PROVA SCRITTA DI TECNOLOGIA DATABASE – 17/02/2012

Corso di Laurea Magistrale in Ingegneria Informatica – DM 270

Corso di Laurea Specialistica in Ingegneria Informatica – DM 509

PROF. SONIA BERGAMASCHI

Esercizio 1 (punti 18)

Dato il seguente schema relazionale:

```
TRENO(CODT,TIPO,DESC)
STAZIONE(NOME,CITTA,NAZIONE)
VIAGGIO(CODV,CODT,PARTENZA,ARRIVO,COSTO,RITARDO)
FK: PARTENZA REFERENCES STAZIONE
FK: ARRIVO REFERENCES STAZIONE
FK: CODT REFERENCES TRENO
PRENOTAZIONE(CF,CODV)
FK: CODV REFERENCES VIAGGIO
(RITARDO è di tipo intero ed esprime i ritardi in minuti)
```

Scrivere in SQL la seguente interrogazione

1) Calcolare per ogni treno il rimborso medio sostenuto per i viaggi con ritardo, sapendo che per ritardi maggiori o uguali a 30 minuti ed inferiori a 120 minuti ogni passeggero ha diritto ad un rimborso pari al 30% del costo del viaggio, mentre per ritardi superiori o uguali ai 120 minuti il rimborso è pari al 50%.

Scrivere in embedded SQL la seguente interrogazione

2) Mostrare, per ciascun treno, il costo medio dei viaggi internazionali, trascurando i 2 viaggi di costo inferiore. In caso di treni con soli due viaggi mostrare come costo medio 0.

Scrivere in linguaggio jsp o asp.net

3) una pagina web che consente di inserire il nome di una nazione e che ne genera un'altra contenente per ogni tipo di treno, il treno che ha compiuto il maggior numero di viaggi nazionali, all'interno della nazione specificata, e il ritardo medio che ha accumulato.

Esercizio 2 (punti 6)

Dato il seguente schema relazionale:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

(FD1) $AC \rightarrow B$

(FD2) $C \rightarrow A$

(FD3) $D \rightarrow C$

Viene richiesto di:

- Determinare la chiave o le chiavi dello schema di relazione;
- Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
- Produrre eventuali decomposizioni dello schema in BCNF che preservano i dati e discuterne la preservazione delle dipendenze funzionali.

  Per la soluzione non ci si può avvalere del teorema 7 sulla preservazione dei dati

Esercizio 3 (punti 6)

Dato la seguente porzione di schema relazionale:

```
VIAGGIO(CODV,COSTO,RITARDO)
PASSEGGERO(CF,NOME,RIMBORSO)
PRENOTAZIONE(CF,CODV)
FK: CF REFERENCES PASSEGGERO
FK: CODV REFERENCES VIAGGIO
```

Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che, a fronte di inserimenti, modifiche o cancellazioni nella relazione VIAGGIO mantenga aggiornati i rimborsi dei passeggeri considerando che per ritardi maggiori o uguali a 30 minuti ed inferiori a 120 minuti ogni passeggero ha diritto ad un rimborso pari al 30% del costo del viaggio, mentre per ritardi superiori o uguali ai 120 minuti il rimborso è pari al 50%.

Esercizio 4 – Un quesito a scelta (punti 3)

1) Si descrivano le proprietà e l'organizzazione di un nodo nei B-Tree.
2) Dato il seguente schema relazionale:

   R(DOC, UNI, INS)

   dove DOC sta per Docente, UNI Università e INS per Insegnamento. Viene richiesto di scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che a seguito di inserimento o modifica di una singola tupla, verifichi il rispetto delle seguenti dipendenze funzionali:

   (FD1) UNI, INS → DOC

   (FD2) DOC → UNI

Soluzione

Esercizio 1

Dato il seguente schema relazionale:

```
TRENO(CODT,TIPO,DESC)
STAZIONE(NOME,CITTA,NAZIONE)
VIAGGIO(CODV,CODT,PARTENZA,ARRIVO,COSTO,RITARDO)
FK: PARTENZA REFERENCES STAZIONE
FK: ARRIVO REFERENCES STAZIONE
FK: CODT REFERENCES TRENO
PRENOTAZIONE(CF,CODV)
FK: CODV REFERENCES VIAGGIO
(RITARDO è di tipo intero ed esprime i ritardi in minuti)
```

Scrivere in SQL le seguenti interrogazioni

1) Calcolare per ogni treno il rimborso medio sostenuto per i viaggi con ritardo, sapendo che per ritardi maggiori o uguali a 30 minuti ed inferiori a 120 minuti ogni passeggero ha diritto ad un rimborso pari al 30% del costo del viaggio, mentre per ritardi superiori o uguali ai 120 minuti il rimborso è pari al 50%.

```
Create view V1 as
Select T.CODT, V.CODV, (count(*) * V.COSTO*0,30) as RIMB_VIAGGIO
From TRENO T, VIAGGIO V, PRENOTAZIONE P
Where T.CODT = V.CODT
And P.CODV = V.CODV
And V.RITARDO >= 30
And V.RITARDO < 120
Group by T.CODT, V.CODV

Create view V2 as
Select T.CODT, V.CODV, (count(*) * V.COSTO*0,50) as RIMB_VIAGGIO
From TRENO T, VIAGGIO V, PRENOTAZIONE P
Where T.CODT = V.CODT
And P.CODV = V.CODV
And V.RITARDO >=120
Group by T.CODT, V.CODV
```

```
Select V1.CODT,
SUM(V1.RIMB_VIAGGIO)+SUM(V2.RIMB_VIAGGIO)/COUNT(*)
From V1, V2
Where V1.CODT = V2.CODT
GROUP BY V1.CODT
```

Soluzione Alternativa:

```
Create view V1 as
Select T.CODT, V.CODV, (count(*) * V.COSTO*0,30) as RIMB_VIAGGIO
From TRENO T, VIAGGIO V, PRENOTAZIONE P
Where T.CODT = V.CODT
And P.CODV = V.CODV
And V.RITARDO >= 30
And V.RITARDO < 120
Group by T.CODT, V.CODV
UNION
Select T.CODT, V.CODV, (count(*) * V.COSTO*0,50) as RIMB_VIAGGIO
From TRENO T, VIAGGIO V, PRENOTAZIONE P
Where T.CODT = V.CODT
And P.CODV = V.CODV
And V.RITARDO >=120
Group by T.CODT, V.CODV

Select V1.CODT, SUM(V1.RIMB_VIAGGIO)/COUNT(*)
From V1
GROUP BY V1.CODT
```

Scrivere in embedded SQL la seguente interrogazione

2) Mostrare, per ciascun treno, il costo medio dei viaggi internazionali, trascurando i 2 viaggi di costo minimo. In caso di treni con soli due viaggi mostrare come costo medio 0.

```
Declare C1 Cursor For
Select V.CODT, V.COSTO
From VIAGGIO V, STAZIONE S1, STAZIONE S2
Where V.PARTENZA = S1.NOME
And V.ARRIVO = S2.NOME
And S1.NAZIONE <> S2.NAZIONE
Order by V.CODT, V.COSTO DESC
open C1;
fetch C1 into :CODT, :COSTO;
while (SQLCODE == 0){
        numViaggi = 0;
        costoTot = 0;
        trenoAttuale == CODT;
        cont = 0;
        exec sql Select count(*) into : numViaggi
                From VIAGGIO V, STAZIONE S1, STAZIONE S2
                Where V.PARTENZA = S1.NOME
                And V.ARRIVO = S2.NOME
                And S1.NAZIONE <> S2.NAZIONE
                And V.CODT = :CODT
        while (trenoAttuale == CODT && SQLCODE == 0){
        /*il costo medio dei viaggi nazionali, trascurando i 2 viaggi di costo minimo*/
        if (cont < (num_viaggi -2)){
                cont = cont +1;
                costoTot = costoTot + COSTO;
        }
                fetch Q1 into :CODT, :COSTO;
        }
        If (cont==0)
                costoMedio = 0;
        else
                costoMedio = costoTot/( numViaggi -2);
```

```
        printf(“treno %s, costoMedio %d”, trenoAttuale, costoMedio);
}
close C1;
```

3) una pagina web che consente di inserire il nome di una nazione e che ne genera un’altra contenente per ogni tipo di treno, il treno che ha compiuto il maggior numero di viaggi nazionali, all’interno della nazione specificata, e il ritardo medio che ha accumulato.

Pagina form.html:

```
<html><head>
<title>Ricerca Treni per Nome Nazione</title>
</head>
<body bgcolor="white">
Inserire il nome della categoria:
<form action="find.jsp" method="get">
<!-- In alternativa:
<form action="find.aspx" method="get">-->
<table>
<tr><td>Nome Nazione:</td>
<td><input type="text" name=“nazione"></td></tr>
<tr><td colspan=2><input type="submit" value="Cerca"></td></tr>
</table>
</form>
</body></html>
```

Pagina find.jsp:

```
<html><head>
<title>Elenco Treni</title>
</head><body>
```

```
<%@ page language="java" import="java.sql.*" %>

Risultati della ricerca

<%
Connection conn = null;


//carica il file di classe del driver
Class.forName("org.postgresql.Driver");

//crea la connessione con l'origine dati
conn = DriverManager.getConnection("jdbc:postgresql://localhost/bollette",
            “postrgres"," postrgres");
String retrieve = " Select  T.TIPO, T.CODT, AVG(RITARDO) as RIT_MEDIO
        From    TRENO T, STAZIONE S1, STAZIONE S2, VIAGGIO V
        Where S1.NAZIONE = ?
        And     T.CODT = V.CODT
        And     V.PARTENZA = S1.NOME
        And     V.ARRIVO = S2.NOME
        And     S1.NAZIONE = S2.NAZIONE
        Group by T.TIPO, T.CODT
        Having count (*) >= ALL (   Select count(*)
                                    From TRENO T1, STAZIONE S11, STAZIONE
                                    S12, VIAGGIO V1
                                    Where T1.TIPO = T.TIPO
                                    And T1.CODT = V1.CODT
                                    And V1.PARTENZA= S11.NOME
                                    And V1.ARRIVO = S12.NOME
                                    And     S11.NAZIONE = S1.NAZIONE
                                    And     S11.NAZIONE = S12.NAZIONE
                                    Group by T1.CODT)”


PreparedStatement st = conn.prepareStatement(retrieve);
st.setString(1, Request.getParameter(“nazione”));
ResultSet result = st.executeQuery();
%>
```

```
<table>
<tr><td>Tipo:</td><td>Treno:</td>><td>Ritardo:</td></tr> <%
while (result.next()) {
        out.println("<tr><td>" + result.getString("TIPO") + "</td><td>" +
        result.getString("CODT")+ "</td ><td>" +    result.getString("RIT_MEDIO")+
        "</td ></tr>");
}
st.close();
conn.close();
%>
</table> </body> </html>
```

Esercizio 2 (punti 6)

Dato il seguente schema relazionale:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

(FD1) $AC \rightarrow B$

(FD2) $C \rightarrow A$

(FD3) $D \rightarrow C$

Viene richiesto di:

- Determinare la chiave o le chiavi dello schema di relazione;
- Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
- Produrre eventuali decomposizioni dello schema in BCNF che preservano i dati e discuterne la preservazione delle dipendenze funzionali.

  Per la soluzione non ci si può avvalere del teorema 7 sulla preservazione dei dati

La chiave dello schema è:

K1 = {D}

(FD1) $AC \rightarrow B$ viola la 3NF

(FD2) $C \rightarrow A$ viola la 3NF

(FD3) $D \rightarrow C$ nessuna violazione

Lo schema viola la 3NF.

Proviamo a decomporre a partire da FD1:

R1(A,B,C) preserva FD1 e FD2

R2(C,D) preserva FD3

Lo schema R1 non è in BCNF a causa della FD2 pertanto decomponiamo R1:

R11(A, C) preserva la FD2

R12(B,C)

R2(C,D) preserva la FD3

Tale decomposizione è loss-less poiché il join naturale è fatto su C, chiave di R11 e R12. Verifichiamo se preserva la dipendenza FD1:

$XPIUG^0(AC) = AC$
$XPIUG^1(AC) = AC \cup (XPIU(AC \cap AC, F) \cap AC) \cup (XPIU(AC \cap BC, F) \cap BC) \cup (XPIU(AC \cap CD, F) \cap CD) = AC \cup (ABC \cap AC) \cup (ABC \cap BC) \cup (ABC \cap CD) = AC \cup AC \cup AB \cup AC = ABC$

Lo schema è loss-less. La decomposizione preserva i dati, le dipendenze e lo schema risultante è in BCNF.

Esercizio 3

Dato la seguente porzione di schema relazionale:

```
VIAGGIO(CODV,COSTO,RITARDO)
PASSEGGERO(CF,NOME,RIMBORSO)
PRENOTAZIONE(CF,CODV)
FK: CF REFERENCES PASSEGGERO
FK: CODV REFERENCES VIAGGIO
```

Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che, a fronte di inserimenti, modifiche e cancellazioni nella relazione VIAGGIO mantenga aggiornati i rimborsi dei passeggeri considerando che per ritardi maggiori o uguali a 30 minuti ed inferiori a 120 minuti ogni passeggero ha diritto ad un rimborso pari al 30% del costo del viaggio, mentre per ritardi superiori o uguali ai 120 minuti il rimborso è pari al 50%.

Sintassi SQLServer:

```
CREATE TRIGGER Aggiorna_Rimborsi
ON VIAGGIO
FOR INSERT, UPDATE, DELETE
```

```
AS
IF UPDATE (RITARDO) OR UPDATE(COSTO) OR UPDATE(CODV)
BEGIN
Declare @codv char(10)
Declare @ritardo int
Declare @costo float


--- Dichiaro il cursore

DECLARE R_cursor CURSOR FOR
SELECT  I.CODV,  I.RITARDO, I.COSTO
FROM Inserted I

--- Apro e carico il cursore
OPEN R_cursor
FETCH NEXT FROM R_cursor INTO @codv, @ritardo, @costo

WHILE @@FETCH_STATUS = 0
BEGIN
IF(@ritardo>=120)
        BEGIN
                UPDATE PASSEGERO SET Rimborso = Rimborso + 0,50 * @costo
                WHERE CF IN (       SELECT P.CF
                                    FROM PASSEGGERO P, PRENOTAZIONE PR
                                    WHERE P.CF = PR.CF
                                    AND PR.CODV = @codv)
                FETCH NEXT FROM R_cursor INTO @codv, @ritardo, @costo
        END
IF(@ritardo<120 && @ritardo>=30)
        BEGIN
                UPDATE PASSEGERO SET Rimborso = Rimborso + 0,30 * @costo
                WHERE CF IN (       SELECT P.CF
                                    FROM PASSEGGERO P, PRENOTAZIONE PR
                                    WHERE P.CF = PR.CF
                                    AND PR.CODV = @codv)
                FETCH NEXT FROM R_cursor INTO @codv, @ritardo, @costo
        END

END
CLOSE R_cursor
DEALLOCATE R_cursor
--- Dichiaro il cursore

DECLARE R2_cursor CURSOR FOR
SELECT  D.CODV,  D.RITARDO, D.COSTO
```

```
FROM Deleted D

--- Apro e carico il cursore
OPEN R2_cursor
FETCH NEXT FROM R2_cursor INTO  @codv, @ritardo, @costo

WHILE @@FETCH_STATUS = 0
BEGIN
    IF(@ritardo<120 && @ritardo>=30)
        BEGIN
            UPDATE PASSEGERO SET Rimborso = Rimborso - 0,30 * @costo
            WHERE CF IN (       SELECT P.CF
                                FROM PASSEGGERO P, PRENOTAZIONE PR
                                WHERE P.CF = PR.CF
                                AND PR.CODV = @codv)
            FETCH NEXT FROM R_cursor INTO @codv, @ritardo, @costo
        END
    IF(@ritardo>=120)
        BEGIN
            UPDATE PASSEGERO SET Rimborso = Rimborso - 0,50 * @costo
            WHERE CF IN (       SELECT P.CF
                                FROM PASSEGGERO P, PRENOTAZIONE PR
                                WHERE P.CF = PR.CF
                                AND PR.CODV = @codv)
            FETCH NEXT FROM R_cursor INTO @codv, @ritardo, @costo
        END

END
CLOSE R2_cursor
DEALLOCATE R2_cursor
END
```

Esercizio 4 (3 punti)

1) Dato il seguente schema relazionale:

   R(DOC, UNI, INS)

   dove DOC sta per Docente, UNI Università e INS per Insegnamento. Viene richiesto di scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che a seguito di inserimento o modifica di una singola tupla, verifichi il rispetto delle seguenti dipendenze funzionali:

   (FD1) UNI, INS → DOC

   (FD2) DOC → UNI

```
CREATE TRIGGER Controllo_DF ON R
FOR INSERT, UPDATE AS
--Dichiaro delle variabili di comodo
Declare @cont int
--controllo UNI,INS --> DOC
--recupero i valori inseriti utilizzando la tabella temporanea inserted
Select @cont = count(distinct R.DOC)
from R, inserted
where R.UNI = inserted.UNI
and R.INS = inserted.INS

if (@cont>1)
begin
     raiserror('L'insegnamento che si vuole inserire per
     l'università scelta ha già un professore
     associato.',16,1);
     rollback transaction
end

--controllo DOC --> UNI

Select @cont = count(distinct R.UNI)
From R, inserted
Where R.DOC = inserted.DOC


if(@cont>1)
begin
     raiserror('Il Professore che si vuole inserire può
     appartenere ad una sola Università.',16,1)
     rollback transaction
end
```