PROVA SCRITTA DI TECNOLOGIA DATABASE – 05/12/2007

Corso di Laurea Specialistica in Ingegneria Informatica - NOD

SONIA BERGAMASCHI E MAURIZIO VINCINI

Esercizio 1 (punti 22)

Dato il seguente schema relazionale:

```
FORNITORE(CODF,Nome)
ARTICOLO(CODA,Nome,Tipo)
FORNITURA(ID,CODF,CODA,Data,Prezzo)
FK: CODF REFERENCES FORNITORE
    CODA REFERENCES ARTICOLO
```

Scrivere in SQL la seguente interrogazione

1) Mostrare, per ciascun fornitore, il tipo di articoli che ha fornito al prezzo medio più alto.

Scrivere in embedded SQL la seguente interrogazione

2) Selezionare, per ogni articolo, il nome del fornitore che ne ha venduto il maggior numero di forniture.
3) Scrivere in linguaggio jsp o asp.net una pagina web che consente di inserire il nome di un fornitore e che ne genera un'altra contenente l'elenco dei codici delle sue forniture ed il costo totale di ciascuna di esse.

Esercizio 2 (punti 5)

Dato il seguente schema relazionale:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

(FD1) $AB \rightarrow C$

(FD2) $B \rightarrow D$

(FD3) $D \rightarrow A$

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)
Data la seguente porzione di schema relazionale:

```
IMPIEGATO(CODI,Nome,Stipendio,CODR)
FK: CODR REFERENCES REPARTO
REPARTO(CODR,CODM,Nome)
FK: CODM REFERENCES IMPIEGATO
CODM identifica il Manager del REPARTO
```

Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che garantisca il seguente vincolo:

ogni Manager guadagna più di ogni impiegato che dirige

Soluzione

Esercizio 1

1) Mostrare, per ciascun fornitore, il tipo di articoli che ha fornito al prezzo medio più alto.

```
SELECT      CODF, Tipo
FROM        ARTICOLO A, FORNITURA F
WHERE       A.CODA = F.CODA
GROUP BY    CODF, Tipo
HAVING AVG(F.Prezzo) >= ALL (SELECT   AVG(F1.Prezzo)
                             FROM     ARTICOLO A1, FORNITURA F1
                             WHERE    A1.CODA = F1.CODA
                              AND     F1.CODF = F.CODF
                             GROUP BY Tipo)
```

2) Selezionare, per ogni articolo, il nome del fornitore che ne ha venduto il maggior numero di forniture.

```
Q1:   SELECT      F.CODA, FO.NOME
      FROM        FORNITURA F, FORNITORE FO
      WHERE       F.CODF = FO.CODF
      GROUP BY    F.CODA, FO.CODF, FO.NOME
      HAVING COUNT(*) >= ALL (SELECT     COUNT(*)
                              FROM       FORNITURA
                              WHERE      CODA = F.CODA
                              GROUP BY CODF)

Declare Cursor "C1" For  Q1
open C1;
fetch C1 into :CODA, :NOME;
while (SQLCODE == 0){
```

```
        printf("CODF  %s, NOME%f\n", CODF, NOME);
        fetch C1 into :CODA, :NOME;
}
close C1;
```

3) Scrivere in linguaggio jsp o asp.net una pagina web che consente di inserire il nome di un fornitore e che ne genera un'altra contenente l'elenco dei codici delle sue forniture ed il costo totale di ciascuna di esse.

Pagina form.jsp:

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

Pagina find.jsp:

```
<html><head>
<title>Elenco forniture</title>
</head><body>
<%@ page language="java" import="java.sql.*" %>
```

Risultati della ricerca

```
<%
Connection conn = null;

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

//crea la connessione con l'origine dati
conn = DriverManager.getConnection"jdbc:postgresql://localhost/azienda",
                “postrgres"," postrgres");
//crea lo statement
Statement st = conn.createStatement();
String retrieve = "SELECT F.ID, FO.Prezzo AS COSTO
                        FROM FORNITURA F, FORNITORE FO
                        WHERE F.CODF = FO.CODF AND F.NOME=?";
st = conn.prepareStatement(retrieve);
st.setString(1, Request.getParameter(“nome”));
ResultSet result = statement.executeQuery();
%>
<table>
<tr><td>ID:</td><td>Costo Totale:</td></tr>
<%
while (result.next()) {
        out.println(“<tr><td>” + result.getString("ID") + “</td><td>” +
        result.getString("COSTO")+ “</td></tr>”
}
st.close();
conn.close();
%>
</table> </body> </html>
```

Pagina find.aspx:

```
<%@ Page Language="VB" %>
```

```
<html>
<head runat="server">
<title>Elenco forniture</title>
</head>
<body>
<form id="form1" runat="server">
<asp:GridView ID="GridView1" Runat="server" DataSourceID="SqlDataSource1"
AutoGenerateColumns="False" HeaderText="Elenco forniture" AllowPaging="True">
 <Columns>
 <asp:BoundField HeaderText="ID Fornitura" DataField="ID" SortExpression="ID" />
<asp:BoundField HeaderText="Costo Totale" DataField="COSTO"
                SortExpression="COSTO" />
</ Columns>
</ asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"
      SelectCommand="SELECT F.ID, FO.Prezzo AS COSTO
                           FROM FORNITURA F, FORNITORE FO
                           WHERE F.CODF = FO.CODF AND F.NOME=@fnome"
ConnectionString="<%$ ConnectionStrings:azienda %>">
<SelectParameters>
<asp:QueryStringParameter Name="fnome"
                           QueryStringField="nome" />
</asp:SqlDataSource>
</form>
</body>
</html>
```

Esercizio 2

Dato il seguente schema relazionale:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

(FD1) $AB \rightarrow C$

(FD2) $B \rightarrow D$

(FD3) $D \rightarrow A$

La chiave dello schema è:

K = B

(FD1) $AB \rightarrow C$ è in 3NF

(FD2) $B \rightarrow D$ è in BCNF

(FD3) $D \rightarrow A$ è in 2NF

Lo schema è pertanto in 2NF.

Decomposizione:

R1(A, <u>D</u>) preserva FD3 (proiettata)

R2(<u>B</u>,C,D) preserva FD2 (proiettata)

Lo schema è loss-less poiché il join naturale è fatto su D, chiave di R1.

Verifica della preservazione di FD1:

$XPIUG^0(AB) = AB$

$XPIUG^1(AB) = AB \cup (XPIU(AB \cap AD, F) \cap AD) \cup (XPIU(AB \cap BCD, F) \cap BCD)$

$= AB \cup A \cup BCD = ABCD$

$XPIUG^2(ABC) = ABC \cup (XPIU(ABC \cap AD, F) \cap AD)$

$\cup (XPIU(ABC \cap BCD, F) \cap BCD)$

$= ABC \cup A \cup BCD = ABCD.$

FD1 e` preservata.

Lo schema risulta in BCNF.

Esercizio 3
Data la seguente porzione di schema relazionale:

```
IMPIEGATO(CODI,Nome,Stipendio,CODR)
FK: CODR REFERENCES REPARTO
REPARTO(CODR,CODM,Nome)
FK: CODM REFERENCES IMPIEGATO
CODM identifica il Manager del REPARTO
```

Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che garantisca il seguente vincolo:

ogni Manager guadagna più di ogni impiegato che dirige

```
CREATE TRIGGER Controllo_Stipendio_Reparto
ON REPARTO
FOR INSERT, UPDATE
AS
IF UPDATE(CODM)
BEGIN
Declare @codr char(10)
Declare @codm char(10)
Declare @stip float

--- Dichiaro il cursore
DECLARE R_cursor CURSOR FOR
SELECT  I.CODR, CODM, Stipendio
FROM Inserted I, IMPIEGATO IM
WHERE I.CODM = IM.CODI

--- Apro e carico il cursore
OPEN R_cursor
FETCH NEXT FROM R_cursor INTO @codr, @codm, @stip
```

```
WHILE @@FETCH_STATUS = 0
BEGIN
    IF @stip< (SELECT MAX(Stipendio)
                    FROM IMPIEGATO
                    WHERE CODR=@codr AND CODI <> @codm )
          BEGIN
             raiserror('Il Manager %s guadagna troppo poco!',16,1, @codm)
              rollback transaction
          END
          ELSE
               PRINT 'Il Manager ' +  STR(@codm)  +  ' guadagna abbastanza'

     FETCH NEXT FROM R_cursor INTO @codr, @codm, @stip
END
CLOSE R_cursor
DEALLOCATE R_cursor
END



CREATE TRIGGER Controllo_Stipendio_Impiegato
ON IMPIEGATO
FOR INSERT, UPDATE
AS
IF UPDATE(CODR) OR UPDATE(Stipendio)
BEGIN
Declare @codr char(10)
Declare @codi char(10)
Declare @stip float

--- Dichiaro il cursore
DECLARE R_cursor CURSOR FOR
SELECT  CODR, CODI, Stipendio
FROM Inserted

--- Apro e carico il cursore
OPEN R_cursor
FETCH NEXT FROM R_cursor INTO @codr, @codi, @stip
```

```
WHILE @@FETCH_STATUS = 0
BEGIN
    IF (
          @stip>= (SELECT I.Stipendio
                FROM REPARTO R, IMPIEGATO I
                WHERE I.CODI = R.CODM AND R.CODR=@codr
                 AND CODI <> @codi )
      OR  ( @codi = (SELECT CODM
              FROM REPARTO
              WHERE CODR = @codr)
         AND
         @stip<= (SELECT MAX(Stipendio)
              FROM IMPIEGATO
              WHERE CODR=@codr AND CODI <> @codi )
          )
         )
           BEGIN
            raiserror('Il dipendente %s non puo'' subire questa modifica!',16,1, @codi)
             rollback transaction
         END
         ELSE
              PRINT 'Il dipendente ' +  STR(@codi)  + ' guadagna uno stipendio coerente
                    alle regole'

    FETCH NEXT FROM R_cursor INTO @codr, @codi, @stip
END
CLOSE R_cursor
DEALLOCATE R_cursor
END
```