1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

CORSO DI LAUREA IN
INFORMATICA APPLICATA
SCUOLA DI
SCIENZE TECNOLOGIE E FILOSOFIA DELL'INFORMAZIONE

Basi di Dati – Progetto a.a. 2020/2021

“Piattaforma Streming”

Sviluppatori:
Simone Cossi, mat. 290796

INDICE

ANALISI DEI REQUISITI

Descrizione dei requisiti in linguaggio naturale

Un cliente (che può avere un solo account) è identificato con username e password oltre a consueti dati anagrafici.
I clienti sottoscrivono un piano che ha data di attivazione e di termine. I piani (3 in tutto) sono mensili e si differenziano tra loro in base al numero di dispositivi che è possibile utilizzare in contemporanea e al livello di definizione dello streaming.

Sia film che documentari hanno: uno o più registi, anno di uscita, durata, descrizione, genere, lingua/e disponibili (audio e sottotitoli) e una durata di permanenza nella piattaforma.

I film hanno riferimenti agli attori, che possono intrepretare più film e che possono essere protagonisti o meno.

Gli eventi sportivi hanno un nome, data, descrizione e sport.

Per tutti i piani è acquistabile anche l'opzione SPORT che consente di fruire anche degli eventi sportivi, altrimenti preclusi ai clienti.
Alcuni eventi sportivi di spicco possono essere acquistati solo in modalità stand-alone, con un costo extra.

Operazioni compiute dal database

Interrogazioni:

- Elenco degli abbonamenti con l'opzione Sport "No"
- Elenco degli Abbonamenti aperti nel 2021 in ordine di data di attivazione
- Storico degli utenti che hanno un abbonamento
- Elenco degli abbonamenti con prezzo superiore a 10 euro
- Elenco dei registi che hanno partecipato a più film
- Storico abbonamento di un dato utente

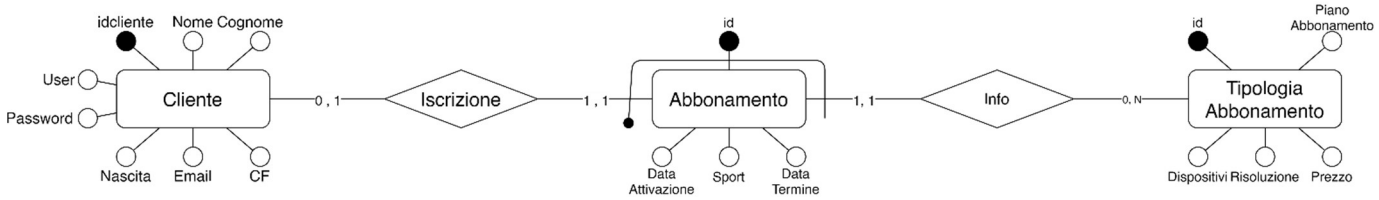
Aggiornamenti:

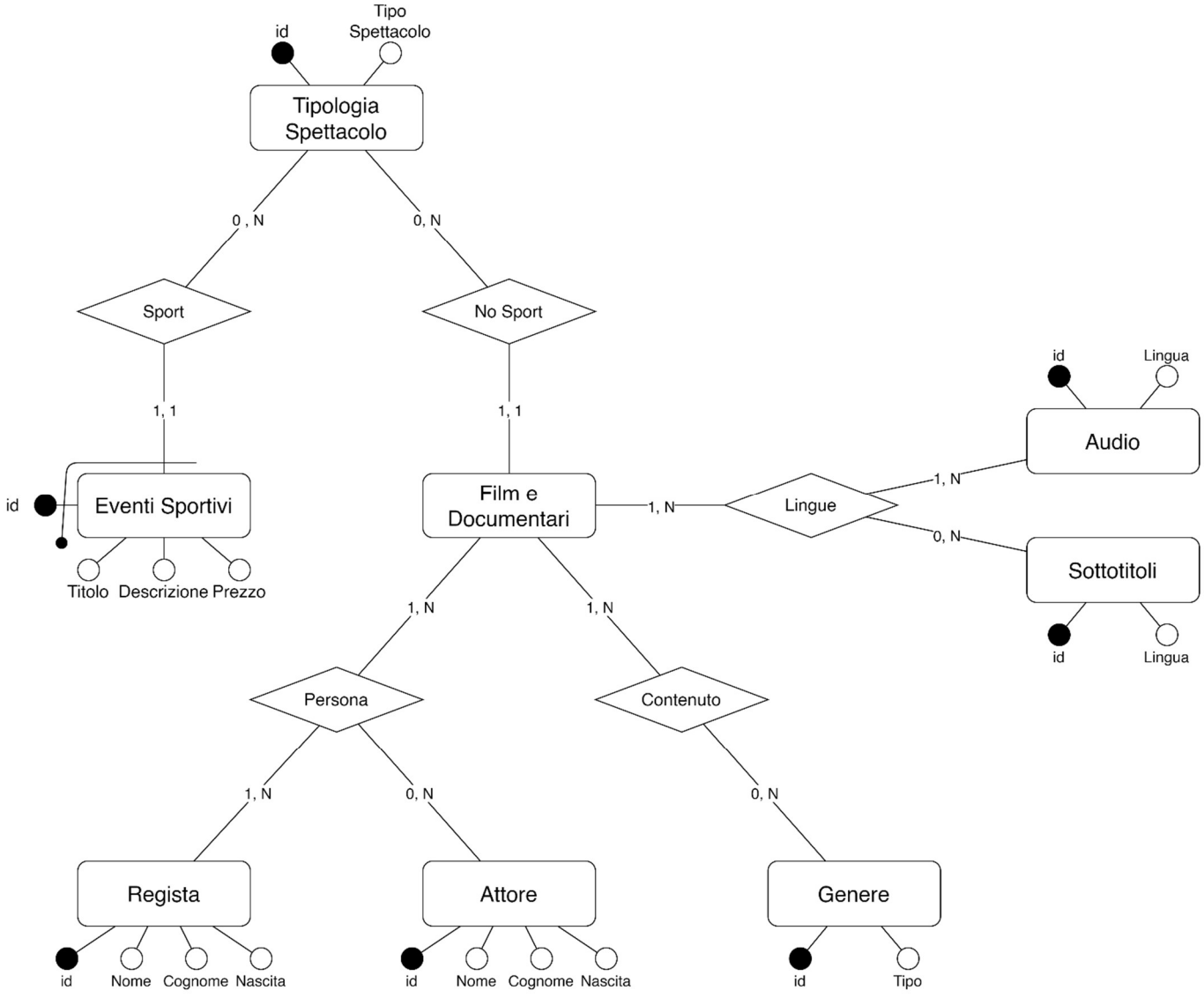
- Iscrizione di un nuovo cliente alla piattaforma
- Eliminazione di un cliente preesistente
- Inserimento di un nuovo abbonamento
- Modifica caratteristiche di un abbonamento già esistente
- Eliminazione di un abbonamento preesistente
- Modifica dei dati degli utenti già registrati
- Inserimento di un nuovo regista
- Inserimento di un nuovo attore
- Inserimento di aventi sportivi
- Eliminazione di un evento sportivo

PROGETTAZIONE CONCETTUALE

Progettare concettualmente un database significa individuare gli oggetti (o entità) che la costituiscono e le relazioni tra un oggetto e l'altro.
Lo scopo della progettazione concettuale è quello di fornire una rappresentazione delle specifiche informali in termini di una descrizione formale e completa del contenuto informativo del database.

Schema E/R





Entità

Di seguito verranno riportate le entità che sono state prese in considerazione per lo sviluppo del database:

cliente → Questa entità contiene i dati di ogni utente registrato

abbonamento → In questa entità sono contenuti gli abbonamenti dei vari clienti

tipologiaabbonamento → Questa entità comprende i tre tipi di abbonamenti da scegliere

tipologiaspettacolo → Questa entità contiene all'interno le tipologie di spettacolo

eventisportivi → Questa entità comprende tutti gli eventi sportivi

filmdocumentari → Questa entità comprende tutto il catalogo film e documentari

generefilm → In questa entità vengono creati i generi dei contenuti in modo tale da abbinare il genere o più generi al film o al documentario

generefilm_has_filmdocumentari → In questa entità sono contenute le associazioni di ogni film o documentario con ogni genere posseduto

registi_has_filmdocumentari → In questa entità sono contenute le informazioni riguardanti l'associazione del regista o più registi a ciascun film o documentario

registi → In questa entità vengono memorizzati i registi che possono far parte di uno o più film e documentari

attori_has_filmdocumentari → Questa entità contiene le associazioni tra ogni attore e ciascun film o documentario

attori → Questa entità memorizza i dati di ogni attore

sottotitoli_has_filmdocumentari → Questa entità contiene le associazioni tra i sottotitoli e i film o documentari

sottotitoli → In questa entità sono memorizzati i sottotitoli

lingue_has_filmdocumentari → Questa entità contiene le associazioni tra le lingue disponibili e i film o documentari che dispongono di determinate lingue

lingue → In questa entità vengono memorizzate tutte le lingue disponibili

Relazioni

Cliente – Abbonamento

Un cliente può avere da 0 (es di un abbonamento scaduto) a 1 abbonamenti al massimo.

Un abbonamento invece appartiene sempre e solo ad una singola persona.

Abbonamento – Tipologia Abbonamento

Un abbonamento deve e può essere solo di una tipologia per volta.

Una tipologia di abbonamento può essere attribuita a nessun abbonamento (es di un db con ancora pochi abbonamenti e nessuno di un determinato tipo) come può essere attribuita a N abbonamenti.

Eventi Sportivi – Tipologia Spettacolo

Ogni evento sportivo deve e può essere solo di un tipo di spettacolo per volta (Sport per l'esattezza).

Una tipologia di spettacolo può non essere attribuita ad alcun evento come può essere attribuita a N eventi.

Film Documentari – Tipologia Spettacolo

Ogni film o documentario deve e può essere attribuito a solo una tipologia di spettacolo per volta.

Una tipologia di spettacolo può non essere attribuita ad alcun film o documentario, allo stesso modo può essere attribuita N volte.

Film Documentari – Genere Film

Ogni film o documentario possiede almeno un genere, nonostante ciò, ne può avere fino a N.

Un genere può non essere attribuito ad alcun film come può invece essere attribuito a N film.

Film Documentari – Registi

Ogni film o documentario possiede almeno un regista, ciò non toglie che un film o un documentario ne possa avere N.

Un regista non è necessariamente attribuito ad almeno un film (es: regista di film non più presenti sulla piattaforma). Potrebbe però essere attribuito a N film o documentari.

Film Documentari – Attori

Un film o un documentario può non avere attori (es: documentario sulla natura con una sola voce narrante), ciò non toglie che un film o un documentario ne possa avere N.

Un attore non è necessariamente presente in almeno un film. Nonostante ciò, potrebbe essere in N film o documentari.

Film Documentari – Sottotitoli

Un film o un documentario non posseggono necessariamente sottotitoli, rimane il fatto che ne potrebbero avere anche N.

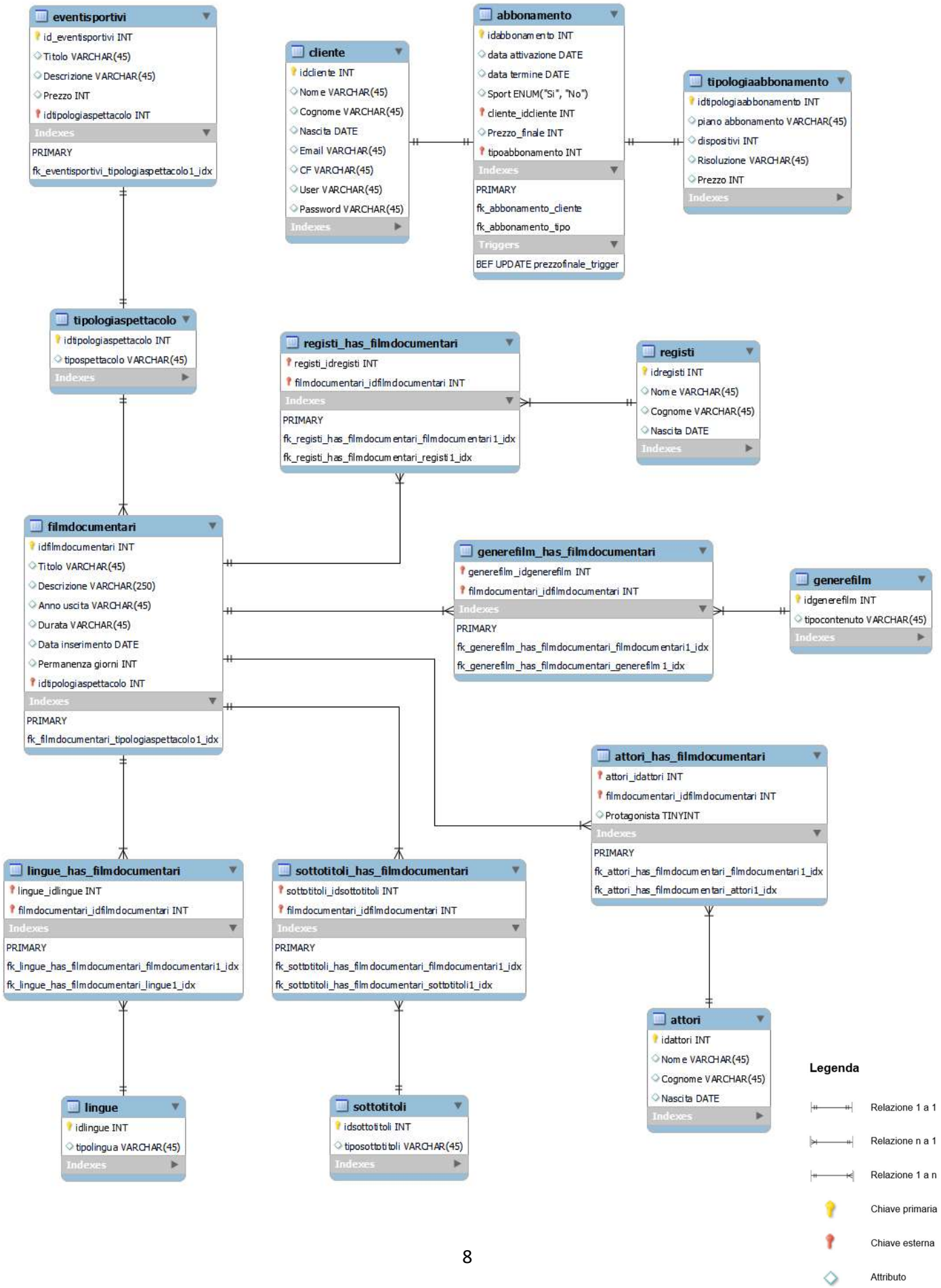
Non ci sono obbligatoriamente sottotitoli per ogni film o documentario, ciò nonostante, ci potrebbero essere N sottotitoli (in lingue diverse) per lo stesso film.

Film Documentari – Audio

Un Film potrebbe non avere un audio come potrebbe avere N audio (in lingue diverse).

Esiste almeno un audio per ogni film, potrebbero però esserci più audio per film.

Schema E/R Finale



PROGETTAZIONE LOGICA

Ottimizzazione

L'ottimizzazione adottata riguarda principalmente la scelta degli identificatori, in quanto, per evitare ambiguità, sono stati generati degli attributi con il compito di fungere esclusivamente da chiave primaria.

Oltre a ciò, sono state generate tabelle in più solo per collegare e accoppiare al meglio le informazioni di diverse tabelle.

Semplificazione

In questo DB sono state eseguite più semplificazioni, nel dettaglio possiamo vedere:

La tipologia di abbonamento, per identificare nel modo più veloce i tipi di abbonamento utilizzati dal cliente, infatti è stata creata una tabella separata da abbonamento per poterla richiamare con l'unico ausilio dell'identificativo.

Un'altra semplificazione è stata suddividere "eventi sportivi" da "film e documentari", in modo tale da identificare gli eventi sportivi gratuiti e stand-alone (a cui si può accedere soltanto se nell'abbonamento è stato scelto sport).

Traduzione

La traduzione consiste nell'evidenziare le chiavi primarie ed esterne per ogni tabella.

N.B. → Le chiavi primarie saranno evidenziate in maiuscolo, mentre le chiavi esterne verranno sottolineate

- cliente: (IDCLIENTE, Nome, Cognome, Nascita, Email, CF, User, Password)
- abbonamento: (IDABBONAMENTO, data attivazione, data termine, sport, prezzo_finale, cliente_idcliente, tipoabbonamento)

FOREIGN KEY ('cliente_idcliente') REFERENCES 'cliente' ('idcliente')

FOREIGN KEY ('tipoabbonamento') REFERENCES 'tipologiaabbonamento' ('idtipologiaabbonamento')

- tipologiaabbonamento: (IDTIPOLOGIAABBONAMENTO, pianoabbonamento, dispositivi, risoluzione, prezzo)
- eventisportivi: (ID_EVENTISPORTIVI, titolo, descrizione, prezzo, idtipologiaspettacolo)

FOREIGN KEY ('idtipologiaspettacolo') REFERENCES 'tipologiaspettacolo'('idtipologiaspettacolo')

- tipologiaspettacolo: (IDTIPOLOGIASPETTACOLO, tipospettacolo)
- filmdocumentari: (IDFILMDOCUMENTARI, titolo, descrizione, anno uscita, durata, data inserimento, permanenza giorni, idtipologiaspettacolo)

FOREIGN KEY (‘idtipologiaspettacolo’) REFERENCES ‘tipologiaspettacolo’(‘idtipologiaspettacolo’)

- generefilm: (IDGENEREFILM, tipocontenuto)
- generefilm_has_filmdocumentari: (generefilm_idgenerefilm, filmdocumentari_idfilmdocumentari)

FOREIGN KEY (‘generefilm_idgenerefilm’) REFERENCES ‘generefilm’ (‘idgenerefilm’)

FOREIGN KEY (‘filmdocumentari_idfilmdocumentari’) REFERENCES ‘filmdocumentari’ (‘idfilmdocumentari’)

- registi: (IDREGISTI, nome, cognome, nascita)
- registi_has_filmdocumentari: (registi_idregisti, filmdocumentari_idfilmdocumentari)

FOREIGN KEY (‘registi_idregisti’) REFERENCES ‘registi’ (‘idregisti’)

FOREIGN KEY (‘filmdocumentari_idfilmdocumentari’) REFERENCES ‘filmdocumentari’ (‘idfilmdocumentari’)

- attori: (IDATTORI, nome, cognome, nascita)
- attori_has_filmdocumentari: (attori_idattori, filmdocumentari_idfilmdocumentari)

FOREIGN KEY (‘attori_idattori’) REFERENCES ‘attori’ (‘idattori’)

FOREIGN KEY (‘filmdocumentari_idfilmdocumentari’) REFERENCES ‘filmdocumentari’ (‘idfilmdocumentari’)

- sottotitoli: (IDSOTTOTITOLI, tiposottotitoli)
- sottotitoli_has_filmdocumentari: (sottotitoli_idsottotitoli, filmdocumentari_idfilmdocumentari)

FOREIGN KEY (‘sottotitoli_idsottotitoli’) REFERENCES ‘sottotitoli’ (‘idsottotitoli’)

FOREIGN KEY (‘filmdocumentari_idfilmdocumentari’) REFERENCES ‘filmdocumentari’ (‘idfilmdocumentari’)

- lingue: (IDLINGUE, tipolingua)
- lingue_has_filmdocumentari: (lingue_idlingue, filmdocumentari_idfilmdocumentari)

FOREIGN KEY (‘lingue_idlingue’) REFERENCES ‘lingue’ (‘idlingue’)

FOREIGN KEY (‘filmdocumentari_idfilmdocumentari’) REFERENCES ‘filmdocumentari’ (‘idfilmdocumentari’)

Normalizzazione

In una base di dati relazionale, una forma normale, è una proprietà che garantisce la qualità e la correttezza dello schema in questione.

La normalizzazione è il processo che permette di trasformare schemi “grezzi” in schemi che soddisfano tale forma.

Esistono più tipi di forme normali, quelle che più ci interessano sono:

- Prima forma normale (1NF)
- Seconda forma normale (2NF)
- Terza forma normale (3NF)
- Forma normale di Boyce-Codd (BCNF)

Prima forma normale

Uno schema è in prima forma normale se e solo se i domini degli attributi sono valori atomici.

- Ogni riga della tabella differisce dalle altre
- Gli attributi sono informazioni elementari
- Non sono presenti attributi ripetitivi

In questo progetto ogni riga è diversa dalle altre, infatti ogni relazione ha come chiave primaria un codice unico che garantisce la loro diversità, le informazioni in esse contenuti sono elementari (es. nome, cognome, età) e, fatta esclusione per le chiavi esterne, non sono presenti attributi ripetitivi.

Possiamo quindi affermare che il progetto è in prima forma normale.

Seconda forma normale

Per ogni relazione tutti gli attributi non chiave dipendono funzionalmente dall’intera chiave composta;

In questo database non esistono chiavi composte e si può dunque affermare che ogni attributo può dipendere da una chiave intera; perciò, ogni relazione rispetta le regole della seconda forma normale.

Terza forma normale

Tutti gli attributi non chiave dipendono solo dalla chiave.

Osservando il database si può affermare che tutte le relazioni sono in terza forma normale.

Forma normale di Boyce-Codd (BCNF)

Per ogni dipendenza funzionale non banale X Y (con Y non in X), X `e una superchiave della relazione.

Andando a individuare le chiavi delle entità:

1. cliente (IDCLIENTE, Nome, Cognome, Nascita, Email, CF, User, Password)

Chiavi candidate:

- idcliente
- Nome, Cognome, CF, Nascita
- Email
- CF

Le dipendenze funzionali presenti in questa entità sono:

- {idcliente} → {User}, {idcliente} → {Nome, Cognome, CF, Nascita}, {idcliente} → {Email}
- {User} → {idcliente}, {User} → {Nome, Cognome, CF, Nascita}, {User} → {Email}
- {Nome, Cognome, CF, Nascita} → {idcliente}, {Nome, Cognome, CF, Nascita} → {User}, {Nome, Cognome, CF, Nascita} → {User}
- {Email} → {idcliente}, {Email} → {Nome, Cognome, CF, Nascita}, {Email} → {User}

BCNF soddisfatta. Il primo termine delle dipendenze funzionali contiene una chiave.

2. abbonamento (IDABBONAMENTO, data attivazione, data termine, sport, prezzo_finale, cliente_idcliente, tipoabbonamento)

Chiavi candidate:

- idabbonamento

Le dipendenze funzionali presenti in questa entità sono:

- {idabbonamento} → {prezzo_finale, dataattivazione, datatermine, sport, prezzo_finale, tipoabbonamento, cliente_idcliente}

BCNF soddisfatta. Il primo termine delle dipendenze funzionali contiene una chiave.

3. TipologiaAbbonamento (IDTIPOLOGIAABBONAMENTO, pianoabbonamento, dispositivi, risoluzione, prezzo)

Chiavi candidate:

- Idtipologiaabbonamento
- pianoabbonamento
- dispositivi, risoluzione, prezzo

Le dipendenze funzionali presenti in questa entità sono:

- {idtipologiaabbonamento} → {piano abbonamento}, {idtipologiaabbonamento} → {dispositivi, risoluzione, prezzo}
- {pianoabbonamento} → {idtipologiaabbonamento}, {piano abbonamento} → {dispositivi, risoluzione, prezzo}
- {dispositivi, risoluzione, prezzo} → {pianoabbonamento}, {dispositivi, risoluzione, prezzo} → {tipologiaabbonamento}

BCNF soddisfatta. Il primo termine delle dipendenze funzionali contiene una chiave.

4. Tipologiaspettacolo (IDTIPOLOGIASPETTACOLO, tipospettacolo)

Chiavi candidate:

- Idtipologiaspettacolo
- tipospettacolo

Le dipendenze funzionali presenti in questa entità sono:

- {idtipologiaspettacolo} → {tipospettacolo}
- {tipospettacolo} → {idtipologiaspettacolo}

BCNF soddisfatta. Essendo chiavi candidate e in dipendenza funzionale, si può dire che la relazione è in forma BCNF.

5. Eventi sportivi (ID_EVENTISPORTIVI, titolo, descrizione, prezzo, idtipologiaspettacolo)

Chiavi candidate:

- id_eventisportivi
- titolo, descrizione

Le dipendenze funzionali presenti in questa entità sono:

- {id_eventisportivi} → {titolo, descrizione, prezzo}
- {titolo, descrizione} → {id_eventisportivi} → {prezzo}

BCNF soddisfatta. Essendo chiavi candidate e in dipendenza funzionale, si può dire che la relazione è in forma BCNF.

6. Filmdocumentari (IDFILMDOCUMENTARI, titolo, descrizione, anno uscita, durata, datainserimento, permanenzagiorni, idtipologiaspettacolo)

Chiavi candidate:

- idfilmdocumentari

Le dipendenze funzionali presenti in questa entità sono:

- {idfilmdocumentari} → {titolo, descrizione, annouscita, durata, datainserimento, permanenzagiorni, idtipolgiaspettacolo}
- {titolo, descrizione, annouscita, durata} → {idfilmdocumentari} → {datainserimento, permanenzagiorni}

BCNF soddisfatta. Il primo termine delle dipendenze funzionali contiene una chiave.

7. Generefilm (IDGENEREFILM, tipocontenuto)

Chiavi candidate:

- Idgenerefilm
- tipocontenuto

Le dipendenze funzionali presenti in questa entità sono:

- {idgenerefilm} → {tipocontenuto}
- {tipocontenuto} → {idgenerefilm}

BCNF soddisfatta. Essendo chiavi candidate e in dipendenza funzionale, si può dire che la relazione è in forma BCNF.

8. Registi (IDREGISTI, nome, cognome, nascita)

Chiavi candidate:

- idregisti
- nome

Le dipendenze funzionali presenti in questa entità sono:

- {idregisti} → {nome, cognome, nascita}

BCNF soddisfatta. Il primo termine delle dipendenze funzionali contiene una chiave.

PROGETTAZIONE FISICA

Implementazione del database in MySQL

Di seguito viene riportata la creazione dello schema del db in linguaggio MySQL

```
-- -----------------------------------------------------
-- Schema databasestreaming
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `databasestreaming` DEFAULT CHARACTER SET utf8mb4 ;
USE `databasestreaming` ;

-- -----------------------------------------------------
-- Table `databasestreaming`.`cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`cliente` (
  `idcliente` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NULL,
  `Cognome` VARCHAR(45) NULL,
  `Nascita` DATE NULL,
  `Email` VARCHAR(45) NULL,
  `CF` VARCHAR(45) NULL,
  `User` VARCHAR(45) NULL,
  `Password` VARCHAR(45) NULL,
  PRIMARY KEY (`idcliente`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`tipologiaAbbonamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`tipologiaabbonamento` (
  `idtipologiaabbonamento` INT NOT NULL AUTO_INCREMENT,
  `pianoabbonamento` VARCHAR(45) NULL,
  `dispositivi` INT NULL,
  `risoluzione` VARCHAR(45) NULL,
  `prezzo` INT NULL,
  PRIMARY KEY (`idtipologiaabbonamento`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`abbonamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`abbonamento` (
  `idabbonamento` INT NOT NULL AUTO_INCREMENT,
  `dataattivazione` DATE NULL,
  `datatermine` DATE NULL,
  `sport` ENUM("Si", "No") NULL,
  `cliente_idcliente` INT NOT NULL,
  `prezzo_finale` INT NULL,
  `tipoabbonamento` INT NOT NULL,
  PRIMARY KEY (`idabbonamento`, `cliente_idcliente`, `tipoabbonamento`),
  INDEX `fk_abbonamento_cliente` (`cliente_idcliente` ASC) ,
  INDEX `fk_abbonamento_tipo` (`tipoabbonamento` ASC) ,
  CONSTRAINT `fk_abbonamento_cliente`
    FOREIGN KEY (`cliente_idcliente`)
    REFERENCES `databasestreaming`.`cliente` (`idcliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_abb_tipoabb`
    FOREIGN KEY (`tipoabbonamento`)
    REFERENCES `databasestreaming`.`tipologiaabbonamento` (`idtipologiaabbonamento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `databasestreaming`.`generefilm`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`generefilm` (
  `idgenerefilm` INT NOT NULL,
  `tipocontenuto` VARCHAR(45) NULL,
  PRIMARY KEY (`idgenerefilm`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`tipologiaspettacolo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`tipologiaspettacolo` (
  `idtipologiaspettacolo` INT NOT NULL AUTO_INCREMENT,
  `tipospettacolo` VARCHAR(45) NULL,
  PRIMARY KEY (`idtipologiaspettacolo`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`filmdocumentari` (
  `idfilmdocumentari` INT NOT NULL,
  `titolo` VARCHAR(45) NULL,
  `descrizione` VARCHAR(250) NULL,
  `annouscita` VARCHAR(45) NULL,
  `durata` VARCHAR(45) NULL,
  `datainserimento` DATE NULL,
  `permanenza` INT NULL,
  `idtipologiaspettacolo` INT NOT NULL,
  PRIMARY KEY (`idfilmdocumentari`, `idtipologiaspettacolo`),
  INDEX `fk_filmdocumentari_tipologiaspettacolo1_idx` (`idtipologiaspettacolo` ASC) ,
  CONSTRAINT `fk_filmdocumentari_tipologiaspettacolo1`
    FOREIGN KEY (`idtipologiaspettacolo`)
    REFERENCES `databasestreaming`.`tipologiaspettacolo` (`idtipologiaspettacolo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`attori`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`attori` (
  `idattori` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NULL,
  `cognome` VARCHAR(45) NULL,
  `nascita` DATE NULL,
  PRIMARY KEY (`idattori`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`registi`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`registi` (
  `idregisti` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NULL,
  `cognome` VARCHAR(45) NULL,
  `nascita` DATE NULL,
  PRIMARY KEY (`idregisti`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`lingue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`lingue` (
```

```
  `idlingue` INT NOT NULL AUTO_INCREMENT,
  `tipolingua` VARCHAR(45) NULL,
  PRIMARY KEY (`idlingue`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`sottotitoli`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`sottotitoli` (
  `idsottotitoli` INT NOT NULL AUTO_INCREMENT,
  `tiposottotitoli` VARCHAR(45) NULL,
  PRIMARY KEY (`idsottotitoli`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`eventisportivi`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`eventisportivi` (
  `id_eventisportivi` INT NOT NULL AUTO_INCREMENT,
  `titolo` VARCHAR(45) NULL,
  `descrizione` VARCHAR(45) NULL,
  `prezzo` INT NULL,
  `idtipologiaspettacolo` INT NOT NULL,
  PRIMARY KEY (`id_eventisportivi`, `idtipologiaspettacolo`),
  INDEX `fk_eventisportivi_tipologiaspettacolo1_idx` (`idtipologiaspettacolo` ASC) ,
  CONSTRAINT `fk_eventisportivi_tipologiaspettacolo1`
    FOREIGN KEY (`idtipologiaspettacolo`)
    REFERENCES `databasestreaming`.`tipologiaspettacolo` (`idtipologiaspettacolo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`registi_has_filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`registi_has_filmdocumentari` (
  `registi_idregisti` INT NOT NULL,
  `filmdocumentari_idfilmdocumentari` INT NOT NULL,
  PRIMARY KEY (`registi_idregisti`, `filmdocumentari_idfilmdocumentari`),
  INDEX `fk_registi_has_filmdocumentari_filmdocumentari1_idx` (`filmdocumentari_idfilmdocumentari` ASC) ,
  INDEX `fk_registi_has_filmdocumentari_registi1_idx` (`registi_idregisti` ASC) ,
  CONSTRAINT `fk_registi_has_filmdocumentari_registi1`
    FOREIGN KEY (`registi_idregisti`)
    REFERENCES `databasestreaming`.`registi` (`idregisti`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_registi_has_filmdocumentari_filmdocumentari1`
    FOREIGN KEY (`filmdocumentari_idfilmdocumentari`)
    REFERENCES `databasestreaming`.`filmdocumentari` (`idfilmdocumentari`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`attori_has_filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`attori_has_filmdocumentari` (
  `attori_idattori` INT NOT NULL,
  `filmdocumentari_idfilmdocumentari` INT NOT NULL,
  `protagonista` TINYINT NULL,
  PRIMARY KEY (`attori_idattori`, `filmdocumentari_idfilmdocumentari`),
  INDEX `fk_attori_has_filmdocumentari_filmdocumentari1_idx` (`filmdocumentari_idfilmdocumentari` ASC) ,
  INDEX `fk_attori_has_filmdocumentari_attori1_idx` (`attori_idattori` ASC) ,
  CONSTRAINT `fk_attori_has_filmdocumentari_attori1`
    FOREIGN KEY (`attori_idattori`)
    REFERENCES `databasestreaming`.`attori` (`idattori`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_attori_has_filmdocumentari_filmdocumentari1`
    FOREIGN KEY (`filmdocumentari_idfilmdocumentari`)
    REFERENCES `databasestreaming`.`filmdocumentari` (`idfilmdocumentari`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`lingue_has_filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`lingue_has_filmdocumentari` (
  `lingue_idlingue` INT NOT NULL,
  `filmdocumentari_idfilmdocumentari` INT NOT NULL,
  PRIMARY KEY (`lingue_idlingue`, `filmdocumentari_idfilmdocumentari`),
  INDEX `fk_lingue_has_filmdocumentari_filmdocumentari1_idx` (`filmdocumentari_idfilmdocumentari` ASC) ,
  INDEX `fk_lingue_has_filmdocumentari_lingue1_idx` (`lingue_idlingue` ASC) ,
  CONSTRAINT `fk_lingue_has_filmdocumentari_lingue1`
    FOREIGN KEY (`lingue_idlingue`)
    REFERENCES `databasestreaming`.`lingue` (`idlingue`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_lingue_has_filmdocumentari_filmdocumentari1`
    FOREIGN KEY (`filmdocumentari_idfilmdocumentari`)
    REFERENCES `databasestreaming`.`filmdocumentari` (`idfilmdocumentari`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`sottotitoli_has_filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`sottotitoli_has_filmdocumentari` (
  `sottotitoli_idsottotitoli` INT NOT NULL,
  `filmdocumentari_idfilmdocumentari` INT NOT NULL,
  PRIMARY KEY (`sottotitoli_idsottotitoli`, `filmdocumentari_idfilmdocumentari`),
  INDEX `fk_sottotitoli_has_filmdocumentari_filmdocumentari1_idx` (`filmdocumentari_idfilmdocumentari` ASC) ,
  INDEX `fk_sottotitoli_has_filmdocumentari_sottotitoli1_idx` (`sottotitoli_idsottotitoli` ASC) ,
  CONSTRAINT `fk_sottotitoli_has_filmdocumentari_sottotitoli1`
    FOREIGN KEY (`sottotitoli_idsottotitoli`)
    REFERENCES `databasestreaming`.`sottotitoli` (`idsottotitoli`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_sottotitoli_has_filmdocumentari_filmdocumentari1`
    FOREIGN KEY (`filmdocumentari_idfilmdocumentari`)
    REFERENCES `databasestreaming`.`filmdocumentari` (`idfilmdocumentari`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `databasestreaming`.`generefilm_has_filmdocumentari`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `databasestreaming`.`generefilm_has_filmdocumentari` (
  `generefilm_idgenerefilm` INT NOT NULL,
  `filmdocumentari_idfilmdocumentari` INT NOT NULL,
  PRIMARY KEY (`generefilm_idgenerefilm`, `filmdocumentari_idfilmdocumentari`),
  INDEX `fk_generefilm_has_filmdocumentari_filmdocumentari1_idx` (`filmdocumentari_idfilmdocumentari` ASC) ,
  INDEX `fk_generefilm_has_filmdocumentari_generefilm1_idx` (`generefilm_idgenerefilm` ASC) ,
  CONSTRAINT `fk_generefilm_has_filmdocumentari_generefilm1`
    FOREIGN KEY (`generefilm_idgenerefilm`)
    REFERENCES `databasestreaming`.`generefilm` (`idgenerefilm`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_generefilm_has_filmdocumentari_filmdocumentari1`
    FOREIGN KEY (`filmdocumentari_idfilmdocumentari`)
```

```
  REFERENCES `databasestreaming`.`filmdocumentari` (`idfilmdocumentari`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `databasestreaming`;
```

Operazioni in MySQL

Operazioni di interrogazioni:

1. Elenco degli abbonamenti con l'opzione Sport 'No':

```
SELECT *
FROM abbonamento
WHERE sport = 'No'
```

2. Elenco degli abbonamenti aperti nel 2021 in ordine di data attivazione:

```
SELECT *
FROM abbonamento
WHERE `dataattivazione` BETWEEN "2021-01-01" AND "2021-12-31"
 ORDER BY `dataattivazione`
```

3. Storico degli utenti che hanno un abbonamento:

```
SELECT *
FROM cliente JOIN abbonamento ON (cliente.idcliente = abbonamento.cliente_idcliente)
```

4. Elenco degli abbonamenti con prezzo superiore a 10 euro:

```
SELECT idabbonamento, cliente_idcliente, prezzo_finale
FROM abbonamento WHERE prezzo_finale >10
ORDER BY cliente_idcliente
```

5. Elenco dei registi che hanno partecipato a più film:

```
SELECT *
FROM registi_has_filmdocumentari INNER JOIN registi INNER JOIN filmdocumentari
WHERE idregisti = registi_idregisti AND idfilmdocumentari = filmdocumentari_idfilmdocumentari
```

6. Storico abbonamento di un dato utente:

```
SELECT *
FROM abbonamento
WHERE cliente_idcliente = ANY (SELECT idcliente from cliente where nome = 'pino') (pino = 'nomeutente')
```

Operazioni di aggiornamento

1. Iscrizione di un nuovo cliente alla piattaforma:

```
INSERT INTO `cliente` (`idcliente`, `Nome`, `Cognome`, `Nascita`, `Email`, `CF`, `User`, `Password`)
VALUES (`idcliente`, `Nome`, `Cognome`, `Nascita`, `Email`, `CF`, `User`, `Password`);
```

Dove i valori in VALUES dovranno essere sostituiti con i valori del cliente.

2. Inserimento di un nuovo abbonamento:

```
INSERT INTO `abbonamento` (`idabbonamento`, `data attivazione`, `data termine`, `sport`, `cliente_idcliente`, `prezzo_finale`, `tipoabbonamento`)
VALUES ((`idabbonamento`, `data attivazione`, `data termine`, `sport`, `cliente_idcliente`, `prezzo_finale`, `tipoabbonamento`);
```

Dove i valori in VALUES dovranno essere sostituiti con i valori dell'abbonamento da inserire.

3. Modifica dei dati dei clienti già registrati:

```
UPDATE cliente
SET nome = 'nuovonome'
WHERE idcliente = 'idcliente'
```

4. Inserimento di un nuovo regista:

```
INSERT INTO `registi` (`idregisti`, `nome`, cognome`, `nascita`)
VALUES (`idregisti`, `nome`, `cognome`, `nascita`);
```

5. Modifica caratteristiche di un abbonamento già esistente:

```
UPDATE abbonamento
SET `tipoabbonamento` = '2'
WHERE `idabbonamento` = '1'
```

6. Inserimento di un nuovo attore:

```
INSERT INTO `attori` (`idattori`, `nome`, `cognome`, `nascita`)
VALUES (`idattori`, `nome`, `cognome`, `nascita`);
```

7. Eliminazione di un cliente preesistente:
   N.B. attenzione, il cliente non deve avere un abbonamento, nel caso in cui avesse un abbonamento associato, bisogna prima eliminare l'abbonamento

```
DELETE from `cliente`
WHERE `idcliente` = '1'
```

8. Eliminazione di un abbonamento preesistente:

```
DELETE FROM abbonamento
WHERE idabbonamento = '1'
```

9. Inserimento di un evento sportivo:

```
INSERT INTO `eventisportivi` (`id_eventisportivi`, `titolo`, `descrizione`, `prezzo`, `idtipologiaspettacolo`)
VALUES (`id_eventisportivi`, `titolo`, `descrizione`, `prezzo`, `idtipologiaspettacolo`);
```

10. Eliminazione di un evento sportivo:

```
DELETE FROM `eventisportivi`
WHERE id_eventisportivi = 'id_eventisportivi'
```

ERRORI CONOSCIUTI

Il DB è stato sviluppato in fase di apprendimento e miglioramento delle mie conoscenze, a causa di ciò sono presenti alcuni errori o per meglio dire delle parti che potevano venir sicuramente progettate meglio, con meno ridondanza.

Un esempio di ciò è la gestione separata degli attori e dei registi, per questa gestione bastava una generalizzazione che non è stata effettuata.

La domanda è perché non si è andato a correggere gli errori?

La risposta è: il tempo… purtroppo avendo già fatto gli schemi ER e parte della relazione, andare a corregge il tutto avrebbe richiesto tempo non indifferente.

Come per il caso precedente è stato lo stesso per le lingue(audio) e i sottotitoli…

Mi scuso per non aver provveduto alla correzione di tutti gli errori ma non posso permettermi di ritardare la consegna del progetto.