5 Prova del 06/02/2017

Schema base di dati

Si consideri il seguente schema relazionale **parziale** (chiavi primarie sottolineate) contenente le informazioni relative alla programmazione di allenamenti:

`ESERCIZIO(`<u>`nome`</u>`, livello, gruppoMuscolare);`
`PROGRAMMA(`<u>`nome`</u>`, livello);`
`ESERCIZIO_IN_PROGRAMMA(`<u>`nomeProgramma, giorno, nomeEsercizio`</u>`, ordine, serie, ripetizioni, TUT, riposo);`

dove entrambi gli attributi `livello` hanno dominio {principiante, intermedio, avanzato}, `gruppoMuscolare` ha dominio GM={petto, schiena, spalle, braccia, gambe}, `giorno` è il nome del giorno della settimana, ordine è un intero che indica l'ordine di esecuzione in un allenamento, `serie` indica quante volte una serie di ripetizioni deve essere svolta, `ripetizioni` indica quante volte un esercizio deve essere ripetuto, `TUT` è il tempo sotto tensione in s, che viene rappresentato come 4 valori interi distinti e `riposo` è il tempo in s di riposo tra una serie e la successiva.

Si sottolinea che un PROGRAMMA è composto da un insieme di esercizi distribuiti su uno o più giorni.

Domanda 1 [5 punti]

Scrivere in codice PostgreSQL la dichiarazione di tutti i domini necessari per implementare lo schema relazionale.

Scrivere una tabella che rappresenti i possibili valori di TUT: ciascuna tupla deve rappresentare un id e una possibile combinazione di 4 interi non negativi.

Scrivere il codice PostgreSQL che generi le tabelle per rappresentare lo schema relazionale con tutti i possibili controlli di integrità e di correttezza dei dati.

Soluzione

I domini sono 3 e possono essere definiti come:

```sql
CREATE DOMAIN Livello AS VARCHAR
   CHECK( VALUE IN('principiante', 'intermedio', 'avanzato') );
CREATE DOMAIN gruppoMuscolare AS VARCHAR
   CHECK( VALUE IN('petto', 'schiena', 'spalle', 'spalle', 'braccia', 'gambe') );
CREATE DOMAIN giorniSettimana AS CHAR(3)
   CHECK( VALUE IN('LUN', 'MAR', 'MER', 'GIO', 'VEN', 'SAB', 'DOM') );
```

Una possibile soluzione per rappresentare i TUT è data dalla seguente dichiarazione:

```sql
CREATE TABLE TUT (
   id VARCHAR PRIMARY KEY,
   eccentrico INTEGER NOT NULL CHECK(eccentrico >=0),-- chi non conosce bene il TUT poteva mettere nomi
     generici tipo 'valore1', 'valore2', ecc.!
   stopEcc INTEGER NOT NULL CHECK(stopEcc >=0),
   concentrico INTEGER NOT NULL CHECK(concentrico >=0),
   stopCon INTEGER NOT NULL CHECK(stopCon >=0)
);
```

La chiave `id` identifica una possibile combinazione di tempi. Gli istruttori tendono a scrivere un tut nel formato 'x-y-w-z' dove x,y,w e z sono valori in secondi.

Le tabelle sono quindi, in ordine di dichiarazione

```sql
CREATE TABLE ESERCIZIO (
   nome VARCHAR PRIMARY KEY,
   livelloE Livello NOT NULL,
   gruppo gruppoMuscolare NOT NULL
);
CREATE TABLE PROGRAMMA (
   nome VARCHAR PRIMARY KEY,
   livelloA Livello NOT NULL
);
CREATE TABLE ESERCIZIO_IN_PROGRAMMA (
   nomeProgramma VARCHAR REFERENCES PROGRAMMA,
   giorno giorniSettimana NOT NULL,
   nomeEsercizio VARCHAR REFERENCES ESERCIZIO,
   serie INTEGER NOT NULL CHECK (serie > 0),
   ripetizioni INTEGER NOT NULL CHECK (ripetizioni > 0),
   tut VARCHAR REFERENCES TUT NOT NULL,
   riposo INTEGER NOT NULL CHECK (riposo >=0),--in s
   PRIMARY KEY (nomeprogramma, giorno, nomeesercizio)
);
```

Domanda 2 [6 punti]

Dato il nome 'X' di un programma di allenamento, scrivere una query che visualizzi tutti gli esercizi da fare (con tutti i dati necessari per l'esecuzione) ordinati per i giorni di allenamento e ordine di esecuzione. Si deve anche visualizzare il gruppo muscolare dell'esercizio.

Domanda 3 [7 punti]

Assumendo di avere una base di dati PostgreSQL che contenga le tabelle di questo tema d'esame, scrivere un programma Python che, leggendo i dati da console, inserisca una o più tuple nella tabella ESERCIZIO_IN_PROGRAMMA facendo un controllo preventivo che le eventuali dipendenze siano rispettate (si considerino solo quelle rispetto ad altre tabelle, no ai domini). Se una dipendenza non è rispettata, il programma deve richiedere di reinserire il dato associato alla dipendenza prima di procedere a inserire la tupla nella tabella ESERCIZIO_IN_PROGRAMMA. Il programma deve visualizzare l'esito di ogni singolo inserimento. È richiesto che il programma suggerisca il tipo di dati da inserire e che non ammetta possibilità di SQL Injection.

Domanda 4 [8 punti]

Scrivere il codice PostgreSQL, definendo anche eventuali viste, per rispondere alle seguenti due interrogazioni nel modo più efficace:

(a) Trovare per ogni programma di allenamento distribuito su almeno 3 giorni il numero totale di esercizi da svolgere e il tempo totale in minuti per ciascun giorno di allenamento. Il risultato deve visualizzare nome programma di allenamento, giorno e i conteggi richiesti per l'allenamento del giorno considerato.

Suggerimento: Ogni ripetizione di esercizio richiede un tempo pari alla somma dei valori del TUT. Per esempio, se il TUT è (4,1,4,1), il tempo totale di una ripetizione è 10 s. Ogni serie richiede un tempo pari al tempo TUT moltiplicato per il numero di ripetizioni più il tempo di riposo a fine serie. Il tempo di una serie moltiplicato per il numero di serie è il tempo totale (in s) per fare l'esercizio.

(b) Trovare tutti i programmi di allenamento (visualizzando nome e livello) ciascuno dei quali contiene esercizi di livello 'principiante' per il gruppo muscolare 'gambe', ha una durata di almeno 45 minuti per ciascun giorno e non contiene esercizi di qualsiasi livello per il gruppo muscolare 'petto'.

Domanda 5 [7 punti]

Considerando le query della domanda 2, si consideri il seguente risultato del comando ANALYZE su una query che risponde alla domanda 2:

```
Sort  (cost=4.08..4.10 ROWS=9 width=49)
  Sort KEY: e.giorno, e.ordine
  ->  Hash JOIN  (cost=2.45..3.93 ROWS=9 width=49)
        Hash Cond: ((e.nomeesercizio)::TEXT = (es.nome)::TEXT)
        ->  Hash JOIN  (cost=1.36..2.72 ROWS=9 width=43)
              Hash Cond: ((e.tut)::TEXT = (t.id)::TEXT)
              ->  Seq Scan ON esercizio_in_programma e  (cost=0.00..1.24 ROWS=9 width=32)
                    Filter: ((nomeprogramma)::TEXT = 'ABC'::TEXT)
              ->  Hash  (cost=1.16..1.16 ROWS=16 width=19)
                    ->  Seq Scan ON tut t  (cost=0.00..1.16 ROWS=16 width=19)
        ->  Hash  (cost=1.04..1.04 ROWS=4 width=14)
              ->  Seq Scan ON esercizio es  (cost=0.00..1.04 ROWS=4 width=14)
```

(a) Indicare quanti e quali indici sono stati usati per risolvere la query giustificando la risposta.

(b) Supponendo che i dati non varino nel tempo, in base al piano di esecuzione conviene creare degli indici? Se sì, quali?