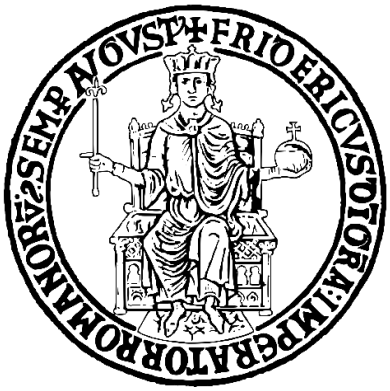
UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TRECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA
INSEGNAMENTO DI BASI DI DATI I
ANNO ACCADEMICO 2020/2021

Progettazione e sviluppo di una base di dati relazionale per un sistema di gestione per un multisala

Autore:
Crescenzo Lucio CICATIELLO
N86003457
cr.cicatiello@studenti.unina.it

Docenti:
Prof. Adriano PERON
Prof. Silvio BARRA

Indice

Descrizione del progetto

Analisi del problema

Si sviluppi una base di dati per la gestione di un cinema multisala. Il sistema deve tenere traccia dei film proiettati in ciascuna sala, della schedulazione di ogni spettacolo e delle caratteristiche di ogni sala (audio dolby, tecnologia IMAX). Il sistema dovrà permettere di stimare quali sono le fasce orarie di maggior affluenza, gli spettacoli più remunerativi e le sale maggiormente occupate durante gli orari di maggior affluenza. Il problema verrà riproposto dopo un'introduzione a vari cenni teorici di un class diagram UML.

Cenni teorici

UML

Per la realizzazione di questo progetto è stato necessario l'utilizzo di un diagramma UML. **L'Unified Modeling Language (UML)** è stato realizzato per rappresentare in modo visivo la struttura ,il comportamento e gli oggetti che compongono un sistema software. Grazie all'uso di questo strumento abbiamo una migliore creazione e modellazione di software principalmente orientato agli oggetti.

Classi

Le **classi** anche dette istanze od entità rappresentato il principale elemento nei diagrammi UML. Ogni classe è composta da degli **attributi**, che descrivono le caratteristiche della classe, e da **operazioni o metodi** che ne derivano il comportamento. Le classi sono rappresentare graficamente come in *Figura 1* In un rettangolo diviso in tre riquadri che contengono rispettivamente nome della classe, attributi ed operazioni.

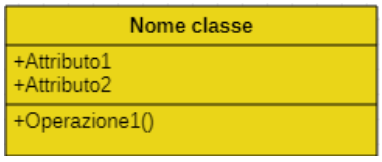


Figura 1 – Esempio di classe

Associazioni

Le **associazioni** rappresentano le relazioni che intercorrono tra le varie classi di un diagramma.

Figura 2 – Rappresentazione di una associazione

Vengono rappresentate con una linea che congiunge le due classi sulla quale sono definite alle estremità le molteplicità dell'associazione ed il ruolo dell'associazione (vedi *Figura 2*).

Le **molteplicità** possono essere ***, 1, 0..1, 1..***.Inoltre l'associazione può essere rappresentata con **una classe collegata con una linea tratteggiata** alla linea di associazione e si possono descrivere le operazioni e gli attributi della classe di associazione.

Le **classi di associazione possono coinvolgere più di due classi** tutte con una molteplicità ***** e vengono rappresentate graficamente con un rompo(vedi *Figura 3*) a cui sono collegate le varie classi con le rispettive linee di associazione.

Inoltre, le associazioni possono specializzarsi e dividersi in:

Aggregazione

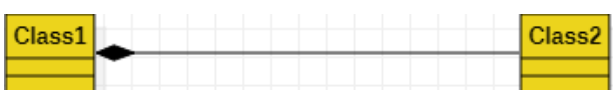


Figura 3 – Rappresentazione di un'aggregazione

Un'aggregazione (vedi *Figura 3*) specifica una associazione tra un aggregato ed una sua parte costituente che può esiste indipendente dall'aggregato. Per esempio, lo studente di

un’aula non dipende dall’aula e viceversa ugual ragionamento è applicabile all’aula. La loro eliminazione non influisce sulle classi ad esse associate.

Composizioni

Figura 4 – Rappresentazione di una composizione

Una composizione (vedi *Figura 4*) identifica invece una forte proprietà tra classi. Per esempio, un’aula che appartiene ad un’università. Se eliminiamo la classe università anche l’aula verrà eliminata di conseguenza.

Generalizzazione

Nel diagramma UML, una **relazione di generalizzazione** (vedi *Figura 5*) è una relazione in cui una classe (sottoclasse) dipende da un'altra classe (classe generale) e che assimila da quest’ultima tutti gli attributi, le operazioni e le relazioni della classe principale. La sottoclasse viene definita specializzazione della classe generale , viceversa la classe generale è detta generalizzazione della sottoclasse.

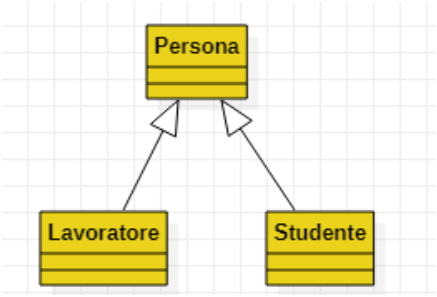


Figura 5 – Generalizzazione

Quando la classe generale è specializzata in varie sottoclassi può essere esclusiva cioè ogni istanza della classe generale deve far parte di una ed una sola delle sottoclassi, mentre nel caso della non esclusività no.

Infine, la generalizzazione può essere totale quindi ogni istanza della classe generale deva far parte almeno ad una sottoclasse mentre in caso della generalizzazione parziale può non appartenere a nessuna delle sottoclassi.

Progettazione concettuale

Class Diagram

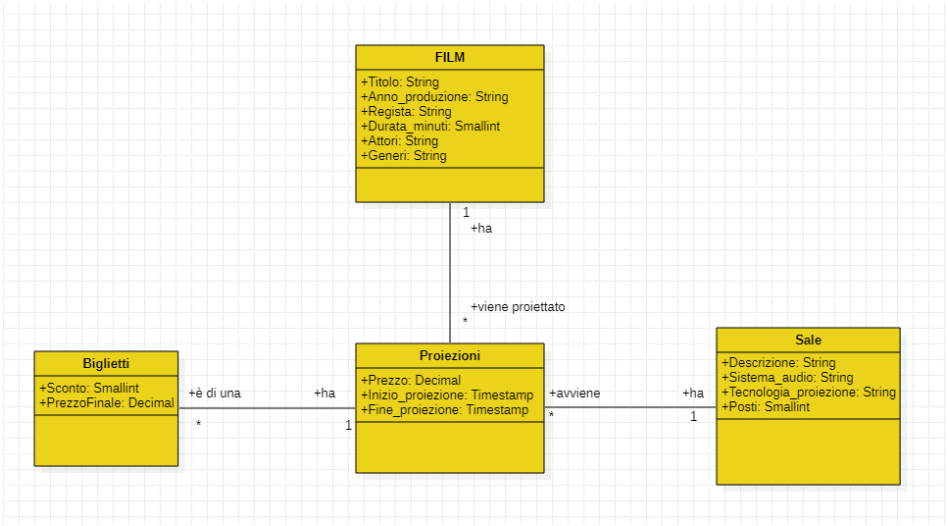


Figura 6 - Class Diagram

Il seguente **class diagram** in *Figura 6* è una prima versione contente attributi multipli ed enumerazioni che successivamente saranno tramutate in entità per poi portare alla creazione di una seconda versione ristrutturata di quest'ultimo.

Attributi multipli

Gli attributi multipli della **attori** sono stati trasformati in classi (come possiamo notare nella *Figura 7)* a causa della loro molteplicità. I generi ed i film come anche gli attori ed i film sono stati collegati da una relazione molti a molti e dalle rispettive associazioni **genere_film** e **attori_film**.

Mentre per gli attributi multipli di **sale** è stato necessario creare delle classi perché non è possibile stimare quali siano o quali saranno i **sistemi audio e tecnologie di proiezione** con il progredire del tempo.

Inoltre, diamo per convenzione che il regista di un film sia al massimo uno.

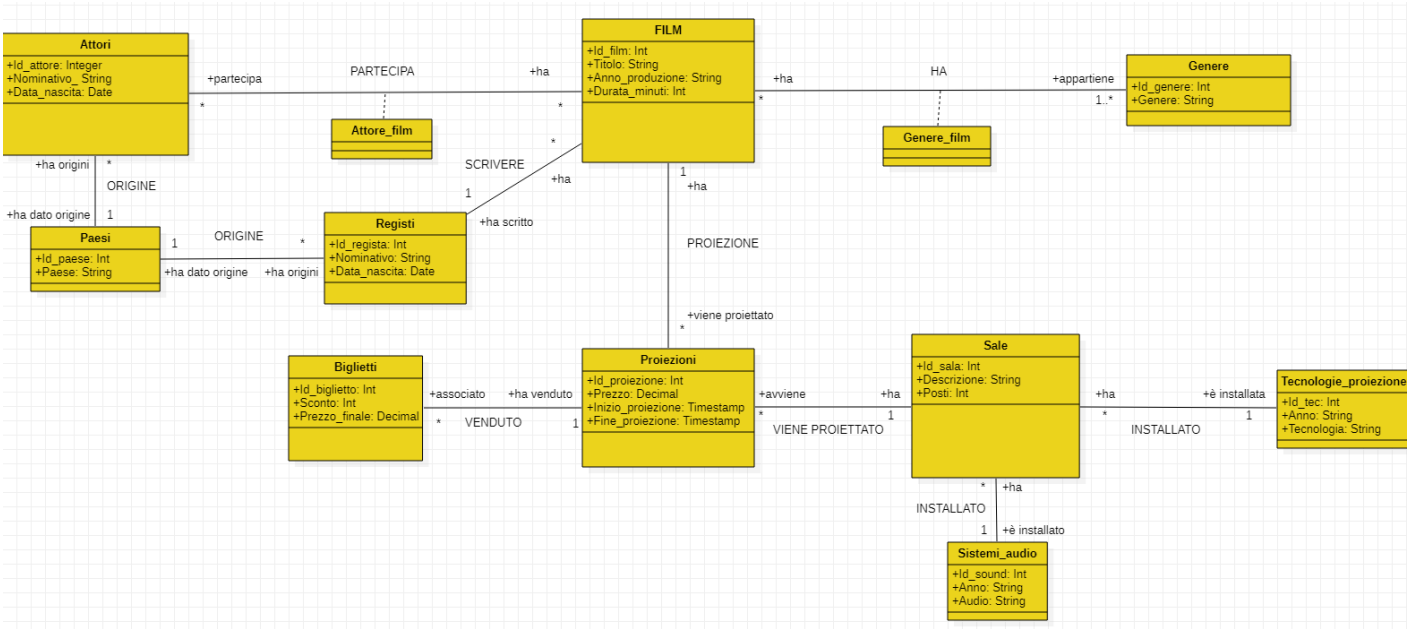


Figura 7- Class Diagram Ristrutturato

La tabella paesi è stata creata per evitare la creazione di un'enumerazione e per aver modo di aggiornare con costanza i paesi col variare del tempo.

Identificativi

Per tutte le classi si è scelto di impiegare chiavi surrogate e quindi per convenzione le chiavi primarie inizieranno con il prefisso Id_.

Dizionari

Dizionario delle classi

Classe	***Descrizione***	Attributi
Film	Definizione della classe film.	**Id_film** (*integer*): chiave che identifica univocamente un'istanza della classe film. **Titolo**(*string*): Nome associato al film. **Anno_produzione**(*string*): Anno in cui il film è stato pubblicato. **Durata_minuti** (*integer*): Durata del film in minuti.
Proiezioni	Definizione della classe proiezioni.	**Id_proiezione** (*integer*): chiave che identifica univocamente un'istanza della classe proiezioni. **Prezzo** (*deciamal*): prezzo della proiezione. **Inizio_proiezione** (*timestamp*): data ed ora in cui inizia la proiezione. **Fine_proiezione** (*timestamp*): data ed ora in cui termina la proiezione.
Sale	Definizione della classe sale.	**Id_sala** (*integer*): chiave che identifica univocamente un'istanza della classe sale. **Descrizione** (*string*): Nome della sala. **Posti** (*integer*): numero di posti massimi.
Biglietti	Definizione della classe biglietti.	**Id_biglietto** (*integer*): chiave che identifica univocamente un'istanza della classe biglietti. **Sconto** (*integer*): sconto da applicare al prezzo della proiezione. **Prezzo_finale** (*decimal*): prezzo finale del biglietto scontato.
Sistemi_audio	Definizione della classe sistemi_audio.	**Id_sound** (*integer*): chiave che identifica univocamente un'istanza della classe sistemi_audio. **Anno** (*string*): anno di creazione. **Audio** (*string*): nome della tecnologia.
Tecnologia_proiezione	Definizione della classe tecnologia_proiezione.	**Id_tec** (*integer*): chiave che identifica univocamente

		un'istanza della classe tecnologia_proiezione. **Anno** (*string*): anno di creazione. **Audio** (*string*): nome della tecnologia.
Registi	Definizione della classe registi.	**Id_regista** (*integer*): chiave che identifica univocamente un'istanza della classe registi. **Nominativo** (*string*): Nome e cognome regista. **Data_nascita** (*date*): data di nascita.
Paesi	Definizione della classe paesi.	**Id_regista** (*integer*): chiave che identifica univocamente un'istanza della classe paesi. **Paese** (*string*): descrizione paese.
Attori	Definizione della classe attori.	**Id_attore** (*integer*): chiave che identifica univocamente un'istanza della classe attori. **Nominativo** (*string*): Nome e cognome regista. **Data_nascita** (*date*): data di nascita.
Attore_film	Descrive l'associazione tra attori e film.	
Genere_film	Descrive l'associazione tra generi e film.	

Dizionario delle associazioni

Associazione	Descrizione	Classi coinvolte
Origine(attore-paesi)	Esprime la relazione tra attore e paesi.	**Classe [1]** *ruolo* (**ha origini**): Indica il paese di origine dell'attore. **Classe [0..*]** *ruolo* (**ha dato origine**): Indica quali attori ha dato origine un paese.
Origine(registi-paesi)	Esprime la relazione tra registi e paesi.	**Classe [1]** *ruolo* (**ha origini**): Indica il paese di origine del regista. **Classe [0..*]** *ruolo* (**ha dato origine**): Indica quali registi ha dato origine un paese.
Partecipa	Esprime la relazione tra attore e film.	**Classe [1..*]** *ruolo* (**ha**): Indica quali attori ha un film. **Classe [1..*]** *ruolo* (**partecipa**): Indica un attore a quali film ha partecipato.
Scrivere	Esprime la relazione tra regista e film.	**Classe [1]** *ruolo* (**ha**): Indica un film da quale regista è stato scritto. **Classe [1..*]** *ruolo* (**ha scritto**): Indica un regista quali film ha scritto.
Ha	Esprime la relazione tra genere e film.	**Classe [1..*]** *ruolo* (**ha**): Indica il film quali generi ha. **Classe [1..*]** *ruolo* (**appartiene**): Indica il genere a quali film appartiene.
Proiezione	Esprime la relazione tra film e proiezioni.	**Classe [1]** *ruolo* (**ha**): Indica quale film viene proiettato. **Classe [1..*]** *ruolo* (**viene proiettato**): Indica in quali proiezioni viene proiettato.
Venduto	Esprime la relazione tra proiezioni e biglietti.	**Classe [0..*]** *ruolo* (**ha venduto**): Indica quali biglietti sono stati venduti per una proiezione. **Classe [1]** *ruolo* (**associato**): Indica a quale proiezione un biglietto è associato.
Viene proiettato	Esprime la relazione tra proiezioni e sale.	**Classe [1]** *ruolo* (**avviene**): Indica in quale sala avviene la proiezione. **Classe [0..*]** *ruolo* (**ha**): Indica una sala quali proiezioni ha.
Installato(tec.proi. -sale)	Esprime la relazione tra sale e tecnologie_proiezione.	**Classe [1]** *ruolo* (**ha**): Indica una sala quale tecnologia di proiezione ha installata. **Classe [0..*]** *ruolo* (**è installata**):Indica in quali sale è installata.
Installato(sis.audio-sale)	Esprime la relazione tra sale e sistemi_audio.	**Classe [1]** *ruolo* (**ha**): Indica una sala quale sistema audio ha installata. **Classe [0..*]** *ruolo* (**è installata**):Indica in quali sale è installata.

Vincolo	*Tipo*	**Descrizione**
DATA_NASCITA_CONCORDE	Dominio	**La data di nascita deve essere concorde e quindi minore della data odierna.**
SOLO_LETTERE	Dominio	**Il nominativo non ammette caratteri numerici.**
CHECKDURATAMAX	Dominio	**Un film non può durare più di 24h.**
CHECKANNO	Dominio	**L'anno di produzione deve essere compreso tra il 1895 e l'anno corrente.**
CHECKPOSTIMAX	Dominio	**Le sale possono essere massimo di 500 posti.**
MAGGIORE_ZERO	Dominio	**Il prezzo deve essere positivo.**
ORARI_CONCORDI	N-upla	**Gli orari di inizio e fine proiezione devono essere concordi.**
INIZIO_FINE_NO24H	N-upla	**Un film non può durare più di 24h.**
SCONTO_0_100	Dominio	**Lo sconto deve esser compreso tra 0% e 100%.**
MIN_PROIEZIONE_CONCORDE_CON_MIN_FILM	Interrelazionale	**I minuti di proiezione di un film devono essere concordi con la durata del film stesso.**
SALA_PIENA	Interrelazionale	**Non possono essere venduti biglietti per una sala piena.**
PROIEZIONE_NON_POSSIBILE	Intrarelazionale	**Non possono esistere due proiezioni nella stessa sala e nello stesso momento.**
SCONTO_COERENTE_CON_PREZZO_FINALE	Interrelazionale	**Il prezzo finale e lo sconto devono essere coerenti con lo sconto applicato al prezzo della proiezione.**
UNIQUE-PAESE	Intrarelazionale	**Non posso esistere due paesi con lo stesso nome.**

UNIQUE-AUDIO	Intrarelazionale	**Non possono esistere due sistemi audio con lo stesso nome.**
UNIQUE-TECNOLOGIA	Intrarelazionale	**Non possono esistere due tecnologie con lo stesso nome.**
UNIQUE-GENERE	Intrarelazionale	**Non possono esistere due generi con lo stesso nome.**

Progettazione logica

Schema logico

Paesi (*Id_paese*,*paese*)

Sistemi_audio (Id_*sound*,*anno*,*audio*)

Tecnologie_proiezione (*Id_tec*,*anno*,*tecnologia*)

Generi (*Id_genere*,*genere*)

Attori (*Id_attore*,*nominativo*,*paese_di_origine*,*data_nascita*)

Chiavi esterne: *paese_di_origine* → Paesi.Id_paese

Registi (*Id_regista*,*nominativo*,*paese_di_origine*,*data_nascita*)

Chiavi esterne: *paese_di_origine* → Paesi.Id_paese

Film (*Id_film*,*titolo*,*anno_produzione*,*id_regista*,*durata_minuti*,*url_poster*)

Chiavi esterne: Id_regista → Regista.Id_regista

Attori_film (*Id_attore*,*Id_film*)

Chiavi esterne: Id_attore → Attori.Id_attore
Id_film → Film.Id_film

Genere_film (*Id_genere*,*Id_film*)

Chiavi esterne: Id_genere → Generi.Id_ genere
Id_film → Film.Id_film

Sale (*Id_sala*,*descrizione*,*sistema_audio*,*tecnologia_proiezione*,*posti*)

Chiavi esterne: *sistema_audio* → Sistemi_audio.Id_sound
tecnologia_proiezione → Tecnologie_proiezione.Id_tec

Proiezioni (*Id_proiezione*,*Id_sala*,*Id_film*,*prezzo*,*inizio_proiezione*,*fine_proiezione*)

Chiavi esterne: Id_sala → Sale.Id_sala
Id_film → Film.Id_film

Biglietti (*Id_proiezione*,*Id_biglietto*,*sconto*,*prezzo_finale*)

Chiavi esterne: Id_proiezione → Biglietti.Id_proiezione

Traduzione delle associazioni

Associazione	***Implementazione***
Ha dato origine(attori)	Chiave esterna in Attori → Paesi
Ha dato origine(registi)	Chiave esterna in Registi → Paesi
Ha scritto	Chiave esterna in Film → Registi
Viene proiettato	Chiave esterna in Proiezioni → Film
Ha venduto	Chiave esterna in Biglietti → Proiezioni
Avviene	Chiave esterna in Proiezioni → Sale
Ha(sale-sistemiaudio)	Chiave esterna in Sale → Sistemi_audio
Ha(sale-tecnologiafilm)	Chiave esterna in Sale → Tecnologia_film
Partecipa	Chiave esterna in Attore_film → Attori
Ha(film-attori)	Chiave esterna in Attore_film → Film
Ha(film-registi)	Chiave esterna in Genere_film → Film
Appartiene	Chiave esterna in Genere_film → Generi

Schema logico

Paesi	(*Id_paese*,*paese*)
Sistemi_audio	(Id_*sound*,*anno*,*audio*)
Tecnologie_proiezione	(*Id_tec*,*anno*,*tecnologia*)
Generi	(*Id_genere*,*genere*)
Attori	(*Id_attore*,*nominativo*,*paese_di_origine*,*data_nascita*)
Registi	(*Id_regista*,*nominativo*,*paese_di_origine*,*data_nascita*)
Film	(*Id_film*,*titolo*,*anno_produzione*,*id_regista*,*durata_minuti*,*url_poster*)
Attori_film	(*Id_attore*,*Id_film*)
Genere_film	(*Id_genere*,*Id_film*)
Sale	(*Id_sala*,*descrizione*,*sistema_audio*,*tecnologia_proiezione*,*posti*)
Proiezioni	(*Id_proiezione*,*Id_sala*,*Id_film*,*prezzo*,*inizio_proiezione*,*fine_proiezione*)
Biglietti	(*Id_proiezione*,*Id_biglietto*,*sconto*,*prezzo_finale*)

Definizioni SQL

Definizioni delle tabelle

Per comodità nelle tabelle è stato usato lo pseudo tipo SERIAL che genera una sequenza di interi per utilizzarli come progressivi unici per le istanze delle tabelle. Inoltre, nella parte finale della definizione SQL verranno inseriti i trigger e le varie view utili ad un futuro applicativo.

Definizione della tabella Attori

```
CREATE TABLE ATTORI(
  ID_ATTORE SERIAL PRIMARY KEY,
  NOMINATIVO VARCHAR(50) NOT NULL,
  PAESE_DI_ORIGINE VARCHAR(30),
  DATA_NASCITA DATE,
CONSTRAINT DATA_NASCITA_CONCORDE CHECK (DATA_NASCITA < CURRENT_DAT
E),
CONSTRAINT SOLO_LETTERE CHECK ((NOMINATIVO ~* '[0-9]')   IS FALSE)
);
```

Definizione della tabella Registi

```
CREATE TABLE REGISTI(
  ID_REGISTA SERIAL PRIMARY KEY,
  NOMINATIVO VARCHAR(50) NOT NULL,
  PAESE_DI_ORIGINE VARCHAR(20) NOT NULL,
  DATA_NASCITA DATE NOT NULL,
CONSTRAINT DATA_NASCITA_CONCORDE CHECK (DATA_NASCITA <   CURRENT_D
ATE),
CONSTRAINT SOLO_LETTERE CHECK ((NOMINATIVO ~* '[0-9]')  IS FALSE)
);
```

Definizione della tabella Paesi

```
CREATE TABLE PAESI(
 ID_PAESE SMALLSERIAL PRIMARY KEY,
 PAESE VARCHAR(60) UNIQUE
);
```

Definizione della tabella Film

```
CREATE TABLE FILM(
  ID_FILM SERIAL PRIMARY KEY,
  TITOLO VARCHAR(100) NOT NULL,
  ANNO_PRODUZIONE VARCHAR(4) NOT NULL,
  ID_REGISTA INTEGER,
  DURATA_MINUTI SMALLINT NOT NULL,
  URL_POSTER VARCHAR(500),
CONSTRAINT CHECKDURATAMAX CHECK (DURATA_MINUTI BETWEEN 1  AND 1440
),
CONSTRAINT CHECKANNO CHECK (CAST(ANNO_PRODUZIONE AS INTEGER) BETWE
EN 1895 AND DATE_PART('YEAR', CURRENT_DATE)),


CONSTRAINT FK_REGISTA FOREIGN KEY(ID_REGISTA) REFERENCES REGISTI(I
D_REGISTA)
);
```

Definizione della tabella Attori_film

```
CREATE TABLE ATTORI_FILM(
  ID_ATTORE INTEGER,
  ID_FILM INTEGER,
CONSTRAINT PK_ATTORI_FILM PRIMARY KEY(ID_ATTORE, ID_FILM),
CONSTRAINT FK_ATTOREFILM FOREIGN KEY(ID_ATTORE) REFERENCE S ATTORI
(ID_ATTORE),
CONSTRAINT FK_FILMATTORE FOREIGN KEY(ID_FILM) REFERENCES FILM(ID_F
ILM)
);
```

Definizione della tabella Generi

```
CREATE TABLE GENERI(
  ID_GENERE SMALLSERIAL PRIMARY KEY,
  GENERE VARCHAR(20) UNIQUE
);
```

Definizione della tabella Genere_film

```
CREATE TABLE GENERE_FILM(
  ID_GENERE INTEGER,
  ID_FILM INTEGER,
CONSTRAINT PK_GENERE_FILM PRIMARY KEY(ID_GENERE, ID_FILM),
CONSTRAINT FK_GENEREFILM FOREIGN KEY(ID_GENERE) REFERENCES GENERI(
ID_GENERE),
CONSTRAINT FK_FILMGENERE FOREIGN KEY(ID_FILM) REFERENCES FILM(ID_F
ILM)
);
```

Definizione della tabella Proiezioni

```
CREATE TABLE PROIEZIONI(
  ID_PROIEZIONE SERIAL PRIMARY KEY,
  ID_SALA INTEGER,
  ID_FILM INTEGER,
  PREZZO DECIMAL NOT NULL,
  INIZIO_PROIEZIONE TIMESTAMP NOT NULL,
  FINE_PROIEZIONE TIMESTAMP NOT NULL,
CONSTRAINT MAGGIORE_ZERO CHECK (PREZZO > 0),
CONSTRAINT FK_SALA FOREIGN KEY(ID_SALA) REFERENCES SALE(ID_SALA),
CONSTRAINT FK_FILM FOREIGN KEY(ID_FILM) REFERENCES FILM(ID_FILM) O
N DELETE CASCADE,
CONSTRAINT ORARI_CONCORDI CHECK (INIZIO_PROIEZIONE < FINE_PROIEZIO
NE),
CONSTRAINT INIZIO_FINE_NO24H CHECK ((DATE_PART('day', FINE_PROIEZI
ONE::TIMESTAMP - INIZIO_PROIEZIONE::TIMESTAMP) * 24 + DATE_PART('h
our', FINE_PROIEZIONE::TIMESTAMP - INIZIO_PROIEZIONE::TIMESTAMP))<
=24)
);
```

Definizione della tabella Biglietti

```
CREATE TABLE BIGLIETTI(
  ID_PROIEZIONE INTEGER,
  ID_BIGLIETTO SERIAL PRIMARY KEY,
  SCONTO SMALLINT,
  PREZZOFINALE DECIMAL NOT NULL,
CONSTRAINT SCONTO_0_100 CHECK (SCONTO BETWEEN 0 AND 100), CONSTRAI
NT FK_PROIEZIONE FOREIGN KEY(ID_PROIEZIONE) REFERENCES PROIEZIONI(
ID_PROIEZIONE) ON DELETE CASCADE
);
```

Definizione della tabella Sale

```
CREATE TABLE SALE(
  ID_SALA SERIAL PRIMARY KEY,
  DESCRIZIONE VARCHAR(20) NOT NULL,
  SISTEMA_AUDIO INTEGER,
  TECNOLOGIA_PROIEZIONE INTEGER,
  POSTI SMALLINT NOT NULL,
CONSTRAINT CHECKPOSTIMAX CHECK (POSTI BETWEEN 1 AND 500)
);
```

Definizione della tabella Sistemi_audio

```
CREATE TABLE SISTEMI_AUDIO(
  ID_SOUND SMALLSERIAL PRIMARY KEY,
  ANNO VARCHAR(4),
  AUDIO VARCHAR(50) UNIQUE
);
```

Definizione della tabella Tecnologie_proiezione

```
CREATE TABLE TECNOLOGIE_PROIEZIONE(
  ID_TEC SMALLSERIAL PRIMARY KEY,
  ANNO VARCHAR(4),
  TECNOLOGIA VARCHAR(50) UNIQUE
);
```

Definizione delle view

Le view definite servono rispettivamente a permettere di stimare quali sono le fasce orarie di maggi or affluenza, gli spettacoli più remunerativi e le sale maggiormente occupate durante gli orari di ma ggior affluenza.

```
CREATE VIEW SPETTACOLI_RENUMERATIVI AS
  (SELECT
     (SELECT F.TITOLO
      FROM FILM F
      WHERE F.ID_FILM=P.ID_FILM)FILM,
          COUNT(B.ID_BIGLIETTO)BIGLIETTI_VENDUTI,
          SUM(B.PREZZOFINALE)GUADAGNO_TOTALE
   FROM BIGLIETTI B
   JOIN PROIEZIONI P ON B.ID_PROIEZIONE=P.ID_PROIEZIONE
   GROUP BY FILM
   ORDER BY BIGLIETTI_VENDUTI,GUADAGNO_TOTALE
);
CREATE VIEW ORARI_MAGGIORE_AFFLUENZA AS
  (SELECT COUNT(*)AFFLUENZA,
         CAST(INIZIO_PROIEZIONE AS TIME)FASCIAINIZIO,
         CAST(FINE_PROIEZIONE AS TIME)FASCIAFINE
   FROM BIGLIETTI B
   JOIN PROIEZIONI P ON B.ID_PROIEZIONE=P.ID_PROIEZIONE
   GROUP BY FASCIAINIZIO, FASCIAFINE
   ORDER BY AFFLUENZA DESC
);

CREATE VIEW AFFLUENZA_SALE_ORARI_MAX AS
  (SELECT COUNT(*)CONTA,
          P.ID_SALA,
          CAST(P.INIZIO_PROIEZIONE AS TIME)FASCIAINIZIO,
          CAST(P.FINE_PROIEZIONE AS TIME)FASCIAFINE
   FROM BIGLIETTI B
   JOIN PROIEZIONI P ON B.ID_PROIEZIONE=P.ID_PROIEZIONE
```

```
   GROUP BY 2,3,4
   HAVING CAST(P.INIZIO_PROIEZIONE AS TIME)||'-'||
   CAST(P.FINE_PROIEZIONE AS TIME) IN (SELECT O.FASCIAINIZIO||'-'
   ||O.FASCIAFINE FROM ORARI_MAGGIORE_AFFLUENZA O
   WHERE O.AFFLUENZA=(SELECT MAX(O.AFFLUENZA)
   FROM ORARI_MAGGIORE_AFFLUENZA O))
   ORDER BY CONTA DESC
);
```

Definizione dei trigger

Questi sono i vincoli interrelazionale mostrati nel paragrafo vincoli.

```
CREATE OR REPLACE FUNCTION CONTROLLO_SALA_PIENA() RETURNS TRIGGER
AS $$
DECLARE
   POSTI_SALA INT;
   BIGLIETTI_VENDUTI INT;
BEGIN
SELECT S.POSTI INTO POSTI_SALA FROM SALE S WHERE S.ID_SALA=(SELECT
P.ID_SALA FROM PROIEZIONI P WHERE P.ID_PROIEZIONE=NEW.ID_PROIEZION
E);
SELECT COUNT(*) INTO BIGLIETTI_VENDUTI FROM BIGLIETTI B WHERE
      B.ID_PROIEZIONE=NEW.ID_PROIEZIONE;
      IF (POSTI_SALA = BIGLIETTI_VENDUTI) THEN
            RAISE EXCEPTION 'SALA PIENA';
      END IF;
      RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
```

```
CREATE OR REPLACE FUNCTION CONTROLLO_SALA_FILM_ORARIO() RETURNS TR
IGGER AS $$

DECLARE

   CONTA INT := 0;

   RIGA PROIEZIONI%ROWTYPE;

BEGIN

       FOR RIGA IN SELECT * FROM PROIEZIONI P

       WHERE P.ID_SALA=NEW.ID_SALA AND

       (P.INIZIO_PROIEZIONE::DATE IN (P.INIZIO_PROIEZIONE::DATE,P.F
INE_PROIEZIONE::DATE) OR

       P.FINE_PROIEZIONE::DATE IN (P.INIZIO_PROIEZIONE::DATE,P.FINE
_PROIEZIONE::DATE))

       LOOP

  IF((NEW.INIZIO_PROIEZIONE BETWEEN RIGA.INIZIO_PROIEZIONE AND RIG
A.FINE_PROIEZIONE)

  OR (NEW.FINE_PROIEZIONE BETWEEN RIGA.INIZIO_PROIEZIONE AND RIGA.
FINE_PROIEZIONE)) THEN

                    CONTA := CONTA + 1;

                  END IF;

       EXIT WHEN CONTA > 0;

       END LOOP;

       IF(CONTA > 0) THEN

             RAISE EXCEPTION 'SALA GIÀ OCCUPATA PER LA DATA E L''OR
ARIO INSERITO';

       END IF;

       RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;
```

```
CREATE OR REPLACE FUNCTION CONTROLLO_PROIEZIONE_DURATA_FILM() RETU
RNS TRIGGER AS $$

DECLARE

   MINUTI_FILM INT := 0;

BEGIN

      SELECT F.DURATA_MINUTI INTO MINUTI_FILM FROM FILM F WHERE F.
ID_FILM=NEW.ID_FILM;

      IF(((DATE_PART('DAY', CAST(NEW.FINE_PROIEZIONE AS timestamp
) - CAST(NEW.INIZIO_PROIEZIONE AS timestamp )) * 24 +

               DATE_PART('HOUR', CAST(NEW.FINE_PROIEZIONE AS times
tamp ) - CAST(NEW.INIZIO_PROIEZIONE AS timestamp ))) * 60 +

               DATE_PART('MINUTE', CAST(NEW.FINE_PROIEZIONE AS tim
estamp ) - CAST(NEW.INIZIO_PROIEZIONE AS timestamp ))) < MINUTI_FI
LM) THEN

            RAISE EXCEPTION 'LA DURATA DEL FILM É MAGGIORE RISPETT
O ALLA DURATA DELLA PROIEZIONE ';

      END IF;

      RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;


/*CONTROLLO BIGLIETTI*/

CREATE OR REPLACE FUNCTION CONTROLLO_SCONTO_PREZZO_FINALE() RETURN
S TRIGGER AS $$

DECLARE

   PREZZO_PROIEZIONE INT := 0;

BEGIN

      SELECT P.PREZZO INTO PREZZO_PROIEZIONE FROM PROIEZIONI P WHE
RE P.ID_PROIEZIONE=NEW.ID_PROIEZIONE;

      IF(PREZZO_PROIEZIONE-(PREZZO_PROIEZIONE*NEW.SCONTO)/100 <> N
EW.PREZZOFINALE) THEN

            RAISE EXCEPTION 'PREZZOFINALE BIGLIETTO NON COERENTE C
ON SCONTO APPLICATO';

      END IF;

      RETURN NEW;
```

```
END;
$$ LANGUAGE PLPGSQL;



CREATE TRIGGER MIN_PROIEZIONE_CONCORDE_CON_MIN_FILM
BEFORE
INSERT ON PROIEZIONI
FOR EACH ROW EXECUTE FUNCTION CONTROLLO_PROIEZIONE_DURATA_FILM();

CREATE TRIGGER SALA_PIENA
BEFORE
INSERT ON BIGLIETTI
FOR EACH ROW EXECUTE FUNCTION CONTROLLO_SALA_PIENA();

CREATE TRIGGER PROIEZIONE_NON_POSSIBILE
BEFORE
INSERT ON PROIEZIONI
FOR EACH ROW EXECUTE FUNCTION CONTROLLO_SALA_FILM_ORARIO();

CREATE TRIGGER SCONTO_COERENTE_CON_PREZZO_FINALE
BEFORE
INSERT ON BIGLIETTI
FOR EACH ROW EXECUTE FUNCTION CONTROLLO_SCONTO_PREZZO_FINALE();
```