SOLUZIONI

Si consideri la realtà medica descritta dalla base di dati relazionale definita dal seguente schema:

PAZIENTE(CodFiscale, Cognome, Nome, Sesso, DataNascita, Citta, Reddito)
MEDICO(Matricola, Cognome, Nome, Specializzazione, Parcella, Citta)
FARMACO(NomeCommerciale, PrincipioAttivo, Costo, Pezzi)
PATOLOGIA(Nome, ParteCorpo, SettoreMedico, Invalidita, PercEsenzione)
INDICAZIONE(Farmaco, Patologia, DoseGiornaliera, NumGiorni, AVita)
VISITA(Medico, Paziente, Data, Mutuata)
ESORDIO(Paziente, Patologia, DataEsordio, DataGuarigione, Gravita, Cronica)
TERAPIA(Paziente, Patologia, DataEsordio, Farmaco, DataInizioTerapia, DataFineTerapia, Posologia)

Risolvere i seguenti esercizi utilizzando la sintassi MySQL. La correttezza dei primi due esercizi è una condizione necessaria per la correzione dell'intero elaborato. Quelle che seguono sono possibili soluzioni degli esercizi proposti. Soluzioni alternative sono corrette purché producano lo stesso risultato e siano semanticamente equivalenti a quelle proposte.

Esercizio 1

Senza fare uso di join né viste, scrivere una query che restituisca nome e cognome dei medici che hanno visitato tutti i pazienti di Roma.

```
SELECT M.Nome, M.Cognome
FROM Medico M
WHERE NOT EXISTS(
      SELECT *
      FROM Paziente P
      WHERE P.Citta = 'Roma'
            AND NOT EXISTS(
                SELECT *
                FROM Visita V
                WHERE V.Paziente = P.CodFiscale
                      AND V.Medico = M.Matricola
            )
     );
```

Esercizio 2

Scrivere una query che restituisca le patologie che nel 2011 hanno colpito, con tasso d'incidenza complessivo superiore al 90%, bambini di età inferiore a 5 anni o anziani di età superiore a 75 anni.

```
SELECT E.Patologia
FROM Paziente P INNER JOIN Esordio E
     ON P.CodFiscale = E.Paziente
WHERE YEAR(E.DataEsordio) = 2011
      AND (P.DataNascita + INTERVAL 5 YEAR > E.DataEsordio
           OR P.DataNascita + INTERVAL 75 YEAR < E.DataEsordio)
GROUP BY E.Patologia
```

```
HAVING COUNT(DISTINCT E.Paziente) >
       0.9*(SELECT COUNT(*)
            FROM Paziente P
            WHERE YEAR(P.DataNascita + INTERVAL 5 YEAR) > 2011
                  OR YEAR(P.DataNascita + INTERVAL 75 YEAR) > 2011
            );
```

Esercizio 3

Scrivere una query che aumenti di un'unità il dosaggio giornaliero indicato di tutti i farmaci per la cura di sole patologie a carico della tiroide che sono sempre stati assunti, tranne nei casi di patologie croniche, a dosaggi superiori rispetto a quello indicato.

```
UPDATE Indicazione I NATURAL JOIN
       (SELECT I2.Farmaco
        FROM Indicazione I2 INNER JOIN Patologia P ON I2.Patologia = P.Nome
        WHERE P.ParteCorpo = 'Tiroide'
              AND I2.Farmaco NOT IN(
                  SELECT I3.Farmaco
                  FROM Indicazione I3 INNER JOIN Patologia P2
                       ON I3.Patologia = P2.Nome
                  WHERE P2.ParteCorpo <> 'Tiroide')
       ) AS D
SET I.DoseGiornaliera = I.DoseGiornaliera + 1
WHERE NOT EXISTS (SELECT *
                  FROM Terapia T NATURAL JOIN Esordio E
                  WHERE T.Farmaco = I.Farmaco
                        AND T.Posologia <= I.DoseGiornaliera
                        AND E.Cronica = 'no');
```

Esercizio 4

Una materialized view MV_MERITO(Specializzazione, TotaleVisite, NuoviPazienti) contiene le specializzazioni della clinica i cui medici, nel corso della settimana precedente, hanno totalizzato un numero di visite superiore alla media delle visite per specializzazione nella stessa settimana, e tale numero di visite abbia coinvolto almeno cinque nuovi pazienti. Creare la materialized view e implementare l'incremental refresh: (i) creando la log table; (ii) scrivendo il codice per la gestione della log table; (iii) scrivendo il temporal trigger per il full refresh, eseguito ogni lunedì alle ore 2:00 del mattino.

```
CREATE TABLE MV_Merito (
      Specializzazione CHAR(100) NOT NULL,
      TotaleVisite INT NOT NULL,
      NuoviPazienti INT NOT NULL,
      PRIMARY KEY (Specializzazione),
      CONSTRAINT FK_Specializzazione FOREIGN KEY (Specializzazione)
                 REFERENCES Medico (Specializzazione)
                 ON UPDATE NO ACTION
                 ON DELETE NO ACTION
) ENGINE=`InnoDB`;


CREATE TABLE MV_Merito_LOG LIKE Visita;

ALTER TABLE MV_Merito_LOG
ADD COLUMN NuovoPaziente TINYINT(1) NOT NULL;
```

```
-- Trigger per la gestione della log table
DELIMITER $$
CREATE TRIGGER LogInsert
AFTER INSERT ON Visita
FOR EACH ROW

BEGIN

      DECLARE nuovoPaziente TINYINT(1) DEFAULT 0;
      DECLARE specializzazione CHAR(50) DEFAULT '';

      SELECT M.Specializzazione INTO specializzazione
      FROM Medico M
      WHERE M.Matricola = NEW.Matricola;

      SELECT COUNT(*) INTO nuovoPaziente
      FROM Visita V INNER JOIN Medico M
           ON V.Medico = M.Matricola
      WHERE V.Data <= NEW.Data
            AND M.Specializzazione = specializzazione;

      INSERT INTO MV_Merito_LOG VALUES
      (NEW.Medico, NEW.Paziente, NEW.Data, NEW.Mutuata,
       IF(nuovopaziente = 0, 1, 0));

END $$


-- Event per l'incremental full refresh della materialized view
CREATE EVENT Refresh_MV_Merito
ON SCHEDULE EVERY 1 WEEK
STARTS CURRENT_DATE + INTERVAL
       (9 - IF(DAYOFWEEK(CURRENT_DATE)=1, 8, DAYOFWEEK(CURRENT_DATE))) DAY
DO
  BEGIN

    DECLARE finito INTEGER DEFAULT 0;
    DECLARE mediaVisite DOUBLE DEFAULT 0;
    DECLARE specializzazione CHAR(50) DEFAULT '';
    DECLARE visite INTEGER DEFAULT 0;
    DECLARE nuovi INTEGER DEFAULT 0;


    -- per ogni specializzazione, si estrae dalla log table il numero
    -- di visite effettuate dai relativi medici nella settimana precedente,
    -- e il numero di nuovi pazienti visitati nella settimana precedente
    CREATE cursoreSpecializzazioni CURSOR FOR
    SELECT M.Specializzazione, COUNT(*) AS NumeroVisite, SUM(NuovoPaziente)
    FROM MV_Merito_LOG MVL INNER JOIN Medico M ON MVL.Medico = M.Matricola
    GROUP BY M.Specializzazione
    HAVING SUM(NuovoPaziente) > 5;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finito = 1;
```

```
    -- numero medio di visite per specializzazione della settimana scorsa
    SELECT AVG(D.NumeroVisite) INTO mediaVisite
    FROM(
         SELECT COUNT(*) AS NumeroVisite
         FROM MV_Merito_LOG MVL INNER JOIN Medico M
              ON MVL.Medico = M.Matricola
         GROUP BY M.Specializzazione
        ) AS D;

    -- flushing della materialized view
    TRUNCATE TABLE MV_Merito;

    OPEN cursoreSpecializzazioni;

    scan: LOOP

       FETCH cursoreSpecializzazioni INTO specializzazione, visite, nuovi;

       IF finito = 1 THEN
          LEAVE scan;
       END IF;

       -- aggiornamento della materialized view
       INSERT INTO MV_Merito VALUES (specializzazione, visite, nuovi);

    END LOOP scan;

    CLOSE cursoreSpecializzazioni;

    -- flushing della log table
    TRUNCATE TABLE MV_Merito_LOG;

  END $$

DELIMITER ;
```

A.A. precedente:

Una tabella ridondante MERITO(Specializzazione, TotaleVisite, NuoviPazienti) contiene le specializzazioni della clinica i cui medici, nel corso della settimana precedente, hanno totalizzato un numero di visite superiore alla media delle visite per specializzazione nella stessa settimana, e tale numero di visite abbia coinvolto almeno cinque nuovi pazienti. Creare la tabella, specificarne la chiave primaria e i vincoli di integrità referenziale, e scrivere un temporal trigger che la aggiorni settimanalmente, supponendola già popolata.

```
CREATE TABLE Merito (
      Specializzazione CHAR(100) NOT NULL,
      TotaleVisite INT NOT NULL,
      NuoviPazienti INT NOT NULL,
      PRIMARY KEY (Specializzazione),
      CONSTRAINT FK_Specializzazione FOREIGN KEY (Specializzazione)
                 REFERENCES Medico (Specializzazione)
                 ON UPDATE NO ACTION
                 ON DELETE NO ACTION
) ENGINE=`InnoDB`;
```

```
DELIMITER $$

CREATE EVENT Aggiorna_Merito
ON SCHEDULE EVERY 1 WEEK
DO
  BEGIN

  -- numero medio di visite per specializzazione della settimana scorsa
  SET @mediaVisite =
      SELECT AVG(D.NumeroVisite)
      FROM(
           SELECT COUNT(*) AS NumeroVisite
           FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola
           WHERE V.Data BETWEEN CURRENT_DATE
                 AND CURRENT_DATE - INTERVAL 7 DAY
           GROUP BY M.Specializzazione
      ) AS D;

  -- cancellazione del contenuto della tabella Merito
  DELETE FROM Merito;

  -- aggiornamento delle informazioni ridondanti della tabella Merito
  INSERT INTO Merito
  SELECT D.Specializzazione, D.TotaleVisite, D.NuoviPazienti
  FROM(
       SELECT M.Specializzazione, COUNT(*) AS NuoviPazienti,
              (SELECT COUNT(*)
               FROM Visita V0 INNER JOIN Medico M0
                    ON V0.Medico = M0.Matricola
               WHERE M0.Specializzazione = M.Specializzazione
                     AND V0.Data BETWEEN CURRENT_DATE
                         AND CURRENT_DATE - INTERVAL 7 DAY
              ) AS TotaleVisite
       FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola
       WHERE V.Data BETWEEN CURRENT_DATE AND CURRENT_DATE - INTERVAL 7 DAY
             AND V.Paziente NOT IN(
                 SELECT V2.Paziente
                 FROM Visita V2 INNER JOIN Medico M2
                      ON V2.Medico = M2.Matricola
                 WHERE M2.Specializzazione = M.Specializzazione
                       AND V2.Data < V.Data)
       GROUP BY M.Specializzazione
       HAVING COUNT(*) > 5
      ) AS D
  WHERE D.TotaleVisite > @mediaVisite;

  END $$

DELIMITER ;
```