GROOVECLAM

Andrea Giacomo Baldan 579117

May 16, 2015

Contents

Abstract

A seguito degli eventi riguardanti il caso 'Napster' nei primi anni 2000, l'industria musicale e la distribuzione del materiale digitale ha subito notevoli cambiamenti e negli anni successivi prese piede il fenomeno del P2P (scambio tra utenti di files musicali, e non solo, mediante la rete) avviato da 'Napster', seguito da piattaforme e siti che offrono un servizio di streaming di file audio/video nel (quasi) totale rispetto dei diritti sugli album pubblicati. Grooveclam è una piattaforma online sulla linea del recente defunto Grooveshark, un sito di streaming audio appunto, e si propone di offrire un servizio di condivisione musicale tra utenti, permettendo di selezionare brani MP3 per l'ascolto, organizzarli in playlist che possono essere condivise tra

utenti connessi tra di loro o semplici code di riproduzione anonime, offrendo la possibilità di generare e popolare la propria libreria personale di brani e di contribuire al popolamento della base di dati su cui poggia la piattaforma aggiungendo le proprie canzoni e rendendole disponibili per l'ascolto a tutti gli utenti.

1 Analisi Dei Requisiti

Si vuole realizzare una base di dati per la gestione di una libreria musicale condivisa e la relativa interfaccia web che permetta interazione tra gli utenti.
Il cuore della libreria è formato da un insieme di album, suddivisi a loro volta in brani musicali; ogni album è identificato da un codice, ed è formato da alcuni metadati(titolo, autore, anno di pubblicazione) ed è specificato se si tratta di un album registrato in studio o una versione live, in quest'ultimo caso è possibile specificare la città in cui si è svolto il concerto, possiede inoltre informazioni opzionali di carattere generale (critiche ricevute, recensioni o breve storia sulla realizzazione dell'album). Gli album possono avere una copertina, a cui fanno riferimento anche tutti i brani che contengono.
Ogni brano musicale contenuto nell'album è identificato da un codice, ed è formato da alcuni metadati quali titolo, genere, durata. Esistono due tipi di utenti che possono accedere alla libreria, ordinari e amministratori, di entrambi interessano l'indirizzo e-mail, uno username e una password, sono opzionali i dati anagrafici quali nome e cognome. Gli utenti odinari possono seguire altri utenti ordinari, eccetto se stessi, inoltre ogni utente ha la possibilità di creare una propria collezione di brani preferiti selezionandoli dalla libreria, creare una coda di riproduzione anonima, o creare delle playlist delle quali interessa sapere il nome, queste ultime possono inoltre essere condivise con altri utenti "seguiti ". All'interno della collezione i brani non possono ripetersi mentre nelle code di riproduzione o nelle playlist uno stesso brano può comparire più volte. All'atto di registrazione un utente può decidere se attivare un abbonamento free o utilizzare un piano premium.

2 Progettazione concettuale

2.1 Classi

- **User:** Rappresenta un utente del servizio.
  - IdUser: *Int* «PK»
  - Name: *String*
  - Surname: *String*
  - Email: *String*

  Sono definite le seguenti sottoclassi disgiunte:
  - **Administrator**: Rappresenta un utente con privilegi amministrativi.
  - **Ordinario**: Rappresenta un utente ordinario.
- **Login:** Rappresenta delle credenziali d'accesso per un utente.
  - Username: *String* «PK»
  - Password: *String*

- **Subscription:** Modella un piano di iscrizione.
  - Type: *Enum* ['Free', 'Premium']
- **Song:** Rappresenta un brano.
  - IdSong: *Int* «PK»
  - Title: *String*
  - Genre: *String*
  - Duration: *Float*
- **Album:** Modella un album di brani.
  - IdAlbum: *Int* «PK»=
  - Title: *String*
  - Author: *String*
  - Info: *String*
  - Year: *Date*

  Sono definite le seguenti sottoclassi disgiunte con vincolo di partizionamento.
  - **Live:** Rappresenta un album registrato durante una performance live.
    - Location: *String*
  - **Studio:** Rappresenta un album registrato in studio.
- **Cover:** Rappresenta una generica cover di album.
  - IdImage: *Int* «PK»
  - Path: *String*
- **Playlist:** Modella una playlist.
  - Name: *String*
- **Collection:** Rappresenta una collezione di brani preferiti dall'utente.
  - IdCollection: *Int* «PK»

2.2 Associazioni

- **User-Collection:** "Crea"
  - Ogni utente può creare zero o una collezione, ogni collezione può essere creata da un solo utente.
  - Molteplicità $1:1$
  - Parziale verso **User**, totale verso **Collection**.

- **User-Song:** "Ascolta"
  - Ogni utente può ascoltare zero o più brani, ogni brano può essere ascoltato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **User**, parziale verso **Song**.
  - Attributi:
    * Timestamp: *Timestamp*
- **User-Song:** "Accoda"
  - Ogni utente può accodare zero o più brani, ogni brano può essere accodato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **User**, parziale verso **Song**.
  - Attributi:
    * Timestamp: *Timestamp*
- **User-User:** "Segue"
  - Ogni utente può seguire zero o più utenti, ogni utente può essere seguito da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso entrambi.
- **User-Playlist**: "Crea"
  - Ogni utente può creare zero o più playlist, ogni playlist può essere creata da un solo utente.
  - Molteplicità N : 1
  - Parziale verso **User**, totale verso **Playlist**.
- **User-Subscription:** "Iscritto"
  - Ogni utente può avere una sola iscrizione, ogni iscrizione può essere associata ad un solo utente.
  - Molteplicità 1 : 1
  - Totale verso **User** e verso **Subscription**.
- **Playlist-Song:** "PopolataDa"
  - Ogni playlist è popolata da zero o più brani, ogni brano popola zero o più playlist.
  - Molteplicità N : N
  - Parziale verso **Playlist**, parziale verso **Song**.
- **Song-Album:** "AppartieneA"
  - Ogni brano appartiene a zero o un brano, ogni brano contiene uno o più brani.

  - Molteplicità 1 : N
  - Parziale verso **Song**, totale verso **Album**.

- **Album-Cover:** "Possiede"
  - Ogni album possiede zero o una cover, ogni cover è posseduta da un solo album.
  - Molteplicità 1 : 1
  - Parziale verso **Album**, totale verso **Cover**.

- **Song-Cover:** "Possiede"
  - Ogni brano possiede una cover, ogni cover è posseduta da una o più canzoni.
  - Molteplicità 1 : N
  - Totale verso **Song**, totale verso **Cover**.

3 Progettazione Logica

3.1 Gerarchie

Tutte le gerarchie presenti nella progettazione concettuale sono state risolte mediante accorpamento in tabella unica, questo perchè nessuna di esse possedeva sottoclassi con un numero significativo di attributi o associazioni entranti da giustificarne un partizionamento di qualche genere.

3.2 Chiavi Primarie

Sono state create alcune chiavi primarie per identificare le istanze di alcune tabelle, quali *IdPlaylist* a **Playlist**.

3.3 Associazioni

- **User-Collection:** "Crea"
  - Ogni utente può creare zero o una collezione, ogni collezione può essere creata da un solo utente.
  - Molteplicità 1 : 1
  - Parziale verso **User**, totale verso **Collection**.
  - Chiave esterna non-nulla in **Collection** verso **User**.

- **User-Song:** "Ascolta"
  - Ogni utente può ascoltare zero o più brani, ogni brano può essere ascoltato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **User**, parziale verso **Song**.
  - Attributi:
    * Timestamp: *Timestamp*

  - Nuova tabella **Heard**, attributi:
    * IdUser: *Int* «PK» «FK(User)»
    * IdSong: *Int* «PK» «FK(Song)»
    * Timestamp: *Timestamp* «PK»

- **User-Song:** "Accoda"
  - Ogni utente può accodare zero o più brani, ogni brano può essere accodato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **User**, parziale verso **Song**.
  - Attributi:
    * Timestamp: *Timestamp*
  - Nuova tabella **Queue**, attributi:
    * IdUser: *Int* «PK» «FK(User)»
    * IdSong: *Int* «PK» «FK(Song)»
    * Timestamp: *Timestamp* «PK»

- **User-User:** "Segue"
  - Ogni utente può seguire zero o più utenti, ogni utente può essere seguito da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso entrambi.
  - Nuova tabella **Follow**, attributi:
    * IdUser: *Int* «PK» «FK(User)»
    * IdFellow: *Int* «PK» «FK(User)»

- **User-Playlist**: "Crea"
  - Ogni utente può creare zero o più playlist, ogni playlist può essere creata da un solo utente.
  - Molteplicità N : 1
  - Parziale verso **User**, totale verso **Playlist**.
  - Chiave esterna non-nulla in **Playlist** verso **User**.

- **User-Subscription:** "Iscritto"
  - Ogni utente può avere una sola iscrizione, ogni iscrizione può essere associata ad un solo utente.
  - Molteplicità 1 : 1
  - Totale verso **User** e verso **Subscription**.
  - Chiave esterna non-nulla in **Subscription** verso **User**.

- **Playlist-Song:** "PopolataDa"

  - Ogni playlist è popolata da zero o più brani, ogni brano popola zero o più playlist.
  - Molteplicità N : N
  - Parziale verso **Playlist**, parziale verso **Song**.
  - Nuova tabella **PlaylistSong**, attributi:
    * IdPlaylist: *Int* «PK» «FK(Playlist)»
    * IdSong: *Int* «PK» «FK(Song)»

- **Song-Album:** "AppartieneA"
  - Ogni brano appartiene a zero o un brano, ogni brano contiene uno o più brani.
  - Molteplicità 1 : N
  - Parziale verso **Song**, totale verso **Album**.
  - Chiave esterna non-nulla in **Song** verso **Album**.

- **Album-Cover:** "Possiede"
  - Ogni album possiede zero o una cover, ogni cover è posseduta da un solo album.
  - Molteplicità 1 : 1
  - Parziale verso **Album**, totale verso **Cover**.
  - Chiave esterna non-nulla in **Cover** verso **Album**.

- **Song-Cover:** "Possiede"
  - Ogni brano possiede una cover, ogni cover è posseduta da una o più canzoni.
  - Molteplicità 1 : N
  - Totale verso **Song**, totale verso **Cover**.
  - Chiave esterna non-nulla in **Song** verso **Cover**.

4 Implementazione Fisica

Query di implementazione DDL SQL della base di dati. Sorgente in *genera.sql*, popolamento in *popola.sql*. E' stata implementata una tabella **Errori**, riempita mediante procedura a sua volta richiamata dai trigger che ne fanno uso, contiene i messaggi d'errore rilevati. *funproc.sql* contiene invece le funzioni, i trigger e le procedure implementate.

```
1  SET FOREIGN_KEY_CHECKS = 0;
2  -- Table Album
3  CREATE TABLE IF NOT EXISTS `Album` (
4      `IdAlbum` INT(11) NOT NULL AUTO_INCREMENT,
5      `Title` VARCHAR(140) NOT NULL,
6      `Author` VARCHAR(140) NOT NULL,
7      `Info` VARCHAR(300) DEFAULT NULL,
8      `Year` DATE NOT NULL,
9      `Live` BOOLEAN NOT NULL,
```

```
10      `Location` VARCHAR(40) DEFAULT NULL,
11      PRIMARY KEY(`IdAlbum`)
12  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
13  -- Table Song
14  CREATE TABLE IF NOT EXISTS `Song` (
15      `IdSong` INT(11) NOT NULL AUTO_INCREMENT,
16      `IdAlbum` INT(11) NOT NULL,
17      `Title` VARCHAR(140) NOT NULL,
18      `Genre` VARCHAR(40) NOT NULL,
19      `Duration` FLOAT,
20      `IdImage` INT(11) NOT NULL,
21      PRIMARY KEY(`IdSong`),
22      FOREIGN KEY(`IdAlbum`) REFERENCES Album(`IdAlbum`) ON DELETE CASCADE ON UPDATE CASCADE,
23      FOREIGN KEY(`IdImage`) REFERENCES Cover(`IdImage`) ON DELETE CASCADE ON UPDATE CASCADE
24  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
25  -- Table Cover
26  CREATE TABLE IF NOT EXISTS `Cover` (
27      `IdImage` INT(11) NOT NULL AUTO_INCREMENT,
28      `IdAlbum` INT(11) NOT NULL,
29      `Path` VARCHAR (40) NOT NULL DEFAULT "img/covers/nocover.jpg",
30      PRIMARY KEY(`IdImage`),
31      FOREIGN KEY(`IdAlbum`) REFERENCES Album(`IdAlbum`) ON DELETE CASCADE ON UPDATE CASCADE
32  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
33  -- Table User
34  CREATE TABLE IF NOT EXISTS `User` (
35      `IdUser` INT(11) NOT NULL AUTO_INCREMENT,
36      `Name` VARCHAR(40) DEFAULT NULL,
37      `Surname` VARCHAR(40) DEFAULT NULL,
38      `Email` VARCHAR(40) NOT NULL,
39      `Administrator` BOOLEAN NOT NULL,
40      `Username` VARCHAR(40) NOT NULL,
41      `Password` VARCHAR(40) NOT NULL,
42      PRIMARY KEY(`IdUser`),
43      UNIQUE(`Username`)
44  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
45  -- Table Follow
46  CREATE TABLE IF NOT EXISTS `Follow` (
47      `IdUser` INT(11) NOT NULL,
48      `IdFellow` INT(11) NOT NULL,
49      CONSTRAINT PRIMARY KEY pk(`IdUser`, `IdFellow`),
50      FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
51      FOREIGN KEY(`IdFellow`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
52      CHECK(`IdUser` != `IdFellow`)
53  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
54  -- Table SharedPlaylist
55  -- CREATE TABLE IF NOT EXISTS `SharedPlaylist` (
56  --  `IdUser` INT(11) NOT NULL,
57  --  `IdFellow` INT(11) NOT NULL,
58  --  `IdPlaylist` INT(11) NOT NULL,
59  --  CONSTRAINT PRIMARY KEY pk(`IdUser`, `IdFellow`, `IdPlaylist`),
60  --  FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
61  --  FOREIGN KEY(`IdFellow`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
62  --  FOREIGN KEY(`IdPlaylist`) REFERENCES Playlist(`IdPlaylist`) ON DELETE CASCADE ON UPDATE CASCADE,
63  --  CHECK(`IdUser` != `IdFellow`)
64  -- ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
65  -- Table Subscription
66  CREATE TABLE IF NOT EXISTS `Subscription` (
67      `IdUser` INT(10) NOT NULL,
68      `Type` ENUM('Free', 'Premium') NOT NULL,
69      PRIMARY KEY(`IdUser`),
70      FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE
71  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
72  -- Table Collection
73  CREATE TABLE IF NOT EXISTS `Collection` (
74      `IdCollection` INT(11) NOT NULL AUTO_INCREMENT,
75      `IdUser` INT(11) NOT NULL,
76      PRIMARY KEY(`IdCollection`),
77      FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE
78  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
79  -- Table SongCollection
80  CREATE TABLE IF NOT EXISTS `SongCollection` (
81      `IdSong` INT(11) NOT NULL,
82      `IdCollection` INT(11) NOT NULL,
83      CONSTRAINT PRIMARY KEY pk(`IdCollection`, `IdSong`),
84      FOREIGN KEY(`IdSong`) REFERENCES Song(`IdSong`) ON DELETE CASCADE ON UPDATE CASCADE,
85      FOREIGN KEY(`IdCollection`) REFERENCES Collection(`IdCollection`) ON DELETE CASCADE ON UPDATE
            CASCADE
86  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
87  -- Table Playlist
88  CREATE TABLE IF NOT EXISTS `Playlist` (
89      `IdPlaylist` INT(11) NOT NULL AUTO_INCREMENT,
90      `IdUser` INT(11) NOT NULL,
91      `Name` VARCHAR(40) NOT NULL,
92      `Private` BOOLEAN DEFAULT FALSE,
93      PRIMARY KEY(`IdPlaylist`),
94      FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE
95  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
96  -- Table PlaylistSong
97  CREATE TABLE IF NOT EXISTS `PlaylistSong` (
98      `IdPlaylist` INT(11) NOT NULL,
99      `IdSong` INT(11) NOT NULL,
100     CONSTRAINT PRIMARY KEY pk(`IdPlaylist`, `IdSong`),
101     FOREIGN KEY(`IdPlaylist`) REFERENCES Playlist(`IdPlaylist`) ON DELETE CASCADE ON UPDATE CASCADE,
102     FOREIGN KEY(`IdSong`) REFERENCES Song(`IdSong`) ON DELETE CASCADE ON UPDATE CASCADE
103 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
104 -- Table Queue
105 -- CREATE TABLE IF NOT EXISTS `Queue` (
106 --  `IdUser` INT(11) NOT NULL,
107 --  FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE
108 -- ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
109 -- Table Queue
110 CREATE TABLE IF NOT EXISTS `Queue` (
111     `IdUser` INT(11) NOT NULL,
112     `IdSong` INT(11) NOT NULL,
113     `TimeStamp` TIMESTAMP NOT NULL,
114     CONSTRAINT PRIMARY KEY pk(`IdUser`, `IdSong`, `TimeStamp`),
115     FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
116     FOREIGN KEY(`IdSong`) REFERENCES Song(`IdSong`) ON DELETE CASCADE ON UPDATE CASCADE
117 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
118 -- Table Heard
119 CREATE TABLE IF NOT EXISTS `Heard` (
120     `IdUser` INT(11) NOT NULL,
121     `IdSong` INT(11) NOT NULL,
122     `TimeStamp` TIMESTAMP NOT NULL,
123     CONSTRAINT PRIMARY KEY pk(`IdUser`, `IdSong`, `TimeStamp`),
124     FOREIGN KEY(`IdUser`) REFERENCES User(`IdUser`) ON DELETE CASCADE ON UPDATE CASCADE,
125     FOREIGN KEY(`IdSong`) REFERENCES Song(`IdSong`) ON DELETE CASCADE ON UPDATE CASCADE
126 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.1 Trigger

4.2 Funzioni e Procedure

5 Query

Alcune query significative.

1. Titolo, album e username dell'utente, degli ultimi 10 brani ascoltati tra i followers.

```
1 SELECT s.Title, a.Title as AlbumTitle, u.Username, h.Timestamp
2 FROM Song s INNER JOIN Album a ON(s.IdAlbum = a.IdAlbum)
3             INNER JOIN Heard h ON(h.IdSong = s.IdSong)
4             INNER JOIN Follow f ON(f.IdFellow = h.IdUser)
5             INNER JOIN User u ON(u.IdUser = f.IdFellow)
6 WHERE h.Timestamp BETWEEN ADDDATE(CURDATE(), -7) AND CURDATE()
7     AND u.IdUser IN (SELECT u.IdUser FROM User u INNER JOIN Follow f ON(f.IdFellow = u.IdUser) WHERE f.
          IdUser = 1)
8     ORDER BY h.Timestamp DESC LIMIT 10;
```

6 Interfaccia Web

Per l'interfaccia web è stato seguito un pattern MVC molto rudimentale, che tuttavia ha permesso di semplificarne la realizzazione modularizzando le operazione da effettuare sulla base di dati mediante le pagine.

6.1 Organizzazione e Struttura Generale

La struttura generale dell'interfaccia consiste di 3 cartelle principali e 2 pagine di servizio contenenti rispettivamente un singleton dedicato esclusivamente alla connessione alla base di dati e un singleton dedicato alla creazione e manipolazione delle sessioni. Le cartelle /models, /views, /controllers seguono le tipiche linee guida del pattern MVC, all'interno di /models troviamo infatti i modelli, oggetti atti ad interfacciarsi con la base di dati ed eseguire le query richieste dalle pagine (routes) contenute nei controllers, infine le view, pagine "di template" contenenti per lo più codice HTML e brevi tratti di PHP, vengono popolate mediante le chiamate ai controllers. La navigazione vera e propria tra le pagine avviene mediante parametri GET che si occupano di selezionare il controller richiesto e l'azione da eseguire (funzioni all'interno del controller richiesto).

6.1.1 Esempi

- Richiedere la pagina albums:

`/basidati/~abaldan/?controller=albums&action=index`

- Visualizzazione brano con id = 4:

`/basidati/~abaldan/?controller=songs&action=show&id=4`

6.2 Pagine Principali

Ci sono 6 pagine principali che consentono la navigazione all'interno dell'interfaccia, accedibili mediante un menù laterale a sinistra. **Home** contiene alcune statistiche sullo stato della BD, ad esempio i brani ascoltati recentemente dai propri followers, questo solo dopo aver effettuato l'accesso con un proprio account registrato, altrimenti in home, come pure in ogni pagina che richiede di essere loggati, viene mostrato un form di login mediantei il quale è anche possibile registrare un account. **Songs** è la pagina adibita alla visualizzazione di tutte le canzoni contenute nella BD o, nel caso di account loggato, offre la possibilità di aggiungere i propri brani alla BD, aggiungerne alla propria collezione o alla coda di riproduzione; **albums** contiene tutti gli album presenti nella piattaforma, sempre previa autenticazione permette di inserirne di nuovi ed è possibile visualizzare i dettagli di ogni album e brano contenuto in esso. **Collection** e **playlist** sono rispettivamente le pagine di gestione della propria collezione brani e playlist, con la possibilità di privatizzare o rendere pubbliche le proprie playlist. **Queue** infine ospita la coda di riproduzione, ordinate in base ai timestamp di aggiunta. E' possibile modificare i dati relativi al proprio account, incluso il piano di'iscrizione, utilizzando la pagina accessibile clickando sul bottone in alto a sinistra **settings**, solo dopo aver loggato.

6.3 Note

Trattandosi di un interfaccia "simulativa", in quanto la principale materia d'interesse è la struttura della base di dati su cui poggia, la riproduzione effettiva dei brani non è stata implementata, e non esistono fisicamente file Mp3 caricati all'interno della base di dati, è stato tuttavia implementato un semplice e rudimentale riproduttore in poche righe di javascript atto a dare un'idea dell'effettivo utilizzo che una completa implementazione della piattaforma porterebbe ad avere. Non sono stati scritti controlli di alcun tipo sull'input da parte dell'utente.