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 del primo esercizio 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

Scrivere una query che restituisca, se esiste, la città dalla quale proviene il maggior numero di pazienti che hanno contratto l'acufene un numero di volte maggiore o uguale a quello degli altri pazienti della loro città.

```
CREATE OR REPLACE VIEW EsordiAcufene AS
SELECT P.CodFiscale,
       P.Citta,
       COUNT(*) AS NumeroEsordi
FROM Esordio E
     INNER JOIN
     Paziente P ON E.Paziente = P.CodFiscale
WHERE E.Patologia = 'Acufene'
GROUP BY P.CodFiscale, P.Citta;

CREATE OR REPLACE VIEW PazientiCitta
SELECT EA.Citta,
       COUNT(*) AS NumeroPazienti
FROM EsordiAcufene EA
WHERE EA.NumeroEsordi >= ALL
      (
       SELECT EA2.NumeroEsordi
       FROM EsordiAcufene EA2
       WHERE EA2.Citta = EA.Citta
      )
GROUP BY EA.Citta;

SELECT PC.Citta
FROM PazientiCitta PC
WHERE PC.NumeroPazienti > ALL
      (
       SELECT PC2.NumeroPazienti
       FROM PazientiCitta PC2
       WHERE PC2.Citta <> PC.Citta
      );
```

Esercizio 2

Data la seguente query, descriverne il risultato e scriverne la versione join-equivalente senza usare view.

```
SELECT T1.Farmaco
FROM Terapia T1
WHERE NOT EXISTS(
        SELECT *
        FROM Terapia T2
        WHERE T2.Farmaco = T1.Farmaco
              AND EXISTS(
                   SELECT *
                   FROM Terapia T3
                   WHERE T3.DataInizioTerapia < T2.DataInizioTerapia
                         AND T3.Farmaco = T2.Farmaco
                  )
        );
```

```
/*
Nome commerciale dei farmaci assunti in un'unica terapia.
*/

SELECT T.Farmaco
FROM Terapia T
GROUP BY T.Farmaco
HAVING COUNT(*) = 1;
```

Esercizio 3

Con cadenza imprevedibile, la direzione della clinica è interessata a conoscere, per ciascuna specializzazione, il numero di nuovi pazienti visitati, il medico che effettua il numero minore di prime visite, e il numero di città diverse dalle quali provengono i pazienti visitati per la prima volta. Si desidera creare uno snapshot REPORT aggiornato mediante partial refresh con tecnica on demand. Pertanto, si richiede: i) la creazione dello snapshot; ii) la creazione della log table e il codice di gestione della stessa; iii) una o più stored procedure per implementare il partial refresh. Le stored procedure non devono fare uso di subquery né view.

```
/* creazione snapshot */
CREATE TABLE Report(
    Specializzazione CHAR(100) NOT NULL,
    PazientiNuovi INTEGER DEFAULT 0,
    MedicoInf CHAR(255) DEFAULT '',
    CittaDiverse INTEGER DEFAULT 0
);

/* creazione log table */
CREATE TABLE Report_Log LIKE Visita;

ALTER TABLE Rerport_Log
DROP COLUMN Mutuata
ADD COLUMN Specializzazione CHAR(100) NOT NULL FIRST;

/* trigger per push */
DROP TRIGGER IF EXISTS PushVisita;

DELIMITER $$

CREATE TRIGGER PushVisita
AFTER INSERT ON Visita
FOR EACH ROW
BEGIN

    DECLARE _specializzazione CHAR(100) DEFAULT '';
```

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

    INSERT INTO Report_Log
    VALUES
    (_specializzazione, NEW.Medico, NEW.Paziente, NEW.`Data`);

END $$

/* stored procedure per nuovi pazienti e città diverse data specializzazione */
DROP PROCEDURE IF EXISTS infoNuoviPazienti $$

CREATE PROCEDURE dettaglio_pazienti (IN _spec CHAR(100),
                                     OUT npaz_ INTEGER,
                                     OUT ncitta_ INTEGER
                                    )
BEGIN
    SELECT COUNT(*) AS NuoviPazienti INTO npaz_,
           COUNT(DISTINCT P.Citta) INTO ncitta_
    FROM Paziente P
         INNER JOIN
         Report_Log RL ON P.CodFiscale = RL.Paziente;
         LEFT OUTER JOIN /* PER INDIVIDUARE LE PRIME VISITE NEL LOG */
         Visita V ON(
                     RL.Medico = V.Medico
                     AND RL.Paziente = V.Paziente
                     AND V.Data < RL.Data
                    )
    WHERE V.Paziente IS NULL
          AND RL.Specializzazione = _spec
          AND RL.Data <= _data;
END $$

/* stored procedure per medico con numero minore di nuovi pazienti */
DROP PROCEDURE IF EXISTS medicoMinNuoviPazienti $$

CREATE PROCEDURE medicoMinNuoviPazienti(IN _spec CHAR(100),
                                        IN _data DATE,
                                        OUT medico_ CHAR(255),
                                       )
 BEGIN

    DECLARE medico CHAR(100) DEFAULT '';
    DECLARE visite INTEGER DEFAULT 0;
    DECLARE medicoMin CHAR(100) DEFAULT '';
    DECLARE minimo INTEGER DEFAULT 0;
    DECLARE i INTEGER DEFAULT 0;
    DECLARE finito INTEGER DEFAULT 0;

    DECLARE primeVisite
    CURSOR FOR
    SELECT RL.Medico,
           IFNULL(RL.Medico, 0, COUNT(*)) AS Visite
    FROM Report_Log RL
         RIGHT OUTER JOIN /* CI SONO MEDICI CHE NON HANNO EFFETTUATO PRIME VISITE */
         Medico M ON RL.Medico = M.Matricola
    WHERE RL.Specializzazione = _spec
          AND RL.Data <= _data
    GROUP BY RL.Medico;
```

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

    OPEN primeVisite;

    scan: LOOP

        FETCH primeVisite INTO medico, visite;

        IF finito = 1 THEN
            LEAVE scan;
        END IF;

        IF (visite <= minimo OR i=0) THEN
            BEGIN
                SET minimo = visite;
                SET medicoMin = medico;
                /*SET medicoMin = CONCAT(medicoMin, medico);*/
            END;
        END IF;

        SET i = i+1;

    END LOOP scan;

    CLOSE primeVisite;

    SET medico_ = medicoMin;

END $$

/* partial refresh incrementale */
DROP PROCEDURE IF EXISTS partial_refresh_report $$

CREATE PROCEDURE partial_refresh_report(IN soglia DATE)

BEGIN

    DECLARE spec CHAR(100) DEFAULT '';
    DECLARE primeVisite INTEGER DEFAULT 0;
    DECLARE citta INTEGER DEFAULT 0;
    DECLARE medicoWorst CHAR(100) DEFAULT '';
    DECLARE finito INTEGER DEFAULT 0;

    DECLARE specializzazioni
    CURSOR FOR
    SELECT DISTINCT M.Specializzazione
    FROM Medico M;

    DECLARE CONTINUE HANDLER
    FOR NOT FOUND
    SET finito = 1;

    OPEN specializzazioni;

    scan: LOOP

        FETCH specializzazioni INTO spec;

        IF finito = 1 THEN
            LEAVE scan;
        END IF;
```

```
        CALL infoNuoviPazienti(spec, soglia, primeVisite, citta);
        CALL medicoMinNuoviPazienti(spec, soglia, medicoWorst);

        REPLACE INTO Report
        VALUES(spec, primeVisite, medicoWorst, citta);

        DELETE FROM Report_Log
        WHERE Data <= soglia;

    END LOOP scan;

    CLOSE specializzazioni;

END $$

DELIMITER ;
```

A.A. precedente

Con cadenza imprevedibile, la direzione della clinica è interessata a conoscere, per ciascuna specializzazione, il numero di nuovi pazienti visitati, il medico che effettua il numero minore di prime visite, e il numero di città diverse dalle quali provengono i pazienti visitati per la prima volta. Si desidera creare una tabella REPORT contenente le precedenti informazioni. Dopo aver creato la tabella, scrivere il codice per popolarla.

```
CREATE TABLE Report(
    Specializzazione CHAR(100) NOT NULL,
    PazientiNuovi INTEGER DEFAULT 0,
    MedicoInf CHAR(255) DEFAULT '',
    CittaDiverse INTEGER DEFAULT 0
);

CREATE TABLE LastReport(
    DataUltimoReport DATE DEFAULT CURRENT_DATE - INTERVAL 1 MONTH);  /* soggettivo */
);

SET @last_report_date =
   (
    SELECT DataUltimoReport
    FROM LastReport
   );

CREATE OR REPLACE VIEW  VisiteNuoviPazienti AS
SELECT V.Medico,
       M.Specializzazione,
       V.Paziente
FROM Visita V
     INNER JOIN
     Medico M ON V.Medico = M.Matricola
WHERE V.Data >= @last_report_date
      AND NOT EXISTS
        (
         SELECT *
         FROM Visita V2
         WHERE V2.Medico = V.Medico
               AND V2.Paziente = V.Paziente
               AND V2.Data < @last_report_date
        );

CREATE OR REPLACE VIEW NuoviPazientiNuoveCitta AS
SELECT VNP.Medico,
       VNP.Specializzazione,
       COUNT(DISTINCT NPPS.Paziente) AS NuoviPazienti,
       COUNT(DISTINCT P.Citta) AS NuoveCitta
```

```
FROM VisiteNuoviPazienti VNP
     INNER JOIN
     Paziente P ON VNP.Paziente = P.CodFiscale
GROUP BY VNP.Medico, VNP.Specializzazione;


CREATE OR REPLACE VIEW MinimoNuoviPazientiSpecializzazione AS
SELECT NPPC.Specializzazione,
       MIN(NuoviPazienti) AS MinimoNuoviPazienti
FROM NuoviPazientiNuoveCitta NPNC
GROUP BY NPPC.Specializzazione;


/* La dipendenza funzionale Specializzazione->Medico è garantita dalla clausola HAVING*/
CREATE OR REPLACE VIEW MediciPeggiori AS
SELECT NPPC.Specializzazione,
       NPPC.Medico,
       MNPS.MinimoNuoviPazienti
FROM NuoviPazientiNuoveCitta NPPC
     NATURAL JOIN
     MinimoNuoviPazientiSpecializzazione MNPS
GROUP BY NPPC.Specializzazione
HAVING COUNT(*) = 1;

/*
   Il contesto di transazione (wrapper START TRANSACTION-COMMIT) benché necessario,
   non era richiesto nell'esercizio
*/

START TRANSACTION;

    TRUNCATE TABLE Report;

    INSERT INTO Report
    SELECT NPNC.Specializzazione,
           NPNC.NuoviPazienti,
           MP.Medico,
           NPNC.NuoveCitta
    FROM NuoviPazientiNuoveCitta NPNC
         NATURAL JOIN
         MediciPeggiori MP;

    UPDATE LastReport
    SET DataUltimoReport = CURRENT_DATE;

COMMIT;
```