

Esercizio 1:

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

$\text{FILM}(\underline{\text{CodiceFilm}}, \text{Titolo}, \text{Regista}, \text{Anno})$;

$\text{ATTORI}(\underline{\text{CodiceAttore}}, \text{Nome}, \text{Cognome}, \text{Sesso}, \text{DataNascita}, \text{Nazionalità})$;

$\text{INTERPRETAZIONE}(\text{Film}, \text{Attore}, \text{Ruolo})$.

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). 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. Per semplicità, si assume che, in ogni film, un attore possa svolgere un solo 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 hanno recitato in almeno due film di Avati;
- il film (i film se più di uno) col maggior numero di attori;
- gli attori che hanno recitato in almeno un film di Tarantino, ma non in tutti.

a.

$\text{FILM_AVATI} \leftarrow \Pi_{\text{CodiceFilm}} (\sigma_{\text{Regista} = \text{AVATI}} (\text{FILM}))$

$\text{ATTORI_VALIDI} \leftarrow \Pi_{\text{FILM}, \text{Attore}} (\sigma_{\text{FILM} = \text{codF}} (\text{FILM_AVATI} \bowtie \text{INTERPRETAZIONE}))$

$\text{ATTORI_VALIDI_1}(\text{FILM}, \text{Attore}) \leftarrow \text{ATTORI_VALIDI}$

$S \leftarrow \Pi_{\text{Attore}} (\sigma_{\text{FILM} < \text{FILM}_1 \text{ AND } \text{Attore} = \text{Attore}_1} (\text{ATTORI_VALIDI} \times \text{ATTORI_VALIDI_1}))$

b.

$\text{FILM_NUM} \leftarrow \Pi_{\text{FILM}} \text{ COUNT}(\text{Attore}) (\text{INTERPRETAZIONE})$

$\text{FILM_NUM_1}(\text{FILM}, \text{COUNT}_1) \leftarrow \text{FILM_NUM}$

$\text{NO_GOOD}(\text{codiceFilm}) \leftarrow \Pi_{\text{FILM}} (\sigma_{\text{FILM} \neq \text{FILM}_1 \text{ AND } \text{COUNT} < \text{COUNT}_1} (\text{FILM_NUM} \times \text{FILM_NUM}_1))$

$S \leftarrow \Pi_{\text{codiceFilm}} (\text{FILM}) - \text{NO_GOOD}$

c. almeno 1 - tutti i film

$\text{FILM_TARANTINO}(\text{FILM}) \leftarrow \Pi_{\text{CodiceFilm}} (\sigma_{\text{Regista} = \text{TARANTINO}} (\text{FILM}))$

$\text{INT_TARANTINO} \leftarrow \Pi_{\text{FILM}, \text{Attore}} (\text{INTERPRETAZIONE} \bowtie \text{FILM_TARANTINO})$

$\text{ALMENO_UNO_TAR}(\text{codiceAttore}, \text{FILM}) \leftarrow \Pi_{\text{Attore}, \text{FILM}} (\text{INT_TARANTINO})$

$\text{REQUISITI}(\text{FILM}, \text{Attore}) \leftarrow \Pi_{\text{FILM}} (\text{FILM_TARANTINO}) \times \Pi_{\text{codiceAttore}} (\text{ATTORI})$

STATO_DI_FATTO \leftarrow TT_FILM, ATTORE (INTERPRETAZIONE)

NO_GOOD (FILM, CODICEATTORE) \leftarrow REQUISITI - STATO_DI_FATTO

TUTTI_TARANTINO \leftarrow TT_CODICEATTORE (ATTORI) - TT_CODICEATTORE (NO_GOOD)

S \leftarrow TT_CODICEATTORE (ALMENO_UNO_TAR) - TUTTI_TARANTINO

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. CREATE VIEW FILM_AVATI (FILM) AS

SELECT CODICEFILM

FROM FILM

WHERE REGISTA = AVATI

SELECT CODICEATTORE

FROM ATTORI A1

WHERE EXIST (SELECT *

FROM INTERPRETAZIONE I1, I2, FILM_AVATI A1, A2

WHERE A1.CODICEATTORE = I1.CODICEATTORE AND

A1.CODICEATTORE = I2.CODICEATTORE AND

I1.FILM < I2.FILM AND

I1.FILM = A1.FILM AND

I2.FILM = A2.FILM

)

b. SELECT FILM, COUNT(ATTORE) AS NUM

FROM INTERPRETAZIONE

GROUP BY FILM

HAVING NUM >= ALL (SELECT COUNT(ATTORE)

FROM INTERPRETAZIONE

GROUP BY FILM

)

c. CREATE VIEW AS FILM_TARANTINO (FILM) AS

SELECT CODICEFILM

FROM FILM

WHERE REGISTA = TARANTINO

SELECT CODICEATTORE

FROM ATTORE A

WHERE A.CODICEATTORE IN (SELECT ATTORE

FROM INTERPRETAZIONE, FILM

WHERE FILM = CODICEFILM AND

REGISTA = TARANTINO

)

EXCEPT

```

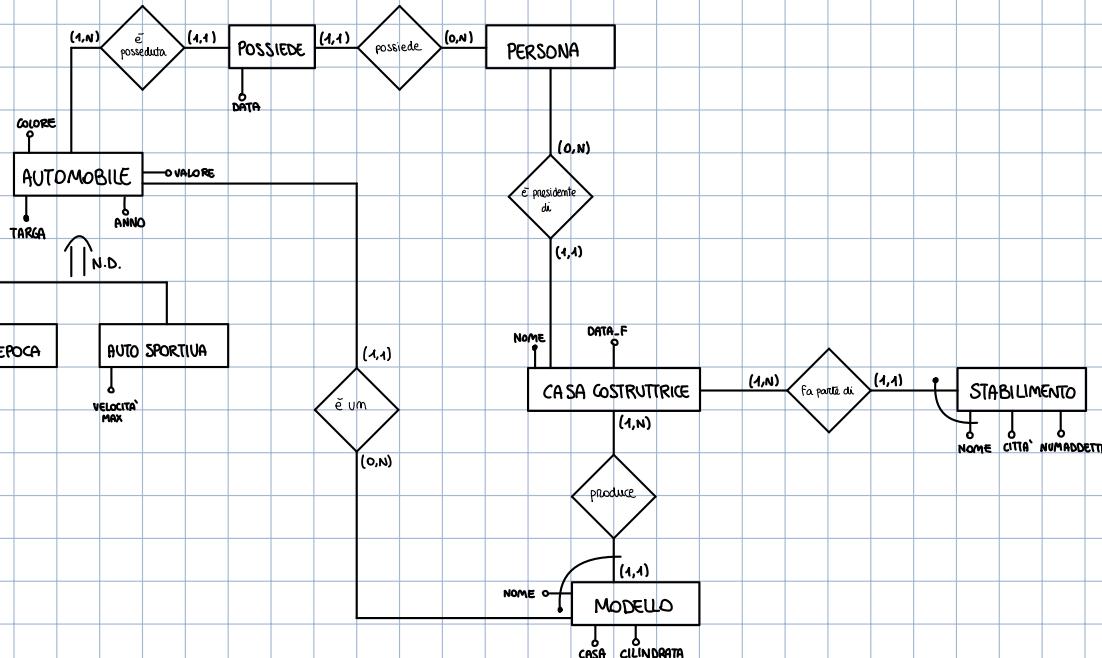
SELECT CODICEATTORE
FROM ATTORIA
WHERE NOT EXISTS (SELECT *
                   FROM FILM_TARANTINO F
                   WHERE NOT EXISTS (SELECT *
                                     FROM INTERPRETAZIONE
                                     WHERE ATTORE = A.CODICEATTORE
                                           AND FILM = F.FILM
                               )
                )
  
```

Esercizio 3:

Si vuole realizzare una base di dati per la gestione di informazioni circa un insieme di automobili caratterizzato dal seguente insieme di requisiti.

- Ogni automobile sia identificata univocamente dalla sua targa e sia caratterizzata da un modello, un anno di fabbricazione, un colore, un valore di mercato e uno o più proprietari. Fra le automobili, vogliamo tener traccia del sottoinsieme delle automobili storiche (un'auto si dice storica se sono trascorsi 25 o più anni dall'anno di fabbricazione), del sottoinsieme delle automobili sportive, caratterizzate dalla velocità massima, e del sottoinsieme delle auto storiche sportive.
- Ogni modello sia caratterizzato da un nome, una casa costruttrice e una cilindrata. Il nome identifichi univocamente il modello all'interno dei modelli proposti dalla casa costruttrice (non si esclude la possibilità che case costruttrici diverse propongano modelli, ovviamente diversi, con lo stesso nome).
- Ogni casa costruttrice sia identificata univocamente dal proprio nome e sia caratterizzata dall'anno di fondazione e da un insieme di stabilimenti. Una stessa persona possa essere presidente di più case costruttrici. Ogni stabilimento sia caratterizzato un nome, che lo identifica univocamente nell'ambito della casa costruttrice, una città ove ha sede e un numero di addetti.

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 considerino i seguenti schedule:

$$s_1 : r_1(x), r_2(x), w_1(x), r_3(x), w_3(x), w_2(y)w_1(y);$$

$$s_2 : r_2(x), r_1(x), w_1(x), r_3(x), w_1(y), w_3(x), w_2(y);$$

$$s_3 : r_2(x), w_2(y), r_1(x), w_1(x), r_3(x), w_1(y), w_3(x).$$

(a) Per ogni coppia di schedule s_i, s_j , con $1 \leq i, j \leq 3$ e $i \neq j$, stabilire se s_i e s_j sono o meno equivalenti rispetto alle viste e se sono o meno equivalenti rispetto ai conflitti.

(b) Stabilire se gli schedule dati appartengono o meno a VSR, CSR, 2PL e 2PL stretto.

a) legge $S_1 = \{(r_3(x), w_1(x))\}$

ultima scrittura $S_1 = \{w_3(x), w_1(y)\}$

legge $S_2 = \{(r_3(x), w_1(x))\}$

ultima scrittura $S_2 = \{w_3(x), w_2(y)\}$

legge $S_3 = \{(r_3(x), w_1(x))\}$

ultima scrittura $S_3 = \{w_3(x), w_1(y)\}$

VISTE $\rightarrow S_1$ e S_3 : equivalenti rispetto alle viste

CONFLITTI $\rightarrow S_1$ e S_3 :

S_1

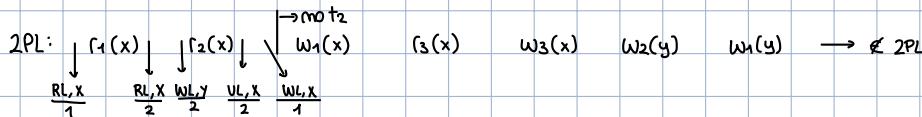
$$\begin{aligned} & r_1(x), w_3(x) \\ & r_2(x), w_1(x) \\ & r_2(x), w_3(x) \\ & w_1(x), r_3(x) \\ & w_1(x), w_3(x) \\ & w_2(y), w_1(y) \end{aligned}$$

S_3

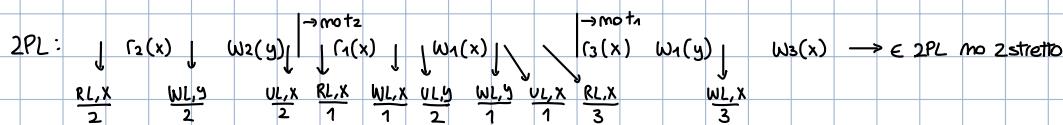
$$\begin{aligned} & r_2(x), w_1(x) \\ & r_2(x), w_3(x) \\ & w_2(y), w_1(y) \\ & r_1(x), w_3(x) \\ & w_1(x), r_3(x) \\ & w_1(x), w_3(x) \end{aligned}$$

\Rightarrow sono equivalenti rispetto ai conflitti

b) $S_1 \in$ VSR, CSR



$S_3 \in$ VSR, CSR



Esercizio 5:

1. Quali sono i due principali meccanismi forniti dal sistema di basi di dati PostgreSQL per la formulazione di vincoli complessi, non codificabili attraverso chiavi primarie, chiavi esterne o dichiarazioni di univocità e not-null? ↗
e
2. Si consideri il seguente schema relazionale:
STUDENTE(matricola, nome, cognome)
ESAME(codice, descrizione)
HA_SOSTENUTO(matricola_studente, codice_esame, valutazione, lode)

Tenendo presente che:

- si vogliono conoscere tutti i dettagli riguardanti studenti ed esami;
- la valutazione è espressa da un numero intero compreso fra 0 e 30 e la lode può essere assegnata solamente a chi ottiene una valutazione pari a 30,

si presenti del codice SQL che implementa le tre tabelle, unitamente alle chiavi primarie, alle chiavi esterne e a qualsiasi eventuale altro vincolo si ritenga necessario imporre.

1. TRIGGER com UDF e TRANSAZIONI
↳ es. CHECK
↳ im DDL ↳ im DML

2. CREATE TABLE STUDENTE (
MATRICOLA INTEGER PRIMARY KEY,
NOME VARCHAR(20),
COGNOME VARCHAR(20)
);

CREATE TABLE ESAME (
CODICE INTEGER PRIMARY KEY,
DESCRIZIONE VARCHAR(50)
);

CREATE TABLE HA_SOSTENUTO(
MATRICOLA_STUDENTE INTEGER REFERENCES STUDENTE,
CODICE_ESAME INTEGER REFERENCES TABLE,
VALUTAZIONE INTEGER CHECK (VALUTAZIONE BETWEEN 0 AND 30),
LODE BOOLEAN CHECK (NOT LODE OR VALUTAZIONE = 30)
);

Esercizio 1:

Sia dato il seguente schema relazionale relativo ai prodotti offerti da un dato fornitore ai suoi clienti:

PRODOTTO(Cod_prodotto, Cod_produttore, Tipologia, Prezzo_Unitario);

CLIENTE(Cod_cliente, Telefono, Email, Settore);

FORNISCE(Cod_cliente, Cod_prodotto).

Si assume che ogni prodotto sia identificato univocamente da un codice e sia caratterizzato dal produttore, dalla tipologia e dal prezzo unitario. Si assume che un produttore possa produrre più prodotti e che il fornitore possa offrire più prodotti della stessa tipologia. Si assume 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 assume, 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. $\text{CLIENTE_TIPO} \leftarrow \Pi_{\text{cod_cliente}, \text{tipologia}} (\text{FORNISCE} \ \Delta \ \text{PRODOTTO})$

$\text{CLIENTE_TIPO_1} (\text{CLIENTE1}, \text{TIPO1}) \leftarrow \text{CLIENTE_TIPO}$

$\text{NON_STESO_TIPO} \leftarrow \Pi_{\text{cod_cliente}} \left(\sigma_{\text{cod_cliente} = \text{CLIENTE1}} (\text{CLIENTE_TIPO} \times \text{CLIENTE_TIPO1}) \right)$

$S \leftarrow \Pi_{\text{cod_cliente}} (\text{CLIENTE}) - \text{NON_STESO_TIPO}$

b. $\text{CLIENTE_CONTO} \leftarrow \Pi_{\text{cod_cliente}} (\text{FORNISCE})$

$\text{CLIENTE_CONTO_1} (\text{cod_cliente1}, \text{COUNT1}) \leftarrow \text{CLIENTE_CONTO}$

$\text{NON_MAX} \leftarrow \Pi_{\text{cod_cliente}} \left(\sigma_{\text{cod_cliente} \neq \text{cod_cliente1}} (\text{CLIENTE_CONTO} \times \text{CLIENTE_CONTO1}) \right)$

$S \leftarrow \Pi_{\text{codice_cliente}} (\text{CLIENTE}) - \text{NON_MAX}$

c. $\text{CLIENTE_PROD} \leftarrow \Pi_{\text{cod_cliente}, \text{cod_produttore}} (\text{FORNISCE} \ \Delta \ \text{PRODOTTO})$

$\text{CLIENTE_PROD1} (\text{CLIENTE1}, \text{PROD1}) \leftarrow \text{CLIENTE_PROD}$

$\text{NON_STESO_PROD} \leftarrow \Pi_{\text{cod_cliente}} \left(\sigma_{\text{cod_cliente} = \text{CLIENTE1}} (\text{CLIENTE_PROD1} \times \text{CLIENTE_PROD1}) \right) \Rightarrow \text{clienti che comprano da più di un produttore}$

$\text{CLIENTE_UN_PROD} \leftarrow \Pi_{\text{cod_cliente}} (\text{CLIENTE}) - \text{NON_STESO_PROD} \Rightarrow \text{clienti con solo un produttore}$

$\text{CLIENTE_PRODUTTORE} \leftarrow (\text{CLIENTE_UN_PROD} \ \Delta \ \text{CLIENTE_PROD})$

$\text{PRODUTT_PROD} (\text{produttore}, \text{prodotto}) \leftarrow \Pi_{\text{cod_produttore}, \text{cod_prodotto}} (\text{PRODOTTO})$

sf.	prod_prod
C1 P1 A1	P1 A1
C2 P1 A2	P1 A2
C2 P2 A3	P3 A3
C3 P3 A4	P3 A4

STATO_DI_FATTO $\leftarrow \prod_{i=1}^n \text{cod_cliente}, \text{cod_produttore}, \text{cod_prodotto}$ (FORNISCE α PRODOTTO)

REQUISITI $\leftarrow \prod_{i=1}^n \text{cod_cliente}, \text{cod_produttore}, \text{cod_prodotto}$ ($\sigma_{\text{cod_produttore} = \text{cod_cliente}} (\prod_{i=1}^n \text{cod_cliente} (\text{CLIENTE_PRODUTTORE}) \times \text{PRODUTT_PROD})$)

NO_GOOD \leftarrow REQUISITI - STATO_DI_FATTO

S $\leftarrow \prod_{i=1}^n \text{cod_cliente} (\text{CLIENTE_UN_PROD}) - \text{NO_GOOD}$

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 EXISTS (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 EXISTS (SELECT *`

`FROM FORNISCE, PRODOTTO P1`

`WHERE P1.FORNITORE = P.FORNITORE`

`COD_PRODOTTO = P.COD_PRODOTTO AND`

`NOT EXISTS (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(codice_fiscale, nome, cognome)

libro(codice_isbn, titolo, anno)

ha_scritto(codice_fiscale_autore, codice_isbn_libro)

CE: $ha_scritto.codice_fiscale_autore \rightarrow autore$

CE: $ha_scritto.codice_isbn_libro \rightarrow libro$

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. rimaneggio un autore se ha scritto almeno un libro in HA_SCritto

create or replace function controlloPresentaLibri()

returns trigger language plpgsql as \$\$

define

trovato dorm_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 controlloAutore

before delete/update on AUTORE

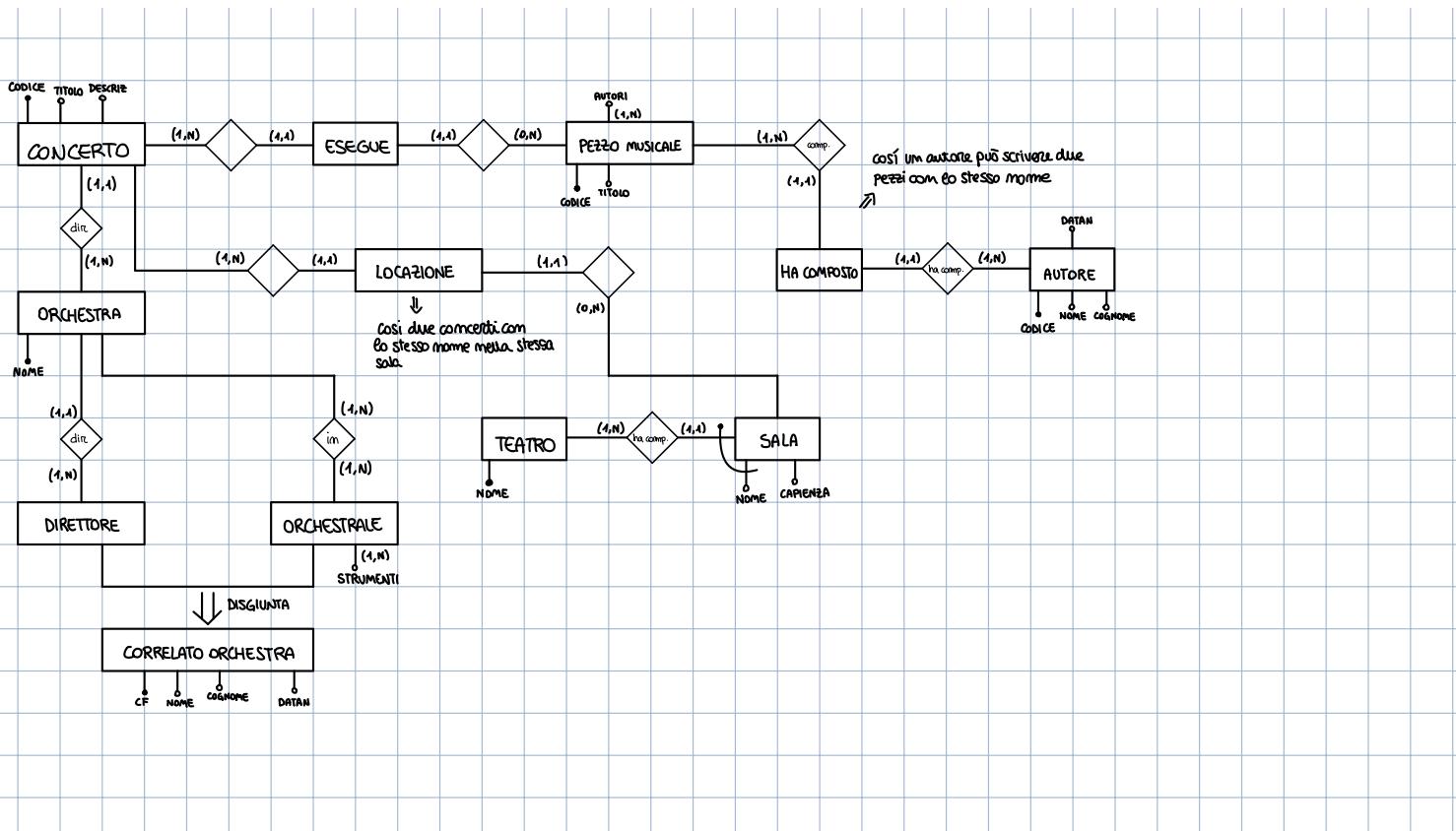
for each row execute procedure controlloPresentaLibri();

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

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

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

$\notin VSR$

$S_2: VSR: legge = \emptyset$

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

$3 < 4$

$2 < 4$

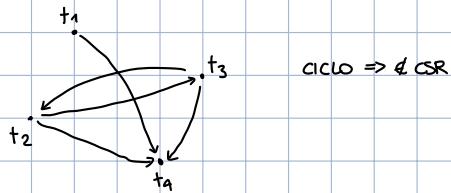
$2 < 3 \Rightarrow \text{leggo da dx verso sx e vedo le variabili in comune su } r(x) - w(x) \text{ e } w(x) - w(x)$

$1 < 3$

$1 < 2$

schedule serial: $t_1 t_2 t_3 t_4$

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



$S_3: VSR: 1 < 3 \quad \text{legge} = \{(r_2(t), w_3(t)), (r_4(t), w_3(t))\}$

$3 < 2$

$2 < 4$

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

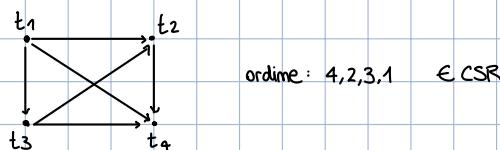
$2 < 4$

$1 < 4$ serial: 1,3,2,4

$2 < 4$

$1 < 2$

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



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

Esercizio 1:

Sia dato il seguente schema relazionale relativo ai giochi olimpici:

Olimpiade(Anno, Città, Edizione);

Atleta(CodiceAtleta, Nome, Cognome, AnnoNascita, Nazione);

HaPartecipato(Atleta, Edizione);

SiTrovaIn(Città, Nazione).

Si assuma che ogni edizione dei giochi olimpici sia identificata univocamente dal suo numero d'ordine (ad esempio, edizione numero 29 dei giochi olimpici) e sia caratterizzata dalla città che l'ha organizzata e dall'anno in cui ha avuto luogo. Si assume, inoltre, che ogni atleta sia identificato univocamente da un codice e sia caratterizzato da un nome, un cognome, un anno di nascita e una nazione di appartenenza. Si assume, infine, che ogni atleta presente nella base di dati abbia partecipato ad almeno un'edizione dei giochi olimpici e che ad ogni edizione dei giochi olimpici presente nella base di dati abbia partecipato almeno un atleta.

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 atleti che hanno partecipato esattamente a una o a tre edizioni dei giochi olimpici;
- per ogni nazione i cui atleti hanno partecipato ad almeno un'edizione dei giochi olimpici, l'edizione dei giochi olimpici alla quale ha partecipato il maggior numero di atleti (di tale nazione);
- le edizioni dei giochi olimpici alle quali ha partecipato almeno un atleta di una nazione i cui atleti non hanno partecipato ad alcuna altra edizione;
- (FACOLTATIVA) gli atleti che hanno partecipato ai giochi olimpici in modo discontinuo, ossia tali che tra la prima e l'ultima edizione dei giochi alle quali hanno partecipato ve ne sia almeno una in cui erano assenti (ad esempio, gli atleti che hanno partecipato all'edizione 25 e all'edizione 28, ma non alle edizioni 26 e 27).

a. esattamente una = almeno 1 - almeno 2
esattamente tre = almeno 3 - almeno 4

ALMENO_UNO $\leftarrow \Pi_{\text{ATLETA}}(\text{HA_PARTECIPATO})$

HA_PARTECIPATO_2(ATLETA2, EDIZIONE2) $\leftarrow \text{HA_PARTECIPATO}$

ALMENO_DUE $\leftarrow \Pi_{\text{ATLETA}} \left(\sigma_{\text{EDIZIONE} < \text{EDIZIONE}2} (\text{HA_PARTECIPATO} \times \text{HA_PARTECIPATO_2}) \right)$
ATLETA = ATLETA2

ESATTAMENTE_UNO $\leftarrow \text{ALMENO_UNO} - \text{ALMENO_DUE}$

HA_PARTECIPATO_3(ATLETA3, EDIZIONE3) $\leftarrow \text{HA_PARTECIPATO}$

ALMENO_TRE $\leftarrow \Pi_{\text{ATLETA}} \left(\sigma_{\text{ATLETA} = \text{ATLETA2} \text{ AND } \text{ATLETA2} = \text{ATLETA3} \text{ AND } \text{EDIZIONE} < \text{EDIZIONE}2 \text{ AND } \text{EDIZIONE}2 < \text{EDIZIONE}3} (\text{HA_PARTECIPATO} \times \text{HA_PARTECIPATO_2} \times \text{HA_PARTECIPATO_3}) \right)$

HA_PARTECIPATO_4(ATLETA4, EDIZIONE4) $\leftarrow \text{HA_PARTECIPATO}$

ALMENO_QUAT $\leftarrow \Pi_{\text{ATLETA}} \left(\sigma_{\text{ATLETA} = \text{ATLETA1} \text{ AND } \text{ATLETA1} = \text{ATLETA3} \text{ AND } \text{ATLETA3} = \text{ATLETA4} \text{ AND } \text{EDIZIONE} < \text{EDIZIONE}2 \text{ AND } \text{EDIZIONE}2 < \text{EDIZIONE}3 \text{ AND } \text{EDIZIONE}3 < \text{EDIZIONE}4} (\text{HA_PARTECIPATO} \times \text{HA_PARTECIPATO_2} \times \text{HA_PARTECIPATO_3} \times \text{HA_PARTECIPATO_4}) \right)$

ESATTAMENTE_TRE $\leftarrow \text{ALMENO_TRE} - \text{ALMENO_QUAT}$

S $\leftarrow \text{ESATTAMENTE_UNO} \cup \text{ESATTAMENTE_TRE}$

b. $NAZIONE_COUNT \leftarrow \sum_{\substack{NAZIONE, EDIZIONE \\ ATLETA}} \text{COUNT}(\text{ATLETA}) \left(\prod_{\substack{NAZIONE, EDIZIONE, ATLETA \\ CODICEATLETA = ATLETA}} (\text{ATLETA} \in \text{HA_PARTECIPATO}) \right)$

$NAZIONE_COUNT_1 (NAZIONE1, EDIZIONE1, COUNT_1) \leftarrow NAZIONE_COUNT$

$NO_GOOD \leftarrow \prod_{\substack{NAZIONE, EDIZIONE \\ ATLETA}} \left(\begin{array}{l} \neg (\text{EDIZIONE} = \text{EDIZIONE1} \text{ AND } \\ \text{NAZIONE} = \text{NAZIONE1} \text{ AND } \\ \text{COUNT} < \text{COUNT_1} \end{array} \right)$

$S \leftarrow \prod_{\substack{NAZIONE, EDIZIONE \\ ATLETA}} (\text{ATLETA} \in \text{HA_PARTECIPATO}) - NO_GOOD$

c. $NAZ_EDIZ \leftarrow \prod_{\substack{NAZIONE, EDIZIONE \\ ATLETA}} (\text{HA_PARTECIPATO} \in \text{ATLETA})$

N1 E1

N1 E2

N2 E1

$NAZ_EDIZ_2 (NAZIONE2, EDIZIONE2) \leftarrow NAZ_EDIZ$

$NAZ_CON_PIU_EDIZ \leftarrow \prod_{\substack{NAZIONE \\ ATLETA}} \left(\begin{array}{l} \neg (\text{NAZIONE} = \text{NAZIONE2} \text{ AND } \\ (\text{NAZ_EDIZ} \times \text{NAZ_EDIZ_2}) < \text{EDIZIONE} \text{ AND } \\ \text{EDIZIONE} < \text{EDIZIONE2} \end{array} \right)$

$NAZ_VALIDE \leftarrow \prod_{\substack{NAZIONE \\ ATLETA}} (\text{ATLETA}) - NAZ_CON_PIU_EDIZ$

$S \leftarrow \prod_{\substack{EDIZIONE \\ ATLETA}} ((\text{NAZ_VALIDE} \in \text{ATLETA}) \in \text{HA_PARTECIPATO})$

d. $UNIVERSO(\text{ATLETA}, \text{EDIZIONE}) \leftarrow \prod_{\substack{ATLETA \\ ATLETA}} (\text{ATLETA}) \times \prod_{\substack{EDIZIONE \\ OLIPIADE}} (\text{OLIPIADE})$

$NON_HA_PARTECIPATO (\text{ATLETA}, \text{EDIZIONE}) \leftarrow \text{UNIVERSO} - \text{HA_PARTECIPATO}$

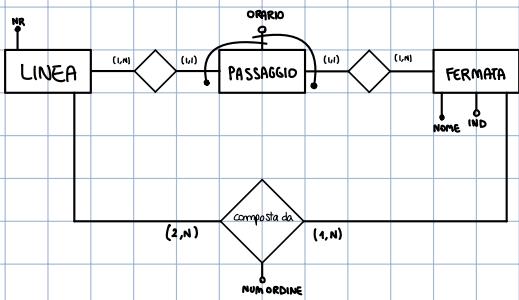
$HA_PARTECIPATO1 (\text{ATLETA}, \text{EDIZIONE1}) \leftarrow \text{HA_PARTECIPATO}$

$S \leftarrow \prod_{\substack{ATLETA \\ ATLETA}} \left(\begin{array}{l} \neg (\text{ATLETA} = \text{ATLETA1} \text{ AND } \\ (\text{HA_PARTECIPATO} \times \text{NON_HA_PARTECIPATO} \times \text{HA_PARTECIPATO1})) \\ \text{ATLETA1} = \text{ATLETA} \text{ AND } \\ \text{EDIZIONE} < \text{EDIZIONE1} \text{ AND } \\ \text{EDIZIONE} < \text{EDIZIONE1} \end{array} \right)$

Esercizio 3:

Si vuole realizzare una base di dati per la gestione giornaliera di un sistema di trasporto urbano mediante autobus caratterizzato dal seguente insieme di requisiti.

- Il servizio sia organizzato in un certo insieme di linee di trasporto urbano. Ogni linea sia identificata univocamente da un numero (ad esempio, linea di trasporto urbano numero 1) e sia caratterizzata da un certo numero di fermate. Si tenga traccia anche dell'ordine della fermate di una data linea.
- Ogni fermata sia identificata univocamente da un nome. Ad ogni fermata sia, inoltre, associato un indirizzo. Una fermata possa essere raggiunta da più linee di trasporto.
- Si tenga traccia anche dei passaggi di una determinata linea in una determinata fermata. Si assuma che una linea possa effettuare più passaggi in una data fermata nel corso della giornata, ovviamente in orari diversi. Si escluda, invece, la possibilità che autobus di linee diverse possano passare nella stessa fermata esattamente nello stesso orario.



VINCOLI: Non posso avere fermate con linee che non hanno quelle fermate in COMPOSTA DA
gli orari di PASSAGGIO devono rispettare l'ordine delle fermate

DATI DERIVATI: