Relazione Sviluppo Base di Dati

Università degli Studi dell'Insubria – Laurea Triennale in Informatica

Progetto Laboratorio B: Emotional Songs.

Sviluppato da:

-Ashley Chudory, matricola 746423

-Bogdan Tsitsiurskyi, matricola 748685

-Maria Vittoria Ratti, matricola 748825

-Miguel Alfredo Valerio Aquino, matricola 748704

Indice

Raccolta e analisi dei requisiti

Specifiche sui dati in linguaggio naturale

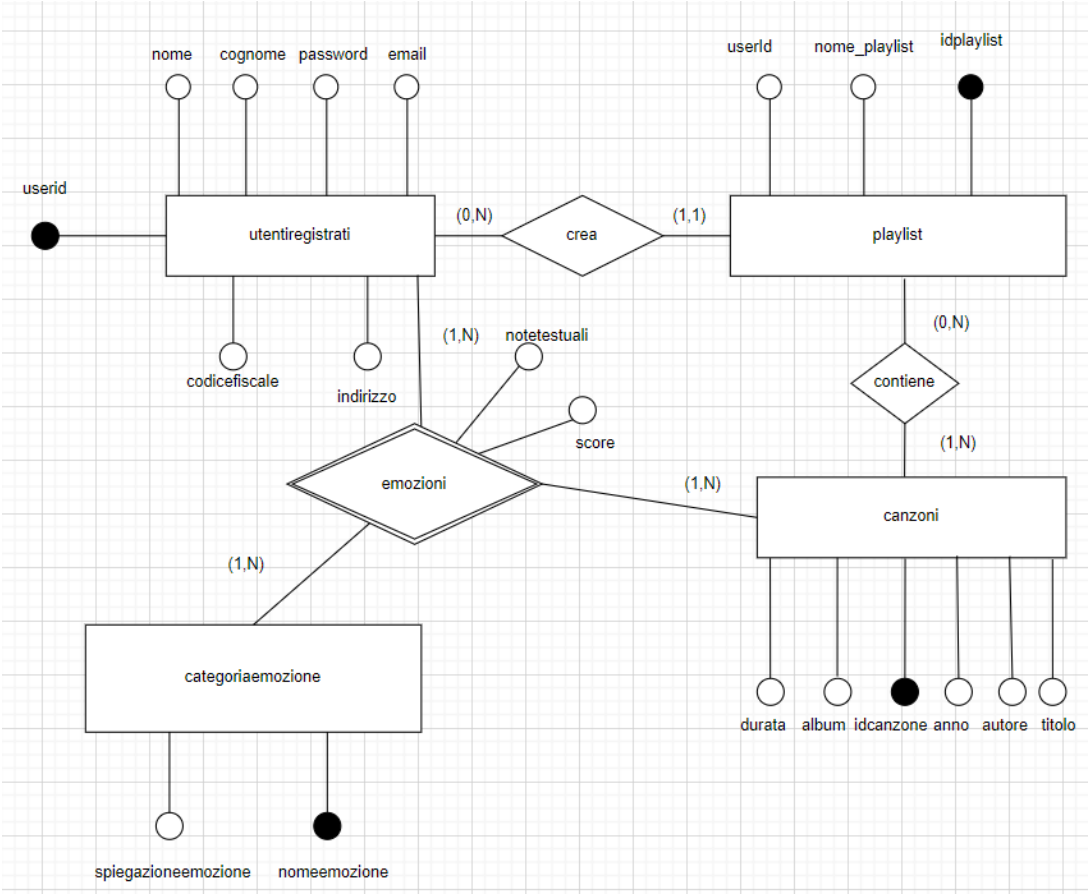
I dati riguardo all'utente che si vuole memorizzare nella base di dati sull'applicazione "Emotional Songs" sono:

- nome varchar
- cognome varchar
- codicefiscale varchar
- indirizzo varchar
- email varchar
- userId varchar, unique
- password varchar

La registrazione viene effettuata dopo che l'utente ha correttamente compilato i campi precedentemente descritti.

L'applicazione tiene traccia delle registrazioni dei profili degli utenti e in particolari memorizza tutti i dati del profilo utente.

Schema concettuale



Scelte progettuali

Il modello ER mostrato è stato sviluppato tenendo in considerazione le richieste del progetto. La nostra attenzione si è focalizzata sulla correttezza, completezza e minimalità dello schema. Infatti, il modello ha l'obiettivo di ridurre le informazioni ridondanti e i concetti superflui cercando di utilizzare in modo appropriato i costrutti messi a disposizione.

In particolare, l'entità "utentiregistrati" contiene i dati che gli utenti devono inserire per registrarsi all'applicazione. Tale entità è costituita da un insieme di proprietà elementari che l'utente deve inserire: *nome, cognome, indirizzo fisico, indirizzo di posta elettronica*, *userid* e *password* utile per accedere al sistema. Nel nostro diagramma tali attributi sono stati definiti come attributi semplici perché non sono ulteriormente scomponibili. Inoltre, ogni attributo è obbligatorio cioè non viene accettato che il dato sia null ovvero mancante, sconosciuto o inapplicabile. Un concetto fondamentale nella modellazione dei dati è sicuramente la scelta

dell'attributo identificatore che consente di determinare univocamente un'istanza di entità da un'altra. Nel caso dell'entità "utentiregistrati" l'attributo identificatore scelto è *userid*.
L'entità "playlist" consente all'utente di creare una playlist di brani musicali ed è costituita da tre attributi tra cui userid, nomeplaylist e idplaylist (identificatore). Le entità "utentiregistrati" e "playlist" sono collegate tramite un'associazione chiamata "crea" che costituisce un legame logico tra le due entità.

Ogni associazione tra due entità ha due versi, questo significa che tra "utentiregistrati" e "playlist" esiste un'associazione *crea,* mentre tra playlist e "utentiregistrati" esiste una correlazione *creataDa.* Inoltre, ogni associazione presenta una determinata cardinalità che esprime i vincoli di partecipazione: la cardinalità può essere minima o massima. Nell'applicazione, l'utente registrato può creare 0 playlist cioè la creazione di una playlist risulta opzionale oppure può produrre un numero illimitato di playlist. Diversamente, una playlist deve essere creata da un solo utente registrato.
Un'altra entità presente nel modello ER è "canzoni" che racchiude le informazioni riguardanti i brani musicali tra cui idcanzone (identificatore), durata, album,autore,anno, titolo. Nel nostro modello ER, l'entità canzone è associata alla playlist questo perché a una playlist è necessario aggiungere delle tracce musicali. In questo caso l'associazione che lega "playlist" a "canzoni" è *contiene*, mentre quella che lega "canzoni" a "playlist" è *contenutaDa.* La playlist può non avere canzoni: l'utente può creare una playlist e assegnarle un nome ma non è obbligato a inserire delle tracce musicali. Il limite massimo di inserimento dei brani è N cioè non è definito un termine massimo di canzoni da inserire.
Nel modello ER è presente anche la relazione "emozioni" che rappresenta una relazione identificativa che collega entità forti e entità deboli. Infatti, gli attributi primari delle altre tabelle vengono ereditati dalla tabella "emozioni" che include lo *userid* riferimento alla tabella "utentiregistrati", *idcanzoni* riferimento alla tabella "canzoni" e il *nomeEmozione* riferimento alla tabella "categoriaemozioni". Questo significa che la tabella "emozioni"non può essere identificata in modo univoco senza la presenza delle tabelle "utentiregistrati", "canzoni" e "categoriaemozioni". Inoltre, nella tabella "emozioni", oltre agli attributi precedentemente descritti, sono presenti anche gli attributi *score* e *notetestuali*.
Per ogni entità che partecipa al diagramma ER viene specificata la cardinalità di relazione. In particolare, la relazione che collega "utentiregirstati" a "emozioni" è di tipo uno a molti: un utente registrato può scegliere una molteplicità di emozioni ma l'emozione scelta è associata a un solo utente. In modo analogo, la relazione che lega "emozioni" a "canzoni" è di tipo uno a molti: a una canzone si possono associare più emozioni.

Un'altra entità presente nel modello ER è "categoriaemozioni" che presenta due attributi: *nomeemozione*(identificatore) e *spiegazioneemozione*. Nell'entità "categoriaemozione" sono memorizzate le 9 emozioni che l'utente può percepire durante l'ascolto di un brano musicale. La relazione che lega "categoriaemozioni" a "emozioni" è di tipo uno a molti.
In queste tabelle sono riportati i costrutti principali del modello ER:

Dizionario delle entità

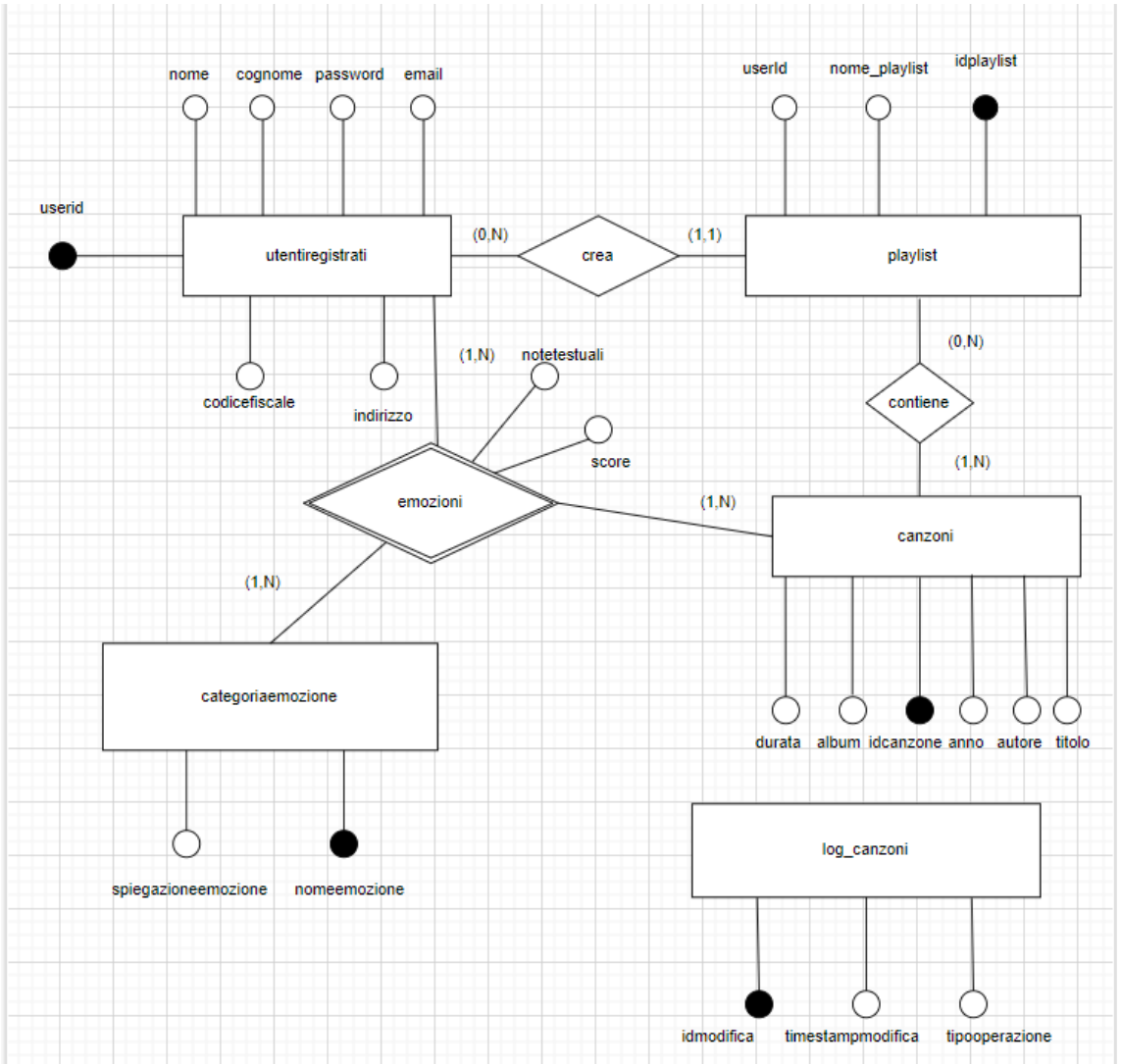
nome	descrizione	attributi	identificatori
utentiregistrati	contiene le informazioni di tutti gli utenti registrati	nome,cognome, password, email,indirizzo, codice fiscale	userid
Playlist	consente di creare delle playlist	nomeplaylist, userid	idPlaylist
canzoni	contiene l'elenco dei brani musicali	durata, album,autore,anno, titolo	idcanzone
categoriaemozione	contiene l'elenco delle 9 emozioni	spiegazione emozione	nomeemozione
emozioni	contiene le informazioni delle emozioni registrate dagli utenti	notetestuali, score	Userid^{utentiregistrati} idCanzone^{Canzoni} nomeEmozione^{CategoriaEmozione}

Dizionario delle associazioni

nome	descrizione	entità collegate
crea	consente all'utente di creare una playlist di brani musicali	utentiregistrati, playlist
contiene	ogni playlist contiene un determinato numero di canzoni scelte dall'utente	playlist, canzoni

Progettazione Logica

Schema Concettuale Ristrutturato



Scelte progettuali

Nello schema ER ristrutturato è stata aggiunta una nuova entità chiamata log_canzoni che serve per tracciare le operazioni di inserimento, aggiornamento ed eliminazioni compiute sulla tabella canzoni. Con l'aggiunta dell'entità log_canzoni è possibile una revisione più dettagliata delle modifiche al contenuto delle canzoni presenti nel nostro database.

In questa tabella sono riportati i costrutti principali del modello ER ristrutturato:

nome	descrizione	attributi	identificatori
log_canzoni	contiene le informazioni per tracciare le modifiche compiute su l'entità canzoni	timestampmodifica tipooperazione	idmodifica

Traduzione

utentiregistrati(

userid, varchar(20)

password, varchar(60)

nome, varchar(30)

cognome, varchar(30)

codiceFiscale, varchar(16)

indirizzo, varchar(100)

email, varchar(100)

)

Playlist(

idPlaylist, SERIAL

nome_playlist, varchar(256)

userid^{utentiregistrati}, varchar(20)

)

Canzoni(

idCanzone, SERIAL

titolo varchar(256),

autore varchar(256),

anno numeric(4),

album varchar(256)

durata numeric,(10)

)

Log_Canzoni (

idModifica, SERIAL

timestampModifica, TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP

tipoOperazione, VARCHAR(10)

)

Playlist_Canzoni(

idPlaylist[Playlist], integer

idCanzone[Canzoni], integer

)

CategoriaEmozione(

nomeEmozione, VARCHAR(30)

spiegazioneEmozione, VARCHAR(100) NOT NULL

)

Emozioni (

Userid[utentiregistrati], VARCHAR(20)

idCanzone[Canzoni], integer

nomeEmozione[CategoriaEmozione], VARCHAR(30)

score, numeric(1)

noteTestuali, VARCHAR(256)

)

Trigger per monitoraggio delle modifiche

Nell'applicazione il trigger denominato trigger_aggiorna_modifiche è stato sviluppato con lo scopo di ottimizzare l'efficienza del progetto client EmotionalSongs. In particolare, il trigger consente di gestire la sincronizzazione tra il progetto client e il database, rendendola efficiente. Nell'ambito del progetto client Emotional Songs, il trigger viene collegato alla tabella "canzoni" per evitare le richieste ridondanti al database. In particolare, tramite le operazioni INSERT, UPDATE, DELETE eseguite su "canzoni", il trigger registra un timestamp nella tabella "log_canzoni". L'idea è quella che grazie al timestamp, il progetto client può appurare se ci sono state modifiche alla tabella "canzoni" dopo l'ultima

sincronizzazione. Se il timestamp delle ultime modifiche risulta successivo all'ultima sincronizzazione del progetto client, allora bisogna richiedere la lista di canzoni per l'aggiornamento.

Pertanto, il trigger porta numerosi vantaggi, in particolare:

- notevole risparmio di risorse: riduce il carico sul database e la quantità di dati tra il database e il progetto client
- efficienza
- sincronizzazione automatica

Inoltre, maggiori dettagli sulla definizione del trigger e la query completa sono disponibili nella sezione "creazione tabelle". Questo è possibile perché la query del trigger si trova su "creazione tabelle"

Progettazione Pratica

Creazione DataBase

```
CREATE DATABASE emotionalsongs;
```

Creazione Tabelle

Utenti

```
CREATE TABLE utentiregistrati(
userid VARCHAR(20) NOT NULL,
password VARCHAR(60) NOT NULL,
nome VARCHAR(30) NOT NULL,
cognome VARCHAR(30) NOT NULL,
codiceFiscale VARCHAR(16) NOT NULL,
```

```
indirizzo VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
PRIMARY KEY (userid)
);
```

Playlist

```
CREATE TABLE Playlist(
 idPlaylist SERIAL,
 nome_playlist varchar(256) not null,
 userid varchar(20) references utentiregistrati(userid),
PRIMARY KEY (idPlaylist));
```

Canzoni

```
CREATE TABLE Canzoni(
idCanzone SERIAL,
titolo VARCHAR(256) NOT NULL,
autore VARCHAR(256) NOT NULL,
anno numeric (4) NOT NULL,
album VARCHAR(256),
durata numeric (10),
PRIMARY KEY (idCanzone)
);
```

Log_Canzoni per tenere traccia delle modifiche fatte sulla tabella Canzoni

```
CREATE TABLE Log_Canzoni (
   idModifica SERIAL PRIMARY KEY,
   timestampModifica TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP, -- Data e ora della modifica
   tipoOperazione VARCHAR(10) NOT NULL -- Tipo di operazione eseguita (INSERT,
UPDATE, DELETE)
);
```

Canzoni nella playlist

```
CREATE TABLE Playlist_Canzoni(
  idPlaylist integer NOT NULL,
  idCanzone integer NOT NULL,
   FOREIGN KEY (idPlaylist) REFERENCES Playlist(idPlaylist) ON DELETE CASCADE,
  FOREIGN KEY (idCanzone) REFERENCES Canzoni(idCanzone) ON DELETE
CASCADE,
PRIMARY KEY (idPlaylist,idCanzone));
```

Categoria delle emozioni

```
CREATE TABLE CategoriaEmozione(
  nomeEmozione VARCHAR(30),
  spiegazioneEmozione VARCHAR(100) NOT NULL,
  PRIMARY KEY (nomeEmozione)
);
```

Inserimento delle emozioni nella tabella “CategoriaEmozione”

```
INSERT INTO CategoriaEmozione (nomeEmozione, spiegazioneEmozione) VALUES
('Amazement', 'Feeling of wonder or happiness'),
('Solemnity', 'Feeling of transcendence, inspiration. Thrills.'),
('Tenderness', 'Sensuality, affect, feeling of love'),
('Nostalgia', 'Dreamy, melancholic, sentimental feelings'),
('Calmness', 'Relaxation, serenity, meditativeness'),
('Power', 'Feeling strong, heroic, triumphant, energetic'),
('Joy', 'Feels like dancing, bouncy feeling, animated, amused'),
('Tension', 'Feeling Nervous, impatient, irritated'),
('Sadness', 'Feeling Depressed, sorrowful');
```

Emozioni associate alle canzoni da parte degli utenti

```
CREATE TABLE Emozioni (
userid VARCHAR(20) NOT NULL,
idCanzone integer NOT NULL,
nomeEmozione VARCHAR(30) NOT NULL,
```

```
score numeric(1) NOT NULL,
noteTestuali VARCHAR(256),
PRIMARY KEY (userid,idCanzone,nomeEmozione),
FOREIGN KEY (userid) REFERENCES UtentiRegistrati(userid),
 FOREIGN KEY (idCanzone) REFERENCES Canzoni(idCanzone),
 FOREIGN KEY (nomeEmozione) REFERENCES CategoriaEmozione(nomeEmozione)
);
```

Funzione trigger (per registrare le operazioni di modifica eseguite sulla tabella Canzoni all'interno della tabella Log_Canzoni)

```
CREATE OR REPLACE FUNCTION aggiorna_modifiche()
RETURNS TRIGGER AS $$
BEGIN
   -- Inserimento nella tabella Log_Canzoni per INSERT
   IF TG_OP = 'INSERT' THEN
      INSERT INTO Log_Canzoni (tipoOperazione)
      VALUES ('INSERT');
   END IF;
   -- Inserimento nella tabella Log_Canzoni per UPDATE
   IF TG_OP = 'UPDATE' THEN
      INSERT INTO Log_Canzoni (tipoOperazione)
      VALUES ('UPDATE');
   END IF;
   -- Inserimento nella tabella Log_Canzoni per DELETE
   IF TG_OP = 'DELETE' THEN
      INSERT INTO Log_Canzoni (tipoOperazione)
      VALUES ('DELETE');
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Collegamento del nuovo trigger (alle operazioni di INSERT, UPDATE e DELETE sulla tabella Canzoni)

```
CREATE TRIGGER trigger_aggiorna_modifiche
AFTER INSERT OR UPDATE OR DELETE ON Canzoni
FOR EACH ROW
EXECUTE FUNCTION aggiorna_modifiche();
```

Operazioni di Insert

Aggiungi utente

```
INSERT INTO utentiregistrati VALUES(userid, password, nome, cognome, codiceFiscale, indirizzo, email);
```

Inserisci canzoni nella playlist

```
INSERT INTO playlist_canzoni  VALUES(idplaylist, idcanzone);
```

Crea playlist

```
INSERT INTO playlist VALUES(nome_playlist, userid);
```

Associa emozione

```
INSERT INTO emozioni VALUES(userid, idcanzone, nomeemozione, score, notetestuali);
```

Query

Login

```
SELECT userid, password
FROM utentiregistrati
WHERE userid = ?
```

Verifica se un userID è disponibile

```
SELECT COUNT(*)
FROM utentiregistrati
WHERE userid = ?
```

Recupera le informazioni di un utente

```
SELECT *
FROM utentiregistrati
WHERE userid = ?
```

Recupera l'elenco delle canzoni restituendo una mappa che associa l'ID della canzone

```
SELECT *
FROM canzoni
ORDER BY idCanzone ASC
```

Timestamp dell'ultima modifica effettuata nella tabella "Log_Canzoni"

```
SELECT MAX(timestampModifica)
FROM Log_Canzoni
```

Playlists associate ad uno specifico utente

```
SELECT *
FROM playlist
WHERE userid = ?
```

Recupera canzoni associate ad una playlist tramite il suo ID

```
SELECT c.*
FROM Canzoni c JOIN Playlist_Canzoni pc ON c.idCanzone = pc.idCanzone
WHERE pc.idPlaylist = ?
```

Elenco completo delle canzoni

```
SELECT *
FROM categoriaemozione
```

Numero di utenti che hanno lasciato almeno un'emozione alla canzone selezionata

```
SELECT COUNT(DISTINCT userid) as numUtenti
FROM emozioni
WHERE idcanzone =?
```

Elenco dei tag emozionali con il numero di utenti associati

```
SELECT nomeEmozione, COUNT(DISTINCT userid)
FROM Emozioni
WHERE idCanzone = ?
GROUP BY nomeEmozione
```

Elenco con il punteggio totale delle emozioni associate a una specifica canzone

```
SELECT nomeEmozione, score
FROM Emozioni
WHERE idCanzone = ?
```

Punteggio delle emozioni associate a una specifica canzone per un utente specifico

```
SELECT nomeEmozione, score
FROM Emozioni
WHERE idCanzone = ? AND userid = ?
```

Elenco dei tag emozionali con i rispettivi commenti per ciascuna emozione

```
SELECT u.nome, e.nomeEmozione, e.noteTestuali
FROM Emozioni e JOIN UtentiRegistrati u ON e.userid = u.userid
WHERE e.idCanzone = ?
```

Operazioni di Delete

Elimina le emozioni associate a una specifica canzone e utente

```
DELETE FROM Emozioni
WHERE userid = ? AND idCanzone = ?
```

Elimina una playlist in base all'ID specificato

```
DELETE FROM Playlist
WHERE idplaylist = ?
```

Rimuove una canzone da una playlist in base agli ID specificati

```
DELETE FROM Playlist_Canzoni
WHERE idplaylist = ? AND idcanzone = ?
```