Descrizione dell'ambiente applicativo

Si suppone di dover realizzare un database per la gestione di un cinema. Lo scopo di tale database è quello di gestire i film durante la settimana/e di proiezione. Il database dovrà contenere la copia digitale dei film, eventualmente può contenere anche una copia originale del film permettendo così di riprodurlo nel linguaggio nativo (nel nostro cinema non c'è una particolare regola, ma supporta sia una cadenza settimanale sia occasioni rare). Il cinema offre anche una migliore esperienza audio grazie al Dolby Atmos, che permette un'immersione totale nella storia; questa caratteristica fa parte di una serie di attributi caratterizzanti i film in proiezione che possono essere sia in formato 2D che 3D, per permettere una differenziazione del prezzo del biglietto.
L'organizzazione delle singole serate del cinema è permessa grazie al assegnamento di sale a proiezioni di pellicole. Il database nasce per permettere di organizzare facilmente la programmazione del cinema e inoltre permette di quantificare gli incassi sia di una singola proiezione che di tutto il cinema in un determinato periodo. Esso offre inoltre altre possibilità: il cliente prenotare telefonicamente e acquistare il biglietto fisicamente solo al botteghino (maggiori dettagli nella sezione Biglietto).
L'ultima macrosezione si occupa dell'interazione che il Cinema ha con case di produzione e sponsor: vengono registrate le spese per il noleggio delle pellicole e gli incassi derivanti da sponsor acquirenti di spazi pubblicitari.

Di seguito vengono riportate le descrizioni delle entità dello schema:

Cliente

Il cliente può scegliere tra comprare il biglietto o prenotarlo. Nel caso in cui decidesse di prenotare il biglietto, questo non significherebbe acquistarlo in anticipo ma il ritirarlo al botteghino. La prenotazione avviene tramite nome (chiave secondaria) e cellulare(chiave primaria).

Proiezione

Un problema riscontrato durante la realizzazione dello schema ER è che l'entità film aveva una notevole quantità di attributi. Inoltre se dovessimo proiettare lo stesso film in più sale, risulterebbe difficile risalire alla singola proiezione. Per questi motivi è nata la necessità di creare l'entità "Proiezione": questa mantiene l'attributo chiave di "Film" tramite chiave esterna ma prende anche la chiave di "Sala", in modo da risolvere l'ambiguità nel caso in cui fosse necessario proiettare un film in più di una sala. Per aggiungere alla unicità di ogni singola proiezione abbiamo messo nella chiave composta anche "Data" e "Ora", in questo modo abbiamo un entità univoca. Gli attributi "Biglietti Disponibili" e "Biglietti Venduti" sono inizializzati e aggiornati tramite triggers sui biglietti.

Biglietto

Il biglietto deve ovviamente contenere tutti i dati precedentemente assegnati a "Proiezione", perché ognuno di questi andrebbe anche stampato sul biglietto fisico. Un biglietto può essere non disponibile e non venduto, grazie all'entità "Prenotazione"(che come precedentemente specificato, non implica l'acquisto immediato del biglietto). Il selettore ci permette, insieme al cellulare, di avere info sul biglietti:

- selettore=1 && cellulare <> NULL, prenotato.
- selettore=0 && cellulare = NULL, disponibile.
- selettore=1 && cellulare = NULL, venduto.

Film

L'entità "Film" differisce dall'entità "Proiezione" in quanto racchiude tutte le informazioni riguardanti le caratteristiche tecniche della pellicola. Questa entità viene maggiormente descritta grazie alla "Scheda" ed essendo un'entità diversa da "Proiezione" ci permette di tenere separati a livello progettuale la pellicola che il cinema affitta dalla casa produttrice dalla proiezione di questa, risolvendo dunque numerosi problemi.

Scheda

La scheda è di appoggio al film: contiene tutti quegli attributi che danno info tecniche sulla pellicola, dalla sua trama presente anche sulla locandina ad informazione sull'Audio Quality e Video Quality.

Sponsor

Lo sponsor invece è rappresentato da una Partita Iva e ha acquisito uno spazio pubblicitario al fine di pubblicizzare la propria Ditta.

Prenotazione

Un cliente si registra nel DB effettuando una prenotazione, ovvero inserendo il suo nome e il suo numero di cellulare. Questo, compare sul biglietto prenotato permettendoci di trovarlo facilmente con un'interrogazione adeguata.

Proiezione

La proiezione è il singolo evento in cui si assiste al film, quindi in un determinato giorno ad una certa ora in una specifica sala. La creazione di questa entità ha semplificato molto la realizzazione del progetto logico e la gestione delle chiavi esterne.

Sala

Identificata da una lettera, indica dove si tiene la proiezione.

Pubblicità

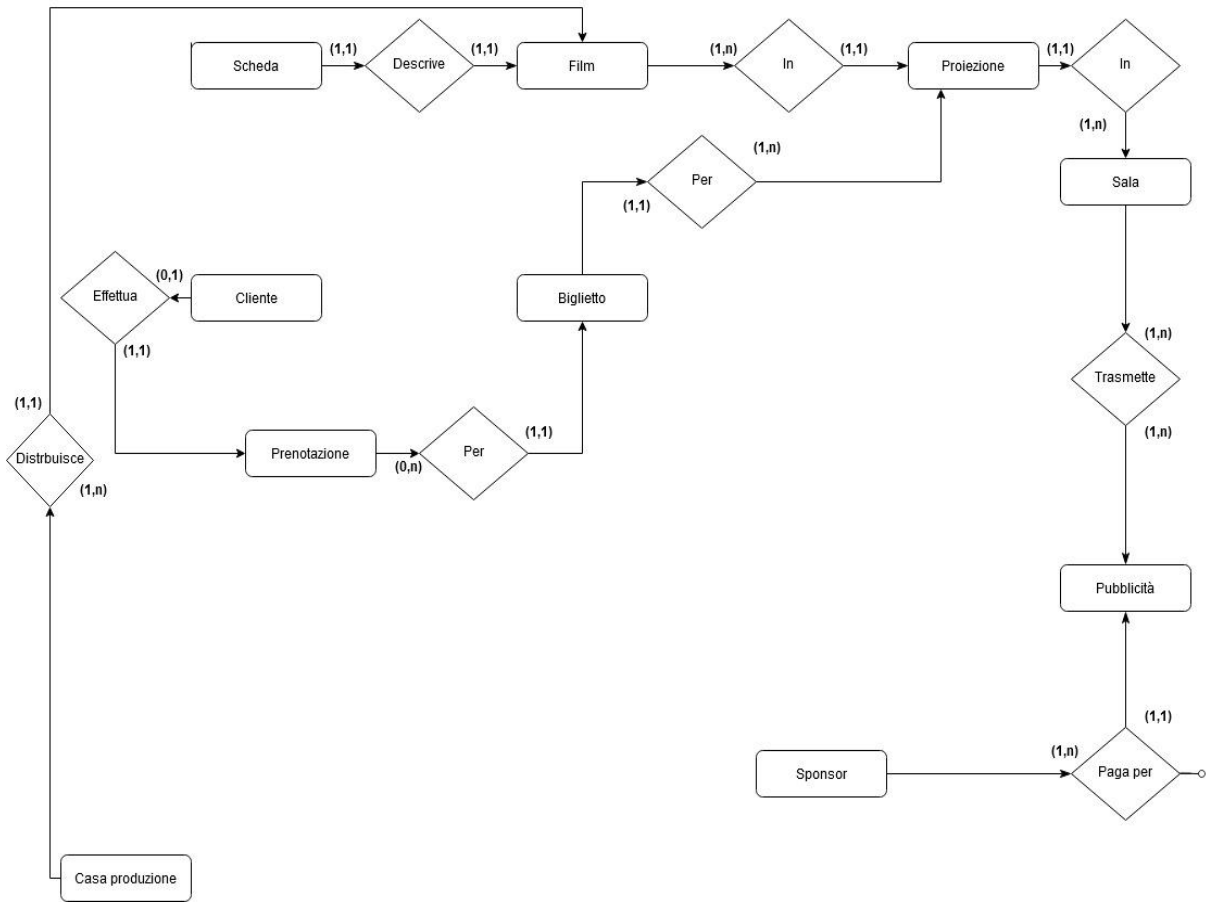
Comprata da uno sponsor, è un'altra fonte di incasso per le casse del cinema. La si identifica tramite un ID casuale generato.

Glossario

TERMINE	ATTRIBUTI	SINONIMI	RELAZIONE CON
Cliente	Cellulare Nome		Prenotazione
Prenotazione	ID prenotazione N Biglietti		Cliente Biglietto
Scheda	ID Scheda AQ VQ Titolo Descrizione Prima/Ultima Visione		Film
Film	Titolo	Pellicola	Casa di produzione Scheda Proiezione
Biglietto	N. posto ID. prenotazione	Ticket	Prenotazione Proiezione
Sponsor	P. IVA Nome		Pubblicità
Proiezione	ID proiezione Data Ora Nomesala		Biglietto Sala Film
Pubblicità	ID Pubblicità		Sponsor Sala
Sala	Capienza Nome Sala		Pubblicità Proiezione
Casa di Produzione	IBAN Nome Prezzo		Film

Schema scheletro

Ecco un’illustrazione di come le entitá sono tra loro relazionate con tutte le rispettive cardinalitá:



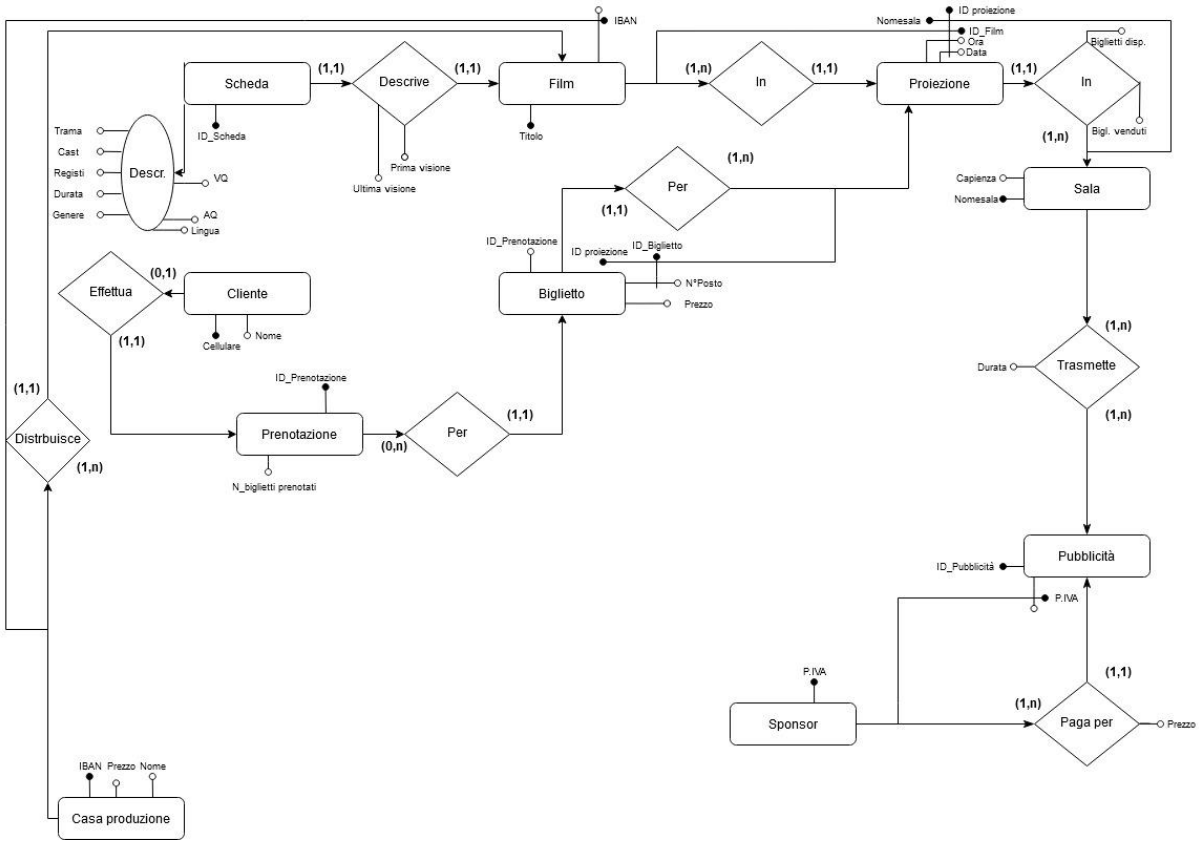
Schema ER

DESCRIZIONE:

La “Casa Produzione” distribuisce uno o molteplici “Film” i quali sono descritti da una “Scheda”. I “Film” o pellicole saranno in “Proiezione” in una “Sala” . È possibile accedere ad una “Proiezione” grazie ad un “Biglietto” che un “Cliente” può prenotare registrando una “Prenotazione”. Infine, uno “Sponsor” compra uno spazio pubblicitario o “Pubblicità”. Quest’ultima verrà quindi trasmessa in una “Sala”. Tutto questo si trasforma in uno schema semplice che permette di monitorare le operazioni organizzative per la vendita dei biglietti e soprattutto (obiettivo finale del progetto) permette di registrare le entrate e le uscite economiche del cinema.

Il tutto si traduce in un diagramma E-R riportato nella pagina seguente;

Diagramma ER completo



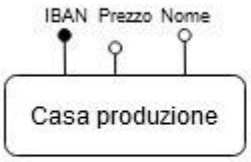
Segue dunque la progettazione logica.

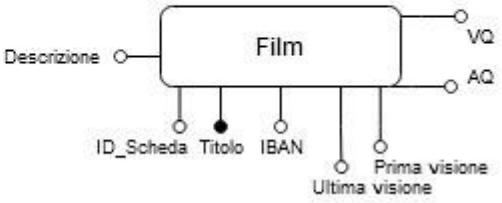
Progettazione logica

Una volta illustrato lo schema ER possiamo addentrarci nella progettazione logica che richiede dunque l'analisi di più aspetti:

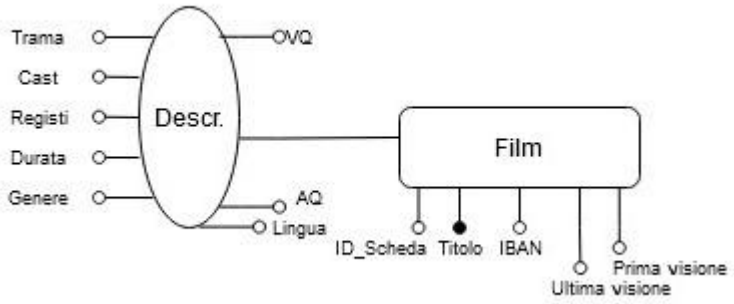
Selezione delle chiavi primarie ed eliminazione delle chiavi esterne

- Casa Produzione: "IBAN" identificativo della C.P..
- Film: "Titolo". Chiave esterna "IBAN" di C.P..

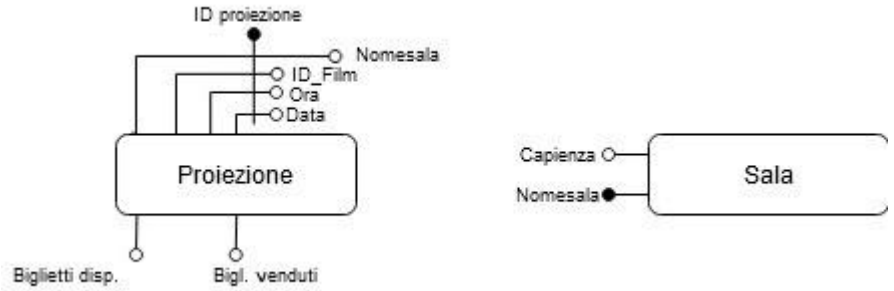




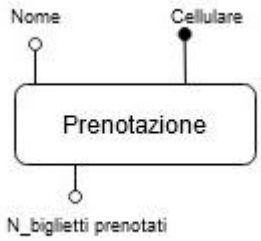
- Scheda: ID_SCHEDA generato automaticamente, diventa parte di FILM.



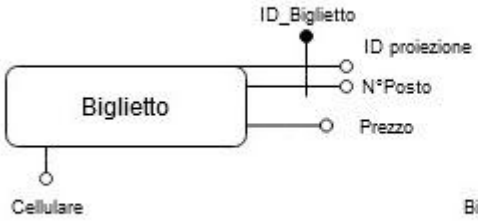
- Sala: <u>NOMESALA</u>
- Proiezione: <u>ID_PROIEZIONE</u> composto da Data, Ora, e dalle chiavi esterne “NomeSala” e “ID_Film”.



- Cliente: “<u>CELLULARE</u>” .
- Prenotazione: “Cellulare” in chiave esterna da CLIENTE vista la cardinalità (0,1).



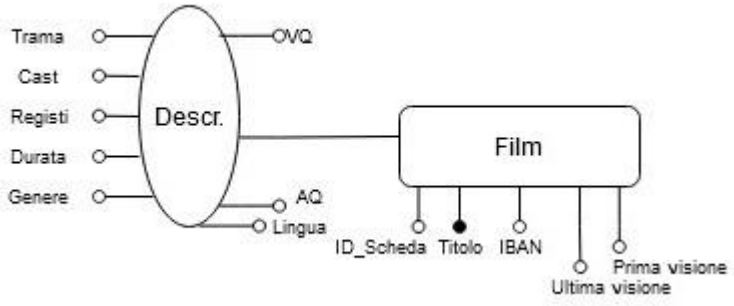
- Biglietto: “ID_BIGLIETTO” composto da “ID_Proiezione”, che è chiave esterna di proiezione e da “N_posto”. Inoltre è presente una chiave esterna atta a identificare una eventuale prenotazione sul biglietto, ovvero “CELLULARE”.



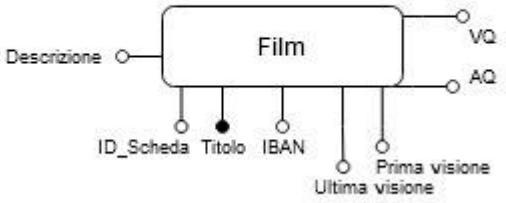
- Sponsor: “P.IVA” della ditta che compra lo spazio pubblicitario.
- Pubblicità: “ID_PUBBLICITÁ” generato automaticamente. Chiave esterna , “P.IVA” di Sponsor.

Trasformazione degli attributi composti

Nello schema si presenta un solo attributo composto, ovvero “Descrizione” il quale contiene un insieme di informazioni riguardanti le caratteristiche del film:

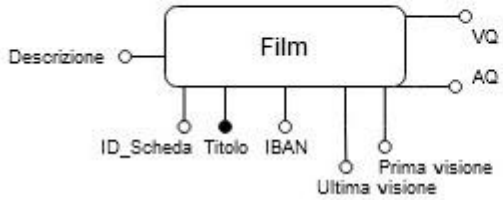


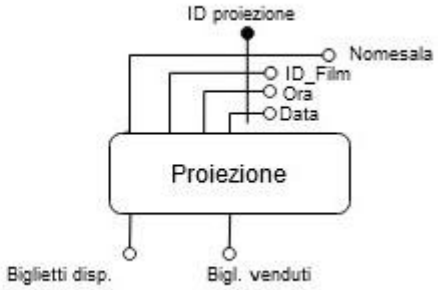
Questo attributo, non di fondamentale importanza per il nostro database, può benissimo essere riportato come singolo attributo di tipo “Descrizione”.



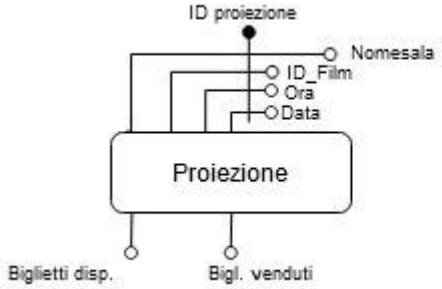
Creazione di schemi di relazioni partendo dall'entità e dalle rispettive associazioni

Film(<u>TITOLO</u>)
Proiezione(<u>DATA, ORA, NOMESALA, TITOLO</u>)
FK: TITOLO REFERENCES FILM
FK: NOMESALA REFERENCES SALA





Proiezione(<u>DATA, ORA, NOMESALA, TITOLO</u>)
FK: TITOLO REFERENCES FILM
FK: NOMESALA REFERENCES SALA
Sala(CAPIENZA, <u>NOMESALA</u>)



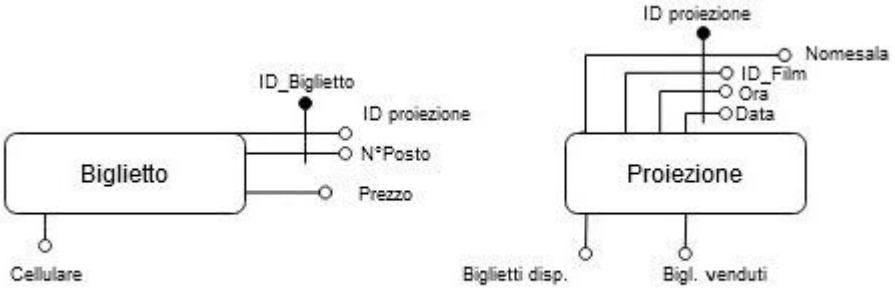
Biglietto(N.POSTO, ID_PROIEZIONE)

FK: ID PROIEZIONE REFERENCES PROIEZIONE

Proiezione(DATA, ORA, NOMESALA, TITOLO)

FK: TITOLO REFERENCES FILM

FK: NOMESALA REFERENCES SALA

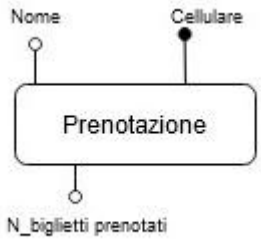


Cliente(CELLULARE, NOME)

Prenotazione(CELLULARE, N_BIGLIETTI PRENOTATI)

FK: CELLULARE REFERENCES CLIENTE

L'associazione binaria con cardinalità (0,1) per Cliente e (1,1) per Prenotazione ci permette di inglobare le informazioni presenti in Prenotazione in un'unica entità Cliente; essa si dimostra vincolante al fine dell'esistenza dell'associazione.

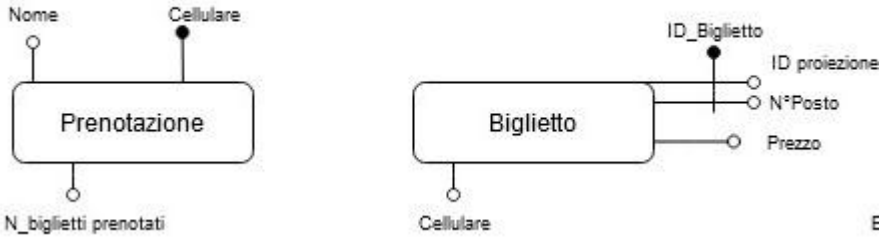


Prenotazione(CELLULARE, N_BIGLIETTI PRENOTATI)

FK: CELLULARE REFERENCES CLIENTE

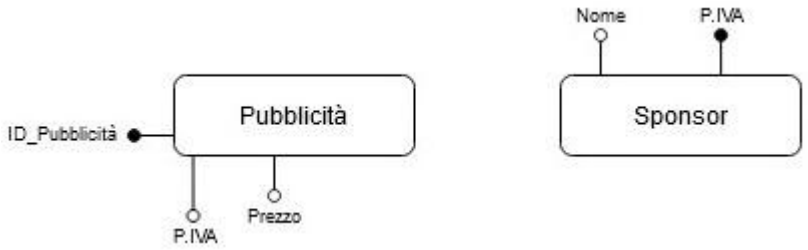
Biglietto (ID_PROIEZIONE, N.POSTO, N.PRENOTAZIONE, BIGLIETTI DISP)

FK: ID PROIEZIONE REFERENCES PROIEZIONE



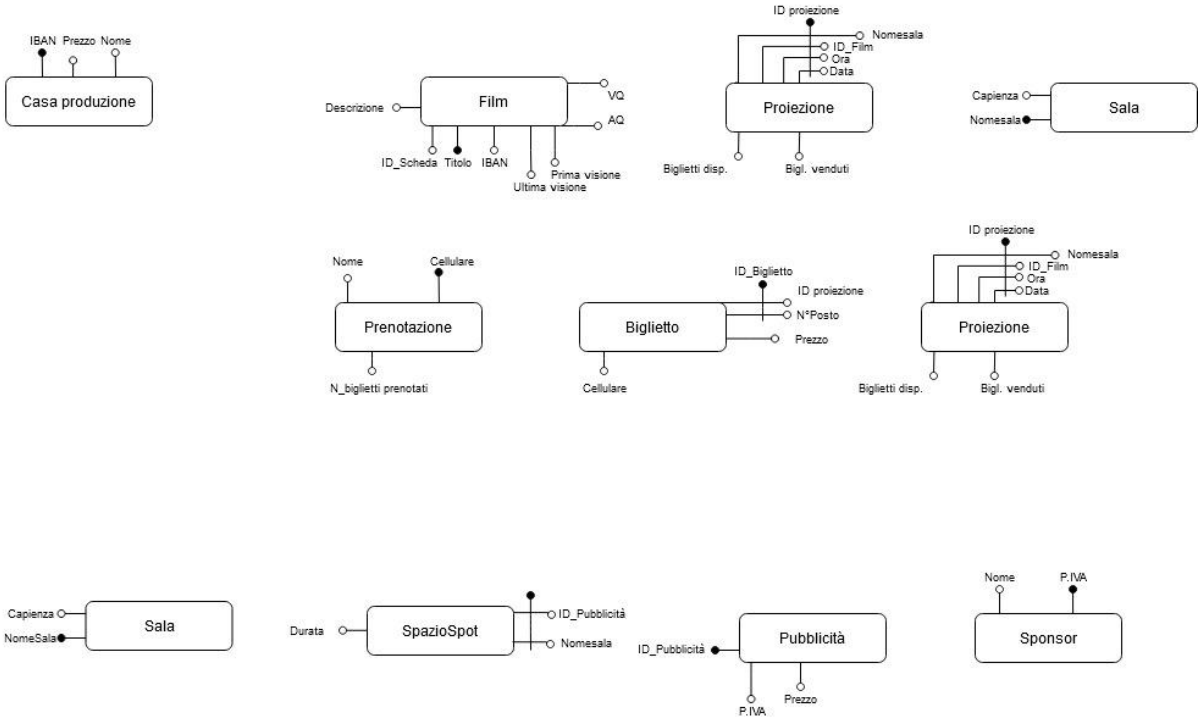
Sala(NOMESALA)
Pubblicità(ID_PUBBLICITÀ)
Si crea una entità SpazioSpot:(ID_PUBBLICITA, NOMESALA)

Sponsor(P.IVA)
Pubblicità(ID_PUBBLICITÀ)



Casa Produzione(IBAN)
Film(TITOLO)

In breve:



Ecco come appare lo schema con solo le entità ed eliminando tutte le relazioni.

SQL

```
CREATE TABLE CASAPRODUZIONE(
        Nome VARCHAR(20)  UNIQUE NOT NULL,
        IBAN CHAR(27)  NOT NULL,
        PRIMARY KEY (NOME)
);

CREATE TABLE FILM(
        NOME VARCHAR(20) NOT NULL,
        TITOLO VARCHAR (20) NOT NULL,
        AQ CHAR(5),
        VQ CHAR(2),
        LINGUA VARCHAR(20),
        PrimaVisione DATE,
        UltimaVisione DATE,
        DESCRIZIONE VARCHAR(350)  NOT NULL,
        PRIMARY KEY(TITOLO),
        FOREIGN KEY(NOME) REFERENCES CASAPRODUZIONE
        ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE SALA(
        NomeSala CHAR(1) PRIMARY KEY NOT NULL,
        Capienza SMALLINT NOT NULL DEFAULT 300
);

CREATE TABLE PROIEZIONE(
        TITOLO VARCHAR (20) NOT NULL,
        DATA  DATE,
        ORA TIME,
        NomeSala CHAR(1) NOT NULL,
        PREZZO SMALLINT DEFAULT NULL,
        LastMod TIMESTAMP without time zone DEFAULT
                CURRENT_TIMESTAMP,
        BigliettiVenduti  SMALLINT  NOT NULL DEFAULT  0,
```

```
        BigliettiDisponibili SMALLINT  NOT NULL DEFAULT -1,
        FOREIGN KEY(TITOLO) REFERENCES FILM
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(NOMESALA) REFERENCES SALA
        ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY(TITOLO,DATA,ORA,NOMESALA),
        CHECK (BigliettiDisponibili >=-1),
        CHECK  (BigliettiVenduti>=0)
);

CREATE TABLE SPONSOR(
        NOME VARCHAR(10) NOT NULL,
    PIVA CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY (NOME)
);

CREATE TABLE PUBBLICITA(
        NOME VARCHAR(10),
        ID_Pubblicita SMALLINT,
        PREZZO SMALLINT,
        PRIMARY KEY (ID_pubblicita),
        FOREIGN KEY (NOME) REFERENCES SPONSOR
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE SPAZIOSPOT(
        DURATA SMALLINT NOT NULL,
        ID_Pubblicita SMALLINT,
        NomeSala Char(1),
        FOREIGN KEY (NomeSala) REFERENCES SALA
        ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (ID_Pubblicita) REFERENCES PUBBLICITA
        ON UPDATE CASCADE ON DELETE CASCADE,
        PRIMARY KEY (ID_Pubblicita, NomeSala)
);

CREATE TABLE PRENOTAZIONE(
```

```
        NOME VARCHAR(10),
        CELLULARE CHAR(10) PRIMARY KEY NOT NULL,
        N_BIGLIETTI SMALLINT,
        CHECK ((N_BIGLIETTI) >=1 AND (N_BIGLIETTI <=8))
);

CREATE TABLE BIGLIETTO(
        Data DATE,
        Ora TIME,
        NomeSala CHAR (1),
        TITOLO VARCHAR(20)  NOT NULL,
        N_posto SMALLINT,
        CELLULARE CHAR(10) DEFAULT NULL,
        FOREIGN KEY (CELLULARE) REFERENCES PRENOTAZIONE
        ON UPDATE CASCADE ON DELETE SET NULL,
        FOREIGN KEY(TITOLO,DATA,ORA, NOMESALA) REFERENCES
         PROIEZIONE
        ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY (TITOLO, DATA, ORA, N_POSTO, NOMESALA),
Selettore SMALLINT DEFAULT -1,
CHECK  ((selettore >-2) and (selettore <2))
);
```

QUERY DI INSERIMENTO:

CASA DI PRODUZIONE

```
INSERT INTO CASAPRODUZIONE (NOME, IBAN) VALUES
('Arco films', 'IT28T0300203280473529811357');
INSERT INTO CASAPRODUZIONE (NOME, IBAN) VALUES
('Kimerafilm', 'IT62A0300203280974276124636');
INSERT INTO CASAPRODUZIONE (NOME, IBAN) VALUES
('Fauno film', 'IT38Z0300203280477146167452');
```

FILM

```
INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Arco films', 'UP', 'norm','no', 'italiano', '2020-08-15', '2020-08-22','[...]');
```

```
INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Arco films','Deadpool', 'atmos', '2D', 'Italiano','2020-08-15','2020-08-22','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Arco films','Avengers: Endgame', 'atmos', '3D', 'Italiano','2020-08-06','2020-08-13','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Arco films','Bao', 'norm', '2D', 'Inglese','2020-08-06','2020-08-13','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Fauno film','Cars', 'atmos', '3D', 'Italiano','2020-08-06','2020-08-13','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Fauno film','Pets', 'norm', '2D', 'Italiano','2020-08-24','2020-08-31','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Fauno film','Bolt', 'atmos', '3D', 'Inglese','2020-08-24','2020-08-31','[...]');

INSERT INTO FILM (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE, ULTIMAVISIONE,
DESCRIZIONE) VALUES
('Fauno film','Psycho','norm','no','italiano','2020-08-24','2020-08-31','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Kimerafilm','Bad boys','atmos','no','italiano','2020-08-09','2020-08-16','[...]');

INSERT INTO FILM  (NOME, TITOLO, AQ, VQ, LINGUA, PRIMAVISIONE,
ULTIMAVISIONE, DESCRIZIONE) VALUES
('Kimerafilm','THOR: Ragnarok','atmos','no','italiano','2020-08-15','2020-08-22','[...]');
```

SALA

```
INSERT INTO SALA (NOMESALA) VALUES('a');
INSERT INTO SALA (NOMESALA) VALUES('b');
INSERT INTO SALA (NOMESALA) VALUES('c');
INSERT INTO SALA (NOMESALA) VALUES('d');
INSERT INTO SALA (NOMESALA) VALUES('e');
```

SPONSOR

```
INSERT INTO SPONSOR (NOME, PIVA) VALUES ('Legnai', '0000000001');
INSERT INTO SPONSOR (NOME, PIVA) VALUES ('Imbiank', '0000000002');
INSERT INTO SPONSOR (NOME, PIVA) VALUES ('Arzi', '0000000003');
```

PUBBLICITA

```
INSERT INTO PUBBLICITA (NOME, ID_Pubblicita, PREZZO) VALUES('Imbiank', 30, 120);
INSERT INTO PUBBLICITA (NOME, ID_Pubblicita, PREZZO) VALUES('Legnai', 910, 30);
INSERT INTO PUBBLICITA (NOME, ID_Pubblicita, PREZZO) VALUES('Imbiank', 264,  60);
INSERT INTO PUBBLICITA (NOME, ID_Pubblicita, PREZZO) VALUES('Legnai', 754, 50);
INSERT INTO PUBBLICITA (NOME, ID_Pubblicita, PREZZO) VALUES('Arzi', 423, 150);
```

SPAZIOSPOT

```
INSERT INTO SPAZIOSPOT (DURATA, ID_PUBBLICITA, NOMESALA) VALUES( 10, 30
,'a');
INSERT INTO SPAZIOSPOT (DURATA, ID_PUBBLICITA, NOMESALA) VALUES( 15, 910
,'b');
INSERT INTO SPAZIOSPOT (DURATA, ID_PUBBLICITA, NOMESALA) VALUES( 30, 264
,'e');
INSERT INTO SPAZIOSPOT (DURATA, ID_PUBBLICITA, NOMESALA) VALUES( 20, 754
,'d');
INSERT INTO SPAZIOSPOT (DURATA, ID_PUBBLICITA, NOMESALA) VALUES( 15, 423
,'c');
```

PROIEZIONE

```
INSERT INTO PROIEZIONE (TITOLO,DATA,ORA,NOMESALA)
VALUES('UP','2020-08-17','16:00:00','a');
INSERT INTO PROIEZIONE (TITOLO,DATA,ORA,NOMESALA)
VALUES('Deadpool','2020-08-21','19:00:00','b');
INSERT INTO PROIEZIONE (TITOLO,DATA,ORA,NOMESALA) VALUES('Avengers:
Endgame','2020-08-07','21:00:00','c');
```

PRENOTAZIONE

```
INSERT INTO PRENOTAZIONE (NOME, CELLULARE, N_BIGLIETTI) VALUES ('SARA',
'3331234567', 3);
INSERT INTO PRENOTAZIONE (NOME, CELLULARE, N_BIGLIETTI) VALUES ('LELLO',
'3471234567', 2);
INSERT INTO PRENOTAZIONE (NOME, CELLULARE, N_BIGLIETTI) VALUES
('GIOVANNI', '3311234567', 4);
```

TRIGGERS
//TRIGGER CHE GENERA I BIGLIETTI una volta inserita una proiezione:
Tramite un ciclo di 300 iterazioni si creano i 300 biglietti per ogni nuova proiezione inserita, vengono in aiuto il flag=-1 e la colonna lastmod.

```
CREATE OR REPLACE FUNCTION public.generabiglietti()
        RETURNS trigger
        LANGUAGE 'plpgsql'

AS $BODY$
DECLARE
i INTEGER := 1;
BEGIN
  while i < 301 loop

INSERT INTO biglietto (n_posto, titolo, data, ora, nomesala, selettore)
SELECT        i, titolo,  data, ora, nomesala, 0
        FROM proiezione
        WHERE bigliettidisponibili=-1
        ORDER BY lastmod ASC
        LIMIT 1;

        i := i+1;
  end loop;

        UPDATE  proiezione
        SET bigliettidisponibili=300, lastmod=current_timestamp
        WHERE lastmod = (SELECT lastmod from proiezione WHERE bigliettidisponibili=-1
        ORDER BY lastmod ASC
        LIMIT 1);
        return new;

END;
$BODY$;

CREATE TRIGGER trigger_oncreate
        AFTER INSERT
        ON public.proiezione
        FOR EACH STATEMENT
        EXECUTE PROCEDURE public.generabiglietti();
```

//TRIGGER ON UPDATE SU <u>CELLULARE</u> DI BIGLIETTO CHE SETTA INDISPONIBILE(selettore=1) SE LA COLONNA CELLULARE E' DIVERSA DA NULL
Come precedentemente spiegato, un biglietto con il numero di cellulare presente è prenotato, di conseguenza dev'essere messo indisponibile.

```
CREATE FUNCTION public.prenotabiglietti()
```

```
        RETURNS trigger
        LANGUAGE 'plpgsql'
        COST 100
        VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
IF (EXISTS(SELECT CELLULARE FROM BIGLIETTO WHERE CELLULARE IS NOT
NULL)) THEN
         UPDATE BIGLIETTO SET SELETTORE = 1
                WHERE CELLULARE IS NOT NULL;
   END IF;
         RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.prenotabiglietti()
        OWNER TO postgres;

CREATE TRIGGER trigger_onupdate
AFTER UPDATE OF CELLULARE ON BIGLIETTO
FOR EACH ROW
EXECUTE PROCEDURE prenotabiglietti();
```

//TRIGGER ONCREATE SU BIGLIETTO CHE SETTA IL PREZZO UNA VOLTA SOLA SU PROIEZIONE IN BASE ALLE SUE INFO.
Per evitare ridondanza, segnamo il prezzo del singolo biglietto una sola volta su Proiezione calcolandolo in base alla AQ e VQ.

```
CREATE OR REPLACE FUNCTION inserisciprezzo()
        RETURNS trigger
        LANGUAGE 'plpgsql'

AS $BODY$
BEGIN

UPDATE PROIEZIONE SET PREZZO = 8
WHERE EXISTS
(SELECT * FROM film WHERE FIIM.titolo = Proiezione.titolo AND vq='2D' ) ;

UPDATE PROIEZIONE SET PREZZO = 10
WHERE EXISTS
(SELECT * FROM film WHERE FIIM.titolo = Proiezione.titolo AND vq='3D' ) ;

UPDATE PROIEZIONE SET PREZZO = 6
WHERE EXISTS
(SELECT * FROM film WHERE FIIM.titolo = Proiezione.titolo AND vq='no' ) ;
```

```
        return new;

END;
$BODY$;

CREATE TRIGGER trigger_oninsert
AFTER INSERT ON PROIEZIONE
FOR EACH ROW
EXECUTE PROCEDURE inserisciprezzo();
```

TRIGGER CHE AGGIORNA LA DISPONIBILITA' DEI BIGLIETTI SU PROIEZIONE
Ogni volta che si esegue una operazione di prenotazione o vendita dei biglietti bisogna aggiornare la quantità di biglietti venduti e prenotati, i quali sono registrati su Prenotazione.

```
CREATE FUNCTION public.disponibilita()
        RETURNS trigger
        LANGUAGE 'plpgsql'
        COST 100
        VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN

update proiezione set bigliettivenduti = (SELECT COUNT(*) FROM biglietto
where selettore=1 and cellulare is null and titolo=new.titolo and data=new.data and
ora=new.ora)

where titolo=new.titolo and data=new.data and ora=new.ora;
```

```
update proiezione set bigliettidisponibili = (SELECT COUNT(*) FROM biglietto
where selettore=0 and  titolo=new.titolo and data=new.data and ora=new.ora)

where titolo=new.titolo and data=new.data and ora=new.ora;
return new;

END;
$BODY$;

ALTER FUNCTION public.disponibilita()
        OWNER TO postgres;

CREATE TRIGGER trigger_disponibilita
        AFTER UPDATE
        ON public.biglietto
        FOR EACH ROW
        EXECUTE PROCEDURE public.disponibilita();
```

//TRIGGER CHE SETTA A 0 (DISPONIBILE) IL SELETTORE DI UN BIGLIETTO PRIMA CHE VENGA ELIMINATA LA PRENOTAZIONE
Se viene disdetta una prenotazione il biglietto liberato deve essere reso disponibile, quindi il suo selettore deve essere messo a 0.

```
CREATE FUNCTION public.eliminaprenotazione()
        RETURNS trigger
        LANGUAGE 'plpgsql'
        COST 100
        VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
   UPDATE BIGLIETTO SET SELETTORE =0 WHERE CELLULARE = OLD.CELLULARE;
   RETURN OLD;
END;
$BODY$;

ALTER FUNCTION public.eliminaprenotazione()
        OWNER TO postgres;

CREATE TRIGGER trigger_beforedelete
        BEFORE DELETE
        ON public.prenotazione
```

```
    FOR EACH ROW
    EXECUTE PROCEDURE public.eliminaprenotazione();
```

//PRENOTARE BIGLIETTI
Piccolo ciclo che si occupa di prenotare i primi biglietti disponibili per una proiezione proiezione dato un numero di cellulare. Questa operazione deve essere effettuata manualmente.

```
DO
$do$
BEGIN
  FOR i IN 1..(SELECT N_BIGLIETTI FROM PRENOTAZIONE WHERE CELLULARE =
'3471234567')
 LOOP
        UPDATE BIGLIETTO SET CELLULARE = '3471234567'
         WHERE (TITOLO = 'Avengers: Endgame'  AND DATA='2020-08-07' AND
ORA='21:00:00'
                          AND N_POSTO= (SELECT N_POSTO FROM BIGLIETTO
                WHERE TITOLO = 'Avengers: Endgame'
                AND DATA='2020-08-07' AND ORA='21:00:00' AND
                        SELETTORE=0 ORDER BY N_POSTO ASC LIMIT 1) );
  END LOOP;
END
$do$;
```

Interrogazioni

Possibili interrogazioni:

- Elencare film :
  SELECT * FROM FILM ;
- Elencare biglietti disponibili di un specifico film:
  SELECT * FROM BIGLIETTO WHERE selettore =0 and TITOLO=”Nome_Film”;
- Elencare titolo e descrizione dei film
  SELECT * FROM SCHEDA
  WHERE TITOLO=”Nome_Film”;
- Elencare prenotazione in base al nome e/o cellulare
  SELECT * FROM PRENOTAZIONE
  WHERE NOME=”Nome_Cliente”;
- Elencare la programmazione dei film
  SELECT * FROM PROIEZIONE;

- Prezzo del biglietto
  SELECT DISTINCT prezzo FROM BIGLIETTO
  WHERE NOMEFILM=”Nome_film” and Data=”data” and selettore=0;
- Incassi per singolo film
  SELECT (SELECT bigliettivenduti FROM proiezione WHERE titolo='Avengers: Endgame')*(SELECT prezzo FROM proiezione WHERE titolo='Avengers: Endgame');