UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione Corso di Laurea in Informatica

Insegnamento di Basi di Dati e Sistemi Informativi I Anno accademico 2021/2022

Progettazione e sviluppo di una base di dati relazionale per la gestione di una libreria musicale

Rescigno Riccardo N86002602 - Milano Raffaele N86004123
Docente: Silvio Barra

Indice

1 Descrizione del progetto **3**
1.1 Descrizione Sintetica . . . 3

2 Progettazione concettuale **3**
2.1 Class Diagram . . . 3
2.2 Ristrutturazione del Class Diagram . . . 4
2.2.1 Analisi delle ridondanze . . . 4
2.3 Dizionario dei dati . . . 5
2.3.1 Dizionario delle classi . . . 5
2.3.2 Dizionario delle associazioni . . . 6
2.3.3 Dizionario dei vincoli . . . 6

3 Progettazione logica **6**
3.0.1 Schema Logico . . . 6

4 Progettazione fisica **7**
4.1 Definizione delle variabili Enum . . . 7
4.2 Definizione della tabella Utente . . . 7
4.3 Definizione della tabella LogUtente . . . 7
4.4 Definizione della tabella Artista . . . 7
4.5 Definizione della tabella Album . . . 8
4.6 Definizione della tabella Traccia . . . 8
4.7 Definizione della funzione Fetch Like . . . 8
4.8 Definizione della funzione Fetch fascia oraria . . . 9

1 Descrizione del progetto

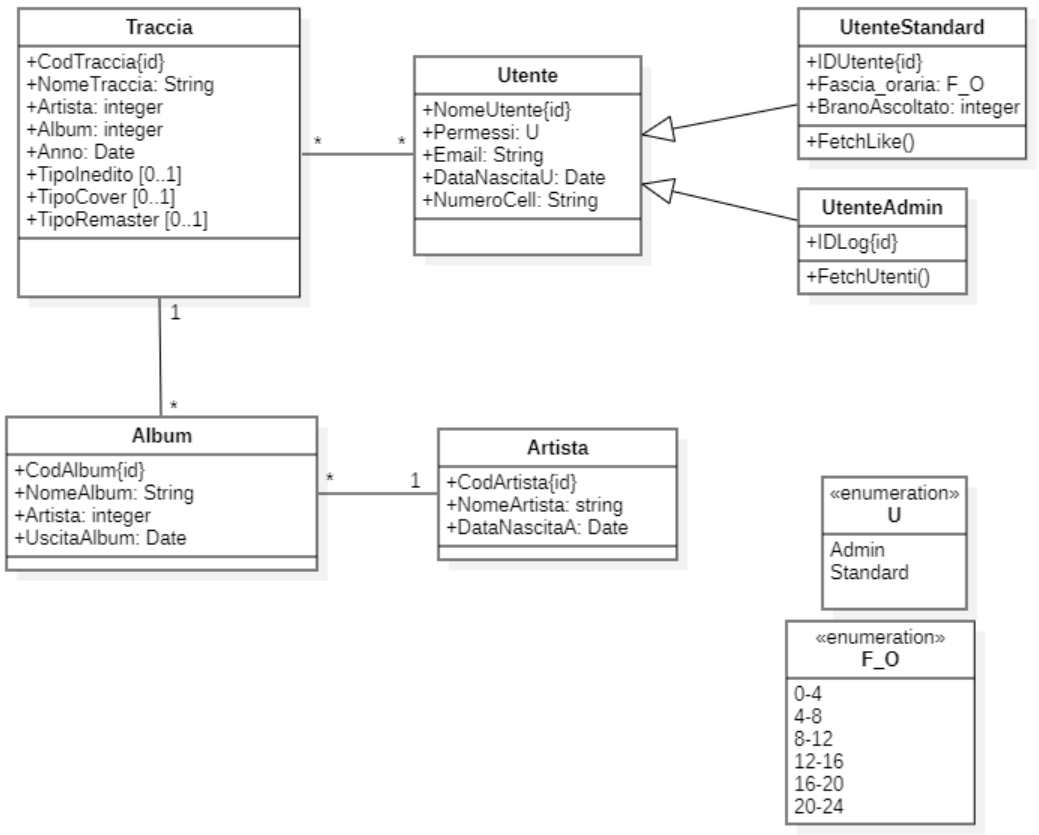
1.1 Descrizione Sintetica

Si progetterà ed implementerà una base di dati relazionale che possa essere d'ausilio alla gestione delle operazioni e memorizzazione di una libreria musicale. Capace di differenziare la tipologia di traccia tra inedito, remaster e cover e che permetta ad un admin di recuperare, data una certa traccia, un sottoinsieme degli utenti che hanno effettuato più ascolti di quella traccia. E' inoltre possibile andare a identificare la fascia oraria in cui un determinato utente ha effettuato pi'u ascolti (definendo un set di fasce orarie, per semplificare il compito).

2 Progettazione concettuale

In questo capitolo inizia la progettazione della base di dati al livello di astrazione più alto. Dal risultato dell'analisi dei requisiti che devono essere soddisfatti si arriverà ad uno schema concettuale indipendente dalla struttura dei dati e dall'implementazione fisica. In tale schema concettuale, che verrà rappresentato usando un class diagram UML, si evidenzieranno le entità (concetti) rilevanti ai fini della rappresentazione dei dati e le relazioni che intercorrono tra esse. Si delineeranno anche eventuali vincoli da imporre.

2.1 Class Diagram

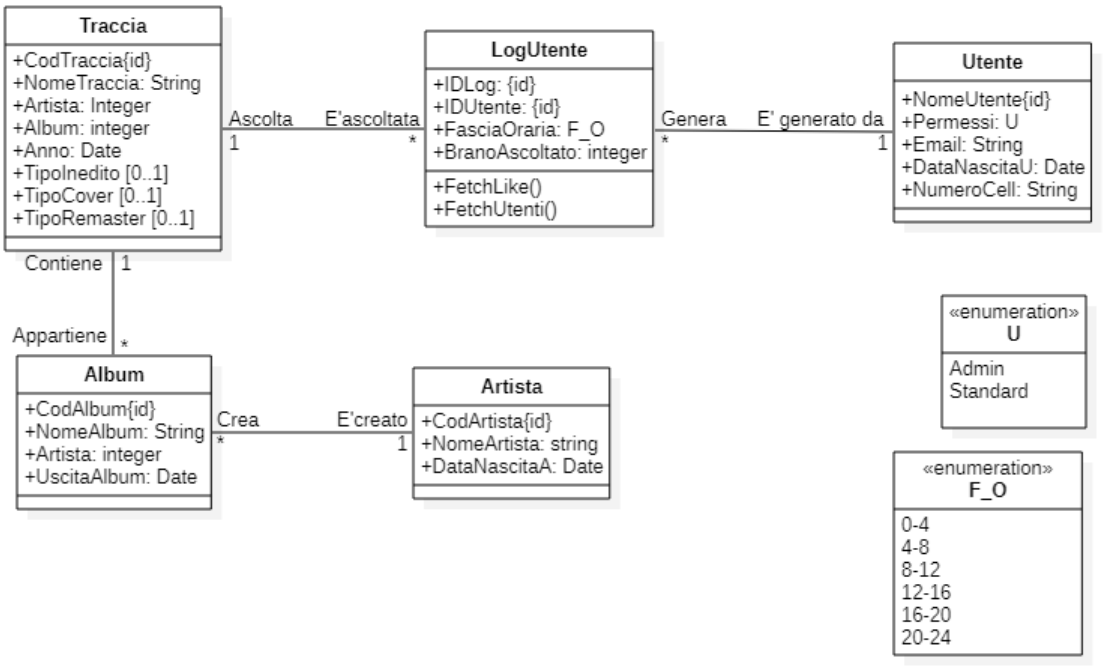


2.2 Ristrutturazione del Class Diagram

Al fine di rendere il class diagram idoneo alla traduzione in schemi relazionali e di migliorare l'efficienza dell'implementazione si procede alla ristrutturazione dello stesso. Al termine del procedimento il class diagram non conterrà attributi strutturati, attributi multipli e gerarchie di specializzazione.

2.2.1 Analisi delle ridondanze

Eliminiamo le due sottoclassi di Utente creando LogUtente che tiene conto di quale utente abbia ascoltato una determinata traccia e contiene le operazioni richieste dalla traccia, cioè il sottogruppo di utenti che hanno effettuato più ascolti su una determinata traccia e il capire quale siano i brani preferiti degli utenti.



2.3 Dizionario dei dati

2.3.1 Dizionario delle classi

Dizionario delle classi Classe	Descrizione	Attributi
Utente	Descrittore di utenti registrati alla libreria	**NomeUtente(String)**: Identifica il nickname dell'utente. **Permessi(string)**: Identifica se un utente è un utente admin o un utente standard. **Email(string)**: Identifica l'email dell'utente. **DataNascitaU(Date)**: Identifica la data di nascita dell'utente. **NumeroCell(String)**:Identifica il numero di cellulare dell'utente.
LogUtente	Descrittore di un determinato utente	**IDLog(Serial)**: Chiave tecnica, identifica univocamente il Log creato dall'utente. **IDUtente(integer)**: Identifica l'utente che crea il log. **FasciaOraria(string)**: Identifica la fascia oraria in cui il log è stato creato. **BranoAscoltato(integer)**: Identifica il codice del brano inserito nel Log.
Traccia	Descrittore di una traccia	**CodTraccia(Serial)**: Chiave tecnica, identifica il codice della traccia univocamente. **Artista(lnteger)**:Chiave esterna riferita a CodArtista(Artista). **NomeTraccia(String)**:Identifica il nome della traccia. **Album(lnteger)**:Chiave esterna riferita a CodAlbum(Album). **Anno(Date)**:Identifica l'anno d'uscita del brano. **Tipolnedito(Bool)**: Identifica se la traccia è di tipo Inedito. **TipoCover(Bool)**: Identifica se la traccia è di tipo Cover. **TipoRemaster(Bool)**: Identifica se la traccia è di tipo Remaster.
Album	Descrittore di un album	**CodAlbum(Serial)**: Chiave tecnica, identifica il codice dell'album univocamente. **Artista(lnteger)**: Chiave esterna riferita a CodArtista(Artista). **NomeAlbum(String)**: Identifica il nome dell'album. **UscitaAlbum(Date)**: Identifica la data d'uscita dell'album.
Artista	Descrittore di un artista	**CodArtista(Serial)**: Chiave tecnica, identifica univocamente un artista. **NomeArtista(String)**: Identifica il nome dell'artista. **DataNascitaA(Date)**: Identifica la data di nascita dell'artista.

2.3.2 Dizionario delle associazioni

Dizionario delle classi Classe	Descrizione	Classi Coinvolte
Genera - E' generato da	Esprime la relazione tra Utente e LogUtente	**Utente[1] ruolo Genera**: Indica che l'utente genera il log utente. **LogUtente[*] ruolo E'generato**: Indica che il log degli utenti è generato dall'utente
Ascolta - E' ascoltata	Esprime la relazione tra LogUtente e Traccia	**LogUtente[*]ruolo Ascolta**: Indica che nel log utente è prensente la traccia ascoltata. **Traccia[1] ruolo E'ascoltata**: indica la traccia ascoltata nel log utente
Contiene - Appartiene	Esprime la relazione tra Album e Traccia	**Album[*]ruolo Contiene**: Indica le tracce che appartengono all'album. **Traccia[1] ruolo Appartiene**: Indica la traccia contenuta nell'album.
Crea - E' creato	Esprime la relazione tra Artista ed Album	**Artista[*] ruolo Crea**: Indica l'artista che crea l'album. **Album[1] ruolo E'creato**: Indica l'album che è creato dall'artista.

2.3.3 Dizionario dei vincoli

Nome Vincolo	Descrizione
CheckTipo	Controlla il tipo della traccia, affinché non ci siano dubbi se la traccia è un inedito, unremaster o una cover.
CheckEmail	Controlla l'indirizzo email affinchè rispetti la sintattica corretta.

3 Progettazione logica

In questo capitolo sarà trattata la fase successiva della progettazione della base di dati scendendo ad un livello di astrazione più basso rispetto alla precedente. Si tradurrà lo schema concettuale (già predisposto in seguito alla ristrutturazione) in uno schema logico, dipendente dal tipo di struttura dei dati prescelto cioè quello relazionale puro. Negli schemi relazionali che seguiranno le chiavi primarie sono indicate con una singola sottolineatura mentre le chiavi esterne con una doppia sottolineatura.

3.0.1 Schema Logico

Schema	Chiavi esterne
Utente(NomeUtente, Permessi, Email, DataNascitaU, NumeroCell)	Nessuna
LogUtente(IDLog, IDUtente, Fascia oraria, BranoAscoltato)	IDUtente->Utente.IDUtente
Artista(CodArtista, NomeArtista, DataNascitaA)	Nessuna
Album(CodAlbum, NomeAlbum, Artista, UscitaAlbum)	Artista->Artista.CodArtista
Traccia(CodTraccia, NomeTraccia, Artista, Album. Uscita, Tipolnedito, TipoCover, TipoRemaster)	Artista->Artista.CodArtista Album->Album.CodAlbum

4 Progettazione fisica

4.1 Definizione delle variabili Enum

```sql
/* Tipo di permessi */
Create type Permessi as enum
    ('Standard','Admin');
/* Elenco fasce orarie */
Create type FasciaOraria as enum
    ('00-04','04-08', '08-12', '12-16', '16-20', '20-24');
```

4.2 Definizione della tabella Utente

```sql
CREATE TABLE utente(
  IDUtente      VARCHAR(20) NOT NULL,
  Permessi      Permessi NOT NULL,
  Email         VARCHAR(25)NOT NULL,
  DataNascitaU  DATE NOT NULL,
  NumeroCell    VARCHAR(20) NOT NULL,
  CONSTRAINT    PK_utente PRIMARY KEY(IDUtente),
  CONSTRAINT    CheckEmail CHECK(Email like '%___@___%')
 );
```

4.3 Definizione della tabella LogUtente

```sql
CREATE TABLE LogUtente(
  IDLog             SERIAL NOT NULL,
  IDUtente          VARCHAR(20) NOT NULL,
  FasciaOraria      FasciaOraria NOT NULL,
  BranoAscoltato    INTEGER NOT NULL,
  CONSTRAINT Log_PK PRIMARY KEY (IDLog),
  CONSTRAINT FK_IDU FOREIGN KEY(IDUtente) REFERENCES utente(IDUtente)
                    ON DELETE CASCADE
);
```

4.4 Definizione della tabella Artista

```sql
CREATE TABLE Artista(
  CodArtista        SERIAL NOT NULL,
  NomeArtista       VARCHAR(35) NOT NULL,
  DataNascitaA      DATE NOT NULL,
  CONSTRAINT Artista_PK PRIMARY KEY (CodArtista)
);
```

4.5 Definizione della tabella Album

```sql
CREATE TABLE album (
  CodAlbum      SERIAL NOT NULL,
  Artista       INTEGER NOT NULL,
  NomeAlbum     VARCHAR(100) NOT NULL,
  UscitaAlbum   DATE NOT NULL,
  CONSTRAINT Album_PK PRIMARY KEY (CodAlbum),
  CONSTRAINT Artista_FK FOREIGN KEY(Artista) REFERENCES Artista(CodArtista)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE
);
```

4.6 Definizione della tabella Traccia

```sql
CREATE TABLE tracce (
  CodTraccia        SERIAL NOT NULL,
  NomeTraccia       VARCHAR(100) NOT NULL,
  Artista           INTEGER NOT NULL,,
  Album             INTEGER NOT NULL,,
  Anno              DATE NOT NULL,,
  TipoInedito       BOOLEAN NOT NULL,
  TipoCover         BOOLEAN NOT NULL,
  TipoRemaster      BOOLEAN NOT NULL,
  CONSTRAINT Traccia_PK PRIMARY KEY (CodTraccia),
  CONSTRAINT Artista_FK2 FOREIGN KEY(artista) REFERENCES artista(CodArtista)
                              ON DELETE CASCADE
                              ON UPDATE CASCADE,
  CONSTRAINT Album_FK FOREIGN KEY(album) REFERENCES album(CodAlbum)
                              ON DELETE CASCADE
                              ON UPDATE CASCADE,
  CONSTRAINT CheckTipo CHECK((TipoInedito=TRUE AND TipoCover=FALSE AND TipoRemaster=FALSE) OR
                          (TipoInedito=FALSE AND TipoCover=TRUE AND TipoRemaster=FALSE) OR
                          (TipoInedito=FALSE AND TipoCover=FALSE AND TipoRemaster=TRUE))
);
```

4.7 Definizione della funzione Fetch Like

```sql
CREATE PROCEDURE Fetch_Like (nTraccia tracce.CodTraccia%TYPE, minAscolti INT)
AS
BEGIN
    SELECT    L.IDUtente,T.CodTraccia, T.NomeTraccia, T.Anno, COUNT(*) AS Ascolti
    FROM      Log_Utente AS L, tracce AS T
    WHERE     nTraccia=T.CodTraccia
    GROUP BY  L.IDUtente, T.NomeTraccia, T.Anno
    HAVING    COUNT(*) >= minAscolti
    ORDER BY  T.Anno ASC;
END;
```

4.8 Definizione della funzione Fetch fascia oraria

```
CREATE FUNCTION Fetch_fascia_oraria(idUtente utente.IDUtente%TYPE)
   RETURN VARCHAR
DECLARE
   fOraria VARCHAR;
   idUt utente.IDUtente%TYPE;
   CURSOR ControlloFasciaOraria is
             SELECT count(*) as ascolti, fasciaOraria
             FROM LogUtente
             WHERE IDUtente=idUt
         GROUP BY fasciaOraria
             ORDER BY (ascolti) DESC LIMIT 1;
   rec_Log ControlloFasciaOraria%ROWTYPE;

BEGIN
   idUt:=idUtente;
   OPEN ControlloFasciaOraria;
    LOOP
        FETCH ControlloFasciaOraria into rec_Log;
        EXIT WHEN ControlloFasciaOraria%NOTFOUND;
        IF rec_Log!=NULL then
            fOraria=rec_Log.fasciaOraria;
        ELSE
            fOraria="None";
        END IF;

   CLOSE ControlloFasciaOraria;
   RETURN fOraria;
END;
```