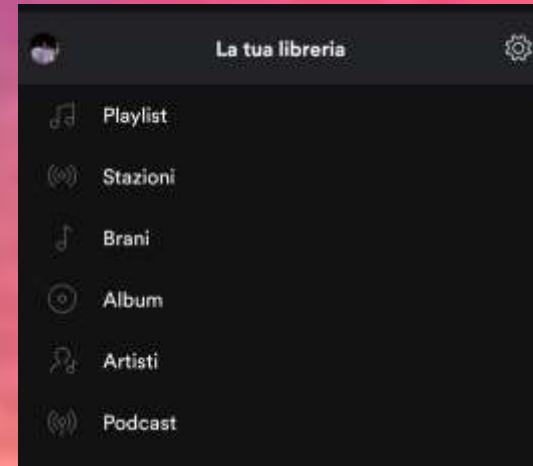
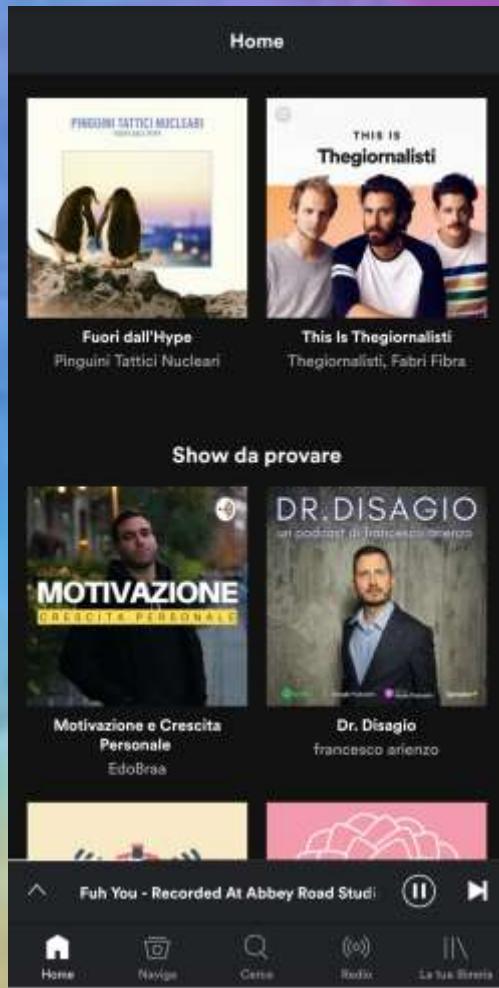
**Descrizione sintetica della realtà di interesse.**

Abbiamo deciso di sviluppare una base di dati relativa ad una piattaforma musicale. La scelta è stata fatta prevalentemente prendendo in considerazioni le numerose app di streaming musicali presenti sui nostri smartphone ma più in generale presenti su tutti gli store.

Prendiamo in considerazione la piattaforma musicale Spotify, il cui funzionamento viene illustrato nelle seguenti immagini:



Il funzionamento della nostra piattaforma è molto simile a quello di Spotify, l'utente attraverso un account accede alla **Home** e attraverso un **Menù** visualizza i servizi offerti all'utente. Il nostro obiettivo è proprio quello di gestire questi vari servizi tralasciando le altre caratteristiche della piattaforma, come per esempio lo storico relativo alla creazione delle playlist e all'iscrizione degli utenti.





# PIATTAFORMA MUSICALE

Si devono raccogliere e organizzare le informazioni relative agli account utente di una piattaforma dedicata allo streaming musicale, che offre agli utenti la possibilità di accedere ad un vasto database di canzoni da poter ascoltare senza limiti.

Ogni account utente è costituito: nickname (lo identifica univocamente), il numero di playlist musicali, il numero di follower e il numero di artisti seguiti.

Ogni utente può creare nessuna o più playlist. Ciascuna playlist è composta da: codice della playlist che la identifica univocamente, nome playlist, numero di brani e la radio della playlist.

Inoltre ciascuna playlist si divide in pubblica e privata, per quella pubblica vogliamo memorizzare il numero di follower.

Ogni playlist è costituita da brani ,per ciascun brano vogliamo definire: il titolo, il codice e la durata.

Ciascun brano viene scritto da un artista e appartiene ad un album.

Ciascun utente può ascoltare nessuno o più podcast, composto da : il numero di episodi, data, durata, descrizione, ideatore , nome podcast.

Inoltre ogni utente può seguire nessuno o più artisti. Di ciascun artista vogliamo memorizzare: il nome artista, la descrizione, il genere(pop ,rock, jazz ,indie etc...), le collaborazioni, le offerte(CD o Vinili), il numero di ascoltatori e infine il numero di album.

Per ogni album definiamo: il nome album, la data di pubblicazione, il codice album ( lo identifica univocamente).

L'album viene prodotto da un artista.

Infine ogni utente può realizzare nessuna o una stazione radio della quale vogliamo memorizzare il nome radio e la frequenza. Ogni stazione radio contiene i brani musicali.





Nel nostro database abbiamo individuato le seguenti **ENTITA'** con relativi **ATTRIBUTI**:

L'entità **ACCOUNT UTENTE** è costituita dai seguenti attributi: **Nickname**(string) che rappresenta la **chiave**, **#follower(int)**, **#artisti (int)** e **#playlist musicali** (int) che rappresentano **due attributi ridondanti**.

L'entità **PLAYLIST** è costituita dai seguenti attributi: **Codice Playlist** (int) che rappresenta la **chiave** , **Nome** (string), **Radio della Playlist** (string) e **#brani** (int) che rappresenta un **attributo ridondante**. Tale entità si suddivide tramite una specializzazione **TOTALE** in **Privata** e **Pubblica**, quest'ultima ha come attributo il **#follower** (int).

L'entità **Podcast** ha come attributi: **Nome Podcast** (string) e **Ideatore** (string) che rappresentano la **chiave composta**, **#episodi** (int), **Durata** (float), **Descrizione** (string) e **Data di Pubblicazione**(date).

L'entità **Stazione Radio** ha come attributi: **Nome** (string) e **Frequenza** (float) che rappresenta l'attributo **chiave**.

L'entità **Artista** è costituita dai seguenti attributi: **Nome Artista** (string) che rappresenta la **chiave**, **Collaborazioni** (string) , **Descrizione** (string) , **Offerte** (string) , **#ascoltatori** (int) , **#album** (int) che rappresenta un **attributo ridondante** e **Genere Musicale** (string) che rappresenta un **attributo multivaleure**.

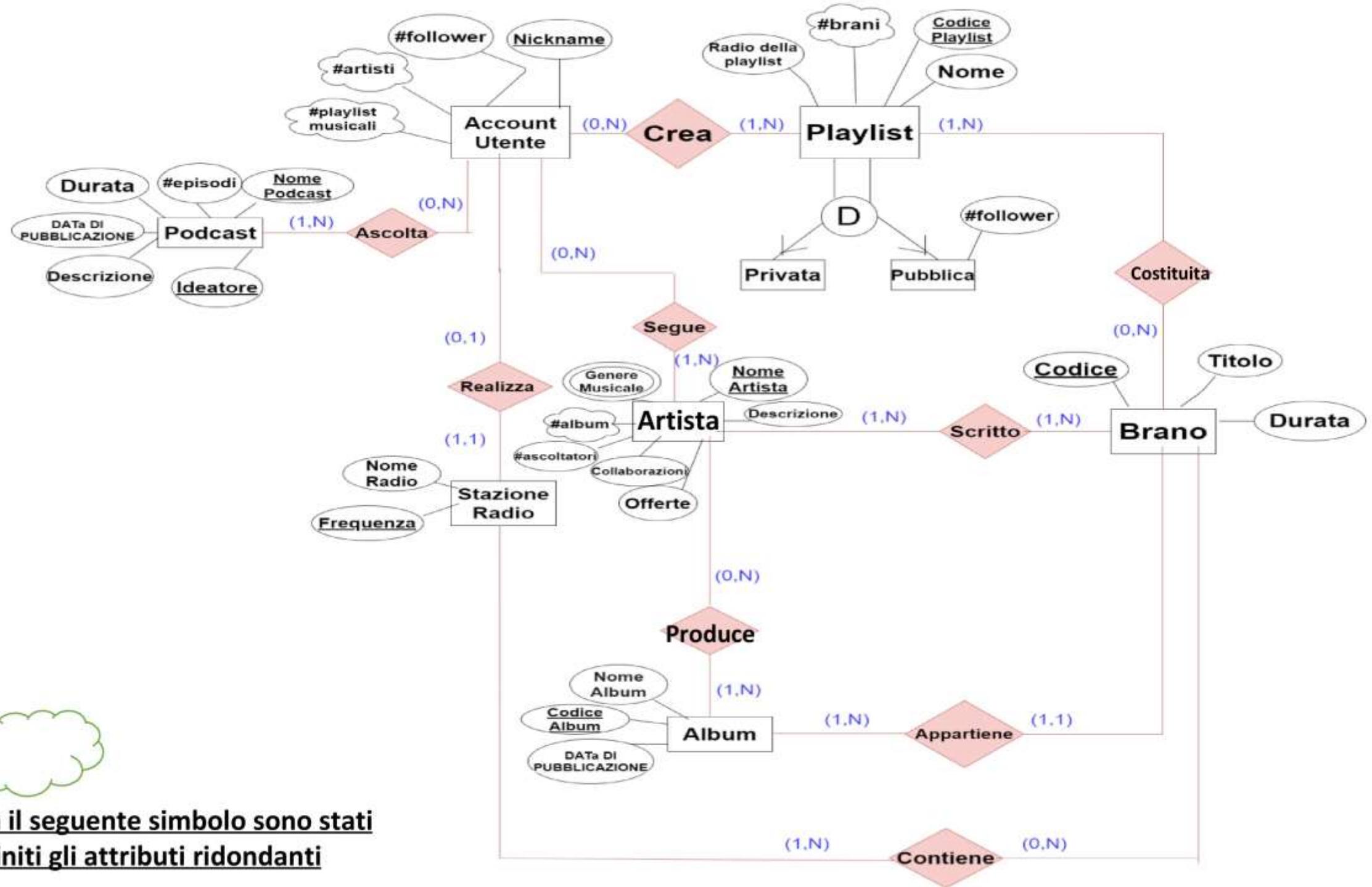
L'entità **Album** ha i seguenti attributi: **Nome Album** (string) , **Codice Album** (int) che rappresenta la **chiave** e la **Data di Pubblicazione** (date).

L'entità **Brano** è costituita dai seguenti attributi: **Codice** (int) che rappresenta la **chiave**, **Titolo** (string) e **Durata** (float).



Con in simbolo # definiamo il concetto di «numero».





Con il seguente simbolo sono stati  
definiti gli attributi ridondanti



Nella realizzazione del precedente diagramma EER abbiamo individuato varie Entità e Attributi che abbiamo già definito. Individuiamo adesso le relazioni che intercorrono tra le medesime entità.

Tra **Account Utente** e **Playlist** abbiamo la relazione **CREARE**. Un account utente può creare nessuna o più playlist, una o più playlist può essere creata da almeno un account utente.

La relazione tra **Playlist** e **Brano** è **COSTITUIRE**, una Playlist deve essere costituita da almeno 1 o più brani. Un Brano può costituire nessuna o più playlist.

Tra l'entità **Brano** e **Album** abbiamo la relazione **APPARTENERE**. Un Brano deve appartenere ad un album ed ad un album deve appartenere almeno 1 brano o più brani.

La relazione tra **Brano** e **Artista** è **SCRIVERE**. Un Artista può scrivere 1 o più brani ed un brano può essere scritto da 1 o più artisti.

L'attributo **GENERE MUSICALE** dell'entità Artista è un attributo multivaleore poichè può assumere differenti valori per esempio: Pop, Jazz, Rock, Indie etc...

La relazione tra **Account Utente** e **Podcast** è **ASCOLTARE**. Un Account Utente può ascoltare nessuno o più podcast, un podcast deve essere ascoltato almeno da un account utente o da più account utente. Nell'entità **Account Utente** il #artisti seguiti è un attributo ridondante perché derivabile dal conteggio della relazione **SEGUIRE**. Sempre in questa entità il #playlist è un attributo ridondante perché derivabile dal conteggio ottenuto dalla relazione **CREARE**. Nell'entità **Playlist** il #brani è un attributo ridondante poiché derivabile dal conteggio relativo alla relazione **COSTITUIRE**.





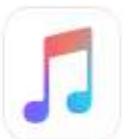
La relazione tra le entità **Account Utente** e **Artista** è **SEGUIRE**, un utente può seguire nessuno o più artisti e ciascun artista deve essere seguito da almeno un Account Utente.

La relazione tra **Artista** e **Album** è **PRODURRE**. Un artista può produrre nessuno o più album, O perché l'artista all'inizio scrive un brano e quindi l'album non è stato ancora creato.

Un album può essere prodotto da uno o più artisti. L'attributo **#album** dell'entità Artista è un attributo ridondante perché derivabile dal conteggio della relazione **PRODURRE**.

La relazione tra **Account Utente** e **Stazione Radio** è **REALIZZARE**. Un Account Utente può realizzare nessuna o una stazione radio, una stazione radio deve essere realizzata da un unico account utente.

La relazione tra **Stazione Radio** e **Brano** è **CONTENERE**. Una stazione radio deve contenere almeno 1 o più brani. Un brano può contenere nessuna o più stazioni radio.





# **RISTRUTTURAZIONE:**

Dopo la realizzazione del modello concettuale passiamo alla fase di **Ristrutturazione** la quale è suddivisa in 4 passi:

- Analisi delle Ridondanze**
- Eliminazione delle Generalizzazioni**
- Partizionamento o Accorpamento di concetti**
- Scelta degli ID principali**

Prima di affrontare queste fasi definiamo dei concetti importanti per l'analisi delle ridondanze: la TAVOLA DEI VOLUMI, la definizione delle Operazioni da effettuare sul medesimo database e la TAVOLA DELLE OPERAZIONI.

## TAVOLA DEI VOLUMI

CONCETTO	TIPO	VOLUME
Account Utente	E	800.000
Crea	R	1.600.000
Playlist	E	1.600.000
Pubblica	E	480.000
Privata	E	1.120.000
Ascolta	R	7.000
Podcast	E	10.000
Segue	R	1.000.000
Artista	E	1.500.000
Realizza	R	50.000
Stazione Radio	E	50.000
Contiene	R	1000
Brano	E	2.750.000
Costituito	R	3500
Produce	R	50
Album	E	250.000
Appartiene	R	11
Scritto	R	550



Si supponga che in tale piattaforma ci siano circa **800.000 account utenti**, i quali in media possono creare **circa 2 playlist**.

La maggior parte degli account utente( **circa il 70%** ) crea playlist private e il restante crea playlist pubbliche.

Pochi account utente ascoltano podcast e non tutti gli utenti seguono gli artisti presenti in tale piattaforma.

Ogni account utente può realizzare nessuna o una stazione radio e per ciascuna di esse ci sono almeno **1000 brani**.

Ogni playlist è costituita da **circa 3500 brani**.

Un artista produce **circa 50 album**.

In tale piattaforma abbiamo circa **250.000 album**, ognuno di essi è composto da **11 brani**, per un totale di **2.750.000 brani** sull'intera piattaforma.

Se ci basiamo sul concetto che un artista produce 50 album e ognuno di essi contiene 11 brani allora possiamo concludere dicendo che un artista scrive circa **550 canzoni**.

# OPERAZIONI:

- **Operazione1**: Creare un account utente.
- **Operazione2**: Leggere il numero di brani presenti in una playlist.
- **Operazione3**: Visualizzare il numero di album prodotti da un'artista.
- **Operazione4**: Creare una playlist.
- **Operazione5**: Leggere il numero di artisti seguiti da un account utente.
- **Operazione6**: Creazione di una stazione radio.
- **Operazione7**: Un account utente inizia a seguire un'artista.
- **Operazione8**: Registrare un nuovo album prodotto da un'artista.
- **Operazione9**: Stampare il numero di playlist create da un utente.
- **Operazione10**: Stampa tutti i dati dei podcast ascoltati dall'utente.



## TAVOLA DELLE OPERAZIONI:

OPERAZIONI	TIPO	FREQUENZA
Op1	I	100/giorno
Op2	I	20/giorno
Op3	I	30/giorno
Op4	I	500/giorno
Op5	I	40/giorno
Op6	I	10/giorno
Op7	I	250/giorno
Op8	I	15/giorno
Op9	I	20/giorno
Op10	B	1/mese

# **ANALISI** **DELLE** **RIDONDANZE**



-Entità Account Utente:  
#artisti(Op5 e Op7) e #playlist  
musicali(Op4 e Op9)



-Entità Playlist: #brani(Op2)



-Entità Artista : #album(Op3 e  
Op8)

# ENTITA': ACCOUNT UTENTE



## #artisti con RIDONDANZA -Op5

Operazione5: Leggere il numero di artisti seguiti da un account utente.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Account Utente	E	1	L

$$40 \times 1 = 40 \text{ accessi al giorno}$$

## #artisti con RIDONDANZA -Op7

Operazione7: Un account utente inizia a seguire un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Account Utente	E	1	L
Segue	R	1	L
Artista	E	1	S
Account Utente	E	1	S

$$(2 \times 250) + (4 \times 250) = 1500 \text{ accessi al giorno}$$

# #artisti SENZA RIDONDANZA –Op5

Operazione5: Leggere il numero di artisti seguiti da un account utente.

CONCETTO	COSTRUTTO	ACCESSO	TIPO	
Account Utente	E	1	L	$2 \times 40 = 80$ accessi al giorno
Segue	R	1	L	

# #artisti SENZA RIDONDANZA –Op7

Operazione7: Un account utente inizia a seguire un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO	
Segue	R	1	S	
Artista	E	1	S	$4 \times 250 = 1000$ accessi al giorno

# SOMMA ACCESSI CON RIDONDANZA

#artisti  
Op5 40 accessi al giorno  
Op7 1500 accessi al giorno

TOTALE= 1540 accessi al giorno

MEMORIA x |#account utente| = 4 byte x 800.000  
 $\approx$  3200 kb

Abbiamo deciso di  
rappresentare un  
intero con 4byte

# SOMMA ACCESSI SENZA RIDONDANZA

#artisti  
Op5 80 accessi al giorno  
Op7 1000 accessi al giorno

TOTALE= 1080 accessi al giorno

MEMORIA= 0

**In questo caso conviene  
eliminare l'attributo  
ridondante**

# ENTITA': ACCOUNT UTENTE



## #playlist musicali con RIDONDANZA –Op4

Operazione4: Creare una playlist.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Account Utente	E	1	L
Crea	R	1	S
Playlist	E	1	S
Account Utente	E	1	S

$(1 \times 500) + (6 \times 500) = 3500$   
accessi al giorno

## #playlist musicali con RIDONDANZA –Op9

Operazione9: Stampare il numero di playlist create da un utente.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Account Utente	E	1	L

$(1 \times 20) = 20$  accessi al giorno

# #playlist musicali SENZA RIDONDANZA –Op4

Operazione4: Creare una playlist.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Crea	R	1	S
Playlist	E	1	S

$(4 \times 500) = 2000$  accessi al giorno

# #playlist musicali SENZA RIDONDANZA –Op9

Operazione9: Stampare il numero di playlist create da un utente.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Account Utente	E	1	L
Crea	R	1	L

$(2 \times 20) = 40$  accessi al giorno

# SOMMA ACCESSI CON RIDONDANZA

#playlist musicali

Op4 3500 accessi al giorno

Op9 20 accessi al giorno

TOTALE= 3520 accessi al giorno

MEMORIA x |#account utente|=4 byte x 800.000

$\approx 3200 \text{ kb}$

Abbiamo deciso di  
rappresentare un  
intero con 4byte

# SOMMA ACCESSI SENZA RIDONDANZA

#playlist musicali

Op4 2000 accessi al giorno

Op9 40 accessi al giorno

TOTALE 2040 accessi al giorno

MEMORIA 0

**In questo caso conviene  
eliminare l'attributo  
ridondante**

## ENTITA': PLAYLIST



### #brani con RIDONDANZA –Op2

Operazione2: Leggere il numero di brani presenti in una playlist.

CONCETTO	COSTRUTTO	ACCESSO	TIPO	
Playlist	E	1	L	$1 \times 20 = 20$ accessi al giorno  $\text{MEMORIA} \times  \text{Playlist}  = 4 \times 1.600.000$ $\approx 6400 \text{ kb}$

### #brani con SENZA RIDONDANZA –Op2

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Costituito	R	1	L
Brano	E	1	L

Abbiamo deciso di rappresentare un intero con 4byte

**In questo caso conviene  
mantenere l'attributo  
ridondante**

$2 \times 20 = 40$  accessi al giorno

ENTITA': ARTISTA

# #album con RIDONDANZA –Op3



Operazione3: Visualizzare il numero di album prodotti da un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Artista	E	1	L

$(1 \times 30) = 30$  accessi al giorno

# #album con RIDONDANZA –Op8

Operazione8: Registrare un nuovo album prodotto da un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Artista	E	1	L
Produce	R	1	S
Album	E	1	S
Artista	E	1	S

$(1 \times 15) + (6 \times 15) = 105$  accessi al giorno

# #album SENZA RIDONDANZA –Op3

Operazione3: Visualizzare il numero di album prodotti da un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Artista	E	1	L
Produce	R	1	L

$(2 \times 30) = 60$  accessi al giorno

# #album SENZA RIDONDANZA –Op8

Operazione8: Registrare un nuovo album prodotto da un'artista.

CONCETTO	COSTRUTTO	ACCESSO	TIPO
Produce	R	1	S
Album	E	1	S

$(4 \times 15) = 60$  accessi al giorno

# SOMMA ACCESSI CON RIDONDANZA

#album

Op3 30 accessi al giorno

Op8 105 accessi al giorno

TOTALE=135 accessi al giorno

MEMORIA x |Artista| = 4byte x 1.500.000  
 $\approx$  6000kb

Abbiamo deciso di  
rappresentare un  
intero con 4byte

# SOMMA ACCESSI SENZA RIDONDANZA

#album

Op3 60 accessi al giorno

Op8 60 accessi al giorno

TOTALE=120 accessi al giorno

MEMORIA=0

**In questo caso conviene  
eliminare l'attributo**

**ridondante**

# ELIMINAZIONE DELLE GENERALIZZAZIONI

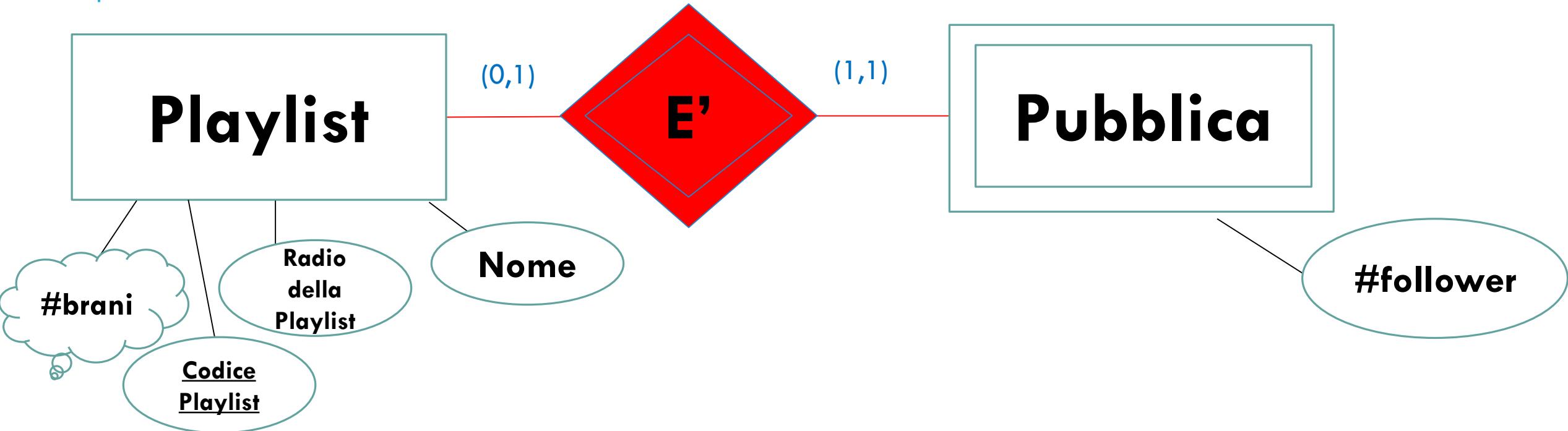


L'entità **Playlist** si suddivide in due sottoclassi: **Pubblica** e **Privata**, attraverso una Specializzazione. Il nostro obiettivo è quello di eliminare tale specializzazione. Possiamo fare questo seguendo vari metodi, ovviamente scegliendo quello più opportuno.

- **NON POSSIAMO:**
- -portare il padre nelle figlie, perché bisognerebbe dividere la relazione CREARE in due nuove relazioni.
- -portare le figlie nel padre perché bisognerebbe trasferire gli attributi delle figlie nel padre, aggiungendo l'attributo TIPO. Ma questo comporta un elevato numero di valori NULL, che sarebbero i seguenti:
  - PLAYLIST= 1.600.000;
  - PUBBLICA=480.000;
  - PRIVATA=1.120.000;

Avremo : 1.200.000 valori NULL se la playlist fosse stata PUBBLICA  
Avremo : 480.000 valori NULL se la playlist fosse stata PRIVATA

IN CONCLUSIONE APPLICHIAMO IL METODO DI AGGIUNGERE RELAZIONI ALLE FIGLIE CON IL PADRE.  
MA VISTO CHE NON AVREBBE SENSO MANTENERE LA SOTTOCLASSE **PRIVATA** POICHÉ PRIVA DI ATTRIBUTI E RELAZIONI, LA INCLUDIAMO NELL'ENTITÀ PLAYLIST.



L'entità **Pubblica** diventa un'entità debole.

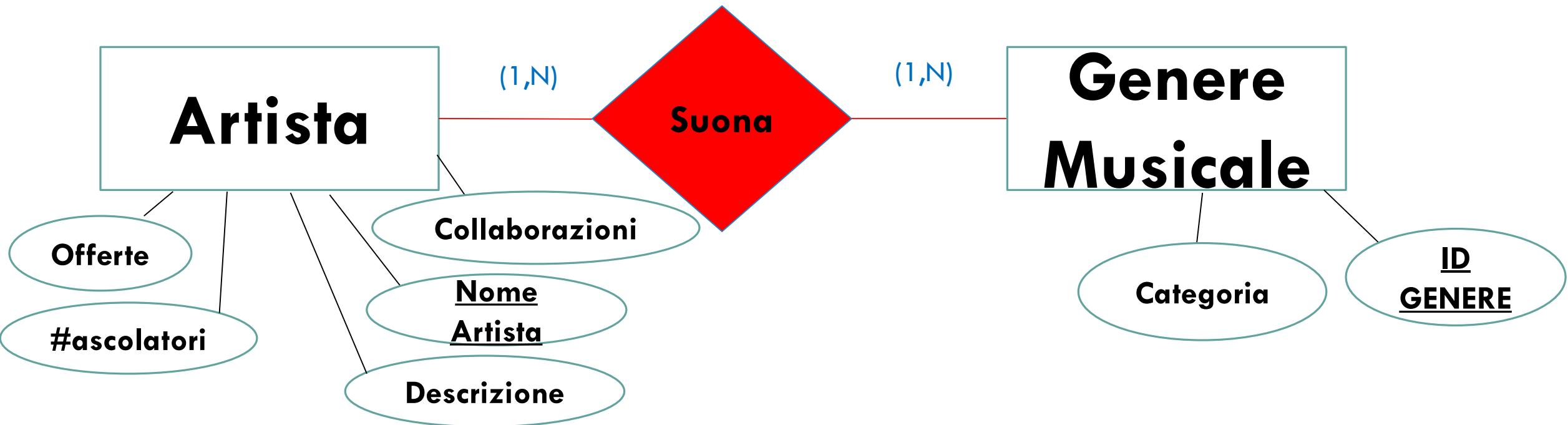
La relazione tra **Playlist** e **Pubblica** è **ESSERE**, una playlist può essere o non essere pubblica e almeno una playlist pubblica deve esistere.

# PARTIZIONAMENTO O ACCORPAMENTO DI CONCETTI

L'attributo **multivalore** GENERE MUSICALE dell'entità Artista viene trasformato in una nuova entità. Tale entità ha come attributi: ID GENERE (int) che rappresenta la **chiave** e l'attributo Categoria(string).

Le entità **Artista** e **Genere Musicale** sono legate dalle relazione **SUONARE**.

Un Artista deve suonare almeno 1 o più generi musicali e ogni genere musicale deve essere suonato da 1 o più artisti.



# SCELTA DEGLI ID

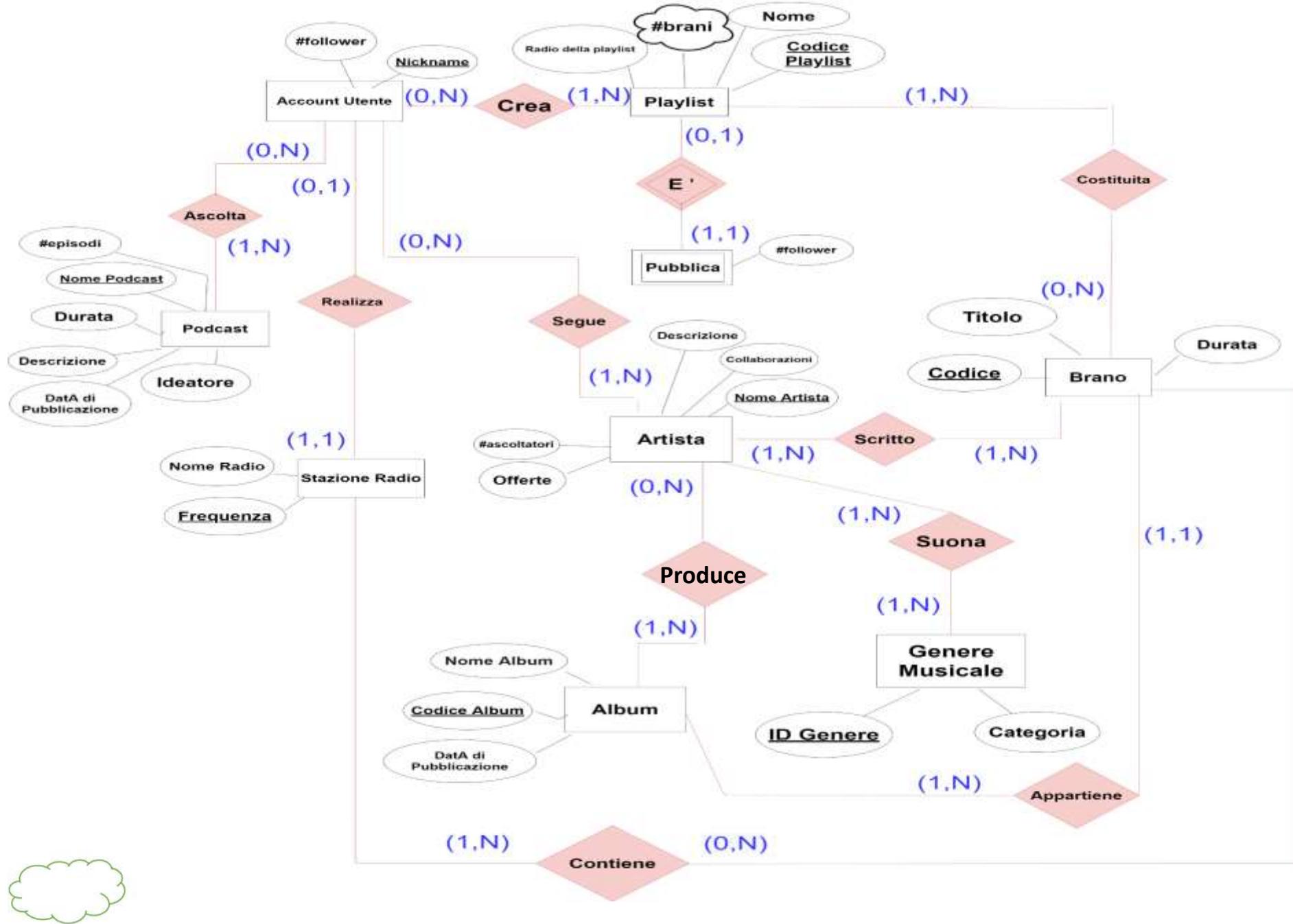
## PRINCIPALI



Nell'entità Podcast abbiamo due attributi **CHIAVE** che sono:

Nome Podcast e Ideatore. La scelta della chiave primaria cade su **Nome Podcast** perché è più veloce da controllare rispetto a Ideatore.

Inoltre non possono esistere podcast con lo stesso nome.



Con il seguente simbolo sono stati  
definiti gli attributi ridondanti

# MAPPING

Account Utente(Nickname, #follower) ;

Creare(Account Utente.Nickname , Playlist.Codice Playlist  ) ;

Playlist( Codice Playlist, Nome, Radio della Playlist, #brani) ;

Pubblica(#followerplaylist,Playlist.Codice Playlist  ) ;

Costituire(Playlist.Codice Playlist , Brano.Codice  ) ;

Brano( Codice, Titolo, Durata, Album.Codice Album  ) ;

Scrivere(Brano.Codice , Artista.Nome Artista  ) ;

Artista( Nome Artista, Collaborazioni, Descrizione, #ascoltatori, Offerte) ;

Seguire(Artista.Nome Artista , Account Utente.Nickname  ) ;

Produce( Artista.Nome Artista , Album.Codice Album  ) ;

Album (Codice Album, Nome Album, Data di Pubblicazione) ;

Suona(Artista.Nome Artista , Genere Musicale.ID Genere  ) ;

Genere Musicale ( ID Genere, Categoria) ;

Contiene(Brano.Codice , Stazione Radio.Frequenza  ) ;

Stazione Radio( Frequenza, Nome Radio, Account Utente.Nickname  ) ;

Ascolta(Account Utente.Nickname , Podcast.Nome Podcast  ) ;

Podcast( Nome Podcast, Ideatore, Data di Pubblicazione, Descrizione, Durata, #episodi) ;



# NORMALIZZAZIONE

## 1NF,2NF,3NF

1NF: LA RELAZIONE NON DEVE PRESENTARE ATTRIBUTI MULTIVALORE, COMPOSTI O UNA LORO QUALSIASI COMBINAZIONE.

LA NORMALIZZAZIONE SI EFFETTUA FORMANDO UNA NUOVA RELAZIONE PER OGNI ATTRIBUTO NON ATOMICO

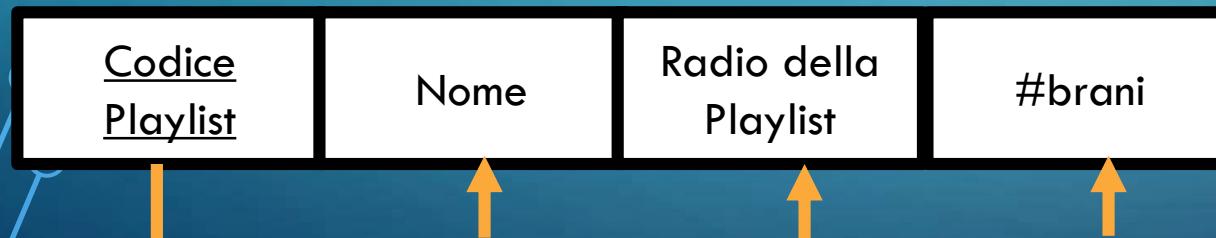
## ACCOUNT UTENTE:



E' già in 1NF, 2NF e 3NF

Nickname-> #follower

## PLAYLIST:



E' già in 1NF, 2NF e 3NF

Codice Playlist-> {Nome, Radio della Playlist, #brani}

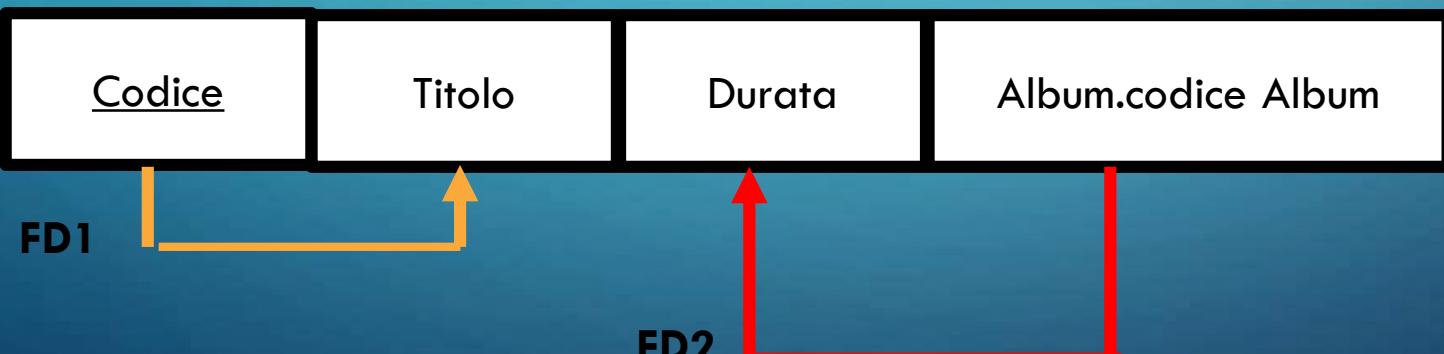
## PUBBLICA:



E' già in 1NF, 2NF e 3NF

Playlist.codice Playlist-> #follower

## BRANO:

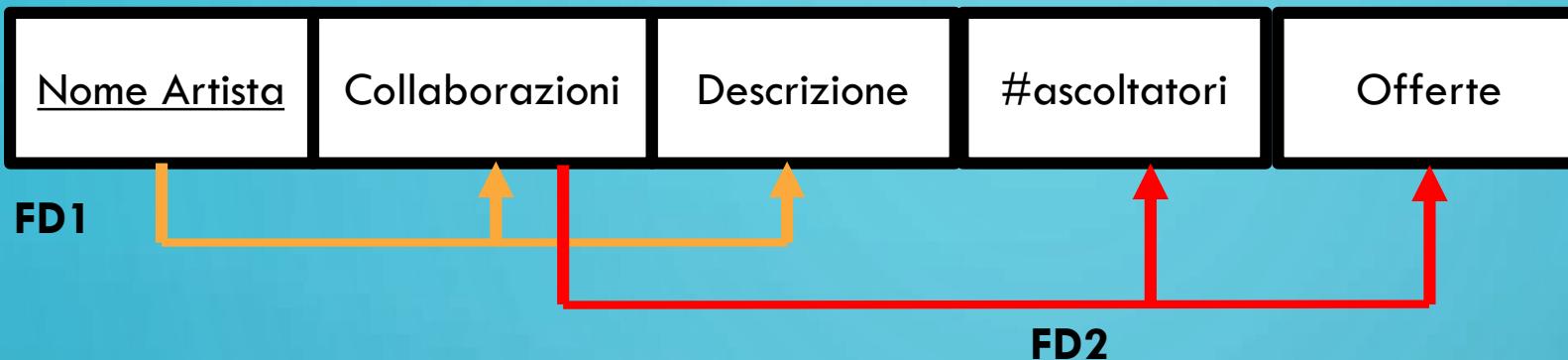


E' già in 1NF

Codice -> Titolo

Album.codice Album -> Durata

## ARTISTA:

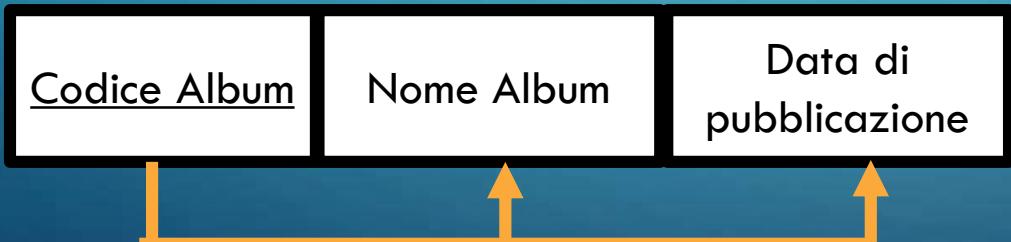


E' già in 1NF E 2NF

**Nome Artista** → {**Collaborazioni, Descrizione,** }

**Collaborazioni** → {**#ascoltatori, Offerte**}

## ALBUM:



E' già in 1NF, 2NF e 3NF

**Codice Album** → {**Nome Album, Data di pubblicazione**}

## GENERE MUSICALE:

<u>ID Genere</u>	Categoria
------------------	-----------

**ID Genere -> Categoria**

**E' già in 1NF, 2NF e 3NF**

## STAZIONE RADIO:

<u>Frequenza</u>	Nome Radio	Account.Nickname
------------------	------------	------------------

**Frequenza -> {Nome Radio, Account. Nickname }**

**E' già in 1NF, 2NF e 3NF**

## PODCAST:

<u>Nome Podcast</u>	Ideatore	Descrizione	Durata	#episodi	Data
FD1			FD2		FD3

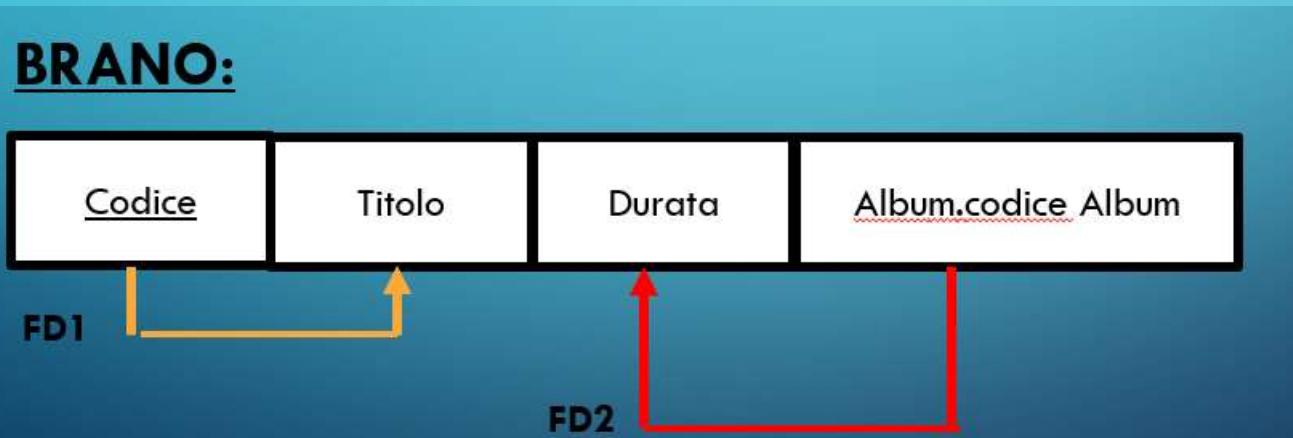
**Nome Podcast -> {Ideatore, Descrizione}**

**Data->#episodi**

**# episodi->Durata**

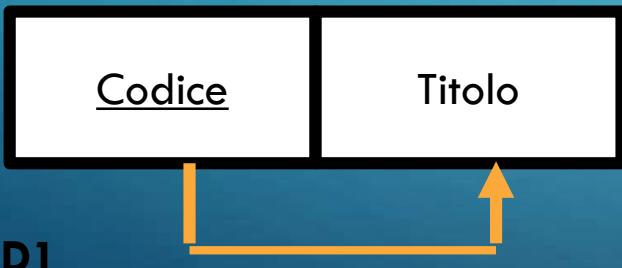
**E' già in 1NF**

2NF: TALE FORMA NORMALE SI BASA SULLA DIPENDENZA FUNZIONALE PIENA.  
LA NORMALIZZAZIONE SI EFFETTUÀ CREANDO UNA RELAZIONE PER OGNI CHIAVE  
PARZIALE CON I SUOI ATTRIBUTI DIPENDENTI.



Tale entità non rispetta le condizioni della 2NF perché l'attributo DURATA non è pienamente dipendente dalla chiave.

B1

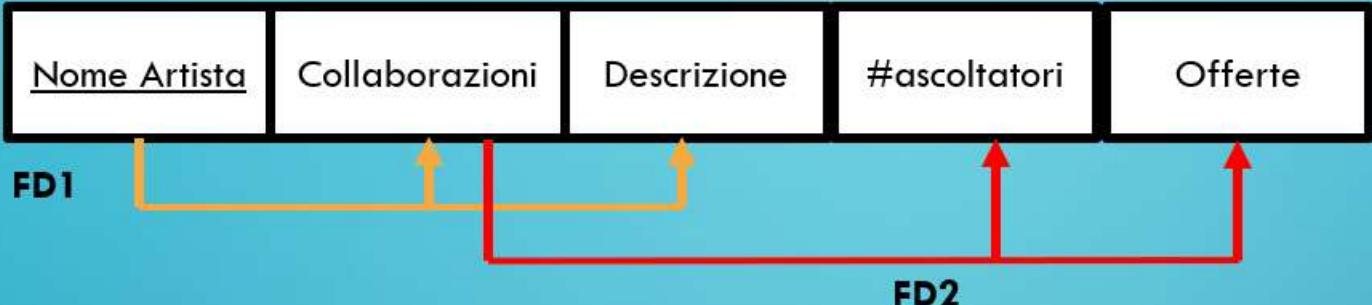


B2



E' già in 3NF

## ARTISTA:

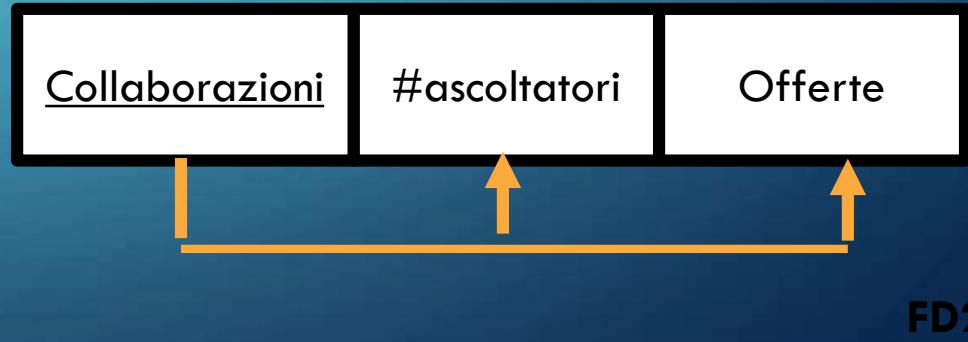


Tale schema rispetta la 2NF ma non le condizioni della 3NF perché gli attributi #ASCOLTATORI e OFFERTE presentano una dipendenza funzionale transitiva.

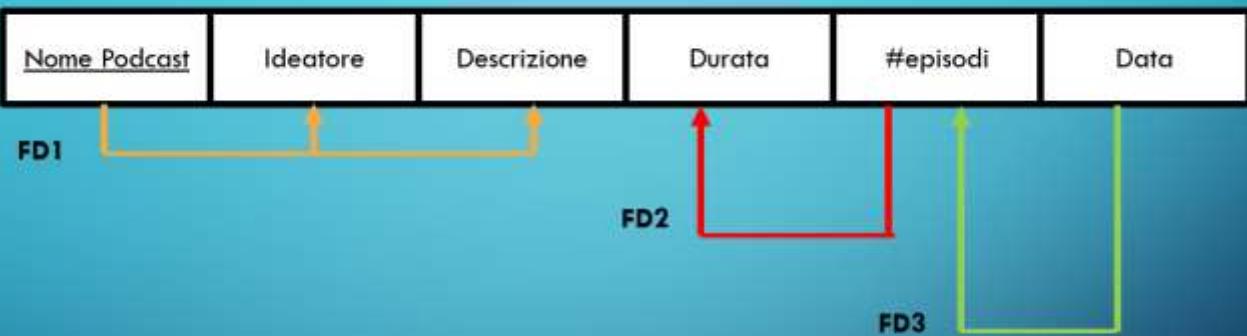
A1



A2



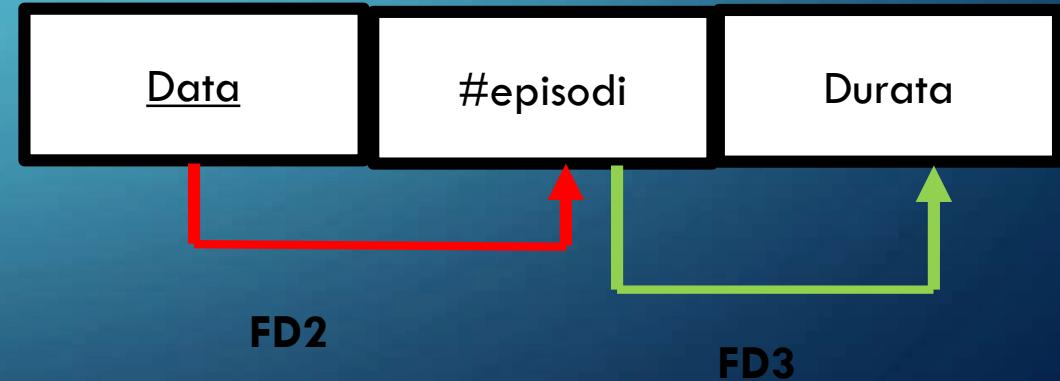
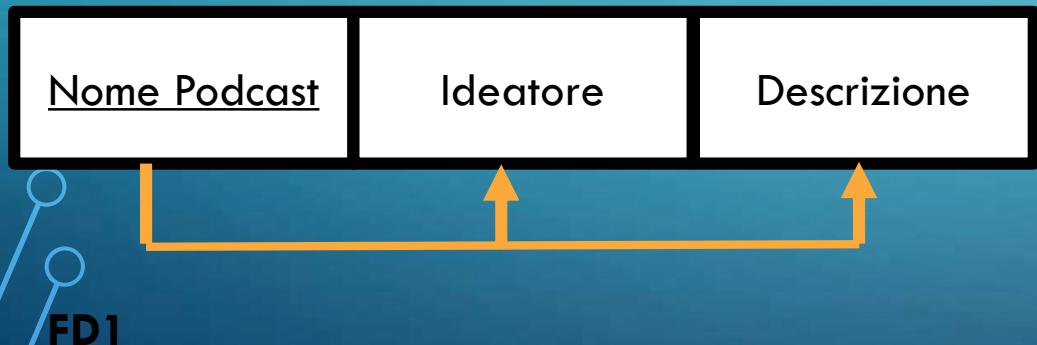
## PODCAST:

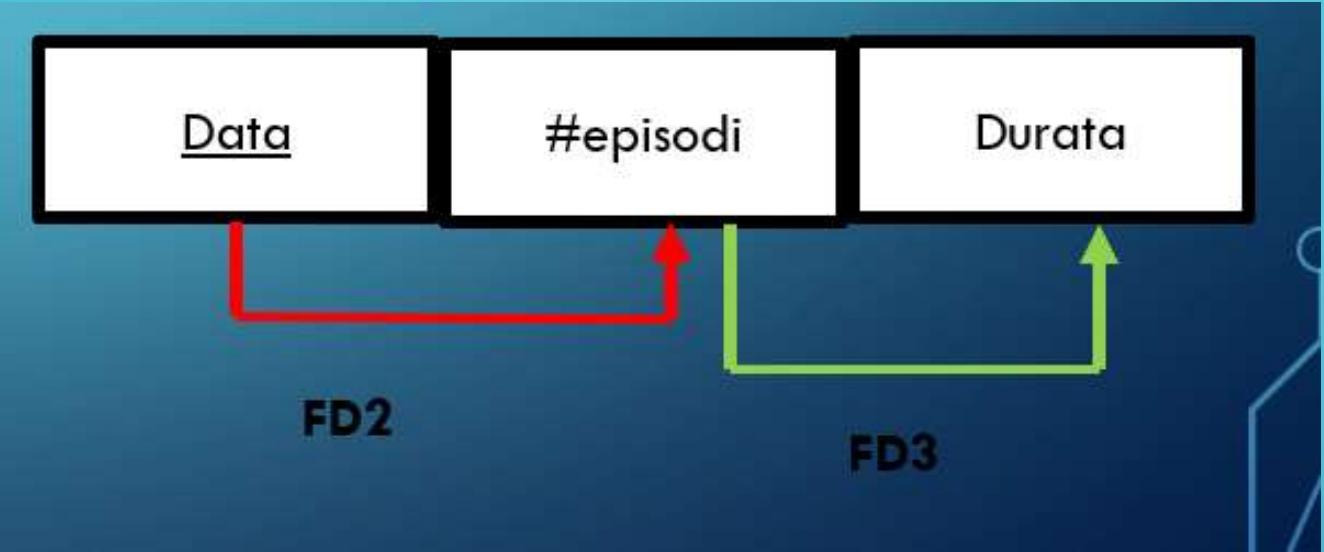


Nome Podcast → {Ideatore, Descrizione}  
Data → #episodi  
# episodi → Durata

E' già in 1NF

Tale entità non rispetta le condizioni della 2NF perché l'attributo DATA non è pienamente dipendente dalla chiave.





Tale entità non rispetta le condizioni della 3NF perché l'attributo DURATA presenta una dipendenza funzionale transitiva.



## TAVOLA DEI VOLUMI MODIFICATA

\*i valori definiti precedentemente poiché troppo elevati sono stati sostituiti con valori più bassi per poter garantire una buona gestione del nostro database.

**\*La scelta dei valori è stata effettuata nel medesimo modo della precedente TABELLA DEI VOLUMI(slides 8).**

CONCETTO	TIPO	VOLUME
Account Utente	E	4
Crea	R	8
Playlist	E	8
Pubblica	E	2
Ascolta	R	2
Podcast	E	3
Segue	R	5
Artista	E	10
Realizza	R	1
Stazione Radio	E	1
Contiene	R	3
Brano	E	10
Costituito	R	36
Produce	R	10
Album	E	10
Appartiene	R	2
Scritto	R	10
Suona	R	10
Genere	E	7

# QUERY IN SQL

```
types.Operator):
    X mirror to the selected
    object.mirror_mIRROR_X"
    "rror X"
```

```
int("please select exactly one object")
```

```
-- OPERATOR CLASSES --
```

```
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

selection at the end -add
ob.select= 1
ler_ob.select=
context.selected_objects.active
("Selected" str(modifier))
irror_ob.select = 0
 bpy.context.selected_objects[one]
data.objects[one.name].sele
```

- 1)Inserire un nuovo account utente.
- 2)Selezionare il nome della playlist e il #brani creati da un account utente con #follower maggiore di 100.
- 3)Selezionare gli artisti , nome artista, descrizione, collaborazioni, che suonano come genere musicale ‘pop’ o ‘classico’.
- 4)Selezionare il nome album e il codice album con data di pubblicazione relativa all’anno 2020,  
ordinando in maniera crescente i nomi degli album.
- 5)Indicare il numero delle playlist che contengono almeno 1 brano scritto da ‘Geolier’.
- 6)Indicare i nomi degli artisti avente il maggior numero di ascoltatori e che ha prodotto almeno 1 album.
- 7)Calcolare il numero medio di episodi relativi ad ogni podcast.
- 8)Selezionare i codici e I titoli dei brani che non sono mai stati scritti da Laura Pausini.
- 9)Indicare il Nickname dell’account utente che segue un’artista che suona il genere indie o ascolta podcast.
- 10)Indicare tutte le playlist costituite da brani che non sono contenuti in una stazione radio.



# TABELLE:

```
1 • DROP database IF EXISTS musicdb;
2 • CREATE database musicdb;
3
4 /*DROP USER [IF EXISTS] 'airuser'@'localhost' ; */
5 • CREATE USER 'airuser'@'localhost' IDENTIFIED BY 'airuser';
6 • GRANT ALL ON musicdb.* TO 'airuser'@'localhost';
7
8 • USE musicdb;
9
10
11 • CREATE TABLE AccountUtente (
12     Nickname varchar(50) not null,
13     NumeroFollower int not null,
14     primary key (Nickname)
15 );
16
17 • load data local infile 'accountutente.sql'
18     into table AccountUtente(Nickname, NumeroFollower);
19
20 • CREATE TABLE Playlist (
21     CodicePlaylist int not null,
22     Nome varchar(20) not null,
23     RadiodellaPlaylist varchar(20),
24     NumeroBrani int not null,
25     primary key(CodicePlaylist)
26 );
27
28 • load data local infile 'playlist.sql'
29     into table Playlist(CodicePlaylist,Nome,RadiodellaPlaylist,NumeroBrani);
30
31 • CREATE TABLE Creare (
32     Nickname varchar(50),
33     CodicePlaylist int not null,
34     primary key(Nickname,CodicePlaylist),
35     foreign key(Nickname) references AccountUtente (Nickname),
36     foreign key(CodicePlaylist) references Playlist (CodicePlaylist)
37 );
38
39 • load data local infile 'creare.sql'
40     into table Creare(Nickname,CodicePlaylist);
```

```
42 • Ⓜ CREATE TABLE Pubblica (
43     NumeroFollowerPlaylist int not null,
44     CodicePlaylist int not null,
45     primary key(CodicePlaylist),
46     foreign key(CodicePlaylist) references Playlist (CodicePlaylist)
47 );
48
49 • load data local infile 'pubblica.sql'
50     into table Pubblica(NumeroFollowerPlaylist,CodicePlaylist);
51
52 • Ⓜ CREATE TABLE Album (
53     CodiceAlbum int not null,
54     NomeAlbum varchar(30) not null,
55     DatadiPubblicazione date not null,
56     primary key(CodiceAlbum)
57 );
58
59 • load data local infile 'album.sql'
60     into table Album(CodiceAlbum,NomeAlbum,DatadiPubblicazione);
61
62 • Ⓜ CREATE TABLE Brano (
63     Codice int not null,
64     Titolo varchar(50) not null,
65     Durata float,
66     CodiceAlbum int not null,
67     primary key(Codice),
68     foreign key(CodiceAlbum) references Album(CodiceAlbum)
69 );
70
71 • load data local infile 'brano.sql'
72     into table Brano(Codice,Titolo,Durata,CodiceAlbum);
73
74 • Ⓜ CREATE TABLE Costituire (
75     CodicePlaylist int not null,
76     CodiceBrano int not null,
77     primary key(CodicePlaylist,CodiceBrano),
78     foreign key(CodicePlaylist) references Playlist(CodicePlaylist),
79     foreign key(CodiceBrano) references Brano(Codice)
80 );
81
82 • load data local infile 'costituire.sql'
83     into table Costituire(CodicePlaylist,CodiceBrano);
```

```
85 • CREATE TABLE Artista (
86     NomeArtista varchar(20) not null,
87     Descrizione varchar(100),
88     Offerta char(10),
89     Collaborazioni char(30),
90     NumeroAscoltatori int not null,
91     primary key(NomeArtista)
92 );
93
94 • load data local infile 'artista.sql'
95     into table Artista(NomeArtista,Descrizione,Offerta,Collaborazioni,NumeroAscoltatori)
96
97 • CREATE TABLE Scrivere (
98     CodiceBrano int not null,
99     NomeArtista varchar(20) not null,
100    primary key(NomeArtista,CodiceBrano),
101    foreign key(NomeArtista) references Artista(NomeArtista),
102    foreign key(CodiceBrano) references Brano(Codice)
103 );
104
105 • load data local infile 'scrivere.sql'
106     into table Scrivere(CodiceBrano,NomeArtista);
108 • CREATE TABLE Seguire (
109     NomeArtista varchar(20) not null,
110     Nickname varchar(50) not null,
111     primary key(NomeArtista,Nickname),
112     foreign key(NomeArtista) references Artista(NomeArtista),
113     foreign key(Nickname) references AccountUtente(Nickname)
114 );
115
116 • load data local infile 'seguire.sql'
117     into table Seguire(NomeArtista,Nickname);
118
119 • CREATE TABLE Produce (
120     NomeArtista varchar(20) not null,
121     CodiceAlbum int not null,
122     primary key(NomeArtista,CodiceAlbum),
123     foreign key(NomeArtista) references Artista(NomeArtista),
124     foreign key(CodiceAlbum) references Album(CodiceAlbum)
125 );
126
127 • load data local infile 'produce.sql'
128     into table Produce(NomeArtista,CodiceAlbum);
```

```
130 • Ⓜ CREATE TABLE GenereMusicale (
131     IDGenere int not null,
132     Categoria char(20) not null,
133     primary key(IDGenere)
134 );
135
136 • load data local infile 'generemusicale.sql'
137     into table GenereMusicale(IDGenere,Categoria);
138
139
140 • Ⓜ CREATE TABLE Suona (
141     NomeArtista varchar(20) not null,
142     IDGenere int not null,
143     primary key(NomeArtista,IDGenere),
144     foreign key(NomeArtista) references Artista(NomeArtista),
145     foreign key(IDGenere) references GenereMusicale(IDGenere)
146 );
147
148 • load data local infile 'suona.sql'
149     into table Suona(NomeArtista,IDGenere);
151 • Ⓜ CREATE TABLE StazioneRadio (
152     Frequenza float not null,
153     NomeRadio varchar(20) not null,
154     Nickname varchar(50) not null,
155     primary key(Frequenza),
156     foreign key(Nickname) references AccountUtente(Nickname)
157 );
158
159 • load data local infile 'stazioneradio.sql'
160     into table StazioneRadio(Frequenza,NomeRadio,Nickname);
161
162
163 • Ⓜ CREATE TABLE Contiene (
164     CodiceBrano int not null,
165     FrequenzaRadio float not null,
166     primary key(CodiceBrano,FrequenzaRadio),
167     foreign key(CodiceBrano) references Brano(Codice),
168     foreign key(FrequenzaRadio) references StazioneRadio(Frequenza)
169 );
170
171 • load data local infile 'contiene.sql'
172     into table Contiene(CodiceBrano,FrequenzaRadio);
```

```
174 • CREATE TABLE Podcast (
175     NomePodcast varchar(30) not null,
176     Ideatore char(20) not null,
177     Descrizione varchar(100),
178     Durata float not null,
179     NumeroEpisodi int not null,
180     DataEp date not null,
181     primary key(NomePodcast)
182 );
183
184 • load data local infile 'podcast.sql'
185     into table Podcast(NomePodcast,Ideatore,Descrizione,Durata,NumeroEpisodi,DataEp);
186
• CREATE TABLE Ascolta (
    Nickname varchar(50) not null,
    NomePodcast varchar(30) not null,
    primary key(Nickname,NomePodcast),
    foreign key(Nickname) references AccountUtente(Nickname),
    foreign key(NomePodcast) references Podcast(NomePodcast)
);
• load data local infile 'ascolta.sql'
    into table Ascolta(Nickname,NomePodcast);
```

# QUERY

1) Inserire un nuovo account utente.

**INSERT INTO AccountUtente (Nickname,NumeroFollower) VALUES ("ll.ferri",200);**

Nickname	NumeroFollower
g.aniello	200
k.buonocore	300
ll.ferri	200
m.coiro	200
r.cuccaro	300

2) Selezionare il nome della playlist e il #brani creata da un account utente con follower maggiore di 100.

```
SELECT Nome,NumeroBrani  
FROM Playlist, Creare, AccountUtente  
WHERE Playlist.CodicePlaylist = Creare.CodicePlaylist AND  
Creare.Nickname = AccountUtente.Nickname AND  
NumeroFollower > 100;
```

nome	numerobrani
Jack	3
Hel	1
Mhh	5
KB	7
MC	2
Come	8
Rita	6
Emis	4

### **3) Selezionare gli artisti , nome artista, descrizione, collaborazioni, che suonano come genere musicale ‘pop’ o ‘classico’.**

```
SELECT Artista.NomeArtista, Descrizione, Collaborazioni
```

```
FROM Artista, Suona ,Genere Musicale
```

```
WHERE Artista.NomeArtista = Suona.NomeArtista AND
```

```
Suona.IDGenere= GenereMusicale.IDGenere AND
```

```
(Categoria = “pop” OR
```

```
Categoria = “classico”);
```

NomeArtista	Descrizione	Collaborazioni
Achille Lauro	Achille Lauro: icona della musica e della moda.	Gow tribe
Irama	NULL	NULL
Jovanotti	Dagli anni 80 è la figura più importante che emerge nella musica pop italiana.	Dardust
Laura Pausini	Il 1993 è l'anno in cui Laura Pausini vince il Festival di Sanremo con La Solitudine.	NULL
TheGiornalisti	I TheGiornalisti sono la più importante rivelazione della musica italiana, hanno dato il via al nuov	NULL
Marcella Bella	NULL	NULL

### **4) Selezionare il nome album e il codice album con data di pubblicazione relativa all’anno 2020, ordinando in maniera crescente i nomi degli album.**

```
SELECT NomeAlbum , CodiceAlbum
```

```
FROM Album
```

```
WHERE DatadiPubblicazione LIKE ‘2020-__-__’
```

```
ORDER BY NomeAlbum ASC ;
```

NomeAlbum	CodiceAlbum
1969	100
Crepe	302
Emanuele	400
Gemelli	200
Napoli51	670

**5)Indicare il numero delle playlist che contengono almeno 1 brano scritto da ‘Geolier’.**

```
SELECT Brano.Codice, COUNT (*) AS NumeroPlaylist  
FROM Costituire, Brano, Scrivere  
WHERE Costituire.CodiceBrano = Brano.Codice AND  
Brano.Codice = Scrivere.CodiceBrano AND  
NomeArtista = "Geolier"  
  
GROUP BY Codice  
HAVING COUNT (*) >= 1;
```

Codice	NumeroPlaylist
7	3

**6)Indicare i nomi degli artisti avente il maggior numero di ascoltatori e che ha prodotto almeno 1 album.**

```
SELECT Artista.NomeArtista, MAX(NumerOAscoltatori) AS MaxAscoltatori  
FROM Artista, Produce  
WHERE Artista.NomeArtista = Produce.NomeArtista  
GROUP BY NomeArtista  
HAVING COUNT (*) >=1;
```

NomeArtista	MaxAscoltatori
Achille Lauro	1000000
Jovanotti	1000000
Marcella Bella	100000
Franco126	1000000
Ernia	200000
Irama	2000000
Geolier	2000000
Nicola Siciliano	700000
Laura Pausini	5000000
TheGiornalisti	900000

## 7)Calcolare il numero medio di episodi relativi ad ogni podcast.

```
SELECT NomePodcast, AVG(NumerEpisodi) AS NumeroMedioEpisodi  
FROM Podcast  
GROUP BY NomePodcast;
```

NomePodcast	NumeroMedioEpisodi
Muschio Selvaggio	41.0000
Racconti da ascoltar	20.0000
Rilassamento	10.0000

## 8)Selezionare i codici e i titoli dei brani che non sono mai stati scritti da Laura Pausini.

```
SELECT Codice, Titolo  
FROM Brano  
WHERE NOT EXISTS ( SELECT *  
                   FROM Scrivere  
                   WHERE NomeArtista = "Laura Pausini" AND  
                         Scrivere.CodiceBrano = Brano.Codice ) ;
```

codice	titolo
2	Completamente
3	16 Marzo
4	Nuova Era
5	Montagne Verdi
6	Superclassico
7	Moncler
8	Crepe
9	Collera
10	Stanza Singola

**9)Indicare il Nickname dell'account utente che segue un'artista che suona il genere indie o ascolta podcast.**

```
SELECT Nickname  
FROM Seguire, Artista, Suona, Genere Musicale  
WHERE Seguire.NomeArtista = Artista.NomeArtista AND  
      Artista.NomeArtista = Suona.NomeArtista AND  
      Suona.IDGenere = GenereMusicale.IDGenere AND  
      Categoria = "Indie"
```

UNION

```
SELECT Nickname  
FROM Ascolta ;
```

nickname
m.coiro
g.aniello

**10)Indicare tutte le playlist costituite da brani che non sono contenuti in una stazione radio.**

```
SELECT CodicePlaylist  
FROM Playlist  
WHERE NOT EXISTS ( SELECT *  
                   FROM Costituire, Brano, Contiene  
                   WHERE Costituire.CodiceBrano = Brano.Codice AND  
                         Brano.Codice = Contiene.CodiceBrano AND  
                         Costituire.CodicePlaylist = Playlist.CodicePlaylist ) ;
```

codiceplaylist
19