6 Prova del 06/02/2017

Schema base di dati

Si consideri il seguente schema relazionale **parziale** (chiavi primarie sottolineate) contenente le informazioni relative alle telefonate eseguite dai clienti di una società telefonica:

`CLIENTE(codice, nome, cognome, nTelefono, indirizzo, città)`
`TELEFONATA(contratto, nTelChiamato, instanteInizio, durata)`
`CONTRATTO(contratto, cliente, tipo, dataInizio, dataFine)`
`CITTA(codice, nome)`

dove `CONTRATTO.dataFine` è NULL per i contratti che sono attivi e parte dei vincoli di integrità sono: TELEFONATA.contratto → CONTRATTO, CONTRATTO.cliente → CLIENTE. L'attributo CONTRATTO.tipo indica il tipo di contratto e può assumere valori come 'privato', 'business', 'corporate', etc. L'insieme di questi valori può variare nel tempo. Si vuole comunque avere il controllo di questo insieme.

Domanda 1 [5 punti]

Indicare quali sono i vincoli di integrità mancanti.

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.

Suggerimento: Si ponga attenzione alla scelta dei domini perché ogni dominio errato dà una penalizzazione.

Domanda 2 [6 punti]

Trovare il cognome, il nome e l'indirizzo dei clienti di Padova che hanno fatto telefonate ieri (ieri = CURRENT_DATE-1) solo in un orario **non** compreso dalla fascia 10:00–17:00.

Suggerimento: l'espressione `CURRENT_DATE - 1 + TIME '10:00'` restituisce il TIMESTAMP che rappresenta ieri alle ore 10:00.

Soluzione

Il fatto che si chieda di selezionare i clienti che hanno fatto telefonate SOLO in orario diverso da quello indicato significa che se un cliente ha fatto telefonate sia in un orario corretto sia in un orario non ammesso NON devono essere selezionati.

Quindi una soluzione del tipo

```
--ERRATA
SELECT DISTINCT C.cognome, C.nome, C.indirizzo
FROM Cliente C JOIN CITTA CI ON C.città=CI.codice JOIN Contratto CO ON C.codice=CO.cliente JOIN
     TELEFONATA T ON T.contratto=CO.contratto
WHERE CI.nome='Padova' AND (
   (t.instanteInizio >= CURRENT_DATE-1 AND t.instanteInizio < CURRENT_DATE - 1 + TIME '10:00')  OR
   (t.instanteInizio > CURRENT_DATE - 1 + TIME '17:00' AND t.instanteInizio < CURRENT_DATE)
);
--ERRATA
```

è errata!

La soluzione corretta è

```
SELECT C.cognome, C.nome, C.indirizzo
FROM Cliente C JOIN CITTA CI ON C.città=CI.codice JOIN Contratto CO ON C.codice=CO.cliente JOIN
     TELEFONATA T ON T.contratto=CO.contratto
WHERE CI.nome='Padova' AND t.instanteInizio >= CURRENT_DATE-1  AND t.instanteInizio < CURRENT_DATE

EXCEPT
SELECT C.cognome, C.nome, C.indirizzo
FROM Cliente C JOIN CITTA CI ON C.città=CI.codice JOIN Contratto CO ON C.codice=CO.cliente JOIN
     TELEFONATA T ON T.contratto=CO.contratto
WHERE CI.nome='Padova' AND t.instanteInizio >= CURRENT_DATE - 1 + TIME '10:00' AND t.instanteInizio <=
     CURRENT_DATE - 1 + TIME '17:00'
```

Domanda 3 [7 punti]

Assumendo di avere una base di dati PostgreSQL che contenga le tabelle di questo tema d'esame, scrivere in modo completo:

(a) Una template JINJA2 per una form HTML 5 che permetta di scegliere una città dalla lista `citta` di dict `{codice, nome}` (presente come parametro) e una data (formato gg/mm/aaaa) e invi i dati all'URL '/telefonateCittaData'.

Scrivere solo la parte della FORM, non tutto il documento HTML.

(b) Un metodo Python che, associato all'URL '/telefonateCittaData' secondo il framework Flask: (1) legga i parametri, (2) usi il metodo `model.Clienti(citta, data)` per ottenere il risultato della query (`[{cognome, nome},...]`), (3) corregga tutti i cognomi rendendoli maiuscoli e (4) usi il metodo `render_template('view.html',...)` per pubblicare il risultato. Se la lista restituita da `model.Clienti` è vuota, il metodo deve passare il controllo a `render_template('erroreParametri.html')`.

Scrivere solo il metodo.

Soluzione

(a)
```
<form action="/telefonateCittaData" method="get">
   <label>Città:</label>
   <select name="citta">
{% for c in citta %}
      <option value="{{ c.codice }}">{{ cs.nome }}</option>
{% endfor %}
   </select><br>
   <label>Data [gg/mm/aaaa]:</label>
   <input type="text" name="data" pattern="[0-9]{1,2}/[0-9]{1,2}/[0-9]{4}">
   <input type="submit" value="Invia">
</form>
```

(b)
```
@app.route('/telefonateCittaData')
def telefonateCittaData():
    '''ritorna la lista degli utenti che hanno fatto almeno una telefonata nella data ma non tra \
       le 10 e le 17.'''
    citta = request.args["citta"]
    data = request.args["data"]
    data = datetime.datetime.strptime(data, '%d/%m/%Y')

    lista = model.Clienti(citta, data)#Si assume che la lista è vuota, lista è None

    if not lista
        return render_template('erroreParametri.html')

    for cliente in lista:
        cliente['cognome'] = cliente['cognome'].upper()

    return render_template('view.html',lista = lista, citta = citta, data = data)
```

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 contratto iniziato nel mese di Gennaio 2007 il numero di telefonate e la durata totale delle telefonate eseguite nel mese di Maggio 2007 dal cliente del contratto, riportando nel risultato la data di inizio del contratto, identificatore del contratto e i conteggi richiesti.

(b) Trovare per ogni contratto il mese in cui ha effettuato il maggior numero di telefonate nell'anno 2016 riportando il numero di contratto, il mese e il numero di telefonate.

Si ricorda che EXTRACT(MONTH FROM <expr>) restituisce il mese (1-12) e che EXTRACT(YEAR FROM <expr>) restituisce l'anno (4 cifre), dove <expr> è un TIMESTAMP.

Domanda 5 [7 punti]

La società prevede di scontare le durate delle telefonate a determinati clienti importanti facendo un update delle loro durate (diminuzione) in modo automatico ogni tot giorni.

Inoltre, come gioco a premi, la società azzera le durate di alcune telefonate che soddisfano certi requisiti, basati anche sulla durata stessa, usando un'altra procedura automatica ogni tot giorni.

Simulare l'esecuzione concorrente di queste due procedure spiegando qual è il livello di isolamento MINIMO richiesto perché le due procedure possano operare in modo concorrente senza lasciare la base di dati in uno stato non corretto (le transazioni possono quindi essere abortite se necessario).