Basi di Dati e Sistemi Informativi I, Prove scritte AA 2003/04

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Scritto del 1 marzo 2010

Si consideri il seguente schema relazionale che descrive un database per la memorizzazione di partite in un gioco per scacchiera.
$PEZZO(\underline{CodiceP}, Tipo, Colore, iX, iY)$
$MOSSA(\underline{Partita, Ordine}, CodiceP, daX, daY, aX, aY)$
$ELIMINA(Partita, Ordine, CodiceP)$

$PEZZO$ formisce la descrizione dei pezzi del gioco (iX e iY indicano le coordinate della posizione iniziale del pezzo sulla scacchiera).
$MOSSA$ memorizza le mosse per ogni partita: Ordine è un intero che indica l'ordine della mossa nella partita; daX e daY sono le coordinate della casella di partaenza e aX, aY sono le coordinate della casella di arrivo. $ELIMINA$ indica i pezzi eliminati durante la partita: ordine indica la mossa che ha portato alla eliminazione.

Esercizio 11 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutte le partite in cui sono stati eliminati tutti i pezzi neri.*

Esercizio 12 *Si scriva una vista SQL che risponda al seguente schema logico:* $VISTA(Partita, Orizzontali, V$ *Orizzontali e Verticali indicano il numero delle mosse orizzontali e verticali della partita. Nella vista devono comparire solo le partite che hanno un numero di mosse superiore o uguale alla media delle mosse di tutte le partite.*

Esercizio 13 *Si scriva una procedura PLSQL che riceve in ingresso il codeice di una partita e l'ordine di una mossa. La procedura deve restituire una stringa di terne che descrivono lo stato della scacchiera nella partita al compimento della mossa di ordine fornito. Le terne hanno la forma (X,Y,occ) dove X e Y sono le coordinate della casella e occ vale NULL se la casella è vuota e contiene il codice di un pezzo se, invece, la casella è occupata da un pezzo. (Si osservi che, una casella X,Y contiene un pezzo alla fine della mossa i se il pezzo è posto in quella casella da una mossa j con* $j < i$ *e nessuna altra mossa k con* $j < k < i$ *ha spostato il pezzo o ha eliminato il pezzo mettendoci un altro pezzo al suo posto).*

Esercizio 14 *Si consideri lo schema relazionale*
$R(A, B, C, D)$ *La relazione gode delle seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $A, B \rightarrow D$;

– $C \rightarrow A$;
– $D \rightarrow B$.

Dire quali sono le chiavi. Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

Soluzione Esercizio 11
Siano PE, MO, EL, le relazioni per PEZZO, MOSSA, ELIMINA rispettivamente.
Tutti i pezzi neri eliminati in tutte le partite.

$$PP \leftarrow \Pi_{Partita}(MO) \times \sigma_{Colore='Nero'}(PE)$$

Partite in cui non sono stati eliminati tutti i pezzi neri.

$$NO \leftarrow \Pi_{Partita}(PP \setminus \Pi_{Partita,CodiceP}(EL))$$

Soluzione

$$\Pi_{Partita}(MO) \setminus NO$$

†
Soluzione Esercizio 11
La vista viene costruita a partire da viste parziali

```
CREATE VIEW Orizz(Partita,Orizzontali) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
WHERE M.daY=M.aY
GROUP BY Partita

CREATE VIEW Vertic(Partita,Verticali) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
WHERE M.daX=M.aX
GROUP BY Partita

CREATE VIEW PartiteC(Partita,NMosse) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
GROUP BY Partita
```

Soluzione

```
SELECT Partita, Orizzontali, Verticali
FROM   (Orizz NATURAL JOIN Vertic) NATURAL JOIN PartiteC
WHERE NMosse ≥ (SELECT AVG(NMosse)
                FROM PartiteC)
```

†

Soluzione Esercizio 13

Della Funzione viene dato solo uno schema di massima trascurando i dettagli.

Intestazione

CREATE FUNCTION Stato (CPartita : Char(8), NMossa Integer) RETURN VARCHAR2(1000)

La parte rilevante del corpo della procedura verifica la presenza di un pezzo sulla casella. La select viene espressa come unione di due SELECT. La prima verifica che il pezzo nella posizione iniziale che non sia stato spostato o eliminato. La seconda verifica la presenza di pezzi messi da una possa in una casella che non siano stati mossi successivamente o eliminati.

```
FORI IN 1 .. MaxX LOOP
    FORJ IN 1 .. MaxY LOOP
        Trovato := NULL
        SELECTP.CodiceP INTO Trovato
        FROM   PEZZO P
        WHEREP.iX = I AND P.iY = J AND
                NOT EXISTS(SELECT *
                        FROM MOSSSA M
                        WHEREM.Ordine ≤ :NMossa AND
                                M.Partita = :CPartita) AND
                P.CodiceP NOT IN
                        (SELECT E.CodiceP
                        FROM ELIMINA E
                        WHEREE.Ordine ≤ :NMossa AND
                                E.Partita = :CPartita)

        IF trovato IS NULL THEN
        (SELECTM.CodiceP
        FROM   MOSSA M
        WHEREM.partita = :CPartita AND M.aX=I AND M.aY=J AND
                NOT EXISTS(SELECT *
                        FROM MOSSSA S
                        WHERES.Ordine BETWEEN M.Ordine AND :NMossa
                                AND M.Partita = :CPartita
                                AND S.daX = M.aX AND S.daY = M.aY AND
                                M.CodiceP = S.CodiceP ) AND
                M.CodiceP NOT IN
                        (SELECT E.CodiceP
                        FROM ELIMINA E
                        WHEREE.Ordine ≤ :NMossa AND
                                E.Partita = :CPartita))
        IF trovato IS NULL THEN risultato := risultato + '('+I + ',' +J ', NULL)'
        ELSE risultato := risultato +'('+I +',' + J ',' + trovato + ')'
        END IF
        ELSE risultato := risultato + '('+I + ',' + J ','+ trovato + ')'
        END IF
```

Essendo interessati solo alle caselle occupate da un pezzo si usa alternativamente un cursore scritto come segue. Il corpo della procedura semplicemente scandisce il cursore formando la stringa di uscita.

```
CURSOR attuali IS (SELECTP.iX,P.iY,P.CodiceP
                   FROM   PEZZO P
                   WHERENOT EXISTS(SELECT *
                                   FROM MOSSSA M
                                   WHEREM.Ordine ≤ :NMossa AND
                                         M.Partita = :CPartita) AND
                         P.CodiceP NOT IN
                                   (SELECT E.CodiceP
                                   FROM ELIMINA E
                                   WHEREE.Ordine ≤ :NMossa AND
                                         E.Partita = :CPartita)

                   UNION
                   (SELECTM.aX,M.aY,M.CodiceP
                   FROM   MOSSA M
                   WHEREM.partita = :CPartita AND
                         NOT EXISTS(SELECT *
                                   FROM MOSSSA S
                                   WHERES.Ordine BETWEEN M.Ordine AND :NMossa
                                         AND M.Partita = :CPartita
                                         AND S.daX = M.aX AND S.daY = M.aY AND
                                         M.CodiceP = S.CodiceP ) AND
                         M.CodiceP NOT IN
                                   (SELECT E.CodiceP
                                   FROM ELIMINA E
                                   WHEREE.Ordine ≤ :NMossa AND
                                         E.Partita = :CPartita))
```

†

Soluzione Esercizio 14

Le chiavi sono date dai seguenti insiemi $\{A, B\}$, $\{C, D\}$, $\{A, D\}$, $\{B, C\}$.
Tutti gli attibutti sono primi e dunque la relazione rispetta la terza forma normale.
Le dipendenze $C \to A$ e $D \to B$ non rispettano la forma normale di Boyce-Codd.
Scomposizione partendo dalla dipendenza $C \to A$.
$R_1 = \{A, C\}$ con la sola dipendenza $C \to A$.
$R_2 = \{C, B, D\}$ con le seguenti dipendenze $D \to B$, $B, C \to D$, $D, C \to B$.
R_2 non rispetta la forma normale di Boyce-Coddper la dipendenza $D \to B$ e dunque serve un altro passo di scomposizione per la realzione R_2.
R_2 viene scomposta nelle relazioni $R_3 = \{D, B\}$ con dipendenza $D \to B$ e $R_4 = \{D, C\}$ con un insieme vuoto di dipendenze.
Il risultato della scomposizione è dato da R_1, R_3 e R_4 con le dipendenze indicate. †