Oggetto: Analisi dettagliata Orale Pistolesi-Vaglini

Data: 22/06/2021
Versione: V 1.0
Autori: Lorenzo Valtriani - l.valtriani2@studenti.unipi.it
Crediti: CRISTINA MARIA RITA LOMBARDO - c.lombardo17@studenti.unipi.it
SARA ALIANI - s.aliani@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
```

“Considerate le terapie dei pazienti aventi reddito inferiore al reddito medio della loro città, scrivere una query che, per ciascun paziente, restituisca il codice fiscale, il cognome, e la durata media delle terapie (oggi terminate) in cui, se avesse assunto il farmaco più economico basato sullo stesso principio attivo del farmaco effettivamente assunto, avrebbe ottenuto un risparmio sul costo totale della terapia superiore al 50%, e a quanto sarebbe ammontato tale risparmio.”

“Implementare una stored procedure che, ricevute in ingresso due date d1 e d2 che delimitano il lasso di tempo L = [d1, d2], associ i giorni di L ai relativi mesi e calcoli, per ogni mese (limitatamente ai giorni in L), il numero di giorni senza terapie di ogni paziente p, cioè quei giorni in cui il paziente p non ha assunto farmaci (ad eccezione delle eventuali terapie a vita). La stored procedure deve stampare a video una classifica dei pazienti in cui un paziente p è in posizione tanto più alta quanto più è bassa la media dei giorni senza terapie, calcolata considerando i giorni senza terapie dei vari mesi, limitatamente ai giorni in L.”

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.

```sql
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?
- Cosa è la divisione insiemistica? Quali sono i due metodi per implementarla in SQL?
    - La parola chiave che dobbiamo associare all'operatore di divisione è "TUTTI"
    - I metodi con cui la implementiamo sono i seguenti: "Il doppio not exists" e "il group by con having".
- Scrivere una Stored Function che restituisca TRUE o 1 se il medico passato come input ha visitato solo pazienti di Roma.
- Cosa è l'incremental refresh?
- L'SQL vivrebbe senza il Group By

```
10
11  select (select avg(m1.matricola)
12          from medico m1
13          where m1.specializzazione=m.specializzazione),m.specializzazione
14  from medico m
```

    - Serve però il DISTINCT nel select esterno.

- Si può scrivere la query, che conta per ogni città i medici che non hanno visitato alcun medico senza usare l'outer join?

```
2   SELECT M.Citta, COUNT(*)
3   FROM Medico M
4   WHERE M.Specializzazione = 'Cardiologia'
5         AND NOT EXISTS
6           (
7            SELECT *
8            FROM Visita V
9            WHERE V.Medico = M.Matricola
10          )
11  GROUP BY M.Cittaj
```

- 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

```
3   SELECT DISTINCT M.Matricola
4   FROM Medico M
5        LEFT OUTER JOIN
6        (
7           SELECT DISTINCT M.Medico
8           FROM Medico M
9                INNER JOIN
10               Visita V ON V.Medico = M.Matricola
11          -- WHERE condizione sulla data
12       ) AS D ON D.Medico = M.Matricola
13  WHERE D.Medico IS NULL
14
```

- 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.

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

- Stored Procedure che classifica i medici di ogni specializzazione in base al totale di pazienti visitati. Più sono i medici, più è alta la posizione in classifica.
- Considerato un anno e un medico, si determinino i medi in cui il medico ha superato, con gli incassi delle visite, i suoi incassi realizzati nel corrispondente mese dell'anno precedente.
- La clausola EXISTS può essere sostituita con (?)
- Medici che hanno visitato solo pazienti di roma (fatta con una subquery non correlata)

```
-- Medici che hanno visitato solo pazienti di Roma

SELECT  *
FROM    medico M
WHERE   M.Matricola NOT IN
            (
            SELECT  V.Medico
            FROM    visita V INNER JOIN paziente P ON V.Paziente = P.CodFiscale
            WHERE   P.Citta <> 'Roma'
            )
```

- Medici che nel mese attuale hanno visitato un paziente che non hanno mai visitato prima con (subquery non correlata)
    - Non si può fare perché ogni riga nella tabella interna deve essere necessariamente in relazione con quella interna e quindi non si può fare con una subquery non correlata.
- Stored procedure (input mese, anno) dei cardiologi, in quale data (della visita), l'incasso cumulativo mensile ha superato l'incasso medio
- Visite dei cardiologi, in quale data è stato superato l'incasso medio mensile della cardiologia degli ultimi 5 anni
- Spiega i due modi con cui implementare la divisione insiemistica. Esempio di query con divisione
- Una stored function che restituisce true se un medico ha visitato solamente pazienti di Roma
- Quando scrivo per esempio una query con raggruppamento, qual è l'esigenza maggiore che mi porta a inserire il predicato group by? Ovvero, se MySQL non avesse il predicato group by, sarebbe comunque ugualmente espressivo? Ovvero, il group by può essere facilmente sostituito nelle query da altri costrutti?
- Cosa fa la funzione lead?
- Se ho una query dove utilizzo exists con query correlata, posso sostituire l'exists con una subquery non correlata? E se sì, posso farlo sempre, mai, o talvolta?
- Si possono effettuare le aggregazioni senza group by?
- In quali casi il distinct è necessario all'interno di una query con raggruppamento? Invece se non uso il raggruppamento, e devo fare un join tra le tabelle T1 e T2. In questo caso, quand'è che si generano duplicati? Sempre, oppure le due tabelle devono avere delle caratteristiche?
- Vogliamo che tutte le terapie che riguardano l'aspirina non si protraggono per più giorni rispetto a quelli per cui è indicata nelle indicazioni. Come faresti a implementare questo vincolo?

- Stored procedure che prende in ingresso un mese e un anno e classifica tutti i medici all'interno delle rispettive specializzazioni, in base al numero di prime visite effettuate
- Capisci cosa fa questa query e scrivi la versione join-equivalente

```
SELECT COUNT(*)
FROM Visita V
WHERE NOT EXISTS
    (SELECT *
    FROM Visita V2
    WHERE V2.Data < V.Data
                AND V2.Medico = V.Medico
                AND EXISTS ( SELECT *
                             FROM Visita V3
                             WHERE V3.Data < V2.Data
                                         AND V3.Medico <> V2.Medico
                           )
    );
```

- Cos'è l'SQL dinamico? Puoi farmi un esempio di SQL dinamico "camuffato" che abbiamo studiato?
    - No, perchè non ne abbiamo visti
- A cosa serve il self-join?
- Gli outer join possono essere sostituiti in qualche modo?
- Vogliamo classificare i medici della clinica sulla base della lunghezza media dei periodi di inattività tra una visita e la successiva. E questo, anno per anno. Per farlo scriviamo una stored procedure
- Che cos'è la differenza insiemistica e come si implementa sia con la subquery sia col join
- Per ogni paziente vogliamo sapere la data in cui è stato visitato per l'ultima volta da Verdi. Aggiungi anche, per ogni visita, i giorni passati dall'ultima visita effettuata con lo stesso Verdi
- Quand'è che si utilizzano i modificatori any e all e con quali costrutti dell'SQL si possono sostituire?
- Cos'è la divisione insiemistica e quali sono i due modi per poterla implementare usando un esempio
- Per ogni medico di Pisa trovare i pazienti visitati nel Maggio 2021
- Spiega cosa fa la funzione rank
- Che fanno il group by e il partitioning? E quali sono i casi in cui si usa il group by, quali i casi in cui si usa il partitioning e quali i casi in cui si usano entrambi?
- Dati i medici, per ciascuna specializzazione individuare i medici che hanno fatto visite prima del 12/05/2014 restituendo la classifica delle specializzazioni (il rank è migliore per la specializzazione nella quale più medici hanno effettuato visite)
- Cos'è la log table
- Quand'è che in una stored procedure non posso assolutamente fare a meno di un cursore?
- Vogliamo realizzare una materialized view che, per ogni specializzazione della clinica, memorizzi il numero di visite effettuate e la matricola del medico che ha

effettuato la visita più recente. voglio mantenere aggiornata la materialized view con un incremental refresh. Scegliamo la modalità on demand
- Parla della divisione insiemistica, cos'è, cosa fa, quali sono i modi per risolverla
- Pazienti di pisa che sono stati visitati da tutti gli otorini in almeno uno degli ultimi 5 anni
- Scrivi una query che per ogni visita otorinolaringoiatrica proietti paziente, medico, data e il numero di otorinolaringoiatri che hanno visitato quel paziente precedentemente
- Posso risolvere una query dove normalmente utilizzerei il raggruppamento con un partitioning (quindi senza raggruppamento) oppure no?
    - Scrivi un esempio di query con raggruppamento ma invece del raggruppamento la risolvi col partition by
    - Se ci fosse stato un predicato where avresti comunque potuto farlo? [*Risposta: Si*]
    - E invece il contrario è possibile (sostituire il partition by col raggruppamento)?
- Se dovessi generare una materialized view che salvasse i dati che ti servono nell'esercizio iniziale(l'esercizio dell'orale del terzo appello -> giorni senza terapia per ogni paziente in ogni mese), come la strutturesti e che politica di aggiornamento useresti?
- Per ogni terapia calcola i giorni nei quali questa terapia è sovrapposta a un' altra dello stesso paziente
- Spiega come lavora la funzione last value
    - Per ogni paziente restituire codice fiscale e la data dell'ultima terapia iniziata
- La divisione insiemistica, cos'è, cosa fa, un esempio.
    - Trova i pazienti visitati da tutti i medici di almeno due specializzazioni diverse. (da svolgere usando entrambi i metodi della divisione insiemistica)
    - Crea una log table che contenga i pazienti che rispettano la condizione dell'esercizio appena fatto e che venga aggiornata con un refresh on demand
- Cosa fa la funzione lag?
    - Calcola il numero medio di giorni che sono intercorsi tra una visita e quella precedente che un paziente ha fatto con un medico della stessa specializzazione
- Nome e cognome dei medici che hanno visitato un solo paziente nella loro stessa città non più di due volte ciascuno
- Spiega come funziona la funzione rank
    - Per ogni paziente classifica le sue terapie terminate sulla base della loro durata in giorni. Una terapia è in posizione tanto più alta in classifica quanto più è durata

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"
"Titoli dei film mai proiettati di domenica al cinema paradiso"
"Nome e cognome dei registi che non hanno mai diretto Marcello Mastroianni"
"I cinema che hanno proiettato tutti i film"
"Titoli film che non sono stati proiettati in alcun cinema nel 2020"
"Attori che hanno interpretato TUTTI i film diretti da Fellini"
Prima prendiamo i film diretti da Fellini. Poi effettuiamo la divisione tra gli attori e i film diretti da fellini.
"Nomi dei registi che non hanno mai diretto un film nel corso del 2020"
"Cinema che hanno proiettato tutti i film di Fellini"
"Quali sono i film interpretati da attori che non sono mai stati diretti da Fellini"
"Quali sono i cinema che non funzionano il giovedì"
"Nome e Cognome dei registi che non hanno mai diretto film comici"
"Quali sono i registi che hanno diretto tutti i film proiettati al cinema Paradiso"
"Gli attori che sono stati proiettati in tutti i cinema (eventualmente anche in film diversi)"

Traduci in algebra relazionale

{A:a, B:b | not exists e,c . R(A:a, B:b, C:c) and S(D:c, E:e)}

- soluzione corretta: $\pi_{A,\,B}(R\ -\ (R \bowtie_{C\,=\,D} S))$

{A:a, B:b | R(A:a, B:b, C:c) and not exists e . S(D:c, E:e)}

- soluzione corretta: $\pi_{A,\,B}(\sigma_{E\ IS\ NULL}(R \bowtie_{C\,=\,D} S))$

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"

"Nomi dei registi che non hanno mai diretto un film nel corso del 2020"

"Quali sono i cinema che non funzionano il giovedì"

"Nome e Cognome dei registi che non hanno mai diretto film comici"

"Titoli film che non sono stati proiettati in alcun cinema nel 2020"

"Nome e cognome dei registi che non hanno mai diretto Marcello Mastroianni"

Effettualo sul calcolo delle tuple

"Lista dei film che non sono mai stati proiettati nel 2020"

"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 sequenza ⇒ AB→C, B→A, AD→E, BD→F

Trova la chiave e controlla se è in 3FN"

"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

"Minimizzare la sequenza AB→C, B→A, AD→E, BD→F, BD→E e portarla in 3FN"
Si tolgono gli attributi estranei quindi AB →C diventa B →C
Si tolgono le df ridondanti, quindi DB → E si toglie
Non è né in 3FN né in BCNF quindi suddividiamo in:
R1(<u>B</u>CA), R2(<u>AD</u>E), R3(<u>BD</u>F)

"A→ BC, CD→ E, B-> D, E→ A
R(A,B,C,D,E) decomposta in R1(A,B,C) and R2(A,D,E)
c'è perdita di informazione?"
Mantengo il Join perché la chiave si trova in almeno una delle due relazioni ma le dipendenze non le manteniamo perché CD → E non è contenuta totalmente in una delle due relazioni.

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

"Se C→ B, A→ D posso dedurre che AC→ B?"

"Se ho AB→D, B→C posso derivare AC→D?"

"Se A→B, AB→D posso dedurre che A→D?"

"Considera le dipendenze A→ BC, CD→ E, B-> D, E→ A. Qual è la chiave? Le decomposizioni R1(A,B,C) , R2(A,D,E) mantengono il join?"

"Minimizzare la sequenza: AB→C, AD→EF, B→A, A→D, B→E"

"Questa tabella di dipendenza definisce una forma normale? A→ BC, CD→ E, B-> D, E→ A"

"Considera la relazione R(ABCDEF), sulla quale sono definite le seguenti dipendenze funzionali: AB→C, AD→EF, B→A, A→D, B→E.

- ➢ L'insieme delle dipendenze è minimale?
  soluzione:
  - ➔ partiamo lasciando un unico attributo a dx:

    AB→C, AD→E, AD→F, B→A, A→D, B→E

  - ➔ verifichiamo nelle dipendenze con più attributi a sx se alcuni attributi possono essere tolti:
    osserviamo che: B→A e quindi A è superfluo in AB→C
    A→D e quindi D è superfluo in AD→E e in AD→F
    quindi otteniamo:

    B→C, A→E, A→F, B→A, A→D, B→E

- ➔ verifichiamo se ci sono dipendenze funzionali ridondanti:
  osserviamo che: B→A e A→E, quindi per transitività B→E (cioè B→E è ridondante)
  quindi otteniamo che l'insieme minimale è:

  B→C, A→E, A→F, B→A, A→D

➢ La relazione è in forma normale?
soluzione:

- ➔ individuiamo la chiave:
  i candidati chiave solo gli attributi che si trovano solo a sx, quindi solo B.

  La chiusura di B è:
  B+ = B
  B+ = B, C (ricavato da: B→C)
  B+ = B, C, A (ricavato da: B→A)
  B+ = B, C, A, E (ricavato da: A→E)
  B+ = B, C, A, E, F (ricavato da: A→F)
  B+ = B, C, A, E, F, D (ricavato da: A→D)

  Quindi B è chiave

- ➔ Non è in forma normale BC (infatti, ad esempio, in A→E, l'attributo A non è una superchiave)
- ➔ Non è in 3FN (infatti, ad esempio, in A→E, l'attributo A non è una superchiave e a dx compare l'attributo E che non fa parte della chiave)+

➢ Per normalizzarla cosa bisogna fare?"
soluzione:

- ➔ consideriamo l'insieme di df minimale (B→C, A→E, A→F, B→A, A→D)
- ➔ decomponiamo in maniera tale da conservare le dipendenze:
  R1(A, E, F, D) con chiave A
  R2(A, B, C) con chiave B
- ➔ esiste già una relazione che contiene la chiave (essendo la chiave formata da un unico attributo), quindi si conserva anche il join

"Data la relazione R(ABCDE), su cui è definito l'insieme di dipendenze:
A→BC, CD→E, B→D, E→A

- ➢ Decomponendo R in R1(A,B,C) e R2(A,D,E), si conserva il join?
- ➢ Si conservano le dipendenze?
- ➢ Come devono essere decomposte affinché vengano conservati entrambi?"

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)

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)

"Queste operazioni vengono effettuate su memoria principale o sfruttano il buffer?"

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), guasto

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)

"Controllare se è conflict-serializzabile"
r1(x), r1(t), r3(z), r4(z), w2(z), r4(x), r3(x), w4(x), w4(y), w3(y), w1(y), w2(t)

"Applica il TS"
r1(x), r4(x), w4(x), r1(y), r4(z), w4(z), w3(y), w3(z), w1(t), w2(z), w2(t)

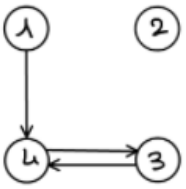
"Classifica il seguente schedule"

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

soluzione:

- ➔ Non è uno schedule seriale (infatti, ad esempio, le operazioni della transazione 1 sono inframezzate da quelle delle transazioni 2, 3 e 4).
- ➔ Non è CS, infatti:

x	y	z	t
r1	w4	r3	r1
r4	w3	r4	w2
r3	w1	w2	
w4			



senza necessità di portare a termine il disegno del grafo dei conflitti, ci rendiamo conto che le operazioni evidenziate fanno sì che si crei un ciclo nel grafo e, quindi, che lo schedule non sia CS.

- ➔ Non è VS, infatti:
  - la transazione 1 legge x da database, quindi deve essere eseguita prima della 4 (visto che questa effettua una scrittura su x).
  - allo stesso tempo, però, la transazione 1 effettua l'ultima scrittura su y, quindi dovrà essere eseguita dopo la 4 (visto che anche questa effettua una scrittura su y).

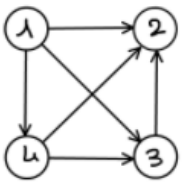
  Queste due necessità sono incompatibili, dunque lo schedule non è VS.

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

soluzione:

- ➔ Non è uno schedule seriale (infatti, ad esempio, le operazioni della transazione 1 sono inframezzate da quelle delle transazioni 3 e 4).
- ➔ È CS (e di conseguenza anche VS), infatti:

x	y	z	t
r1	r1	r4	w1
r4	w3	w4	w2
w4		w3	
		w2	



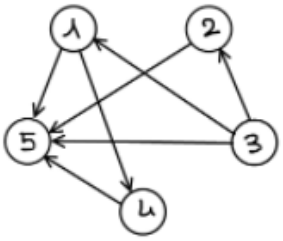
Il grafo dei conflitti è aciclico, dunque lo schedule è CS.

3) r1(x), r3(y), w1(y), w4(x), w1(t), w5(x), r2(z), r3(z), w2(z), w5(z), r4(t), r5(t)

soluzione:

- ➔ Non è uno schedule seriale (infatti, ad esempio, le operazioni della transazione 1 sono inframezzate da quelle delle transazioni 3 e 4).
- ➔ È CS (e di conseguenza anche VS), infatti:

x	y	z	t
r1	r3	r2	w1
w4	w1	r3	r4
w5		w2	r5
		w5	



Il grafo dei conflitti è aciclico, dunque lo schedule è CS.

“Qual è un possibile schedule seriale equivalente?”

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

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. Nel calcolo sulle tuple non si può eseguire l'unione (ogni tupla del risultato può provenire da una sola relazione).
- 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)
- Cos'è questo? Chi lo scrive? Dove?
  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)ù
- Quali sono le regole di Armstrong?
- Come si sincronizza la scrittura nel Log e la rispettiva scrittura in memoria secondaria?
    - Ovviamente dobbiamo scrivere le operazioni prima sul Log, però possiamo scegliere se scrivere in secondaria solo dopo aver fatto il commit oppure prima.
      Noi invece non facciamo caso se scriverle prima o dopo del commit della transazione.. basta che si scrivano sempre prima nel log.
- Differenza tra 2PL e TS
    - Il 2PL ha due problemi, la lettura sporca e il deadlock. Mentre il primo si gestisce tramite il 2PL stretto, il secondo lo si gestisce bene con il TS che è molto più restrittiva.
- Come si gestisce il deadlock?
    - Tramite un tempo (Timeout) in cui una transazione può essere in attesa, così da abortire se il tempo viene superato.
- Se una relazione ha solo una chiave perché tramite una decomposizione se è in 3FN allora è anche in BCNF?
- Quando il gestore inserisce le transazioni nel checkpoint cosa succede?
    - Quando si inseriscono le transazioni all'interno del commit le transazioni non possono fare né commit né abort ma le altre operazioni possono essere svolte.
    - Registrare nel ck quali transazioni sono attive
- Gli indici cosa sono e cosa servono
    - Indica come vengono rappresentate le tuple all'interno delle pagine:
        - Indici sequenziali: le tuple vengono messe una dopo l'altra, possono essere disordinate se sono ordinate in ordine cronologico, altrimenti se hanno un criterio di ordinamento sono ordinati.
        - Hash: utile per trovare immediatamente tramite il suo valore il valore stesso all'interno della pagina tramite una funzione hash
        - Albero:
    - Nell'indice primario l'ordine delle tuple è necessario e può non essere denso.
    - Un indice secondario invece l'ordine non è obbligatorio ma deve essere necessariamente denso.
- Che differenza c'è tra i protocolli 2PL e time stamp rispetto all'equivalenza negli schedule seriali?
- Cos'è il buffer e cosa fa? Com'è organizzato?
- Come viene gestito il buffer nel contesto del recovery?
  (a cosa serve? Cosa fa? Quando si verifica un guasto cosa accade al buffer?)
- Quando una transazione fa commit, i dati modificati da quella transazione vengono subito riportati in memoria secondaria?
- Quali sono i possibili algoritmi per implementare il join a livello fisico? Descrivimi come funzionano
  (ulteriori domande inerenti fatte:
    - ➢ ci sono problemi legati alla memoria aggiuntiva?

        - ➢ avendo due tabelle (una di n elementi e una di m elementi), quanti confronti saranno effettuati al massimo?
            - n x m confronti
        - ➢ da cos'è determinato il costo dell'operazione di join?
            - Dal numero di accessi in memoria secondaria
        - ➢ nel nested loop quale tabella viene scelta come tabella interna?)
            - Quella di dimensione inferiore, così da ridurre gli accessi in memoria secondaria
- Una volta definite delle relazioni, si controlla se queste sono in una qualche forma normale. Cosa vuol dire? Cosa si cerca di diminuire quando si normalizzano le relazioni?
- Che differenza c'è tra le varie forme normali? Dal punto di vista delle ridondanze c'è differenza?
- Se si utilizza il protocollo TS per gestire le transazioni concorrenti, che tipo di schedule si ottiene alla fine? Tale schedule può essere equivalente ad un qualsiasi schedule o corrisponde ad uno schedule specifico?
    - Lo schedule che si ottiene è CS e lo schedule seriale corrispondente è univocamente determinato dalle etichette temporali delle transazioni: infatti, anche se si decide di far abortire una transazione, questa verrà fatta ripartire con un nuovo timestamp e alla fine si otterrà comunque uno schedule basato sui timestamp attribuiti alle transazioni al momento del begin