Raccolta dei testi delle prove scritte dal 2016 al 2017 e alcuni temi selezionati dal 2019

Aggiornato al 21 giugno 2019

La raccolta è limitata a soli due anni in quanto gli esercizi hanno sempre il medesimo pattern.

Regole generali

Durata: 2h
Rispondere alle domande in ordine usando uno o più fogli formato A4 distribuiti dal docente. Ciascun foglio deve riportare in alto a sinistra il proprio nome, cognome e numero di matricola. Consegnare solo i fogli che contengono le risposte, no fogli usati per prove o brutte copie.
Se non si raggiunge il punteggio di 13 con le prime tre domande, la prova non è superata e le rimanenti domande non vengono valutate.

1 Prova del 16/05/2016

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) contenente i dati delle auto noleggiate da una agenzia:
`AUTO(targa, marca, modello, posti, cilindrata)`
`NOLEGGIO(targa, cliente, inizio, fine)`
`CLIENTE(nPatente, cognome, nome, paeseProvenienza, nInfrazioni)`
dove `inizio` e `fine` devono permettere di determinare il noleggio in termini di ore e `fine` può essere nullo per i noleggi in corso. I vincoli di integrità sono: `NOLEGGIO.targa`→`AUTO`, `NOLEGGIO.Cliente`→`CLIENTE`.

Domanda 1 [5 punti]
Scrivere il codice PostgreSQL che generi le tabelle rappresentanti le relazioni giustificando, dove necessario, la scelta del dominio e inserendo tutti i possibili controlli di integrità.

Domanda 2 [6 punti]
Scrivere il codice PostgreSQL per trovare i clienti che non hanno mai noleggiato auto della marca 'X', riportando il cognome, il nome e la provenienza del cliente.

Domanda 3 [8 punti]
Scrivere il codice PostgreSQL, definendo anche eventuali viste, per rispondere alle seguenti due interrogazioni nel modo più efficace:

(a) Trovare per ogni marca d'auto che ha avuto almeno un noleggio: il numero complessivo di auto di quella marca, il numero di noleggi in cui è stata utilizzata un'auto di quella marca e il numero complessivo di ore di noleggio per le auto di quella marca, riportando la marca e i tre conteggi richiesti.

(b) Trovare la marca con il massimo numero di ore di noleggio visualizzando la marca e il numero di ore di noleggio.

Suggerimento: la funzione `EXTRACT(epoch FROM <INTERVAL>)` restituisce l'intervallo in secondi.

Domanda 4 [7 punti]
Scrivere il codice PostgreSQL che crei uno o più indici che possono migliorare le prestazioni dell'interrogazione della seconda domanda giustificando la scelta. Attenzione a non creare indici già presenti (per ogni indice proposto già presente la valutazione è penalizzata)!

Domanda 5 [7 punti]
Assumendo di avere una base di dati PostgreSQL che contenga le tre tabelle, scrivere un programma Java che si interfacci alla base di dati e, dopo aver chiesto da input la marca 'X', visualizzi su monitor il risultato dell'interrogazione della domanda 2.

2 Prova del 20/06/2016

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) contenente le ricette di un ristorante:
`INGREDIENTE(Id, Nome, Calorie, Grassi, Proteine, Carboidrati);`
`COMPOSIZIONE(Ricetta, Ingrediente, Quantità)`
`RICETTA(Id, Nome, Regione, Porzioni, TempoPreparazione)`
La quantità di grassi, proteine e carboidrati è in grammi su 100 grammi di ingrediente; la quantità nella tabella `COMPOSIZIONE` si assume espressa in grammi. La unità di misura dell'attributo `INGREDIENTE.Calorie` si assume essere kcal. Vincoli di integrità: `COMPOSIZIONE.Ricetta`→`RICETTA`, `COMPOSIZIONE.Ingrediente`→`INGREDIENTE`.

Domanda 1 [5 punti]
Scrivere il codice PostgreSQL che generi le tabelle rappresentanti le relazioni. Si inseriscano tutti i possibili controlli di integrità e di correttezza dei dati. Si giustifichi, dove necessario, la scelta del dominio.

Suggerimento: un confronto tra un intervallo e una costante richiede l'uso del cast sulla costante. Esempio: `'00:00:00'::INTERVAL`

Domanda 2 [6 punti]
Trovare il nome e il tempo di preparazione delle ricette della regione Veneto che contengono almeno un ingrediente con più del 40% di carboidrati.

Domanda 3 [8 punti]
Scrivere il codice PostgreSQL, definendo anche eventuali viste, per rispondere alle seguenti due interrogazioni nel modo più efficace:

(a) Trovare per ogni ricetta la quantità totale di proteine e la quantità totale di grassi, riportando anche il nome della ricetta.

(b) Trovare le ricette che hanno la massima quantità di grassi per porzione, riportando il nome della ricetta e la sua quantità di grasso totale.

Domanda 4 [7 punti]
Dato il seguente query plan:

```
                                QUERY PLAN
-------------------------------------------------------------------------
 Nested Loop  (cost=... ROWS= width=22)
   JOIN Filter: (c.ingrediente = i.id)
   ->  Seq Scan ON ingrediente i  (cost=0.00..1.04 ROWS=1 width=4)
         Filter: (carboidrati > '40'::NUMERIC)
   ->  Hash JOIN  (cost=... ROWS= width=26)
         Hash Cond: (c.ricetta = r.id)
         ->  Seq Scan ON composizione c  (cost=0.00..200 ROWS=180 width=8)
         ->  Hash  (cost=... ROWS= width=26)
               ->  Seq Scan ON ricetta r  (cost= ROWS=... width=26)
                     Filter: ((regione)::TEXT = 'Veneto'::TEXT)
```

Scrivere il codice PostgreSQL che crei uno o più indici che possono migliorare le prestazioni dell'interrogazione. Attenzione a non creare indici già presenti (per ogni indice proposto già presente la valutazione è penalizzata)!

Domanda 5 [7 punti]
Assumendo di avere una base di dati PostgreSQL che contenga le tre tabelle di questo tema d'esame, scrivere un programma Python che si interfacci alla base di dati e, dopo aver chiesto da input il nome di una regione, visualizzi su monitor il risultato dell'interrogazione della domanda 2 considerando la regione data in input.

3 Prova del 12/07/2016

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) contenente i risultati dei ricoveri nelle divisioni di una Azienda Sanitaria Locale che gestisce più ospedali:

```
DIVISIONE(id, idOspedale, nome, numeroAddetti);
RICOVERO(divisione, paziente, descrizione, urgenza, dataAmmissione, dataDimissione);
PAZIENTE(codiceFiscale, nome, cognome, regione, nazione);
```

Si ricorda che: (1) il codice fiscale italiano è un codice alfanumerico a lunghezza fissa di 16 caratteri, (2) l'attributo regione in PAZIENTE rappresenta la regione italiana che paga le spese sanitarie, (3) le regioni italiane hanno un nome proprio fissato per legge e (4) la nazione in PAZIENTE rappresenta la nazione di nascita del paziente ed è codificata con il codice ISO 3166-1 alpha-3 (lungo esattamente 3 caratteri).

Domanda 1 [5 punti]

Lo schema manca della definizione di una o più entità/relazioni. Inoltre nel testo non è specificato quali sono i vincoli di integrità. Indicare quali sono le entità/relazioni mancanti (può aiutare leggere anche la seconda domanda prima di rispondere) e quali sono i vincoli di integrità che si possono desumere dal testo.

Scrivere il codice PostgreSQL che generi il dominio contente tutti i nomi delle regioni e 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.

Domanda 2 [6 punti]

Scrivere il codice PostgreSQL per trovare tutti gli identificatori e i nomi degli ospedali dove sono stati ricoverati almeno una volta pazienti nati in 'Svizzera'.

Domanda 3 [8 punti]

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

(a) Trovare il nome delle regioni aventi pazienti ricoverati che non hanno mai avuto ricoveri nella Divisione di 'Cardiochirurgia' dell'ospedale 'Borgo Trento'.

(b) Si considerino solo due divisioni, quella 'Cardiologia' e quella di 'Cardiochirurgia' dell'ospedale 'Borgo Trento'. Trovare le regioni con il numero di ricoveri non urgenti presso le due divisioni numericamente maggiore dei ricoveri urgenti sempre nelle due divisioni e relativi sempre alla stessa regione. Nel risultato si riporti il nome della regione, il numero di ricoveri non urgenti e la loro durata media.

Domanda 4 [7 punti]

Considerando le query della domanda 3, 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)!

Domanda 5 [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 RICOVERI. Il programma deve visualizzare l'esito di ogni singolo inserimento. Non è richiesto che il programma visualizzi i dati possibili per gli attributi con domini vincolati. È richiesto che il programma suggerisca il tipo di dati da inserire e che non ammetta possibilità di SQL Injection.

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(codice, nome, gestore, lunghezza);
RAGGIUNGE(autostrada, comune, numeroCaselli);
COMUNE(codiceISTAT, 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:

```
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.:

```
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.

```
"""
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:

```
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.

```
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:

```
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';
```

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(nome, livello, gruppoMuscolare);
PROGRAMMA(nome, livello);
ESERCIZIO_IN_PROGRAMMA(nomeProgramma, giorno, nomeEsercizio, 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:

```
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:

```
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

```
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,
   ordine INTEGER,
   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?

6 Prova del 13/06/2017

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).

7 Prova del 04/07/2017

Schema base di dati

Si consideri il seguente schema relazionale **parziale** (chiavi primarie sottolineate) contenente le informazioni relative alla gestione prestiti in una rete di biblioteche:

`UTENTE(codiceFiscale, nome, cognome, telefono, dataIscrizione, stato)`
`PRESTITO(idRisorsa, idBiblioteca, idUtente, dataInizio, durata)`
`RISORSA(id, biblioteca, titolo, tipo, stato)`

dove `PRESTITO.idBiblioteca` e `RISORSA.biblioteca` fanno riferimento alla chiave primaria (id) dell'entità `BIBLIOTECA` che si assume già definita come tabella. L'attributo `UTENTE.stato` può assumere il valore 'abilitato' o 'ammonito' o 'sospeso'; `RISORSA.tipo` indica il tipo di risorsa. Esempio: 'articolo', 'libro', etc. L'insieme di questi valori può variare nel tempo ma si vuole mantenere un controllo stretto. `RISORSA.stato` può assumere il valore 'solo consultazione' o 'disponibile' o 'on-line'.

Domanda 1 [5 punti]

(a) Scrivere il codice PostgreSQL per definire i domini/tabelle ausiliare necessarie.

(b) Indicare i 4 vincoli di integrità referenziale con la notazione `Tabella.attributo/i → Tabella.attributo/i`.

(c) Scrivere il codice PostgreSQL che generi le tabelle per rappresentare lo schema relazionale scegliendo i domini più appropriati, inserendo tutti i possibili controlli di integrità e di correttezza dei valori/formato dei dati. In particolare, si deve garantire che il formato del codice fiscale si 6 caratteri + 2 cifre + carattere + 2 cifre + carattere + 3 cifre + carattere, il formato del numero di telefono sia il carattere '+' seguito da 10 cifre, e che una durata del prestito sia positiva.

Domanda 2 [6 punti]

Dati i due seguenti piani di esecuzione, a sinistra il piano ottenuto senza l'uso di indici, a destra quello ottenuto usando degli indici, scrivere il codice PostgreSQL che definisce gli indici usati per ottenere il piano di esecuzione a destra. Indicare inoltre qual è il guadagno in termini di pagine disco che si ottiene.

```
Hash JOIN  (cost=287.80..6428.47 ROWS=30)
  Hash Cond: (ie.id_insegn = i.id)
  ->  Seq Scan ON inserogato ie  (cost=0.00..6140.26
    ROWS=30)
       Filter: (((annoaccademico)::TEXT =
    '2006/2007'::TEXT) AND (id_corsostudi = 4))
  ->  Hash  (cost=185.69..185.69 ROWS=8169)
        ->  Seq Scan ON insegn i  (cost=0.00..185.69
    ROWS=8169)
```

```
Nested Loop  (cost=5.07..359.43 ROWS=36)
  -> Bitmap Heap Scan ON inserogato ie
    (cost=4.79..140.27 ROWS=36)
       Recheck Cond: (((annoaccademico)::TEXT =
    '2006/2007'::TEXT) AND (id_corsostudi = 4))
       -> Bitmap INDEX Scan ON i2  (cost=0.00..4.78
    ROWS=36)
            INDEX Cond: (((annoaccademico)::TEXT =
    '2006/2007'::TEXT) AND (id_corsostudi = 4))
  -> INDEX Scan USING i1 ON insegn i  (cost=0.28..6.08
    ROWS=1)
      INDEX Cond: (id = ie.id_insegn)
```

Domanda 3 [7 punti]

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

(a) Un template JINJA2 per una form HTML 5 che: (1) permetta di acquisire un codice fiscale (controllando il formato), (2) di selezionare una biblioteca dalla lista `biblioteche` passata come parametro al template e (3) invi i dati all'URL `/prestitiUtente` in modalità GET. Il formato di `biblioteche` è `[ {id, nome}, ...]`. Scrivere solo la parte della FORM, non tutto il documento HTML.

(b) Un metodo Python che, associato all'URL `/prestitiUtente` secondo il framework Flask, (1) legga i parametri codice fiscale e identificatore biblioteca, (2) si connetta alla base di dati 'X' (si assuma di dover specificare solo il nome della basi di dati) e recuperi tutti i prestiti (idRisorsa, dataInizio, durata) associati al codice fiscale e biblioteca dati come parametri (scrivere la query!), (3) usi il metodo `render_template('view.html',...)` per pubblicare il risultato passando la lista del risultato. Se il risultato dell'interrogazione è vuoto, il metodo deve passare il controllo a `render_template('nessunPrestitoOErrore.html')`. Scrivere solo il metodo.

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 utente che abbia fatto prestiti presso almeno due biblioteche, il numero di prestiti terminati alla data corrente presso ciascuna biblioteca e la loro durata totale sempre per ciascuna biblioteca. Il risultato deve riportare il codice fiscale dell'utente, l'id della biblioteca e i conteggi richiesti.

(b) Trovare per ogni biblioteca (specificata solo dal suo id), l'utente/i con il maggior numero di prestiti e l'utente/i con la durata complessiva maggiore, riportando nel risultato l'id della biblioteca, il codice fiscale dell'utente e i conteggi richiesti (se gli utenti per ciascuna biblioteca coincidono, si deve stampare solo una riga).

Domanda 5 [7 punti]

Completare il seguente pezzo di codice Java affinché la procedura stampi in console il risultato dell'interrogazione dell'esercizio 3.

```
public static void main(String[] args) throws Exception {
   // Caricamento driver
   Class.forName("org.postgresql.Driver");
   String codiceFiscale = args[0];
   String idBiblio = args[1];

   // Creazione connessione
   try (Connection con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/X","","")) {

      ...parte del codice da completare...

         while (rs.next()) {
            System.out.println(String.format("| %20s | %20s | %20s |", rs.getInt("idRisorsa"),
               sdf.format(rs.getDate("dataInizio")), ((PGInterval)rs.getObject("durata")).getValue());
         }
      } catch (SQLException e) {
         System.out.println("Problema durante estrazione dati: " + e.getMessage());
         return;
      }
   }
}
```

8 Prova del 19/09/2017

Schema base di dati

Si consideri il seguente schema relazionale **parziale** (chiavi primarie sottolineate) contenente le informazioni relative a una gestione semplificata di prenotazioni voli:

`CLIENTE(codiceFiscale, nome, cognome, telefono, sesso)`
`VOLO(iataCompagnia, numero, orarioPartenza, icaoCompagnia, partenza, destinazione, durata, postiBusiness, postiEconomy, postiBusinessComprati, postiEconomyComprati)`
`PRENOTAZIONE(iataCompagnia, numeroVolo, orarioPartenza, codiceFiscale, isBusiness)`
`AEROPORTO(iata, nome, città, nazione)`
`COMPAGNIA(icao, iata, nome, nazione)`

dove `iataCompagnia` è il codice di 2 lettere della compagnia aerea, `VOLO.numero` è un intero positivo, `VOLO.orarioPartenza` è la data e l'orario con fuso orario della partenza, `VOLO.partenza/destinazione` sono i codici IATA degli aeroporti (da 3 a 4 lettere). `PRENOTAZIONE.isBusiness` indica se il posto acquistato è della classe business (true) o economy (false). `icao` è il codice identificativo della compagnia aerea (4 lettere). Per ragioni storiche, `iataCompagnia` non identifica una compagnia aerea anche se, ancora oggi, il codice è usato per costruire l'identificatore del volo aereo.

Per questioni di semplicità, si assume che i codici IATA e ICAO siano già definiti come domini di nome, rispettivamente, `IATACompagnia, IATAAeroporto, ICAOCompagnia`.

Domanda 1 [5 punti]

(a) Indicare i 5 vincoli di integrità referenziale con la notazione `Tabella.attributo/i → Tabella.attributo/i`.
Suggerimento: nella tabella `VOLO`, si deve garantire che la compagnia esista e che il codice IATA del volo sia lo stesso della compagnia indicata nel volo.

(b) Scrivere il codice PostgreSQL che generi le tabelle per rappresentare lo schema relazionale scegliendo i domini più appropriati, inserendo tutti i possibili controlli di integrità e di correttezza dei valori/formato dei dati. Per esempio: si deve garantire che il formato del codice fiscale si 6 caratteri + 2 cifre + carattere + 2 cifre + carattere + 3 cifre + carattere, il formato del numero di telefono sia il carattere '+' seguito da 10 cifre, e che una durata del volo sia positiva.

Domanda 2 [6 punti]

Dati i due seguenti piani di esecuzione, a sinistra il piano ottenuto senza l'uso di indici, a destra quello ottenuto usando degli indici, scrivere il codice PostgreSQL che definisce gli indici usati per ottenere il piano di esecuzione a destra. Indicare inoltre qual è il guadagno in termini di pagine disco che si ottiene.

```
Aggregate  (cost=6095.00..6095.01 ROWS=1 width=8)
->  Hash JOIN  (cost=97.06..6094.72 ROWS=111 width=0)
      Hash Cond: (ie.id_corsostudi = cs.id)
      ->  Seq Scan ON inserogato ie  (cost=0.00..5970.21
      ROWS=7024 width=4)
            Filter: ((annoaccademico)::TEXT =
      '2010/2011'::TEXT)
      ->  Hash  (cost=96.94..96.94 ROWS=10 width=4)
            ->  Seq Scan ON corsostudi cs
      (cost=0.00..96.94 ROWS=10 width=4)
                  Filter: (id < 10)
```

```
Aggregate  (cost=740.03..740.04 ROWS=1 width=8)
->  Nested Loop  (cost=4.77..739.75 ROWS=111 width=0)
      ->  Bitmap Heap Scan ON corsostudi cs
      (cost=4.35..34.42 ROWS=10 width=4)
            Recheck Cond: (id < 10)
            ->  Bitmap INDEX Scan ON c1
      (cost=0.00..4.35 ROWS=10 width=0)
                  INDEX Cond: (id < 10)
      ->  INDEX ONLY Scan USING c2 ON inserogato ie
      (cost=0.42..70.36 ROWS=17 width=4)
            INDEX Cond: ((annoaccademico =
      '2010/2011'::TEXT) AND (id_corsostudi = cs.id))
```

Domanda 3 [8 punti]

Assumendo di avere una base di dati PostgreSQL che contenga le tabelle di questo tema d'esame, scrivere un metodo Python che, associato all'URL `/prenota` secondo il framework Flask, realizzi le seguenti azioni[1]:

1. legga i parametri `codiceFiscale`, identificatore di un volo (3 parametri: `iataCompagnia, numeroVolo, orarioVolo (%Y-%m-%d %H:%M:%S%z)`) e il parametro `isBusiness` ('0' o '1') nell'oggetto `request` associato al metodo,
2. Si connetta alla base di dati 'X' (si assuma di dover specificare solo il nome della basi di dati).
3. Verifichi se c'è un posto a disposizione nella classe indicata.
4. Se c'è un posto, aggiorni la tabella `Volo` e la tabella `Prenotazione` inserendo la prenotazione per il cliente corrente e passi il controllo a `render_template('prenotazioneEffettuata.html')`,

[1] Chiaramente le azioni richieste sono un sottoinsieme di quelle che si dovrebbero fare!

5. altrimenti passi il controllo a `render_template('nessunVoloOPostiEsauriti.html')`.

Scrivere solo il metodo.

Domanda 4 [7 punti]
Scrivere in PostgreSQL l'interrogazione che determina, per ciascun volo dall'aeroporto di MPX all'aeroporto di LGW nel giorno 31 ottobre 2017 (nel fuso orario dell'Europa Centrale), il numero di clienti maschi già prenotati. Il risultato deve mostrare la chiave del volo, i codici dei due aeroporti e il conteggio richiesto.

Domanda 5 [7 punti]
Scrivere il codice PostgreSQL, definendo anche eventuali viste, per rispondere alle seguenti due interrogazioni nel modo più efficace:

(a) Trovare per ogni utente che abbia fatto prenotazioni con almeno due compagnie distinte, il numero di voli già effettuati (finiti) all'istante corrente presso ciascuna compagnia e la loro durata totale, sempre per ciascuna compagnia. Il risultato deve riportare il codice fiscale dell'utente, il codice ICAO della compagnia aerea e i conteggi richiesti.

(b) Trovare per ogni compagnia (specificata solo dal suo ICAO), l'utente (gli utenti) con il maggior numero di voli e l'utente (gli utenti) che ha (hanno) totalizzato il tempo di volo complessivo maggiore, riportando nel risultato: ICAO della compagnia, il codice fiscale dell'utente e i conteggi richiesti (se gli utenti per ciascuna compagnia coincidono, si deve stampare solo una riga). Si può non tenere conto se i voli sono già stati effettuati o no all'instante corrente.

9 Prova del 19/06/2019

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) inerente a una possiile organizzazione di istituti di ricerca:

```
SEZIONE(codice, nome, direttore)
RICERCATORE(codiceFiscale, nome, cognome, sezApp) --sezApp è il codice della sezione in
    cui il ricercatore lavora
PARTECIPA(ricercatore, progetto)PROGETTO(\underline{codice}, obiettivo, CodRespProg, dataInizio,
    dataFine)-CodRespProg è il c.f.  del responsabile del progetto
```

Domanda 1 [7 punti]

1) Determinare i vincoli di integrità che si possono desumere usando la notazione '→'.
2) Creare un dominio per rappresentare il codice fiscale
3) Scrivere il codice PostgreSQL che generi TUTTE le tabelle per rappresentare lo schema relazionale specificando tutti i possibili controlli di integrità referenziale e almeno 2 controlli di correttezza dei dati usando il CHECK.

Suggerimento: PostgreSQL ammette vincoli di integrità referenziali circolari. Essi devono essere aggiunti DOPO la creazione delle tabelle con il comando 'ALTER TABLE ADD constraintTable' e si devono dichiarare con le opzioni 'DEFERRABLE INITIALLY DEFERRED' (sono 2 opzioni!). Tutti gli INSERT/UPDATE/DELETE che riguardano le tabelle con vincoli circolari deve essere fatti dentro delle transazioni.'

Soluzione

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

SEZIONE.direttore → RICERCATORE.codiceFiscale
RICERCATORE.sezApp → SEZIONE.codice
I due vincoli precedenti sono in dipendenza reciproca (circolarità).
PROGETTO.CodRespProg → RICERCATORE.codiceFiscale
PARTECIPA.ricercatore → RICERCATORE.codiceFiscale
PARTECIPA.progetto → PROGETTO.codice

2) e 3)

```
CREATE DOMAIN codiceFiscaleD CHAR(16) CHECK (VALUE SIMILAR TO
     '[A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]');

CREATE TABLE SEZIONE (
   codice VARCHAR PRIMARY KEY,
   nome VARCHAR NOT NULL CHECK (nome<>''),
   direttore codiceFiscaleD NOT NULL
);

CREATE TABLE RICERCATORE (
   codiceFiscale codiceFiscaleD PRIMARY KEY,
   nome VARCHAR NOT NULL CHECK (nome<>''),
   cognome VARCHAR NOT NULL CHECK (cognome<>''),
   sezApp VARCHAR NOT NULL REFERENCES SEZIONE
);

ALTER TABLE SEZIONE ADD FOREIGN KEY(direttore) REFERENCES RICERCATORE DEFERRABLE
     INITIALLY DEFERRED;

CREATE TABLE PROGETTO (
   codice VARCHAR PRIMARY KEY,
   obiettivo VARCHAR NOT NULL CHECK (obiettivo<>''),
```

```
    CodRespProg codiceFiscaleD NOT NULL REFERENCES RICERCATORE,
    dataInizio DATE NOT NULL,
    dataFine DATE NOT NULL,--si può assumere che ci sia una data fine prevista
    CHECK (dataInizio < dataFine)
);

CREATE TABLE PARTECIPA (
    ricercatore codiceFiscaleD  REFERENCES RICERCATORE,
    progetto VARCHAR REFERENCES SEZIONE,
    PRIMARY KEY (ricercatore, progetto)
);
```

Domanda 2 [7 punti]

Scrivere il codice PostgreSQL che determina le soluzioni alle seguenti due interrogazioni usando il minor numero di `JOIN`:

(a) Fra i progetti aventi responsabili della stessa sezione, visualizzare quello(i) con durata maggiore, mostrando il codice del progetto, il codice fiscale del responsabile, il nome della sezione e la durata in giorni.

(b) Trovare i progetti che coinvolgano tutte le sezioni. In altre parole, i progetti dove partecipa almeno un ricercatore per sezione. Mostrare il codice del progetto e il numero totale dei ricercatori che partecipano al progetto.

Soluzione

Una possibile soluzione per la parte (a) prevede di usare una vista per semplificare la scrittura della query finale.

```
(a)
CREATE VIEW DURATA_MASSIMA_PROGETTO_PER_SEZIONE AS
  SELECT R.sezApp AS sezione, MAX(P.dataFine-P.dataInizio) AS durata
  FROM RICERCATORE R JOIN PROGETTO P ON R.codiceFiscale=P.CodRespProg
  GROUP BY R.sezApp;

SELECT P.codice, R.codicefiscale, S.nome, V.durata
FROM SEZIONE S JOIN RICERCATORE R ON R.sezApp=S.codice JOIN PROGETTO P ON
     R.codiceFiscale=P.CodRespProg
   JOIN DURATA_MASSIMA_PROGETTO_PER_SEZIONE V ON S.codice=V.codice
WHERE P.dataFine-P.dataInizio = V.durata;

--(b)
--Una soluzione è basata su una vista che calcola il numero di ricercatori e il
     numero di sezioni per ciascun progetto
CREATE VIEW SEZIONI_PER_PROGETTO AS
  SELECT P.codice, COUNT(DISTINCT R.sezApp) AS n_sezioni, COUNT(R.codiceFiscale)
     AS n_ricercatori
  FROM RICERCATORE R JOIN PARTECIPA PA ON R.codiceFiscale=PA.ricercatore JOIN
     PROGETTO P ON PA.progetto=P.codice
  GROUP BY P.codice;

SELECT V.codice, V.n_ricercatori
FROM SEZIONI_PER_PROGETTO V
WHERE V.n_sezioni = (
   SELECT COUNT(*)
   FROM sezione
);

--Un'altra soluzione è basata ponendo una query innestata nella condizione having
SELECT P.progetto, COUNT(8) AS n_ricercatori
```

```
FROM Partecipa P JOIN Ricercatore R ON (P.ricercatore=R.codiceFiscale)
GROUP BY P.progetto
HAVING COUNT(DISTINCT R.sezApp) = (
   SELECT COUNT(*)
   FROM sezione
);
```

Domanda 3 [7 punti]

Data una base di dati PostgreSQL che contenga le tabelle della Domanda 1, scrivere un programma Java (solo metodo main) che, leggendo il codice di una sezione da console, visualizzi il risultato della query dell'esercizio 2(a). Se il codice della sezione non esiste, il programma deve richiedere il dato.

Soluzione

```
public static void main(String[] args) throws ClassNotFoundException, SQLException {
   Class.forName("org.postgresql.Driver");
   try (Scanner keyb = new Scanner(System.in);
         Connection con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/posenato",
     "posenato", "")) {
      String codiceSezione = null;
      ResultSet rs = null;
      while (codiceSezione == null) {
         System.out.print("Inserire il codice sezione: ");
         codiceSezione = keyb.next();
         try (PreparedStatement pst = con.prepareStatement("SELECT count(*) FROM SEZIONE S WHERE
     s.codice = ?")) {
            pst.clearParameters();
            pst.setString(1, codiceSezione);
            rs = pst.executeQuery();
            rs.next();
            if (rs.getInt(1) != 1) {
               System.out.println("Non esiste la sezione con codice " + codiceSezione " + ". Ripetere
     inserimento.");
               codiceSezione = null;
               continue;
            }
         } catch (SQLException e) {
            System.out.println("Problema durante prima query: " + e.getMessage());
            return;
         }
      }
      //la sezione esiste
      try (PreparedStatement pst = con.prepareStatement(<codice query 2a>+ "AND s.codice = ?")) {
            pst.clearParameters();
            pst.setString(1, codiceSezione);
            rs = pst.executeQuery();
            if (!rs.isBeforeFirst()) {//isBeforeFirst restituisce false se non ci sono righe!
               System.out.println("Non esiste nessun progetto associato alla sezione con codice " +
     codiceSezione);
            } else {
               // rs ha i dati cercati
               System.out.println(String.format(
                  "I progetti di durata massima in giorni della sezione data sono:\n" + "%20s %16s %20s
     %20s %d", "Progetto", "Codice Fiscale", "Sezione", "Sezione", "Durata"));
               while (rs.next()) {
                  System.out.println(String.format("%20s %16s %20s %20s %d", rs.getString(1),
     rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
               }
            }
      } catch (SQLException e) {
            System.out.println("Problema durante seconda query: " + e.getMessage());
            return;
      }
   } catch (SQLException e) {
      System.out.println("Problema durante la connessione iniziale alla base di dati: " +
     e.getMessage());
      return;
   }
}
```

Domanda 4 [5 punti]

Si considerino le tabelle `SEZIONE` e `RICERCATORE` della Domanda 1.

Un vincolo con opzione 'INITIALLY DEFERRED' viene valutato solo all'esecuzione di un commit.

Si scriva quindi uno script SQL che popoli la tabella SEZIONE e la tabella RICERCATORE (che sono vuote) con una nuova sezione e due nuovi ricercatori della sezione, di cui uno è anche il direttore.

Si scriva poi uno script che cambia la sezione al direttore della sezione 'x' e assegni un nuovo direttore alla sezione 'x'. Come si deve scrivere per garantire la coerenza?

Soluzione

```
BEGIN;
   INSERT INTO RICERCATORE VALUES ('AAABBB01A01Z112E', 'AAA', 'BBB', 1),
      ('BBBCCC01A01Z112C', 'BBB', 'CCC', 1);
   INSERT INTO SEZIONE VALUES (1, 'sezione', 'BBBCCC01A01Z112C');
COMMIT;
```

Il secondo script deve essere all'interno di una transazione.

```
BEGIN;
   UPDATE PERSONA SET sezApp = 2 --assumo che sia 2 la nuova sezione
   WHERE codiceFiscale IN (SELECT direttore FROM sezione WHERE codice = x);

   UPDATE sezione SET direttore = 'AAABBB01A01Z112E'
   WHERE codice = x;
COMMIT;
```

Domanda 5 [6 punti]

Si consideri il seguente risultato del comando ANALYZE su una query inerenti a 2 tabelle:

```
GroupAggregate  (cost=7712.31..7919.04 ROWS=7709...)
GROUP KEY: i.nomeins
-> Sort  (cost=7712.31..7755.52 ROWS=17285...)
   Sort KEY: i.nomeins
   -> Hash JOIN  (cost=287.80..6495.68
     ROWS=17285...)
      Hash Cond: (ie.id_insegn = i.id)
      ->  Seq Scan ON inserogato ie
     (cost=0.00..5970.21 ROWS=17285...)
           Filter: ((annoaccademico)::TEXT >
     '2010/2011'::TEXT)
      ->  Hash  (cost=185.69..185.69 ROWS=8169
     width=43)
           ->  Seq Scan ON insegn i
     (cost=0.00..185.69 ROWS=8169 width=43)
```

```
GroupAggregate  (cost=7484.53..7691.26 ROWS=7709...)
GROUP KEY: i.nomeins
-> Sort  (cost=7484.53..7527.75 ROWS=17285...)
   Sort KEY: i.nomeins
   -> Hash JOIN  (cost=694.18..6267.91 ROWS=17285...)
      Hash Cond: (ie.id_insegn = i.id)
      -> Bitmap Heap Scan ON inserogato ie
     (cost=406.38..5742.44 ROWS=17285)
         Recheck Cond: ((annoaccademico)::TEXT >
     '2010/2011'::TEXT)
         -> Bitmap INDEX Scan ON
     inserogato_annoaccademico_idx1  (cost=0.00..402.06
     ROWS=17285 width=0)
               INDEX Cond: ((annoaccademico)::TEXT >
     '2010/2011'::TEXT)
         -> Hash  (cost=185.69..185.69 ROWS=8169 width=43)
            -> Seq Scan ON insegn i  (cost=0.00..185.69
     ROWS=8169 width=43)
```

A sinistra la query originale, a destra la query dopo la creazione di uno o più indici.

Desumere il testo della query e scrivere il codice del/degli indici creati.

Soluzione

È un hash join, quindi c'è un JOIN con condizione ie.id_insegn = i.id Poi c'è un Sep Scan con condizione (annoaccademico)::text > '2010/2011'. Infine, c'è un group key. Quindi c'è un group by i.nomeins.

```
SELECT i.nomeins, COUNT(DISTINCT ie.annoaccademico) AS naa
FROM insegn i JOIN inserogato ie ON ie.id_insegn=i.id
WHERE ie.annoaccademico> '2010/2011'
GROUP BY(i.nomeins);
```

Lo studente non era tenuto a scrivere `COUNT(DISTINCT ie.annoaccademico) AS naa` perché non desumibile dalla query, mentre `i.nomeins` era da inserire perché molto probabile visto il `GROUP BY`. Sicuramente NON si poteva inserire `*`.

L'unico indice creato è quello relativo all'anno accademico. Infatti nel secondo explain, compare Index Cond: ((annoaccademico)::text > '2010/2011'::text)

```
CREATE INDEX ON inserogato (annoaccademico);
```

10 Prova del 04/07/2019

Schema base di dati

Si consideri il seguente schema relazionale (chiavi primarie sottolineate):
`NEGOZI(codice, nome, città)` *--città deve essere selezionabile da una tabella*
`PRODOTTI(codice, nome, marca)` *--marca deve essere selezionabile da una tabella*
`LISTINO(negozio, prodotto, prezzo)`
con vincoli di integrità (oltre a quelli indirettamente indicati nei commenti) fra gli attributi di `LISTINO` e le chiavi di `NEGOZI e PRODOTTI`.

Domanda 1 [7 punti]
1) Indicare i 4 vincoli di integrità usando la notazione '→'.
2) Scrivere il codice PostgreSQL che generi TUTTE le tabelle per rappresentare lo schema relazionale specificando tutti gli attributi, almeno 2 controlli di correttezza dei dati usando il CHECK, MA SENZA I vincoli di integrità referenziale.
3) Scrivere tutti i comandi `ALTER TABLE` che aggiungono SOLO tutti i vincoli di integrità referenziale.

Domanda 2 [7 punti]
Scrivere il codice PostgreSQL che determina le soluzioni alle seguenti due interrogazioni usando il minor numero di `JOIN` e NON usando il DATA BINDING:

(a) Per ciascuna città, determinare i prodotti con prezzo maggiore (nella città) visualizzando il nome della città, il codice del negozio dove viene venduto il prodotto, il codice e nome del prodotto, e il suo prezzo.

(b) Trovare i prodotti che vengono venduti in una sola città mostrando codice e nome prodotto e il nome della città, ordinati per nome città.

Domanda 3 [7 punti]
Data una base di dati PostgreSQL che contenga le tabelle e viste della Domanda 1, scrivere un programma Python che, leggendo il codice di un prodotto da console, visualizzi la città risultato della query dell'esercizio 2(b) (adattata al fatto che si fornisce il codice del prodotto). In particolare, se il prodotto non è venduto o venduto in più città, il programma deve segnalarlo. Se, invece, viene venduto solo in una città, il programma deve visualizzare la città. Se il codice del prodotto non esiste, il programma deve segnalarlo e richiedere il dato.

Domanda 4 [6 punti]
Si scrivano due transazioni in cui: 1) nella prima transazione si aumenta del 5% il prezzo a tutti i prodotti dei negozi della città Verona, mentre 2) nell'altra si abbassa il prezzo del 10% a tutti i farmaci che hanno prezzo maggiore o uguale a 1050 euro. Che tipo di errore si può verificare se le due transazioni sono concorrenti? Che tipo di isolamento si deve dichiarare per eliminare l'errore? In quale transazione?

Domanda 5 [5 punti]
Si consideri il seguente risultato del comando ANALYZE su una query inerenti a 3 tabelle:

```
GroupAggregate (cost=8994..9232 ROWS=7709)
 GROUP KEY: i.nomeins
 Filter: (COUNT(DISTINCT d.id_persona) > 2)
 -> Sort (cost=8994..9026 ROWS=12927)
     Sort KEY: i.nomeins
     -> Hash JOIN (cost=6474..8111 ROWS=12927)
         Hash Cond: (ie.id_insegn = i.id)
         -> Hash JOIN (cost=6186..7645 ROWS=12927)
             Hash Cond: (d.id_inserogato = ie.id)
             -> Seq Scan ON docenza d (cost=0..1139
     ROWS=50867)
             -> Hash (cost=5970..5970 ROWS=17285)
                 -> Seq Scan ON inserogato ie
     (cost=0..5970 ROWS=17285)
                       Filter:
     ((annoaccademico)::TEXT > '2010/2011'::TEXT)
         -> Hash (cost=185..185 ROWS=8169)
             -> Seq Scan ON insegn i (cost=0..185
     ROWS=8169)
```

```
GroupAggregate  (cost=8770.13..9009.12 ROWS=7709)
GROUP KEY: i.nomeins
Filter: (COUNT(DISTINCT d.id_persona) > 2)
-> Sort (cost=8770..8802 ROWS=12952)
    Sort KEY: i.nomeins
    -> Hash JOIN (cost=6247..7885 ROWS=12952)
          Hash Cond: (ie.id_insegn = i.id)
          -> Hash JOIN (cost=5959..7419 ROWS=12952)
                Hash Cond: (d.id_inserogato = ie.id)
                -> Seq Scan ON docenza d (cost=0..1139
    ROWS=50867)
                -> Hash (cost=5743..5743 ROWS=17319)
                      ->  Bitmap Heap Scan ON inserogato
    ie (cost=406..5743 ROWS=17319)
                            Recheck Cond:
    ((annoaccademico)::TEXT >'2010/2011'::TEXT)
                            -> Bitmap INDEX Scan ON i1
    (cost=0..402 ROWS=17319)
                                  INDEX Cond:
    ((annoaccademico)::TEXT >'2010/2011'::TEXT)
          -> Hash  (cost=185..185 ROWS=8169)
                -> Seq Scan ON insegn i (cost=0..185
    ROWS=8169)
```

A sinistra la query originale, a destra la query dopo la creazione di uno o più indici.

Desumere il testo della query e scrivere il codice del/degli indici creati.
*Suggerimento: La clausola SELECT ha una parte che si può desumere perché molto probabile, una parte no. Scrivere solo la parte che si può desumere senza aggiungere * o altro!*