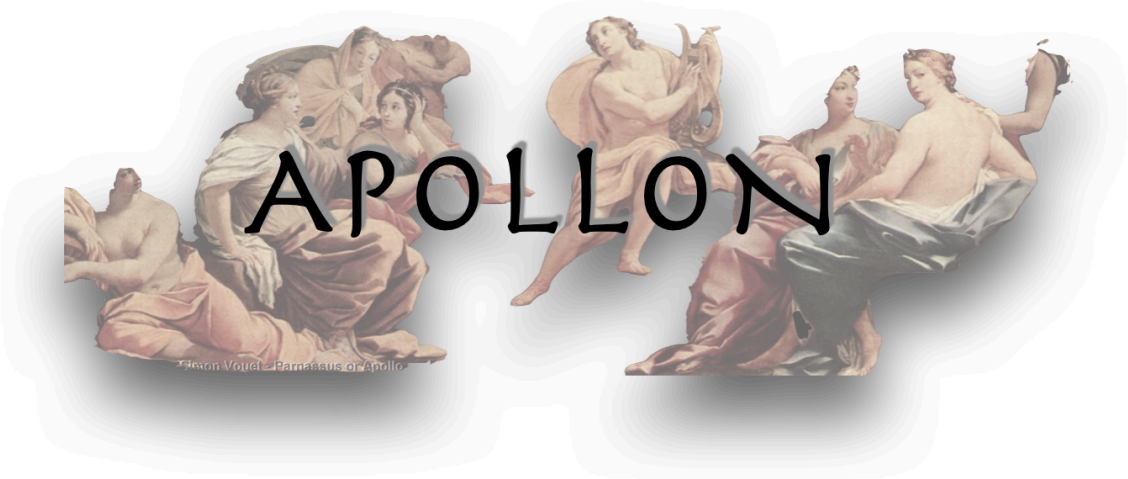


GUSATTO DEREK - 2042330
MARGARIT ANE-MARIE - 1217649

PROGETTO per il corso BASI di DATI
2022-2023

1. Abstract

Apollon è una piattaforma di streaming online di musica che offre agli utenti un vasto catalogo di brani musicali da ascoltare in streaming.
Fondata nel 2021, Apollon si è rapidamente affermata come una delle principali piattaforme di streaming musicale grazie alla sua vasta libreria musicale e alle sue caratteristiche innovative.
Apollon offre agli utenti la possibilità di accedere a milioni di brani provenienti da vari generi musicali, inclusi pop, rock, rap, blues, jazz e molti altri.
Gli utenti possono cercare e riprodurre brani specifici, creare playlist personalizzate (pubbliche e/o private) e seguire gli artisti preferiti.
La piattaforma è disponibile su diversi dispositivi, inclusi dispositivi mobili (iOS e Android) e desktop, consentendo agli utenti di accedere alla loro musica preferita ovunque si trovino.
È diventata una scelta popolare per gli appassionati di musica che cercano un'esperienza di ascolto completa e coinvolgente.

2. Analisi Dei Requisiti

2.1 Descrizione testuale

Il progetto mira a rappresentare una base di dati che includa album e brani nazionali e internazionali pubblicati dal 1951 in poi, tenendo a mente il fatto che la prima produzione di musica digitale nota fu realizzata nel 1951, per questo la scelta di includere nella base di dati soltanto i brani digitali.
In particolare si vuole sapere quali artisti sono memorizzati all'interno della piattaforma, quali brani e album hanno prodotto e l'etichetta discografica a cui appartengono.
Gli utenti che accedono alla piattaforma possono creare delle playlist personalizzate contenenti i brani sopra citati, inoltre la base di dati memorizzerà ogni ascolto effettuato dall'utente.

Dei **brani** si hanno informazioni quali un titolo, la data del rilascio, il guadagno al secondo per l'artista e la durata del brano stesso.

Ai brani possono essere assegnati dei **premi**, identificati da un codice e dei quali interessa il titolo e la società che li assegna. I premi si suddividono in certificazioni discografiche, assegnate per gli ascolti in streaming, vendite e passaggi in radio e premi assegnati tramite concorsi, per i quali vanno memorizzati il concorso e la data.

I brani possono essere raggruppati in **album** dei quali interessa memorizzare un titolo, un prezzo unitario e l'anno di pubblicazione.

Dell'**artista**, identificato dal proprio codice fiscale, devono essere salvati il nome e il cognome, la nazionalità, la data di nascita, i generi musicali ed un eventuale nome d'arte.

Si devono memorizzare anche le **etichette discografiche** con le quali gli artisti firmano i contratti. Esse sono identificate dalla propria partita IVA e memorizzate con nome e città, e per ognuna di esse è necessario sapere quali artisti produce.

Gli **utenti** che accedono alla piattaforma sono identificati da un codice che li distingue univocamente e vengono salvate informazioni anagrafiche come nome, cognome, data di nascita ed email.

L'utente ha a propria disposizione un **abbonamento**, identificato da un codice e dal proprietario, di cui interessa la data di sottoscrizione. L'abbonamento può essere base oppure premium, nel secondo caso si memorizza il metodo di pagamento e se esso abbiente su base mensile o annuale.

Ogni utente può ascoltare qualsiasi brano desideri e ogni **ascolto** deve essere salvato insieme all'utente che lo ha effettuato, la data in cui è stato effettuato l'ascolto e l'ora di inizio e di fine.

Ogni utente ha la possibilità di personalizzare la propria piattaforma creando playlist che contengono i brani memorizzati nella base di dati. Le **playlist**, identificate da un codice e dal proprietario, hanno un nome e una visibilità (la playlist può essere pubblica o privata).

2.2 Glossario dei termini

Termine	**Descrizione**	**Collegamenti**
Nome d'arte	Appellativo che va a sostituire o semplificare nome e cognome anagrafici dell'artista.	Artista
Genere	Categoria che identifica i brani e gli artisti in base a dei criteri di affinità	Attributo di Artista
Rilascio	Pubblicazione di una collezione di brani	Album
Etichetta	Marchio commerciale con il quale l'artista firma un contratto per stampare e vendere dischi	Artista
Contratto	Accordo di vendita o licenza che viene concluso tra artisti ed etichette discografiche	Etichetta
pIVA	Serie di 11 numeri che identifica univocamente il titolare e serve a contribuire l'IVA all'Agenzia delle entrate	Attributo di Etichetta, Artista
Guadagno secondo	Profitto economico ottenuto ogni secondo	Attributo di Brano, Artista
Data rilascio	Tempo in cui avviene la pubblicazione di un album	Attributo di Brano, Album
Premio	Ricompensa o riconoscimento assegnati in corrispondenza a determinati meriti legati al brano.	Brano
Discografico	Che riguarda la produzione di dischi e delle registrazioni digitali	Entità figlia di Premio
Società	Insieme di persone che si riuniscono per l'organizzazione della devoluzione del premio	Attributo di Premio
Passaggi Radio	Numero totale di volte in cui un brano è stato riprodotto alla radio	Attributo di Premio
Numero stream	Numero totale di volte in cui un brano è stato riprodotto sulla piattaforma musicale	Attributo di Premio
Ascolto	Esecuzione di un brano da parte di un utente	Brano, Utente
Playlist	Collezione di brani musicali presenti sulla piattaforma	Brano

Privata	Visibilità di una playlist che può essere visualizzata da un solo utente, o da più utenti	Attributo di Playlist
Utente	Chi usufruisce dei servizi che offre la piattaforma	Brano, Playlist, Ascolto, Abbonamento
Premium	Abbonamento a pagamento che offre servizi e vantaggi aggiuntivi sulla piattaforma	Attributo di Abbonamento
Annuale	Tipo di abbonamento che dura un anno	Attributo di Abbonamento
Metodo di pagamento	Modalità per pagare i servizi che offre la piattaforma (VISA , Mastercard, Maestro, PayPal, etc…)	Attributo di Abbonamento

2.3 Operazioni

OPERAZIONE	**TIPO**	**FREQUENZA**
Calcolare per quanto tempo è stato ascoltato un determinato brano	L	700 al giorno
Stampare l’elenco di tutti gli utenti che hanno sottoscritto un abbonamento Base e/o Premium	L	7000 al giorno
Registrazione di un nuovo utente sulla piattaforma	S	500 al giorno
Inserimento di un nuovo album sulla piattaforma	S	3000 al mese
Ricerca di tutti gli artisti appartenenti al genere “Blues” presenti sulla piattaforma	L	250 al giorno
Visualizzare i premi vinti da un determinato brano	L	100 al giorno
Aggiornamento di una playlist sulla piattaforma	S	50000 al giorno
Aggiunta di un premio discografico	S	100 al mese
Calcolare quanti brani contiene una playlist	L	5000 al giorno

3. Progettazione Concettuale

3.1 Lista delle entità

Se non specificato l'attributo è NOT NULL

- Artista:
    - Cf: char (16) primary key
    - Nome: varchar (40)
    - Cognome: varchar (40)
    - NomedArte: varchar (40) può essere NULL
    - dataNascita: date
    - Nazionalità: char (2)
    - Genere: varchar (15)

- Album:
    - Id: char (5)
    - Prezzo: decimal
    - Titolo: varchar (40)
    - Anno: int

- Etichetta:
    - pIVa: char (11) primary key
    - Nome: varchar (40)
    - Città: varchar (40)

- Brano:
-
    - Id: char (7) primary key
    - Titolo: varchar (40)
    - Durata: int
    - GuadagnoSecondo: decimal
    - DataRilascio: date

- Premio
    - Codice: char (4) primary key
    - Titolo: varchar (40)
    - Società: varchar (40)

L'entità Premio si specializza in una sottocategoria con una generalizzazione totale:

- Discografico:
    - PassaggiRadio: int può essere NULL
    - NumeroStream: int può essere NULL
    - TotaleVendite: int può essere NULL

- Concorso:
    - TitoloConcorso: varchar (40) può essere NULL
    - Data: date può essere NULL

- Playlist:
    - Id: char (7) primary key
    - Id_Utente: char (7) primary key

    - Privata: bool
    - Nome: varchar (40)

- Utente:

    - Id: char (7) primary key
    - Nome: varchar (40)
    - Cognome: varchar (40)
    - DataNascita: date
    - Email: varchar (50)

- Abbonamento:

    - Id: char (6) primary key
    - Id_Utente: varchar (7) primary key
    - Premium: bool
    - MetodoPagamento: varchar (40) può essere NULL
    - Annuale: bool

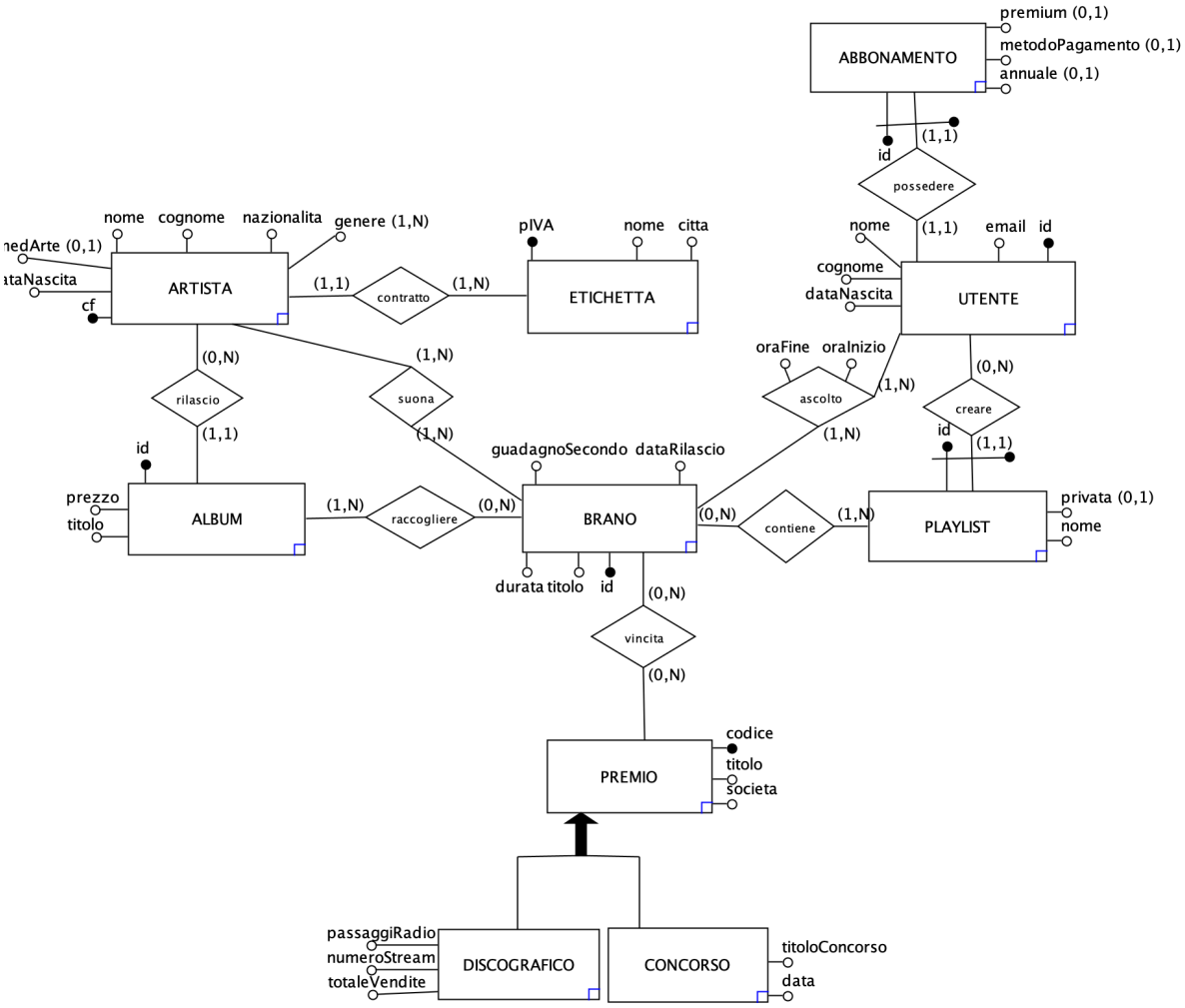
3.2 Tabella delle associazioni

Associazione	Entità coinvolte	Descrizione	Attributi
Rilascio	Artista (0,N) Album (1,1)	Un artista può rilasciare più album o non rilasciarne alcuno, un album può essere rilasciato da un solo artista	Nessuno
Contratto	Artista (1,1) Etichetta (1,N)	Un artista può firmare un solo contratto con l'etichetta discografica, mentre l'etichetta discografica può avere una licenza con uno o più artisti	Nessuno
Suona	Artista (1,N) Brano (1,N)	Un artista può suonare più brani, mentre un brano può essere suonato da uno o più artisti	Nessuno
Raccogliere	Album (1,N) Brano (0,N)	Un album può essere una collezione di uno o più brani ,un brano può essere contenuto in nessuno o più album	Nessuno
Vincita	Brano (0,N) Premio (0,N)	Un brano può vincere zero o più premi, un premio può essere assegnato ad alcun brano o a più brani	Nessuno
Contiene	Brano (0,N) Playlist (1,N)	Un brano può essere aggiunto in nessuna playlist o in più playlist. Una playlist deve contenere almeno un brano	Nessuno
Ascolto	Brano (1,N) Utente (1,N)	Un brano può essere ascoltato una o più volte. Un utente può ascoltare uno o più brani	OraInizio, OraFine: timestamp

Creare	Playlist (1,1) Utente (0,N)	Un utente può creare zero o più playlist differenti. Una playlist può essere creata da un solo utente	Nessuno
Possedere	Utente (1,1) Abbonamento (1,1)	Un utente può avere un solo abbonamento, un abbonamento può appartenere a un solo utente	Nessuno

3.3 Diagramma E-R

Per BRANO si è scelto di inserire un attributo *id* come chiave primaria, per semplicità, velocità di accesso e mancanza di altre possibili valide alternative.



3.4 Vincoli non rappresentabili

Vi sono alcuni vincoli logici non rappresentabili nel diagramma E-R:

- Un utente non può ascoltare due brani contemporaneamente
- L'ora di fine di un ascolto non può essere oltre all'ora di inizio sommato alla durata di un brano.

4. Progettazione Logica

L'attributo *multivalore* genere di *ARTISTA* viene ristrutturato come entità *GENERE* e collegata all'entità originale con una relazione molti a molti.

4.1 Analisi Delle Ridondanze

L'associazione *suona*, tra *BRANO* ed *ARTISTA* può creare ridondanza: l'artista che suona un brano presente all'interno di un album può essere dedotto Tuttavia la ridondanza, oltre a limitare la complessità della base di dati e semplificare le operazioni, non è eliminabile, poiché si assume che non tutti i brani debbano essere raccolti in almeno un album, quindi di sceglie di correre il rischio di inconsistenza.

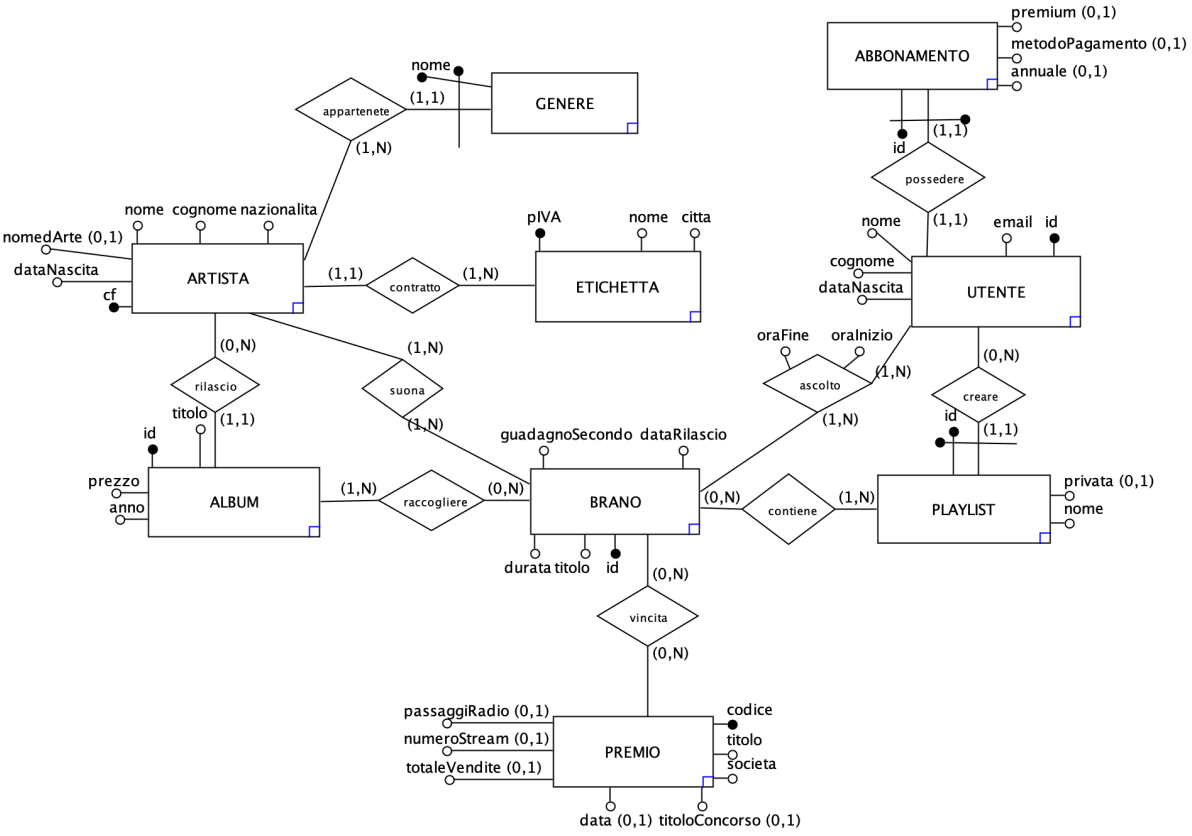
4.2 Eliminazione delle Generalizzazioni

La generalizzazione su PREMIO viene risolta con l'accorpamento delle entità figlie nell'entità genitore per diversi motivi, primo tra tutti per mantenere la struttura semplice e compatta, anche visto il ridotto numero di attributi delle entità figlie. Inoltre è altamente insolito che si debba accedere alle informazioni salvate nelle entità figlie senza che siano richiesti anche i dati sul premio stesso. La scelta è presa in coscienza del fatto che questo limita la possibilità di aggiungere facilmente altri tipi di premi in futuro.

4.3 Scelta degli identificatori primari

Per *ALBUM* la chiave primaria era stata pensata composta da *titolo* e *cf* dell'artista (la chiave esterna verso l'entità *ARTISTA*) ma essendo i entrambi i campi delle stringhe è conveniente inserire un codice univoco *id* per facilitare e alleggerire l'accesso.

4.4 Diagramma E-R ristrutturato



5. Schema relazionale e vincoli di integrità referenziale

Gli attributi sottolineati indicano la chiave primaria, quelli con l'asterisco possono avere valori nulli, di seguito ad ogni relazione sono riportate le chiavi esterne e i relativi attributi coinvolti.

```
GENERE(nome, cf)
      GENERE.cf → ARTISTA.cf
ARTISTA(cf, nome, cognome, nazionalita, dataNascita, nomedArte*,
contratto)
      ARTISTA.contratto → ETICHETTA.pIVA
ETICHETTA(pIVA, nome, citta)
ALBUM(id, titolo, anno, prezzo)
RILASCIO(idAlbum, cfArtista)
      RILASCIO.cfArtista → ARTISTA.cf
      RILASCIO.idBrano → BRANO.id
BRANO(id, titolo, durata, guadagnoSecondo, dataRilascio)
SUONA(idBrano, cfArtista)
      SUONA.cfArtista → ARTISTA.cf
      SUONA.idBrano → BRANO.id
RACCOGLIERE(idBrano, idAlbum)
      RACCOGLIERE.idAlbum → ALBUM.id
      RACCOGLIERE.idBrano → BRANO.id
PREMIO(codice, titolo, societa, data*, titoloConcorso*, passaggiRadio*,
numeroStream*, totaleVendite*)
      PREMIO.idBrano → BRANO.id
VINCITA(idBrano, codPremio)
      VINCITA.idBrano→ BRANO.id
      VINCITA.codPremio→ PREMIO.codice
PLAYLIST(id, idUtente, nome, privata)
      PLAYLIST.idUtente → UTENTE.id
CONTIENE(idBrano, idPlaylist, idUtente)
      CONTIENE.idBrano → BRANO.id
      CONTIENE.idPlaylist → PLAYLIST.idPlaylist
      CONTIENE.idUtente → PLAYLIST.idUtente
ASCOLTO(numAscolto, idUtente, idBrano, oraInizio, oraFine, data)
      ASCOLTO.idUtente → UTENTE.id
      ASCOLTO.idBrano → BRANO.id
UTENTE(id, nome, cognome, dataNascita, email, abbonamento)
      UTENTE.abbonamento → ABBONAMENTO.id
ABBONAMENTO(id, premium, metodoPagamento*, annuale*)
```

In *ABBONAMENTO* p*remium* e *annuale* sono considerati campi booleani per maggiore leggibilità della base di dati.
In ASCOLTO si sceglie come chiave primaria *numAscolto* id auto-incrementale piuttosto che una chiave composta complessa e poco funzionale all'accesso in una tabella probabilmente sarà spesso al centro di interrogazioni del database.

6. Query

1. Il nome delle etichette discografiche che producono artisti vincitori di almeno due dischi d'oro o di platino, con i relativi artisti:

```
SELECT DISTINCT ETICHETTA.nome AS etichetta, T.nome, cognome, nomedArte
FROM ETICHETTA
JOIN (
      SELECT contratto, nome, cognome, nomedArte
      FROM ARTISTA
      JOIN SUONA ON ARTISTA.cf = SUONA.cfArtista
      JOIN VINCITA ON VINCITA.idBrano = SUONA.idBrano
      WHERE codPremio = 'DORO' OR codPremio = 'DPLA'
) AS T ON T.contratto = ETICHETTA.pIVA
```

	etichetta character varying (40)	nome character varying (40)	cognome character varying (40)	nomedarte character varying (40)
1	Big Hit Music	Adele Laurie Blue	Adkins	Adele
2	Sony Music	Raffaella Maria Roberta	Pelloni	Raffaella Carrà

2. L'album di un'artista del genere 'POP' con più brani premiati dalla discografia:

```
SELECT idAlbum, numPremiAlbum
FROM(
  SELECT RACCOGLIERE.idAlbum, SUM(numPremi) AS numPremiAlbum
  FROM (
    SELECT RILASCIO.idAlbum
    FROM GENERE
    JOIN RILASCIO ON  genere.cf = RILASCIO.cfArtista
    WHERE GENERE.nome='POP') AS AlbumPOP
  JOIN RACCOGLIERE ON RACCOGLIERE.idAlbum = AlbumPOP.idAlbum
  JOIN (
    SELECT COUNT(codPremio) AS numPremi, idBrano
    FROM VINCITA
    JOIN PREMIO ON VINCITA.codPremio = PREMIO.codice
    WHERE numeroStream IS NOT NULL
    GROUP BY idBrano) AS ConteggioPremi
    ON RACCOGLIERE.idBrano = ConteggioPremi.idBrano
  GROUP BY RACCOGLIERE.idAlbum
) AS temp
ORDER BY  numPremiAlbum DESC
LIMIT 1
```

	idalbum character (7)	numpremialbum numeric
1	GHTRS	3

3. Artisti americani nati prima dell'11 Novembre dell'anno 2000 con più di un brano registrato:

```
SELECT ARTISTA.nome, ARTISTA.cognome, ARTISTA.nomedarte, COUNT(SUONA.idbrano) AS
NumeroBrani
FROM ARTISTA
JOIN SUONA ON ARTISTA.cf = SUONA.cfartista
WHERE ARTISTA.datanascita <= '2000-02-11' AND ARTISTA.nazionalita = 'US'
GROUP BY ARTISTA.cf
HAVING COUNT(SUONA.idbrano) > 1;
```

	nome character varying (40)	cognome character varying (40)	nomedarte character varying (40)	numerobrani bigint
1	Louise Veronica	Ciccone	Madonna	5
2	Giselle	Knowles-Carter	Beyoncé	2
3	James	Hetfiels	Metallica	3
4	Michael Joseph	Jackson	Michael Jackson	2
5	Cherilyn	Sarkisian	Cher	2
6	Anna Mae	Bullock	Tina Turner	2
7	Justin	Timberlake	[null]	2

4. Gli utenti con il numero di artisti totali ascoltati per ciascun genere musicale:

```
SELECT DISTINCT UTENTE.nome,UTENTE.cognome, GENERE.nome AS Genere,
COUNT(DISTINCT ARTISTA.cf) AS NumeroArtistiAscoltati
FROM UTENTE
JOIN ASCOLTO ON UTENTE.id =ASCOLTO.idUtente
JOIN BRANO ON ASCOLTO.idBrano =BRANO.id
JOIN SUONA ON BRANO.id= SUONA.idBrano
JOIN ARTISTA ON SUONA.cfartista= ARTISTA.cf
JOIN GENERE ON ARTISTA.cf= GENERE.cf
GROUP BY UTENTE.id, GENERE.nome
```

	nome character varying (40)	cognome character varying (40)	genere character varying (15)	numeroartistiascoltati bigint
1	Mario	Rossi	DANCE	2
2	Tony	Dinozzo	POP	3
3	Piero	Angela	POP	4
4	Lana	Parilla	DANCE	1
5	Angela	Bassett	ELETTRONICA	1
6	Piero	Angela	DANCE	2
7	Mark	Ruffalo	CLASSICAL	1
8	AJ	Cook	BLUES	1
9	Mark	Ruffalo	POP	5
25	Penelope	Cruz	POP	4
26	Piero	Angela	ELETTRONICA	1
27	Penelope	Cruz	CLASSICAL	1

5. Gli utenti che hanno un abbonamento Premium e hanno aggiunto alle proprie playlist almeno una canzone di un determinato artista a una playlist privata:

```
SELECT UTENTE.nome AS nome_utente, SUONA.cfArtista AS artista_aggiunto,
PLAYLIST.nome AS playlist, BRANO.titolo AS brano_aggiunto
FROM UTENTE
JOIN ABBONAMENTO ON UTENTE.abbonamento = ABBONAMENTO.id
JOIN PLAYLIST ON UTENTE.id = PLAYLIST.idutente
JOIN CONTIENE ON PLAYLIST.id = CONTIENE.idplaylist
JOIN BRANO ON CONTIENE.idbrano =BRANO.id
JOIN SUONA ON BRANO.id= SUONA.idbrano
WHERE ABBONAMENTO.premium = TRUE
AND PLAYLIST.privata = TRUE
AND SUONA.cfArtista = 'CU78DS4VHSK5BVR5';
```

	nome_utente character varying (40)	artista_aggiunto character (16)	playlist character varying (40)	brano_aggiunto character varying (40)
1	Jungkook	CU78DS4VHSK5BVR5	Purple Hearts	Bad Decisions

6. VINCOLI

6.1. Utenti che ascoltano due brani contemporaneamente

```
SELECT A1.idUtente, A1.idBrano AS brano1, A2.idBrano AS brano2
FROM ASCOLTO A1
JOIN ASCOLTO A2 ON A1.idUtente = A2.idUtente
WHERE A1.idBrano <> A2.idBrano
AND ((A1.oraInizio >= A2.oraInizio AND A1.oraInizio < A2.oraFine)
OR (A2.oraInizio >= A1.oraInizio AND A2.oraInizio < A1.oraFine))
```

Vincolo non rappresentabile: un utente non può ascoltare due brani contemporaneamente, quindi la relazione risultante dalla query dovrebbe essere vuota affinché il vincolo sia rispettato.

6.2. Ascolti in cui l'ora di fine risulta maggiore dell'ora di inizio sommata alla durata del brano

```
SELECT *
FROM ASCOLTO
JOIN BRANO ON ASCOLTO.idBrano = BRANO.id
WHERE ASCOLTO.oraFine > ASCOLTO.oraInizio + INTERVAL '1 second' * BRANO.durata
```

Vincolo non rappresentabile: l'ascolto non può avere durata maggiore della durata del brano ascoltato, quindi la relazione risultante dalla query dovrebbe essere vuota affinché il vincolo sia rispettato.
7. Inserimento di un nuovo brano

```
BEGIN;
INSERT INTO BRANO(id, titolo, durata, guadagnoSecondo, dataRilascio) VALUES
('ABC0012', 'rumore' '320', 0,003', '12/07/1970');
INSERT INTO SUONA(idBrano, cfArtista) VALUES ('ABC0012', 'BBHN567VHSK554CF');
COMMIT;
```

7. Indici

- Indice sulla colonna *id* della tabella *BRANO*:
  `CREATE INDEX idx_brano_id ON BRANO(id);`
  Questo indice è utile per accelerare le operazioni di ricerca e join che coinvolgono la colonna *id* nella tabella *BRANO*. Questo indice potrebbe migliorare le prestazioni perchè molte delle operazioni svolte sul database coinvolgono questa tabella.
- Indice sulle colonne id e idUtente della tabella PLAYLIST
  `CREATE INDEX idx_playlist_id ON PLAYLIST(id, idUtente);`
  Un indice su queste colonne è utile per migliorare le prestazioni delle query che coinvolgono la tabella *PLAYLIST* poiché la combinazione di *idUtente* e *id* forma la chiave primaria della tabella ed è probabile che vengano eseguite frequentemente operazioni di ricerca e join basate su queste colonne.
  Creando questo indice, è possibile accedere più rapidamente alle righe desiderate durante l'esecuzione di query che filtrano o uniscono la tabella *PLAYLIST* in base a queste colonne, vista anche la complessità del tipo di dato di *id* e *idUtente*.

In entrambi i casi si sceglie un indice di tipo B+Tree, preferibile per ricerca di valore esatto, query di intervallo (anche se poco frequenti), ordinamento e flessibilità.

8. Codice C++

8.1 Restore del database

Per poter eseguire il programma è necessario prima eseguire il restore del database, utilizzando Postgres e pgAdmin sarà sufficiente creare un database vuoto (*database_name*) e successivamente eseguire da terminale il seguente comando:

```
psql -U <username> -d <database_name> -f APOLLON_dump.sql
```

8.2 Descrizione e utilizzo del codice

Prima di poter compilare il file è necessario copiare i file *libpq.dll* e *libpq.lib* in *"./dependencies/lib"* e copiare *libpq-fe.h*, *pg_consig_ext.h* e *postgres_ext.h* in *"./dependencies/include"*, dove "./" è il percorso della directory che contiene il *apollon.cpp*.

Per la compilazione è del codice C++ è sufficiente eseguire il seguente comando all'interno della cartella in cui si trova il file *apollon.cpp:*

```
g++ apollon.cpp -L dependencies/lib -lpq -o apollon
```

Per poi eseguire il programma con *`./apollon`*

Per poter eseguire il login al database è necessario ridefinire le seguenti costanti definite all'inizio del file *apollon.cpp*

`PG_HOST, PG_USER, PG_PASS, PG_PORT, PG_DB`

All'avvio verrà mostrata a schermo la lista delle query eseguibili all'interno del programma.
1. Il nome delle etichette discografiche che producono artisti vincitori di almeno due dischi d'oro o di platino, con i relativi artisti.
2. L'album di un'artista del genere POP con più brani premiati dalla discografia.
3. Il numero totale di brani registrati dai soli artisti americani nati prima dell'11 Novembre dell'anno 2000.
4. Gli utenti con il numero di artisti totali ascoltati per ciascun genere musicale.
5. Gli utenti che hanno un abbonamento Premium e hanno aggiunto alle proprie playlist almeno una canzone di un determinato artista a una playlist privata.
6. Vincoli
    1. Utenti che ascoltano due brani contemporaneamente
    2. Ascolti il cui l'ora di fine risulta maggiore dell'ora di inizio sommata alla durata del brano
7. Inserimento di un nuovo brano

Per eseguire una query bisogna inserire da tastiera il numero della query scelta, mentre per terminare l'esecuzione del programma va inserito `-1`. Alcune query (la numero 5 e 7) richiedono l'inserimento di alcuni parametri da parte dell'utente.

8.3 Documentazione del codice

Funzioni di utility
- `string getTodayDate()`: ritorna la data corrente in formato di stringa
- `int selectQuery()`: stampa la lista delle query e prende in input l'intero corrispondente alla query desiderata
- `int selectVincolo()`: stampa la lista dei vincoli e prende in input l'intero corrispondente al vincolo desiderato

Funzioni relative alle query
- `void checkResults(PGresult* res, const PGconn* conn)`: verifica che i risultati di una query siano consistenti
- `void executeQuery(PGconn * conn, const char* q)`: esegue una query generica (ovvero non parametrica)
- `void printArtists(PGconn * conn)`: stampa la lista degli artisti coi corrispondenti codici fiscali
- `void premiumUsersWithPlaylistContainingSongsFrom(PGconn * conn)`: esegue la query numero 5, compreso l'inserimento dei parametri
- `void insertSongWithArtist(PGconn * conn)`: esegue la query di inserimento numero 7, compreso l'inserimento dei parametri

Funzioni della libreria *libpq-fe.h*
- `PGconn *PQconnectdb(const char *conninfo)`: stabilisce una connessione a un database PostgreSQL specificando i parametri di connessione, come nome utente, password, nome database, host e porta.
- `PGresult *PQexec(PGconn *conn, const char *query)`: consente di eseguire query SQL nel database.
- `ExecStatusType PQresultStatus(const PGresult *res)`: utile per determinare se la query è stata eseguita con successo o se si è verificato un errore durante l'esecuzione. In base al codice di stato restituito, è possibile prendere decisioni o gestire gli errori in modo appropriato nel codice.
- `int PQntuples(const PGresult *res)`: ritorna il numero di righe di un risultato di una query
- `int PQnfields(const PGresult *res)`: ritorna il numero di campi di un risultato di una query
- `char *PQerrorMessage(const PGconn *conn)`: restituisce il messaggio di errore associato all'ultima operazione