Progetto di Basi Di Dati 2020

A cura di:
Marco Bustaffa
Matteo Noro

1.Abstract

Waver nasce, dall'intuito di Dr. Markus, come un nuovo servizio di streaming musicale a basso costo. Il servizio disponibile sia via App che via browser necessita chiaramente di un efficiente DataBase per curare lo storage della grande mole di canzoni al suo interno, quali possono essere chiaramente raccolte in album e curate con un featuring. Chiunque può iscriversi al servizio e dedicarsi all'ascolto di musica con un limite giornaliero di un'ora a meno di una iscrizione ad abbonamento che permette ascolto senza limiti e la possibilità di aggiungere singoli brani alla propria libreria. Al servizio, oltre agli utenti consumer, vi sono altre due categorie di utenti quali artisti e responsabili: i primi sono il fulcro della distribuzione musicale, coloro che producono i brani ed eventualmente, per pubblicare un album, si affiliano ai secondi che si identificano nella casa discografica.
Quest'ultima può oltretutto produrre podcast e organizzare concerti online, nati dalla necessità di collegare le persone da tutto il mondo in questa situazione di difficoltà.
Ogni utente premium può aggiungere singoli brani alle playlists che può man mano creare e l'insieme di queste si identificano nell'intera libreria personale. Vista la crescente necessità delle persone di poter condividere con un click i propri interessi, waver permette inoltre di poter rendere pubblica una propria playlist in modo da rendere partecipi gli altri utenti dei propri gusti musicali.

2.Analisi dei Requisiti

Si vuole realizzare una base di dati per l'organizzazione e supporto di un servizio di streaming musicale chiamato Waver.
In particolare un individuo può iscriversi come **Utente** free o in abbonamento Premium, decidendo quindi di usufruire, o meno, di tutti i servizi integrati in Waver, come per esempio l'ascolto illimitato e la possibilità di aggiungere brani alla propria libreria.
Indipendentemente dal tipo di abbonamento scelto, l'utente dovrà registrare i seguenti dati:
-Codice Fiscale
-Email
-Password
-Nome
-Cognome
-Data di Nascita
Nel caso un utente si registri come **Artista** dovrà inserire due ulteriori dati oltre a quelli sopracitati ovvero il Codice della su Partita IVA valido ed eventualmente il nome d'arte che lo contraddistingue.
L'artista è il creatore del cuore principale del servizio, ovvero le **Canzoni,** alle quali possono partecipare altri creator tramite featuring, e sono contraddistinte da:
-Titolo Canzone
-Durata
-Genere
-Data Rilascio
Un Utente **Responsabile**, invece, deve sempre inserire i dati comuni ad Utente Consumer e Artista e in più anche un recapito telefonico; un responsabile inoltre avrà anche uno stipendio.
Quest'ultimo ha il compito di dirigere la **Casa Discografica**, la quale avrà:
-Nome
-Data di fondazione
-Indirizzo di Sede (se fisica)
La Casa Discografica si occupa di rilasciare diversi contenuti quali Album, Podcast e Concerti Online.
Gli **Album** raccolgono singoli brani di un Artista in affiliazione con una Casa Discografica e sono contraddistinti da:
-Nome
-Data rilascio
-Genere
-Durata
-Posizione Classifica
Questi non possono essere aggiunti alle playlist se non selezionando i singoli brani che raccolgono.
I **Podcast** invece sono la nuova frontiera della condivisione di interessi e discussioni che va a sostituire le trasmissioni Radiofoniche permettendo, anche l'ascolto in differita e in alta qualità.
Questi sono caratterizzati da:
-Titolo Podcast
-Descrizione
-Data Creazione
E raggruppano gli **Episodi**, diversificati per:
-Titolo episodio
-Durata
-Riassunto

Infine i **Concerti Online** sono una enorme opportunità offerta dal servizio, che permette così agli utenti di seguire performance dei propri artisti preferiti comodamente da casa.
A questi eventi possono accedere utenti con qualsiasi sottoscrizione pagando un biglietto virtuale soggetto a disponibilità.
Ogni Concerto Online avrà quindi:
-Titolo Concerto
-Numero posti massimo
-Data concerto
-Prezzo biglietto singolo
Un utente premium ha la possibilità di seguire ma soprattutto creare **Playlist** (private o pubbliche) aggiungendoci singole canzoni disponibili nel servizio andando a creare così la sua libreria personale. Ogni playlist avrà quindi:
-Nome Playlist
-Genere principale
-Data Creazione
Se un utente decide di rendere pubblica una sua playlist, questa può essere seguita da altre persone così da creare una rete di condivisione musicale.

2.1 Glossario dei Termini

TERMINE	DESCRIZIONE	SINONIMO	COLLEGAMENTO
Utente Free* *presente solo nello schema logico	Persona iscritta al servizio musicale e pertanto fruitore. La seguente sottoscrizione presenta limiti quali limite di ascolto e impossibilità di aggiungere brani alle playlist, nonché di poterne seguire altre. Può, però, seguire i Concerti Online.	Cliente, fruitore, persona, individuo	Concerto Online
Utente Premium* *presente solo nello schema logico	Persona iscritta al servizio musicale e pertanto fruitore. Non presenta i limiti dell'utente free.	Abbonato, Cliente, fruitore, persona, individuo,	Playlist, Episodio, Concerto Online
Abbonato	Persona iscritta al servizio musicale Waver. Può sottoscrivere o meno un abbonamento premium che gli consente di fruire del servizio senza alcun limite di ascolto e di funzionalità. Ha inoltre la possibilità di seguire Concerti Online.	Utente, fruitore, persona, individuo, Cliente	Playlist, Episodio, Concerto Online
Responsabile	Individuo con il compito di dirigere la Casa Discografica	Dirigente	Casa Discografica, Cellulare
Cellulare	Recapito telefonico di un Responsabile	Numero di telefono, Recapito, Telefonino, Smartphone	Responsabile
Artista	Utente creatore di Canzoni, può appoggiarsi ad una casa discografica per pubblicare album e partecipare ad altre creazioni.	Pubblicatore, Creator, Musicista	Canzone, Concerto Online, Casa Discografica
Canzone	Contenuto base di ascolto del servizio, pubblicato da un Artista ed eventualmente parte di un album previa firma con una Casa Discografica	Brano, Contenuto, Creazione	Album, Artista, Playlist

Album	Raccoglitore di canzoni create da un Artista, pubblicato da una Casa Discografica previa firma del Creator	Raccolta, Raccoglitore	Canzone, Casa Discografica
Casa Discografica	Sotto direzione di un Responsabile, si occupa della pubblicazione di Album, in collaborazione con gli artisti, Podcast e Concerti Online	Publisher, Etichetta	Album, Responsabile, Podcast, Concerto Online
Concerto Online	Iniziativa da parte di una Casa Discografica in collaborazione con vari artisti per rendere fruibile una performance online agli abbonati	Performance, Live	Casa Discografica, Artista, Abbonato
Podcast	Trasmissione online per la fruizione di contenuti attinenti ai propri interessi, simil trasmissione radiofonica. Si suddivide in Episodi ed è pubblicata da una Casa Discografica	Trasmissione	Episodio, Casa Discografica
Episodio	Contenuto di un Podcast, fruibile dagli Abbonati	Puntata, Parte (di)	Podcast, Abbonato
Playlist	Raccolta di brani selezionati dall'Abbonato che vanno a crearne la libreria. Può essere pubblica o privata.	Raccolta, Libreria (per estensione)	Canzone, Abbonato

2.2.Operazioni sul Database

OPERAZIONE	WRITING / READING	FREQUENZA
Caricamento di Canzoni	W	50.000 al giorno
Caricamento di Album	W	4000 al giorno
Aggiunta canzoni in una Playlist	W/R	3000 al giorno
Iscrizione nuovi Utenti	W	2500 al giorno
Caricamento Episodi di un Podcast	W	300 al giorno
Creazione di nuovi Podcast	W	100 al giorno
Ascolti Episodio	R	500 al giorno
Concerti Online	W	3 al mese
Prenotazione Concerto Online	R/W	600 al giorno

3.Progettazione concettuale:

3.1.Lista delle entità:

Le parole il simbolo * rappresentano gli attributi chiave, sono tutte di tipo Not Null a meno di specifica contraria contraddistinta dalla presenza di una lettera “N” fra parentesi.

ABBONATO			
codiceFiscale*	varchar(16)	Codice fiscale dell’utente	Derivate dall’entità padre “Utente” presente nello schema logico.
nome	varchar(30)	Nome dell’utente	
cognome	varchar(40)	Cognome dell’utente	
email	varchar(60)	Email dell’utente	
password	varchar(100)	Password dell’utente	
bdate	date	Data di nascita dell’utente	
isPremium	boolean	Denota la sottoscrizione scelta, se premium o free	

RESPONSABILE			
codiceFiscale*	varchar(16)	Codice fiscale del resp.	Derivate dall’entità padre “Utente” presente nello schema logico.
nome	varchar(30)	Nome del resp.	
cognome	varchar(40)	Cognome del resp.	
email	varchar(60)	Email del resp.	
password	varchar(100)	Password del resp.	
bdate	date	Data di nascita del resp.	
stipendio	decimal(10,3)	Stipendio del Responsabile	
idCasaDiscografica	Integer	Identificativo della Etichetta sotto amministrazione	

ARTISTA			
codiceFiscale*	varchar(16)	Codice fiscale dell’art.	Derivate dall’entità padre “Utente” presente nello schema logico.
nome	varchar(30)	Nome dell’art.	
cognome	varchar(40)	Cognome dell’art..	
email	varchar(60)	Email dell’art.	
password	varchar(100)	Password dell’art.	
bdate	date	Data di nascita dell’art.	
partitaIVA	varchar(11)	Codice della Partita IVA dell’artista	
idCasaDiscografica	Integer (N)	Identificativo della Etichetta affiliata	

CASA DISCOGRAFICA		
idCasa*	serial	Identificativo della Etichetta
nome	varchar(70)	Nome della Etichetta
dataFondazione	date	Data di fondazione della Etichetta
cap	varchar(5)	CAP della sede dell’etichetta
via	varchar(100)	Nome della via dove presente la sede
numeroCivico	varchar(5)	Numero civico nella via dove presente la sede

PLAYLIST			
idPlaylist*	serial	Identificativo della Playlist	Derivate dall’entità padre “Playlist” presente nello schema logico.
codiceAbbonato	varchar(16)	Codice Fiscale di chi ha creato la Playlist	
nomePlaylist	varchar(70)	Nome della Playlist	
dataCreazione	date	Data creazione della Playlist	
generePlaylist	varchar(50)	Genere principale della Playlist	
isPubblica	boolean	Denota se una Playlist è stata resa pubblica o meno.	Derivato dall’accorpamento delle entità figlie

CELLULARE		
numero*	varchar(10)	Numero di cellulare del Responsabile
codiceResponsabile	varchar(16)	Codice Fiscale del Responsabile

CANZONE		
idCanzone*	serial	Identificativo del brano
codiceArtista	varchar(16)	Codice Fiscale dell'Artista di riferimento
titoloCanzone	varchar(30)	Titolo del brano
dataDrop	date	Data di rilascio del brano
durata	integer	Durata in secondi del brano
genereCanzone	varchar(50)	Genere musicale del brano
idAlbum	Integer (N)	Identificativo dell'eventuale Album contenitore

ALBUM		
idAlbum*	serial	Identificativo dell'Album
idCasaDiscografica	integer	Identificativo della Etichetta pubblicatrice
titoloAlbum	varchar(70)	Titolo dell'Album
dataRilascio	date	Data rilascio dell'Album
posizioneClassifica	integer	Posizione nella classifica dei migliori Album
durataTotale	integer	Durata totale della canzone dell'Album
genereAlbum	varchar(50)	Genere principale dell'Album

PODCAST		
idPodcast*	serial	Identificativo del Podcast
idCasaDiscografica	integer	Identificativo della Etichetta pubblicatrice
titoloPodcast	varchar(30)	Titolo del Podcast
dataCaricamento	date	Data di caricamento del Podcast
riassunto	varchar(1000) (N)	Breve riassunto dell'argomento del Podcast

EPISODIO		
numeroEpisodio*	integer	Numero dell'episodio podcast
idPodcast*	serial	Identificativo dell'episodio podcast
titoloEpisodio	varchar(30)	Nome dell'episodio podcast
durata	integer	Durata dell'episodio podcast in secondi
descrizione	varchar(500) (N)	Breve descrizione del contenuto dell'episodio

CONCERTO ONLINE		
idConcerto*	serial	Identificativo del Concerto
idCasaDiscografica	serial	Identificativo dell'Etichetta pubblicatrice
titoloConcerto	varchar(30)	Titolo del Concerto
dataConcerto	date	Data del Concerto
prezzoBiglietto	decimal(6,2)	Prezzo del biglietto per la visione del Concerto
numeriPostiMax	integer	Numeri di posti virtuali del Concerto

Circa le Generalizzazioni:

- *Utente* è una generalizzazione di Abbonato, Responsabile e Artista; di tipo completo ed esclusiva.
- *Abbonato* è una generalizzazione di Free e Premium; di tipo completo ed esclusiva.
- *Playlist* è una generalizzazione di Privata e Pubblica; di tipo completo ed esclusiva.

Circa i vincoli di integrità:

- *DurataTotale* in Album deve risultare come la somma in secondi della Durata dei singoli brani che lo compongono.
- *dataRilascio* in Album non deve essere minore della dataDrop dell'ultimo brano, pubblicato in ordine cronologico, che lo compone.
- *numeriPostiMax* in Concerto Online determina il massimo numero di biglietti acquistabili per la sua visione, pertanto facendo riferimento alla relazione Guarda (vedi successivamente nella sez.3.2), un Abbonato non può acquistare un biglietto *nPostoOccupato* con valore maggiore di *numeriPostiMax.*

Circa gli attributi composti:

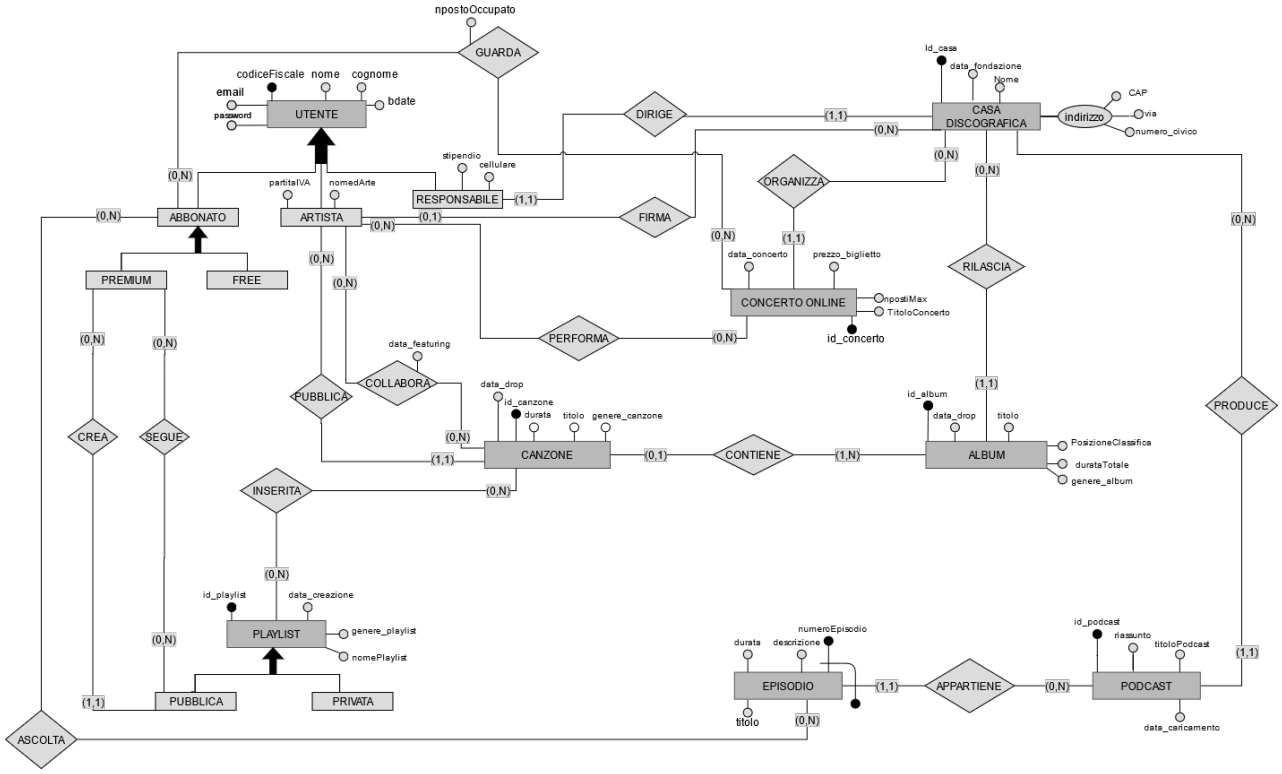
- Casa Discografica fa uso dell'attributo *Indirizzo* quale è composto da CAP, Via e numeroCivico.

3.2.Lista delle relazioni e cardinalità

Si mostrano di seguito le relazioni presenti nello schema ristrutturato con le rispettive cardinalità:

1. **Abbonato – Playlist:** CREA
    a. Un abbonato può creare delle playlist (0,N)
    b. Ogni playlist appartiene ad un solo abbonato (1,1)
2. **Abbonato – Playlist**: SEGUE
    a. Un abbonato può seguire delle playlist pubbliche (0,N)
    b. Una playlist può essere seguita da più persone (0,N)
3. **Abbonato – Episodio:** ASCOLTA
    a. Un abbonato può aver ascoltato diversi episodi podcast (0,N)
    b. Un episodio podcast può esser stato riprodotto da diversi abbonati (0,N)
4. **Abbonato – Concerto Online:** GUARDA
    a. Un abbonato può aver guardato diversi concerti online (0,N)
    Ad ogni concerto a cui partecipa gli viene assegnato un posto [attributo *nPostoOccupato*]
    b. Un concerto online può avere diversi abbonati a visualizzarlo (0,N)
5. **Artista – Canzone:** PUBBLICA
    a. Un artista può pubblicare diverse canzoni (0,N)
    b. Una canzone ha un unico artista come creator principale (1,1)
6. **Artista – Canzone:** COLLABORA
    a. Un artista collabora a diverse canzoni (0,N)
    b. Una canzone ha diversi featuring (0,N)
7. **Artista – Casa Discografica:** FIRMA
    a. Un artista si può affiliare ad una casa discografica (0,1)
    b. Una casa discografica firma con diversi artisti (0,N)
8. **Artista – Concerto Online:** PERFORMA
    a. Un artista può partecipare a più Concerti Online (0,N)
    b. Ad un concerto online possono partecipare più artisti (0,N)
9. **Responsabile – Casa Discografica:** DIRIGE
    a. Un responsabile dirige una casa discografica (1,1)
    b. Una casa discografica è diretta da un solo responsabile (1,1)
10. **Casa Discografica – Concerto Online:** ORGANIZZA
    a. Una etichetta può organizzare diversi concerti online (0,N)
    b. Un concerto online è organizzato da una sola etichetta (1,1)
11. **Casa Discografica – Album:** RILASCIA
    a. Una casa discografica rilascia diversi album (0,N)
    b. Un album è rilasciato da una casa discografica (1,1)
12. **Casa Discografica – Podcast:** PRODUCE
    a. Una casa discografica può pubblicare diversi podcast (0.N)
    b. Un podcast è pubblicato da una sola casa discografica (1,1)
13. **Podcast – Episodio:** APPARTIENE
    a. Un podcast può avere diversi episodi (0,N)
    b. Un episodio appartiene ad un solo podcast (1,1)
14. **Album – Canzone:** CONTIENE
    a. Un album contiene diverse canzoni (1,N)
    b. Una canzone può appartenere o meno ad un album (0,1)
15. **Canzone – Playlist:** INSERITA
    a. Una canzone può essere inserita in diverse playlists (0,N)
    b. Una playlist può contenere diverse canzoni (0,N)

3.3.Schema Concettuale (E-R)



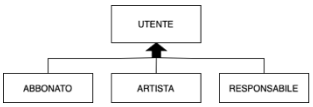
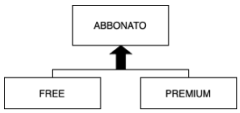
4.Progettazione logica:

4.1.1 Analisi delle ridondanze

La presenza del campo *durata* in Album è ridondante, in quanto si potrebbe ottenere il medesimo dato con la somma delle durate totali dei brani che lo compongono. Si stima però che al giorno vengano caricati all'incirca 4000 album e supponendo una media di 12 canzoni che li compongono si dovrebbe analizzare la durata di 48.000 canzoni ed eseguire le relative somme appropriate.
Proprio per questa motivazione risulta molto più efficiente avere un attributo atomico (una volta impostato non cambia) da specificare al caricamento dell'album nel database; in tal modo sarà necessaria solo una semplice query per visualizzare la durata totale di un album.

4.1.2 Eliminazione delle generalizzazioni

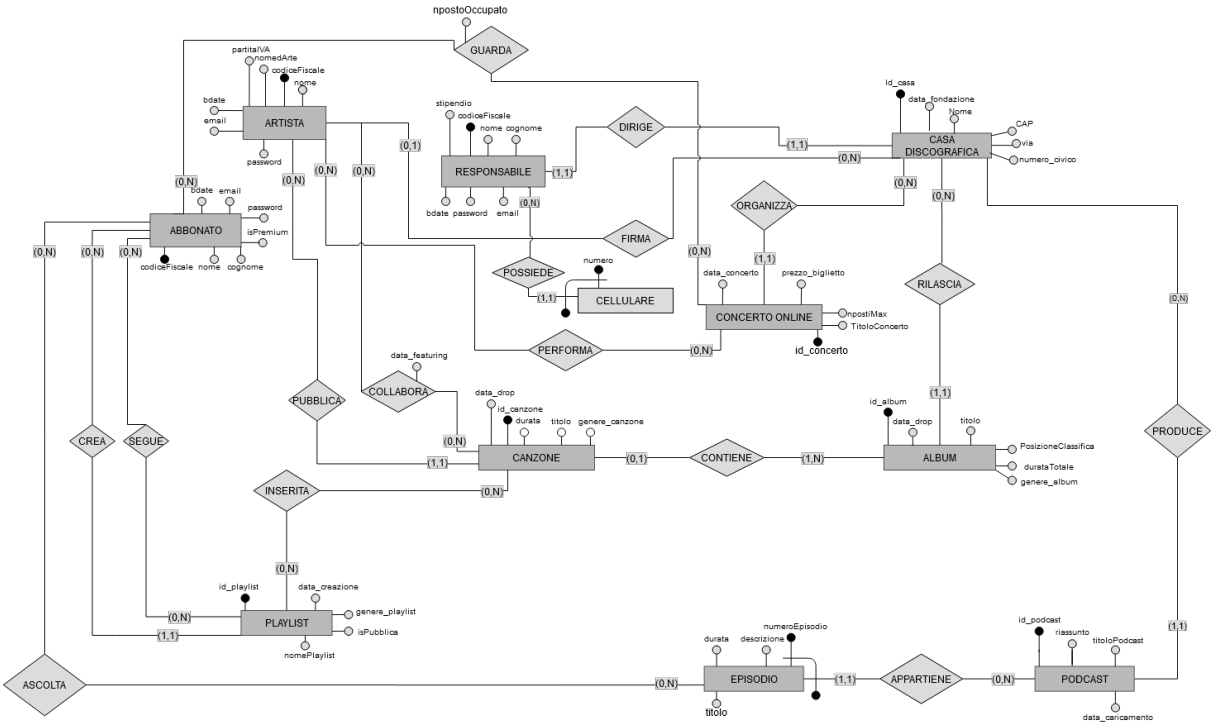
Si motivano di seguito le eliminazioni delle generalizzazioni presenti nello schema E-R (Sez. 3.3)

Generalizzazione	Risultato
	Si è deciso di accorpare l'entità padre *Utente* nelle entità figlie *Artista, Abbonato* e *Responsabile.* Tale scelta è dettata dalla necessità di distinguere le entità figlie per i loro attributi specifici e le relazioni che ne determinano l'operato.
	Si è deciso di accorpare le entità figlie *Free* e *Premium* nella padre *Abbonato*, introducendo di conseguenza un attributo booleano *isPremium* per differenziarle. Tale scelta è motivata dal fatto che le entità figlie hanno attributi comuni e devono solo essere distinte logicamente.
	Si è deciso di accorpare le entità figlie *Pubblica* e *Privata* nella padre *Playlist*, introducendo un attributo booleano *isPubblica* per differenziare le due possibilità. Tale scelta è motivata dal fatto che le entità figlie non presentano alcun attributo proprio ma sono solo una distinzione logica.

4.1.3 Scelta identificatori primari

Un identificatore primario non banale si trova nella tabella *Casa Discografica:* si poteva optare, infatti, per identificare l'Etichetta tramite il suo attributo composto *indirizzo,* contenente i campi *CAP, Via, numero_civico;* ma questa scelta avrebbe comportato la copia di questi tre attributi in tutte le tabelle che fanno riferimento a *Casa Discografica.*
Ad esempio *Album* ha una relazione verso l'entità in esame con cardinalità (1,1) e questo implica che *Album* debba contenere l'identificatore primario di *Casa Discografica,* e se questo fosse stato rappresentato dalla triade di attributi prima citata si sarebbe ottenuta una inefficienza generale, in particolare nell'inserimento dei dati di un Album.
Si è deciso pertanto di inserire un campo numerico *id_casa* come chiave primaria, auto incrementale, in modo da facilitare il riferimento esterno alla tabella.

4.1.4 Diagramma schema ristrutturato



4.2.Descrizione schema relazionale ed eventuali vincoli di integrità referenziale

1. **abbonato** (codiceFiscale, nome, cognome, email, password, bdate, isPremium)
2. **artista** (codiceFiscale, nome, cognome, email, nickname, password, bdate, partitaIVA, idCasaDiscografica)
    a. artista.idCasaDiscografica → casaDiscografica.idCasa
3. **responsabile** (codiceFiscale, nome, cognome, email, password, bdate, stipendio, idCasaDiscografica)
    a. responsabile.idCasaDiscografica → casaDiscografica.idCasa
4. **canzone (**idCanzone, codiceArtista, titoloCanzone, dataDrop, durata, genereCanzone, idAlbum)
    a. canzone.codiceArtista → artista.codiceFiscale
    b. canzone.idAlbum → album.idAlbum
5. **playlist** (idPlaylist, codiceAbbonato, nomePlaylist, dataCreazione, genere Playlist, isPubblica)
    a. playlist.codiceAbbonato → abbonato.codiceFiscale
6. **concertoOnline** (idConcerto, idCasaDiscografica, titoloConcerto, dataConcerto, prezzoBiglietto, numeroPostiMax)
    a. concertoOnline.idCasaDiscografica → casaDiscografica.idCasa
7. **casaDiscografica** (idCasa, nome, dataFondazione, cap, via, numeroCivico)
8. **album** (idAlbum, idCasaDiscografica, dataRilascio, titoloAlbum, posizioneClassifica, durataTotale, genereAlbum)
    a. album.idCasaDiscografica → casaDiscografica.idCasa
9. **podcast** (idPodcast, idCasaDiscografica, titoloPodcast, dataCaricamento, riassunto)

    a. podcast.idCasaDiscografica → casa.idCasa
10. **cellulare** (numero, codiceResponsabile)
    a. cellulare.codiceResponsabile → responsabile.codiceFiscale
11. **episodio** (numeroEpisodio, idPodcast, titoloEpisodio, durata, descrizione)
    a. episodo.idPodcast → podcast.idPodcast
12. **abbonato_playlist** (codiceAbbonato, idPlaylist)
    a. abbonato_playlist.codiceAbbonato → abbonato.codiceFiscale
    b. abbonato_playlist.idPlaylist → playlist.idPlaylist
13. **abbonato_concerto** (codiceAbbonato, idConcerto, npostoOccupato)
    a. abbonato_concerto.codiceAbbonato → abbonato.codiceFiscale
    b. abbonato_concerto.idConcerto → concerto.idConcerto
14. **abbonato_episodio** (codiceAbbonato, numeroEpisodio, idPodcast)
    a. abbonato_episodio.codiceAbbonato → abbonato.codiceFiscale
    b. abbonato_episodio.numeroEpisodio → episodio.numeroEpisodio
    c. abbonato_episodio.idPodcast → podcast.idPodcast
15. **artista_canzone** (codiceArtista, idCanzone, dataFeaturing)
    a. artista_canzone.codiceArtista → artista.codiceFiscale
    b. artista_canzone.idCanzone → canzone.idCanzone
16. **artista_concerto** (codiceArtista, idConcerto)
    a. artista_concerto.codiceArtista → artista.codiceFiscale
    b. artista_concerto.idConcerto → concertoOnline.idConcerto
17. **playlist_canzone** (idPlaylist, idCanzone)
    a. playlist_canzone.idPlaylist → playlist.idPlaylist
    b. playlist_canzone.idCanzone → canzone.idCanzone

5.Query:

Query 1)

I dirigenti della Case Discografiche hanno un enorme margine di guadagno attraverso queste piattaforme e tradizionali metodi di distribuzione musicale. Molto spesso inoltre, alcune Case Discografiche si concentrano nella pubblicazione di materiale attinente ad un genere principale, seguendo la tendenza.
Si vogliono quindi trovare le Case Discografiche che producono solo album Hip Hop e ne abbiamo pubblicati più di due. Inoltre queste devono avere un dirigente che guadagna più di 5000 euro al mese.

```
drop view if exists case_discografiche_con_album_hiphop;
create view case_discografiche_con_album_hiphop as
select a.idCasaDiscografica, a.genereAlbum, count(*) as nalbum
from album as a where a.genereAlbum = 'Hip Hop'
group by (a.idCasaDiscografica, a.genereAlbum)
having a.idCasaDiscografica not in (select a2.idCasaDiscografica
                                    from album as a2
                                    where a2.genereAlbum != 'Hip Hop');
select cds.nome, cds.dataFondazione, cds.cap, cds.via, cds.numeroCivico
from case_discografiche_con_album_hiphop as cdh
join responsabile as r on r.idCasaDiscografica = cdh.idCasaDiscografica
join casaDiscografica as cds on cds.idCasa = cdh.idCasaDiscografica
where r.stipendio >= 5000 and cdh.nalbum >= 2;
```

Creata una view per identificare le Etichette che hanno prodotto album hip hop e non di altro genere, si utilizza questa in una ulteriore query che estrae le Case Discografiche con conteggio degli album hip hop, presenti, maggiore di due e che hanno un dirigente con lo stipendio maggiore della soglia data.

	nome character varying (70)	datafondazione date	cap character varying (5)	via character varying (100)	numerocivico character varying (5)
1	Death Row Records	1991-02-03	58010	Via Pietro Nenni	11
2	Lazy Pug	1990-09-07	32100	Via Mestro caravaggio	21

Query 2)

Per motivi di indagine di mercato, torna spesso utile verificare gli interessi degli utenti circa un determinato argomento, suddivisi per fasce di età. I risultati che si ottengono si possono usare per fare delle *Top Charts,* ovvero delle classifiche suddivise per categoria.
Si vogliono quindi trovare, per esempio, le 5 playlist pubbliche con più canzoni, degli utenti nati prima del 2000.

```
drop view if exists aux;
create view aux as
select pl.idPlaylist
from playlist as pl
join abbonato as ab on pl.codiceAbbonato = ab.codiceFiscale
where pl.isPubblica = true and bdate < '2000-01-01';

select nomePlaylist from playlist
where idPlaylist in (select idPlaylist from (select p_c.idPlaylist, count(p_c.idCanzone)
                                             from playlist_canzone as p_c
                                             join aux on aux.idPlaylist = p_c.idPlaylist
                                             group by (p_c.idPlaylist)
                                             order by count(p_c.idCanzone) desc
                                             limit 5) as tmp);
```

	nomeplaylist character varying (50)
1	Best of hip hop 2019
2	EDM
3	Ricordi di una vita passata
4	sksk
5	bling bling

Identificate e inserite in una view le playlist pubbliche degli abbonati nati prima del 2000, si utilizza questa per identificare il nome di quelle con più canzoni.
Si identificano le playlist interessate tramite il loro ID che deve essere contenuto nel sottoinsieme delle playlist con più canzoni, trovato tramite sotto-query

Query 3)

Durante il periodo di lockdown, gli utenti hanno potuto acquistare i biglietti per dei Concerti Online. Nel caso in cui la data del concerto dovesse cadere in un periodo dove le restrizioni consentano di eseguire le performance con annesso pubblico, Waver ha pensato ad un modo per far sì che metà degli utenti, maggiorenni, possano presentarsi nel luogo dell'evento e vedere l'artista dal vivo.
Si vogliono quindi trovare gli Abbonati, maggiorenni, che hanno acquistato un biglietto per un concerto dell'artista X con numero del posto dispari.

```
drop view if exists concerti_dell_artista_x;
create view concerti_dell_artista_x as
select distinct co.idConcerto, co.titoloConcerto
from artista_concerto as ac
join concertoOnline as co on ac.idConcerto = co.idConcerto
where ac.codiceArtista = 'HJLRTL83S12G430B'
order by co.idConcerto;
select *
from abbonato_concerto as abc
join concerti_dell_artista_x as cdax on abc.idConcerto = cdax.idConcerto
where abc.nPostoOccupato % 2 != 0
and exists (select *
            from abbonato as ab1
            where abc.codiceabbonato = ab1.codicefiscale
            and DATE_PART('year', current_date::date) - DATE_PART('year', ab1.bdate::date) > 18 );
```

Trovata una view per identificare i concerti dell'artista dato (qui per esempio, il creator con codice fiscale 'HJLRTL83S12G430B'), si utilizza questa per ricavare gli abbonati che hanno acquistato un biglietto per i concerti trovati. Questi abbonati inoltre devono avere un numero di posto dispari, ricavato tramite modulo, e una età maggiore di 18 anni dalla data odierna, che devono quindi esistere in una sotto-query che seleziona tutti gli abbonati maggiorenni.

	codiceabbonato character varying (16)	idconcerto integer	npostooccupato integer	idconcerto integer	titoloconcerto character varying (30)	codiceartista character varying (16)
1	PDSMLS33C54G392E	1	23	1	Revival	HJLRTL83S12G430B
2	KSMBVU77A06B812F	5	7045	5	La caccia alle streghe	HJLRTL83S12G430B

Query 4)

Waver permette ai propri iscritti di poter accedere ad una vasta libreria di podcast che coprono una ricca quantità di interessi. Chiaramente è interessante, a fini statistici, capire quali episodi hanno avuto maggiore seguito. Si vuole quindi capire qual è l'episodio maggiormente riprodotto dagli abbonati.

```
drop view if exists episodi_maggiormente_ascoltati;
create view episodi_maggiormente_ascoltati as
select ep.numeroEpisodio, ep.idPodcast, count(*) as quantity
from abbonato_episodio as ep
group by (ep.numeroEpisodio, ep.idPodcast);

select p.titolopodcast, ama.numeroEpisodio, ama.idpodcast
from episodi_maggiormente_ascoltati as ama
join podcast as p on p.idpodcast = ama.idpodcast
where quantity = (select max(quantity)
                  from episodi_maggiormente_ascoltati);
```

	titolopodcast character varying (30)	numeroepisodio integer	idpodcast integer
1	Filosofia di Pavel	1	1
2	Filosofia di Pavel	2	1

Trovata una view per identificare gli episodi più riprodotti dagli abbonati, con relativo conteggio, si usa questa per trovare (tramite sotto-query) l'episodio con conteggio massimo, che corrisponderà al più riprodotto. Si selezionano anche i relativi dati del podcast di provenienza.

Query 5)

Nel mondo musicale alcuni artisti, magari agli albori, sono definiti indipendenti, cioè non sono affiliati ad alcuna Casa Discografica. Questi però possono ugualmente collaborare con altri Creator, al fine di fare esperienza ed emergere. Risulta però più difficile il contrario, cioè che qualche altro artista partecipi ad un featuring di un indipendente.
Si vogliono quindi trovare tutti gli artisti indipendenti che hanno partecipato a dei featuring ma non ne hanno avuti.

```
select art.codiceFiscale, art.nickname
from artista as art
join canzone as c on c.codiceArtista = art.codiceFiscale
where art.idCasaDiscografica is null
      and art.codiceFiscale in (select ac.codiceArtista from artista_canzone as ac)
      and c.idCanzone not in (select ac.idCanzone from artista_canzone as ac);
```

	codicefiscale [PK] character varying (16)	nickname character varying (100)
1	HBPPMR83P08D310X	CXXson
2	HKONSK94C58B914D	Sdrumox
3	VZJSBW42D14G415X	legend

Si trovano gli artisti e relative canzoni. Questi artisti per essere indipendenti devono avere il campo della Casa Discografica nullo. Inoltre, per le specifiche della query, il loro codice fiscale deve apparire nella tabella delle partecipazioni alle canzoni mentre l'identificativo di qualsiasi loro brano no.

Query 6)

In questo genere di piattaforme online, le Case Discografiche hanno fra loro una forte concorrenza e la forza che esse detengono in termini di seguito è fondamentale per l'immagine aziendale.
Si vuole quindi trovare la Casa Discografica con più seguito in termini di Concerti Online e Podcast.

```
drop view if exists ascolti_episodi;
create view ascolti_episodi as
select ae.numeroEpisodio, ae.idPodcast, count(*) as ascolti
from abbonato_episodio as ae
group by (ae.numeroEpisodio, ae.idPodcast);

drop view if exists partecipanti_al_concerto;
create view partecipanti_al_concerto as
select ac.idConcerto, count(*) as partecipanti
from abbonato_concerto as ac
group by (ac.idConcerto);

select tmp.nome, sum(tmp.ascoltiTot) as results
from ( select cd.nome, aep.ascolti as ascoltiTot
```

```
       from ascolti_episodi as aep
       join podcast as p on p.idPodcast = aep.idPodcast
       join casaDiscografica as cd on cd.idCasa = p.idCasaDiscografica
       group by (cd.nome, aep.ascolti)
       union
       select cad.nome, pac.partecipanti as partecipantiTot
       from partecipanti_al_concerto as pac
       join concertoOnline as co on pac.idConcerto = co.idConcerto
       join casaDiscografica as cad on cad.idCasa = co.idCasaDiscografica
       group by (cad.nome, pac.partecipanti) ) as tmp
group by (tmp.nome)
order by (results) desc limit 1;
```

	nome character varying (70)	results numeric
1	Death Row Records	11

Si creano due view: una per raggruppare i Podcast con relativi numeri di ascolto e una per i Concerti Online con il conteggio dei partecipanti. Le due view vengono usate nella query successiva per capire qual è la Casa Discografica maggiormente seguita, tramite l'unione della somma dei loro valori.

6.Indici:

In un servizio di streaming musicale come Waver è fondamentale avere un metodo veloce di indicizzazione dei brani. Difatti, l'azione principale nella piattaforma è la ricerca di nuovi brani, da aggiungere alle proprie playlist e/o per ricondividerli con altre persone.
La ricerca tramite interfaccia avviene ovviamente digitando il nome del brano che si desidera.
Si è deciso perciò di inserire un indice *ricerca_titoli_canzoni* al fine di velocizzare la ricerca tramite nome dei brani.

```
drop index if exists ricerca_titoli_canzoni;
create index ricerca_titoli_canzoni on canzone(titoloCanzone);
```

Nelle query che riguardano questo genere di ricerca si evidenzia un netto miglioramento nel tempo di risposta.
(La differenza nelle tempistiche di risposta è tanto più evidente quanto più sono i dati indicizzati).

7.Note:

Abbiamo scelto di creare un singolo file c++ chiamato queryVisualizer visto le modeste dimensioni del file e la semplicità della sua funzione.
Per la compilazione è necessario avere nella stessa directory del file queryVisualizer.cpp la cartella dependencies con il contenuto mostrato a lezione.
Il comando per compilare è il seguente: g++ queryVisualizer.cpp -L ./dependencies/lib -lpq -o queryVisualizer.o
La compilazione è avvenuta correttamente su un sistema operativo Windows 10 x64 bit con compilatore minGW -> gcc versione: 4.8.3.