Oggetto: Analisi dettagliata Orale Pistolesi-Vaglini

Data: 22/06/2021
Versione: V 1.0
Autori: Lorenzo Valtriani - l.valtriani2@studenti.unipi.it

Sommario Contenuti del documento

PISTOLESI

Testo d'esame unico, da fare entro 40/45 minuti, da inviare per posta al prof.
Se non è finito allora lo discuterà all'esame.
Non è possibile fare domande alcune.

"Considerati i soli pazienti di Pisa e Firenze che hanno contratto al massimo una patologia per settore medico (una o più volte), scrivere una query che, per ogni paziente, restituisca il nome, il cognome, la città, il farmaco usato nel maggior numero di terapie, considerando nel complesso le varie patologie, e la posologia media. In caso di pari merito fra i farmaci usati da un paziente, completare il record con valori NULL."

Mia soluzione (Non prendetela per giusta)

```sql
WITH PazientiMaxPatologiaXSettore AS (
    SELECT P.*
    FROM Paziente P
         LEFT OUTER JOIN
        (
            SELECT DISTINCT P.CodFiscale
            FROM Paziente P
                 INNER JOIN
                 Esordio E ON E.Paziente = P.CodFiscale
                 INNER JOIN
                 Patologia PA ON PA.Nome = E.Patologia
            GROUP BY P.CodFiscale, PA.SettoreMedico
            HAVING COUNT(DISTINCT PA.Nome) > 1
        ) AS D ON P.CodFiscale = D.CodFiscale
    WHERE D.CodFiscale IS NULL
          AND (P.Citta = "Pisa" OR P.Citta = "Firenze")
),
FarmaciUsatiDaPazienti AS (
    SELECT T.Paziente, PMPXS.Nome, PMPXS.Cognome, PMPXS.Citta, T.Farmaco, COUNT(*) AS Usi, AVG(T.Posologia) as PosMedia
    FROM PazientiMaxPatologiaXSettore PMPXS
         INNER JOIN
         Terapia T ON T.Paziente = PMPXS.CodFiscale
    GROUP BY T.Paziente, T.Farmaco
),
FarmaciPiuUsatiDaPazienti AS (
    SELECT *
    FROM FarmaciUsatiDaPazienti FUP
    WHERE FUP.Usi >= ALL (
                            SELECT FUP0.Usi
                            FROM FarmaciUsatiDaPazienti FUP0
                            WHERE FUP0.Paziente = FUP.Paziente
                                  AND FUP0.Farmaco <> FUP.Farmaco
                        )
),
FarmacoPiuUsatoDaPazienti AS (
    SELECT DISTINCT FPUDP.Paziente, FPUDP.Nome, FPUDP.Cognome, FPUDP.Citta, D.Farmaco, D.PosMedia
    FROM FarmaciPiuUsatiDaPazienti FPUDP
         LEFT OUTER JOIN
        (
            SELECT *
            FROM FarmaciPiuUsatiDaPazienti FPUDP0
            GROUP BY FPUDP0.Paziente
            HAVING COUNT(*) = 1
        ) AS D ON FPUDP.Paziente = D.Paziente
)
SELECT *
FROM FarmacoPiuUsatoDaPazienti
```

DOMANDE ALL'ORALE:

- Differenza tra Stored-Procedure e Stored-Function:
  - *Stored Procedure*, un programma che sfrutta SQL a livello procedurale, che ha anche più parametri di input e anche di output. Non può ritornare un result-set, se non con una temporary table. Si chiama attraverso la CALL
  - *Stored-Function*, una funzione che ritorna solo un valore, può essere deterministica se per lo stesso input ritorna sempre lo stesso output, oppure non deterministica. La funzione viene chiamata all'interno del codice dichiarativo o procedurale, il modo più facile è scrivere SELECT function()
- Dato un codice fiscale e dato un intero che rappresenta un mese, restituisce il numero medio di visite del paziente in quel mese fra tutte le specializzazioni

```
4   DELIMITER $$
5   DROP FUNCTION IF EXISTS funzione;
6   CREATE FUNCTION funzione(_CodFiscale VARCHAR(10), _Mese INT)
7   RETURNS DOUBLE DETERMINISTIC
8   BEGIN
9       DECLARE VisiteMedieMese DOUBLE DEFAULT 0;
10
11      SELECT AVG(NViste) INTO VisiteMedieMese
12      FROM
13          (
14              SELECT M.Specializzazione, COUNT(*) AS NVisite
15              FROM Visita V
16                   INNER JOIN
17                   Paziente P ON P.CodFiscale = V.Paziente
18                   INNER JOIN
19                   Medico M ON M.Matricola = V.Medico
20              WHERE P.CodFiscale = _CodFiscale AND MONTH(V.Data) = _Mese
21              GROUP BY M.Specializzazione
22           ) AS D;
23
24      RETURN VisiteMedieMese;
25  END $$
26  DELIMITER ;
```

- Differenza tra Group By e Partition By:
  - Il Group By effettua un raggruppamento di tutte le righe della tabella con gli attributi su cui raggruppiamo sono uguali, viene fatto lo squash e quindi comprime i record, il risultato della query di aggregamento contiene un valore minore e uguale delle righe della tabella di partenza.
  - Il Partition By non effettua lo squash però raggruppa allo stesso modo e ritorna una tabella con lo stesso numero di righe della tabella iniziale. La utilizziamo sempre con funzioni di aggregazione.
- Cosa è una Materialized View:
  - Esistono queste tabelle vere e proprie per far si che si restituisca il risultato di una query. Usate quando bisogna eseguire alcune query molto complesse.
- Politiche di refresh per le Materialized View:
  - Immediate: Effettuate con dei trigger, per ogni nuovo inserimento sui row data che potrebbero far sì che la Materialized View sia obsoleta. Quest'ultima sarà sempre aggiornata però ci sarà un costo maggiore molte azioni da parte dell'utente che andrebbero a chiamare questo trigger.
  - On Demand: Effettuate con delle Stored Procedure, viene aggiornata attraverso una chiamata di quest'ultima, ci sarà molto probabilmente sempre un momento in cui non è aggiornata, ma non ci sono costi aggiuntivi per ogni insert.

- Come si progetta una Log-Table:
    - E' una tabella effettiva che serve per alleggerire i carichi dell'Incremental Refresh che non vanno ad effettuare una query sui row data, ma sulla log table che è assai più piccola in numero di righe.
- Trigger che blocca un inserimento di una terapia se non sono passate più di 2 settimane dall'utilizzo dello stesso farmaco.

```
DROP TRIGGER IF EXISTS DropTerapia;
DELIMITER $$
CREATE TRIGGER DropTerapia
BEFORE INSERT ON Terapia FOR EACH ROW
BEGIN

    IF EXISTS (
        SELECT *
        FROM Terapia T
        WHERE T.Paziente = NEW.Paziente
              AND T.Farmaco = NEW.Farmaco
              AND DATEDIFF(CURRENT_DATE, T.DataFineTerapia) < 14
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Errore';
    END IF;
END $$
DELIMITER ;
```

- Possiamo sostituire l'uso del Group By con altri metodi, senza fare raggruppamenti?
  Si, basta usare le Subquery Correlate nel Where
- Correlated-Subquery:
  Una correlated subquery differisce dalla non-correlated perché come da nome fa esistere una correlazione tra query esterna e query interna
- Funzioni Lead-Lag
- Per ogni visita ortopedica si vuole trovare il tempo trascorso, per ogni paziente, della visita precedente con lo stesso medico e con un altro medico della stessa specializzazione
- Cosa sono i cursori e come funzionano, sono sempre necessari?
- Handler not found come funziona
  L'Handler è un gestore di situazioni nel caso visto a lezione l'handler gestisce il caso di un cursore vuoto e tramite una variabile di appoggio gestisce il caso in cui il cursore non ha più dati da estrarre
- Cosa fa questa query? Qual'è la versione Join Equivalente?

```
SELECT M.Citta, COUNT(*)
FROM Medico M
WHERE M.Specializzazione = 'Cardiologia'
      AND NOT EXISTS
        (
         SELECT *
         FROM Visita V
         WHERE V.Medico = M.Matricola
        )
GROUP BY M.Citta;
```

- Trovare i medici che hanno visitato almeno un paziente più di 5 volte
- Trovare la matricola dei medici che hanno visitato per la prima volta almeno un paziente nel mese in corso

```sql
SELECT DISTINCT M.Matricola
FROM Medico M
     LEFT OUTER JOIN
     (
        SELECT DISTINCT M.Medico
        FROM Medico M
             INNER JOIN
             Visita V ON V.Medico = M.Matricola
       -- WHERE condizione sulla data
     ) AS D ON D.Medico = M.Matricola
WHERE D.Medico IS NULL
```

- Le query Correlate possono essere portate nelle query non correlate?
  - Si, in alcuni casi.
- Fare una procedura che prenda in input: cf, intero da 1 a 6 (bimestre) e deve restituire per tutte le patologie, la differenza tra esordi che hanno coinvolto il cf* nel bimestre* rispetto al bimestre dell'anno precedente
- Per ogni visita trovare il tempo che è passato dall'ultima volta che quel medico ti ha visitato

```sql
SELECT V.*, DATEDIFF(V.Data, LAG(V.Data, 1) OVER (
                                                   PARTITION BY V.Paziente, V.Medico
                                                   ORDER BY V.Data
                                                  )
                    )
FROM Visita V
```

VAGLINI

SCHEMA LOGICO:

- *persona* (*id_persona*, *nome*, *cognome*)
- *film* (*id_film*, *id_regista*, *titolo*, *genere*, *anno*): dove *id_regista* e' una chiave esterna che fa riferimento a *persona*;
- *partecipazione* (*id_attore*,*id_film*, *ruolo*): dove *id_attore* ed *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *persona* e *film*;
- *cinema* (*id_cinema*,*nome*,*indirizzo*)
- *proiezione* (*id_cinema*,*id_film*,*giorno*): dove *id_cinema* e *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *cinema* e *film*.

Algebra Relazionale:

"Elencare gli attori che hanno interpretato film proiettati al cinema Zenith dopo il 2002"
"Attori che non hanno mai interpretato film drammatici"
"Film interpretati da X che non sono più proiettati dal 2010"
"Nome e cognome delle persone che sono state sia attore che regista dello stesso film"
"I film che non sono stati mai proiettati al cinema zenit"
"Attori che hanno partecipato a Film diretti da X ma non da Y"
"Lista dei film che non sono mai stati proiettati nel 2020"
"Lista di film che sono stati prodotti nel 2010, ma mai proiettati"

Calcolo Relazionale:

Cosa fa questo calcolo sui domini?, La differenza fra le due?
{A:a, B:b | not exists e,c . R(A:a, B:b, C:c) and S(D:c, E:e) }
{A:a, B:b | R(A:a, B:b, C:c) and not exists e . S(D:c, E:e) }

Effettualo sul calcolo sui domini
"Attori che hanno partecipato a Film diretti da X ma non da Y"

Effettualo sul calcolo delle tuple
"Lista dei film che non sono mai stati proiettati nel 2020"

Effettualo sul calcolo delle tuple
"Lista di film che sono stati prodotti nel 2010, ma mai proiettati"

Dipendenze Funzionali:

"Minimizzare la seguente sequenza ⇒ A→ BC, CD→ E, B-> D, E→ A"
A → B, A → C, CD → E, B → D, E → A

C+ = {C}
D+ = {D} Quindi CD → E non cambia
Risposta Esatta: A → B, A → C, CD → E, B → D, E → A (?)

"Questo insieme di dipendenze è minimale,si può ridurre? => A→ BC, CD→ E, B-> D, A→ D
"E' in 3NF? Il risultato dell'esercizio, e in BCNF, se non è in 3NF allora portacela"
A → B, A → C, CD → E, B → D, A → D

"Minimizzare la seguente sequenza ⇒ A→ BCD, CD→ E, B-> D
E' in 3NF? In BCNF?"

"Minimizzare la seguente sequenza ⇒ A → B, C → AD, AF → EC"
Trova anche le chiavi

"Minimizzare la seguente sequenza =>
E → N, N→ D, EN → LCD, C → S, D → M, M → D, ED → A, NLC → A
E' in 3NF o BCNF?"

E → N, N → D, EN → L, EN → C, EN → D, C → S, D → M, M → D, ED → A, NLC → A

Affidabilità dai guasti:
"Ripresa a caldo di questo log":
DUMP, B(T1), B(T2), B(T3), I(T1,O1,A1), D(T2,O2,B2), B(T4), U(T4,O3,B3,A3), U(T1,O4,B4,A4), C(T2), CK(T1,T3,T4), B(T5), B(T6), U(T5, O5, B5,A5), A(T3), CK(T1,T4,T5,T6), B(T7), A(T4), U(T7,O6,B6,A6), U(T6,O3,B7,A7), B(T8), A(T7)

"Ripresa a freddo derivata da un problema hardware"
DUMP, B(T1), B(T2), B(T3), I(T1,O1,A1), D(T2,O2,B2), B(T4), U(T4,O3,B3,A3), U(T1,O4,B4,A4), C(T2), CK(T1,T3,T4), B(T5), B(T6), U(T5, O5, B5,A5), A(T3), CK(T1,T4,T5,T6), B(T7), A(T4), U(T7,O6,B6,A6), U(T6,O3,B7,A7), B(T8), A(T7)

"Gestisci il problema SW"
DUMP, B(T1), B(T2), B(T3), I(T1, O1, A1), D(T2, O2, B2), B(T4), U(T4, O3, B3, A3), U(T1, O4, B4, A4), C(T2), CK(T1, T3, T4), B(T5), B(T6), U(T5, O5, B5, A5), A(T3), CK(T1, T4, T5, T6), B(T7), A(T4), U(T7, O6, B6, A6), U(T7, O3, B7, A7), B(T8), C(T7), I(T8, O2, A8), D(T8, O2, B8)

Problema della concorrenza:
"Applicare il 2PL Stretto"
r1(x), r1(t), r3(z), r4(z), w2(z), r4(x), r3(x), w4(x), w4(y), w3(y), w1(y), w2(t)

r1(x)
r1(t)
r3(z)
r4(z)
w2(z) -- la 2 aspetta il commit della 3 e della 4
r4(x)

r3(x)
w4(x) -- la 4 aspetta il commit della 1 e 3
w3(y) -- COMMIT
w1(y) -- COMMIT
w4(y) -- COMMIT
w2(t) -- COMMIT

r1(x), r1(t), r3(z), r4(z), r4(x), r3(x), w3(y), w1(y), w4(x), w4(y), w2(z), w2(t)

"Applicare il 2PL Stretto"
r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), c3, w1(y), c1, r2(x), c2

"Applica il TS"
r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)

"Applica il TS e il 2PL Stretto"
r1(A), r2(B), r3(A), r2(A), c2, w1(A), c1 , w3(A) c3

"Controllare che sia CSR, se si, controllalo anche con il 2PL"
r1(x), w2(x), r3(x), r1(y), r4(z), w2(y), r1(v), w3(v), r4(v), w4(y), w5(y), w5(z)

Teoria:

- Come faccio a vedere se uno schedule è view-serializzabile?
  - Deve essere view-equivalente ad uno schedule seriale, bisogna controllare che le relazioni leggi da sono uguali e l'insieme delle letture finali sia sempre lo stesso. Possiamo facilitarci la vita controllando che sia CSR oppure conforme al 2PL o TS, e quindi sarà sicuramente VSR ma non il contrario.
- Cosa si intende quando una transazione esegue un'operazione di abort? Cosa bisogna fare?
- Differenza tra Equi-Join e Natural-Join
  - Il Natural Join effettua il join su istanze con attributi con lo stesso nome, e questo ha meno colonne, ma ha bisogno di ridenominazione.
  - L'Equi Join non ha bisogno di ridenominazione, in quanto siamo noi a scegliere gli attributi su cui fare join
  - L'Outer Join invece serve a mantenere tuple che non fanno join
- Come si arriva a definire due protocolli come 2PL oppure TS?
  - Perché verificando tramite il VSR si usa un algoritmo di complessità esponenziale, quindi molto problematico per il database, mentre 2PL e TS hanno complessità polinomiale.
- Come avvengono le scritture fra il log e la memoria secondaria per garantire la consistenza in caso di guasto?
  - Viene scritto prima nel Log rispetto alla memoria secondaria perché il log si trova in memoria "stabile" e quindi è molto più sicuro, se avvenisse un problema HW la memoria secondaria potrebbe risentirne
- Come funziona il protocollo TS
  - Assegna un identificativo ad ogni transazione, più alto se questa è arrivata dopo. Spiegazione del come funziona …
- Come funziona il 2PL
- Come lavora il gestore dell'ottimizzazione? Rispetto a cosa fa le ottimizzazioni?
  - Il Query Processor, trasforma la query in un'espressione algebrica, facendo alcune operazioni per ottimizzarla

- Il Gestore del Buffer utilizza alcune primitive, le spieghi.
    - Fix → Richiesta di una pagina
    - SetDirty → Comunica che è stata modificata una pagina, mettendo il flag a 1
    - UnFix →
- Differenza tra calcolo sui domini e sulle tuple
    - La differenza sta nella dichiaratività, da una parte hai i domini separati e nell'altro le tuple separate, l'SQL si basa sul calcolo delle tuple, il primo è dipendente dal dominio e da origine anche a espressioni senza senso.
- Quando c'è un problema HW come funziona il gestore dell'affidabilità?
- Differenza tra 2PL e 2PL stretto
    - Nel 2PL intanto esistono due fasi, una fase crescente in cui si prendono i lock e una in discesa in cui si rilasciano i lock, quando comincia a rilasciare può solo rilasciare.
    - Nel 2PL Stretto si previene le letture sporche in quanto rilasciamo i lock solamente al momento del commit.
- Come viene gestito il file di log?
    - Viene conservato in memoria stabile, e il gestore dell'affidabilità inserisce i vari ck e i dump, assieme a tutte le operazioni che vengono fatte all'interno della base di dato. Di solito si scrive prima sul log, per essere sicuri che quello che viene effettivamente modificato in memoria secondaria sia comunque salvato sul log. (Scrivere bene il momento in cui si scrive l'after state e il before state all'interno del log per ogni operazione)