4 Prova del 20/09/2016

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) contenente le informazioni relative alle autostrade italiane:

AUTOSTRADA(<u>codice</u>, nome, gestore, lunghezza);
RAGGIUNGE(<u>autostrada, comune</u>, numeroCaselli);
COMUNE(<u>codiceISTAT</u>, nome, numeroAbitanti, superficie);

Si ricorda che: (1) il codice delle autostrade italiane considerate è composto dal carattere 'A' seguito da un numero. (2) il codice ISTAT di un comune è una stringa di 6 cifre. (3) l'attributo numeroCaselli indica il numero di caselli dell'autostrada presenti nel territorio del comune (potrebbe essere anche zero).

Domanda 1 [5 punti]

Nella descrizione dello schema non è specificato quali sono i vincoli di integrità. Indicare quali quali sono i vincoli di integrità che si possono desumere usando la notazione '→'.

Scrivere il codice PostgreSQL che generi TUTTE le tabelle per rappresentare lo schema relazionale. Si inseriscano tutti i possibili controlli di integrità e di correttezza dei dati. Si giustifichi, dove necessario, la scelta del dominio.

Soluzione

I vincoli di integrità secondo la notazione '→' sono:

RAGGIUNGE.autostrada → AUTOSTRADA.codice
RAGGIUNGE.comune → COMUNE.codiceIstat

Una possibile soluzione è data dalle seguenti dichiarazioni:

```sql
CREATE TABLE AUTOSTRADA (
   codice VARCHAR(5) PRIMARY KEY CHECK (codice LIKE 'A%'), --meglio (SIMILAR TO
     'A[0-9]+'),
   nome VARCHAR(50) UNIQUE NOT NULL,
   gestore VARCHAR NOT NULL,
   lunghezza NUMERIC(6,3) NOT NULL CHECK (lunghezza > 0) --in km con precisione
     al metro
);

CREATE TABLE COMUNE (
   codiceIstat CHAR(6) PRIMARY KEY CHECK(SUBSTRING(codiceIstat,1,1) >='0' AND
     SUBSTRING(codiceIstat,1,1) <= '9' AND ... -- si ripete per le altre
     posizioni
   -- un controllo migliore è CHECK(SIMILAR TO '[0-9]{6}')
   nome VARCHAR(50) UNIQUE NOT NULL,
   numeroAbitanti INTEGER NOT NULL CHECK (numeroAbitanti>=0),
   superficie NUMERIC NOT NULL CHECK (superficie > 0) --in km quadrati
);

CREATE TABLE RAGGIUNGE (
   autostrada VARCHAR(5) REFERENCES AUTOSTRADA,--non specificando ON
     UPDATE/DELETE, si pone la restrizione maggiore.
   comune CHAR(6) REFERENCES COMUNE,
   numeroCaselli INTEGER NOT NULL CHECK (numeroCaselli >=0),
   PRIMARY KEY(autostrada,comune)
);
```

Domanda 2 [6 punti]

Trovare i comuni che non sono raggiunti da autostrade gestite dal gestore X, riportando il codice, il nome e gli abitanti del comune.

Soluzione

Un comune può non essere presente nella tabella RAGGIUNGE o può essere con numero di caselli = 0 (che è equivale a dire che non è raggiunto dall'autostrada associata).

Una soluzione possibile richiede quindi di selezionare tutti i comuni e togliere quelli che sono raggiunti dall'autostrada data con un numero di caselli significativo.:

```sql
SELECT c.codiceIstat, c.nome, c.numeroAbitanti
FROM COMUNE c
WHERE c.codiceIstat NOT IN (
   SELECT r.comune
   FROM RAGGIUNGE r JOIN AUTOSTRADA a ON r.autostrada=a.codice
   WHERE a.gestore = X AND r.numeroCaselli > 0
);
```

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 RAGGIUNGE facendo un controllo preventivo che le eventuali dipendenze siano rispettate. 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 RAGGIUNGE. 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.

Soluzione

La soluzione dovrà prevede un ciclo per chiedere se si vuole continuare ad inserire tuple. Inoltre, per ogni comune e autostrada dati da input, si deve verificare che siano presenti nelle loro rispettive tabelle. In questa soluzione la verifica che la tupla (autostrada,comune,...) nella tabella RAGGIUNGE non sia già inserita non viene fatta. Nel caso in cui la tupla (autostrada,comune,...) fosse già presente, l'inserimento non verrebbe effettuato perché PostgreSQL solleverebbe un'eccezione e il programma riporterebbe l'errore.

```python
"""
Gestione semplice tabella RAGGIUNGE su PostgreSQL
@author: posenato
"""

import os
import psycopg2

def clear():
    """
    Per cancellare la console.
    """
    os.system("clear")

rowcount = 0
with psycopg2.connect(host="localhost", database="posenato", user="posenato") as conn:
    with conn.cursor() as cur:
        inserisci = input("Vuoi inserire una tupla nella tabella RAGGIUNGE? [qualsiasi carattere=Si, \
     No=No]: ")
        while inserisci != 'No':
            clear()
            # RAGGIUNGE ha i seguenti attributi: autostrada,comune e numeroCaselli
            print("Inserire i seguenti dati. Tutti i dati sono necessari.")

            noInserita = True
            autostrada = ''
            while noInserita:
                autostrada = input("Inserire autostrada [codice alfanumerico 'A*']: ")
                cur.execute("SELECT count(*) FROM autostrada WHERE codice = %s", (autostrada,))
                status = cur.fetchone()
                if status != None and status[0] == 1:
                    noInserita = False
                else:
                    print("L'autostrada inserita non esiste. Ripetere inserimento.")

            noInserita = True
            comune = ''
            while noInserita:
```

```
            comune = input("Inserire comune [codice numerico]: ")
            cur.execute("SELECT count(*) FROM comune WHERE codiceIstat = %s", (comune,))
            status = cur.fetchone()
            if status != None and status[0] == 1:
                noInserita = False
            else:
                print("Il comune inserito non esiste. Ripetere inserimento.")

        numeroCaselli = input("Inserire numero caselli [intero]: ")
        numeroCaselli = int(numeroCaselli)

        cur.execute("INSERT INTO Raggiunge (autostrada,comune,numeroCaselli) VALUES (%s,%s,%s)", \
            (autostrada, comune, numeroCaselli))
        print("Esito inserimento tabella: ", cur.statusmessage)
        rowcount += cur.rowcount
        inserisci = input("Vuoi inserire una tupla nella tabella RAGGIUNGE? [qualsiasi carattere=Si, \
     No=No]: ")

print("In questa sessione di lavoro sono state inserite {0:d} righe".format(rowcount))
```

Avendo usato il meccanismo proprio del metodo execute per passare i valori alla query parametrizzata, si ha la ragionevole certezza che eventuali tentativi di SQL Injection sono bloccati perché il metodo execute esegue delle conversioni dei tipi di dati. As esempio, se un parametro è di tipo string, allora execute esegue tutti gli escape necessari in modo che il valore sia inserito nel testo della query come valore stringa.

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 autostrada che raggiunga almeno 10 comuni, il numero totale di comuni che raggiunge e il numero totale di caselli, riportando il codice dell'autostrada, la sua lunghezza e i conteggi richiesti.

(b) Selezionare le autostrade che hanno un potenziale di utenti diretti (=numero di abitanti che la possono usare dal loro comune) medio rispetto al numero dei caselli dell'autostrada stessa superiore alla media degli utenti per casello di tutte le autostrade. Si deve riportare il codice dell'autostrada, il suo numero totale di utenti, la media di utenti per casello.

Soluzione

Alla prima query si risponde con:

```sql
SELECT a.codice, a.lunghezza, COUNT(*) AS numeroComuni, SUM(r.numeroCaselli) AS
      numeroCaselli
FROM COMUNE c JOIN RAGGIUNGE r ON c.codiceIstat=r.comune JOIN AUTOSTRADA a ON
      r.autostrada=a.codice
GROUP BY a.codice, a.lunghezza
HAVING COUNT(*)>=10;
```

Per la seconda query, si costruisce una vista che, per ogni autostrada, conta il numero totale di abitanti delle città raggiunte e il numero totale di caselli autostradali.

```sql
CREATE VIEW utentiPerAutostrada AS
SELECT a.codice, SUM(c.numeroAbitanti) AS numeroUtenti, SUM(r.numeroCaselli) AS
      numeroCaselli
FROM COMUNE c JOIN RAGGIUNGE r ON c.codiceIstat=r.comune JOIN AUTOSTRADA a ON
      r.autostrada=a.codice
GROUP BY a.codice
```

Alla domanda si risponde quindi come:

```sql
SELECT v.codice, v.numeroUtenti, v.numeroUtenti/v.numeroCaselli AS
      mediaPerCasello
FROM utentiPerAutostrada v
WHERE v.numeroUtenti/v.numeroCaselli >= ALL (
   SELECT AVG(numeroUtenti/numeroCaselli)
   FROM utentiPerAutostrada
);
```

Domanda 5 [7 punti]

(a) Considerando le query della domanda 4, illustrare quali sono gli indici da definire che possono migliorare le prestazioni e, quindi, scrivere il codice PostgreSQL che definisce gli indici illustrati. Attenzione a non creare indici già presenti (per ogni indice proposto già presente la valutazione è penalizzata)!

(b) Si consideri poi il seguente risultato del comando ANALYZE su una query inerenti alle tre tabelle considerate:

```
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Hash JOIN  (cost=12.72..28.00 ROWS=5 width=118)
   Hash Cond: (c.codiceistat = r.comune)
   ->  Seq Scan ON comune c  (cost=0.00..13.80 ROWS=380 width=146)
   ->  Hash  (cost=12.66..12.66 ROWS=5 width=28)
         ->  Bitmap Heap Scan ON raggiunge r  (cost=4.19..12.66 ROWS=5 width=28)
               Recheck Cond: ((autostrada)::TEXT = 'A1'::TEXT)
               ->  Bitmap INDEX Scan ON raggiunge_pkey  (cost=0.00..4.19 ROWS=5)
                     INDEX Cond: ((autostrada)::TEXT = 'A1'::TEXT)
```

Desumere il testo della query.

Suggerimento: la query inizia con `SELECT * FROM ...`

Soluzione

(a) I join sono fatti tutti usando le chiavi primarie, per le quali il sistema già definisce gli indici in modo automatico.

Quindi non si devono dichiarare ulteriori indici.

(b) È un hash join, quindi c'è un JOIN con condizione c.codiceistat = r.comune. Poi c'è un Healp Scan con condizione autostrada = 'A1'. Quindi è un JOIN tra COMUNE e RAGGIUNGE dove c'è una selezione sul nome dell'autostrada nella tabella RAGGIUNGE.

```
SELECT *
FROM comune c JOIN raggiunge r ON c.codiceistat=r.comune
WHERE r.autostrada='A1';
```