PROGETTO BASI DI DATI

CINEMA

Studente: SAMUELE MARIA GALLINA

Matricola: **********

Descrizione

Si vuole progettare un database per la gestione di un Cinema con un certo numero di **sale**,dove ogni sala è caratterizzata da informazioni come il numero di posti a sedere e dalla grandezza dello schermo. Ogni sala ha le **proiezioni**(caratterizzate dalla sala,dall'id del film,dall'orario).All'interno del cinema troviamo i **dipendenti** con un certo id, uno stipendio e una descrizione della sua mansione in particolare e un'informazione riguardo al suo giorno libero,il dipendente può essere assegnato a un **luogo di lavoro**(biglietteria,sala,bar)

All'interno del cinema troviamo le **biglietterie che appunto vendono biglietti**. Sempre all'interno del cinema troviamo anche i **bar,che vendono dei prodotti**, prelevandoli, in caso siano finiti,viene effettuato un **ordine**.

Termine	Descrizione	Termini collegati
Luogo di lavoro	Un dipendente può esservi assegnato	Sala,biglietteria,bar, dipendenti
Sala	Ha un certo schermo e un numero di posti a sedere	Proiezione,film
FILM	info	Proiezione,sala
Proiezione	E’ fatta in una certa sala a un orario e data e si proietta un film	sala,biglietteria
Dipendenti	Hanno uno stipendio, una descrizione della loro mansione e un giorno libero,possono essere assegnati a una sala, un biglietteria o un bar	Sala,biglietteria,bar
Biglietteria	Vende i biglietti per delle proiezioni	Proiezione
Bar	Vende dei determinati prodotti	Prodotti
Ordine	Effettuato se il prodotto è assente	Dipendenti
Prodotti	Ha tutti i prodotti con le loro quantità	

Dati

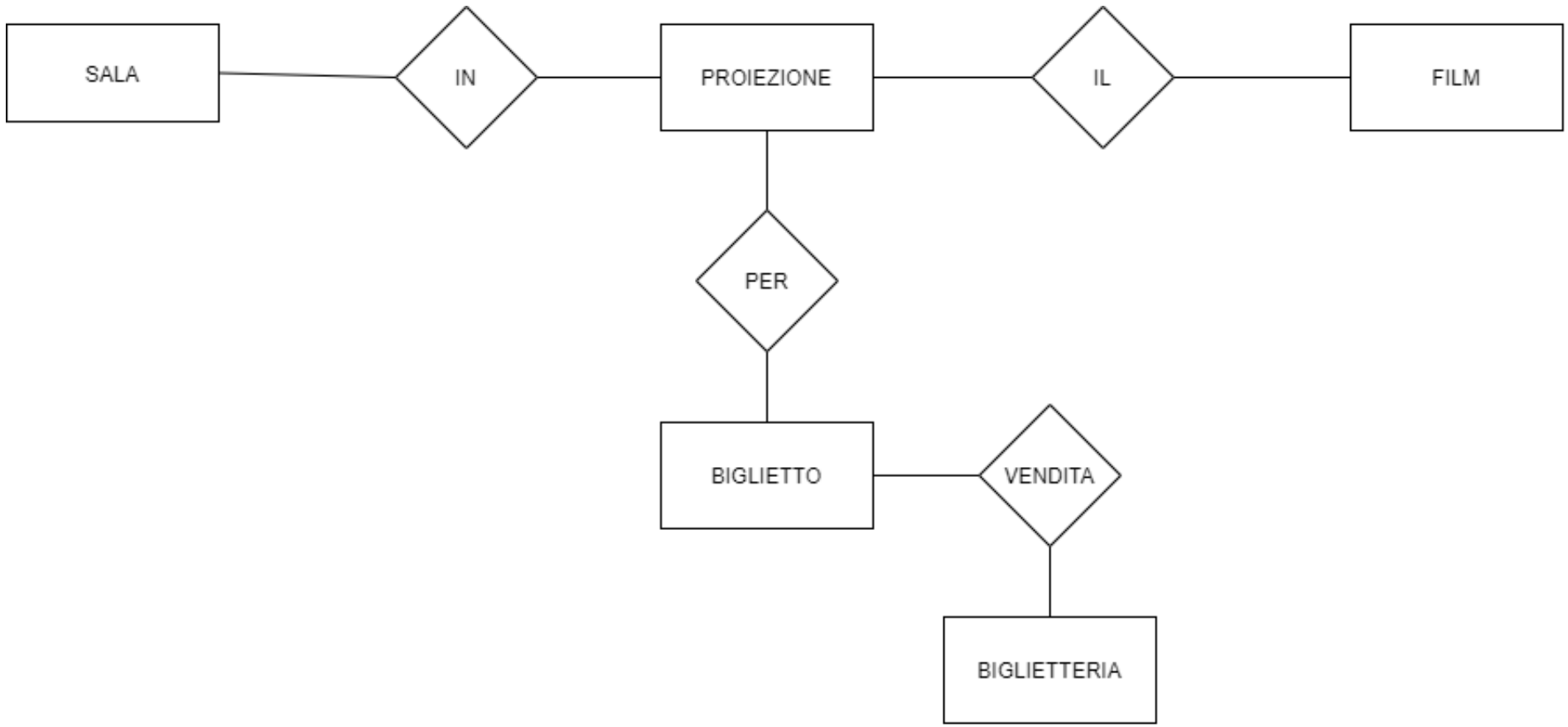
Il cinema ha 10 sale e per ogni sala abbiamo 100 proiezioni,i film noleggiati sono 5000. I biglietti venduti sono 100000, le biglietterie sono 3.

I bar sono 3 e vende 50 prodotti.

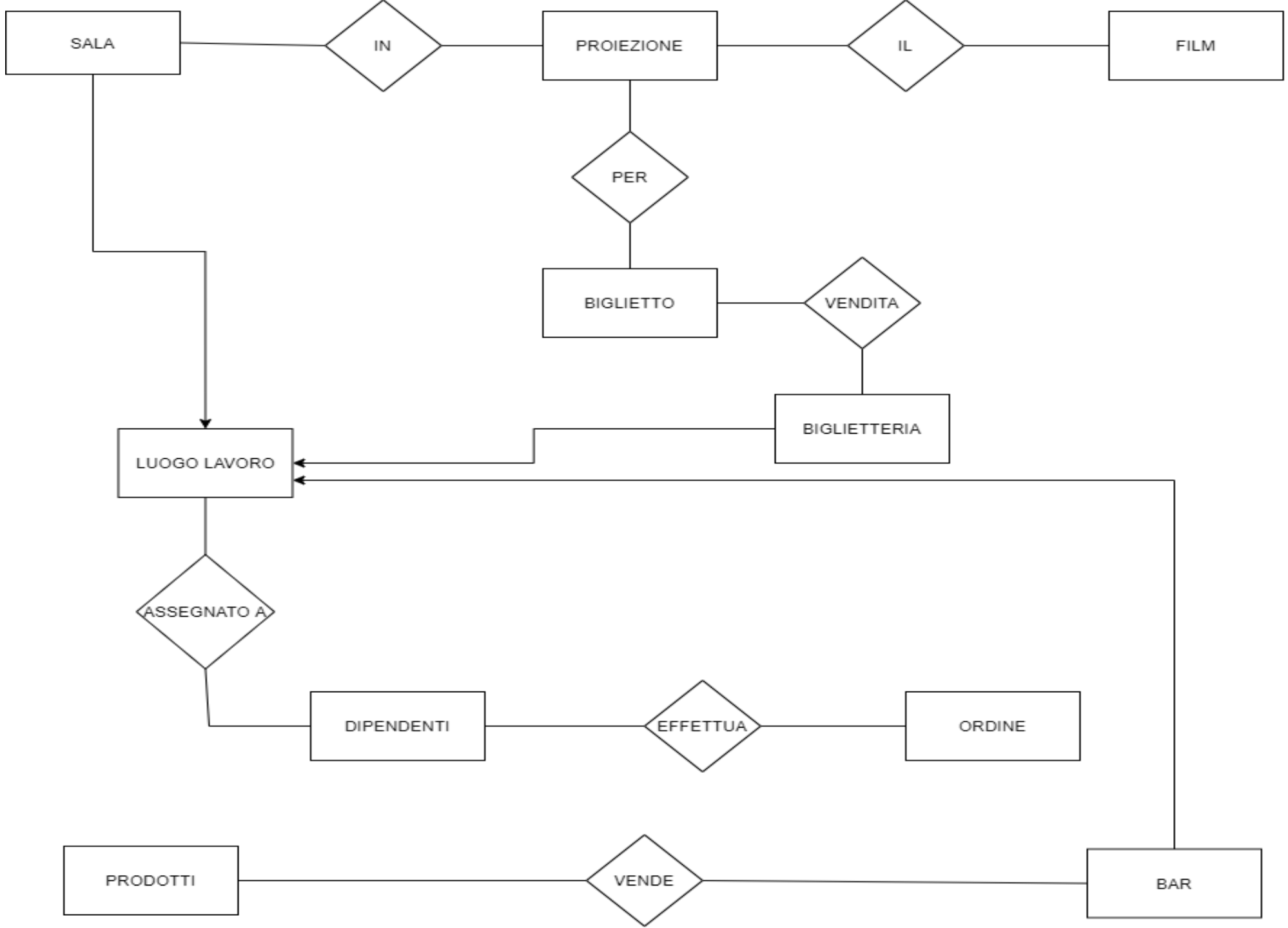
I dipendenti all’interno del cinema sono 30,il bar vende 50 prodotti,sono stati venduti 10000 prodotti.

Gli ordini sono 500.

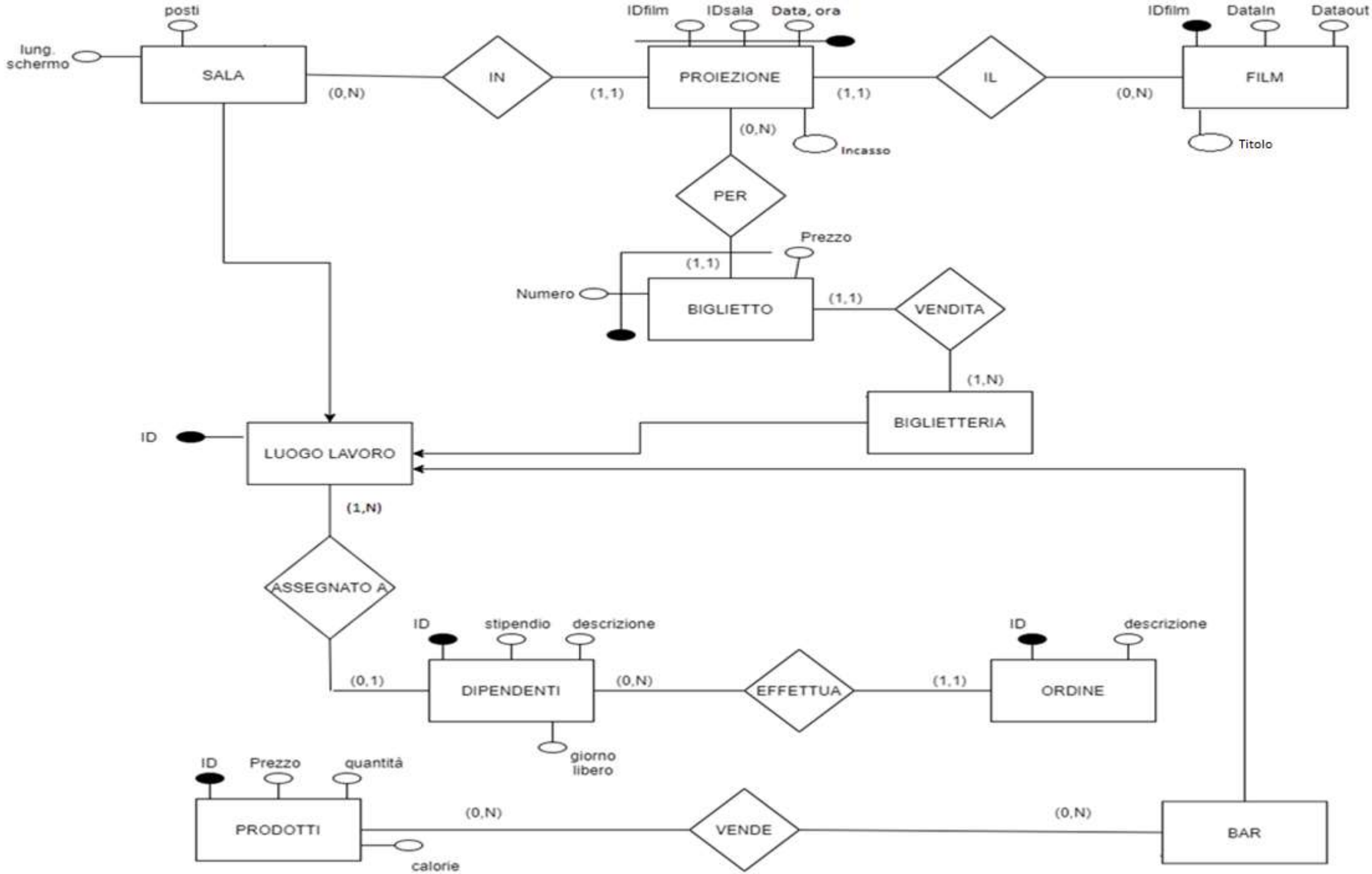
Schema scheletro



Aggiungiamo tutti i dettagli relativi al nostro sistema informativo



Aggiungiamo attributi(tra cui id.) e cardinalità per riportarci al modello ER



Vincoli non esprimibili dall'ER

- Un prodotto non può essere venduto se l'attributo quantità è 0
- Se l'attributo quantità è 0, un dipendente effettua un ordine
- Quando è venduto un biglietto per una proiezione, l'attributo incasso è aggiornato sommando il prezzo
- Un film non è proiettabile se la data corrente ha superato DataOut o se è prima di DataIn
- Ovviamente DataOut non può essere una data precedente a DataIn

Dizionario dati, entità

Entità	Descrizione	Attributi	Identificatore
Luogo lavoro	Si suddivide in sala,biglietteria,bar	ID	ID
Sala	Ha un certo schermo e un numero di posti a sedere	Vedi luogo lavoro + lung.schermo,posti	Vedi luogo lavoro
Biglietteria	Vende i biglietti per delle proiezioni	Vedi luogo lavoro	Vedi luogo lavoro
Bar	Vende dei determinati prodotti	Vedi luogo lavoro	Vedi luogo lavoro
Proiezione	Caratterizzata da un film,una sala e un momento,ha fatto un certo incasso	IDfilm,IDsala, data e ora e incasso	IDfilm,IDsala, data e ora

Entità	Descrizione	Attributi	Identificatore
Biglietto	Fa riferimento a una certa proiezione e ha un numero	Vedi proiezione + prezzo + numero	Identificatore di proiezione + numero
Dipendenti	Hanno uno stipendio,un giorno libero, e una descrizione della loro mansione,possono essere assegnati a un luogo e possono effettuare ordini	ID,stipendio,desc.,giorno libero	ID
Film	Hanno un loro ID e sono noleggiati da una certa data fino a un’altra data	ID,DataIN,DataOut	ID
Ordine	Effettuato se il prodotto è assente da un dipendente	ID,descrizione	ID
Prodotti	Ha tutti i prodotti con le loro quantità e prezzo, si sa anche il apporto calorico	ID,quantità,prezzo,calorie	ID

Per le **relazioni**; abbiamo semplicemente **IN** che collega SALA a PROIEZIONE con una relazione 1 a molti(una proiezione può essere fatta in una sola sala ma una sala ha più proiezioni).

La relazione **IL** collega FILM a PROIEZIONE sempre con una relazione 1 a molti.

VENDITA collega BIGLIETTERIA a BIGLIETTO con una 1 a molti(un persona potrebbe comprare più biglietti nello stesso momento), **PER** collega PROIEZIONE a BIGLIETTO con una 1 a molti sempre per lo stesso ragionamento.

ASSEGNATO A collega DIPENDENTI a LUOGO LAVORO con una 1 a molti(una persona può essere assegnata a un luogo, a un luogo possono essere assegnate più persone), allo stesso modo DIPENDENTI si collega a ORDINE con **EFFETTUA** in una 1 a molti, e BAR si collega a PRODOTTI tramite **VENDE** con una 1 a molti.

Operazioni più importanti

- O1: Vendi biglietto: 1000 al giorno
- O2: Aggiungi proiezione: 30 al giorno
- O3: Aggiungi film: 1 ogni 3 giorni
- O4: Aggiungi ordine: 1 a settimana
- O5: Aggiungi vendita prodotti: 100 al giorno
- O6: Visualizza incasso proiezione :30 al giorno
- O7: Modifica valore quantità prodotti: 3 a settimana
- O8: Leggere incasso vendita prodotti di un giorno: 1 al giorno

Volume dati

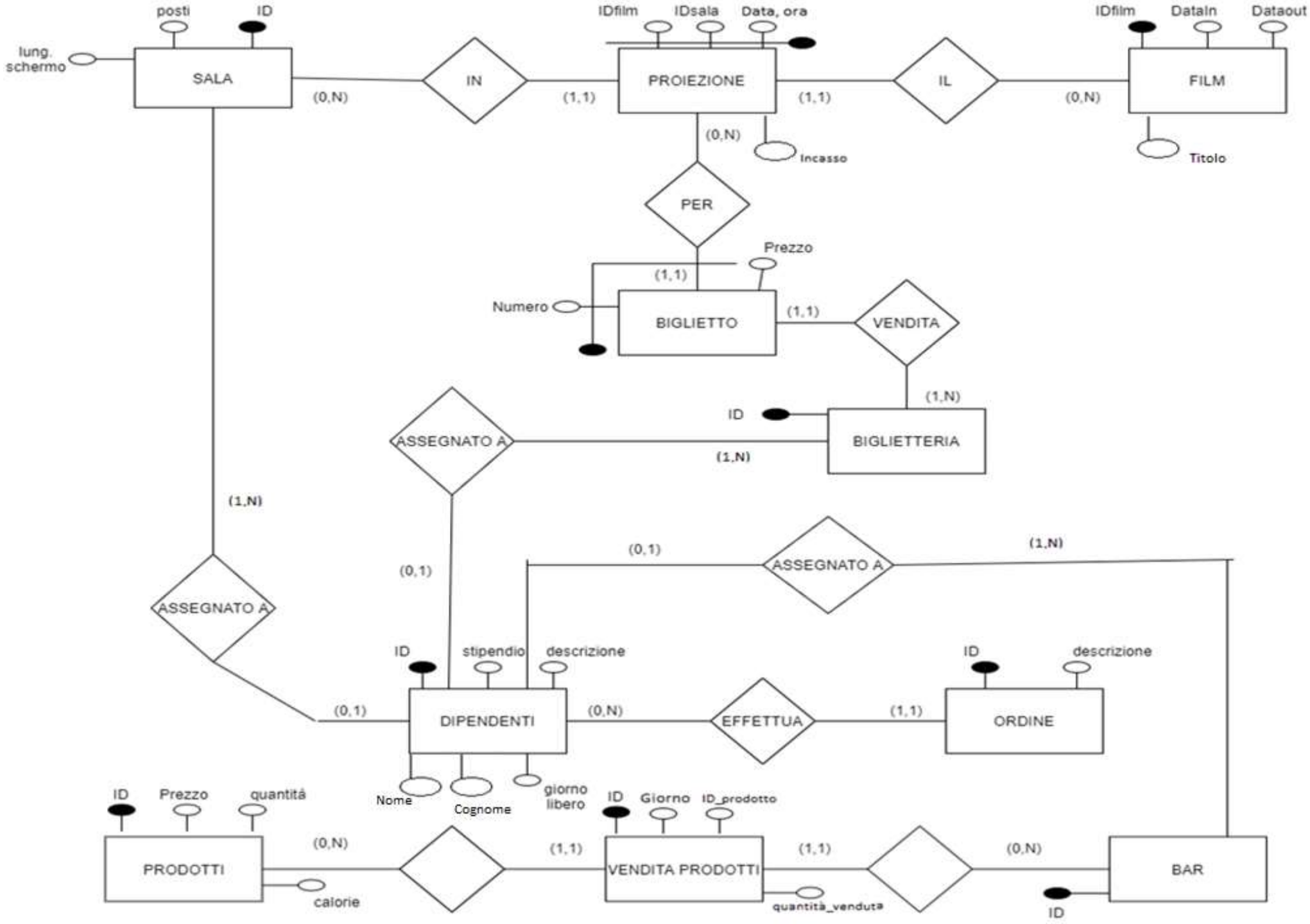
Ribadiamo i dati presenti nel nostro sistema:

- Sala: 10
- Proiezioni: 100*10
- Film:5000
- Biglietti venduti: 100000
- Biglietterie: 3
- Bar: 3
- Prodotti: 50
- Vendita prodotti: 10000
- Ordini: 500

Traduzione verso il modello relazionale

Primi passaggi:
Traformazione VENDE
in una entità che tiene conto del prodotto venduto(e relativa quantità) in un certo giorno

Eliminazione gerarchia



Abbiamo fatto nascere una nuova entità; VENDITA PRODOTTI, per mantenere informazioni riguardo a vendite passate, con:

- ID che identifica la vendita
- Giorno per sapere in che giorno è avvenuta la vendita
- Quantità_venduta per sapere quanti pezzi del prodotto sono stati venduti
- ID_prodotto per sapere il prodotto che è stato venduto

Nascono nuovi vincoli non esprimibili:

- Nel momento in cui è inserita una vendita, il giorno è il giorno corrente
- quantita_venduta non può essere un numero maggiore di quantita nella entità PRODOTTI e non può essere né 0 né un numero negativo

Valutiamo se è conveniente mantenere l'attributo ridondante «incasso», infatti esso sarebbe calcolabile.

Le operazioni che ci interessano in questo caso sono 2, ovvero la O1 e la O6.

O1:

Quando inseriamo un biglietto, dobbiamo successivamente leggere il dato incasso per poi aggiornarlo,dunque dobbiamo fare una scrittura del biglietto,una lettura di incasso e una scrittura di incasso, dunque 5 letture.

Oggetto	Accessi	tipo
Biglietto	1	S
Proiezione	1	L
Proiezione	1	S

Considerando che l'operazione è eseguita 1000 volte…

abbiamo 5000 letture se il dato è presente.

Se invece non abbiamo il dato dobbiamo semplicemente fare una scrittura di biglietto, incasso non c'è quindi non va aggiornanto

Oggetto	Accessi	tipo
Biglietto	1	S

Abbiamo solo una scrittura, quindi 2 letture, moltiplicate per le 1000 volte in cui è eseguita l'operazione abbiamo 2000 letture se il dato è assente

O6:

Se dobbiamo invece visualizzare l'incasso, col dato dobbiamo solo leggere una volta il dato incasso

Oggetto	Accessi	tipo
Proiezione	1	L

L'operazione è eseguita 30 volte, quindi abbiamo 30 letture se il dato è presente.

Se non abbiamo il dato dobbiamo effettuare qualche calcolo in più. Per ogni proiezione abbiamo in media 100 (deriva da 100000/1000) biglietti

Oggetto	Accessi	tipo
Biglietti	100	L

dobbiamo leggere questi 100 biglietti per 30 volte al giorno,ovvero abbiamo 3000 letture.

Conclusione sul dato derivato «incasso»

Dati questi risultati:

O1 con dato: 5000L

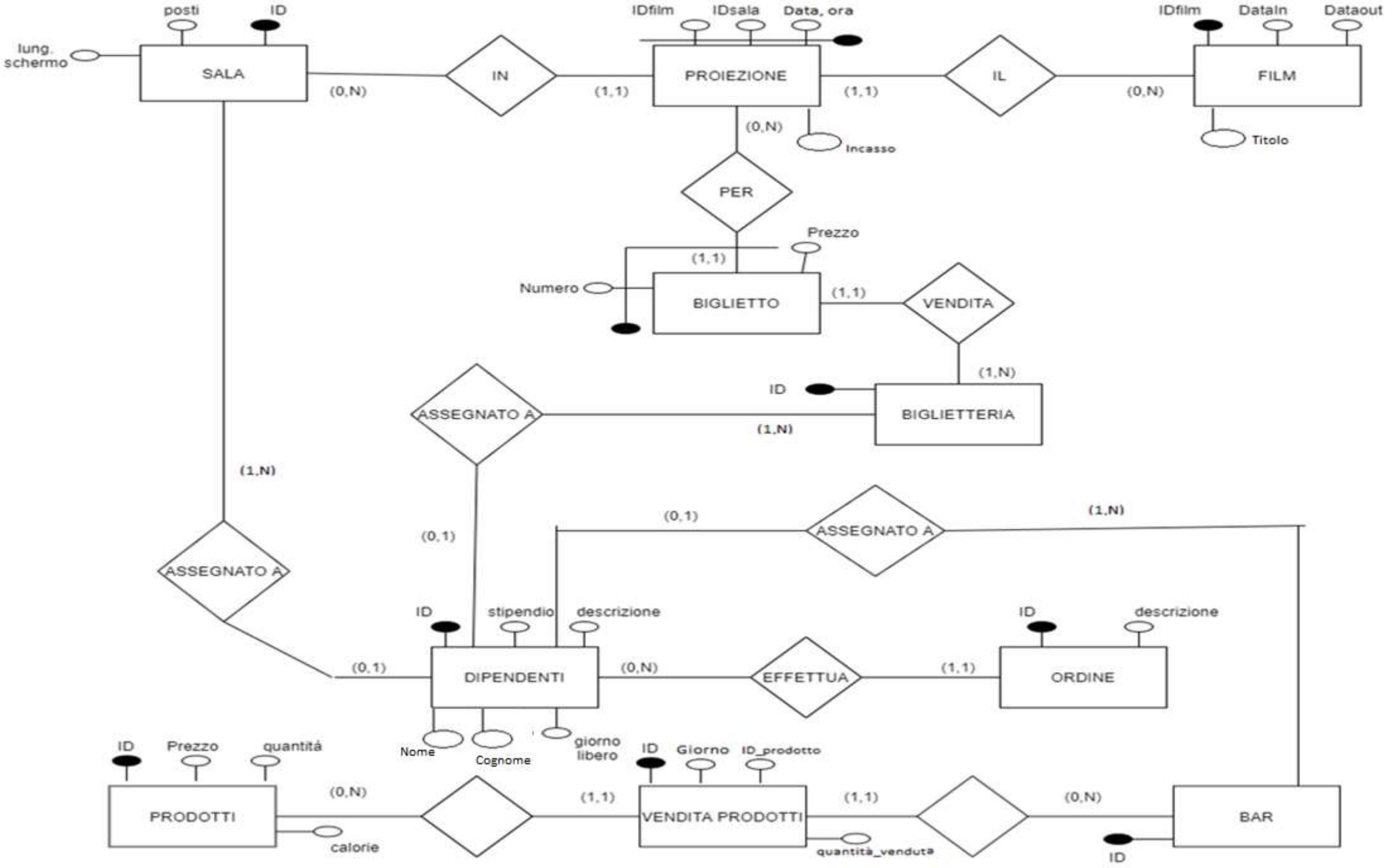
O1 senza dato: 2000L

O6 con dato: 30L

O6 senza dato: 3000L

Dunque se abbiamo il dato eseguiamo 5030 letture, mentre se non abbiamo il dato eseguiamo 5000 letture, la differenza è trascurabile, si decide di mantenere il dato nel sistema informativo per semplicità

Modello relazionale ottenuto



Trad. nel modello logico

Grassetto: chiave
Sottolineato: chiave esterna

Sala(**IDsala**,lung_schermo,posti)

Film(**IDfilm**,DataIn,DataOut,Titolo)

Proiezione(**IDfilm,IDsala,Data,Ora**,Incasso)

Biglietto(Prezzo,**Numero,IDsala,IDfilm,Data,Ora**,IDbiglietteria)

Biglietteria(**IDbiglietteria**)

Dipendenti(**IDdipendente**,Nome,Cognome,stipendio,giorno_libero,descrizione,
assegnato-a-bar , assegnato-a-sala , assegnato-a-biglietteria)

Ordine(**IDordine**,descrizione,IDdipendente)

Bar(**IDbar**)

VenditaProdotti(**IDvendita**,IDbar,IDprodottoVenduto,quantità_venduta,
Giorno)

Prodotti(**IDprodotto**,quantità,prezzo,calorie)

Livello fisico

- Passiamo alla traduzione del nostro sistema informativo per essere utilizzabile dal DBMS,scriviamo le istruzioni che andrebbero eseguite se implementassimo a basso livello da linea di comando.
  Nella pratica poi è comunque usuale ormai usare vari tools
  (es; PHPmyAdmin)

Implementazione tabelle in SQL

```sql
CREATE TABLE Sala(
    IDsala int primary key,
    lung_schermo int,
    posti int
);
```

```sql
CREATE TABLE Film(
    IDfilm int AUTO_INCREMENT primary key,
    DataIn date,
    DataOut date,
    Titolo varchar(30)
);
```

```sql
CREATE TABLE Proiezione(
    IDfilm int,
    IDsala int,
    Data_proiezione date,
    Ora time,
    incasso float default 0,
    primary key(IDfilm,IDsala,Data_proiezione,Ora),
    FOREIGN KEY(IDfilm) REFERENCES Film(IDfilm),
    FOREIGN KEY(IDsala) REFERENCES Sala(IDsala)
);

CREATE TABLE Biglietto(
    prezzo float,
    numero int,
    IDsala int,
    IDfilm int,
    Data_proiezione date,
    Ora time,
    IDbiglietteria int,
    primary key(numero,IDfilm,IDsala,Data_proiezione,Ora),
    FOREIGN KEY(IDfilm) REFERENCES Film(IDfilm),
    FOREIGN KEY(IDsala) REFERENCES Sala(IDsala),
    FOREIGN KEY(IDbiglietteria) REFERENCES Biglietteria(IDbiglietteria)

);
```

```sql
CREATE TABLE Biglietteria(
    IDbiglietteria int primary key
);


CREATE TABLE Dipendenti(
    IDdipendente int primary key,
    Nome varchar(20),
    Cognome varchar(20),
    stipendio float,
    giorno_libero varchar(10),
    descrizione varchar(60),
    assegnato_a_bar int,
    assegnato_a_sala int,
    assegnato_a_biglietteria int,

    FOREIGN KEY(assegnato_a_biglietteria) REFERENCES Biglietteria(IDbiglietteria),
    FOREIGN KEY(assegnato_a_bar) REFERENCES Bar(IDbar),
    FOREIGN KEY(assegnato_a_sala) REFERENCES Sala(IDsala)
);
```

```sql
CREATE TABLE Ordine(
    IDordine int AUTO_INCREMENT primary key,
    descrizione varchar(60),
    IDdipendente int,

    FOREIGN KEY(IDdipendente) REFERENCES Dipendenti(IDdipendente)
);



CREATE TABLE Bar(
    IDbar int primary key
);
```

```sql
CREATE TABLE VenditaProdotti(
    IDvendita int AUTO_INCREMENT primary key,
    IDbar int,
    IDProdottoVenduto int,
    quantita_venduta int,
    Giorno date,

    FOREIGN KEY(IDProdottoVenduto) REFERENCES Prodotti(IDprodotto),
    FOREIGN KEY(IDbar) REFERENCES Bar(IDbar)
);
```

```sql
CREATE TABLE Prodotti(
    IDProdotto int primary key,
    quantita int,
    prezzo float,
    calorie float
);
```

IMPLEMENTAZIONE OPERAZIONI IN SQL

```
--O1:VENDERE BIGLIETTO
--Passo 1: inserire nuova tupla biglietto
--(Supponendo le variabili passate a values abbiano dei valori)
INSERT INTO Biglietto(prezzo,numero,IDsala,IDfilm,Data_proiezione,Ora,IDbiglietteria)
VALUES(prezzo,numero,IDsala,IDfilm,Data_proiezione,Ora,IDbiglietteria);


--Passo 2: aggiornare Incasso della proiezione corrispondente

CREATE TRIGGER Aggiorna_incasso
AFTER INSERT ON Biglietto
FOR EACH ROW
BEGIN
    If(EXISTS(SELECT *
              FROM Proiezione
              WHERE IDsala=new.IDsala AND IDfilm=new.IDfilm AND
              Data_proiezione=new.Data_proiezione AND Ora=new.Ora
                ))
    THEN UPDATE Proiezione set incasso=incasso + new.prezzo
    WHERE (IDsala=new.IDsala AND IDfilm=new.IDfilm AND
           Data_proiezione=new.Data_proiezione AND Ora=new.Ora);
    ELSE delete from Biglietto
    WHERE (numero=new.numero AND IDsala=new.IDsala AND IDfilm=new.IDfilm AND
            Data_proiezione=new.Data_proiezione AND Ora=new.Ora);
    END IF;
END
```

```
--O2: Aggiungere proiezione
INSERT INTO Proiezione(IDfilm,IDsala,Data_proiezione,Ora)
VALUES(IDfilm,IDsala,Data_proiezione,Ora);
--Non possiamo proiettare un film se non è disponibile
CREATE TRIGGER fix_proiezione
AFTER INSERT ON Proiezione
FOR EACH ROW
BEGIN
    declare data1 date;
    declare data2 date;
    IF(NOT EXISTS(SELECT*
                  FROM Film
                  WHERE IDfilm=new.IDfilm))
    THEN delete from Proiezione
    where (IDfilm=new.IDfilm AND IDsala=new.IDsala AND Data_proiezione=new.Data_proiezione AND Ora=new.Ora);
    END IF;


    SELECT DataIn INTO data1
    From Film
    Where IDfilm=new.IDfilm;


    SELECT DataOut INTO data2
    From Film
    Where IDfilm=new.IDfilm;

    IF ((data1 > new.Data_proiezione) OR( data2 < new.Data_proiezione))
    THEN delete from Proiezione
    WHERE IDfilm=new.IDfilm AND IDsala=new.IDsala AND Data_proiezione=new.Data_proiezione AND Ora=new.Ora;
    END IF;
END
```

```
--O3: aggiungi film

INSERT INTO Film(IDfilm,DataIn,DataOut,Titolo)
VALUES(IDfilm,DataIn,DataOut,Titolo);

--Non possiamo aggiungere film in cui DataIn è dopo DataOut

CREATE TRIGGER fix_film
AFTER INSERT ON Film
FOR EACH ROW
BEGIN
    IF(new.DataIn > new.DataOut)
    THEN delete from Film where IDfilm=new.IDfilm;
END
```

```sql
--04 aggiungere ordine,quando quantita arriva a 0

CREATE TRIGGER aggiungi_ordine
AFTER UPDATE ON Prodotti
FOR EACH ROW
BEGIN
    declare x integer;
    declare y varchar(30);

    SELECT IDdipendente INTO x
    FROM Dipendenti
    ORDER BY RAND()
    LIMIT 1;

    SELECT CAST(new.IDprodotto AS nchar) INTO y;

    IF new.quantita=0
    THEN insert into Ordine(descrizione,IDdipendente) values(y,x) ;
    END IF;
END
```

```sql
--05 aggiungi vendita prodotto

INSERT INTO VenditaProdotti(IDbar,IDProdottoVenduto,quantita_venduta,Giorno)
values(IDbar,IDProdottoVenduto,quantita_venduta,Giorno) ;

CREATE TRIGGER fix_vendita
AFTER INSERT ON VenditaProdotti
FOR EACH ROW
BEGIN
    declare x integer;
    SELECT quantita INTO x
    FROM Prodotti
    WHERE IDprodotto=new.IDProdottoVenduto;

    IF x<new.quantita_venduta
    THEN delete from VenditaProdotti WHERE IDvendita=new.IDvendita;
    ELSE update Prodotti set quantita=quantita - new.quantita_venduta WHERE IDprodotto=new.IDprodottoVenduto;
    END IF;
END
```

```sql
--O6 visualizza incasso proiezione
--Questa operazione è stata implementata con una
-- stored procedure, supponendo siano inserite le variabili
-- necessarie per la query

SELECT incasso
FROM Proiezione
WHERE IDsala=IDsala AND IDfilm=IDfilm AND Data_proiezione=Data_proiezione AND Ora=Ora ;



--O7 modifica quantita prodotti
-- x e IDprodotto variabili inserite
Update Prodotti set quantita=quantita + x where IDprodotto=IDprodotto ;
```

```sql
--08 leggere incassi prodotti di un giorno
--Questa operazione è stata implementata con una
-- stored procedure, supponendo siano inserite le variabili
-- necessarie per la query
SELECT sum(prezzo)
FROM VenditaProdotti , Prodotti
WHERE Giorno=giorno AND IDprodotto=IDprodottoVenduto;
```