Spotty Database - Relazione

1. Progettazione Concettuale

1.1 Entità

Track

- Traccia, Audio, Canzone

Entità base del database. Descrive le canzoni.

Album

Direttamente collegata con l'entità Track. Ogni Track appartiene ad un album (anche i singoli).

Ogni Track fa parte di un Album, proprio come in Spotify se un Artista desidera caricare un singolo viene creato un **Single** (generalizzazione) che contiene una singola Track.

Un Album con più di una traccia sarà definito come **NotSingle** (generalizzazione)

La differenza è utile poichè quando un Album è un singolo verrà mostrato in maniera diversa dall'applicazione. Non vi è differenza in numero di attributi o di associazioni tra le due ed è per questo che l'accorpamento nell'entità genitore è un opzione valida.

Playlist

Più generale possibile. Contiene sia quelle create dagli utenti che quelle generate dal sistema (nella generalizzazione). Le playlist possono essere pubbliche o private.

DailySuggestion (generalizzazione) contiene le playlist generate dal sistema. Si è deciso di creare la generalizzazione per gestire con più facilità il suggerimento della playlist.

Questa entità contiene solo la playlist del giorno dell'utente, non altre playlist consigliate.

Ogni User ha una e una sola DailySuggestion. Ogni DailySuggesion ha uno e un solo User.

La playlist non viene creata nuovamente ogni giorno. Invece viene modificata dal sistema in automatico in base agli ascolti più recenti. La DailySuggestion viene mostrata nella home dell'utente.

Category

- Genere, Categoria

Categorie delle canzoni. Utile per fare una previsione sul gradimento dell'utente o per raccogliere le canzoni per categoria.

Revenue

- Guadagno, Rendita

Contiene un'unica entrata contenente il rapporto fra numero ascolti e revenue generata (in euro). Si è deciso di implementarla nel database invece che nel backend per rendere eventuali modifiche più semplici.

Artist

- Musicista, Cantante, Artista

Contiene gli artisti in generale, è usata per i solisti, ma anche per le band e i loro componenti. La relazione fondamentale è quella con Album (Making).

Band e **SingleArtist** sono le generalizzazioni di **Artist**, sono collegati da un associazione **ArtistBelongsToBand**, una Band potrebbe essere composta da Artisti con una carriera attiva anche da SingleArtist, è utile quindi differenziare gli account della Band e dei singoli componenti.

Spesso però i componenti delle Band non sono in SingleArtist, ed è per questo che di **molte Band non conosciamo la loro composizione**, vengono trattate quindi come un entità unica (unico artista).

User

- Utente

Normale utente, possiede un profilo visitabile, può seguire artisti e altri utenti.

1.2 Associazioni

Making

Making è un'associazione **N-N** tra **Album** e **Artist**, un artista può creare N Album, un album è creato da N artisti.

ArtistBelongsToBand

ArtistBelongsToBand è un'associazione **N-N** tra **SingleArtist** e **Band** (generalizzazioni di Artist), serve per rappresentare i componenti di una Band. Una Band può essere composta da N artisti, la partecipazione è però facoltativa poichè non tutte le Band hanno una composizione nota.

ArtistBelongsToBand ha un attributo **role**, una stringa che descrive il compito dell'artista all'interno della band, esempio: "Cantante", "Batterista/Bassista", "Musicista" etc...

Features

Features è un'associazione **N-N** tra **Artist** e **Track**, un artista può partecipare a N featuring per diverse tracce, una traccia può essere composta da diversi artisti in featuring.

TrackBelongsToAlbum è un'associazione **N-N** tra **Album** e **Track**, una traccia deve essere contenuta in almeno un Album, spesso però una traccia viene prima pubblicata come singolo e poi successivamete aggiunta anche in un Album completo, una traccia può quindi essere contenuta in N Album. Un Album può contenere N tracce, con il vincolo che ce ne sia almeno una.

È utile salvare l'ordine con cui l'artista ha deciso di inserire le tracce, in modo che ogni album mostri le tracce sempre nello stesso ordine, per questo TrackBelongsToAlbum ha un attributo **position**.

Similarity

Similarity è un'associazione **N-N** tra **Track** e **Track**, viene utilizzata per suggerire all'utente delle traccie simili tra quelle più ascoltate e/o piaciute. Viene calcolata per ogni traccia la sua similarità con tutte le altre traccie quindi N-N. Abbiamo inserito la partecipazione come facoltativa perchè per nuove tracce inserite la similarità potrebbe non essere calcolata in modo istantaneo.

L'associazione ha come attributo **amount**, cioè la similarità calcolata usando la distanza euclidea ad N (# di attributi) dimensioni tra le due tracce che partecipano alla relazione.

ListenedTo

ListenedTo è un'assocazione **N-N** tra **Track** e **User**, un utente può ascoltare N tracce, e una traccia può essere ascoltata da N utenti.

NowListeningTo

NowListeningTo è un'associazione **1-N** tra **Track** e **User**, un utente può stare ascoltando 1 traccia e una traccia può essere contemporaneamente ascoltata da N utenti.

LikesTrack

LikesTrack è un'associazione **N-N** tra **Track** e **User**, un utente può mettere like a N tracce e una traccia può ricevere like da N user.

TrackBelongsToCategory

TrackBelongsToCategory è un'associazione **N-N** tra **Track** e **Category**, un traccia può appartenere a N categorie, ad una categoria possono appartenere N tracce.

TrackBelongsToPlaylist

TrackBelongsToPlaylist è un'associazione **N-N** tra **Track** e **Playlist**, una traccia può appartenere a N playlist, una playlist può essere formata da N Track, al contrario degli Album le playlist possono anche essere vuote.

SuggestedPlaylist

SuggestedPlaylist è un'associazione **1-1** tra **User** e **DailySuggestion**, per ogni utente viene creata una specifica DailySuggestion, e una DailySuggestion è stata creata per un particolare utente.

LikesPlaylist

LikesPlaylist è un'associazione **N-N** tra **User** e **Playlist**, un utente può mettere like a N tracce, un traccia può avere like da N utenti

LikesAlbum

LikesAlbum è un'associazione **N-N** tra **User** e **Album**, un utente può mettere like a N album, un album può avere like da N utenti.

FollowUser

FollowUser è un'associazione **N-N** tra **User** e **User**, un utente può seguire N utenti e quindi un utente può essere seguito da N utenti.

FollowArtist

FollowArtist è un'associazione **N-N** tra **User** e **Artist**, un utente può seguire N artisti, un artista può essere seguito da N utenti.

PlaylistBelongsToUser

PlaylistBelongsToUser è un'associazione **1-N** tra **User** e **Playlist**, un utente può creare N playlists una playlist deve essere creata da 1 utente.

Possiede un attributo **date** per memorizzare la data di creazione della playlist.

1.3 Documentazione Associata

Dizionario dei dati

	Entità	Descrizione	Attributi	Identificatore
1	Track	Contiene le canzoni	id title, audio, danceability, energy, speechiness, acousticness, liveness, valence, instrumentalness, loudness, tempo, durationMs, explicit	id
2	Album	Ogni canzone appartiene ad un album (anche i singoli)	id, title, image,durationMs, releaseDate	id
3	Playlist	Collezione di canzoni. Vengono create dagli utenti o generate dal sistema	id, name, description, public	id
4	Category	Categorie delle canzoni	name	name
5	Revenue	Unica entrata contenente il rapporto fra numero ascolti e revenue generata (in euro)	reproductions, paid	
6	Artist	Compositori di canzoni	id, name, isVerified, nFollowers	id
7	User	Normale utente con profilo visitabile	id, image, email, password, dob	id

	Relazioni	Descrizione	Componenti	Attributi
1	Making	Creazione Album	Album, Artist	
2	ArtistBelongsToBand	Componenti Band	Band, SingleArtist	role
3	Features	Canzoni con featuring	Track, Artist	
4	TrackBelongsToAlbum	Canzone all'interno di Album	Track, Album	position
5	Similarity	Canzoni con caratteristiche simili	Track	amount
6	ListenedTo	Singola riproduzione di una canzone	Track, User	date
7	NowListeningTo	Canzone che si sta ascoltando adesso	Track, User	
8	LikesTrack	Mi piace alle canzoni	Track, User	date
9	TrackBelongsToCategory	Canzone appartiene ad una Categoria	Track, Category	
10	TrackBelongsToPlaylist	Canzone all'interno di Playlist	Track, Playlist	addedDate
11	SuggestedPlaylist	Playlist suggerita dal sistema	DailySuggestion, User	

	Relazioni	Descrizione	Componenti	Attributi
12	LikesPlaylist	Mi piace alle Playlist	Playlist, User	date
13	LikesAlbum	Mi piace agli Album	Album, User	date
14	FollowUser	Follow tra utenti	User	
15	FollowArtist	Follow Artista	User, Artist	
16	PlaylistBelongsToUser	Playlist dell'utente	User, Playlist	date

Vincoli di integrità e regole di derivazione

	Vincoli di Integrità
1	Un utente non può seguire se stesso
2	Un utente può ascoltare una canzone per volta
3	Il valore di Image e Audio deve essere un URL valido
4	Solamente un SingleArtist può essere componente di una Band

	Regole di Derivazione
1	Il numero degli ascolti per ogni canzone si ottiene contando le tuple di ListenedTo
2	La quantità di denaro prodotta da una traccia si ottiene moltiplicando il numero di ascolti diviso Reproduction moltiplicato per Paid

2. Progettazione Logica

2.1 Tavola dei Volumi

	Concetto	Tipo	Volume
1	Track	E	50M
2	Album	E	2M
3	SingleArtist	E	6M
4	Band	E	2M
5	User	E	200M
6	Playlist	E	4.500M
7	DailySuggestion	E	200M
8	Category	E	5000
9	TrackBelongsToAlbum	R	65M
10	TrackBelongsToPlaylist	R	90.000M
11	TrackBelongsToCategory	R	150M
12	Similarity	R	5.000.000M
13	ListenedTo	R	2.000.000M
14	NowListeningTo	R	200M
15	LikesTrack	R	100.000M
16	FollowsUser	R	2.000M
17	FollowsArtist	R	4.000M
18	Features	R	10M
19	ArtistBelongsToBand	R	6M
20	LikesAlbum	R	6.000M
21	LikesPlaylist	R	8.000M
22	SuggestedPlaylist	R	200M

	Concetto	Tipo	Volume
23	Making	R	6M
24	PlaylistBelongsToUser	R	2.000M

Di seguito alcuni dati statistici delle principali piattaforme di music streaming che sono stati utilizzati per la costruzione della tavola dei volumi:

- Ogni Artist fa in media 3 album.
- Ogni user segue in media 20 artist.
- Ogni user segue in media altri 10 user.
- Ogni user mette like a 30 album in media.
- Ogni Band ha in media 3 componenti aventi il collegamento.
- Ogni canzone contiene in media 0.2 features (non si include l'artista proprietario).
- Ipotizziamo ogni utente ascolti nella sua vita circa 10.000 canzoni (distinte) quindi la cardinalità di ListenedTo è (numero di user) * 10.000 = 2.000.000M.
- Ogni user mette like a circa 500 canzoni, ovviamente questo numero deve essere minore delle canzoni che ascolta, e supponiamo non gli piacciano tutte quelle che ascolta.

2.2 Tavola delle Operazioni

	Operazione	Esecuzioni	Unità di tempo
1	Ricerca	150M	Giorno
2	Aggiunta brano in una playlist	400M	Giorno
3	Calcolo Similarity tra canzoni	200M	Giorno
4	Creazione DailySuggestion	200M	Giorno
5	Inserimento di un nuovo Album	40.000	Giorno
6	Visualizzazione numero ascolti dell'ultimo mese per artisti	200M	Settimana
7	Visualizzazione numero di like dell'ultimo mese per utenti	150M	Settimana
8	Visualizzazione nomi dei followers	50M	Settimana
9	Richiesta URL tracks	2.000M	Giorno

Informazioni aggiuntive:

- La creazione della classifica degli artisti ed il calcolo della retribuzione degli artisti, nonostante siano operazioni possibili e fatte periodicamente dal DB, non sono incluse nella tavola delle operazioni. La loro frequenza è di una volta a settimana.
- Visto che ogni brano deve per forza essere contenuto in un album consideriamo nelle operazioni l'inserimento dell'album invece che delle singole track.
- La ricerca viene fatta contemporaneamente nelle tabelle: Tracks, Album, Artist, User, e Playlist.

2.3 Tavole degli accessi

1	Concetto	Costrutto	Accessi	Tipo
1	Artist	Entità	3	R

1	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	2	R

1	Concetto	Costrutto	Accessi	Tipo
1	Playlist	Entità	3	R

1	Concetto	Costrutto	Accessi	Tipo
2	User	Entità	3	R
3	PlaylistBelongsToUser	Relazione	3	R

1	Concetto	Costrutto	Accessi	Tipo
1	Album	Entità	3	R
2	Artist	Entità	3	R
3	Making	Relazione	3	R

1	Concetto	Costrutto	Accessi	Tipo
1	Track	Entità	6	R
2	Artist	Entità	10	R
3	Album	Entità	10	R
4	TrackBelongsToAlbum	Relazione	6	R
5	Making	Relazione	10	R
6	Features	Relazione	4	R

2	Concetto	Costrutto	Accessi	Tipo
1	Track	Entità	1	R
2	Playlist	Entità	1	R
3	TrackBelongsToPlaylist	Relazione	1	W

3	Concetto	Costrutto	Accessi	Tipo
1	Track	Entità	100.000	R
2	Similarity	Relazione	100.000	W

4	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	200M	R
2	TrackBelongsToPlaylist	Relazione	4.000M	W
3	ListenedTo	Relazione	200M	R
4	LikesTrack	Relazione	200M	R
5	TrackBelongsToCategory	Relazione	200M	R

5	Concetto	Costrutto	Accessi	Tipo
1	Album	Entità	1	W
2	Track	Entità	10	W
3	TrackBelongsToAlbum	Relazione	10	W
4	Artist	Entità	6	R
5	Making	Relazione	1	W
6	Features	Relazione	7	W

6	Concetto	Costrutto	Accessi	Tipo
1	Artist	Entità	1	R
2	Making	Relazione	3	R
3	Album	Entità	3	R
4	TrackBelongsToAlbum	Relazione	30	R
5	Features	Relazione	10	R
6	Track	Entità	40	R
7	ListenedTo	Relazione	16M	R

7	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	1	R
2	LikesTrack	Relazione	500	R

7	Concetto	Costrutto	Accessi	Tipo
3	LikesPlaylist	Relazione	40	R
4	LikesAlbum	Relazione	30	R

8	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	11	R
2	FollowUser	Relazione	10	R

9	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	1	R
2	Track	Entità	1	R
3	NowListening	Relazione	1	W
4	ListenedTo	Relazione	1	W

Informazioni aggiuntive:

- **Operazione 1:** La prima operazione (ricerca complessiva) è una operazione complessa e scomponibile, considerato che sono delle ricerche su entità separate. Quindi per chiarezza si è deciso di scomporre nelle singole operazioni atomiche.
    1. In media 3 artisti con lo stesso nome.
    2. In media 2 utenti con lo stesso nome.
    3. In media 3 playlist con lo stesso nome create da 3 diversi utenti.
    4. In media 3 Album con lo stesso nome creati da 3 diversi artisti.
    5. In media 6 tracks con lo stesso nome, delle quali 4 sono dei featuring, contenute in 10 diversi album di 10 diversi artisti.
- **Operazione 2**: Possiamo aggiungere un solo brano per volta ad un playlist.
- **Operazione 3**: Per ogni canzone aggiunta si prendono 100.000 tracks e si calcola la similarity tra la canzone aggiunta e queste 100.000 tracks.
- **Operazione 4**: Per ogni utente si controlla la canzone più ascoltata, l'ultima canzone a cui hanno messo like, le categorie preferite e sulla base di queste informazioni si aggiornano le 20 tracks all'interno delle playlist suggerite.
- **Operazione 5**: Un album mediamente contiene 10 track ed è creato da un solo artista. Delle 10 Track, 7 hanno featuring, ipotiziamo con 5 artisti
- **Operazione 6:** Ogni artist ha in media 3 album, aventi circa 10 track con un featuring. Ogni track viene ascoltata circa 400.000 volte, quindi 40*400.000 è il numero di accessi a ListenedTo.
- **Operazione 7**: Un utente mediamente, mette like a 500 track, 40 playlist e 30 Album.
- **Operazione 8**: Ogni utente segue in media 10 Utenti (vogliamo i nomi)
- **Operazione 9:** Un utente non può ascoltare più tracce contemporaneamente. Un utente ascolta in media 10 Track al giorno.

2.4 Analisi delle ridondanze

Operazione 2 analisi accessi

Si è valutato di salvare il numero di canzoni presenti in una playlist, aggiungendo un dato ridondante.

L'operazione più importante che coinvolge le playlist è la 2. Le altre sono considerate poco costose perché poco frequenti o particolarmente semplici.

Allo stato attuale del DB (senza ridondanza):

2	Concetto	Costrutto	Accessi	Tipo
1	Track	Entità	1	R
2	Playlist	Entità	1	R

2	Concetto	Costrutto	Accessi	Tipo
3	TrackBelongsToPlaylist	Relazione	1	W

Se invece lo memorizziamo la tabella diventa (con ridondanza):

2	Concetto	Costrutto	Accessi	Tipo
1	Track	Entità	1	R
2	Playlist	Entità	1	R
3	TrackBelongsToPlaylist	Relazione	1	W
4	Playlist	Entità	1	W

Si è quindi deciso di non aggiungere la ridondanza.

Operazione 6,9 analisi accessi

Si è valutato di aggiungere un attributo a track memorizzando le volte in cui è stata riprodotta da un utente. È una ridondanza perché tale dato si può calcolare contando le tuple in ListenedTo collegate alla particolare Track. Tale aggiunta velocizzerebbe la operazione 6 perché si eviterebbe di fare un join con ListenedTo che è molto costoso. Tuttavia rallenterebbe di molto la operazione 9. L'operazione 9 viene fatta ogni volta che un utente ascolta una canzone.

Analizziamo il DB senza ridondanza:

	Operazione	Esecuzioni	Unità di tempo
6	Visualizzazione numero ascolti dell'ultimo mese per artisti	200M	Settimana
9	Richiesta URL tracks	2.000M	Giorno

6	Concetto	Costrutto	Accessi	Tipo
1	Artist	Entità	1	R
2	Making	Relazione	3	R
3	Album	Entità	3	R
4	TrackBelongsToAlbum	Relazione	30	R
5	Features	Relazione	10	R
6	Track	Entità	40	R
7	ListenedTo	Relazione	16M	R

9	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	1	R
2	Track	Entità	1	R
3	NowListening	Relazione	1	W
4	ListenedTo	Relazione	1	W

In questo modo abbiamo settimanalmente:

$6) \quad 200M \cdot 16M$

$9) \quad 2000M \cdot 7 \cdot 2(R) + 2000M \cdot 7 \cdot 2 \cdot 2(W)$

$Total = 3.2 \cdot 10^{15} + 8.4 \cdot 10^{10} \simeq 3.200084 \cdot 10^{15}$

Analizziamo ora il DB con la ridondanza:

Track(ID, Title, Audio, Danceability, Energy, Loudness, Speechiness, Acousticness, Liveness, Valence, Instrumentalness, Tempo, DurationMS, Explicit, **plays**)

	Operazione	Esecuzioni	Unità di tempo
6	Visualizzazione numero ascolti dell'ultimo mese per artisti	200M	Settimana
9	Richiesta URL tracks	2.000M	Giorno

6	Concetto	Costrutto	Accessi	Tipo
1	Artist	Entità	1	R
2	Making	Relazione	3	R
3	Album	Entità	3	R

6	Concetto	Costrutto	Accessi	Tipo
4	TrackBelongsToAlbum	Relazione	30	R
5	Features	Relazione	10	R
6	Track	Entità	40	R

9	Concetto	Costrutto	Accessi	Tipo
1	User	Entità	1	R
2	Track	Entità	1	R
3	NowListening	Relazione	1	W
4	ListenedTo	Relazione	1	W
5	Track	Entità	1	W

$6) \quad 200M \cdot 100$

$9) \quad 2000M \cdot 7 \cdot 2(R) + 2000M \cdot 7 \cdot 3 \cdot 2(W)$

$Total = 20000M + 1.12 \cdot 10^{11} \simeq 1.32 \cdot 10^{11}$

Arriviamo alla conclusione che **conviene** aggiungere il numero di plays in Track

2.5 Eliminazione delle Generalizzazioni

Generalizzazione Album

Dal momento che Single e Not-Single differiscono solo per numero di track, abbiamo deciso di accorparle sull'entità genitore, aggiungendo l'attributo **numberOfTracks**.

Generalizzazione Artisti

Dal momento che la maggior parte delle operazioni coinvolgono solo l'entità Artist (genitore), si è deciso di accorparle in un'unica entità, distinguendo Band e SingleArtist con l' attributo **isBand**. L'associazione fra Band e SingleArtist diventa quindi un'associazione ricorsiva su Artist. Ovviamente si impongono i vincoli di integrità necessari (vedi tabella).

Generalizzazione Playlist

Dal momento che le DailySuggestion sono Playlist molto particolari e le normali operazioni su playlist non si applicano alle DailySuggetion, si è deciso di sostituire la generalizzazione con un' associazione 1 a 1, aggiungendo l'attributo **isDailySuggestion** a Playlist. In questo modo la ricerca delle playlist avverrà solo tra quelle non suggerite (isDailySuggestion=0) e l'accesso giornaliero effettuato per modificare le playlist suggerite avverrà su una tabella più piccola, diminuendo gli accessi.

2.6 Partizionamento/accorpamento di entità e associazioni

Dopo una approfondita analisi non abbiamo trovato occasione di utilizzare il partizionamento e/o l'accorpamento. Infatti ispezionando le operazioni più frequenti del DB (seguendo la regola 80-20) non abbiamo trovato modi alternativi di strutturare il DB per ridurre gli accessi.

2.7 Studio sulle Chiavi Primarie

Esistono due tipi di identificatori:

- Identificatore Interni
- Identificatori Esterni

Gli identificatori interni utilizzano un sottoinsieme degli attributi dell'entità per identificarla univocamente, mentre gli identificatori esterni utilizzano anche le relazioni a cui un entità partecipa per poterla identificare in modo univoco.

Nel nostro progetto abbiamo utilizzato **Spotify Dev API**, che ci ha permesso di caricare Album, Artist e Track direttamente dal database orginale di Spotify.

Spotify utilizza un identificatore basato su un codice alfanumerico di 22 caratteri, questo facilita la costruzione degli url, esempio `spotty.app\album\{id}` , `spotty.app\artist\{id}` .

Abbiamo quindi deciso per una migliore gestione della app di utilizzare un identicatore basato su codice per ogni entità.

Le uniche entità in cui non abbiamo utilizzato un ID sono Category e Revenue. In category la chiave primaria è il suo unico attributo (name). Essendo un solo attributo non avrebbe avuto senso fare diversamente.

2.8 Schema delle Tabelle

- **Track**(id, title, audio, danceability, energy, loudness, speechiness, acousticness, liveness, valence, instrumentalness, tempo, durationMs, isExplicit, plays)
- **User**(id, image, username, email, password, dob, *track)
- **Artist**(id, name, isVerified, nFollowers, isBand)
- **Album**(id, title, numberOfTracks, image, durationMs, releaseDate)
- **Playlist**(id, name, description, *creator, isDailySuggestion)
- **DailySuggestion**(id, *suggestedFor)
- **ArtistBelongsToBand**(*singleArtist, *band, role)
- **TrackBelongsToPlaylist**(*track, *playlist, addedDate)
- **TrackBelongsToAlbum**(*album, *track, position)
- **TrackBelongsToCategory**(*category, *track)
- **Similarity**(*track1, *track2, amount)
- **Features**(*artist, *track)
- **ListenedTo**(*user, *track, date)
- **LikesPlaylist**(*playlist, *user, date)
- **LikesAlbum**(*album, *user, date)
- **LikesTrack**(*user, *track, date)
- **FollowUser**(*follower, *followed)
- **FollowArtist**(*user, *artist)
- **Making**(*artist, *album)
- **Revenue**(reproduction, paid)
- **Category**(name)

2.9 Specifiche sul Funzionamento

Riproduzione delle track

Per facilitare l’operazione numero 6, diversamente molto complessa, si usa plays, che contiene il numero di riproduzioni dell’ultimo mese. Viene aggiornato con un trigger.

Like alle Playlist

Il committente richiedeva la possibilità di, mettendo like ad una playlist/album, estendere il like a tutte le canzoni al suo interno. Questa funzionalità non è implementabile in maniera coerente con la definizione dei like alle singole tracce, quindi si è deciso di darla come opzione. Si può mettere il like all’album o alla playlist, e per entrambi si può inoltre usare il bottone “Like Bomb” che estende il like a tutte le canzoni al suo interno.

Numero canzoni in Playlist

Nelle playlist il numero di canzoni viene calcolato in quanto le canzoni sono aggiunte dinamicamente.

Likes ad Album e Playlist

Il numero dei like degli utenti viene calcolato ogni volta perché cambia in maniera dinamica, e viene richiesto raramente, quindi conviene calcolarlo le poche volte che viene richiesto.

Follows User e Artist

Il numero di followers degli utenti viene calcolato ogni volta per via della sua scarsa richiesta e della quantità ridotta. Per l'artista è previsto l'attributo nFollowers, che viene aggiornato con i trigger, per i motivi opposti.

3 Progettazione Fisica

3.1 Popolazione del Database

Il database è stato popolato utilizzando le API di spotify per le entità in cui questo fosse possibile.

Le restanti entità e le associazioni fra loro sono state create con degli script in python o a mano.

La popolazione effettiva del database non è una rappresentazione fedele a quella prevista per l'applicazione funzionante.

Le query usate per la creazione, ed il popolamento del DB sono disponibili come file .sql, così da poterlo ricreare in qualunque momento.

3.2 Trigger

Update nFollower in Artist

```
CREATE TRIGGER nFollowerArtist
AFTER INSERT ON FollowArtist
FOR EACH ROW
UPDATE Artist SET Artist.nFollowers = 1 + Artist.nFollowers WHERE Artist.id = new.artist;

INSERT INTO FollowArtist(user,artist) VALUES('QKedDkxLZFOPQS8pAhkj01','0bz9hDpUbAw5JElgEiuIYZ');
```

Update Plays in Track

```
CREATE TRIGGER trackPlays
AFTER INSERT ON ListenedTo
FOR EACH ROW
UPDATE Track SET Track.plays = 1 + Track.plays WHERE Track.id = new.track;

INSERT INTO ListenedTo(track,user,date) VALUES('4VwvwlbNZavsDkRdIKFGBi','QKedDkxLZFOPQS8pAhkj01','2021-01-05');
```

Query

1 Ricerca

1.1

```
SELECT *
FROM Artist
WHERE Artist.name LIKE '%Imagine Dragons%';
```

1.2

```
SELECT *
FROM User
WHERE User.username LIKE '%Emma rock%';
```

1.3

```
SELECT *
FROM Playlist,User
WHERE User.id=Playlist.creator AND Playlist.name LIKE '%This is Gospel%';
```

1.4

```
SELECT *
FROM Album, Artist, Making
WHERE Album.id=Making.album AND Artist.id=Making.artist AND Album.title LIKE '%If I Know Me%';
```

1.5

```
SELECT *
FROM Track,Artist,Album,TrackBelongsToAlbum,Making
WHERE Track.id=TrackBelongsToAlbum.track AND Album.id=TrackBelongsToAlbum.album AND Making.album=Album.id AND Making.artist=Artist.id

SELECT *
FROM Track,Features,Artist
WHERE Track.id=Features.track AND Artist.id=Features.artist AND Track.title LIKE '%What do you mean?%';
```

2 Aggiunta brano in una playlist

```
INSERT INTO TrackBelongsToPlaylist(track, playlist, addedDate)
VALUES('22zRzwWJp3gfM1O00CsRGm', 'PrsS4E1r054PsZUyl4MTix', NOW());
```

3 Similarity tra canzoni (similarity.csv, utilizzato da similarity.py)

```
SELECT * FROM Track track1 INNER JOIN Track track2 ON track1.id < track2.id;
```

4 Creazione daily suggestion (usato da suggester.py)

```


GET_USERS_QUERY = (
    "select U.id, D.id "
    "from User U inner join DailySuggestion D on U.id=D.suggestedFor;"
)

REMOVE_TRACK = (
    "delete from TrackBelongsToPlaylist "
    "where playlist='{}'"
)

GET_LIKED_TRACK = (
    "select T.id "
    "from Track T inner join LikesTrack L on T.id=L.track "
    "where L.user='{}' "
    "order by date desc "
    "limit 5"
)

GET_SIMILAR_TRACK = (
    "select track1, track2 "
    "from Similarity "
    "where track1='{}' or track2='{}' "
    "order by amount "
    "limit 3"
)
```

```
INSERT_TRACK = (
    "insert into TrackBelongsToPlaylist (track,playlist,addedDate) "
    "values ('{}','{}',now())"
)

GET_MOST_LISTENED_GENRES = (
    "select C.category, count(*) as n "
    "from ListenedTo L inner join Track T on L.track=T.id inner join TrackBelongsToCategory C on T.id=C.track "
    "where user='{}' "
    "group by C.category "
    "order by n desc "
    "limit 2"
)

GET_TRACKS = (
    "select C.track "
    "from TrackBelongsToCategory C "
    "where C.category='{}' "
)
```

5 Inserimento nuovo album

```
INSERT INTO Album(id, title, numberOfTracks, image, durationMs, releaseDate)
VALUES ( );

INSERT INTO Track(id, title, audio, danceability, energy, loudness, speechiness, acousticness, liveness, valence, instrumentalness, te
VALUES ( );

INSERT INTO TrackBelongsToAlbum(album, track, position)
VALUES ( );

INSERT INTO Making(artist,album)
VALUES ( );
```

6 Visualizzazione numero ascolti dell’ultimo mese per artisti

```
SELECT  SUM(plays) plays
FROM(
            SELECT SUM(Track.plays) plays
            FROM ((((Artist JOIN Making ON Making.artist=Artist.id) JOIN Album on Making.Album=Album.id) JOIN TrackBelongsToAlbum on T
            WHERE Artist.id='04gDigrS5kc9YWfZHwBETP'
            UNION ALL
            SELECT SUM(Track.plays) plays
            FROM ((Artist JOIN Features ON Artist.id=Features.artist) JOIN Track ON Track.id=Features.track)
            WHERE Artist.id='04gDigrS5kc9YWfZHwBETP'
) LastMonthPlays;
```

7 Visualizzazione numero di like ultimo mese per utenti

```
SELECT COUNT(LikesTrack.track)
FROM (User JOIN LikesTrack on User.id=LikesTrack.user)
WHERE User.id='QKedDkxLZFOPQS8pAhkj01' AND LikesTrack.date > DATE_ADD(NOW(),INTERVAL -30 DAY);

SELECT COUNT(LikesPlaylist.playlist)
FROM (User JOIN LikesPlaylist on User.id=LikesPlaylist.user)
WHERE User.id='QKedDkxLZFOPQS8pAhkj01' AND LikesPlaylist.date > DATE_ADD(NOW(),INTERVAL -30 DAY);

SELECT COUNT(LikesAlbum.album)
FROM (User JOIN LikesAlbum on User.id=LikesAlbum.album)
WHERE User.id='QKedDkxLZFOPQS8pAhkj01' AND LikesAlbum.date > DATE_ADD(NOW(),INTERVAL -30 DAY);
```

8 Visualizzazione nomi dei followers

```
SELECT Follower.id,Follower.username
FROM ((User AS MainUser JOIN FollowUser ON MainUser.id=FollowUser.followed) JOIN User AS Follower ON Follower.id=FollowUser.follower)
```

```
WHERE MainUser.id='QKedDkxLZFOPQS8pAhkj01';
```

9 Richiesta URL tracks

```
SELECT Track.audio
FROM Track
WHERE Track.id='1YEUxb72rwYLEUv48AC3eE';
```

10.1 Like Bomb playlist

```
SELECT track
FROM TrackBelongsToPlaylist
WHERE playlist="id_playlist";

#then, inside a loop for each track id
INSERT INTO LikesTrack(user,track,date)
VALUES ('userid','trackid', NOW());
```

10.2 Like Bomb album

```
SELECT Track.id
FROM TrackBelongsToAlbum
WHERE album="id_album";

#then, inside a loop for each track id
INSERT INTO LikesTrack(user,track,date)
VALUES ('userid','trackid',NOW()) ;
```

11 Classifica autori più ascoltati della settimana

```
SELECT  ID, NAME, SUM(NListen) TotalListen
FROM (
            SELECT Artist.id ID,Artist.name NAME,COUNT(ListenedTo.track) NListen
            FROM (((((Artist JOIN Making ON Making.artist=Artist.id) JOIN Album on Making.Album=Album.id) JOIN TrackBelongsToAlbum on
            WHERE ListenedTo.date > DATE_ADD(NOW(),INTERVAL -7 DAY)
            GROUP BY Artist.id,Artist.name
            UNION ALL
            SELECT Artist.id ID,Artist.name NAME,COUNT(ListenedTo.track) NListen
            FROM (((Artist JOIN Features ON Artist.id=Features.artist) JOIN Track ON Track.id=Features.track) JOIN ListenedTo ON Track
            WHERE ListenedTo.date > DATE_ADD(NOW(),INTERVAL -7 DAY)
            GROUP BY Artist.id,Artist.name
) WeeklyLeaderboard
GROUP BY ID, NAME
ORDER BY TotalListen DESC;
```

12 Calcolo revenue ((0.30/100plays+0.8/100like)/settimana)

```
DROP VIEW WeeklyLeaderboard;
DROP VIEW WeeklyLikes;

CREATE VIEW WeeklyLeaderboard AS
SELECT  ID, NAME, SUM(NListen) TotalListen
FROM (
            SELECT Artist.id ID,Artist.name NAME,COUNT(ListenedTo.track) NListen
            FROM (((((Artist JOIN Making ON Making.artist=Artist.id) JOIN Album on Making.Album=Album.id) JOIN TrackBelongsToAlbum on
            WHERE DATE_ADD(NOW(),INTERVAL -7 DAY) < ListenedTo.date
            GROUP BY Artist.id
            UNION ALL
            SELECT Artist.id ID,Artist.name NAME,COUNT(ListenedTo.track) NListen
            FROM (((Artist JOIN Features ON Artist.id=Features.artist) JOIN Track ON Track.id=Features.track) JOIN ListenedTo ON Track
            WHERE DATE_ADD(NOW(),INTERVAL -7 DAY) < ListenedTo.date
            GROUP BY Artist.id
) WeeklyLeaderboard
```

```
GROUP BY ID, NAME
ORDER BY TotalListen DESC;

CREATE VIEW WeeklyLikes AS
SELECT ID, NAME, SUM(NLikes) TotalLikes
FROM(
            SELECT Artist.id ID,Artist.name NAME,COUNT(LikesTrack.track) NLikes
            FROM (((((Artist JOIN Making ON Making.artist=Artist.id) JOIN Album on Making.Album=Album.id) JOIN TrackBelongsToAlbum on
            WHERE DATE_ADD(NOW(),INTERVAL -7 DAY) < LikesTrack.date
            GROUP BY Artist.id
            UNION ALL
            SELECT Artist.id ID,Artist.name NAME,COUNT(LikesTrack.track) NLikes
            FROM (((Artist JOIN Features ON Artist.id=Features.artist) JOIN Track ON Track.id=Features.track) JOIN LikesTrack ON Track
            WHERE DATE_ADD(NOW(),INTERVAL -7 DAY) < LikesTrack.date
            GROUP BY Artist.id
) WeeklyLikes
GROUP BY ID,NAME
ORDER BY TotalLikes DESC;

#VERSIONE CHE CONSIDERA I LIKE
SELECT WeeklyLikes.ID,WeeklyLikes.NAME,
ROUND((
  ((WeeklyLikes.TotalLikes/100)*0.8)
  +
  ((WeeklyLeaderboard.TotalListen/(SELECT reproduction FROM Revenue))*(SELECT paid FROM Revenue))
), 2) Revenue
FROM WeeklyLeaderboard JOIN WeeklyLikes ON WeeklyLikes.ID=WeeklyLeaderboard.ID;

#VERSIONE CHE CONSIDERA SOLO LE RIPRODUZIONI
SELECT WeeklyLikes.ID,WeeklyLikes.NAME,
ROUND (((WeeklyLeaderboard.TotalListen/(SELECT reproduction FROM Revenue))*(SELECT paid FROM Revenue)), 2) Revenue
FROM WeeklyLeaderboard JOIN WeeklyLikes ON WeeklyLikes.ID=WeeklyLeaderboard.ID;
```