Esercizio 1:

Sia dato il seguente schema relazionale relativo ai prodotti offerti da un dato fornitore ai suoi clienti:
$PRODOTTO(\underline{Cod\_prodotto}, Cod\_produttore, Tipologia, Prezzo\_Unitario)$;
$CLIENTE(\underline{Cod\_cliente}, Telefono, Email, Settore)$;
$FORNISCE(\underline{Cod\_cliente, Cod\_prodotto})$.
Si assuma che ogni prodotto sia identificato univocamente da un codice e sia caratterizzato dal produttore, dalla tipologia e dal prezzo unitario. Si assuma che un produttore possa produrre più prodotti e che il fornitore possa offrire più prodotti della stessa tipologia. Si assuma che ogni cliente sia identificato univocamente da un codice e sia caratterizzato da un recapito telefonico, un recapito di posta elettronica e un settore di attività. Si assuma, infine, che il fornitore possa fornire più prodotti ad uno stesso cliente e che uno stesso prodotto possa essere fornito a più clienti. Non si escluda la possibilità che vi siano dei prodotti che non vengono richiesti da alcun cliente e dei clienti ai quali non viene fornito alcun prodotto.
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 a cui vengono forniti solo prodotti di un'unica tipologia;

(b) il cliente (i clienti se più di uno) a cui viene fornito il maggior numero di prodotti;

(c) le coppie (produttore,cliente) tali che il fornitore fornisca a quel cliente tutti e soli i prodotti di quel produttore.

a. CLIENTE_TIPO $\leftarrow \pi_{\text{COD\_CLIENTE, TIPOLOGIA}}$ (FORNISCE ⋈ PRODOTTO)

CLIENTE_TIPO1 (CLIENTE1, TIPO1) $\leftarrow$ CLIENTE_TIPO

NON_STESSO_TIPO $\leftarrow \pi_{\text{COD\_CLIENTE}}$ ($\sigma_{\text{COD\_CLIENTE = CLIENTE1 AND TIPOLOGIA < TIPOLOGIA1}}$ (CLIENTETIPO × CLIENTETIPO1))

S $\leftarrow \pi_{\text{COD\_CLIENTE}}$ (CLIENTE) − NON_STESSO_TIPO

b. CLIENTE_CONTO $\leftarrow {}_{\text{COD\_CLIENTE}}\Im_{\text{COUNT(*)}}$ (FORNISCE)

CLIENTE_CONTO_1 (COD_CLIENTE1, COUNT1) $\leftarrow$ CLIENTE_CONTO

NON_MAX $\leftarrow \pi_{\text{COD\_CLIENTE}}$ ($\sigma_{\text{COD\_CLIENTE} \neq \text{COD\_CLIENTE1 AND COUNT} \leq \text{COUNT1}}$ (CLIENTE_CONTO × CLIENTE_CONTO1))

S $\leftarrow \pi_{\text{CODICE\_CLIENTE}}$ (CLIENTE) − NON_MAX

c. CLIENTE_PROD $\leftarrow \pi_{\text{COD\_CLIENTE, COD\_PRODUTTORE}}$ (FORNISCE ⋈ PRODOTTO)

CLIENTE_PROD1 (CLIENTE1, PROD1) $\leftarrow$ CLIENTE_TIPO

NON_STESSO_PROD $\leftarrow \pi_{\text{COD\_CLIENTE}}$ ($\sigma_{\text{COD\_CLIENTE = CLIENTE1 AND COD\_PRODUTTORE < PROD1}}$ (CLIENTETIPO × CLIENTETIPO1)) => clienti che compriamo da più di un produttore

CLIENTE_UN_PROD $\leftarrow \pi_{\text{COD\_CLIENTE}}$ (CLIENTE) − NON_STESSO_PROD => clienti con solo un produttore

CLIENTE_PRODUTTORE $\leftarrow$ (CLIENTE_UN_PROD ⋈ CLIENTE_PROD)

PRODUTT_PROD (PRODUTTORE, PRODOTTO) $\leftarrow \pi_{\text{COD\_PRODUTTORE, COD\_PRODOTTO}}$ (PRODOTTO)

SF.	
C1	P1 A1
C2	P1 A2
C2	P2 A3
C3	P3 A4

PROD_PROD	
P1	A1
P1	A2
P3	A3
P3	A4

STATO_DI_FATTO ⟵ $\Pi_{COD\_CLIENTE, COD\_PRODUTTORE, COD\_PRODOTTO}$ (FORNISCE ⋈ PRODOTTO)

REQUISITI ⟵ $\Pi_{COD\_CLIENTE, COD\_PRODUTTORE, COD\_PRODOTTO}$ ($\sigma_{\sim COD\_PRODUTTORE = PRODUTTORE}$ ($\Pi_{COD\_CLIENTE}$ (CLIENTE_PRODUTTORE) × PRODUTT_PROD))

NO_GOOD ⟵ REQUISITI − STATO_DI_FATTO

S ⟵ $\Pi_{COD\_CLIENTE}$ (CLIENTE_UNI_PROD) − 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).

a.
```
SELECT C.CODICE_CLIENTE
FROM   CLIENTE C
WHERE   NOT EXIST ( SELECT *
                    FROM   FORNISCE F1, F2
                    WHERE  F1.COD_CLIENTE = C.COD_CLIENTE AND
                           F2.COD_CLIENTE = C.COD_CLIENTE AND
                           F1.PRODOTTO ≠ F2.PRODOTTO
                  )
```

b.
```
SELECT C.CODICE_CLIENTE
FROM   CLIENTE C, FORNISCE
WHERE  COD_CLIENTE = C.COD_CLIENTE
GROUP BY C.CODICECLIENTE
HAVING COUNT(*) ≥ ALL ( SELECT COUNT(*)
                        FROM FORNISCE
                        GROUP BY COD_CLIENTE
                      )
```

c.
```
SELECT  C.COD_CLIENTE, P.PRODUTTORE
FROM    CLIENTE C, PRODOTTO P, FORNISCE
WHERE    C.COD_CLIENTE = COD_CLIENTE AND
         P.COD_PRODOTTO = COD.PRODOTTO AND
          NOT EXIST ( SELECT *
                      FROM  FORNISCE, PRODOTTO P1
                      WHERE P1.FORNITORE = P.FORNITORE
                            COD_PRODOTTO = P.COD_PRODOTTO AND
                            NOT EXIST ( SELECT *
                                        FROM  FORNISCE
                                        WHERE C.COD-CLIENTE = COD_CLIENTE
                                         AND   COD_PRODOTTO = P1.COD_PRODOTTO
                                      )
                    )
EXCEPT
SELECT  C.COD_CLIENTE, P.PRODUTTORE
FROM    CLIENTE C, PRODOTTO P, FORNISCE
```

```
WHERE   C.COD_CLIENTE = COD_CLIENTE AND
        P.COD_PRODOTTO = COD_PRODOTTO AND
        EXIST ( SELECT *
                FROM   FORNISCE F1, F2, PRODOTTO P1, P2
                WHERE  F1.COD_CLIENTE = C.COD_CLIENTE AND
                       F2.COD_CLIENTE = C.COD_CLIENTE AND
                       F1.COD_PROD = P1.COD_PRODOTTO AND
                       F2.COD_PROD = P2.COD_PRODOTTO AND
                       P1.COD_PRODUTTORE < P2.COD_PRODUTTORE
              )
```

Esercizio 3:

Si consideri il seguente schema relazionale:

autore(<u>codice_fiscale</u>, nome, cognome)

libro(<u>codice_isbn</u>, titolo, anno)

ha_scritto(<u>codice_fiscale_autore</u>, <u>codice_isbn_libro</u>)

CE: *ha_scritto*.codice_fiscale_autore → *autore*

CE: *ha_scritto*.codice_isbn_libro → *libro*

Tenendo presente che ogni libro ha almeno un autore e che, in generale, può avere più autori:

1. si elenchino le operazioni sulla base di dati che possono comportare una violazione di tale vincolo di integrità;
2. si fornisca del codice SQL che permetta di mantenere tale vincolo tramite trigger, limitatamente ad una delle operazioni precedentemente individuate.

1. le operazioni di cancellazione su AUTORE, delete/update su HA_SCRITTO
   ↳ se elimini tutte le istanze di un libro da HA_SCRITTO questo non ha più autori

2. mantengo un autore se ha scritto almeno un libro in HA_SCRITTO

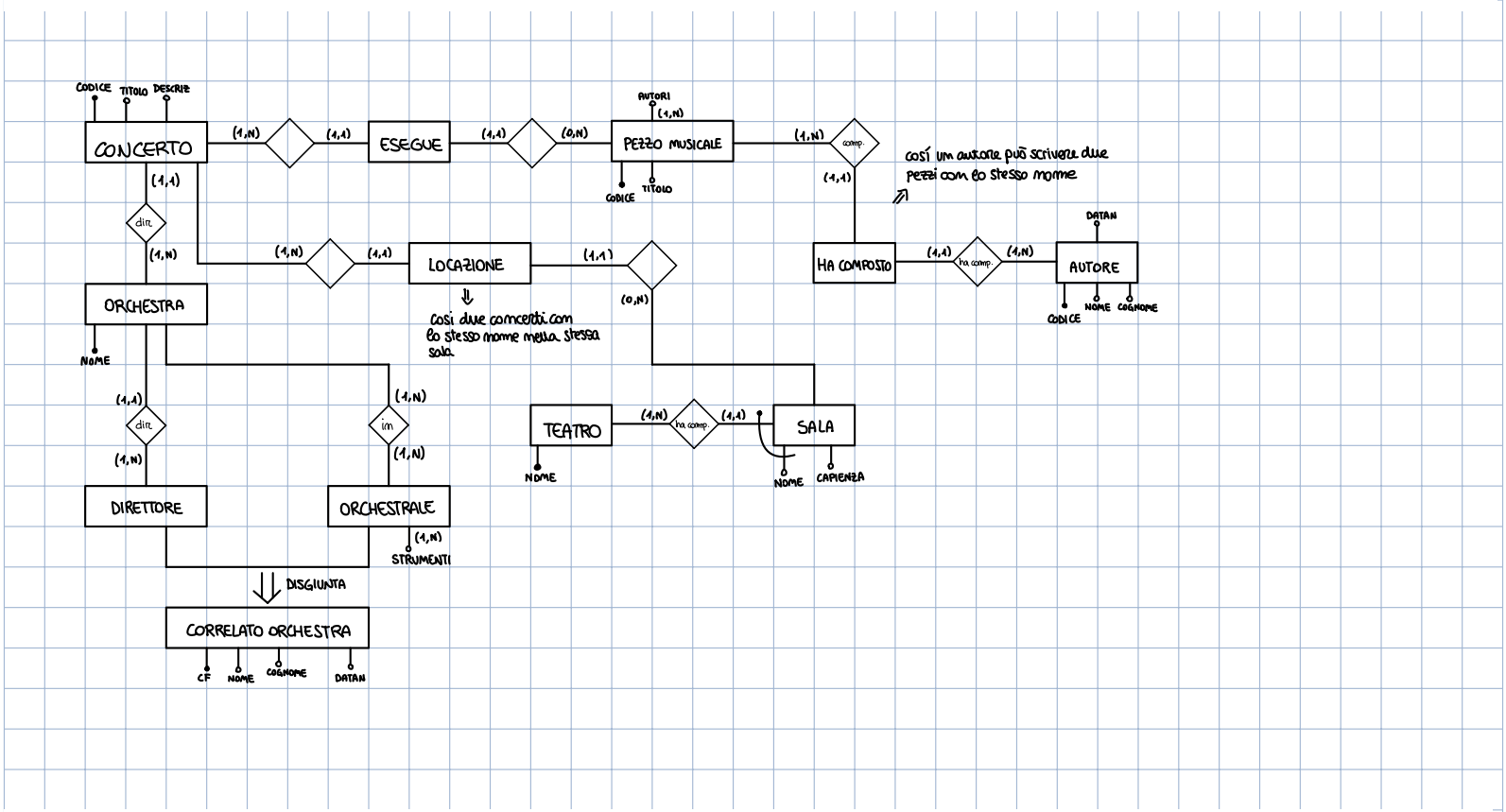
```
create or replace function controlloPresenzaLibri()
returns trigger language pgpsql as $$
define
   trovato dom_autore
begin
    select * into trovato from HA_SCRITTO where OLD.CODICE_FISCALE = CODICE_FISCALE;
    if NOT FOUND then
        return OLD;
    else
        return NULL;
   end;
$$;

create trigger controllaAutore
before delete/update on AUTORE
for each row execute procedure controlloPresenzaLibri();
```

Esercizio 4:

Si vuole realizzare una base di dati per gestire la stagione concertistica di una data città sulla base del seguente insieme di requisiti.

- La stagione concertistica si articola in una serie di concerti. Ogni concerto è identificato univocamente da un codice. Ogni concerto è contraddistinto da un titolo e da una descrizione. Si assuma che vi possano essere più concerti col medesimo titolo. Ogni concerto comprende un certo insieme di pezzi musicali, in generale di autori diversi.
- Ogni pezzo musicale ha un codice, un titolo e uno o più autori, ciascuno caratterizzato da un codice univoco, un nome, un cognome e una data di nascita. Lo stesso pezzo può essere eseguito in più concerti.
- Ogni concerto è eseguito da un'orchestra. Ogni orchestra ha un nome, che la identifica univocamente, un direttore e un insieme di orchestrali. Ogni direttore è caratterizzato da un codice fiscale, un nome, un cognome e una data di nascita, e può dirigere più orchestre. Ogni orchestrale ha un codice fiscale, un nome e un cognome, suona uno o più strumenti e può far parte di più orchestre. Un orchestrale non può dirigere un'orchestra e un direttore non può partecipare ad un'orchestra come orchestrale.
- Ogni concerto è tenuto in una sala di un teatro, in una certa data. Ogni teatro è identificato univocamente dal suo nome e contiene una o più sale. Ogni sala è identificata univocamente dal suo nome all'interno di un determinato teatro e ha una determinata capienza. Non si esclude la possibilità che vi siano sale con lo stesso nome in teatri diversi.

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 5:

Si stabilisca se i seguenti schedule appartengono o meno a 2PL, 2PL stretto, TS, CSR e VSR.

1. $s_1 : r_3(z), r_1(x), w_4(z), r_4(y), w_2(x), r_2(x), r_3(y), w_1(x), w_4(y)$;
2. $s_2 : r_2(z), w_1(t), w_3(z), r_2(x), w_4(t), r_1(y), w_2(z), w_3(y), r_1(x), w_4(z), w_2(x)$;
3. $s_3 : r_4(y), w_1(x), r_1(y), w_3(t), r_2(t), w_2(x), r_2(y), w_4(y), r_1(z), w_4(x), r_4(t), w_3(z)$.

$(r_2(x), w_1(x)) \notin legge \longrightarrow t_2 > t_1$

S_1: VSR: legge = $\{(r_2, w_2(x))\}$ dato che $(r_1(x), w_2(x)) \notin legge \rightarrow t_1 > t_2$ CONTRADDIZ

ultima scrittura = $\{w_4(z), w_1(x), w_4(y)\}$

$\notin$ VSR

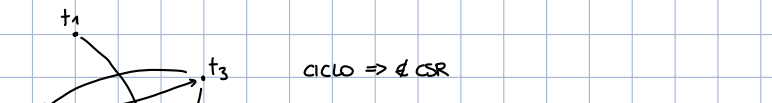
S_2: VSR: legge = $\emptyset$

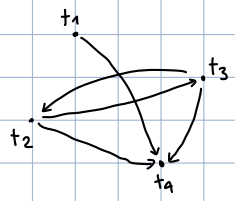
ultima scrittura = $\{w_2(x), w_4(z), w_4(t)\}$

$3<4$
$2<4$
$2<3$ => leggo da dx verso sx e vedo le variabili in comune su r(x) - w(x) e w(x) - w(x)
$1<3$
$1<2$

schedule seriale: $t_1 t_2 t_3 t_4$

CSR: $s_2 : r_2(z), w_1(t), w_3(z), r_2(x), w_4(t), r_1(y), w_2(z), w_3(y), r_1(x), w_4(z), w_2(x);$





CICLO => $\notin$ CSR

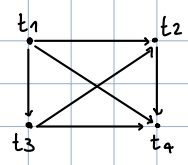
S_3: VSR:
$1<3$
$3<2$
$2<4$
$2<4$
$1<4$
$2<4$
$1<2$

legge = $\{(r_2(t), w_3(t)), (r_4(t), w_3(t))\}$

ultima scrittura = $\{w_3(z), w_4(x), w_4(y), w_3(t)\}$

seriale: 1,3,2,4

CSR: $s_3 : r_4(y), w_1(x), r_1(y), w_3(t), r_2(t), w_2(x), r_2(y), w_4(y), r_1(z), w_4(x), r_4(t), w_3(z).$



ordine: 4,2,3,1 $\in$ CSR

2PL: $\notin$ 2PL perché $w_3(t), r_2(t)$