

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

REQUISITI (FILM, ATTORE)	$\leftarrow (FILM_SPIELBERG \times \Pi_{CodiceAttore} (ATTORE))$	REQ	S.F.	NO GOOD
REQUISITI (FILM, ATTORE)	$\leftarrow (FILM_SPIELBERG \times \Pi_{CodiceAttore} (ATTORE))$	SP F1	A1 F1	A2 F2
		SP F2	A1 F2	A4 F1
STATO_DI_FATTO	$\leftarrow \Pi_{FILM, ATTORE} (INTERPRETAZIONE)$	RB F3	A2 F1	A4 F2
		LC F4	A2 F2	
NO_GOOD (FILM, CodiceAttore)	$\leftarrow \text{REQUISITI} - \text{STATO_DI_FATTO}$		A3 F1	
			A3 F2	
TUTTI_SPIEL	$\leftarrow \Pi_{CodiceAttore} (ATTORI) - \Pi_{CodiceAttore} (NO_GOOD)$		A4 F1	
			A4 F2	

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

$FILM_NON_SPIEL \leftarrow \Pi_{CodiceFilm} (FILM) - FILM_SPIELBERG$

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

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`

`)`

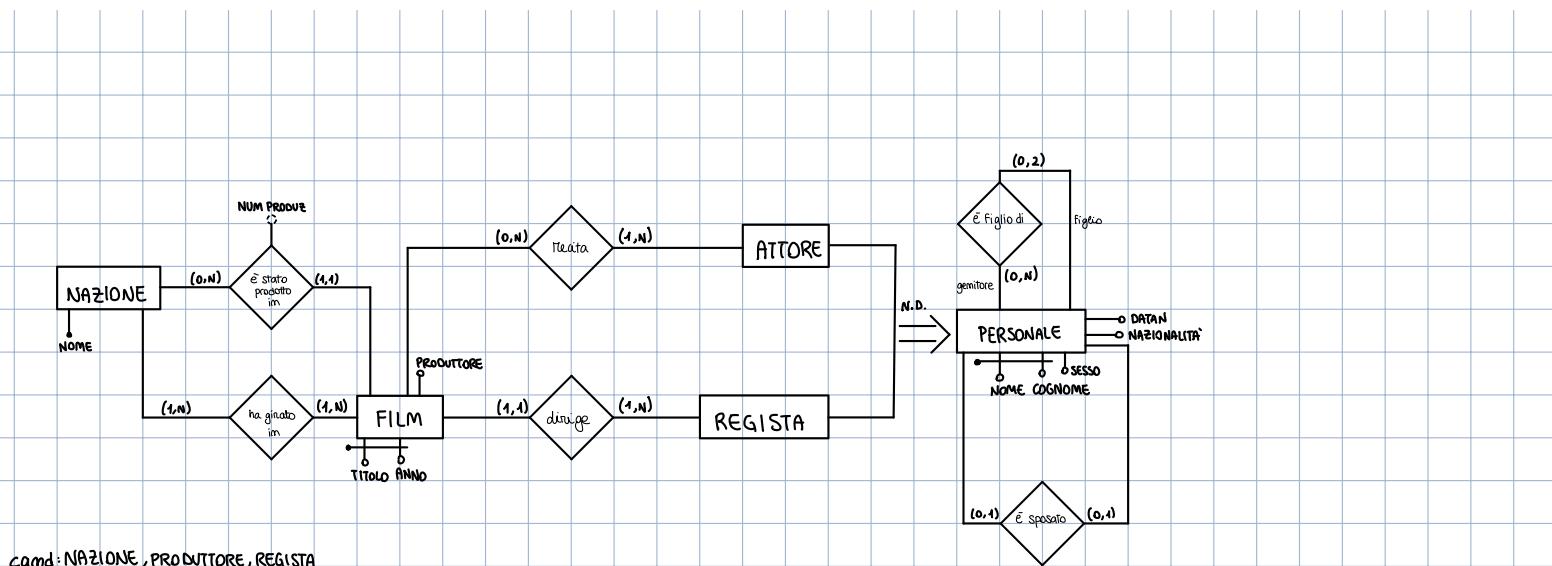
`)`

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

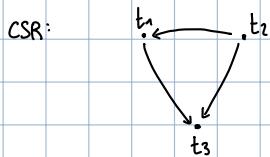
VSR: Legge = $\{ (r_2(y), w_1(y)) \} \Rightarrow t_1 < t_2$ dato che $r_1(y), w_2(y) \notin \text{legge} \Rightarrow t_2 > t_1 \Rightarrow \text{ASSURDO} \Rightarrow \in \text{VSR}$
 $r_2(x), w_1(x) \notin \text{legge} \Rightarrow t_2 < t_1$
Ultima scrittura = $\{ w_1(x), w_2(y) \}$

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

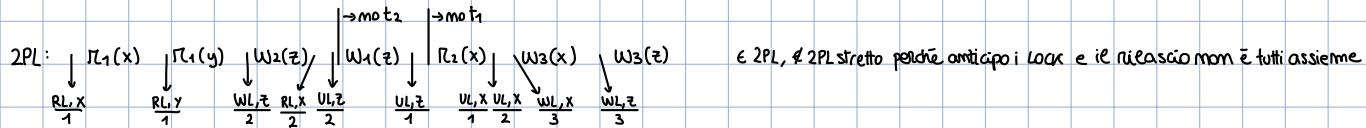
VSR: Legge = \emptyset dato che $(r_1(x), w_3(x)) \notin \text{legge} \Rightarrow t_1 < t_3 \Rightarrow \in \text{VSR}$

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: $t_2 t_1 t_3 \Rightarrow w_2(z), r_2(x), r_1(x), r_1(y), w_1(z), w_3(x), w_3(z)$



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

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

Sch. 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()
```