

1.

a.

```

SELECT NOME
FROM REPARTO R1
WHERE NOT EXISTS (SELECT *
                   FROM MEDICO, SI_TROVA_IN
                   WHERE REPARTO = R1.NOME
                   AND CITTARESIDENZA = CITTÀ
                   AND REGIONE <> 'PIEMONTE')
AND EXISTS (SELECT *
             FROM MEDICO, SI_TROVA_IN
             WHERE REPARTO = R1.NOME
             AND CITTARESIDENZA = CITTÀ
             AND CITTÀ = 'TORINO')

```

b.

```

SELECT NOME, COUNT(*) AS NUM_MEDICI
FROM MEDICO, REPARTO
WHERE REPARTO.NOME = REPARTO
GROUP BY REPARTO.NOME
HAVING NUM_MEDICI = (SELECT COUNT(*)
                      FROM MEDICO
                      WHERE REPARTO = 'TERAPIA INTENSIVA')

```

2.

NO\_GOOD\_REP\_NO\_SOLO\_VENETO (NOME)  $\leftarrow \Pi_{\text{REPARTO}} \left( \sigma_{\text{REGIONE} \leftrightarrow \text{Veneto}} (\text{MEDICO} \bowtie_{\text{CITTARESIDENZA} = \text{CITTÀ}} \text{SI_TROVA_IN}) \right)$   $\Rightarrow$  reparti che contengono medici non veneti

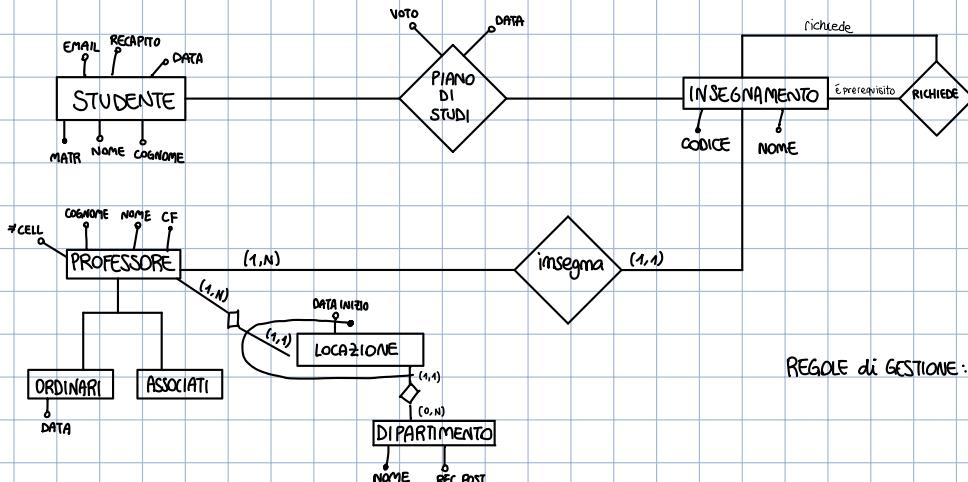
CITTÀ\_VENETO  $\leftarrow \Pi_{\text{CITTÀ}, \text{REGIONE}} \left( \sigma_{\text{REGIONE} = \text{Veneto}} \text{SI_TROVA_IN} \right)$

STATO\_DI\_FATTO  $\leftarrow \Pi_{\text{REPARTO}, \text{CITTÀ}, \text{REGIONE}} \left( \Pi_{\text{NOME}} (\text{REPARTO}) - \text{NO_GOOD_REP_NO_SOLO_VENETO} \right)$

REQUISITI  $\leftarrow \Pi_{\text{REPARTO}, \text{CITTÀ}, \text{REGIONE}} (\text{CITTÀ_VENETO} \times \text{REPARTO})$

S  $\leftarrow \Pi_{\text{REPARTO}} (\text{REQUISITI} - \text{STATO_DI_FATTO})$

3.



REGOLE DI GESTIONE: professore può cambiare dip. e tempo lo storico

Un esame è superato se la data è segnata

VOTO > 18

4. quali schedule sono serializzabili rispetto alle viste (VSR)

So:  $r_0(x) r_1(x) w_1(x) w_0(x) r_0(y) w_0(y)$

per capire a mente senza fare: es. to legge x t1 legge x e to e t1 riscriviamo i valori che hanno

legge =  $\emptyset$  ultima scrittura =  $\{(w_0(x), w_0(y))\}$  dato che  $(r_1(x), w_0(x)) \notin$  legge  $\rightarrow \notin$  VSR  
 $t_0 < t_1$

letta prima  $\rightarrow$  non tengono conto delle  
modifiche

$S_1: r_0(x) w_0(x) r_0(y) r_1(x) w_1(x) w_0(y)$

$t_0 < t_1$

legge =  $\{(r_1(x), w_0(x))\}$  ultima scrittura =  $\{(w_1(x), w_0(y))\} \rightarrow \in$  VSR  $t_0 < t_1 \rightarrow S_1': r_0(x) w_0(x) r_0(y) w_0(y) r_1(x) w_1(x)$

$S_2: r_0(x) w_0(x) r_0(y) w_0(y) r_1(x) w_1(x)$

è SERIALE

## Esercizio 1:

Sia dato il seguente schema di una base di dati relazionale contenente alcune informazioni relative alle visite eseguite dai pazienti di una data unità sanitaria locale:

*paziente(codiceTesseraSanitaria, nome, cognome, numeroTelefono, città));*  
*visita(Paziente, medico, dataInizio, oraInizio, durata);*  
*medico(CodFisc, cognome, nome, specialità).*

Si assume che ogni paziente sia identificato univocamente dal codice della sua tessera sanitaria e sia caratterizzato da nome, cognome, numero di telefono e città). Si assume che pazienti diversi possano avere lo stesso recapito telefonico. Si assume, inoltre, che ogni medico sia identificato dal suo codice fiscale e sia caratterizzato da nome, cognome e specialità. Si assume anche che per ogni specialità siano disponibili uno o più medici. Infine, si assume che ogni visita sostenuta da un paziente con un dato medico sia caratterizzata dal giorno e dall'ora in cui inizia e dalla durata. Si assume che ogni visita sia effettuata da un unico medico e che un paziente possa sostenere più visite con lo stesso medico lo stesso giorno (ovviamente in ore diverse).

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in algebra relazionale che permettano di determinare (senza usare l'operatore di divisione e usando solo se e quando necessario le funzioni aggregate):

- per ogni paziente visitato almeno una volta da un medico specializzato in Neurologia, il numero e la durata media delle visite fatte nel primo semestre 2012;
- il cognome e il nome dei medici specializzati in Ortopedia che nel mese di maggio 2013 non hanno fatto visite a pazienti di Verona;
- i medici che hanno visitato un sottoinsieme dei pazienti visitati dal medico Vincenzo Rossi (si assume che vi sia un solo medico con tale nome e cognome).

a.

$\text{PAZIENTI} \leftarrow \Pi_{\text{PAZIENTE}} \left( \sigma_{\text{SPECIALITÀ} = 'NEUROLOGIA'} \text{ (VISITA} \bowtie \text{ MEDICO)} \right)$   
medico = CodFisc

$S \leftarrow \exists_{\text{PAZIENTE COUNT}(\text{S}), \text{ AVG}(\text{DURATA})} \left( \sigma_{\text{DATA INIZIO} \text{ BETWEEN } '01.01.2012' \text{ AND } '31.06.2012'} \text{ (PAZIENTI} \bowtie \text{ VISITE)} \right)$

b.

$\text{MEDICI\_VISITE\_VERONA\_2013} \leftarrow \Pi_{\text{MEDICO}} \left( \sigma_{\text{CITTÀ} = 'VERONA' \text{ AND DATA INIZIO BETWEEN } '01.05.2012' \text{ AND } '31.05.2012'} \text{ (VISITA} \bowtie \text{ PAZIENTE)} \right)$   
PAZIENTE = CodiceTesseraSanitaria

$\text{ORTOPEDICI} \leftarrow \Pi_{\text{MEDICO}} \left( \sigma_{\text{SPECIALITÀ} = 'ORTOPEDIA'} \text{ (MEDICO)} \right)$

$S \leftarrow \text{ORTOPEDICI} - \text{MEDICI\_VISITE\_VERONA\_2013}$

c.

"Non devo trovare un paziente che non è stato visitato da Vincenzo Rossi"

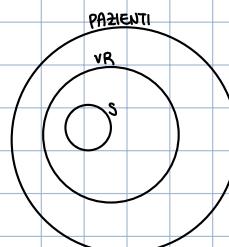
$\text{PAZIENTI\_ROSSI} \leftarrow \Pi_{\text{PAZIENTE}} \left( \sigma_{\text{NOME} = 'VINCENZO' \text{ AND } \text{COGNOME} = 'ROSSI'} \text{ (VISITA} \bowtie \text{ MEDICO)} \right)$   
medico = CodFisc

$\text{STATO\_DI\_FATTO} \leftarrow \Pi_{\text{PAZIENTE}, \text{MEDICO}} \text{ (VISITA)}$

$\text{REQUISITI} \leftarrow \Pi_{\text{PAZIENTE}, \text{MEDICO}} \text{ (PAZIENTI\_ROSSI} \times \Pi_{\text{CodFis}} \text{ (MEDICO))}$

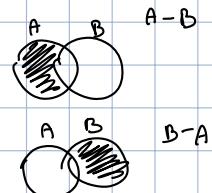
$\text{NO\_GOOD} \leftarrow \Pi_{\text{MEDICO}} \text{ (STATO\_DI\_FATTO} - \text{REQUISITI)}$

$S \leftarrow \Pi_{\text{CodFis}} \text{ (MEDICO)} - \text{NO\_GOOD}$



S.F.  
P1 | ROSSI  
P2 | ROSSI  
P3 | VRD  
P3 | ROSSI

REQ.  
P1 | ROSSI  
P2 | ROSSI  
P3 | ROSSI



## Esercizio 2:

Con riferimento all'Esercizio 1, formulare opportune interrogazioni in SQL che permettano di determinare quanto richiesto (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate).

a.

```
SELECT DISTINCT PAZIENTE, COUNT(*), AVG(DURATA) AS MEDIA
FROM VISITA, PAZIENTE, MEDICO
WHERE SPECIALITA' = 'NEUROLOGIA' AND
DATA_INIZIO BETWEEN '01.01.2012' AND '31.06.2012'
AND MEDICO = CODFISC
GROUP BY PAZIENTE
```

b.

```
SELECT CODFISC
FROM MEDICO
WHERE SPECIALITA' = 'ORTOPEDIA'
EXCEPT
SELECT MEDICO
FROM VISITA, PAZIENTE
WHERE PAZIENTE = CODICETESSERASANITARIA
AND CITTÀ = 'VERONA' AND
DATA_INIZIO BETWEEN '01.05.2012' AND '31.05.2012'
```

c.

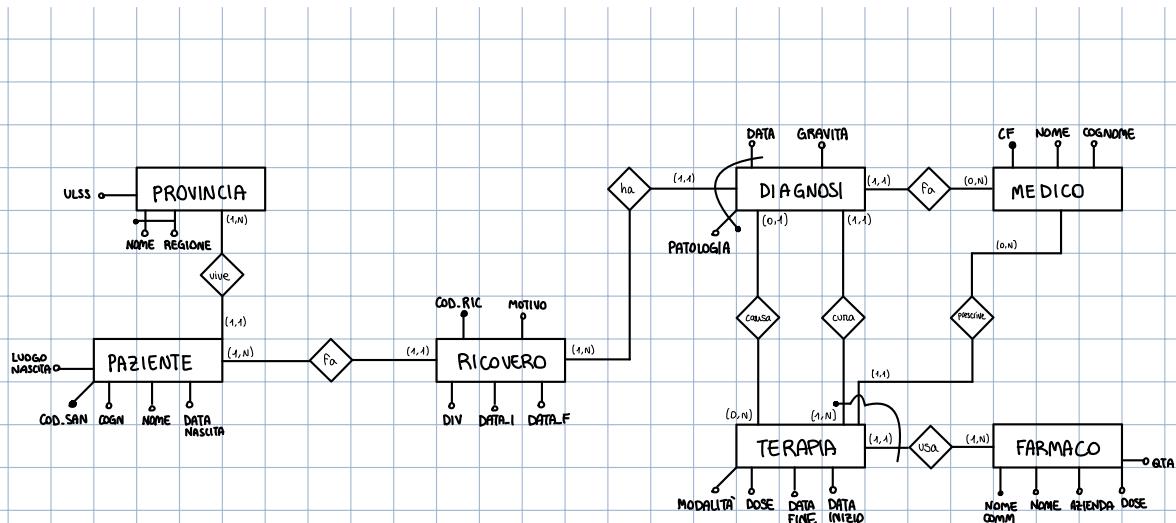
```
SELECT COD_FISC
FROM MEDICO M1
WHERE NOT EXISTS (SELECT *
                   FROM VISITA V1
                   WHERE MEDICO = M1.CF
                   AND NOT EXISTS (SELECT *
                                   FROM VISITA, MEDICO
                                   WHERE NOME = 'VINCENZO' AND
                                         COGNOME = 'ROSSI' AND
                                         PAZIENTE = V1.PAZIENTE
                               )
                )
```

### Esercizio 3:

Si voglia modellare il seguente insieme di informazioni riguardanti un sistema per la gestione delle diagnosi e delle terapie dei pazienti ricoverati in un dato ospedale.

- Di ogni ricovero, il sistema deve memorizzare il codice univoco, il nome della divisione ospedaliera (Cardiologia, Reumatologia, Ortopedia, ), il paziente ricoverato, le date di inizio e fine del ricovero e il motivo principale del ricovero.
- Di ogni paziente, il sistema deve memorizzare il codice sanitario (univoco), il cognome, il nome, la data di nascita, il luogo di nascita e la provincia di residenza. Per i pazienti residenti fuori regione, vengono memorizzati anche il nome della ULSS e la regione di appartenenza.
- Di ogni diagnosi effettuata durante il ricovero del paziente, sono memorizzati la patologia diagnosticata, col suo codice ICD10 (classificazione internazionale delle patologie) e l'indicazione della sua gravità (grave: sì/no), la data e il nome e cognome del medico che ha effettuato la diagnosi.
- Nella base di dati si tiene traccia delle terapie prescritte ai pazienti durante il ricovero. Di ogni terapia, si memorizzano il farmaco prescritto, la dose giornaliera, le date di inizio e di fine della prescrizione, la modalità di somministrazione ed il medico che ha prescritto la terapia.
- Di ogni farmaco sono memorizzati il nome commerciale (univoco), l'azienda produttrice, il nome e la quantità dei principi attivi contenuti e la dose giornaliera raccomandata.
- Si tiene, infine, traccia delle diagnosi che hanno motivato le terapie. In particolare, ogni terapia è prescritta al fine di curare una o più patologie diagnosticate. Può capitare anche che una nuova patologia (registrata come nuova diagnosi) sia causata, come effetto collaterale, da una terapia precedentemente prescritta. Tale legame causa-effetto va registrato nella base di dati.

Si definisca uno schema Entità-Relazioni che descriva il contenuto informativo del sistema, illustrando con chiarezza le eventuali assunzioni fatte. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità (identificando le possibili chiavi) e relazione. Vanno specificati accuratamente i vincoli di cardinalità e partecipazione di ciascuna relazione. Si definiscano anche eventuali regole di derivazione e/o vincoli di integrità necessari per codificare alcuni dei requisiti attesi del sistema.



## Esercizio 4:

Si consideri il seguente schema di base di dati per la gestione dell'informazione relativa a un campionato mondiale di calcio:

`PARTITA(squadra1, squadra2, goal1, goal2, data, orario, sede)`

Chiave esterna: `squadra1` → `SQUADRA(codice)`

Chiave esterna: `squadra2` → `SQUADRA(codice)`

Vincolo Not Null: `{sede, orario}`

$squadra1 \neq squadra2$

`SQUADRA(codice, nome, gruppo : {A, ..., H})`

Vincolo Not Null: `{nome, gruppo}`

Vincolo di unicità: `{nome}`

Ciascuna squadra è identificata da un codice di tre lettere (ad esempio, ESP per Spagna, BRA per il Brasile, e così via) ed è assegnata a un gruppo di squadre, denotato da una lettera da A a H. Ciascuna tupla di `PARTITA` memorizza i dati di una partita “`squadra1–squadra2`” (in quest’ordine). Gli attributi “`goal1`” e “`goal2`” rappresentano i goal segnati dalla prima e dalla seconda squadra, rispettivamente.

Si traduca lo schema relazionale dato in SQL usando opportuni domini e includendo tutti i vincoli d’integrità sopra specificati.

```
CREATE TABLE SQUADRA (
  CODICE CHAR(3) PRIMARY KEY,
  NOME VARCHAR(50) NOT NULL UNIQUE,
  GRUPPO CHAR(1) NOT NULL CHECK (GRUPPO BETWEEN A AND H)
)
```

```
CREATE TABLE PARTITA (
  SQUADRA1 CHAR(3) REFERENCES SQUADRA(CODICE) ON UPDATE CASCADE ON DELETE RESTRICT,
  SQUADRA2 CHAR(3) REFERENCES SQUADRA(CODICE) ON UPDATE CASCADE ON DELETE RESTRICT CHECK (SQUADRA1 <> SQUADRA2),
  GOAL1 INT CHECK (GOAL1 > 0),
  GOAL2 INT CHECK (GOAL2 > 0),
  DATA DATE,
  ORARIO TIME NOT NULL,
  SEDE VARCHAR(50) NOT NULL,
  PRIMARY KEY(SQUADRA1, SQUADRA2, DATA)
)
```

### Esercizio 1:

Sia dato il seguente schema relazionale relativo alle filiali di una data banca presenti sul territorio nazionale:

*Filiale(CodiceFiliale, Città, Direttore, TotaleDepositi); chiavi candidate: direttore*

*SiTrovaIn(Città, Regione);*

*CCclienti(CodiceFiscale, Filiale, NumeroCC).*

Si assuma che ogni filiale sia identificata univocamente dal suo codice e sia caratterizzata dalla città in cui si trova, dal codice fiscale del direttore e dall'ammontare complessivo di denaro depositato presso di essa (attributo *TotaleDepositi*). Si assuma che filiali diverse abbiano direttori diversi. Si assuma anche che la banca possa avere più di una filiale in una data città. Ogni città sia identificata univocamente dal suo nome. Ogni cliente sia identificato unicamente dal suo codice fiscale e possa possedere uno o più conti correnti presso una o più filiali. Ogni conto corrente sia identificato univocamente dal suo numero, sia associato ad una sola filiale e abbia un unico proprietario.

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in algebra relazionale che permettano di determinare (senza usare l'operatore di divisione e usando solo se necessario le funzioni aggregate):

- (a) i clienti che possiedono conti correnti solo in filiali della banca con sede in città della regione Veneto;
  - (b) per ogni città con almeno 3 filiali, il numero di filiali con un valore dell'attributo *TotalDepositi* maggiore di 50000000 di euro;
  - (c) i clienti che hanno un conto corrente in tutte le filiali in cui ha un conto corrente il cliente MLNGVN11S11H333P.

a. CANDIDATI  $\leftarrow$  TI(CODICEFISC ((O REGIONE = 'VENETO' (FILIALE IN SITROVAINI))  $\bowtie$  CLIENTE) FILIALE = CODICEFILAIE)

NO.GOOD  $\leftarrow$  TI CODICEFISC ((O REGIONE  $\leftrightarrow$  'VENETO' (FILIALE  $\bowtie$  SITROVAIN))  $\bowtie$  CLIENTE)  
FILIALE = CODICEFILAIE

$S \leftarrow \text{CANDIDATI - NO-GOOD}$

b. ALMENO 3  $\rightarrow$   $A \times A \times A$

FILIALE1 (..1,..1,...)  $\leftarrow$  FILIALE

FILIALE 2 (..2,..2,...)  $\leftarrow$  FILIALE

$S \leftarrow \text{CITTÀ} \setminus \text{COUNT}(\text{*}) \left( \omega \right)$  TOTALE DEPOSITI = 5000...  $\left( \omega \text{ CODICE FILIALE} < \text{CODICE FILIALE}1 \text{ AND } \text{CITTÀ} = \text{CITTÀ}1 \text{ AND } \text{CITTÀ} = \text{CITTÀ}2 \right)$   $\left( \text{FILIALE} \times \text{FILIALE}1 \times \text{FILIALE}2 \right)$  /  
 OR TOTALE DEPOSITI 1 = ...  
 OR TOTALE DEPOSITI 2 = ...  $\left( \text{CODICE FILIALE}1 < \text{CODICE FILIALE}2 \text{ AND } \text{CITTÀ} = \text{CITTÀ}1 \text{ AND } \text{CITTÀ} = \text{CITTÀ}2 \right)$  /

C. CONTICLIENTE  $\leftarrow$  TI FILIALE (o codicefiscale = "MLN..." (CCCLIENTI))

STATO\_DI\_FATTO ← TT CODFISC, FILIALE (CC CLIENTI)

REQUISITI ← TT codfisc (cc clienti) x conti\_cliente

NO\_GOOD  $\leftarrow$  REQUISITI - STATO\_DL\_FATTO

$S \leftarrow \text{TI CODFIS(CC_CLIENTE)} - \text{TI CODFISC(NO_GOOD)}$

## Esercizio 4:

Si considerino le due transazioni seguenti:

```
start transaction; -- T1
  select qta from Articolo where id = 123;
  select qta from Articolo where id = 123;
  commit;

start transaction; -- T2
  update Articolo set qta = qta - 1 where id = 123;
  commit;
```

Si supponga che T1 e T2 siano sottomesse al sistema simultaneamente. Si spieghi, giustificando la risposta, quali risultati possono produrre le due `select`, nei casi in cui le due transazioni siano eseguite:

1. nel livello **Serializable**;
2. nel livello **repeatable read**;
3. nel livello **read committed**;
4. nel livello **read uncommitted**.

nel caso T1 e T2 siano sequenziali (uno dei due ordini) allora le due select hanno risultati consistenti per tutti i livelli.  
l'unica anomalia è **LETTURA INCONSENTE** ed è possibile solo quando...

T1: `select qta from Articolo...`  
T2: `update Articolo set qta =...`  
T1: `select qta from Articolo...`

nel caso di livello:  
- **READ UNCOMMITTED**: non rilevata → legge valori diversi  
- **READ COMMITTED**: non rilevata → legge valori diversi  
- **REPEATABLE READ**: la lettura incorms. NON PUÒ AVVENIRE perché 2PL stretto in lettura/scrittura → una delle due transaz. viene messa in attesa finché l'altra non fa commit

se non c'è lo snapshot

↑  
viene messa in attesa finché l'altra non fa commit

## Esercizio 5:

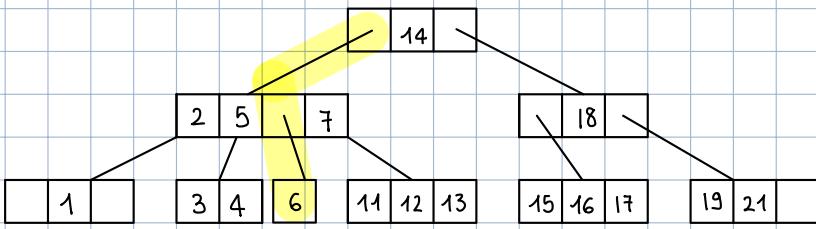
Dato il seguente insieme di chiavi:

2, 1, 18, 17, 21, 14, 13, 7, 6, 15, 12, 11, 3, 19, 5, 4, 16,

mostrare il **B-albero**, con ordine dei nodi interni  $p = 4$ , ottenuto inserendo un elemento dopo l'altro nell'ordine dato (riportando la sequenza di alberi generata dal processo di inserimento).

Successivamente, si identifichino i nodi del B-albero visitati nella ricerca di rispettivamente: (i) il record contraddistinto dal valore 9 e (ii) il record contraddistinto dal valore 6.

$q$ , ovvero il numero di puntatori per modo è:  $\lceil \frac{p}{2} \rceil \leq q \leq p$ , quindi il numero di valori è  $1 \leq q \leq 3$



cammino 6: modi 14, 2-5-7, 6

cammino 9 (anche se non c'è): modi 14, 2-5-7, 11-12-13

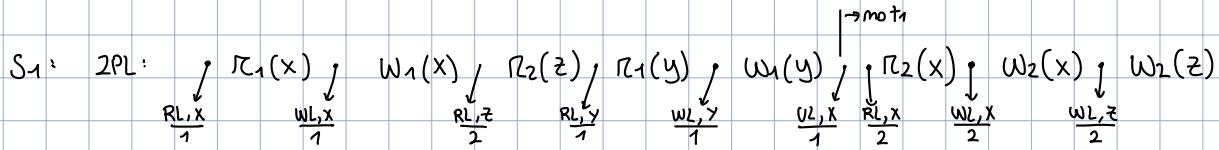
#### Esercizio 4:

Stabilire se i seguenti schedule sono o meno serializzabili rispetto al metodo del locking a due fasi, al metodo del locking a due fasi stretto e al metodo basato sui timestamp.

$s_1: r_1(x), w_1(x), r_2(z), r_1(y), w_1(y), r_2(x), w_2(x), w_2(z);$

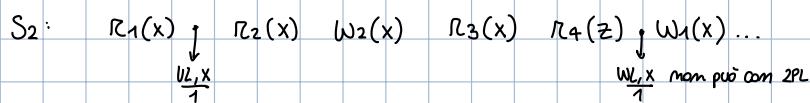
$s_2: r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), w_3(y), w_3(x), w_1(y), w_5(x), w_1(z), w_5(y), r_5(z);$

$s_3: r_1(x), r_3(y), w_1(y), w_4(x), w_1(t), w_5(x), r_2(z), r_3(z), w_2(z), w_5(z), r_4(t), r_5(t).$



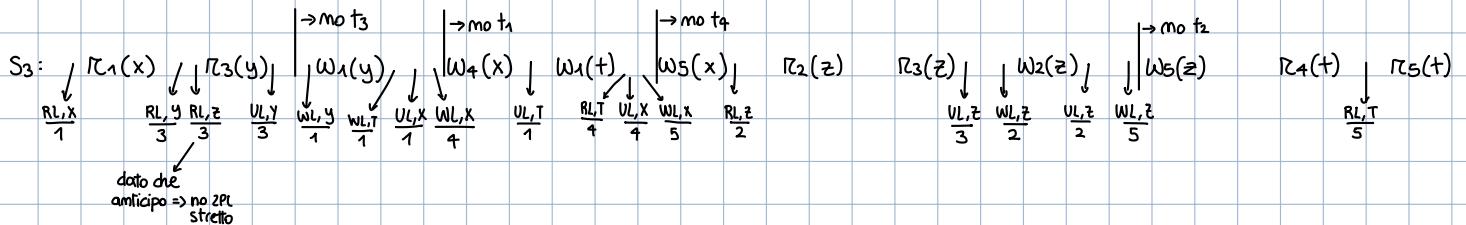
$\in$  in 2PL e 2PL stretto

$$\begin{array}{llll} \text{TIMESTAMP: } & WTM(x) = 2 & WTM(y) = 1 & WTM(z) = 2 \\ & RTM(x) = 2 & RTM(y) = 1 & RTM(z) = 2 \end{array} \Rightarrow \in \text{TIMESTAMP}$$



$S_1 \notin$  2PL, 2PL stretto

$S_1 \notin$  TIMESTAMP



$S_3 \in$  2PL  $\notin$  2PL perché anticipa i lock

$S_3 \notin$  TIMESTAMP perché  $(R_2(y), W_1(y))$

## Esercizio 5:

Si consideri la seguente implementazione (incompleta) di una specializzazione parziale ed esclusiva:

```
create table Documento (
    codice dom_codice primary key,
    titolo dom_titolo not null
);

create table PDF (
    codice dom_codice primary key references Documento,
    pagine dom_pagine not null
);

create table Video (
    codice dom_codice primary key references Documento,
    durata dom_durata not null
);
```

Si spieghi in che modo è possibile implementare il vincolo di esclusività in SQL, descrivendo anche il codice corrispondente.

Si utilizzano due TRIGGER che verificano all'inserimento/modifica che nell'altra tabella figlio non esista già un valore corrispondente

controllo nei PDF:

```
create or replace function controlla_pdf()
returns trigger language plpgsql as $$
declare
    cod dom_codice;
begin
    select codice into cod from PDF where new.codice = codice;--new è un oggetto Video
    if not found then
        return new;
    else
        raise exception 'C'è già un PDF con lo stesso codice';
        return null;
    end if;
end;
$$;

create trigger controllaPDF
before insert or update on Video
for each row
execute procedure controlla_pdf();
```

```
create or replace function controlla_video()
returns trigger language plpgsql as $$
declare
    cod dom_codice;
begin
    select codice into cod from Video where codice = new.codice;
    if not found then
        return new;
    else
        return null;
    end if;
end;
$$;

create trigger controllaVideo
before insert or update on PDF
for each row
execute procedure controlla_video();
```

controllo nei PDF:

```
create or replace function controlla_pdf()
returns trigger language plpgsql as $$
declare
    cod dom_codice;
begin
    select codice into dom_codice from PDF where new.codice = codice;
    if not found then
        return new; //inserimento corretto
    else
        raise exception 'C'è già un PDF con questo codice';
        return null;
    end if;
end;
$$;
```

create trigger controllaPDF
after update or insert on Video
for each row
execute procedure controlla\_pdf();

analogo per i DVD

## Esercizio 1:

Sia dato il seguente schema relazionale per la gestione di una classe di una scuola media superiore:

STUDENTE(codiceFiscale, nome, cognome, annoDiNascita);

MATERIA(nome, docente, oreSettimanali);

RISULTATO(studente, materia, voto).

Si assume che ogni studente sia identificato univocamente dal suo codice fiscale e sia caratterizzato da nome, cognome e anno di nascita. Si assume, inoltre, che ogni materia sia identificata univocamente dal nome e sia caratterizzata da un docente e dal numero di ore settimanali. Si assume che non vi possano materie diverse con lo stesso nome e che un docente possa insegnare più materie. Attraverso la relazione RISULTATO si vuol registrare il voto finale (pagella) ottenuto dai diversi studenti in ciascuna materia. Si assume che il valore dell'attributo voto sia un numero compreso tra 0 e 10.

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in algebra relazionale che permettano di determinare (senza usare l'operatore di divisione e usando solo se e quando necessario le funzioni aggregate):

- (i) gli studenti che hanno ottenuto un voto insufficiente (minore o uguale a 5) in due materie;  $\leftarrow$  se esattamente 2 = almeno 2 - almeno 3
- (ii) gli studenti che hanno ottenuto il voto più alto in almeno una materia;
- (iii) gli studenti che hanno ottenuto la sufficienza (voto maggiore o uguale a 6) in un soprainsieme delle materie in cui Matteo Rosso ha ottenuto la sufficienza.

i)  $RISULTATO_1(\text{studente}_1, \text{materia}_1, \text{voto}_1) \leftarrow RISULTATO$

$$S \leftarrow \Pi_{\text{STUDENTE}} \left( \begin{array}{l} \sigma_{\text{studente} = \text{studente}_1} \\ \text{AND } \text{materia} \leftrightarrow \text{materia}_1 \\ \text{AND } \text{voto} \leq 5 \end{array} \right) (RISULTATO \times RISULTATO_1)$$

ii)  $RISULTATO_1(\text{studente}_1, \text{materia}_1, \text{voto}_1) \leftarrow RISULTATO$

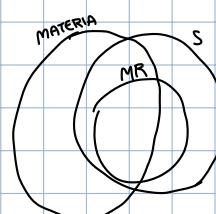
$$NO\_GOOD(\text{STUDENTE}, \text{MATERIA}) \leftarrow \Pi_{\text{STUDENTE}, \text{MATERIA}} \left( \begin{array}{l} \sigma_{\text{studente} < \text{studente}_1} \\ \text{AND } \text{materia} = \text{materia}_1 \\ \text{AND } \text{voto} \leq \text{voto}_1 \end{array} \right) (RISULTATO \times RISULTATO_1)$$

$$R \leftarrow \Pi_{\text{STUDENTE}, \text{MATERIA}}(RISULTATO) - NO\_GOOD$$

$$S \leftarrow \Pi_{\text{STUDENTE}}(R)$$

iii) "Studenti con le sufficienze di Matteo Rosso"

$$MAT\_ROSSO \leftarrow \Pi_{\text{MATERIA}} \left( \begin{array}{l} \sigma_{\text{voto} \geq 6} \\ \text{AND } \text{NOME} = \text{MATTEO} \\ \text{AND } \text{COGNOME} = \text{ROSSO} \end{array} \right) (STUDENTE \bowtie RISULTATO)$$



$$REQUISITI(\text{STUDENTE}, \text{MATERIA}) \leftarrow \Pi_{\text{CF}}(\text{STUDENTE}) \times MAT\_ROSSO$$

$$STATO\_DI\_FATTO(\text{STUDENTE}, \text{MATERIA}) \leftarrow \Pi_{\text{STUDENTE}, \text{MATERIA}}(\sigma_{\text{voto} \geq 6}(RISULTATO))$$

$$NO\_GOOD \leftarrow REQUISITI - STATO\_DI\_FATTO$$

$$VALIDI \leftarrow \Pi_{\text{CF}}(\text{STUDENTE}) - \Pi_{\text{STUDENTE}}(NO\_GOOD)$$

## Esercizio 2:

Con riferimento all'Esercizio 1, formulare opportune interrogazioni in SQL che permettano di determinare quanto richiesto (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate).

i) `SELECT DISTINCT R.studente  
FROM RISULTATO R, RISULTATO R1  
WHERE R.studente = R1.studente  
AND R.materia <> R1.materia  
AND R.voto <= 5`

ii) SELECT studente  
FROM RISULTATO R1  
WHERE voto > ALL (SELECT voto  
FROM RISULTATO  
WHERE studente <> R1.studente  
AND materia = R1.materia  
)

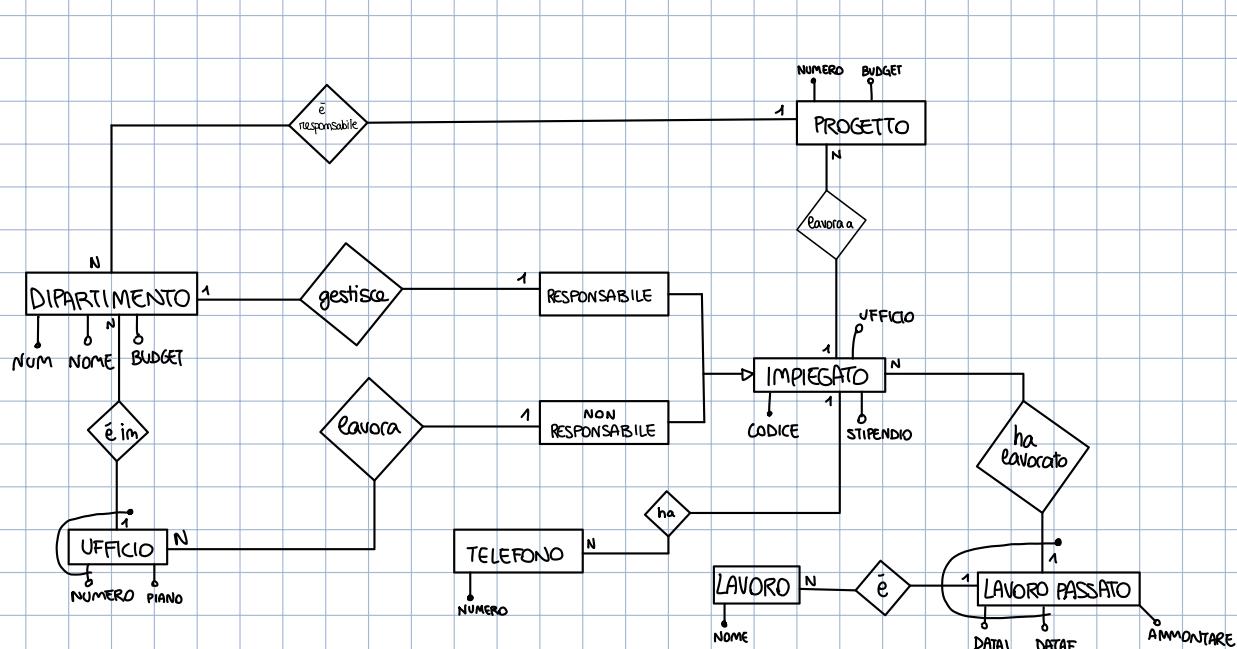
iii) SELECT codiceFiscale  
FROM STUDENTE S1  
WHERE NOT EXIST( SELECT \*  
FROM RISULTATO R1, STUDENTE  
WHERE NOME = 'MATTED' AND  
COGNOME = 'ROSSO' AND  
CF = R1. STUDENTE AND  
R1.VOTO >= 6 AND NOT EXIST( SELECT \*  
FROM RISULTATO  
WHERE STUDENTE = S1. CODICEFISCALE  
AND R1. MATERIA = MATERIA  
AND VOTO >= 6  
))

### Esercizio 3:

Si vuole progettare una base di dati per la gestione del personale di un'azienda. L'azienda è organizzata in un insieme di dipartimenti. Ogni dipartimento è caratterizzato da un insieme di impiegati, un insieme di progetti e un insieme di uffici. Ogni impiegato ha una storia lavorativa (insieme dei lavori svolti in passato). Inoltre, per ciascuno di tali lavori, l'impiegato ha una storia retributiva (insieme degli stipendi percepiti nel periodo durante il quale ha svolto un certo lavoro). Ogni ufficio ha un insieme di telefoni. La base di dati dovrà contenere le seguenti informazioni.

- Per ogni dipartimento, il numero, il nome, il budget e il codice dell'impiegato responsabile.
- Per ogni impiegato, il codice dell'impiegato, il numero del progetto cui sta attualmente lavorando, il numero dell'ufficio e il numero del telefono a lui assegnato (si assuma che ogni impiegato disponga di un telefono e che ogni telefono sia assegnato ad un impiegato). Inoltre, per ogni lavoro svolto in passato, il tipo di lavoro svolto, più la data di inizio, la data di fine e l'ammontare complessivo del compenso ricevuto per tale lavoro.
- Per ogni progetto, il numero del progetto, il numero del dipartimento responsabile del progetto e il budget. Il budget di ciascun dipartimento deve essere maggiore o uguale alla somma dei budget dei progetti da esso coordinati.
- Per ogni ufficio, il numero dell'ufficio (che identifica univocamente l'ufficio nell'ambito del dipartimento cui appartiene; ad esempio, ufficio 23 del dipartimento 5), il piano ove si trova (i assuma che tutti i gli uffici di tutti i dipartimenti si trovino in uno stesso edificio a più piani) e i numeri di tutti i telefoni presenti nell'ufficio. Si assuma, infine, che più impiegati possano condividere uno stesso ufficio.

Si definisca uno schema Entità-Relazioni che descriva il contenuto informativo del sistema, illustrando con chiarezza le eventuali assunzioni fatte. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità (identificando le possibili chiavi) e relazione. Vanno specificati accuratamente i vincoli di cardinalità e partecipazione di ciascuna relazione.



## Esercizio 5:

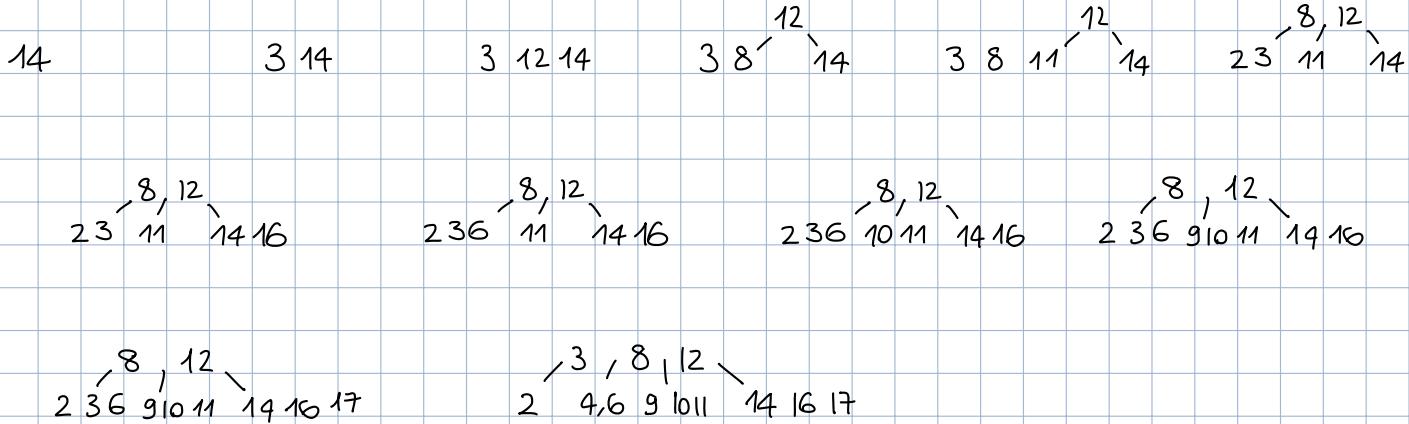
Dato l'insieme di chiavi:

14, 3, 12, 8, 11, 2, 16, 6, 10, 9, 17, 4

mostrare il  $B$ -albero, con ordine dei nodi interni  $p = 4$ , ottenuto inserendo un elemento dopo l'altro nell'ordine dato (riportando la sequenza di alberi generata dal processo di inserimento).

Successivamente, si illustrino i passi eseguiti rispettivamente nella ricerca di: (i) il record contraddistinto dal valore 13 e (ii) il record contraddistinto dal valore 4.

$$p=4 \quad 2 \leq m \leq 3$$



i) parte arriva fino al punti dopo 12, scende e trova 14  $\rightarrow$  non trovato

ii) parte, arriva fino a 8, scende nel ramo tra 3 e 8 e trova 4  $\rightarrow$  trovato

## Esercizio 4:

Sia data la seguente istanza di una base di dati sullo schema  $T(A, B)$ :

A	B
a	7
a	3
b	9
b	2

Si assuma che le due seguenti transazioni siano sottomesse al sistema simultaneamente:

### Transazione 1:

```
start transaction;
set transaction isolation level serializable;
create view X as select sum(B) as somma from T where A = 'a';
insert into T(A,B) select 'b', somma from X;
commit;
```

### Transazione 2:

```
start transaction;
set transaction isolation level serializable;
create view Y as select sum(B) as somma from T where A = 'b';
insert into T(A,B) select 'a', somma from Y;
commit;
```

Si determini quali istanze possono risultare a valle dell'esecuzione concorrente delle due transazioni.

## Esercizio 1:

Sia dato il seguente schema relazionale relativo a film e attori:

$FILM(CodiceFilm, Titolo, Regista, Anno)$ ;

$ATTORE(CodiceAttore, Cognome, Nome, Sesso, DataNascita, Nazionalità)$ ;

$INTERPRETAZIONE(Film, Attore, Personaggio)$

Ogni film sia identificato univocamente da un codice e sia caratterizzato da un titolo, da un regista e dall'anno in cui è uscito. Per semplicità, si assuma che ogni film sia diretto da un unico regista e che un regista sia identificato univocamente dal suo Cognome. Non si escluda la possibilità che film diversi abbiano lo stesso titolo (è questo il caso, ad esempio, dei remake). Ogni attore sia identificato univocamente da un codice e sia caratterizzato da un nome, un cognome, un sesso, una data di nascita e una nazionalità. Più attori possano recitare in uno stesso film e un attore possa recitare in più film. Per semplicità, si assuma che in un film un attore possa interpretare un solo personaggio.

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in algebra relazionale che permettano di determinare (senza usare l'operatore di divisione e usando solo se necessario le funzioni aggregate):

- gli attori che hanno recitato in tutti i film diretti dal regista Antonioni;
- gli attori che hanno recitato in al più due film diretti dal regista Allen;
- le coppie di attori tali che esista un film in cui il primo ha recitato e il secondo no, e viceversa.

a.  $FILM\_ANTO \leftarrow \Pi_{FILM} \left( \sigma_{\text{REGISTA} = \text{ANTONINI}} \left( \Pi_{FILM \bowtie INTERPRETAZIONE} \right) \right)$   
 $\text{FILM} = \text{CodiceFilm}$

$\text{REQUISITI}(FILM, ATTORE) \leftarrow FILM\_ANTO \times \Pi_{\text{CodiceAttore}}(ATTORE)$

$\text{STATO\_DI\_FATTO} \leftarrow \Pi_{FILM, ATTORE}(INTERPRETAZIONE)$

$\text{NO\_GOOD} \leftarrow \text{REQUISITI} - \text{STATO\_DI\_FATTO}$

$S \leftarrow \Pi_{\text{CodiceAttore}}(ATTORE) - \Pi_{\text{CodiceAttore}}(\text{NO\_GOOD})$

b.  $FILM\_ATT\_ALLEN \leftarrow \Pi_{FILM, ATTORE} \left( \sigma_{\text{REGISTA} = \text{ALLEN}} \left( \Pi_{FILM \bowtie INTERPRETAZIONE} \right) \right)$   
 $\text{CodiceFilm} = \text{Film}$

AL PIÙ DUE = TUTTI - ALMENO 3

$\text{FILM\_ATT\_ALLEN\_1}(FILM1, ATTORE1) \leftarrow \text{FILM\_ATT\_ALLEN}$

$\text{FILM\_ATT\_ALLEN\_2}(FILM2, ATTORE2) \leftarrow \text{FILM\_ATT\_ALLEN}$

$\text{NO\_GOOD} \leftarrow \Pi_{ATTORE} \left( \sigma_{\text{FILM} \in \text{FILM1} \text{ AND } ((\text{FILM\_ATT\_ALLEN} \bowtie \text{FILM\_ATT\_ALLEN\_1}) \bowtie \text{FILM\_ATT\_ALLEN\_2})} \right)$   
 $\text{FILM} \in \text{FILM2}$   
 $\text{ATTORE} = \text{ATTORE1}$   
 $\text{ATTORE} = \text{ATTORE2}$

$S \leftarrow \Pi_{\text{CodiceAttore}}(ATTORE) - \text{NO\_GOOD}$

c.  ~~$SX \leftarrow \Pi_{ATTORE, ATTORE1} \left( \sigma_{\text{ATTORE} < \text{ATTORE1}} \left( \text{INT} \times \text{INT1} \right) \right) \Rightarrow \text{quelli dove l'attore a sx ha recitato e dx no}$~~

~~$DX \leftarrow \Pi_{ATTORE, ATTORE1} \left( \sigma_{\text{ATTORE} > \text{ATTORE1}} \left( \text{INT} \times \text{INT1} \right) \right) \Rightarrow \text{quelli dove l'attore a dx ha recitato e sx no}$~~

$\text{ATTORI\_FILM\_DIVERSI} \leftarrow SX \cup DX$

$\text{TUTTI\_ATTORI}(\text{ATTORE}, \text{ATTORE}) \leftarrow \sigma_{\text{ATTORE} < \text{ATTORE1}} \left( \Pi_{\text{CodiceAttore}}(\text{ATTORE}) \times \Pi_{\text{CodiceAttore}}(\text{ATTORE1}) \right)$

$\text{NO\_GOOD} \leftarrow \text{TUTTI\_ATTORI} - \text{ATTORI\_FILM\_DIVERSI}$

A1	F1	A1 F1
A1	F2	A1 F2
A1	F3	A1 F3
A2	F1	A2 F1
A2	F2	A2 F2
A2	F4	A2 F4
A3	F1	A3 F1

ATTORI-STESSI-FILM  $\leftarrow \neg \exists \text{ATTORE} \neg \exists \text{ATTORE}_1 (\text{INT} \times \text{INT1})$   
 AND FILM = FILM1

STATO-DI-FATTO  $\leftarrow \neg \exists \text{ATTORE} \neg \exists \text{ATTORE}_1 (\text{INT} \times \text{INT1})$

ATTORI-DIVERSI  $\leftarrow \text{STATO-DI-FATTO} - \text{ATTORI-STESSI-FILM}$

A1	A2
A1	A3
A2	A1
A2	A3
A3	A1
A3	A2

C. INTERPRETAZ-VERA  $\leftarrow \exists \text{FILM}, \text{ATTORE} (\text{INTERPRETAZ})$

NON-INTERPRETAZ  $\leftarrow (\exists \text{codicefilm} (\text{FILM}) \times \exists \text{codiceattore} (\text{ATTORE})) - \text{INTERPRETAZ-VERA}$

A1-A2-NO (FILM, ATTORE, NATTORE)  $\leftarrow \neg \exists \text{FILM} = \text{codicefilm} (\text{INTERPRETAZ-VERA} \times \text{NON-INTERPRETAZ})$

A1-NO-A2 (FILM2, ATTORE2, NATTORE2)  $\leftarrow \text{A1-A2-NO}$

S  $\leftarrow \neg \exists \text{ATTORE} \neg \exists \text{ATTORE}_2 (\exists \text{ATTORE}, \text{ATTORE}_2 (\neg \exists \text{ATTORE} = \text{NATTORE}_2 (\text{A1-A2-NO} \times \text{A1-NO-A2}) \text{ AND } \text{NATTORE} = \text{ATTORE}_2))$

## Esercizio 2:

Con riferimento all'Esercizio 1, formulare opportune interrogazioni in SQL che permettano di determinare quanto richiesto (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate).

a. SELECT CODICEATTORE

FROM ATTORE A

WHERE NOT EXISTS ( SELECT \*

FROM INTERPRETAZIONE, FILM F

WHERE CODICEFILM = F.FILM AND

REGISTA = ANTONIONI AND

NOT EXISTS ( SELECT \*

FROM INTERPRETAZIONE

WHERE ATTORE = A.ATTORE

AND FILM = F.FILM

)

)

b. CREATE VIEW FILM-ALLEN (FILM, ATTORE) AS

SELECT FILM, ATTORE

FROM INTERPRETAZIONE, FILM

WHERE FILM = CODICEFILM AND

REGISTA = ALLEN

SELECT CODICEATTORE

FROM ATTORE A

WHERE NOT EXISTS ( SELECT \*

FROM FILM-ALLEN 11 12 13

WHERE 11.FILM < 12.FILM AND

12.FILM < 13.FILM AND

A. CODICEATTORE = 11.ATTORE AND

B. CODICEATTORE = 12.ATTORE AND

C. CODICEATTORE = 13.ATTORE AND

)

```

C. SELECT 11.ATTORE AND 12.ATTORE
FROM INTERPRETAZIONE 11 12
WHERE 11.ATTORE < 12.ATTORE AND
11.ATTORE NOT IN (SELECT ATTORE
FROM INTERPRETAZIONE
WHERE FILM = 12.FILM
)
AND 12.ATTORE NOT IN (SELECT ATTORE
FROM INTERPRETAZIONE
WHERE FILM = 11.FILM
)

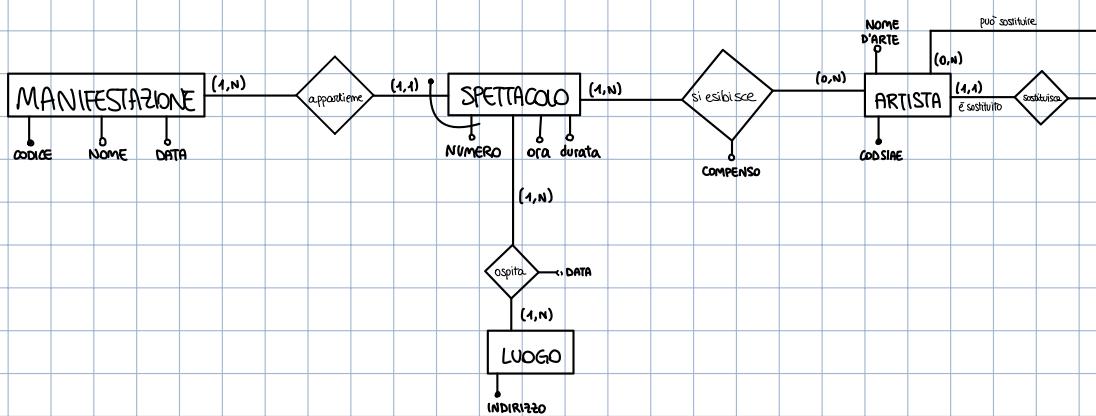
```

### Esercizio 3:

Si vuole realizzare una base di dati per gestire le manifestazioni artistiche offerte durante le stesse da una data città sulla base del seguente insieme di requisiti.

- Una manifestazione, identificata univocamente da un codice, è caratterizzata da un nome e da una data, e consiste di uno o più spettacoli.
- Ogni spettacolo è identificato da un numero, univoco all'interno della manifestazione alla quale appartiene, ed è caratterizzato dall'ora di inizio e dalla durata.
- Durante uno spettacolo, si esibiscono uno o più artisti (un artista si può esibire al massimo una volta durante lo stesso spettacolo), ricevendo un certo compenso. Uno stesso artista si può esibire in più spettacoli di una data manifestazione. Un artista è identificato univocamente dal codice SIAE e è caratterizzato da un nome d'arte. Per ogni artista, si deve indicare necessariamente un altro artista che lo sostituisca in caso di indisponibilità; un'artista può essere indicato come sostituto di più artisti.
- Per ospitare gli spettacoli vengono adibiti opportuni luoghi. In una certa data, un luogo può ospitare al massimo tre diversi spettacoli, sia della stessa manifestazione che di manifestazioni differenti.

Si definisca uno schema Entità-Relazioni che descriva il contenuto informativo del sistema, illustrando con chiarezza le eventuali assunzioni fatte. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità (identificando le possibili chiavi) e relazione. Vanno specificati accuratamente i vincoli di cardinalità e partecipazione di ciascuna relazione. Si definiscono anche eventuali regole di gestione (regole di derivazione e vincoli di integrità) necessarie per codificare alcuni dei requisiti attesi del sistema.



VINCOLI AZIENDALI: in OSPITA, per una certa DATA al massimo 3 spettacoli diversi

#### Esercizio 4:

Si svolgono i seguenti due punti.

1. Con riferimento allo standard ANSI/ISO SQL-92, si illustrino le differenze fra i due livelli di isolamento *Read Committed* e *Repeatable Read*, per quanto riguarda le anomalie permesse o meno da essi.
2. Siano  $T_1$  e  $T_2$  due transazioni sottomesse simultaneamente al sistema. Le istruzioni che le compongono vengono eseguite secondo il seguente ordine temporale:

$T_1$   
start transaction;

select stipendio from Dipendente  
where id = 'D03';

$T_2$

start transaction;

update stipendio from Dipendente  
set stipendio = stipendio - 100  
where id = 'D03';

commit;

select stipendio from Dipendente  
where id = 'D03'

commit;

Assumendo che il valore dell'attributo *stipendio* per la tupla con *id* = 'D03' sia inizialmente pari a 2000 si spieghi, giustificando la risposta, quali risultati possono produrre le due *select* di  $T_1$  nei diversi livelli di isolamento previsti dallo standard ANSI/ISO SQL-92.

1. A differenza del *Read Committed*, il *Repeatable Read* non soffre dell'anomalia della lettura inconsistente

2. fino a *Read Committed* legge 1900 perché

READ con LOCK e RILASCIO IMMEDIATO

select stipendio from Dipendente  
where id = 'D03'

start transaction;

update stipendio from Dipendente  
set stipendio = stipendio - 100  
where id = 'D03';

commit;

invece con *SERIALIZABLE* e *REPEATABLE READ* legge 2000 perché c'è lock 2PL stretto in lettura e quindi una volta che  $T_1$  acquisisce in lettura, non rilascia la risorsa fino al commit, perciò  $T_2$  non fa l'update finché  $T_1$  non fa commit

#### Esercizio 5:

Si consideri un file contenente 100.000.000 record di dimensione prefissata pari a 500 byte, memorizzati in blocchi di dimensione pari a 4096 byte in modo unspanned. La dimensione del campo chiave primaria  $V$  sia 14 byte; la dimensione del puntatore a blocco  $P$  sia 6 byte. Si chiede di confrontare fra loro le seguenti soluzioni, in termini di numero medio di accessi a blocco e di dimensione dell'indice.

- (a) Ricerca basata su un indice multilivello statico ottenuto a partire da un indice secondario costruito sul campo chiave primaria  $V$ .
- (b) Ricerca basata su un  $B$ -albero, con campo di ricerca il campo chiave primaria  $V$ , puntatore ai dati di dimensione pari a 7 byte e puntatore ausiliario di dimensione pari a 6 byte, assumendo che ciascun nodo del  $B$ -albero sia pieno al 70%.
- (c) Ricerca basata su un  $B^+$ -albero, con campo di ricerca il campo chiave primaria  $V$ , puntatore ai dati di dimensione pari a 7 byte e puntatore ausiliario di dimensione pari a 6 byte, assumendo che ciascun nodo del  $B^+$ -albero sia pieno al 70%.

M = 100.000.000 record

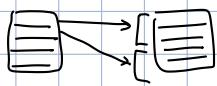
dim = 500 byte

B = 4096 byte

V = 14 byte

P = 6 byte

a. indice secondario:  $b_{fri} = \left\lceil \frac{4096}{20} \right\rceil = 204$  record



$$mbi = \left\lceil \frac{100.000.000}{204} \right\rceil = 490197$$
 blocchi

indice multilivello:  $f_0 = b_{fri} = 204$

$$2^{\text{livello}}: \left\lceil \frac{490197}{204} \right\rceil = 2403 \quad \text{ha 4 livelli} \rightarrow 5 \text{ accessi}$$

$$3^{\text{livello}}: \left\lceil \frac{2403}{204} \right\rceil = 12$$

$$4^{\text{livello}}: \left\lceil \frac{11}{204} \right\rceil = 1$$

b.  $V = 14$  byte

$P_R = 7$  byte

$$P \cdot P + (P-1)(V+P_R) \leq B \quad 6P + 21P - 21 \leq 4096 \Rightarrow P = 152$$

$$\text{effettivo} = 152 \cdot 0,7 = 106$$

$P = 6$  byte

$\sigma = 0,7$

livello	modi	dati	puntatori	
radice	1	105	106	
1	106	11130	11236	$\Rightarrow$ ha 4 livelli + 1 accesso $\Rightarrow 5$ accessi
2	11236	1179780	1191016	$\text{dim. indice} = 1202350$
3	1191016	125056680		$\hookrightarrow \Sigma \text{ modi}$

c. con B<sup>+</sup> albero

$$P \cdot P + (P-1)(V) \leq B \rightarrow 20P - 14 \leq 4096 \Rightarrow P = 205 \Rightarrow P = 143$$

$$P_{LEAF} (P_R + V) + P \leq B \rightarrow 21P_{LEAF} + 6 \leq 4096 \Rightarrow P_{LEAF} = 143 \Rightarrow P_{LEAF} = 136$$

livello	modi	dati	puntatori	
radice	1	142	143	ha 4 livelli $\Rightarrow 5$ accessi
1	143	20306	20449	$\text{dim. indice} = 2944801$
2	20449	2903758	2924207	
3	2924208	397692288		$\uparrow$ $\times P_{LEAF}!$

## Esercizio 1:

Sia dato il seguente schema relazionale relativo a medici e reparti:

MEDICO(MedicoId, Nome, Cognome, Specializzazione, Genere, AnnoNascita, Reparto, CittàResidenza);

REPARTO(Nome, Edificio, Piano, Primario);

SI\_TROVA\_IN(Città, Regione).

Si assuma che ogni medico sia identificato da un codice, che lo individua univocamente fra tutti i medici dell'ospedale, e sia caratterizzato da un nome, un cognome, una specializzazione (per semplicità, assumiamo di registrare una e una sola specializzazione per ogni medico), un genere (maschio o femmina), un anno di nascita, un reparto di appartenenza (ogni medico sia assegnato ad uno e un solo reparto) e una città di residenza.

Si assuma che ogni reparto sia identificato univocamente dal suo nome e sia caratterizzato dalla sua collocazione (edificio e piano) e dal capo reparto (primario). Si assume che un medico possa essere il primario di al più un reparto (quello al quale afferisce). Non si escluda la possibilità che due diversi reparti siano collocati nello stesso piano dello stesso edificio.

Si assuma, infine, che la tabella SI\_TROVA\_IN contenga tutte e sole le città italiane già capoluogo di provincia. Città e regioni siano identificate univocamente dal loro nome.

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in SQL che permettano di determinare quanto richiesto (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate):

- i reparti in cui sono presenti sia medici di sesso femminile che medici di sesso maschile, tutti nati dopo il 1960 (al più 59 anni);
- (FACOLTATIVO) il reparto (i reparti se più di uno) col numero più alto di medici di sesso femminile.

a. ~~SELECT NOME  
FROM REPARTO R  
WHERE EXIST (SELECT \*  
FROM MEDICO  
WHERE GENERE = FEMMINA AND  
ANNO\_NASCITA > 1960 AND  
REPARTO = R.NOME  
)  
AND EXIST (SELECT \*  
FROM MEDICO  
WHERE GENERE = MASCHIO AND  
ANNO\_NASCITA > 1960 AND  
REPARTO = R.NOME  
)~~

b. ~~SELECT REPARTO, COUNT(\*) AS NUM\_FEM  
FROM MEDICO  
WHERE GENERE = FEMMINA  
GROUP BY REPARTO  
HAVING NUM\_FEM > (SELECT COUNT(\*)  
FROM MEDICO  
WHERE GENERE = FEMMINA  
GROUP BY REPARTO  
)~~

a. ~~SELECT NOME  
FROM REPARTO R  
WHERE EXIST (SELECT \*  
FROM MEDICO M1, M2  
WHERE M1.GENERE = MASCHIO  
AND M2.GENERE = FEMMINA  
AND M1.REPARTO = R.NOME  
AND M2.REPARTO = R.NOME  
)  
AND 1960 < ALL (SELECT ANNONASCITA  
FROM MEDICO  
WHERE REPARTO = R.NOME  
)~~

## Esercizio 2:

Con riferimento all'Esercizio 1, formulare un'interrogazione in algebra relazionale che permetta di determinare quanto richiesto (senza usare l'operatore di divisione e usando solo se necessario le funzioni aggregate):

- (a) i reparti i cui medici risiedono in tutte e sole le città del Veneto.

tutte le città del Veneto

$CITTA\_VENETO \leftarrow \Pi_{CITTA} (\sigma_{REGIONE = VENETO} (SI\_TROVA\_IN))$

mo semmai elimino  
tutti i reparti

$REQUISITI (REPARTO, MEDICO_ID, CITTA\_RESIDENZA) \leftarrow \Pi_{REPARTO, MEDICO_ID} (MEDICO) \times CITTA\_VENETO$

$STATO\_DI\_FATTO \leftarrow \Pi_{REPARTO, MEDICO_ID, CITTA\_RESIDENZA} (MEDICO)$

$NO\_GOOD (NOME, MEDICO, CITTA) \leftarrow REQUISITI - STATO\_DI\_FATTO$

$TUTTO\_VENETO \leftarrow \Pi_{NOME} (REPARTO) - \Pi_{NOME} (NO\_GOOD)$

$NON\_VENETO (NOME) \leftarrow \Pi_{REPARTO} (\sigma_{REGIONE \neq VENETO} (MEDICO \in SI\_TROVA\_IN))$

CITTA\\_RESIDENZA = CITTA

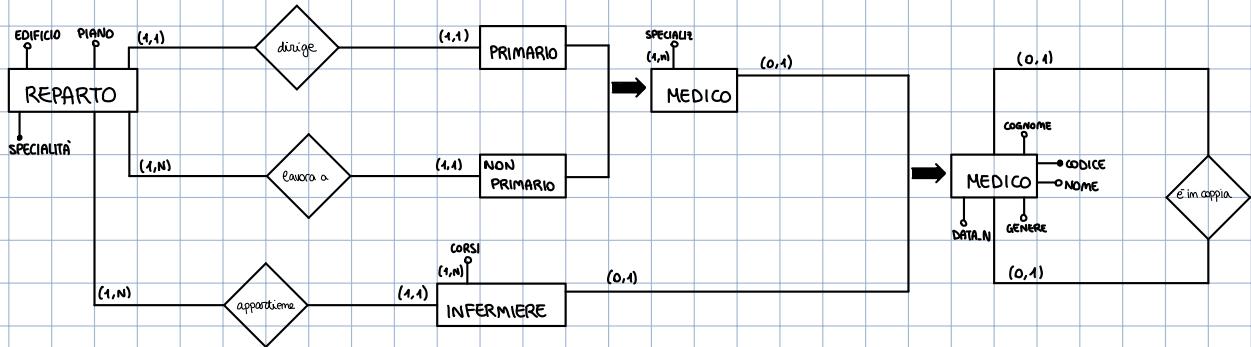
$S \leftarrow TUTTO\_VENETO - NON\_VENETO$

## Esercizio 3:

Si vuole realizzare una base di dati per la gestione di informazioni relative a reparti, medici e infermieri di un dato ospedale sulla base del seguente insieme di requisiti.

- L'ospedale sia organizzato in un certo numero di reparti. Ogni reparto sia caratterizzato da una specialità medica, che lo identifica univocamente (assumiamo che vi sia al più un reparto per ogni specialità medica), una collocazione (edificio e piano), un insieme di medici, un primario (scelto fra i medici del reparto) e un insieme di infermieri. Non escludiamo la possibilità che due reparti diversi siano collocati nello stesso piano dello stesso edificio.
- Ad ogni medico sia assegnato un codice, che lo identifica univocamente fra tutti i medici dell'ospedale. Ogni medico sia caratterizzato da un nome, un cognome, una o più specializzazioni, un genere (maschio o femmina), una data di nascita e il reparto cui appartiene (si assuma che ogni medico sia assegnato ad uno e un solo reparto).
- A ogni infermiere sia assegnato un codice, che lo identifica univocamente fra tutti gli infermieri dell'ospedale. Ogni infermiere sia caratterizzato da un nome, un cognome, un genere (maschio o femmina), una data di nascita, l'insieme dei corsi di formazione ai quali ha partecipato e il reparto cui appartiene (si assuma che ogni infermiere sia assegnato ad uno e un solo reparto).
- Si tenga traccia dei legami coniugali esistenti fra i membri del personale ospedaliero (medici e infermieri), comprendenti le coppie medico/medico, infermiere/infermiere e medico/infermiere.

Si definisca uno schema Entità-Relazioni che descriva il contenuto informativo del sistema, illustrando con chiarezza le eventuali assunzioni fatte. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità (identificando le possibili chiavi) e relazione. Vanno specificati accuratamente i vincoli di cardinalità e partecipazione di ciascuna relazione. Si definiscano anche eventuali regole di gestione (regole di derivazione e vincoli di integrità) necessarie per codificare alcuni dei requisiti attesi del sistema.



#### Esercizio 4:

Si consideri la seguente istanza di base di dati contenente la tabella **MontagneVisitate**, con chiave primaria la *coppia* (nome, data).

MontagneVisitate	
nome	data
Celva	20/06/2020
Bondone	03/05/2020
Coglians	03/05/2020

Si consideri la seguente coppia di transazioni:

T1	T2
<pre> start transaction; insert into MontagneVisitate values('Matajur', '12/12/2019');  commit; </pre>	<pre> start transaction; insert into MontagneVisitate values('Matajur', '12/12/2019');  commit; </pre>

Indicare quali sono i 4 livelli di isolamento previsti dallo standard SQL. Per ognuno di essi, indicare qual è l'esito della transazione *T2* qualora, al posto dei puntini, venga inserita l'istruzione *commit* e qualora, invece, venga inserita l'istruzione *rollback*.

i 4 livelli previsti sono: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE

Se viene inserito *commit*:

- RV: ]  
 RC: → T2 si blocca perché il lock è in 2PL stretto su WRITE, e se T1 scrive allora.  
 RR: → T2 non riesce ad inserire → T2 fa rollback  
 SR: ]

Se viene scritto *ABORT*:

- RV: ]  
 RC: → T2 si blocca perché il lock è in 2PL stretto su WRITE, e se T1 fa ROLLBACK  
 RR: → allora T2 scrive e inserisce → T2 fa commit  
 SR: ]

## Esercizio 1:

Sia dato il seguente schema relazionale relativo a film e attori:

$FILM(CodiceFilm, Titolo, Regista, Anno)$ ; chiave cand: Titolo, Anno

$ATTORI(CodiceAttore, Nome, Cognome, Sesso, DataNascita, Nazionalità)$ ;

$INTERPRETAZIONE(Film, Attore, Ruolo)$ .

Si assuma che ogni film sia identificato univocamente da un codice e sia caratterizzato da titolo, regista e anno di uscita. Per semplicità, si assume che ogni film sia diretto da un unico regista e ogni regista sia identificato univocamente dal suo cognome. Si ammetta la possibilità che vi siano film diversi con lo stesso titolo (questo è il caso, ad esempio, dei remake), ma si escluda la possibilità che due film con lo stesso titolo escano lo stesso anno. Si assume che la base di dati non contenga film privi di attori (ad esempio, film di animazione), ossia che in ogni film reciti almeno un attore.

Ogni attore sia identificato univocamente da un codice e sia caratterizzato da nome, cognome, sesso, data di nascita e nazionalità. Si assume che più attori possano recitare in un dato film e che un attore possa recitare in più film. Infine, si assume che, in ogni film, un attore possa svolgere più di un ruolo.

Definire preliminarmente le chiavi primarie, le eventuali altre chiavi candidate e, se ve ne sono, le chiavi esterne delle relazioni date. Successivamente, formulare opportune interrogazioni in algebra relazionale che permettano di determinare (senza usare l'operatore di divisione e usando solo se necessario le funzioni aggregate):

- gli attori che non hanno recitato in alcun film di Ken Loach;
- per ogni regista, il film (i film se più di uno) da lui/lei diretto col minor numero di attori;
- gli attori che hanno recitato in tutti i film di Steven Spielberg e in almeno un film di un altro regista.

a.  $NO\_GOOD(CodiceAttore) \leftarrow \Pi_{ATTORE} \left( \sigma_{\text{REGISTA} = \text{KEN LOACH}} \left( \Pi_{FILM \times INTERPRETAZIONE} \right) \right)$

$S \leftarrow \Pi_{CodiceAttore} (ATTORI) - NO\_GOOD$

b.  $REG\_FILM\_COUNT(REGISTA, FILM, NUMATTORI) \leftarrow \Pi_{REGISTA, FILM} \left( \sigma_{COUNT(ATTORE)} \left( \Pi_{REGISTA, FILM, ATTORE} (FILM \times INTERPRETAZIONE) \right) \right)$

$REG\_FILM\_COUNT\_2(REGISTA2, FILM2, NUMATTORI2) \leftarrow REG\_FILM\_COUNT$

$NO\_GOOD \leftarrow \Pi_{REGISTA, FILM} \left( \sigma_{\text{REGISTA} = \text{REGISTA2}} \left( REG\_FILM\_COUNT \times REG\_FILM\_COUNT\_2 \right) \right)$

$S \leftarrow \Pi_{REGISTA, FILM} (INTERPRETAZIONE) - NO\_GOOD$

R1 F3 5	R1 F3 5
R1 F1 10	R1 F1 10
R2 F2 20	R2 F2 20
R3 F4 7	R3 F4 7
R1 F3 5	R1 F1 10

c.  $FILM\_SPIELBERG \leftarrow \Pi_{CodiceFilm} \left( \sigma_{\text{REGISTA} = \text{STEVEN SPIELBERG}} (FILM) \right)$

REQ.	S.F.	NO GOOD
SP F1	A1 F1	A2 F2
SP F2	A1 F2	A4 F1
RB F3	A2 F1	A4 F2
LC F4	A2 F2	A2 F1
	A3 F1	A3 F1
	A3 F2	A3 F2
	A4 F1	A4 F1
	A4 F2	A4 F4

$TUTTI\_SPIEL \leftarrow \Pi_{CodiceAttore} (ATTORI) - \Pi_{CodiceAttore} (NO\_GOOD)$

$PARTECIPAZ(ATTORE, FILM) \leftarrow \Pi_{ATTORE, FILM} \left( \sigma_{\text{CodiceFilm} = \text{FILM}} (TUTTI\_SPIEL \times INTERPRETAZIONE) \right)$

$FILM\_NON\_SPIEL \leftarrow \Pi_{CodiceFilm} (FILM) - FILM\_SPIELBERG$

$S \leftarrow \Pi_{ATTORE} \left( \sigma_{\text{ATTORE} = \text{ATTORE}} \left( PARTECIPAZ \times (FILM\_NON\_SPIEL \times INTERPRETAZIONE) \right) \right)$

## Esercizio 2:

Con riferimento all'Esercizio 1, formulare opportune interrogazioni in SQL che permettano di determinare quanto richiesto (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate).

a. `SELECT CODICEATTORE`

`FROM ATTORI A`

`WHERE NOT EXISTS (SELECT *`

`FROM FILM F`

`WHERE F.REGISTA = KEN LOACH AND`

`EXISTS (SELECT *`

`FROM INTERPRETAZIONE`

`WHERE FILM = F.CODICEFILM AND`

`ATTORE = A.CODICEATTORE`

`)`

`)`

b. `SELECT REGISTA, FILM, COUNT(ATTORE) AS NUM`

`FROM FILM F, INTERPRETAZIONE`

`WHERE F.CODICEFILM = FILM`

`GROUP BY REGISTA, FILM`

`HAVING NUM <= ALL (SELECT COUNT(ATTORE)`

`FROM INTERPRETAZIONE, FILM`

`WHERE CODICEFILM = FILM AND`

`F.REGISTA = REGISTA AND`

`F.FILM ≠ FILM`

`GROUP BY FILM`

`)`

c. `CREATE VIEW FILM_SPIEL (CODICEFILM) AS`

`SELECT CODICEFILM`

`FROM FILM`

`WHERE REGISTA = SPIELBERG`

`SELECT CODICEATTORE`

`FROM ATTORI A`

`WHERE NOT EXISTS (SELECT *`

`FROM FILM_SPIEL F`

`WHERE NOT EXISTS (SELECT *`

`FROM INTERPRETAZIONE`

`WHERE FILM = F.CODICEFILM AND`

`ATTORE = A.CODICEATTORE`

`)`

`)`

`AND EXISTS (SELECT *`

`FROM FILM, INTERPRETAZIONE`

`WHERE REGISTA ≠ SPIELBERG AND`

`FILM = CODICEFILM AND`

`ATTORE = A.CODICEATTORE`

`)`

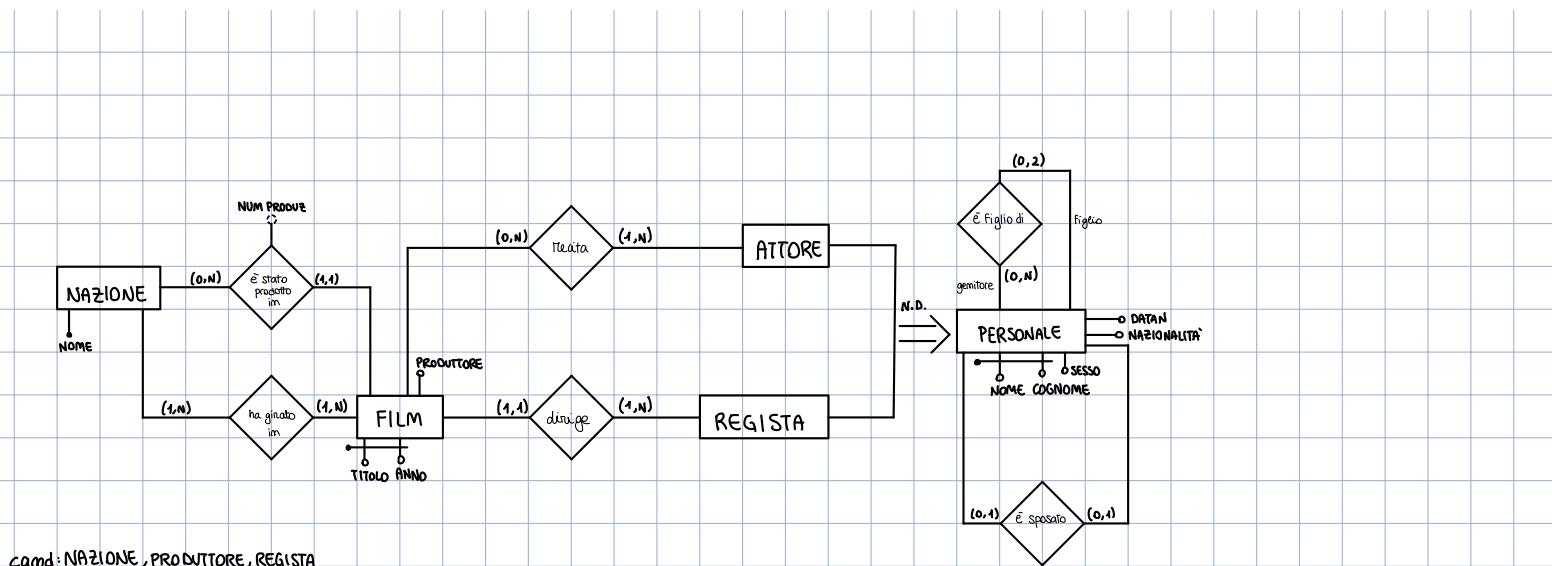
`)`

### Esercizio 3:

Si vuole realizzare una base di dati per la gestione di informazioni cinematografiche relative a film, registi e attori sulla base del seguente insieme di requisiti.

- Si assuma che sia i registi che gli attori siano identificati univocamente da nome e cognome, e siano caratterizzati dalla loro data di nascita e dalla loro nazionalità. Si assuma anche che l'insieme dei registi e l'insieme degli attori non siano necessariamente disgiunti. Si tenga traccia delle relazioni moglie/marito e genitore/figlio sull'insieme unione dell'insieme dei registi e dell'insieme degli attori (col termine figlio indichiamo genericamente figli di sesso maschile e figli di sesso femminile).
- Ogni film sia caratterizzato da titolo, anno di uscita, nazione in cui è stato prodotto, produttore, regista e attori. L'insieme dei film possa includere film d'animazione, privi di attori. Si assuma che ogni film venga prodotto in un'unica nazione, da un unico produttore, e abbia un unico regista. Non si escluda la possibilità che vi siano film diversi con lo stesso titolo (questo è il caso, ad esempio, dei remake), ma si escluda la possibilità che due film con lo stesso titolo escano lo stesso anno. Si tenga traccia delle nazioni (in generale, più di una) dove un film è stato girato.
- Infine, si tenga traccia del numero di film prodotti ogni anno in ogni nazione.

Si definisca uno schema Entità-Relazioni che descriva il contenuto informativo del sistema, illustrando con chiarezza le eventuali assunzioni fatte. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità (identificando le possibili chiavi) e relazione. Vanno specificati accuratamente i vincoli di cardinalità e partecipazione di ciascuna relazione. Si definiscano anche eventuali regole di gestione (regole di derivazione e vincoli di integrità) necessarie per codificare alcuni dei requisiti attesi del sistema.



### Esercizio 4:

Siano dati i seguenti schedule:

- $s_1 : r_2(x), r_1(y), w_1(y), r_2(y), w_1(x), w_2(y);$
- $s_2 : r_1(x), r_1(y), w_2(z), w_1(z), r_2(x), w_3(x), w_3(z);$
- $s_3 : r_1(x), w_2(z), w_1(z), w_3(z), w_3(y), w_1(y).$

Si stabilisca se appartengono o meno a VSR, CSR, 2PL, 2PL stretto e TS. Nel caso di schedule appartenenti a VSR, si forniscano tutti gli schedule seriali ad esso equivalenti; lo stesso nel caso di schedule appartenenti a CSR.

S1:  $r_2(x), r_1(y), w_1(y), r_2(y), w_1(x), w_2(y)$

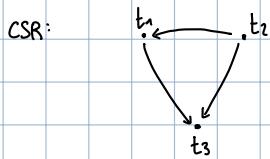
VSR: Legge =  $\{ (r_2(y), w_1(y)) \} \Rightarrow t_1 < t_2$  dato che  $r_1(y), w_2(y) \notin \text{legge} \Rightarrow t_2 > t_1 \Rightarrow \text{ASSURDO} \Rightarrow \in \text{VSR}$   
 $r_2(x), w_1(x) \notin \text{legge} \Rightarrow t_2 < t_1$   
Ultima scrittura =  $\{ w_1(x), w_2(y) \}$

S2:  $r_1(x), r_1(y), w_2(z), w_1(z), \underbrace{r_2(x), w_3(x)}, \underbrace{w_3(z)}$

VSR: Legge =  $\emptyset$  dato che  $(r_1(x), w_3(x)) \notin \text{legge} \Rightarrow t_1 < t_3 \Rightarrow \in \text{VSR}$

Ultima scrittura =  $\{ w_3(z), w_3(x) \} \rightarrow t_3 > t_1$   
 $t_3 > t_2$

Schedule seriali:  $t_1 t_2 t_3 \Rightarrow r_1(x), r_1(y), w_1(z), w_2(z), r_2(x), w_3(x), w_3(z)$   
equivalenti  
 $t_2 t_1 t_3 \Rightarrow w_2(z), r_2(x), r_1(x), r_1(y), w_1(z), w_3(x), w_3(z)$   
↳ stesso legge e  
ultima s.



ACICLICO  $\Rightarrow \in \text{CSR}$  schedule seriali equiv:  $t_2 t_1 t_3 \Rightarrow w_2(z), r_2(x), r_1(x), r_1(y), w_1(z), w_3(x), w_3(z)$

2PL:  
 $\begin{array}{ccccccccc} \downarrow & \downarrow \\ r_1(x) & r_1(y) & w_2(z) & w_1(z) & r_2(x) & w_3(x) & w_3(z) \\ \text{RL, } x \\ 1 & & \text{WL, } z \\ & & \text{RL, } x \\ & & \text{UL, } z \\ & & 2 & & 2 & & & \\ & & \text{UL, } z & & \text{UL, } x & & \text{UL, } z \\ & & 1 & & 1 & & 2 & \\ & & \text{UL, } x & & \text{UL, } x & & \text{UL, } x \\ & & 2 & & 2 & & 3 & \\ & & \text{WL, } x & & \text{WL, } x & & \text{WL, } z \\ & & 3 & & 3 & & 3 & \\ & & \text{WL, } z & & \text{WL, } z & & \text{WL, } z & \\ & & 1 & & 2 & & 3 & \\ \end{array}$

$\in \text{2PL}$ ,  $\notin \text{2PL}$  stretto perché anticipa i lock e il rilascio non è tutti assieme

TS:  $\text{RTM}(x): 1$   $\text{RTM}(y): 1$   $\text{RTM}(z):$   $\notin \text{TS}$  perché  $w_2(z) w_1(z)$   
 $\text{WTM}(x):$   $\text{WTM}(y):$   $\text{WTM}(z): 2$

S3:  $r_1(x), w_2(z), w_1(z), w_3(z), w_3(y), w_1(y)$

VSR: Legge =  $\emptyset$  dato che

Ultima scrittura =  $\{ w_3(z), w_1(y) \} \rightarrow t_3 > t_2$   
 $t_1 > t_2$

Schedule seriali:  $t_2 t_1 t_3 : w_2(z), r_1(x), w_1(z), w_1(y), w_3(z), w_3(y)$  Legge =  $\emptyset$  u.s. =  $\{ w_3(z), w_3(y) \} \rightarrow$  mom è equiv.

$t_2 t_3 t_1 : w_2(z), w_3(z), w_3(y), r_1(x), w_1(z), w_1(y)$  Legge =  $\emptyset$  u.s. =  $\{ w_1(y), w_1(z) \} \rightarrow$  mom è equiv.

S3  $\notin$  VSR

## Esercizio 5:

Si consideri il seguente schema relazionale:

*MUSICISTA*(*nickname*, *eta*, *strumento*)

- PK : *nickname*
- vincolo NOT NULL: *eta*, *strumento*

*GENERE*(*gid*, *nome*)

- PK : *gid*

*PIACE*(*mid*, *gid*)

- PK : (*mid*, *gid*)
- FK : *mid* → *MUSICISTA*.*nickname*
- FK : *gid* → *GENERE*.*gid*

*SUONA\_CON*(*mid1*, *mid2*)

- PK : (*mid1*, *mid2*)
- FK : *mid1* → *MUSICISTA*.*nickname*
- FK : *mid2* → *MUSICISTA*.*nickname*

1. Scrivere il codice SQL corrispondente allo schema relazionale dato.

2. Si consideri il vincolo: "nessun musicista può suonare con un altro musicista a cui piace l'Indie".

2.1 Elencare quali operazioni e su quali tabelle possono violare questo vincolo.

2.2 Si scelga almeno una delle operazioni individuate al punto precedente e si scriva il codice SQL di un trigger che eviti la/e violazione/i provocata/e da tale operazione.

```
1. CREATE TABLE MUSICISTA(  
    NICKNAME VARCHAR(20) PRIMARY KEY,  
    ETA INTEGER NOT NULL,  
    STRUMENTO VARCHAR(50) NOT NULL  
)
```

```
CREATE TABLE GENERE(  
    GID INTEGER PRIMARY KEY,  
    NOME VARCHAR(20)  
)
```

```
CREATE TABLE PIACE(  
    MID VARCHAR(20) REFERENCES MUSICISTA,  
    GID INTEGER REFERENCES GENERE,  
    PRIMARY KEY (MID,GID) AS PK_PIACE  
)
```

```
CREATE TABLE SUONA_CON(  
    MID1 VARCHAR(20) REFERENCES MUSICISTA,  
    MID2 VARCHAR(20) REFERENCES MUSICISTA,  
    PRIMARY KEY (MID1,MID2) AS PK_SUONA  
)
```

2. 2.1. → le operazioni di INSERT, UPDATE su *SUONA\_CON*, *PIACE*, *GENERE*  
violerrebbero il vincolo

2.2 → inserimento/aggiorn. di *SUONA\_CON* su musicisti a cui piace l'Indie

```
CREATE OR REPLACE FUNCTION CONTROLLAMUSICISTA()  
RETURNS TRIGGER LANGUAGE PLPGSQL AS $$  
DECLARE  
    GENERE1 VARCHAR(20),  
    GENERE2 VARCHAR(20);  
BEGIN  
    SELECT NOME INTO GENERE1 FROM GENERE,PIACE WHERE GID=GID AND NEW.MID1 = MID;  
    SELECT NOME INTO GENERE1 FROM GENERE,PIACE WHERE GID=GID AND NEW.MID2 = MID;  
    IF GENERE1 = 'INDIE' THEN  
        RETURN NULL;  
    END IF;  
    IF GENERE2 = 'INDIE' THEN  
        RETURN NULL;  
    END IF;  
    ELSE  
        RETURN NEW;  
    END;  
END;
```

```
CREATE TRIGGER CONTROLLO_SUONA  
BEFORE INSERT OR UPDATE ON SUONA_CON  
FOR EACH ROW  
EXECUTE PROCEDURE CONTROLLAMUSICISTA()
```