Capitolo 5

Esercizio 5.1

Realizzare una procedura in un linguaggio di programmazione di alto livello che tramite SQL Embedded elimina dalla tabella DIPARTIMENTO l'elemento che ha il nome che viene fornito come parametro alla procedura.

Soluzione:

Soluzione in C:

```
#include<stdlib.h>
main()
{
char Nome1;
char Nome2;
begin declare section;
         char Nome;
exec sql end declare section;

exec sql declare DipCursore for
         select Nome
         from Dipartimento;
exec sql open DipCursore;

do {
exec sql fetch DipCursore into
         :Nome1;
if (Nome1 == Nome2)
         exec sql delete from Dipartimento
         where nome = :Nome1;
}while (sqlca.sqlcode==0)
}
```

Esercizio 5.2

Realizzare un programma in un linguaggio di programmazione di alto livello che tramite SQL Embedded costruisce una videata in cui si presentano le caratteristiche di ogni dipartimento seguito dall'elenco degli impiegati che lavorano nel dipartimento, ordinati per cognome

Soluzione:

```
#include<stdlib.h>
main()
{

exec sql begin declare section;
        char Nome[20], Città[20], CognomeImpiegato[20],
                NomeImpiegato[20];
        int NumeroDip;
exec sql end declare section;

exec sql declare DipCursore cursor for
        select Nome, Citta, NumDip
        from DIPARTIMENTO;

exec sql declare ImpCursore cursor for
        select CognomeImpiegato, NomeImpiegato
        from Impiegato join Dipartimento on
        Impiegato.Dipartimento=Dipartimento.:Nome
        order by CognomeImpiegato;
exec sql open DipCursore;
do
{
exec sql fetch DipCursore into
        :Nome, :Città, :NumeroDip;
printf("Dipartimento: ", Nome, " situato in: ",Città,
        " Numero Dipendenti: ",NumeroDip." "Dipendenti:");
        exec sql open ImpCursore;
        do
        {
        exec sql fetch ImpCursore into
                :CognomeImpiegato, :NomeImpiegato;
        printf( CognomeImpiegato," ", NomeImpiegato);
        }while(sqlca.sqlcode==0);
        exec sql close cursor ImpCursore;
}while(sqlca.sqlcode==0);
exec sql close cursor DipCursore;
}
```

Esercizio 5.3

Realizzare l'esercizio precedente usando ADO

Soluzione:

```
#include<stdlib.h>
main()
{

conn ADODB.Connection;
impiegati ADODB.Recordset;
dipartimenti ADODB.Recordset;
comandoSQL ADODB.Command;
buffer string;

conn = new ADODB.connection;
conn.open("Server","Utente","Password");

dipartimenti = New ADODB.Recordset;
buffer="select Nome, Citta, NumDip from DIPARTIMENTO";
comandoSQL.CommandText = buffer;
dipartimenti.Open ComandoSQL(conn);




do{
        printf("Dipartimento: ", dipartimenti!Nome, " situato in:
        ",dipartimenti.Città," Numero Dipendenti:
        ",dipartimenti.NumeroDip." "Dipendenti:");

do
{
        impiegati = New ADODB.Recordset;
        buffer = "select CognomeImpiegato, NomeImpiegato
                from Impiegato join Dipartimento on
                Impiegato.Dipartimento=Dipartimento.",
                dipartimenti!Nome,
                "order by CognomeImpiegato";
        comandoSQL.CommandText = buffer;
        impiegati.Open ComandoSQL(conn);

        printf( impiegati.CognomeImpiegato," ",
                impiegati.NomeImpiegato);
        impiegati.movenext;
        }while(impiegati.EOF);
dipartimenti.movenext;
}while(dipartimenti.EOF);
}
```

Esercizio 5.4

Realizzare un programma java che scandisce gli impiegati ordinati per cognome e inserisce ogni impiegato che si trova in una posizione che è un multiplo di 10 in una tabella IMPIEGATIESTRATTI

Soluzione:

```
import java.sql.*;
public class ImpiegatiEstratti {
public static void main(String[] arg){

connection conn = null;
try{
        // carica driver e inizializza connessione
        Class.forName("sun.jdbc.odbc.jbdcObdcDriver");
        conn = DriverManager.GettConnection("jdbc:odbc:impiegati)
        }
try{
        Statement interrogazione = conn.createStatement();
        ResultSet risultato = interrogazione.executeQuery(
                "select *
                 from IMPIEGATI"




        Statement interrogazione1 = conn.createStatement();
        ResultSet risultato1 = interrogazione1.executeQuery(
                "create table IMPIEGATIESTRATTI
                        (
                        NomeImpiegato char(20),
                        CognomeImpiegato char(20),
                        dipartimento char(20),
                        stipedio integer,
                        primary key(NomeImpiegato,CognomeImpiegato)
                        )
        i=0;
        While(risultato.next()) {

        if(i=10)
        {
        string NomeImpiegato = risultato.getString("NomeImpiegato");
        string CognomeImpiegato =          risultato.getString
("CognomeImpiegato");
        string Dipartimento= risultato.getString("Dipartimento");
        int Stipendio = risultato.getString("Stipendio");
```

```
        Statement interrogazione2 = conn.createStatement();
        ResultSet risultato2 = interrogazione2.executeQuery(
                "insert into IMPIEGATIESTRATTI
                        values(" NomeImpiegato","
                        CognomeImpiegato","
                        dipartimento","
                        stipedio",
                        )";
        i=0;
}}}}
```

Esercizio 5.5

Realizzare un programma che accede al contenuto di una tabella

```
Capitolo(Numero, Titolo, Lunghezza)
```

che descrive i capitoli di un libro, con il numero e la dimensione delle pagine. Il programma quindi popola una tabella

```
Indice(Numero, Titolo, NumPagine)
```

in cui si presenta il numero di pagina nel quale inizia il capitolo, supponendo che il capitolo 1 inizia sulla prima pagina e che i capitoli devono iniziare su pagine dispari (eventualmente introducendo una pagina bianca alla fine del capitolo)

Soluzione:

```
#include<stdlib.h>
main()
{
exec sql begin declare section;
        char Titolo[50];
        int Numero, Lunghezza;
exec sql end declare section;

exec sql declare CapCursore cursor for
        select Numero, Titolo, Lunghezza
        from Capitolo;
exec sql open CapCursore;
exec sql create table Indice
        (
        Numero integer primary key,
        Titolo char(50),
        NumPagine integer
        );
do{
exec sql fetch CapCursore into
        :Numero, :Titolo, :Lunghezza;
if (Lunghezza%2 != 0)
        Lunghezza = Lunghezza + 1; // Pagina Bianca
exec sql insert into Indice
        values(:Numero,:Titolo,:Lunghezza)";
}while(sqlca.sqlcode == 0)

exec sql close cursor CapCursore;
}
```

Esercizio 5.6

Si supponga di avere le tabelle:

Magazzino(Prodotto, QtaDisp, Soglia, QtaRiordino)
OrdineInCorso(Prodotto, Qta)

Scrivere una procedura SQL che realizza il prelievo dal magazzino accettando 2 parametri, il prodotto `prod` e la quantità da prelevare `QtaPrelievo`. La Procedura deve verificare inizialmente che `QtaPrelievo` sia inferiore al valore di `QtaDisp` per il prodotto indicato. `QtaPrelievo` viene quindi sottratta al valore di `QtaDisp`. A questo punto la procedura verifica se per il prodotto `QtaDisp` risulta minore di `Soglia`, senza che in `OrdineInCorso` compaia già una tupla relativa al prodotto prelevato; se si, viene inserito un nuovo elemento nella tabella `OrdineInCorso`. Con i valori di `Prod` e del corrispondente attributo `QtaRiordino`.

Soluzione:

```
#include<stdlib.h>
main()
{
exec sql begin declare section;
        char Prodotto, prod;
        int QtaDisp, Soglia, QtaRiordino, Qta, Qta1, i;
exec sql end declare section;

exec sql declare MagazzinoCursore cursor for
        select  Prodotto, QtaDisp, Soglia, QtaRiordino
        from Magazzino;
exec sql open MagazzinoCursore;

prod = sceltaProdotto();
Qta= sceltaQta();
i=0;

do{
exec sql fetch MagazzinoCursore into
        :Prodotto, :QtaDisp, :Soglia, :QtaRiordino;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
        printf("ERRORE – Prodotto non trovato");
        exit(1);
        }
if (QtaDisp < Qta){
        printf("ERRORE – Quantità non disponibile");
        exit(1);
        }
QtaDisp = QtaDisp – Qta;

if (QtaDisp < Soglia){
        Qtariordino = QtaRiordino + QtaDisp – Soglia;
        }
```

```
exec sql update magazzino
         set QtaDisp = :QtaDisp
         set QtaRiordino = :QtaRiordino
         where current of MagazzinoCursore;

exec sql declare OrdineCursore cursor for
         select  Prodotto, Qta
         from OrdineInCorso;
exec sql open OrdineCursore ;

i=0;
do{
exec sql fetch  OrdineCursore into
         :Prodotto, :Qta1;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
         exec sql update OrdineInCorso
                  set Qta = :QtaRiordino;
                  where current of OrdineCursore;
         }
else exec sql insert into OrdineInCorso
         values(:prod,:QtaRiordino)";

}
```