Andrea Santi 2084624
Caterina Vallotto 2076434

Progetto Basi Di Dati - DB-records

Indice:

1. Abstract
2. Analisi dei requisiti
3. Progettazione concettuale
4. Progettazione logica
5. Implementazione dello schema logico
6. Definizione delle Query e degli indici associati
7. Discussione dell'applicazione software che accede al DB per effettuare le Query del punto 6

1. Abstract

La DataBase-records (DB-records) è una casa discografica che opera sul territorio nazionale e che cura la vita musicale di una molteplicità di artisti, da cantanti solisti a band, fino ad arrivare anche a singoli musicisti, producer o strumentisti che siano. Oltre ad assisterli nel loro processo di crescita artistica, si occupa anche di assistenza nell'incisione e nella pubblicazione di album, ma anche singole canzoni, che possono appartenere a svariati generi, in base alla propensione musicale di ogni artista. La DB-records si impegna a stringere contratti unici che vadano a valorizzare le qualità dei singoli focalizzandosi principalmente su quest'ultime piuttosto che sul raggiungimento immediato di risultati poiché, come detto in precedenza, l'obiettivo è e deve rimanere quello della crescita artistica. Nonostante ciò, gli artisti che collaborano con la DB-records hanno già raggiunto grandi successi come testimoniato dalle numerose certificazioni ottenute e dai numerosi sold-out riscontrati dai vari concerti 'live' in locali, palazzetti e stadi. Concretamente la casa discografica offre ai propri assistiti svariati e moderni studi di registrazione in cui poter dar vita alla propria musica. Per riuscire a fornire il miglior supporto possibile la DB-records ha quindi optato per la creazione di un database in modo tale da tenere monitorati i vari aspetti della vita artistica dei propri clienti.

2. Analisi dei requisiti

Si vuole quindi creare un database che vada a gestire la realtà sopra descritta.
In particolare un artista è descritto da:

- *Nome d'arte*
- Nome
- Cognome
- Codice Fiscale
- Data di nascita
- Anno di inizio attività
- Mail
- Telefono

L'artista viene interpretato come cantante singolo, band o musicista.
Ogni cantante singolo è caratterizzato da:

- Tipo di voce (soprano, contralto, tenore, basso)

Ogni band invece può avere un cantante frontman e deve essere composta da uno o più musicisti, indicando:

- Numero componenti

La figura del musicista è definita da:

- Competenza strumentale (principiante, intermedio, esperto)

e può essere intesa come producer o strumentista; per quest'ultimo vengono messi in evidenza:

- Strumento/i

Gli artisti si contraddistinguono per i generi musicali trattati, dove indichiamo:

- *Tipo*

Ogni artista, per definirsi tale, deve aver composto almeno una canzone che appartiene ad un determinato genere. Ad una canzone si associano:

- *Id*
- Titolo
- Durata
- Data di uscita
- Numero ascolti

Oltre alla singola canzone, un artista può anche incidere uno o più album formato da molteplici canzoni, sempre appartenenti a uno o più generi, caratterizzati da:

- *Id*
- Nome
- Durata
- Data di uscita
- Numero ascolti
- Numero canzoni

Sia le singole canzoni che gli album devono essere incisi in uno studio di registrazione delle casa discografica, descritti da:

- *Città*
- *Indirizzo*

Dal momento in cui una canzone risulta essere particolarmente originale, la DB-records assegna certificazioni caratterizzate da:

- *Id*
- Data

Le certificazioni possono essere del tipo oro, platino e diamante.

Ogni artista può fare uno o più concerti live, in cui si ha:

- *Data*
- Indirizzo
- SoldOut
- Numero posti

specificando però se il concerto si svolge in un locale, in uno stadio o in un palazzetto.

Infine, per rappresentare legalmente il rapporto tra la casa discografica e l'artista, si stipula un contratto, firmato in una certa data, descritto da:

- *Id*
- Durata (determinato, indeterminato)
- Remunerazione

3. Progettazione concettuale

3.1 Lista entità

Se non specificato, gli attributi sono NOT NULL

- **Artista**: rappresenta il cliente da tutelare
    - nomeArte: varchar(20)
    - CF: varchar(16) NULL
    - nome: varchar(20) NULL
    - cognome: varchar(20) NULL
    - annoNascita: date NULL
    - annoInizioAttività: int

    - telefono: varchar(9) NULL
    - mail: varchar(50) NULL
- **Cantante**: rappresenta un singolo artista
    - tipoVoce: varchar(20)
- **Band**: rappresenta un singolo artista
    - numComponenti: int
- **Musicista**: rappresenta un singolo artista
    - competenzaStrumentale: varchar(20)
- **Producer**: rappresenta un musicista
- **Strumentista**: rappresenta un musicista
    - strumento: attributo (1,N)
- **Concerto**: rappresenta un esibizione 'live' dell'artista
    - data: date
    - nomeArte: varchar(20) (da entità Artista)
    - indirizzo: varchar(50)
    - soldOut: bool
    - numeroPosti: int
- **Stadio**: rappresenta un luogo in cui si può svolgere un concerto
- **Locale**: rappresenta un luogo in cui si può svolgere un concerto
- **Palazzetto**: rappresenta un luogo in cui si può svolgere un concerto
- **Contratto**: rappresenta il rapporto che lega la casa discografica e un singolo artista
    - id: varchar(5)
    - durata: varchar(20)
    - remunerazione: int
- **Genere**: rappresenta lo stile dell'artista che, interpretando più generi, può comporre album e canzoni di tipologie differenti
    - tipo: varchar(20)
- **Album**: rappresenta un insieme di canzoni dell'artista "riunite" in un unico progetto
    - id: varchar(5)
    - nome: varchar(20)
    - numCanzoni: int
    - durata: int
    - numAscolti: int
    - dataUscita: date
- **Canzone**: rappresenta un brano musicale cantato e interpretato dall'artista
    - id: varchar(5)
    - durata: int
    - titolo: varchar(20)
    - numAscolti: int
    - dataUscita: date
- **Studio di registrazione**: rappresenta il luogo in cui vengono registrati album e canzoni
    - città: varchar(20)
    - indirizzo: varchar(50)
- **Certificazione**: rappresenta, ufficialmente, il 'premio' ottenuto dall'artista, non per il raggiungimento di un determinato numero di ascolti, ma bensì per l'originalità della sua produzione musicale
    - id: varchar(5)
    - data: date
- **Disco d'oro**: rappresenta la prima tipologia di certificazioni
- **Disco di platino**: rappresenta la seconda tipologia di certificazioni
- **Disco di diamante**: rappresenta la terza e ultima tipologia di certificazioni

3.2 Lista relazioni

- **Cantante-Band**: (1:1)
    - Ogni cantante può fare parte di una sola band
    - Ogni band può avere un solo cantante come frontman
- **Musicista-Band**: (1:N)
    - Ogni musicista può appartenere ad una sola band
    - Ogni band deve essere composta da più musicisti
- **Concerto-Artista**: (1:N)
    - Ogni concerto deve essere opera di un artista
    - Ogni artista durante la sua carriera può prendere parte a più concerti
- **Artista-Contratto**: (1:1) → attributo: dataFirma
    - Ogni artista può firmare un solo contratto
    - Ogni contratto deve essere firmato da un solo artista
- **Canzone-Artista**: (1:N)
    - Ogni canzone deve essere composta da un solo artista
    - Ogni artista può comporre almeno una canzone (o anche più)
- **Artista-Genere**: (N:N)
    - Ogni artista deve saper interpretare uno o più generi
    - Ogni genere invece può essere interpretato da uno o più artisti
- **Canzone-Genere**: (1:1)
    - Una canzone ha un unico genere
    - Un genere può appartenere a una canzone
- **Album-Genere**: (1:1)
    - Un album ha un unico genere
    - Un genere può appartenere a un album
- **Album-Artista**: (1:N)
    - Un album appartiene unicamente a un artista
    - Un artista può comporre uno o più album
- **Canzone-Album**: (1:N)
    - Una canzone può fare parte di un album
    - Un album deve avere più canzoni
- **Album-Studio di registrazione**: (1:N)
    - Un album deve essere registrato in uno studio di registrazione
    - In uno studio di registrazione possono essere registrati più album
- **Canzone-Studio di registrazione**: (1:N)
    - Una canzone deve essere registrata in uno studio di registrazione
    - In uno studio di registrazione possono essere registrate più canzoni
- **Album-Certificazione**: (N:N)
    - Ogni album può ricevere una o più certificazioni
    - Ogni certificazione può premiare uno o più album
- **Canzone-Certificazione**: (N:N)
    - Ogni canzone può essere premiata con una o più certificazioni
    - Ogni certificazione può essere assegnata ad una o più canzoni

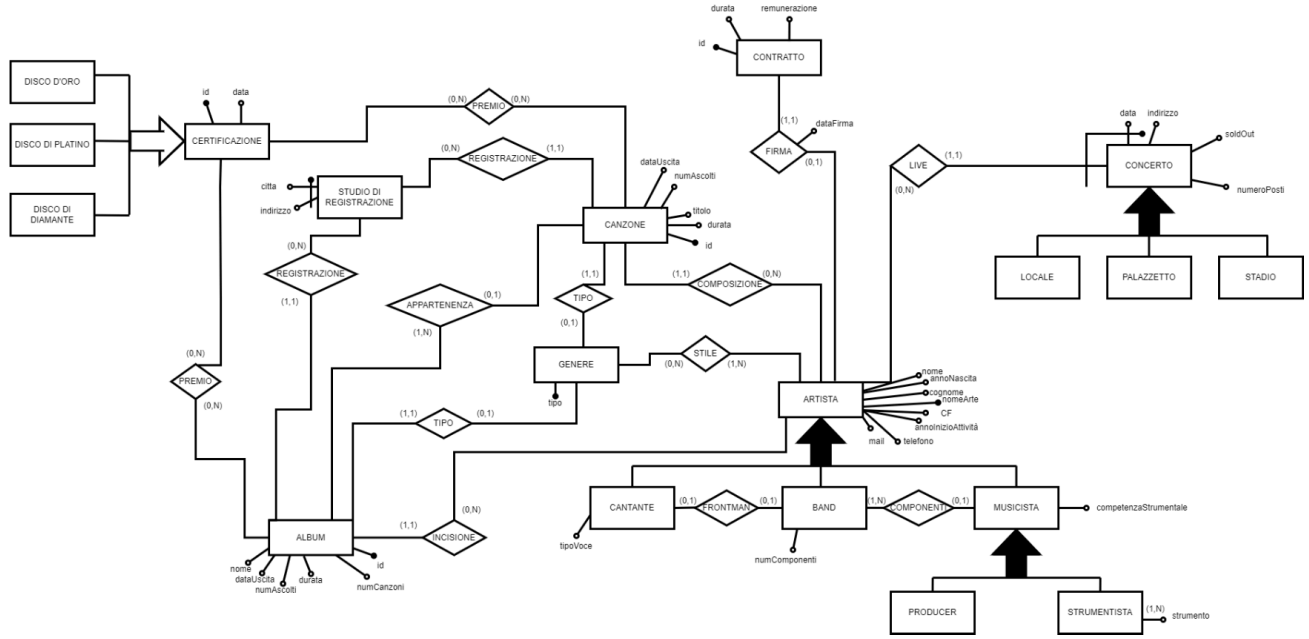
3.3 Lista generalizzazioni

- **Concerto** è una generalizzazione totale di Locale, Palazzetto, Stadio
- **Certificazione** è una generalizzazione parziale di Disco d'oro, Disco di platino, Disco di diamante
- **Artista** è una generalizzazione totale di Cantante, Band, Musicista
- **Musicista** è una generalizzazione totale di Producer, Strumentista

3.4 Vincoli testuali

- Ogni band (oltre ad essere vista come un unico artista), può anche essere composta da soli musicisti (tipo nella realtà Jazz) oppure avere un unico frontman cantante con più musicisti
- Ogni artista ha un nome d'arte univoco, non ci possono quindi essere due o più artisti con lo stesso nome d'arte
- La DB-records attribuisce proprie certificazioni che si basano sull'originalità delle composizioni
- Dato che in "Artista" sono presenti band, i campi riguardanti gli aspetti anagrafici/personali (nome, cognome, CF…) possono essere NULL
- Si osservi che, per quanto riguarda le band, i singoli componenti stipulano contratti con la casa discografica e non l'intera band

3.5 Schema E-R



4. Progettazione Logica

4.1 Analisi delle ridondanze

NB: si noti che i volumi presi in considerazione hanno scopo puramente dimostrativo e non rappresentano i valori reali con cui il database è implementato.

Analizzando lo schema E-R possiamo notare che l'attributo "numCanzoni" dell'entità "Album" potrebbe essere calcolato dalla relazione "Appartenenza". Tale attributo è infatti la somma di tutte le canzoni che fanno parte di un album. Analizziamo quindi le operazioni riguardanti questo attributo per capire se mantenerlo o no.

- **Operazione 1** (50 volte alla settimana): inserimento di una nuova canzone da aggiungere ad un album già esistente
- **Operazione 2** (100 volte alla settimana): visualizzare il numero di canzoni di un album

Concetto	Costrutto	Volume
Album	Entità	50.000
Canzone	Entità	600.000
Appartenenza	Relazione	500.000

Con ridondanza

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Canzone	Entità	1	Scrittura
Appartenenza	Relazione	1	Scrittura
Album	Entità	1	Lettura
Album	Entità	1	Scrittura

Costo: 50 x 1 + 50 x 1 + 50 x 1 + 50 x 1 = 150 in scrittura + 50 in lettura

- **Operazione 2**

Concetto	Costrutto	Accesso	Tipo
Album	Entità	1	Lettura

Costo: 1 x 100 = 100

Assumendo costo doppio per gli accessi in scrittura
Costo settimanale totale: 150 x 2 + 50 + 100 = 450

Senza ridondanza

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Canzone	Entità	1	Scrittura
Appartenenza	Relazione	1	Scrittura

Costo: 50 x 1 + 50 x 1 = 100 in scrittura

- **Operazione 2** (ogni album ha mediamente 10 canzoni)

Concetto	Costrutto	Accesso	Tipo
Album	Entità	1	Lettura
Appartenenza	Relazione	10	Lettura

Costo: 1 x 100 + 10 x 100 = 1100

Assumendo costo doppio per gli accessi in scrittura
Costo settimanale totale: 100 x 2 + 1100 = 1300

Conviene quindi mantenere l'attributo "numCanzoni"
Possiamo poi notare che l'attributo "numComponenti" dell'entità figlia "Band" potrebbe essere calcolato dalla relazione "Frontman" e "Componenti". Analizziamo quindi le operazioni riguardanti questo attributo per capire se mantenerlo o no.

- **Operazione 1** (2 volte al mese): un membro della band (cantante) lascia il gruppo

NB: l'operazione di eliminazione viene considerata come SCRITTURA dato che si va ad accedere al database con l'obbiettivo di modificarlo.

- **Operazione 2** (100 volte al mese): visualizzare il numero di membri di una band

Concetto	Costrutto	Volume
Band	Entità figlia	20
Cantante	Entità figlia	65
Musicista (producer o strumentista)	Entità figlia	100
Frontman	Relazione	14
Componenti	Relazione	80

<u>Con ridondanza - prima della ristrutturazione della generalizzazione</u>

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Cantante	Entità figlia	1	Scrittura
Frontman	Relazione	1	Scrittura
Band	Entità figlia	1	Lettura
Band	Entità figlia	1	Scrittura

Costo: 2 x 1 + 2 x 1 + 2 x 1 + 2 x 1 = 6 in scrittura + 2 in lettura

- **Operazione 2**

Concetto	Costrutto	Accesso	Tipo
Band	Entità figlia	1	Lettura

Costo: 1 x 100 = 100

Assumendo costo doppio per gli accessi in scrittura
Costo mensile totale: 6 x 2 + 2 + 100 = 114

<u>Senza ridondanza - prima della ristrutturazione della generalizzazione</u>

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Cantante	Entità figlia	1	Scrittura
Frontman	Relazione	1	Scrittura

Costo: 2 x 1 + 2 x 1 = 4 in scrittura

- **Operazione 2** (ogni band ha mediamente 4 musicisti e un frontman)

Concetto	Costrutto	Accesso	Tipo
Band	Entità figlia	1	Lettura
Frontman	Relazione	1	Lettura
Componenti	Relazione	4	Lettura

Costo: 1 x 100 + 1 x 100 + 4 x 100 = 600

Assumendo costo doppio per gli accessi in scrittura
Costo settimanale totale: 4 x 2 + 600 = 608

Anche in questo caso è conveniente mantenere l'attributo "numComponenti"

Con ridondanza - dopo la ristrutturazione della generalizzazione

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Cantante	Entità	1	Scrittura
Frontman	Relazione	1	Scrittura
Band	Entità	1	Lettura
Band	Entità	1	Scrittura

Costo: 2 x 1 + 2 x 1 + 2 x 1 + 2 x 1 = 6 in scrittura + 2 in lettura

- **Operazione 2**

Concetto	Costrutto	Accesso	Tipo
Band	Entità	1	Lettura

Costo: 1 x 100 = 100

Assumendo costo doppio per gli accessi in scrittura
Costo mensile totale: 6 x 2 + 2 + 100 = 114

Senza ridondanza - dopo la ristrutturazione della generalizzazione

- **Operazione 1**

Concetto	Costrutto	Accesso	Tipo
Cantante	Entità	1	Scrittura
Frontman	Relazione	1	Scrittura

Costo: 2 x 1 + 2 x 1 = 4 in scrittura

- **Operazione 2** (ogni band ha mediamente 4 musicisti e un frontman)

Concetto	Costrutto	Accesso	Tipo
Band	Entità	1	Lettura
Frontman	Relazione	1	Lettura
Componenti - producer	Relazione	1	Lettura
Componenti - strumentisti	Relazione	3	Lettura

Costo: 1 x 100 + 1 x 100 + 1 x 100 + 3 x 100 = 600

Assumendo costo doppio per gli accessi in scrittura
Costo settimanale totale: 4 x 2 + 600 = 608

Anche in questo caso conviene mantenere l'attributo "numComponenti" dato che l'analisi della ridondanza ha prodotto gli stessi risultato del caso "prima della ristrutturazione della generalizzazione"

4.2 Eliminazione di attributi multivalore

Nell'entità figlia "Strumentista" è presente l'attributo multivalore "strumento". Procediamo quindi alla sua eliminazione aggiungendo una nuova entità "Strumento" con un attributo (che è anche chiave primaria):

- nome: indica il nome dello strumento

Quindi "Strumentista" sarà ora messo in relazione a "Strumento" tramite una relazione "Utilizzo" che ha cardinalità massima (N:N) dato che:

- Uno strumentista deve suonare almeno uno strumento
- Uno strumento può essere suonato da più strumentisti

4.3 Eliminazione delle generalizzazioni

- **Certificazione**: è una generalizzazione parziale, legata da una relazione molti a molti sia con "Album" che con "Canzone". L'opzione più conveniente è accorpare le figlie nell'entità padre aggiungendogli il seguente attributo:
  - tipologia: contiene la tipologia della certificazione (oro, platino o diamante)
- **Concerto**: questa invece è una generalizzazione totale, legata da una relazione uno a molti con l'entità "Artista". Le entità figlie "Locale", "Palazzetto", "Stadio" non hanno ulteriori attributi/relazioni rispetto all'entità padre, quindi anche in questo caso conviene accorpare le figlie nel padre aggiungendo un attributo:
  - luogo: indica se il concerto è svolto in un locale, in un palazzetto o in uno stadio

  Dato che le entità figlie non avevano ulteriori attributi, non è stato necessario aggiungerne di opzionali a "Concerto", quindi il valore degli attributi NULL è pari a zero.
- **Musicista**: anche musicista è una generalizzazione totale, la cui figlia "Producer" non ha attributi o relazioni, mentre la figlia "Strumentista" presenta una relazione con l'entità "Strumento" (ottenuta dal punto 4.2). In questo caso optiamo per accorpare il padre nelle figlie in modo da non dover modificare la cardinalità minima della relazione tra "Strumentista" e "Strumento" (che diventerebbe una relazione 'opzionale'). Inoltre non abbiamo attributi con valore NULL. Otteniamo perciò che la generalizzazione "Artista" non avrà più l'entità figlia "Musicista" ma bensì due nuove entità "Producer" e "Strumentista", ognuna delle quali è messa in relazione con "Band"
- **Artista**: per quanto riguarda questa generalizzazione totale, l'opzione più conveniente è sostituire le generalizzazioni con relazioni in quanto le relazioni presenti tra le varie entità figlie renderebbero le altre opzioni (accorpamento nel padre/accorpamento nelle figlie) di difficile gestione, dato che:

- accorpamento del padre nelle figlie: per ogni entità figlia (“Cantante”, “Band”, “Producer”, “Strumentista”) si dovrebbero aggiungere gli attributi del padre (“Artista”) e creare delle relazioni con tutte le entità a cui il padre era ‘collegato’ (ovvero “Album”, “Genere”, “Contratto”, “Canzone”, “Concerto”) che risulterebbe di difficile gestione (dato il grande numero di relazioni che si creerebbero, ovvero 5 per ognuna delle 4 entità figlie, senza contare le relazioni già presenti per rappresentare la band)
- accorpamento delle figlie nel padre: si dovrebbero aggiungere tutti i singoli attributi di ogni entità figlia al padre, aumentando i valori NULL all’interno di “Artista”; inoltre le relazioni presenti tra le quattro entità figlie non sarebbero rappresentabili tramite la semplice aggiunta di attributi

L’opzione ottimale quindi per rappresentare e mantenere le relazioni presenti è trasformare le generalizzazioni in relazioni.

4.4 Partizionamento/accorpamento di entità e relazioni

Dato che la DB-records privilegia il lato artistico e non puramente contrattuale dei propri clienti, ha deciso di partizionare l'entità “Artista” in due entità:

- Artista: rappresenta i dati prevalentemente artistici dei clienti (come nomeArte e annoInizioAttività)
- Cliente: contiene tutti i dati anagrafici e personali di ogni cliente della DB-records (CF, nome, cognome, mail, telefono, annoNascita)

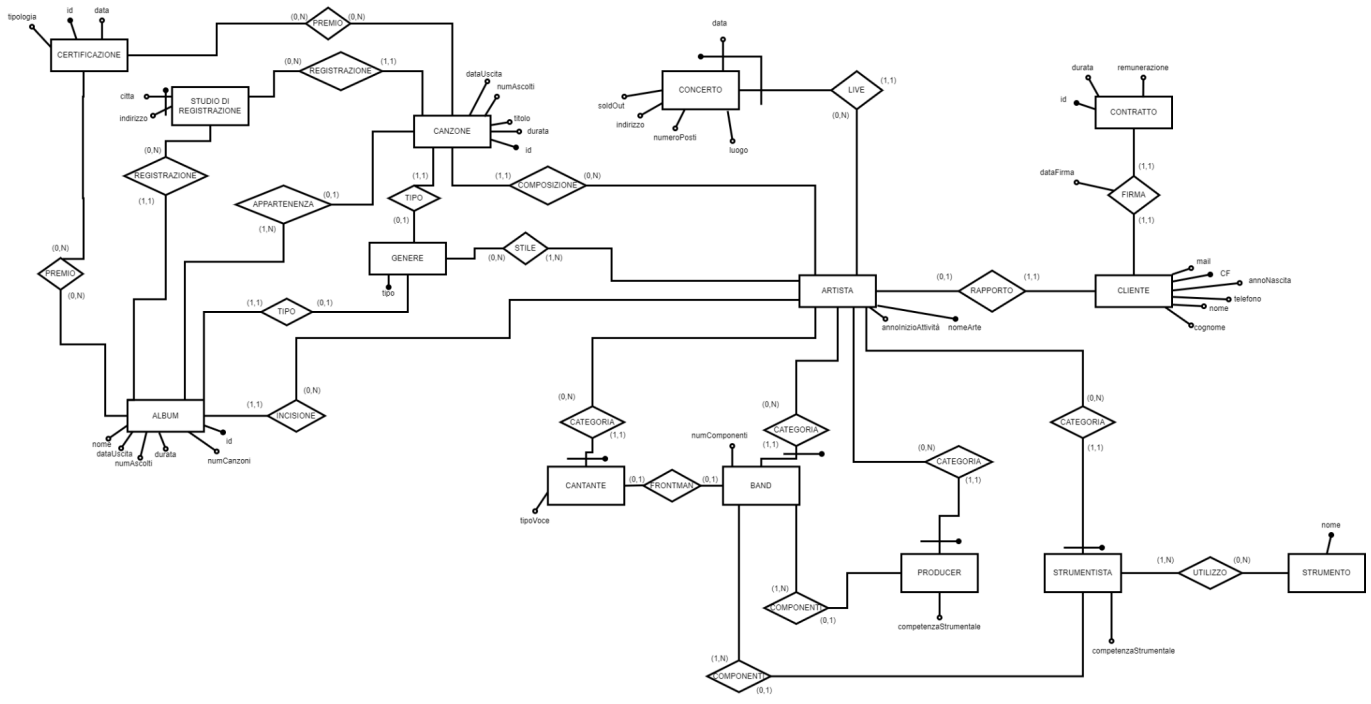
Ovviamente “Artista” e “Cliente” sono messe in relazione da “Rapporto”. Inoltre questo partizionamento permette di dividere gli ambiti che riguardano l’aspetto pubblico-artistico e privato-contrattuale di ogni cliente. Infine, si osservi che, con tale ‘separazione’, nell’entità “Artista” non saranno più presenti valori NULL; si è andato a quindi a privilegiare la diminuzione dei valori NULL piuttosto che la minimizzazione del numero di tabelle.

4.5 Scelta degli identificatori primari

Gli identificatori delle nuove entità generate dalla ristrutturazione del diagramma E-R sono:

- Artista: nomeArte
- Cliente: CF
- Cantante, Band, Producer, Strumentista: chiave esterna di Artista (nomeArte)
- Strumento: nome

4.6 Diagramma schema ristrutturato



4.7 Descrizione schema relazionale + vincoli di integrità referenziale

Artista (nomeArte, annoInizioAttività)
Cliente (CF, nome, cognome, annoNascita, telefono, mail, artista)
    Cliente.artista → Artista.nomeArte
Contratto (id, durata, remunerazione, dataFirma, cliente)
    Contratto.cliente → Cliente.CF
Concerto (data, artista, soldOut, indirizzo, numeroPosti, luogo)
    Concerto.artista → Artista.nomeArte
Cantante (artista, tipoVoce)
    Cantante.artista → Artista.nomeArte
Band (artista, numComponenti, frontman*)
    Band.artista → Artista.nomeArte
    Band.frontman → Cantante.artista
Producer (artista, competenzaStrumentale, band*)
    Producer.artista → Artista.nomeArte
    Producer.band → Band.artista
Strumentista (artista, competenzaStrumentale, band*)
    Strumentista.artista → Artista.nomeArte
    Strumentista.band → Band.artista
Strumento (nome)
Utilizzo (strumentista, strumento)
    Utilizzo.strumentista → Strumentista.nomeArte
    Utilizzo.strumento → Strumento.nome
Genere (tipo)
Stile (artista, genere)
    Stile.artista → Artista.nomeArte
    Stile.genere → Genere.tipo
StudioRegistrazione (città, indirizzo)
Album (id, nome, numCanzoni, durata, numAscolti, dataUscita, artista, genere, cittàReg, indReg)
    Album.artista → Artista.nomeArte
    Album.genere → Genere.tipo
    Album.(cittàReg, indReg) → StudioRegistrazione.(città, indirizzo)
Canzone (id, titolo, durata, numAscolti, dataUscita, artista, genere, album*, cittàReg, indReg)
    Canzone.artista → Artista.nomeArte
    Canzone.genere → Genere.tipo
    Canzone.album → Album.id
    Canzone.(cittàReg, indReg) → StudioRegistrazione.(città, indirizzo)
Certificazione (id, data, tipologia)
PremioCanzone (canzone, certificazione)
    PremioCanzone.canzone → Canzone.id
    PremioCanzone.certificazione → Certificazione.id
PremioAlbum (album, certificazione)
    PremioAlbum.album → Album.id
    PremioAlbum.certificazione → Certificazione.id

5. Implementazione dello schema logico

Per l'implementazione del database sono allegati i file CreazioneTabelle.sql con i comandi per la creazione delle tabelle e PopolamentoTabelle.sql per l'inserimento dei dati.

6. Definizione delle query e degli indici associati

Query 1

Query per ottenere un report con le informazioni principali di ogni artista che ha ricevuto almeno una certificazione (per album o per canzone): nome d'arte, il numero di canzoni e album totali che ha prodotto nella sua carriera con i rispettivi ascolti, numero di certificazioni per album e canzoni ottenute.

```
SELECT a.nomeArte AS Artista, COUNT(DISTINCT c.id) AS NumeroCanzoni,
    COUNT(DISTINCT al.id) AS NumeroAlbum,
    SUM(c.numAscolti) AS TotaleAscoltiCanzoni,
    SUM(al.numAscolti) AS TotaleAscoltiAlbum,
    COUNT(DISTINCT pc.certificazione) AS NumeroCertificazioniCanzoni,
    COUNT(DISTINCT pa.certificazione) AS NumeroCertificazioniAlbum
FROM Artista a LEFT JOIN Canzone c ON a.nomeArte = c.artista
    LEFT JOIN Album al ON a.nomeArte = al.artista
    LEFT JOIN PremioCanzone pc ON c.id = pc.canzone
    LEFT JOIN PremioAlbum pa ON al.id = pa.album
GROUP BY a.nomeArte
HAVING COUNT(DISTINCT pc.certificazione) > 0 OR COUNT(DISTINCT pa.certificazione) > 0
ORDER BY a.nomeArte;
```

	artista character varying (20)	numerocanzoni bigint	numeroalbum bigint	totaleascolticanzoni bigint	totaleascoltialbum bigint	numerocertificazionicanzoni bigint	numerocertificazionialbum bigint
1	Cinematic	4	1	15000	52000	1	0
2	Clabburn	8	2	60000	288000	1	1
3	Defranci	5	1	19000	95000	0	1
4	Geak	4	1	14000	44000	1	0
5	Haddya	2	0	11000	[null]	1	0
6	Hesbrook	4	1	18000	68000	1	1
7	IDiable	5	1	17000	65000	1	1
8	Sdring	1	0	2000	[null]	1	0
9	Simon	4	1	18000	100000	1	0
10	Slop	6	1	23000	72000	0	1
11	Terlej	4	1	22000	60000	1	0
12	TuttiMatti	4	1	13000	52000	1	0

Query 2

Query per ottenere la somma degli strumentisti che suonano i diversi strumenti.
Ciò che si intende ottenere da tale query, quindi, è il numero di artisti che suonano ogni strumento.

```
SELECT st.nome AS Strumento, COUNT(u.strumentista) AS NumeroStrumentisti
FROM Utilizzo u
JOIN Strumento st ON u.strumento = st.nome
GROUP BY st.nome;
```

	strumento character varying (20)	numerostrumentisti bigint
1	Basso	5
2	Pianoforte	3
3	Batteria	2
4	Violino	2
5	Sassofono	4
6	Chitarra	7

Query 3

Query per ottenere la somma degli album e delle canzoni che sono state incise nei tre studi di registrazione della DB-records.

```
SELECT st.città, st.indirizzo,
    COUNT(DISTINCT al.id) AS NumeroAlbum,
    COUNT(DISTINCT c.id) AS NumeroCanzoni
FROM StudioRegistrazione st
    JOIN Album al ON al.cittàReg = st.città AND al.indReg = st.indirizzo
    JOIN Canzone c ON c.cittàReg = st.città AND c.indReg = st.indirizzo
GROUP BY st.città, st.indirizzo;
```

	città [PK] character varying (20)	indirizzo [PK] character varying (50)	numeroalbum bigint	numerocanzoni bigint
1	Firenze	Via Tornabuoni 15	10	47
2	Milano	Via Dante Alighieri 10	11	53
3	Roma	Piazza del Popolo 3	9	45

Query 4

Query che elenca i vari artisti NON BAND che hanno fatto uno o più concerti nei palazzetti ma mai negli stadi.

```
SELECT a.nomeArte AS NomeArte, c.nome AS NomeReale, c.cognome AS CognomeReale
FROM Artista a JOIN Cliente c ON a.nomeArte = c.artista
LEFT JOIN Band b ON a.nomeArte = b.artista
WHERE b.artista IS NULL AND a.nomeArte IN (
    SELECT DISTINCT CO.artista
    FROM Concerto co
    WHERE co.luogo = 'palazzetto'
)
AND a.nomeArte NOT IN (
    SELECT DISTINCT co.artista
    FROM Concerto co
    WHERE co.luogo = 'stadio'
);
```

	nomearte character varying (20)	nomereale character varying (20)	cognomereale character varying (20)
1	Carley	Giacomo	Bianchi
2	Slop	Franco	Battista
3	Simon	Davide	Colombo
4	Frankis	Luca	Nardi
5	Kippen	Milena	Guerra
6	Hesbrook	Claudio	Ralli
7	Harvett	Daniele	Scotti
8	Woodfor	Francesco	Rossi
9	Terlej	Lorenzo	Marini
10	Malans	Marina	Rabbia
11	Loneio	Luca	Gennari
12	Cinematic	Miriam	Barbieri
13	IDiable	Marcella	Petti

Query 5

Query per ottenere il nome d'arte degli artisti che hanno stipendio massimo per ogni tipo di genere musicale. Per ottenere ciò creiamo prima una view che ci visualizzi lo stipendio massimo per ogni genere, andiamo poi ad interrogarla per ottenere tutti gli artisti che hanno tale stipendio.

```
CREATE VIEW StipendioMaxPerGenere AS
SELECT s.genere, MAX(c.remunerazione) AS stipendiomax
FROM Stile s
    JOIN Artista a ON s.artista = a.nomeArte
    JOIN Cliente cl ON a.nomeArte = cl.artista
    JOIN Contratto c ON cl.CF = c.cliente
GROUP BY s.genere;
```

	genere character varying (20)	stipendio_massimo integer
1	Pop	3200
2	Rock	3100
3	Rap	3200
4	Jazz	3100
5	Country	3100

```
SELECT s.genere, a.nomeArte, v.stipendiomax
FROM Stile s
    JOIN Artista a ON s.artista = a.nomeArte
    JOIN Cliente cl ON a.nomeArte = cl.artista
    JOIN Contratto c ON cl.CF = c.cliente
    JOIN StipendioMaxPerGenere v ON s.genere = v.genere AND c.remunerazione = v.stipendiomax;
```

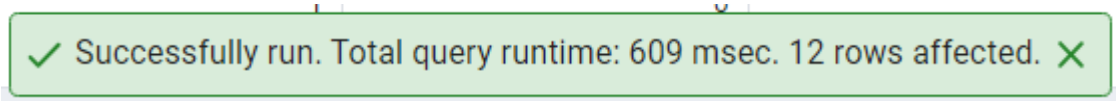
	genere character varying (20)	nomearte character varying (20)	stipendiomax integer
1	Rap	Slop	3200
2	Rap	Kippen	3200
3	Pop	Kippen	3200
4	Country	Hesbrook	3100
5	Rock	Liffin	3100
6	Jazz	Carme	3100

Indice per la query numero 3

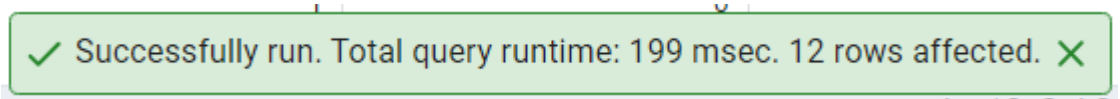
Indice creato sulla query 3 dato che la tabella "Canzone" ha un numero di istanze elevato.

```
CREATE INDEX idx_canzone_artista_nomearte_numAscolti ON Canzone (artista,numAscolti);
```

Query 1 prima della creazione dell'indice:



Query 1 dopo la creazione dell'indice:



Come si può notare dal miglioramento delle prestazione nell'esecuzione della query 1, l'indice idx_canzone_artista_nomearte_numAscolti risulta essere efficiente per lo scenario di applicazione della nostra query, in quanto:

- nel join tra Artista e Canzone su attributo artista, l'indice ci permette di trovare rapidamente tutte le canzoni di un determinato artista, riducendo il numero di ricerche necessarie
- dato che i risultati sono raggruppati per a.nomeArte, il nostro indice garantisce un raggruppamento più rapido in quanto i dati sono già ordinati per artista
- dopo che vengono individuati tutti i record per un dato artista, il database riesce a sommare facilmente tutti i numAscolti senza dover riorganizzare i dati

Se invece avessimo creato un indice su Canzone (numAscolti, artista) sarebbe stato poco efficiente dato che:

- il database dovrebbe cercare le corrispondenze per artista in un indice basato su ascolti; i dati quindi non sarebbero organizzati per artista e sarebbe necessario effettuare più confronti e scansioni
- anche durante il processo di aggregazione il database dovrebbe riorganizzare i dati in quanto l'indice è ordinato per numAscolti

7. Discussione dell'applicazione software che accede al DB per effettuare le query del punto 6

Il codice C incluso nel progetto si compone di un file codePro.c: l'applicazione software ricavata da tale file è stata ottenuta con il comando *gcc codePro.c -L dependencies\lib -lpq -o codice* all'interno della directory in cui risiede il file.
Prima di avviare l'esecuzione occorre definire le macro necessarie, specificando i propri dati di accesso, per una corretta connessione al database (PG_HOST, PG_USER, PG_PASSWORD, PG_PORT, PG_DB).

Le funzioni utilizzate sono:
checkResults(): verifica lo stato dei risultati ottenuti dalle cinque query, se una query non ha prodotto risultati consistenti, la funzione stampa un messaggio di errore e termina il programma.
printFormattedTable(): stampa formattando correttamente e ordinatamente le cinque query: trova la massima larghezza di ogni colonna per garantire un allineamento corretto.
executeAndPrintQuery(): esegue una query sul database, verifica i risultati con checkResults() e stampa la tabella formattata con printFormattedTable().
main(): la funzione main inizia costruendo la stringa di connessione con sprintf e stabilisce la connessione con il database, verificando la sua corretta esecuzione; se la connessione fallisce, viene stampato un messaggio di errore e il programma termina. Se la connessione è andata a buon fine, viene presentato un menù interattivo che permette all'utente di selezionare una delle 5 query implementate digitando il numero della query che si vuole visualizzare. Per terminare il programma è necessario digitare "6", in tal caso la connessione al database viene chiusa.