PROVA SCRITTA DI TECNOLOGIA DATABASE – 07/02/2012

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:

```
LIBRERIA(CODL,NOME,CITTA)
LIBRO(CODB,NOME,PREZZO,CATEGORIA)
ORDINE(CODL,CODB,DATA,QTY)
FK: CODL REFERENCES LIBRERIA
FK: CODB REFERENCES LIBRO
```

Scrivere in SQL le seguenti interrogazioni

1) Calcolare per ciascun libro, la città (o le città) in cui il costo medio degli ordini è massimo, trascurando gli ordini per cui la quantità ordinata è inferiore a 2;

Scrivere in embedded SQL la seguente interrogazione

2) Calcolare, per ciascuna città, il numero medio degli ordini effettuati nell'anno 2011 dalle librerie, e il costo medio totale che esse hanno sostenuto;

Scrivere in linguaggio jsp o asp.net

3) una pagina web che consente di inserire il nome di un libro e di una città e che ne genera un'altra contenente i nomi delle librerie della città specificata che hanno ordinato quel libro in ordine decrescente di quantità totale ordinata.

Esercizio 2 (punti 6)

Dato il seguente schema relazionale:

R(A,B,C,D,E)

e considerando le seguenti dipendenze funzionali:

(FD1) $A \rightarrow B$

(FD2) $C \rightarrow D$

(FD3) $BD \rightarrow C$

(FD4) $AD \rightarrow E$

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:

```
AGENTE(CODA,Provvigioni)
POLIZZA(CODP,Premio,Importo)
STIPULA(CODA,CF,CODP)
FK: CODA REFERENCES AGENTE
FK: CODP REFERENCES POLIZZA
```

Per ciascuna polizza un agente incassa il 10% di provvigione sull'importo.
Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che, a fronte di inserimenti, modifiche e cancellazioni nella relazione STIPULA mantenga aggiornate le provvigioni degli agenti.
.

Esercizio 4 – Un quesito a scelta (punti 3)

1) Si descrivano l'enunciato e le finalità della Formula di Cardenas
2) Dato il seguente schema relazionale:

   CARTA_DI_CREDITO(CODC, CF, Cell)

   viene richiesto di scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che verifichi che ad ogni carta di credito possano essere associati al massimo due numeri di cellulare.

Soluzione

Esercizio 1

M Dato il seguente schema relazionale:

```
LIBRERIA(CODL,NOME,CITTA)
LIBRO(CODB,NOME,PREZZO,CATEGORIA)
ORDINE(CODL,CODB,DATA,QTY)
FK: CODL REFERENCES LIBRERIA
FK: CODB REFERENCES LIBRO
```

Scrivere in SQL le seguenti interrogazioni

1) Calcolare per ciascun libro, la città (o le città) in cui il costo medio degli ordini di quel libro è massimo, trascurando gli ordini con per cui la quantità è inferiore a 2;

```
SELECT      B.CODB, L.CITTA
FROM        LIBRO B, LIBRERIA L, ORDINE O
WHERE       O.CODL = L.CODL
AND         O.CODB = B.CODB
AND         O.QTY > =2
GROUP BY    B.CODB, L.CITTA
HAVING AVG(QTY*PREZZO) >= ALL (SELECT AVG(QTY*PREZZO)
                               FROM  LIBRO B2, ORDINE O2,
                                     LIBRERIA L2
                               WHERE  O2.CODB = O.CODB
                               AND    B2.CODB = O2.CODB
                               AND    L2.CODL = O2.CODL
                               AND    O2.QTY > =2
                               GROUP BY CITTA)
```

2) Calcolare, per ciascuna città, il numero medio degli ordini effettuati dalle librerie nell'anno 2011, e il costo medio totale che esse hanno sostenuto;

Q1:

```
CREATE      VIEW V1 AS
```

```
SELECT      L.CITTA, L.CODL, COUNT(*) AS N_ORD_LIB,
SUM(QTY*PREZZO) AS COSTO_LIB
FROM        LIBRERIA L, ORDINE O, LIBRO B
WHERE       L.CODL = O.CODL
AND         B.CODB = O.CODB
AND         O.DATA BETWEEN '01/01/2011' AND '31/12/2011'
GROUP BY    L.CITTA, L.CODL


Q2: SELECT  CITTA, AVG(N_ORD_LIB) AS NUM_MEDIO_ORD,
            AVG(COSTO_LIB) AS COSTO_MEDIO
    FROM    V1
    GROUP BY CITTA


Exec sql Q1;
Declare "C1" Cursor For  Q2
open C1;
fetch C1 into : CITTA, : NUM_MEDIO_ORD, :COSTO_MEDIO;
while (SQLCODE == 0){
        printf("Città %s, Numero Ordini  %s, Costo Medio  %F,
        CITTA, NUM_MEDIO_ORD, COSTO_MEDIO);
        fetch C1 into: CITTA, : NUM_MEDIO_ORD, :COSTO_MEDIO;
}
close C1;
```

3) una pagina web che consente di inserire il nome di un libro e di una città e che ne genera un'altra contenente i nomi delle librerie della città specificata che hanno ordinato quel libro in ordine decrescente di quantità totale ordinata.

Pagina form.html:

```
<html><head>
<title>Ricerca Libri per Nome e Città</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 Libro:</td>
<td><input type="text" name=“libro"></td></tr>
<tr><td>Nome Città:</td>
<td><input type="text" name=“città"></td></tr>
<tr><td colspan=2><input type="submit" value="Cerca"></td></tr>
</table>
</form>
</body></html>
```

Pagina find.jsp:

```
<html><head>
<title>Elenco Libreria</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 L.NOME, SUM(QTY) AS QTY
FROM ORDINE O, LIBRERIA L, LIBRO B
```

```
WHERE L.CITTA = ?
AND B.NOME = ?
AND O.CODB = B.CODB
AND O.CODL = L.CODL
GROUP BY O.CODL, L.NOME
ORDER BY 2 DESC”

PreparedStatement st = conn.prepareStatement(retrieve);
st.setString(1, Request.getParameter(“citta”));
st.setString(2, Request.getParameter(“libro”));
ResultSet result = st.executeQuery();
%>
<table>
<tr><td>Libreria:</td><td>Quantità:</td></tr> <%
while (result.next()) {
        out.println(“<tr><td>” + result.getString("NOME") + “</td><td>” +
        result.getString(“QTY”)+ “</td ></tr>”);
}
st.close();
conn.close();
%>
</table> </body> </html>
```

Esercizio 2 (punti 6)

Dato il seguente schema relazionale:

R(A,B,C,D,E)

e considerando le seguenti dipendenze funzionali:

(FD1) $A \rightarrow B$

(FD2) $C \rightarrow D$

(FD3) $BD \rightarrow C$

(FD4) $AD \rightarrow E$

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

Le chiavi dello schema sono:

K1 = AC

K2 = AD

(FD1) $A \rightarrow B$ è in 1NF

(FD2) $C \rightarrow D$ è in 3NF

(FD3) $BD \rightarrow C$ è in 3NF

(FD4) $AD \rightarrow E$ è in BCNF

Lo schema non è in 2NF a causa di FD1. Lo schema non è in BCNF sia a causa di FD2 che di FD3.

Poichè lo schema non è in 2NF a causa di FD1 decompongo R a partire da tale violazione:

R1(A,B) preserva FD1 (proiettata)

R2(A,C,D,E) preserva FD2 e FD3 (proiettate)

Lo schema è loss-less poiché il join naturale è fatto su A, chiave di R1. Lo schema R2 non è in BCNF a causa di FD2, pertanto decomponiamo R2:

R21(C,D) preserva FD2 (proiettata)

R22(A,C,E)

Tale decomposizione è loss-less verifichiamo se preserva le dipendenze FD3 e FD4.

Verifica di FD3:

$XPIUG^0(BD) = BD$

$XPIUG^1(BD) = BD \cup (XPIU(BD \cap AB, F) \cap AB) \cup (XPIU(BD \cap CD, F) \cap CD) \cup (XPIU(BD \cap ACE, F) \cap ACE)$

$= BD \cup B \cup D \cup 0 = BD$

FD3 non e` preservata.

Verifica di FD4:

$XPIUG^0(AD) = AD$

$XPIUG^1(AD) = AD \cup (XPIU(AD \cap AB, F) \cap AB) \cup (XPIU(AD \cap CD, F) \cap CD) \cup (XPIU(AD \cap ACE, F) \cap ACE) = AD \cup AB \cup D \cup A = ABD$

$XPIUG^2(ABC) = ABD \cup (XPIU(ABD \cap AB, F) \cap AB) \cup (XPIU(ABD \cap CD, F) \cap CD) \cup (XPIU(ABD \cap ACE, F) \cap ACE)$

$= ABD \cup AB \cup D \cup A = ABD$

FD4 non e` preservata.

Tale decomposizione è lossless ma non preserva la FD4 e la FD3

Lo schema è in BCNF.

Esercizio 3

Dato la seguente porzione di schema relazionale:

```
AGENTE(CODA,Provvigioni)
POLIZZA(CODP,Premio,Importo)
STIPULA(CODA,CF,CODP)
FK: CODA REFERENCES AGENTE
FK: CF REFERENCES CLIENTE
```

Per ciascuna polizza un agente incassa il 10% di provvigione sull'importo.
Scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che, a fronte di inserimenti, modifiche e cancellazioni nella relazione STIPULA mantenga aggiornate le provvigioni degli agenti.

Sintassi SQLServer:

```
CREATE TRIGGER Aggiorna_Polizza
ON STIPULA
FOR INSERT, UPDATE, DELETE
AS
IF UPDATE(CODA) or UPDATE (CODP)
```

```
BEGIN
Declare @coda char(10)
Declare @importo float

--- Dichiaro il cursore
DECLARE R_cursor CURSOR FOR
SELECT  CODA, Importo
FROM Inserted I, POLIZZA P
WHERE I.CODP = P.CODP

--- Apro e carico il cursore
OPEN R_cursor
FETCH NEXT FROM R_cursor INTO @coda, @importo

WHILE @@FETCH_STATUS = 0
BEGIN
     UPDATE AGENTE SET Provvigioni = Provvigioni +  0,10 * @importo
     WHERE CODA=@coda
     FETCH NEXT FROM R_cursor INTO @coda, @importo
END
CLOSE R_cursor
DEALLOCATE R_cursor
--- Dichiaro il cursore

DECLARE R_cursor CURSOR FOR
SELECT  CODA, Importo
FROM Deleted
--- Apro e carico il cursore
OPEN R2_cursor
FETCH NEXT FROM R2_cursor INTO @coda, @importo

WHILE @@FETCH_STATUS = 0
BEGIN
     UPDATE AGENTE SET Provvigioni = Provvigioni -  0,10 * @importo
      WHERE CODA=@coda
     FETCH NEXT FROM R2_cursor INTO @coda, @importo
END
CLOSE R2_cursor
DEALLOCATE R2_cursor
END
```

Sintassi DB2:

```
CREATE TRIGGER AGGIORNA_PROVVIGIONI
AFTER UPDATE ON STIPULA
```

```
REFERENCING NEW AS N OLD AS O
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
UPDATE AGENTE
SET PROVVIGIONI= PROVVIGIONI + 0,1 *(      SELECT IMPORTO
                                           FROM POLIZZA
                                           WHERE N.CODP=P.CODP)
WHERE N.CODA = AGENTE.CODA)


UPDATE AGENTE
SET PROVVIGIONI= PROVVIGIONI - 0,1 *(      SELECT IMPORTO
                                           FROM POLIZZA
                                           WHERE O.CODP=P.CODP)
WHERE O.CODA = AGENTE.CODA)


END

CREATE TRIGGER AGGIORNA_PROVVIGIONI
AFTER INSERT ON STIPULA
REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
UPDATE AGENTE
SET PROVVIGIONI= PROVVIGIONI + 0,1 *(      SELECT IMPORTO
                                           FROM POLIZZA
                                           WHERE N.CODP=P.CODP)
WHERE N.CODA = AGENTE.CODA)


END


CREATE TRIGGER AGGIORNA_PROVVIGIONI
AFTER DELETE ON STIPULA
REFERENCING OLD AS O
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
UPDATE AGENTE
SET PROVVIGIONI= PROVVIGIONI - 0,1 *(      SELECT IMPORTO
                                           FROM POLIZZA
                                           WHERE O.CODP=P.CODP)
WHERE O.CODA = AGENTE.CODA)
```

END

Esercizio 4 (3 punti)

3) Dato il seguente schema relazionale:

   CARTA_DI_CREDITO(CODC, CF, Cell)

   viene richiesto di scrivere il Trigger (secondo la sintassi IBM DB2, MS SQLServer o ORACLE) che verifichi che ad ogni carta di credito possano essere associati al massimo due numeri di cellulare

Nel caso di inserimenti o modifiche di un singolo record :

```
-- Inizio Trigger
CREATE TRIGGER Controllo_Numero_Carte
ON CARTA_DI_CREDITO
FOR INSERT, UPDATE
AS
--- Dichiaro delle variabili di comodo
Declare @cont int
Select  @cont = count(Distinct Cell)
from    Carta_di_Credito, Inserted
where   Carta_di_Credito.CODC = inserted.CODC

if @cont > 2
begin
        raiserror('Una carta di credito può essere associata al massimo a due numeri di cellulare.',16,1)
        rollback transaction
end
```

Oppure nel caso di inserimenti o modifiche di più record:

```
CREATE TRIGGER Controllo_Numero_Carte
ON CARTA_DI_CREDITO
FOR INSERT, UPDATE
AS
Begin
--- Dichiaro delle variabili di comodo
Declare @cont int
Declare C Cursor For
Select  count(Distinct Cell)
from    Carta_di_Credito
```

```
group by CODC

Open C
Fetch next from C into @cont
While @@FETCH_STATUS = 0
Begin
If (@cont>2)
begin
        raiserror('Una carta di credito può essere associata al massimo a due numeri di
cellulare.',16,1)
        rollback transaction
end
Fetch next from C into @cont
End
Close C
Deallocate C
End
```

Entrambe le soluzioni sono state considerate esatte.