Esercizio 1:

Sia dato il seguente schema relazionale relativo alla partecipazione di un dato insieme di sciatori ad un certo insieme di gare:

$SCIATORE(\underline{CodS}, Nome, Cognome, Nazione, AnnoNascita)$;

$GARA(\underline{CodG}, Luogo, Data, Disciplina)$;

$PARTECIPAZIONE(\underline{Gara, Sciatore}, PosizioneArrivo, Tempo)$.

Si assuma che ogni sciatore sia identificato da un codice, che lo individua univocamente fra tutti gli sciatori, e sia caratterizzato da un nome, un cognome, una nazionalità e un anno di nascita. Si assuma che ogni gara sia identificata univocamente da un codice e sia caratterizzata dal luogo e dalla data in cui si svolta e dalla disciplina (discesa libera, slalom gigante, ..). Si assuma che, in uno stesso luogo, possano essere svolte più gare della stessa disciplina, ma in date diverse, e che nella stessa data e nello stesso luogo possano essere svolte più gare, ma di discipline diverse. Si assuma che a tutti gli atleti che hanno partecipato ad una gara venga assegnata una posizione. Si ammetta la possibilità di ex aequo nella classifica finale di una gara.

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 (senza usare l'operatore CONTAINS e usando solo se e quando necessario le funzioni aggregate):

(a) le gare la cui classifica finale non presenta ex aequo;

(b) gli sciatori che hanno partecipato solo a gare che si sono svolte a Schladming, ma non a tutte (si assuma che a Schladming si siano svolte almeno due gare).

(FACOLTATIVO) Formulare un'interrogazione in algebra relazionale che permetta di determinare quanto richiesto al punto (b), senza usare l'operatore di divisione e usando solo se necessario le funzioni aggregate.

a.

```
SELECT G.CODG
FROM GARA G
WHERE UNIQUE ( SELECT POSIZIONEARRIVO
               FROM PARTECIPAZIONE
               WHERE GARA = G.CODGARA
             );
```

b.

SOLO A GARE SCH → ∄ GARA NON A SCH

NON A TUTTE SCH → ∃ GARA SCH DOVE NON C'È

```
SELECT S.CODS
FROM SCIATORE S
WHERE NOT EXIST ( SELECT *
                  FROM PARTECIPAZIONE, GARA
                  WHERE LUOGO ≠ 'SCHLADMING' AND
                        CODG = GARA AND
                        SCIATORE = S.CODS
                )
EXCEPT
SELECT S.CODS
FROM SCIATORE S
WHERE NOT EXIST ( SELECT *
                  FROM PARTECIPAZIONE, GARA G
                  WHERE G.LUOGO = 'SCHLADMING' AND
                        AND GARA = G.CODG AND
                        NOT EXIST ( SELECT *
                                    FROM PARTECIPAZIONE
                                    WHERE GARA = G.COD_G AND
                                          SCIATORE = S.CODS
```

Esercizio 2:

Si scriva il codice SQL per creare e popolare le seguenti tabelle:

Dipartimento

id_dip	nome_dip	universita	direttore
id111	Computer Science	Stanford University	Donald Knuth
id000	Computer Science	Cambridge University	Alan Turing
id222	Computer Science	California Institute of Technology	John McCarthy

Ricercatore

nome	eta	afferenza
Alan Turing	41	id000
Donald Knuth	82	id111
John McCarthy	84	id222
Robert Tarjan	72	id222

dove:

- nome_dip, universita, direttore, nome e afferenza sono stringhe variabili di 50 caratteri,
- id_dip una stringa variabile di 5 caratteri,
- eta un numero intero.
- afferenza una chiave esterna verso id_dip

Si scriva il codice SQL che corrisponde allo spostamento del ricercatore Robert Tarjan dal California Institute of Technology verso la Standford University.

Si consideri il seguente vincolo: il direttore di ogni dipartimento deve afferire al dipartimento stesso. Quali azioni (inserimento/aggiornamento/cancellazione) e su quali tabelle possono violare tale vincolo? L'aggiornamento di cui sopra può violare questo vincolo?

Si scelga un'azione fra quelle elencate precedentemente e si scriva un trigger SQL per evitare che essa violi il vincolo.

1.
```
CREATE TABLE DIPARTIMENTO(
  ID_DIP varchar(5) PRIMARY KEY,
  NOME_DIP varchar(50),
  UNIVERSITA varchar(50),
  DIRETTORE varchar(50)
);
```

```
CREATE TABLE RICERCATORE(
  NOME varchar(50) PRIMARY KEY,
  ETA integer,
  AFFERENZA varchar(50) REFERENCES DIPARTIMENTO(ID_DIP)
)
```

```
INSERT INTO DIPARTIMENTO (ID_DIP, NOME_DIP, UNIVERSITA, DIRETTORE) VALUES
('id111','computer science', 'Stamford University', 'Donald Knuth'),
(  "          "          "          "          )
```

2.
```
UPDATE RICERCATORE SET AFFERENZA = (SELECT ID_DIP FROM DIPARTIMENTO WHERE UNIVERSITA = 'STANFORD...') WHERE NOME = 'TARJAN...';
```

3. SU tabella DIPARTIMENTO: INSERIMENTO/AGGIORNAMENTO con un DIRETTORE che non esiste in RICERCATORE o che non appartiene a quel dipartimento
   SU tabella RICERCATORE: AGGIORNAMENTO dell'afferenza di un RICERCATORE DIRETTORE in un DIPARTIMENTO non suo
   CANCELLAZIONE di un direttore

Esercizio 4:

Si considerino i seguenti schedule:

s_1: $r_2(x), r_3(y), w_1(x), w_4(y), r_1(x), w_2(x), w_0(x), r_3(v)$;

s_2: $r_0(x), w_0(x), r_1(x), r_2(y), w_1(x), w_0(y)$.

Stabilire se gli schedule dati sono o meno serializzabili rispetto alle viste, ai conflitti, al metodo del locking a due fasi e al metodo del locking a due fasi stretto.

S_1: VSR: legge = $\{(r_1(x), w_1(x))\}$ → dato che $(r_2(x), w_4(x)) \notin$ legge $\Rightarrow t_2 < t_1$ $\Rightarrow$ ASSURDO $\notin$ VSR, CSR, 2PL, TS
$(r_1(x), w_2(x)) \notin$ legge $\Rightarrow t_1 < t_2$

ultima scrit = $\{w_0(x), w_4(y)\}$?

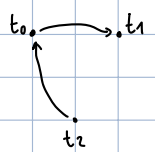
$2 < 0$
$1 < 0$
$1 < 2$
$3 < 4$
$2 < 1$

S_2: VSR: legge = $\{(r_1(x), w_0(x))\}$ ($t_0 < t_1$) dato che $(r_2(y), w_0(y)) \notin$ legge → $t_2 < t_0$ $\Rightarrow$ schedule seriale: $t_2 t_0 t_1$ → $\in$ VSR

ultima scrit = $\{w_0(y), w_1(x)\}$

$2 < 0$
$0 < 1$

CSR: $r_0(x), w_0(x), r_1(x), r_2(y), w_1(x), w_0(y)$



$\in$ CSR $\Rightarrow$ sch seriale: $t_1 t_0 t_2$

2PL: $r_0(x)$ $w_0(x)$ $r_1(x)$ ($\rightarrow$ no t_0) $r_2(y)$ $w_1(x)$ $w_0(y)$ $\Rightarrow \notin$ 2PL