Basi di Dati e Sistemi Informativi I, Prove scritte AA. 2002-03

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 Prova scritta del 2-03-04

Si consideri il seguente schema relazionale per un sistema di fatturazione e gestione di magazzino:
CLIENTE(PI,NOME)
ARTICOLO(COD-A,DESCRIZIONE,PREZZO,DISPONIBILITA)
FATTURA(ID,NUMERO,ANNO,PI,PAGATA)
VOCEFATTURA(ID,COD-A,PREZZO,QUANTITA)

Ad ogni cliente associato un numero di partita IVA (PI) ed un nome, mentre ogni articolo è caratterizzato da un codice (COD-A), da una descrizione da un prezzo e dal numero di pezzi disponibile. All'atto della vendita di alcuni prodotti ad un cliente viene emessa una FATTURA caratterizzata da un numero progressivo (azzerato allinizio di ogni anno) e da un anno, dalla partita IVA del cliente, da una chiave (ID) e da un flag (PAGATA) che indica se è avvenuto il pagamento. Ogni fattura è composta da un insieme di voci, riportate nella tabella VOCEFATTURA, che specificano il codice dell' articolo, il prezzo effettivamente pagato e la quantità acquistata.

Esercizio 11 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce linsieme delle terne (nome cliente, descrizione articolo, anno) per le quali larticolo non è mai stato acquistato dal cliente nellanno indicato.*

Esercizio 12 *Si scriva una query in SQL che fornisca per ogni fattura emessa: il numero progressivo, lanno, il nome del cliente e limporto della fattura*

Esercizio 13 *Si crei una vista che fornisca per ogni coppia (cliente, anno) la partita iva, il nome del cliente, anno, fatturato complessivo sviluppato dal cliente, numero totale di fatture emesse , importo pagato, importo non anora pagato, numero di fatture pagate, numero di fatture sospese.*

Esercizio 14 *Si scriva una procedura in C o Pascal che accettando in ingresso la partita iva di un cliente guidi interattivamente lutente alla emissione di una fattura. In dettaglio occorrerà creare una nuova fattura con un numero progressivo maggiore di uno del pi alto per lanno corrente, quindi iterativamente chiedere in input delle coppie [codice articolo,quantità] dei prodotti acquistati ed inserirle opportunamente nella base di dati. Si presti attenzione alla correttezza del codice articolo e alle disponibilità effettive del magazzino. Infine i stampi, a video, la fattura completa (partita iva, nome cliente, importo totale e per ogni articolo la quantità, la descrizione il prezzo unitario e quello totale).*

Esercizio 15 *Si mostri un diagramma ER che generi lo schema della base di dati in esame. Si illustrino i vincoli di integrit referenziale intercorrenti tra le varie tabelle*

2 Prova scritta del 10-02-04

Si consideri il seguente schema relazionale che descrive la composizione di piani di studio in un corso di laurea:
$ESAME(\underline{COD-E}, TITOLO, CFU, TIPO)$
$VINCOLO(\underline{COD-E, PROP})$
$ANNO(\underline{COD-E}, AC)$
$PIANIS(\underline{MATR, COD-E}, AC)$.
Lo schema $ESAME$ contiene la descrizione di un insegnamento: codice insegnamento, nome, numero di CFU e tipo ('Obbligatorio' o 'Scelta'); Lo schema $VINCOLO$ contiene la descrizione dei vincoli di propedeuticità tra esami: se un esame di codice xxxx è propedeutico ad un esame di codice yyyy, è presente nella tabella corrispondente una riga $(yyyy, xxxx)$; Lo schema $ANNO$ contiene la descrizione degli anni di corso (con valori I, II, III etc) nell'attributo AC). Se, ad esempio, un esame di codice xxxx puo essere seguito indifferentemente al II o III anno, la tabella corrispondente contiene due righe $(xxxx, II)$ e $(xxxx, III)$. Lo schema $VINCOLO$ contiene il piano di studi di ogni studente identificato dall'attributo MATR. La tabella corrispondente contiene un una riga per ogni esame (sia obbligatorio sia a scelta) presente nel piano di studi.

Esercizio 21 *Un esame si dice di base se non ha esami propedeutici. Si scriva una espressione in algebra relazionale che, se valutata, fornisce i nomi delgi esami che se hanno esami propedeutici, hanno solo esami propedeutici di base.*

Esercizio 22 *Scrivere una interrogazione SQL che fornisca titolo e anno degli esami non obbligatori che possono essere collocati solo in un anno (non c'è scelta d'anno).*

Esercizio 23 *Scrivere una vista che fornisca, per ogni studente, un riepilogo che permetta di controllare la correttezza dei piani di studio. La vista deve avere i seguenti campi: la matricola dello studente, il numero di obbligatori mancanti, il numero di esami collocati in anni sbagliati, il numero di CFU complessivo derivante da esami al al I anno, al II anno e al III anno.*

Esercizio 24 *Si scriva una procedura in C o Pascal che riceve in ingresso una matricola di uno studente e restituisce in uscita il numero degli insegnamenti presenti nel piano di studi per i quali manchino esami propedeutici. Inoltre, la procedura, per ogni insegnamento propedeutico mancante, inserisce una riga in una tabella (da supporre esistente al momento dell'esecuzione della procedura) avente schema*
$ERRATI(MATR, COD-E, COD-PROP)$ *(COD-E è il codice dell'insegnamento presente nel piano di studi e COD-PROP è il codice del suo insegnamento propedeutico mancante).*

Esercizio 25 *Si dica cosa si intende con il termine transazione e si dica quali sono i costrutti per programmare una transazione. Si descrivano inoltre brevemente le proprietà 'acide' (ACID) delle transazioni.*

3 Prova scritta del 28-09-03

Si consideri il seguente schema relazionale che descrive il trasferimento di file in un sistema di file-sharing:
$UTENTE(NOME, BANDA-IN, BANDA-OUT)$
$POSSIEDE(NOME, NOMEF)$
$DOWNLOAD(PROP, RIC, NOMEF, BANDA)$

Ogni utilizzatore ($UTENTE$) del sistema è caratterizzato da un $Nome$, da una banda in ingresso ($Banda-IN$) e da una banda in uscita ($Banda-OUT$). Ogni utente può condividere diversi file come espresso dalla tabella $POSSIEDE$, dove $Nomef$ è il nome del file e $Nome$ è il nome di un utente che lo condivide. Lo schema $DOWNLOAD$ rappresenta i trasferimenti attivi, $Prop$ è il nome dellutente da cui viene scaricato il file, $Nomef$ è il nome del file scaricato, Ric è il nome dellutente che lo sta scaricando, mentre $Banda$ è la banda impiegata per quel trasferimento.

Esercizio 31 *Si scriva un'espressione in algebra relazionale che, se valutata, fornisce i nomi di tutti i file posseduti da un solo utente (non si usino operazioni di conteggio).*

Esercizio 32 *Scrivere un'interrogazione SQL che fornisce tutte le possibili coppie* $(Usr, Nomef)$ *dove* $Nomef$ *non è posseduto da* Usr.

Esercizio 33 *Si crei una vista che fornisca per ogni utente: banda in ingresso totale, banda effettivamente impiegata da download, banda in uscita totale, banda effettivamente impiegata da upload, numero di download ed upload attivi. Si assegni all'utente USER il diritto di lettura sulla vista.*

Esercizio 34 *Si scriva una procedura in C o in PASCAL che riceve in ingresso il nome di un utente, il nome di un file, e la banda richiesta per il download. La procedura dovrà inserire una nuova riga in DOWNLOAD relativo al file e allutente indicato. Ciò dovrà essere fatto provando a scaricare il file dallutente che ha disponibile in uscita (tenedo conto degli upload correnti) la più piccola banda maggiore di quella richiesta. Nel caso nessuno abbia a disposizione una banda in uscita sufficiente, allora il download sarà effettuato presso lutente con maggior banda disponibile (ovviamente occorrerà controllare che la banda richiesta sia compatibile con la banda in ingresso disponibile dell'utente che richiede il file). In uscita la procedura restituirà il nome dell'utente da cui viene scaricato il file e la banda assegnata. Nel caso il file non sia disponibile presso nessun utente oppure sia già posseduto dall'utente indicato si restituirà una segnalazione di errore.*

Esercizio 35 *Nell'ambito del controllo di affidabilità dei sistemi per la gestione delle basi di dati si descriva l'utilità e la struttura dei file di log. Inoltre si descriva l'operazione di ripresa a caldo in seguito ad un malfunzionamento.*

Soluzione Esercizio 31
Sia PS la relazione per POSSIEDE.
Nomi di file posseduti da almeno due persone.

$$PIU \leftarrow \Pi_{Nomef}(\sigma_C(\rho_{PR1(Usr1,Nomef1)}(PS) \times PS))$$

dove C è la condizione $Usr <> Usr1 \wedge Nomef = Nomef1$.
Risultato.

$$\Pi_{Nomef}(PS) \setminus PIU$$

† **Soluzione Esercizio 32**

```
SELECT A.Usr,B.Nomef
FROM   UTENTE AS A, POSSIEDE AS B
WHERE B.Nomef NOT IN
       (SELECT Nomef
       FROM   POSSIEDE AS C
       WHERE  A.Usr = C.Usr)
```

Una soluzione alternativa basata sulle operazioni insiemistiche può essere la seguente:

```
SELECT A.Usr,B.Nomef
FROM   UTENTE AS A, POSSIEDE AS B
EXCEPT
SELECT *
FROM   POSSIEDE
```

†
Soluzione Esercizio 33
Vista per il computo dell'informazione riguardante UPLOAD per un generico utente.

```
CREATE VIEW Upload(Usr,BandaU,NU) AS
       SELECT Prop,SUM(Banda),Count(*)
       FROM   DOWNLOAD
       GROUP BY Prop
```

Vista per il computo dell'informazione riguardante DOWNLOAD per un generico utente.

```
CREATE VIEW Download(Usr,BandaD,ND) AS
       SELECT Ric,SUM(Banda),Count(*)
       FROM   DOWNLOAD
       GROUP BY Ric
```

Creazione della vista finale.

```
CREATE VIEW Conteggio(Usr, Banda-IN, BandaD, ND, Banda-OUT, BandaU, NU) AS
       SELECT Usr, Banda-IN, BandaD, ND, Banda-OUT, BandaU, NU,
       FROM   UTENTE NATURAL JOIN Download NATURAL JOIN
              Upload
```

Assegnazione dei privilegi
GRANT SELECT ON Conteggio TO USER †
Soluzione Esercizio 34
La procedura proposta, per semplicità di soluzione, non adotta una strategia di interrogazione del database efficiente (duplicazione dell'interrogazione).

```
PROCEDURE Scarica (IN Utente,File: string, Banda: integer
                   OUT Error : boolean)
```

```
VAR
EXEC SQL BEGIN DECLARE SECTION
     Ut, Ut2, Fl,Pr : string;
     Bd, Ds : integer
EXEC SQL END DECLARE SECTION

BEGIN
Ut:=Utente; Fl:= File; Bd:= Banda; Error:= False
{**Verifica che il file non sia gia' posseduto**}
EXEC SQL SELECT Usr INTO :Ut2
            FROM   POSSIEDE
            WHERE Usr = :Ut AND Nomef = :Fl;
IF SQLCODE=0 THEN Error := TRUE;
{**Verifica che il file sia disponibile con una banda sufficiente**}
IF NOT Error THEN
BEGIN
EXEC SQL DECLARE disp CURSOR FOR
            SELECT Usr, Banda-OUT - SUM(Banda) AS Libero
            FROM   UTENTE,POSSIEDE,DOWNLOAD
            WHERE UTENTE.Usr = POSSIEDE.Usr AND
                        UTENTE.Usr = DOWNLOAd.PROP AND
                        POSSIEDE.Nomef = :Fl
            GROUP BY Usr
            HAVING Libero ≥ :Bd
            ORDER BY Libero;
EXEC SQL OPEN disp;
EXEC SQL FETCH disp INTO :Pr, :Ds;
IF SQLCODE = 0 THEN
{**File trovato con banda sufficiente**}
EXEC SQL INSERT INTO DOWNLOAD VALUES (:Pr,:Ut,:Fl,:Bd)
ELSE
BEGIN
{**Cerca file con banda inferiore alla richiesta**}
EXEC SQL DECLARE disp2 CURSOR FOR
            SELECT usr, Banda-OUT - SUM(Banda) AS Libero
            FROM   UTENTE,POSSIEDE,DOWNLOAD
            WHERE UTENTE.Usr = POSSIEDE.Usr AND
                        UTENTE.Usr = DOWNLOAd.Prop AND
                        POSSIEDE.Nomef = :Fl
            GROUP BY usr
            HAVING Libero > 0
            ORDER BY Libero DESC;
EXEC SQL OPEN disp2;
EXEC SQL FETCH disp INTO :Pr, :Ds;
IF SQLCODE = 0 THEN
EXEC SQL INSERT INTO DOWNLOAD VALUES (:Pr,:Ut,:Fl,:Ds)
ELSE Error := True
END
END
```

END

†

4 Prova scritta del 8-09-03

Si consideri il seguente schema relazionale che descrive la composizione di linee ferroviarie, della composizione dei treni che effettuano i tragitti e delle prenotazioni dei posti sui treni. $LINEA(\underline{COD-L}, COD-T, ST-P, ST-A, OP, OA)$
$TRATTA(\underline{COD-L, ST-P}, ST-A, OP, OA)$
$C-TRENO(\underline{COD-T, N-VAG}, Posti, Tipo)$
$PRENOTAZIONI(\underline{Cod-P}, Data, COD-L, ST-P, ST-A, OP, OA, N-VAG, N-Posto)$

Ogni linea ferroviaria (LINEA) è indicata univocamente da un codice COD-L, da un codice del treno che effettua il servizio (COD-T), dalle stazioni di partenza e destinazione (ST-P e ST-A) e dai relativi orari di partenza ed arrivo (OA e OP). Le fermate intermedie di una linea sono descritte dallo schema TRATTA. Una tratta va intesa come una porzione della linea che non ha fermate intermedie ed è indicata dalla stazione di partenza ed arrivo (ST-P e ST-A) e dai relativi orari (OA e OP). (Una istanza di tratta è presente in TRATTA anche se la linea corrispondente è costituita da un'unica tratta.) Lo schema C-TRENO descrive la composizione dei treni (in vagoni): ogni istanza descrive un vagone che compone il treno COD-T indicandone il numero (N-VAG) ed il numero complessivo di posti disponibili nel vagone (si assume che che se N è il numero di posti complessivi i posti siano indicati dai numeri progressivi 1..N). Le prenotazioni sono descritte in $PRENOTAZIONI$. Ogni istanza di prenotazione è identificata da un codice, data, linea ferroviaria, stazione ed ora di partenza ed arrivo, numero di vagone e di posto. La prenotazione riguarda soltanto il sottoinsieme delle tratte della linea comprese tra la stazione di partenza ed arrivo (e non, dunque, necessariamente una singola tratta o l'intera linea).

Esercizio 41 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce il codice di linea, la data, il numero di vagone e di posto a cui siano associate nell'intero tragitto almeno due prenotazioni distinte. (Esempio: nella stessa corsa sulla linea NA-RM-FI un posto può avere due diverse prenotazioni, una sulla tratta NA-RM e l'altra sulla tratta RM-FI). Nello svolgimento dell'esercizio NON si usino operazioni di raggruppamento.*

Esercizio 42 *Scrivere una interrogazione SQL che fornisca come risultato, codice di linea, data, numero di vagone e di posto, per posti che siano prenotati per l'intera linea (non si trascuri il caso generale in cui la prenotazione sull'intera linea risulta dal cumulo di prenotazioni distinte su parti della linea).*

Esercizio 43 *Si crei una vista che fornisca data, codice di linea, numero complessivo di posti disponibili nel treno, numero complessivo di posti per cui esiste una prenotazione almeno su una tratta (non necessariamente sull'intera linea), numero di posti per cui non esiste nessuna prenotazione, numero di posti che sono prenotati in alcune tratte ma non in tutte. Si assegni un diritto di lettura all'utente 'USER' sulla vista.*

Esercizio 44 *Si scriva una procedura in C o Pascal che riceve in ingresso i seguenti dati per effettuare una prenotazione: Codice di prenotazione,Data, Codice di linea,Stazione di partenza,Stazione di arrivo (Ora di partenza e di arrivo non vengono forniti e devono essere calcolati all'interno della procedura). La procedura prima controlla tra i posti che già sono stati prenotati per verificare se è possibile assegnare la prenotazione ad un posto già parzialmente prenotato. Se esistono posti che soddisfano questi requisiti la prenotazione viene effettuata su uno qualsiasi di essi. Se invece posti parzialmente occupati compatibili non esistono, allora si verifica se vi siano posti liberi su tutta la linea: se ve ne sono, si prenota il primo posto disponibile (ordinando i posti per numero di vagone e di posto) altrimenti si fornisce una segnalazione di indisponibilità.*

Esercizio 45 *Nell'ambito del controllo di affidabilità dei sistemi per la gestione delle basi di dati si descriva l'utilità e la struttura del file di log. In particolare si indichi a quali operazioni di sistema corrispondono i record di Dump e Checkpoint che nel file di log possono essere registrati.*

4.1 Soluzioni

(NOTA. Nella valutazione del compito, ha ottenuto pieno punteggio chi abbia svolto correttamente almeno tutti gli esercizi ad eccezione del quarto)
Soluzione Esercizio 41
Sia PR la relazione per PRENOTAZIONI.
Copia della relazione di prenotazione.

$$PR1 \leftarrow \rho_{PR1(Cod-P_1,Data_1,...,N-Vag_1,N-Posto_1)}(PR)$$

Risultato.

$$\Pi_{Cod_L,Data,N-Vag,N-Posto}(PR \bowtie_C PR_1)$$

dove C è la condizione $Cod-L = Cod-L_1 \wedge Data = Data_1 \wedge N-Vag = N-Vag_1 \wedge N-Posto = N-Posto_1 \wedge Cod-P <> Cod-P_1$ †
Soluzione Esercizio 42
L'idea adottata in questa soluzione è di elencare i posti per i quali non vi sia una tratta non coperta da prenotazione.

```
SELECT Cod-L,Data,N-Vag,N-Posto
FROM   PRENOTAZIONI AS A
WHERE NOT EXISTS
       (SELECT*
       FROM   TRATTA AS B
       WHERE A.Cod-L = B.Cod-L AND
              NOT EXISTS
              (SELECT*
              FROM   PRENOTAZIONI AS C
              WHERE A.Cod-L = C.Cod-L AND A.Data = C.Data AND
                     A.N-Vag = C.N-Vag AND A.N-Posto = C.N-Posto AND
                     C.OP ≤ B.OP AND C.OA ≥ B.OA))
```

†
Soluzione Esercizio 43
Vista per il computo del numero di posti complessivi

```
CREATE VIEW PostiTotatli(Cod-L,TotPosti) AS
        SELECT Cod-L,SUM(Posti)
        FROM   LINEA NATURAL JOIN C-TRENO
        GROUP BY Cod-L
```

Vista per il computo dei posti occupati

```
CREATE VIEW PostiOccupati(Cod-L,Data,TotOccupati) AS
        SELECT Cod-L, Data, COUNT(DISTINCT N-Vag,N-Posto)
        FROM   PRENOTAZIONE
        GROUP BY Cod-L, Data
```

Vista per il computo dei posti occupati su ogni tratta (riconducibile all'esercizio 42)

```
CREATE VIEW SempreOccupati(Cod-L,Data,TotSempre) AS
        SELECT Cod-L, Data, COUNT(DISTINCT N-Vag,N-Posto)
        FROM   *Come in esercizion 42
        WHERE *Come in esercizion 42
        GROUP BY Cod-L, Data
```

Creazione della vista finale.

```
CREATE VIEW Conteggio(Cod-L,Data,TotPosti,Occupati,Liberi,Parziali) AS
        SELECT Cod-L, Data, TotPosti,TotOccupati,
                TotPosti-TotOccupati,TotOccupati-TotSempre
        FROM   PostiTotali NATURAL JOIN PostiOccupati NATURAL JOIN
SempreOccupati
```

Assegnazione dei privilegi GRANT SELECT ON Conteggio TO USER †

Soluzione Esercizio 44

```
PROCEDURE Prenota (IN CodiceP,CodiceL,DataP,SP,SA : string,
                        OUT Prenotato : boolean)

VAR
EXEC SQL BEGIN DECLARE SECTION
    OraP,OraA : string;
    CodPren,CodLin,StazP,StazA,DPart : string;
    Vagone,Posto : integer
EXEC SQL END DECLARE SECTION

BEGIN
CodPren:=CodiceP; CodLin:= CodiceL; Dpart:=DataP
StazP:=SP; StazA:=SA;
Prenotato:=False;
{**Calcolo dell'ora di partenza ed arrivo**}
EXEC SQL SELECT OP INTO :OraP
            FROM   TRATTA
            WHERE Cod-L = :CodLin AND ST-P = :StazP;
EXEC SQL SELECT OA INTO :OraA
            FROM   TRATTA
```

```
            WHERE Cod-L = :CodLin AND ST-A = :StazA;
{**Definizione del cursore per l'insieme dei posti gia' prenotati che consentono l'aggiunta
della prenotazione richiesta **}
EXEC SQL DECLARE occ CURSOR FOR
            SELECT N-Vag, N-Posto
            FROM   PRENOTAZIONI AS A
            WHERE Cod-L = :CodLin AND Data = :Dpart AND NOT EXISTS
                        (SELECT *
                        FROM PRENOTAZIONI AS B
                        WHERE A.Data = B.Data AND A.Cod-L = B.Cod-L AND
                                    A.N-VAg = B.N-Vag AND A.N-Posto = B.N-Posto AND
                                    ((OP ≤ :OraP AND OA > :OraP) OR
                                    (OP ≥ :OraP AND OP < :OraA) ))
            ORDER BY N-Vag, N-Posto;
{**Definizione del cursore per l'insieme dei posti liberi su cui inserire la prenotazione.
Un posto libero e' il successivo del posto di numero massimo occupato in un vagone **}
EXEC SQL DECLARE lib CURSOR FOR
            SELECT = N-Vag, MAX(N-Posto)+1
            FROM   PRENOTAZIONI NATURAL JOIN LINEE NATURAL JOIN C-TRENO
            WHERE Cod-L = :CodLin AND Data = :Dpart
            GROUP BY N-Vag
            HAVING MAX(N-Posto)< C-TRENO.Posti
            ORDER BY N-Vag
EXEC SQL OPEN occ;
{**Se il cursore occ contiene almeno una riga esiste un posto prenotato a cui e' possibile
aggiungere la prenotazione corrente **}
EXEC SQL FETCH occ INTO :Vagone, :Posto;
IF SQLCODE=0 THEN
            BEGIN
            Prenotato := TRUE;
            INSERT INTO PRENOTAZIONI VALUES (:CodPren,:Dpart,:CodLin,
                        :StazP,:StazA,:OraP,:OraA,:Vagone,:Posto)
            END
ELSE
            BEGIN
            EXEC SQL OPEN lib;
            IF SQLCODE=0 THEN
                        BEGIN
                        Prenotato := TRUE;
                        INSERT INTO PRENOTAZIONI VALUES (:CodPren,:Dpart,:CodLin,
                                    :StazP,:StazA,:OraP,:OraA,:Vagone,:Posto)
                        END
            END
END
```

†

5 Prova scritta del 22-07-03

Si consideri il seguente schema relazionale che descrive la composizione di tratte ferroviarie:
$LINEA(\underline{COD-L}, NomeTreno, CittaP, CittaA, OraP, OraA, km)$
$TRATTA(\underline{COD-L, CittaP}, CittaA, OraP, OraA, Km)$
$GEO(Citta, Provincia, Regione)$.
Lo schema $LINEA$ contiene la descrizione del tragitto complessivo del treno: città di partenza e arrivo, ora di partenza e arrivo, lunghezza complessiva. Lo schema $TRATTA$ contiene la descrizione delle singole tratte che costituisco una linea (sottopercorso senza fermate intermedie): città di partenza e arrivo, ora di partenza e arrivo, lunghezza della tratta.

Esercizio 51 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce l'insieme dei nomi dei treni regionali, cioè dei treni su linee con tutte le fermate nella medesima regione.*

Esercizio 52 *Scrivere una vista che fornisca l'elenco delle coppie dei codici delle linee (cioè uno schema del tipo $INTERSECT(Cod-L1, Cod-L2)$) che contenga le coppie dei codici delle linee che hanno una stazione di coincidenza, cioè di arrivo per una linea e partenza per l'altra (considerare anche le stazioni interne alle linee).*
Sfruttare la vista ottenuta per fornire l'elenco delle coppie delle città che non sono collegate da una linea diretta ma che sono raggiungibili mediante un unico cambio di treno.

Esercizio 53 *La base di dati sopra descritta è accessibile a due categorie di utenti: operatori e generici. Un operatore ha accesso in lettura a tutte le informazioni e può solo fare variazioni sui campi degli orari delle tabelle. L'utente generico ha accesso in lettura ad una tabella dei treni regionali (di schema $LINEAREG(Regione, \underline{COD-L}, NomeTreno, CittaP, CittaA, OraP, OraA, km)$) ed alla sottotabella di TRATTE che si riferisce ai treni regionali.*

Esercizio 54 *Si immagini che oltre alle tabelle sopra descritte, nella base di dati vi sia una tabella $COPPIE(\underline{COD-L, CittaP, CittaA}, OraP, OraA, Km)$*
che mantiene informazione su coppie di città partenza-destinazione che stanno su una medesima linea e che a differenza di $TRATTE$ possono avere tra la partenza e la destinazione delle altre tappe intermedie (km in questo caso riporta la distanza complessiva tra la partenza e la destinazione).
Si scriva una procedura in C o Pascal che riceve in ingresso il codice di una linea e che ha come effetto l'inserimento di tutte le coppie partenza-destinazione nella tabella $COPPIE$ per la linea data. La distanza complessiva tra la partenza e la destinazione è la somma delle lunghezze delle singole tratte. (Suggerimento: si interroghi la base di dati per le coppie ammissibili e per ciascuna coppia si ottenga mediante altra interrogazione il computo dei km complessivi.)

Esercizio 55 *Si descriva la tecnica del timestamping per la gestione dell'accesso concorrente alle basi di dati. Si forniscano poi, come esemplificazione, due sequenze di letture e scrittura operate (ciascuna) da due transazioni $T1$ e $T2$. La prima sequenza di operazioni sia costruita in modo tale da rispettare la politica di timestamping (nessuna lettura o scrittura della sequanza viene cancellata). La seconda sequenza sia costruita in modo da violare la politica di timestamping.*

5.1 Soluzioni

Soluzione Esercizio 51
Siano LN, TT e GO la relazione per LINEA, TRATTA e GEO, rispettivamente.
Tratte con indicazione delle regioni di partenza e arrivo.

$$TT - REG \leftarrow \Pi_{Cod-L,Reg-P,Reg-A}(((TT \bowtie \rho_{GOP(CittaP,PrP,Reg-P)}(GO)) \bowtie \bowtie \rho_{GOA(CittaA,PrA,Reg-A)}(GO))$$

Codici di linea che non hanno tratte interregionali.

$$LN - REG \leftarrow \Pi_{Cod_L}(LN) \setminus \Pi_{Cod-L}(\sigma_{Reg-P \neq Reg-A}(TT - REG))$$

Soluzione

$$\Pi_{NomeTreno}(LN - REG \bowtie LN)$$

† **Soluzione Esercizio 52**
La soluzione proposta adotta l'ipotesi che per ogni linea in una direzione esista anche la linea corrispondente e similare (stesse tratte) nella direzione opposta.
Creazione della vista Intersect

```
CREATE VIEW Intersect(Cod-L1,Cod-L2) AS
        SELECT A.Cod-l,B.Cod-l
        FROM   TRATTA A, TRATTA B
        WHERE A.Cod-l <> B.Cod-l AND
                (A.CittaP=B.CittaA OR A.CittaA=B.CittaP)

SELECT A.CittaP,B.CittaA
FROM   TRATTA A, TRATTA B, Intersect
WHERE A.Cod-L=Cod-L1 AND B.Cod-L= Cod-L2 AND
        A.CittaP<>B.CittaA
EXCEPT
SELECT A.CittaP,B.CittaA
FROM   TRATTA A, TRATTA B
WHERE A.Cod-L=B.Cod-L
```

†
Soluzione Esercizio 53
Per l'operatore non vanno create viste e i privilegi vanno assegnati direttamente sulle tabelle. GRANT SELECT ON LINEA TO Operatore
GRANT SELECT ON TRATTA TO Operatore
GRANT SELECT ON GEO TO Operatore
GRANT UPDATE(OraP,OraA) ON LINEA TO Operatore
GRANT UPDATE(OraP,OraA) ON TRATTA TO Operatore
Creazione della vista delle linee regionali per l'utente generico.

```
CREATE VIEW LineaRegionale AS
        SELECT Regione, A.*
        FROM   (LINEA AS A) JOIN GEO ON (CittaP=Citta)
        WHERE A.Cod-L NOT IN
                SELECT Cod-L
```

```
        FROM    (TRATTA JOIN (GEO AS K) ON CittaP = K.Citta)
                JOIN GEO AS H ON CittaA = H.Citta
        WHERE H.Regione<>K.Regione
```

Creazione della vista delle tratte regionali per l'utente generico.

```
CREATE VIEW TrattaRegionale AS
        SELECT *
        FROM    TRATTA
        WHERE Cod-L IN (SELECT Cod-L FROM LineaRegionale)
```

Assegnazione dei privilegi a Generico GRANT SELECT ON LineaRegionale TO Generico
GRANT SELECT ON TrattaRegionale TO Generico †

Soluzione Esercizio 54

```
PROCEDURE Caricacoppie (IN Cod : string)
VAR
EXEC SQL BEGIN DECLARE SECTION
    OraPartenza,OraArrivo : date;
    Codice,CittaPartenza,CittaArrivo : string;
    Totale : integer
EXEC SQL END DECLARE SECTION

BEGIN
Codice:=Cod;
EXEC SQL DECLARE cc CURSOR FOR
            SELECT A.CittaP, B.CittaA, A.OraP, B.OraA
            FROM    TRATTA AS A JOIN TRATTA AS B ON A.Cod-L = B.od-L
            WHERE A.Cod-L = :Codice AND A.OraP < B.OraA;
EXEC SQL OPEN cc;
EXEC SQL FETCH cc INTO :Partenza, :Arrivo, :OraPartenza, :OraArrivo;
WHILE SQLCODE=0 DO
            BEGIN
            EXEC SQL SELECT SUM(km) INTO :Totale
                        FROM TRATTA
                        WHERE Cod-L = :Codice AND
                                    OraP >= :OraPartenza AND OraA <= :OraArrivo;
            EXEC SQL INSERT INTO Coppie VALUES
                        (:Codice,:Partenza, :Arrivo, :OraPartenza, :OraArrivo, :Totale);
            EXEC SQL FETCH cc INTO :Partenza, :Arrivo, :OraPartenza, :OraArrivo;
            END;
END

†
```

6 Prova scritta del 25-06-2003

Si consideri il seguente schema relazionale che descrive il modo in cui alcuni pezzi meccanici semplici sono integrati tra loro per comporre nuovi pezzi e che tiene informazione sui produttori dei pezzi:
$PEZZO(\underline{COD-P}, TIPO)$
$PRODUTTORE(\underline{MARCA}, PI, INDIRIZZO)$
$COMPONE(\underline{COD-PARTE, COMPOSTO}, QUANTITA)$
$CATALOGO(\underline{COD-P, MARCA}, COSTO)$.
Lo schema $COMPONE$ indica se un pezzo (COD-PARTE) è parte costitutiva di di un pezzo composto (COMPOSTO) e in quale quantità. Lo schema $CATALOGO$ indica quali produttori forniscono i pezzi e a quale prezzo.

Esercizio 61 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce l'elenco del TIPO dei pezzi composti da soli pezzi di base (cioè non scomponibili in altri pezzi).*

Esercizio 62 *Fornire mediante una interrogazione SQL l'elenco dei produttori (MARCHE) e la loro Partita Iva (PI) in grado di fornire almeno tutti i pezzi forniti dal produttore ALLMEC.*

Esercizio 63 *Si immagini che vi siano le seguenti categorie di utenti: Amministrazione, Segreteria, Utente. Si progetti in SQL un insieme di viste (con relativi diritti) sapendo che*

- *l'utente Amministrazione ha accesso in lettura e scrittura a tutte le tabelle;*
- *l'utente Segreteria puó accedere in lettura a tutte le tabelle, inserire dati nelle tabelle PEZZI, PRODUZIONE e CATALOGO, variare solo il costo nella tabella CATALOGO;*
- *l'utente Utente può accedere in lettura al catalogo dei pezzi di base limitatamente ai campi MARCA, COSTO, TIPO e ad un catalogo riassuntivo dei pezzi di base dove ad ogni tipologia di pezzo viene associata la marca che garantisce il prezzo minimo, il prezzo minimo, la marca che grantisce il prezzo massimo ed il prezzo massimo.*

Esercizio 64 *Si integri lo schema relazionale sopra descritto con due nuovi schemi di relazioni che fungono da deposito storico per i pezzi fuori commercio e per il catalogo fuori commercio:* $STORICOPEZZO(\underline{COD-P}, TIPO)$ *e*
$STORICOCATALOGO(\underline{COD-P, MARCA}, COSTO)$.
Si definisca una procedura in C o PASCAL che riceve in ingresso il codice di un pezzo e sortisce come risultato il trasferimento delle righe del catalogo che hanno quel codice dalle tabelle $PEZZO$ *e* $CATALOGO$ *alle tabelle* $STORICOPEZZO$ *e* $STORICOCATALOGO$*, rispettivamente. Per organizzare le operazioni si tenga conto che vi è un vincolo di integrità referenziale tra la tabella* $CATALOGO$ *e la tabella PEZZI che impedisce la violazione del vincolo in cancellazione.*

Esercizio 65 *Si definisca la nozione di* indice primario *e se ne descriva brevemente la struttura.*

7 Elaborato di prova, Maggio 2003

Negli esercizi di seguito elencati si faccia riferimento agli schemi relazionali di seguito riportati:

$$Paziente(\underline{CF}, Nome, Cognome, Indirizzo, Telefono)$$

$$Ricovero(\underline{CodR}, DataI, DataF, CF)$$

$$Esame(\underline{CodE, CodR, data}, esito)$$

$$TipoEsame(\underline{CodE}, Descrizione, Ticket)$$

dove, CF indica il Codice Fiscale del paziente; $CodR$, $DataI$ e $DataF$ indicano il codice identificativo la data di inizio e di fine, rispettivamente, del ricovero; $CodE$ indica il codice identificativo dell'esame.

Esercizio 71 *Scrivere una espressione dell'algebra relazionale che, valutata, fornisca Nome, Cognome dei pazienti la Descrizione degli esami sostenuti almeno due volte nell'ambito di uno stesso ricovero.*

Esercizio 72 *Scrivere una interrogazione SQL che fornisca Nome e Cognome dei pazienti e l'ammontare totale dei tiket relativi agli esami a cui si sono sottoposti nel loro ultimo ricovero.*

Esercizio 73 *Si immagini che la base di dati sia accessibile a tre categorie distinte di utenti: Medico, Amministratore, Esterno. Gli utenti hanno i seguenti privilegi di accesso: l'utente Medico ha accesso in lettura e scrittura (r/w) a tutti i dati; l'utente Amministratore ha acecsso r/w ai dati di TipoEsame, in lettura ai dati di Paziente e Ricovero, ed ha accesso in lettura solo alla data e al costo (ticket) degli Esami (non deve conoscere la tipologia dell'esame e l'esito); l'utente Esterno ha accesso in lettura soltanto ad un elenco con nome e cognome degli utenti attualmente ricoverati.*
Definire un insieme di viste e privilegi per le tre tipologie di utenti descritti.

Esercizio 74 *Scrivere una procedura che riceve in ingresso una data e, dopo aver recuperato nella base di dati tutti i pazienti ancora degenti a partire da quella data, restituisce in uscita un array di dimensioni (Mx2) inizializzatao col nome e cognome dei pazienti selezionati.*

Esercizio 75 *Si discutano le linee essenziali della tecnica del locking gerarchico.*