Si consideri il seguente schema relazionale, relaivo ad una base di dati per gestire la programmazione cinematografica giornaliera in un certo insieme di cinema:

CINEMA(CodC, Nome, Indirizzo, Tel, NSale)
SALE(CodS, Nome, Nposti, CodC)
FILM(CodF, Titolo, CodRegista, CodProtagonista, Durata)
PERSONE(CodA, Nome, Cognome)
ATTORI(CodA, CodF)
PROIEZIONI(CodS, OraInizio, CodF)

Nella relazione CINEMA, Nome è chiave secondaria. Nella relazione FILM, CodRegista, e CodProtagonista sono chiavi esterne sulla tabella PERSONE.

Si richiede di:

a. progettare uno schema concettuale ad oggetti per il dominio applicativo precedente, motivando le scelte effettuate. Si assuma che le tabelle non abbiano chiavi secondarie. Si richiede di dare un nome ad entrambi i traversal path delle associazioni tra classi.
b. Specificare le classi utilizzando il linguaggio ODL
c. Rispondere alle seguente interrogazioni sullo schema progettato utilizzando il linguaggio OQL:
    a. Trovare il nome delle sale del cinema *Augustus*.
    b. Determinare le proiezioni (ore inizio e titolo film) delle sale con almeno 100 posti.
    c. Determinare i titoli dei film con attore *Brad Pitt*.
    d. Determinare i titoli dei film con attore *Brad Pitt* in cui Brad Pitt non è protagonista.
    e. Determinare per ogni sala del cinema Augustus il numero totale di film proiettati giornalmente.
    f. Trovare nome e cognome degli attori che hanno recitato il film diretti da Roberto Benigni.
    g. Per ogni cinema, restituire il suo nome e i nomi delle sale che contiene
    h. Restituire tutti i nomi di sale dei cinema con più di 100 posti.
    i. Per ogni film restituire il suo titolo e il numero di attori che hanno lavorato in quel film.
    j. Restituire l'insieme dei registi di film con durata superiore alle 2 ore.
d. A partire dalla schema ad oggetti realizzato al punto 1, si richiedere di definire uno schema ad oggetti in cui le associazioni sono modellate attraverso attributi. Non si rappresentino sempre le associazioni con le loro inverse, ma si faccia una scelta in base alle operazioni che effettueranno sullo schema (le operazioni da tenere in considerazione sono le interrogazioni al punto c).
e. A partire dallo schema al punto (d), presentare gli statement di creazione dello schema logico, utilizzando Oracle.
f. Scrivere in SQL, le interrogazioni del punto (c) ed effettuare le seguenti nuove operazioni:
    k. Supponendo che le tabelle siano popolate, presentare gli statement per inserire nella base di dati due nuove proiezioni per la sala identificata dal codice S3, con ora di inizio 18, per il film di codice f1, e con ora di inizio 22 per il film con codice f2.
g. Si supponga di voler garantire i seguenti vincoli di integrita':
    a. Mantenere aggiornato il campo NSale della tabella CINEMA, a fronte di inserimenti e cancellazioni dalla tabella SALE.
    b. Supponete di avere la tabella PROIEZ (NumFilm) che contiene la quantita' totale di film proiettati. Si vuole aggiornare automaticamente questa tabella.
    c. Si vuole garantire il seguente vincolo di integrita': Non si vuole inserire una nuova proiezione di un film in una sala, se in quella sala quel film viene gia' proiettato.
    d. Quando si inserisce un nuovo film, aggiungere come attore del film il protagonista.
    e. Quando si inserisce una nuova sala in un cinema C, se il numero dei posti e' NULL, assegnare a tale campo il numero minimo di posti delle sale del cinema C.

# 1. Schema ad oggetti

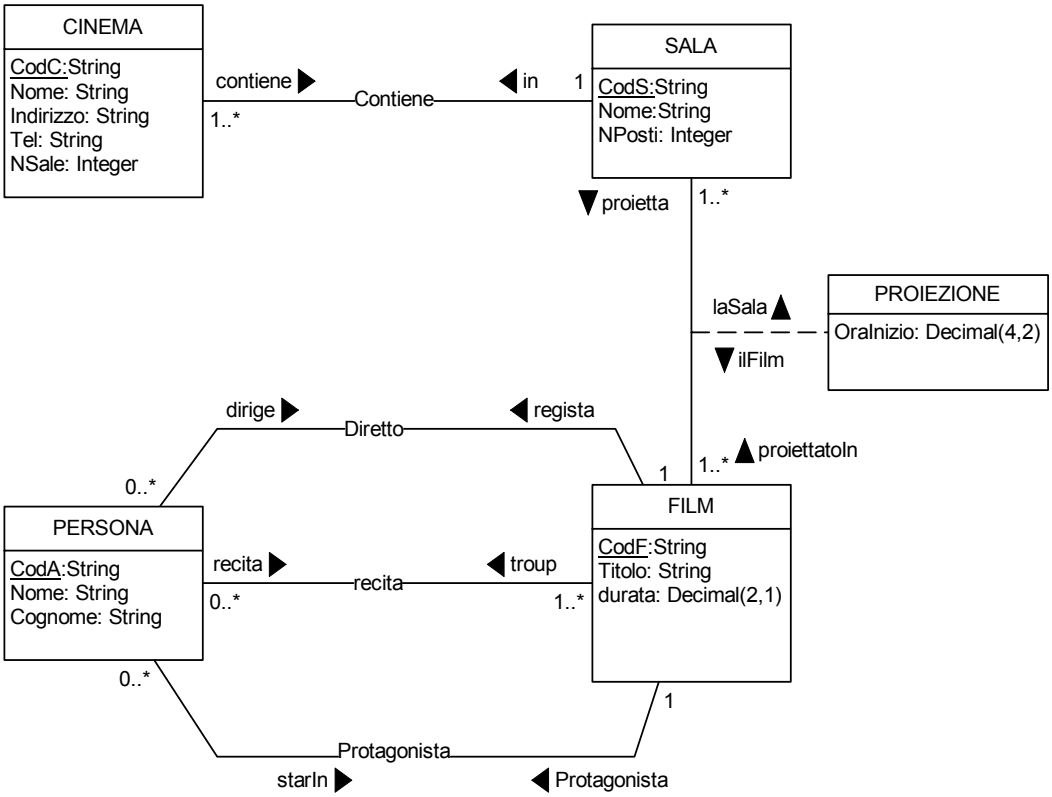


**Figura 1. Schema concettuale ad oggetti**

A partire dalla schema relazione proposto dal testo, si sono individuate le quattro entità/classi: Cinema, Sala, Film, Persona. Dopodiché si è individuata una associazione binaria tra Sala e Film con un attributo (OraInizio). Inoltre, si è inserito nelle classi i nomi di attributi con i relativi tipi. Si noti che per le associazioni è stato riportato sia il nome della associazione che i nomi dei traversal path.

# 2. Rappresentazione dello schema ad oggetti in OQL

A partire dal diagramma di Figura 1 si definiscono i seguenti statement di creazione delle classi e delle associazioni che esistono tra di loro.

```
Class Cinema
(extent Cinemas key CodC)
{  attribute string CodC;
   attribute string Nome;
   attribute string Indirizzo;
   attribute string Tel;
   attribute integer NSale;

   relationship Set<Sala> contiene inverse Sala::in;
}
```

```
Class Sala
(extent Sale key CodS)
{  attribute string CodS;
   attribute string Nome;
   attribute integer NPosti;

   relationship Cinema in inverse Cinema::contiene;
   relationship set<Proiezione> proietta inverse Proiezione::inSala; }

Class Proiezione
(extent Proiezioni key (OraInizio,inSala))
{  attribute integer OraInizio;

   relationship Sala inSala inverse Sala::proietta;
   relationship Film ilFilm inverse Film::proiettato;
}

Class Film
(extent Films key CodF)
{  attribute string CodF;
   attribute string Titolo;
   attribute integer Durata;

   relationship set<Proiezione> proiettato inverse Proiezione::ilFilm;
   relationship Persona regista inverse Persona::dirige;
   relationship Persona protagonista inverse Persona::starIn;
   relationship set<Persona> troup inverse Persona::recita;
}

Class Persona
(extent Persone key CodA)
{  attribute string CodA;
   attribute string Nome;
   attribute string Cognome;

   relationship Film dirige inverse Film::regista;
   relationship Film  starIn inverse Film::protagonista;
   relationship set<Film> recita inverse Film::troup;
}
```

## 3. Risposte alle interrogazione

a. Trovare il nome delle sale del cinema *Augustus*.

```
SELECT s.Nome
FROM Sale s
WHERE s.in.Nome = 'Augustus'
```

b. Determinare le proiezioni (ore inizio e titolo film) delle sale con almeno 100 posti.

```
SELECT p.OraInizio, p.ilFim.Titolo
FROM Proiezioni p
WHERE p.inSala.Nposti >= 100
```

c. Determinare i titoli dei film con attore *Brad Pitt*.

```
SELECT f.Titolo
FROM Persone p, p.recita f
WHERE p.Cognome = 'Pitt' and p.Nome = 'Brad'
```

d. Determinare i titoli dei film con attore *Brad Pitt* in cui Brad Pitt non è protagonista.

```
SELECT f.Titolo
FROM Persone p, p.recita f
WHERE p.Cognome = 'Pitt' and p.Nome = 'Brad' and f NOT IN p.starIn
--- soluzione alternativa ----------→ and f.primoAttore !=  p
---  N.B.: quest'ultima soluzione è da preferire perché molto più efficiente
```

e. Determinare per ogni sala del cinema Augustus il numero totale di film proiettati giornalmente.

```
SELECT s.Nome, COUNT(s.proietta)
FROM Sale s
WHERE s.in.Nome = 'Augustus'
```

f. Trovare nome e cognome degli attori che hanno recitato il film diretti da Roberto Benigni.

```
SELECT DISTINCT t.Cognome, t.Nome
FROM Film f, f.troup t
WHERE f.regista IN (SELECT p FROM Persone p WHERE p.Cognome='Benigni' and
p.Nome='Roberto')
--- oppure --- f.regista.Cognome='Benigni' and f.regista.Nome='Roberto'
-- N.B.: quest'ultima soluzione è da preferire perché molto più efficiente
```

g. Per ogni cinema, restituire il suo nome e i nomi delle sale che contiene

```
SELECT struct(cinema: c.Nome, sale: (SELECT s.Nome FROM Sale s WHERE s.in=c)
FROM Cinema c
```

h. Restituire tutti i nomi di sale dei cinema con più di 100 posti.

```
SELECT DISTINCT s.Nome
FROM  Sale s
WHERE s.Nposti > 100
```

i. Per ogni film restituire il suo titolo e il numero di attori che hanno lavorato in quel film.

```
SELECT f.Titolo, COUNT(f.troup)
FROM Film f
```

j. Restituire l'insieme dei registi di film con durata superiore alle 2 ore.

```
SELECT DISTINCT f.Regista
FROM Film f
WHERE f.Durata > 2
```

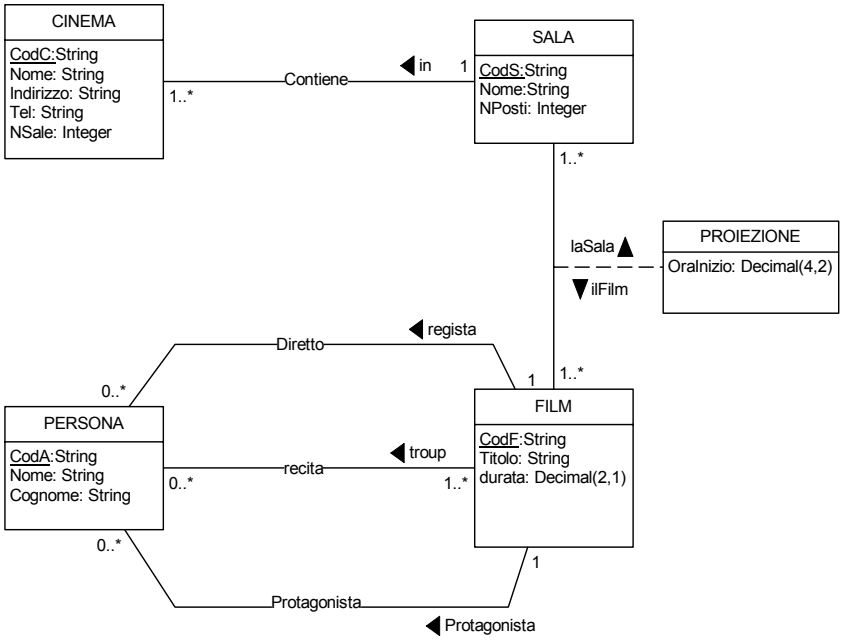


**Figura 2. Schema logico ad oggetti (con scelta dei traversal paths)**

# 4. Passaggio dallo schema concettuale ad oggetti allo schema logico relazionale ad oggetti

Una volta definito lo schema concettuale ad oggetti, è possibile ottenere uno schema logico relazionale ad oggetti, effettuando i seguenti passi:

1. Identificare i traversal paths che si vogliono mantenere nel modello relazionale ad oggetti. Nel modello relazionale ad oggetti non esistono le relationships del modello ad oggetti. Per cui le associazioni devono essere modellate attraverso attributi. Questo comporta che mantenere entrambi i traversal paths e' costoso (bisogna garantire l'integrita' referenziale), per cui si sceglie di mantenerne uno solo (quello che risulta piu' comodo in base al tipo di operazioni – interrogazioni -- che si vogliono effettuare sulla base di dati). In Figura 2 viene riportato lo schema concettuale presentato in Figura 1 in cui vengono mantenuti solo i traversal path che ci si aspetta di trovare nel modello relazionale ad oggetti. Questi traversal path permettono di rendere piu' efficienti la maggior parte delle interrogazioni previste per la base di dati.
2. Stabilire quali classi del modello ad oggetti (oppure quali entita' nel modello ER) devono essere rappresentate
    a. come relazioni oppure
    b. come tipi oggetto ed, eventualmente, come relazioni tipate.

    Come detto a lezione, ci possono essere vari motivi che influenzano questa scelta. La presenza di metodi in una classe o di sue sottoclassi porta a modellare una classe come tipo oggetto. Un altro motivo per modellare una classe come tipo oggetto e' il fatto che la classe e' "puntata" da un traversal path. Negli altri casi e' possibile modellare una classe come una relazione. Si noti infine che sebbene una classe possa essere modellata come un tipo oggetto, questo non implica che venga poi creata una relazione di tale tipo (relazione tipata). La relazione tipata viene creata qualora la classe nel modello concettuale e' rilevante indipendentemente dalle classi che la puntano. Utilizzando una terminologia object-oriented, questo vuol dire che la classe e' un entry-point per la formulazione delle interrogazioni (cioe' si e' interessati a recuperare informazioni a partire da tale classe).

Nella tabella in Figura 3, per ogni classe viene presentata la relativa rappresentazione del modello relazionale ad oggetti. La classe puo' essere rappresentata come una relazione, oppure come un tipo. Inolte a partire dal tipo e' possibile creare una relazione per contenere gli oggetti di quel tipo (relazione tipata). Nell'ultima colonna della tabella vengono riportati gli attributi di tipo oggetto dichiarati nella relazione/tipo/relazione tipata corrispondente. Si noti che, a parte la classe Cinema – il cui trattamento viene discusso di seguito--, le classi sono rappresentate attraverso tipi perche' vengono riferite da altre classi dello schema.

| **Classe** | **Relazione** | **Tipo** | **Relazione Tipata** | **Attributi di tipo oggetto** |
|---|---|---|---|---|
| Cinema | Cinema (Nota 1) | | | |
| Sala | | Tsala | Sale | |
| Proiezione | | Tproiezione (Nota 2) | proiezioni | insala REF Tsala<br>ilFilm REF Tsala |
| Persona | | Tpersona | persone | |
| Film | | Tfilm | film | protagonista REF Tpersona<br>regista REF Tpersona<br>troup Tattori |
| | | Tattori (TABLE OF REF Tpersona) | | |

**Figura 3. Mappatura dal modello concettuale ad oggetti al modello logico relazionale ad oggetti**

Nota 1: La classe Sala punta la classe Cinema. Per cui in questo caso, la classe Cinema poteva essere modellata come tipo oggetto e poi creata una relazione tipata. Dal momento pero' che c'e' una sola classe che riferisce la classe cinema, in questo caso si e' deciso di modellare cinema come una relazione e di realizzare il riferimento da parte della classe Sala inserendo un costraint di integrita' referenziale tipico del modello relazione puro. Questa e' una scelta molto opinabile. Ad ogni modo, permette di mettere in evidenza che il meccanismo di integrita' referenziale nel modello relazionale puro, e' totalmente supportato nel modello relazionale ad oggetti.

Nota 2: Anche la scelta di modellare le proiezioni come tipo oggetto e' opinabile. In questo caso si poteva modellare come una relazione perche' la classe proiezione punta due classi (Sala e Film) ma non viene puntata da nessuna classe.

## 3. Statement Oracle per la creazione dello schema

A partire dalla mappatura riportata in Figura 3 e' possibile derivare i seguenti statement Oracle per la definizioni della schema relazionale ad oggetti.

```
create table cinema(
CodC varchar(5) PRIMARY KEY,
Nome varchar(10) NOT NULL,
Indirizzo varchar(20),
Tel varchar(10),
NSale integer
);

create type Tsala as OBJECT
(
CodS varchar(5),
Nome varchar(10),
Nposti integer,
inCinema varchar(5)
);
```

```
create table sala of tsala
(
PRIMARY KEY(CodS),
CONSTRAINT cinema_fk FOREIGN KEY (inCinema) REFERENCES cinema(CodC)
);

create type Tpersona AS OBJECT(
CodP varchar(5),
Nome varchar(10),
Cognome varchar(10)
);

create table persona of tpersona
(PRIMARY KEY(CodP));

CREATE TYPE Tattori as TABLE OF REF Tpersona;

create type TFilm AS OBJECT(
CodF varchar(5),
Titolo varchar(10),
Durata integer,
troup Tattori,
protagonista REF TPersona,
regista REF TPersona
);

create table Film of TFilm(
PRIMARY KEY(CodF)
) NESTED TABLE troup STORE AS troup_tab;

create type Tproiezioni AS OBJECT
( inSala REF Tsala,
  ilFilm REF Tfilm,
  oraInizio integer
);

create table Proiezioni of Tproiezioni;
```

## 4. Statement Oracle per la popolazione dello schema

Lo schema realizzato al passo precedente puo' essere popolato utilizzando i seguenti comandi del DML di Oracle.

```
insert into cinema values ('c100','Augustus','Via Pioppi, 35','0103536638',3);
insert into cinema values ('c200','Universal','Via Mazzini, 38','0103536699',1);

insert into sala values ('s1','Gala',100,'c100');
insert into sala values ('s2','Trionfo',90,'c100');
insert into sala values ('s3','Platea',110,'c100');
insert into sala values ('s4','Gala',100,'c200');

insert into persona values ('p1','Roberto','Benigni');
insert into persona values ('p2','Brad','Pitt');
insert into persona values ('p3','Sharon','Stone');
insert into persona values ('p4','Sabrina','Ferilli');
insert into persona values ('p5','Wolfgang','Petersen');
insert into persona values ('p6','Julian','Glover');
insert into persona values ('p7','Diane','Kruger');
insert into persona values ('p8','Nicoletta','Braschi');
insert into persona values ('p9','Giorgio','Cantarini');
```

```
insert into film values('f1','La vita e''', 2,tattori(), NULL, NULL);
update film
set protagonista= (select ref(p) from persona p where p.cognome='Benigni')
where codF='f1';

update film
set regista= (select ref(p) from persona p where p.cognome='Benigni')
where codF='f1';

insert into table(select troup from film where codF='f1')
select ref(p) from persona p where p.codP IN ('p1','p8','p9');

insert into film values('f2','troy', 1,tattori(), NULL, NULL);

update film
set protagonista= (select ref(p) from persona p where p.cognome='Pitt')
where codF='f2';

update film
set regista= (select ref(p) from persona p where p.cognome='Petersen')
where codF='f2';

insert into table(select troup from film where codF='f2')
select ref(p) from persona p where p.codP IN ('p2','p6','p7');

insert into proiezioni
   select ref(s), ref(f), 10
   from film f, sala s
   where f.codf='f1' and s.codS='s1';

insert into proiezioni
   select ref(s), ref(f), 12
   from film f, sala s
   where f.codf='f1' and s.codS='s1';

insert into proiezioni
   select ref(s), ref(f), 20
   from film f, sala s
   where f.codf='f1' and s.codS='s1';

insert into proiezioni
   select ref(s), ref(f), 10
   from film f, sala s
   where f.codf='f1' and s.codS='s4';

insert into proiezioni
   select ref(s), ref(f), 12
   from film f, sala s
   where f.codf='f1' and s.codS='s4';

insert into proiezioni
   select ref(s), ref(f), 20
   from film f, sala s
   where f.codf='f1' and s.codS='s4';

insert into proiezioni
   select ref(s), ref(f), 10
   from film f, sala s
   where f.codf='f2' and s.codS='s3';
insert into proiezioni
   select ref(s), ref(f), 12
   from film f, sala s
   where f.codf='f2' and s.codS='s3';
```

```
insert into proiezioni
select ref(s), ref(f), 20
from film f, sala s
where f.codf='f2' and s.codS='s3';
```

## 5. Semplici interrogazioni SQL

Rispondere alle seguente interrogazioni sullo schema progettato utilizzando il linguaggio SQL di Oracle:

a. Trovare il nome delle sale del cinema *Augustus*.

```
SELECT s.Nome
FROM Sala s, Cinema c
WHERE s.inCinema=c.CodC AND c.Nome = 'Augustus';
```

N.B. Questo tipo di interrogazione deriva dalla scelta effettuata in fase di passaggio dal modello concettuale a quello logico.

b. Determinare le proiezioni (ore inizio e titolo film) delle sale con almeno 100 posti.

```
SELECT p.OraInizio, p.ilFilm.Titolo
FROM Proiezioni p
WHERE p.inSala.Nposti >= 100;
```

c. Determinare i titoli dei film con attore *Brad Pitt*.

```
SELECT DISTINCT f.Titolo
FROM Film f, TABLE(f.Troup) t
WHERE value(t).Cognome = 'Pitt' and value(t).Nome = 'Brad';
```

d. Determinare i titoli dei film con attore *Brad Pitt* in cui Brad Pitt non è protagonista.

```
SELECT f.Titolo
FROM Film f
WHERE f.protagonista.Cognome != 'Pitt' and f.protagonista.Nome != 'Brad'
and EXISTS  (select p from TABLE(f.Troup) p where
value(p).Cognome='Pitt' AND value(p).Nome='Brad');
```

e. Determinare per ogni sala del cinema Augustus il numero totale di film proiettati giornalmente.

```
SELECT p.inSala.Nome, COUNT(*)
FROM Proiezioni p, Cinema c
WHERE c.codC=p.inSala.inCinema AND c.nome='Augustus'
GROUP BY p.inSala.Nome;
```

f. Trovare nome e cognome degli attori che hanno recitato il film diretti da Roberto Benigni.

```
SELECT DISTINCT value(t).Cognome, value(t).Nome
FROM Film f, TABLE(f.troup) t
WHERE  f.regista.Cognome='Benigni' and f.regista.Nome='Roberto';
```

g. Per ogni cinema, restituire il suo nome e i nomi delle sale che contiene

```
SELECT c.nome, s.Nome
FROM Cinema c, Sala s
WHERE c.CodC=s.inCinema;
```

h. Restituire tutti i nomi di sale dei cinema con più di 100 posti.

```
SELECT DISTINCT s.Nome
FROM  Sala s
WHERE s.Nposti > 100;
```

i. Per ogni film restituire il suo titolo e il numero di attori che hanno lavorato in quel film.

```
SELECT f.Titolo, COUNT(*)
FROM Film f, TABLE(f.troup)
GROUP BY f.Titolo;
```

j. Restituire l'insieme dei registi di film con durata superiore alle 2 ore.

```
SELECT deref(f.regista)
FROM Film f
WHERE f.durata > 2;
```

k. Supponendo che le tabelle contengano un certo numero di tuple, presentare gli statement per inserire nella base di dati due nuove proiezioni per la sala identificata dal codice S3, con ora di inizio 18, per il film di codice f1, e con ora di inizio 22 per il film con codice f2.

```
insert into proiezioni
select ref(s), ref(f), 18
from film f, sala s
where f.codf='f1' and s.codS='s3';

insert into proiezioni
select ref(s), ref(f), 22
from film f, sala s
where f.codf='f2' and s.codS='s3';
```

# 6. Triggers

Si supponga di voler garantire i seguenti vincoli di integrita':

f. Mantenere aggiornato il campo NSale della tabella CINEMA, a fronte di modifiche della tabella SALE.

```
-- versione in oracle funzionante
CREATE OR REPLACE TRIGGER Aggiorna_NumSale
AFTER UPDATE OF inCinema OR DELETE OR INSERT ON Sala
FOR EACH ROW
begin
  IF UPDATING
  THEN
    UPDATE Cinema t
    SET NSale = NSale - 1
    WHERE :old.inCinema=t.CodC;

    UPDATE Cinema t
    SET NSale = NSale + 1
    WHERE :new.inCinema=t.CodC;
  END IF;

  IF DELETING
  THEN
    UPDATE Cinema t
    SET NSale = NSale - 1
    WHERE :old.inCinema=t.CodC;
  END IF;

IF INSERTING
  THEN
    UPDATE Cinema t
    SET NSale = NSale + 1
    WHERE :new.inCinema=t.CodC;
  END IF;

end;
```

g. Supponete di avere la tabella PROIEZ(NumFilm) che contiene la quantita' di film proiettati. Si vuole aggiornare automaticamente questa tabella.

```
    -- questi trigger sono espressi in SQL-99 perche' in Oracle non e' possibile far
    --  riferimento a una tabella (solo a ROW)
CREATE TRIGGER Aggiorna_Proiez
AFTER INSERT ON Proiezioni
REFERENCING NEW TABLE AS New
FOR EACH STATEMENT
UPDATE Proiez
SET NumFilm = NumFilm + SELECT COUNT(*) FROM New;

CREATE TRIGGER Aggiorna_Proiez
AFTER DELETE ON Proiezioni
REFERENCING OLD TABLE AS Old
FOR EACH STATEMENT
UPDATE Proiez
SET NumFilm = NumFilm - SELECT COUNT(*) FROM Old;
```

h. Si vuole garantire il seguente vincolo di integrita': Non si vuole inserire una nuova proiezione di un film in una sala, se in quella sala quel film viene gia' proiettato.
i. Quando si inserisce un nuovo film, aggiungere come attore del film il protagonista.

Quando si inserisce una nuova sala in un cinema C, se il numero dei posti e' NULL, assegnare a tale campo il numero minimo di posti delle sale del cinema C.