GeoTema SQL Database Rapport

Af Frederik Foss Nielsen

## Indholdsfortegnelse

[Indholdsfortegnelse](#_n52bbqe3yhsq) 2

[**Database og Server 2012**](#_6r0fm3hcaf2f) **4**

[Opgavebeskrivelse](#_kaayf6qv3k27) 4

[Installation af Windows Server](#_epsxts5qix4c) 4

[Installation af MS SQL](#_lg4eig0j2wz) 4

[UML](#_8ags0shmdmj2) 5

[Tasks](#_gqixu8rmq0vw) 5

[**C# Program**](#_j1sy6assuts6) **9**

[Opgavebeskrivelse](#_rmn5fx46skix) 9

[Tidsplan](#_ncwpxz15mdk1) 9

[Kode forklaring](#_hb3jfb72f90g) 9

[Login](#_kdqlvgeli5zw) 9

[Graf](#_lm4mc0uu00t5) 9

[Redigere](#_b15phge2chrh) 10

[Søge](#_p42k7cqvm14m) 10

[Slette](#_2hwej3tjuyvv) 10

[Oprettelse](#_x5073x8f0kkj) 10

[Bruger redigering, oprettelse og sletning](#_bt7j1drm6nl5) 10

[Logout](#_jjuv1s4799yj) 10

[Program guide](#_frrds0hmbu4k) 11

[Login](#_9n5w871qhagd) 11

[Redigere](#_sm8lk2tq8ofg) 11

[Søge](#_geejpwkyx6pt) 11

[Slette](#_m87skzrrhb0) 11

[Oprettelse](#_vf1aa38g15uq) 11

[Bruger redigering, oprettelse og sletning](#_e0myq38f65sr) 11

[Graf](#_32deibmnm79s) 11

[**Bilag**](#_pgmqus3kfzlt) **12**

[Database og Server 2012](#_l6r5t51oc2i) 12

[SQL Kode](#_3szvz7ecn6bh) 12

[C# program kode](#_jp09a16bywzk) 14

[Index.cs(startskærm)](#_lichsdnoh649) 14

[Program.cs](#_ugvht0wwru1c) 19

[deleteform.cs(sletning af rækker)](#_u7qoqyqzallc) 30

[editform.cs(redigering af tabeller)](#_4dpyw6ojq5qy) 31

[User\_edit.cs(Redigering, oprettelse, og sletning af brugere)](#_7o6sdhoszauc) 34

[New Form.cs(opretning af rækker)](#_v51cjf4djsx8) 38

[Graph.cs](#_1jrl5rjwbt6s) 40

[C# program billeder](#_utbs76pavfba) 42

[Index](#_i1q5pvna59g7) 42

[New Form.cs(Ny række i tabel)](#_fhd5xwbp9kx9) 42

[editform.cs(redigering af tabeller)](#_17s9ba2jorwh) 43

[User\_edit.cs(Redigering, oprettelse, og sletning af brugere)](#_27fg4brczo64) 43

[deleteform.cs](#_5ei07ps7p78l) 44

[Graph.cs](#_sf7uisjzdt4u) 44

## 

## 

## 

# 

# 

# 

# 

# 

# Database og Server 2012

## Opgavebeskrivelse

Denne øvelse er baseret på at installere virtual maskine, hvor der skal installeres en sql server. Sql serveren skal have en database, i denne øvelse kommer man igennem kolonner, værdier foreign key, primary key, SQL Joins og normalisering.

## 

## Installation af Windows Server

For at installere Windows Server skal man lave en virtuel pc på Hyper-V hvorefter man linker Iso filen til den virtuelle så den loader windows når man starter den virtuelle, derefter følger du vejledningerne som Windows giver og til sidst er du ved login skærmen. Vi skal nu til at installere de services vi skal bruge for at have en Sql server. Hvis man klikker på add features knappen kommer der et vindue op hvor man kan vælge nogen services og pakker man kan installere. I det her tilfælde vil vi gerne installere Net Framework 3.5 R1 da det er en ting man skal bruge for at køre MS SQL, når installationen er færdig er det en god ide at genstarte serveren.

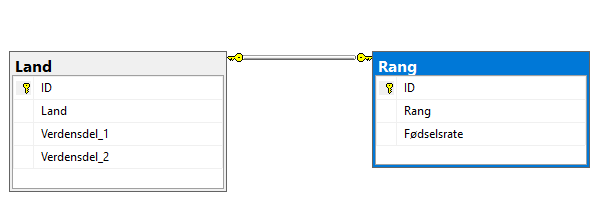
## 

## Installation af MS SQL

Nu skal vi installere SQL, så vi starter med at mounte iso filen som vi har på computeren så vi kan se den i den virtuelle. Når det er gjort burde den gerne komme op som en cd, vi skal køre den setup som er på disken. Når vinduet kommer frem skal vi klikke på installation og derefter på den knap hvor der står install SQL. når du når til det punkt hvor den spørg dig om de ting der skal installeres vælger du bare dem alle, når installationen er færdig burde din SQL server virke.

## UML

Her er et UML som viser databasens tables, grunden til at “Rang” tabellen har en primary key er så man kunne lave et FULL TEXT INDEX der gjorde det muligt at søge efter dele af ordet og ikke var nød til at skrive hele ordet.



## 

## Tasks

Her er de tasks jeg fik som opgave at løse

* Task 1
  + Installation af windows server på hyper-v
  + Installation af framework 3.5 R1
  + Installation af ms sql 2014
  + Oprette en bruger på ms sql 2014
    - Brugernavn: sql\_admin (adganskode:Passw0rd) valgte sa som brugernavn i stedet
* Task 2
  + Se videoen: <https://www.youtube.com/watch?v=ZanGRT1MsHM>
  + Læs bøgerne “Introduktion DB” og guiden “Mini guide til SQL” som kan findes i fællesdrevet
* Task 3
  + Opret database som hedder “fødselsrate\_2017”
  + Opret følgende tabeller
    - Land
    - Rang
  + Indsæt følgende data ind i de 2 tabeller
    - Land
      * Id Land Verdensdel 1 Verdensdel 2
      * 1 Danmark Europa
      * 2 Canada Nordamerika
      * 3 Nigeria Afrika
      * 4 Indien Afrika
      * 5 Brasilien Sydamerika
      * 6 New Zealand Oceanien
      * 7 USA Nordamerika
      * 8 Norge Europa
      * 9 Tyrkiet Europa Asien
      * 10 Ukraine Europa Asien
    - Rang
      * Id Rang Fødselsrate
      * 1 169 10,27
      * 2 168 10,28
      * 3 1 45,45
      * 4 85 19,5
      * 5 129 14,46
      * 6 139 13,33
      * 7 145 12,38
      * 8 149 12,14
      * 9 114 16,33
      * 10 162 10,72
* Task 4
  + Forklar hvilken SQL syntaks der skal bruges for at oprette databasen?
    - USE master;
    - GO
    - CREATE DATABASE fødselsrate\_2017
  + Forklar hvilken SQL syntaks der skal bruges for at oprette Tabellerne / kolonerne?
    - USE fødselsrate\_2017;
    - GO
    - CREATE TABLE Land (
    - ID int PRIMARY KEY IDENTITY(1,1),
    - Land varchar(255),
    - Verdensdel\_1 varchar(255),
    - Verdensdel\_2 varchar(255)
    - );
  + Forklar hvilken SQL syntaks der skal bruges for at indsætte værdierne?
    - INSERT INTO Land(Land,Verdensdel\_1,Verdensdel\_2)
    - VALUES
    - ('Danmark','Europa',''),
    - ('Canada','Nordamerika',''),
    - ('Nigeria','Afrika', ''),
    - ('Indien','Afrika',''),
    - ('Brasilien','Sydamerika',''),
    - ('New Zealand','Oceanien',''),
    - ('USA','Nordamerika',''),
    - ('Norge','Europa',''),
    - ('Tyrkiet','Europa','Asien'),
    - ('Ukraine','Europa','Asien');
    - GO
  + Forklar hvordan du skal normalisere data fra GeoTema?
    - Ved at Lave en tabel for sig selv med verdensdele og så referere til dem i Land tabellen, så hvis nu europa skulle ændre navn ville man kun skulle ændre det et sted nemlig i verdensdele tabellen.
* Task 5
  + I denne task skal der laves Primary key og Foreign key
    - der er vist hvordan man gør oppe i task 4
* Task 6
  + Vis alle Data fra tabellen med navnet “Rang”
    - Det gøres ved at man bruger følgende sql syntaks
      * SELECT \* FROM Rang
  + Vis alle Data fra tabellen med navnet “Land”
    - Det gøres ved at man bruger samme syntaks som i opgaven ovre denne men i stedet for at skrive Rang skriver man Land.
  + Vis alle Data med “Land” og “Fødselsrate”
    - Det gøres ved at bruge en SELECT og så i stedet for \* vælger vi det vi vil have, som er vist herunder.
      * SELECT Land.Land, Rang.Fødselsrate FROM Land,Rang
  + Forklar hvad forskellen er på et inner join, left join, right join og full outer join
    - **INNER JOIN**: returnere noget når der er match i begge tabeller
    - **LEFT JOIN**: returnere alt fra den venstre tabel selvom der ikke er noget match i tabellen til højre
    - **RIGHT JOIN**: Gør det samme som LEFT JOIN bare omvendt
    - **FULL JOIN**: Den kombinere resultaterne fra både LEFT og RIGHT Join
  + Oprette 3 brugere ved hjælp af SQL Syntax
    - CREATE LOGIN Superbruger
    - WITH PASSWORD = 'Passw0rd';
    - CREATE LOGIN Administrator
    - WITH PASSWORD = 'Passw0rd';
    - CREATE LOGIN Standardbruger
    - WITH PASSWORD = 'Passw0rd';
    - CREATE LOGIN [FrederikSrv-12\frnie]
    - FROM WINDOWS;
    - CREATE SERVER ROLE Admins AUTHORIZATION sysadmin;
    - ALTER SERVER ROLE Admins ADD MEMBER Administrator;
    - GO
    - USE fødselsrate\_2017;
    - GO
    - CREATE USER Superbruger FOR LOGIN Superbruger;
    - CREATE USER Administrator FOR LOGIN Administrator;
    - CREATE USER Standardbruger FOR LOGIN Standardbruger;
    - CREATE USER [FrederikSrv-12\frnie] FOR LOGIN [FrederikSrv-12\frnie];
    - CREATE ROLE Standardusers;
    - ALTER ROLE Standardusers ADD MEMBER Standardbruger;
    - CREATE ROLE Superusers;
    - ALTER ROLE Superusers ADD MEMBER Superbruger;
    - CREATE ROLE Windowsusers;
    - ALTER ROLE Windowsusers ADD MEMBER [FrederikSrv-12\frnie];
    - ALTER ROLE db\_owner ADD MEMBER Administrator;
    - GO
    - GRANT SELECT ON DATABASE::fødselsrate\_2017 TO Standardusers;
    - GRANT SELECT,UPDATE ON DATABASE::fødselsrate\_2017 TO Windowsusers;
    - GRANT SELECT,UPDATE ON DATABASE::fødselsrate\_2017 TO Superusers;

# 

# 

# C# Program

## Opgavebeskrivelse

Opgaven er nu at lave et planlægning for at løse virksomhedens IT problem, du har nu fået til opgave at hjælpe GeoTema med at lave offline version af deres statistik program ved hjælp af C# kodning og ms sql database.

## Tidsplan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mandag | Tirsdag | Onsdag | Torsdag | Fredag |
| Database opsætning | Database og Start af C# Kode | fik gjort så man kunne vise data og tilføje data | fik gjort så man kunne redigere, søge og slette data | fik gjort så man kunne oprette, redigere og slette brugere |

## Kode forklaring

### Login

“Login\_btn\_click” metoden bliver kaldt når der bliver klikket på login knappen i index. Den kalder så en metode som hedder “Login”, metoden opretter en klasse ved at kalde en metode i Program.cs filen der hedder “Login” der så forbinder til sql serveren og tjekker det indtastede brugernavn og adgangskode. Udover det tjekker den også hvad brugeren har rettigheder til. Når metoden er færdig med at oprette klassen returnere den klassen til “Login” metoden i index.cs som så sætter klassen til at være lig med den som er i Program.cs som hedder “LoggedUser”. efter det tjekker den om klassen har adgang til de forskellige ting og enabler så knapperne hvis den har. efter det kalder den en metode som hedder “RefreshDataTable” som så har den funktion at hente alle tables og vise dem i listen.

### Graf

“Graph\_btn\_Click” metoden bliver kaldt når man klikker på Graph i index. Metoden laver så en ny graph form som så vises. når den bliver loadet henter den alle dataene fra serveren og lægger dem ind i diagrammet som den så viser.

### Redigere

“Edit\_btn\_Click” metoden bliver kaldt når der bliver klikket på Edit knappen i index. Når metoden bliver kaldt opretter den en ny form som så popper op.

Når formen popper op ligger den alle ID’er fra tabellen ind i en Selector. Derfra kan man så vælge et id og så klikke på select for at få ID’ets oplysninger frem i de 5 felter. de fem felter kan så sendes til sql serveren ved at klikke på Edit. Og så vil den UPDATE alle dataene for det valgte ID, og derefter lukke ned.

### Søge

“Search\_btn\_Click” metoden bliver kaldt når man klikker på search knappen i index. Når knappen så er blevet klikket kalder den en funktion som hedder “Search” og den metode opretter så en search string den så sender videre ind i en metode som ligger i program.cs som så sørger for at hente dataene der bliver søgt efter og returnere dem til “Search” metoden som så opdatere den nuværende tabel med de værdier den har fundet.

### Slette

“Delete\_btn\_Click” metoden bliver kaldt når man klikker på Delete knappen i index. Metoden laver så en ny delete form og viser den. når den er vist loader den alle ID’erne som der er i tabellen og så viser dem i en selector. man kan så vælge et id og klikke på delete. når der bliver klikket på delete bliver der kaldt en metode som så køre en sqlcommand og sletter alt med det valgte ID. Den lukker derefter formen.

### Oprettelse

“New\_btn\_Click” metoden bliver kaldt når man klikker på New knappen i index. Metoden laver så en ny new form og viser den. når den er loadet kan man se 5 bokse man kan udfylde og når man har udfyldt dem kan man trykke på create knappen som så kalder en metoder der sender en request til sql serveren og beder den om at oprette en ny række. Derefter lukker den vinduet ned.

### Bruger redigering, oprettelse og sletning

“User\_btn\_Click” metoden bliver kaldt når man klikker på User i index. Metoden laver så en ny User\_edit form og viser den. Når den loader henter den alle brugere fra sql serveren og ligger dem i en liste, den henter også alle grupper og lægger i en anden liste. Man har så mulighed for at ændre en bruger, slette en bruger og oprette en bruger. udfra hvad man vælger at gøre sender den de valgte instruktioner til serveren som så gør det valgte med brugeren. når den er færdig lukker den vinduet igen.

### Logout

“Logout\_btn\_Click” metoden bliver kaldt når man klikker på Logout i index. Metoden Disabler så alle felter på nær login, username og password.

## Program guide

### Login

Knappen bruges til at loggen ind, dog skal man have indtastet et brugernavn og en adgangskode.

### Redigere

Man kan redigere ved at klikke på Edit knappen. Der vil så blive åbnet et nyt vindue op. man skal så vælge det ID man vil ændre i selectoren oppe til venstre. derefter skal man klikke på Select knappen. så burde alt info om det valgte id gerne komme op i boksene nedenunder. Man kan så ændre dataene og så til sidst klikke på Edit knappen eller cancel hvis du vil fortryde.

### Søge

Man kan søge ved at taste sit søgeord ind i feltet til venstre fra Search knappen. og så klikke på Search knappen. for at få vist alle data igen skal man bare klikke på Refresh knappen

### Slette

Man kan slette ved at klikke på Delete knappen, og så kommer der et vindue frem hvor man kan vælge et ID. Når et ID er valgt kan man klikke på Delete knappen og så bliver det ID slettet, eller man kan klikke på cancel for at fortryde.

### Oprettelse

man kan oprette ved at klikke på New knappen, og så kommer der et vindue frem hvor man kan indtaste data ind i 5 felter. når man har udfyldt felterne kan man klikke på Create knappen. Eller cancel knappen hvis man vil fortryde.

### Bruger redigering, oprettelse og sletning

man kan oprette, ændre, og slette brugere ved at klikke på Users knappen. Der vil så komme et vindue frem der viser alle brugere og også en som der hedder New User. hvis man vælger New User kan man ved at udfylde brugernavn, adganskode og brugergruppe oprette en ny bruger hvis man klikker på Create knappen. Hvis man vil ændre et password eller et navn på en bruger kan man vælge brugeren og så klikke på Edit knappen. Hvis man i stedet vil slette en bruger kan man vælge brugeren og så klikke på delete.

### Graf

Man kan ved at klikke på Graph knappen få vist et diagram over alle data i sql databasen.

# Bilag

## Database og Server 2012

### SQL Kode

|  |
| --- |
| USE master DROP DATABASE fødselsrate\_2017; GO CREATE DATABASE fødselsrate\_2017; GO USE fødselsrate\_2017; GO create table Land ( ID int PRIMARY KEY IDENTITY(1,1), Land varchar(255), Verdensdel\_1 varchar(255), Verdensdel\_2 varchar(255) ); GO create table Rang ( ID int IDENTITY(1,1) FOREIGN KEY REFERENCES Land(ID), Rang int, Fødselsrate float ); GO INSERT INTO Land(Land,Verdensdel\_1,Verdensdel\_2) VALUES ('Danmark','Europa',''), ('Canada','Nordamerika',''), ('Nigeria','Afrika', ''), ('Indien','Afrika',''), ('Brasilien','Sydamerika',''), ('New Zealand','Oceanien',''), ('USA','Nordamerika',''), ('Norge','Europa',''), ('Tyrkiet','Europa','Asien'), ('Ukraine','Europa','Asien'); GO INSERT INTO Rang(Rang, Fødselsrate) VALUES (169,10.27), (168,10.28), (1,45.45), (85,19.50), (129,14.46), (139,13.33), (145,12.38), (149,12.14), (114,16.33), (162,10.72); GO SELECT \* FROM Rang;  SELECT \* FROM Land;  SELECT Fødselsrate FROM Rang; USE master; GO DROP LOGIN Superbruger; DROP LOGIN Standardbruger; DROP LOGIN "FrederikSrv-12\frnie"; DROP LOGIN Administrator; GO CREATE LOGIN Superbruger WITH PASSWORD = 'Passw0rd'; CREATE LOGIN Administrator WITH PASSWORD = 'Passw0rd'; CREATE LOGIN Standardbruger WITH PASSWORD = 'Passw0rd'; CREATE LOGIN [FrederikSrv-12\frnie]  FROM WINDOWS; ALTER SERVER ROLE Admins DROP MEMBER Administrator; DROP SERVER ROLE Admins; CREATE SERVER ROLE Admins AUTHORIZATION sysadmin; ALTER SERVER ROLE Admins ADD MEMBER Administrator; GO USE fødselsrate\_2017; GO CREATE USER Superbruger FOR LOGIN Superbruger; CREATE USER Administrator FOR LOGIN Administrator; CREATE USER Standardbruger FOR LOGIN Standardbruger;  CREATE USER [FrederikSrv-12\frnie] FOR LOGIN [FrederikSrv-12\frnie]; CREATE ROLE Standardusers;  ALTER ROLE Standardusers ADD MEMBER Standardbruger; CREATE ROLE Superusers;  ALTER ROLE Superusers ADD MEMBER Superbruger; CREATE ROLE Windowsusers;  ALTER ROLE Windowsusers ADD MEMBER [FrederikSrv-12\frnie]; ALTER ROLE db\_owner ADD MEMBER Administrator;  GO GRANT SELECT ON DATABASE::fødselsrate\_2017 TO Standardusers; GRANT SELECT,UPDATE ON DATABASE::fødselsrate\_2017 TO Windowsusers; GRANT SELECT,UPDATE ON DATABASE::fødselsrate\_2017 TO Superusers; |

## C# program kode

### Index.cs(startskærm)

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class Base : Form  {  public Base()  {  InitializeComponent();  }   private void Save\_btn\_Click(object sender, EventArgs e)  {   }   private void New\_btn\_Click(object sender, EventArgs e)  {  Form NewForm = new New\_Form();  NewForm.ShowDialog();  NewForm.Dispose();  RefreshDataTable();  }   private void Edit\_btn\_Click(object sender, EventArgs e)  {  Form EditForm = new editform();  EditForm.ShowDialog();  EditForm.Dispose();  RefreshDataTable();  }   private void Delete\_btn\_Click(object sender, EventArgs e)  {  Form DeleteForm = new deleteform();  DeleteForm.ShowDialog();  DeleteForm.Dispose();  RefreshDataTable();  }   private void Refresh\_btn\_Click(object sender, EventArgs e)  {  RefreshDataTable();    }   private void DataTable\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {   }   private void Login\_btn\_Click(object sender, EventArgs e)  {  Login();  }   private void Logout\_btn\_Click(object sender, EventArgs e)  {  Login\_btn.Enabled = true;  Logout\_btn.Enabled = false;  Search\_btn.Enabled = false;  Delete\_btn.Enabled = false;  Refresh\_btn.Enabled = false;  New\_btn.Enabled = false;  Edit\_btn.Enabled = false;  DataTable.ReadOnly = true;  Seach\_for\_textbox.Enabled = false;  Table\_SelectBox.Enabled = false;  Row\_SelectBox.Enabled = false;  User\_btn.Enabled = false;  Graph\_btn.Enabled = false;  }  private void Search()  {  List<String> Tables = Program.getTables();  List<List<String>> Rows = Program.getRows(Tables);  string row = Row\_SelectBox.SelectedText.ToString();  string searchString = "( ";  for (int i = 1; i < Tables.Count - 1; i++)  {   foreach (string value in Rows[i-1])  {  if (value == "ID") continue;  searchString += Tables[i] + "." + value + " LIKE '%" + Seach\_for\_textbox.Text + "%' OR ";  }  }  searchString = searchString.Remove(searchString.LastIndexOf("O"), 3);  searchString += ") AND " + Tables[1] + "." + "ID" + " = " + Tables[2] + "." + "ID";   DataTable.AutoGenerateColumns = false;  DataTable data = new DataTable();  BindingSource sdata = new BindingSource();   data = Program.Search(searchString);  sdata.DataSource = data;  DataTable.DataSource = sdata;  for (int i = 0; i < DataTable.Columns.Count; i++)  {  DataTable.Columns[i].DataPropertyName = Row\_SelectBox.Items[i].ToString();  }    DataTable.Refresh();   }  private void RefreshDataTable()  {  List<String> Tables = Program.getTables();  List<List<String>> Rows = Program.getRows(Tables);  Rows[1].Remove(Rows[1][0]);  Table\_SelectBox.Items.Clear();  Row\_SelectBox.Items.Clear();  string table = "";  foreach (string i in Tables)  {  Table\_SelectBox.Items.Add(i);  table += i + ",";  }  table = table.Remove(table.LastIndexOf(","));  table = table.Remove(0, 4);  foreach (List<String> i in Rows)  {  foreach (string item in i)  {  Row\_SelectBox.Items.Add(item);  }  }  Row\_SelectBox.Items.Add("ALL");  DataTable.AutoGenerateColumns = false;  DataTable data = new DataTable();  BindingSource sdata = new BindingSource();   data = Program.Refresh(Tables);  sdata.DataSource = data;  DataTable.DataSource = sdata;  for (int i = 0; i < DataTable.Columns.Count; i++)  {  DataTable.Columns[i].DataPropertyName = Row\_SelectBox.Items[i].ToString();  }     DataTable.Refresh();  }   private void Seach\_for\_textbox\_TextChanged(object sender, EventArgs e)  {   }   private void Search\_btn\_Click(object sender, EventArgs e)  {  Search();  }   private void Password\_textbox\_KeyPress(object sender, KeyPressEventArgs e)  {  }  private void Login()  {  Program.LoggedUser = Program.Login(Username\_textbox.Text, Password\_textbox.Text);  if (!Program.LoggedUser.CredentialsOK)  {  Username\_textbox.Text = "Failed";  Password\_textbox.Text = "";  return;  }  Login\_btn.Enabled = false;  Logout\_btn.Enabled = true;  Search\_btn.Enabled = true;  if (Program.LoggedUser.Edit)  {  Edit\_btn.Enabled = true;  DataTable.ReadOnly = false;  }  if (Program.LoggedUser.Delete) Delete\_btn.Enabled = true;  if (Program.LoggedUser.Insert) New\_btn.Enabled = true;  if (Program.LoggedUser.ALTER) User\_btn.Enabled = true;  Refresh\_btn.Enabled = true;  Seach\_for\_textbox.Enabled = true;  Table\_SelectBox.Enabled = true;  Row\_SelectBox.Enabled = true;  Graph\_btn.Enabled = true;  RefreshDataTable();  }   private void User\_btn\_Click(object sender, EventArgs e)  {  User\_edit NewForm = new User\_edit();  NewForm.ShowDialog();  NewForm.Dispose();  RefreshDataTable();  }   private void Graph\_btn\_Click(object sender, EventArgs e)  {  graph NewForm = new graph();  NewForm.ShowDialog();  NewForm.Dispose();  }  } } |

### Program.cs

|  |
| --- |
| using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using System.Windows.Forms; using System.Data.SqlClient; using System.Data;  namespace GeoTema\_Program {  static class Program  {   static string ServerName = "FrederikSrv-12\\GEOTEM\_SQL\_2018";  static string DataBaseName = "fødselsrate\_2017";  public static User LoggedUser;  static string Username = "";  static string password = "";  public static string[] New\_values = new string[5];    /// <summary>  /// The main entry point for the application.  /// </summary>  [STAThread]  static void Main()  {  Application.EnableVisualStyles();  Application.SetCompatibleTextRenderingDefault(false);  Application.Run(new Base());   }  public static List<string> getRoles()  {  SqlConnection conn = Connect(Username, password);  List<string> roles = new List<string>();  conn.Open();  SqlCommand command = new SqlCommand("Select \* From sysusers where issqlrole = 1", conn);  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  roles.Add(reader["name"].ToString());  }  }  conn.Close();  return roles;  }  public static List<string> getUsers()  {  SqlConnection conn = Connect(Username, password);  List<string> users = new List<string>();  conn.Open();  SqlCommand command = new SqlCommand("SELECT name FROM sys.database\_principals where(type= 'S' AND default\_schema\_name = 'dbo')", conn);  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  users.Add(reader["name"].ToString());  }  }  conn.Close();  return users;  }   public static SqlConnection Connect(string UserName, string Password)  {  Username = UserName;  password = Password;  SqlConnection conn = new SqlConnection();  conn.ConnectionString =  String.Format($"Data Source={ServerName};") +  String.Format($"Initial Catalog={DataBaseName};") +  String.Format($"User id={UserName};") +  String.Format($"Password={Password};");  return conn;  }  public static User Login(string username, string password)  {  SqlConnection conn = Connect(username, password);  bool credentialsisok;  bool UPDATE = false;  bool DELETE = false;  bool INSERT = false;  bool ALTER = false;  try  {  conn.Open();  credentialsisok = true;  }  catch  {  User faileduser = new User(UPDATE, DELETE,INSERT,ALTER);  faileduser.CredentialsOK = false;  return faileduser;  }    SqlCommand command = new SqlCommand("USE fødselsrate\_2017; SELECT permission\_name FROM fn\_my\_permissions('dbo', 'SCHEMA')", conn);  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  if (reader["permission\_name"].ToString() == "UPDATE")  {  UPDATE = true;  }  else if (reader["permission\_name"].ToString() == "DELETE")  {  DELETE = true;  }  else if(reader["permission\_name"].ToString() == "INSERT")  {  INSERT = true;  }  else if(reader["permission\_name"].ToString() == "ALTER")  {  ALTER = true;  }  }  }  conn.Close();  User user = new User(UPDATE, DELETE, INSERT,ALTER);  user.CredentialsOK = credentialsisok;  return user;  }   internal static void Update(List<string> values)  {  List<string> tables = getTables();  List<List<String>> rows = getRows(tables);  SqlConnection conn = Connect(Username, password);  SqlCommand command = new SqlCommand($"UPDATE {tables[1]} SET {rows[0][1]} = '{values[1]}', {rows[0][2]} = '{values[2]}', {rows[0][3]} = '{values[3]}' WHERE ID = {values[0]}; " +  $"UPDATE {tables[2]} SET {rows[1][1]} = {values[4]}, {rows[1][2]} = {values[5].ToString().Replace(",",".")} WHERE ID = '{values[0]}';", conn);  conn.Open();  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  conn.Close();  }   public static DataTable Search(string searchstring)  {  List<string> table = getTables();  SqlConnection conn = Connect(Username, password);  DataTable dataTable = new DataTable();  try  {  using (conn)  {  using (SqlCommand cmd = new SqlCommand(String.Format($"Select \* from {table[1] + "," + table[2]} where {searchstring}")))  {  cmd.Connection = conn;  using (SqlDataAdapter sda = new SqlDataAdapter(cmd))  {  DataTable dt = new DataTable();  sda.Fill(dataTable);  }  }  }  }  catch  {   }     return dataTable;  }  public static List<String> getTables()  {  SqlConnection conn = Connect(Username, password);  conn.Open();  SqlCommand command = new SqlCommand("USE fødselsrate\_2017; SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE='BASE TABLE' AND TABLE\_CATALOG='fødselsrate\_2017'", conn);  List<String> Tables = new List<String>();  Tables.Add("ALL");  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  Tables.Add(String.Format("{0}", reader["TABLE\_NAME"]));  }  }  conn.Close();  return Tables;  }  public static List<List<String>> getRows(List<String> tables)  {  SqlConnection conn = Connect(Username, password);  List<List<String>> Rows = new List<List<String>>();  foreach (string item in tables)  {  if (item == "ALL")  {  continue;  }   conn.Open();   List<String> Row = new List<string>();   SqlCommand command = new SqlCommand($"USE fødselsrate\_2017; SELECT \* FROM fødselsrate\_2017.INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = N'{item}'", conn);   using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  Row.Add(String.Format("{0}", reader["COLUMN\_NAME"]));  }  }  Rows.Add(Row);  conn.Close();  }  return Rows;  }  public static DataTable Refresh(List<String> tables)  {  SqlConnection conn = Connect(Username, password);  DataTable dataTable = new DataTable();   using (conn)  {  using (SqlCommand cmd = new SqlCommand($"USE fødselsrate\_2017; Select \* from {tables[1] + "," + tables[2]} where {tables[1]}.ID = {tables[2]}.ID"))  {  cmd.Connection = conn;  using (SqlDataAdapter sda = new SqlDataAdapter(cmd))  {  DataTable dt = new DataTable();  sda.Fill(dataTable);  }  }  }   return dataTable;  }  public static void Create(string[] values)  {  List <string> tables = getTables();  SqlConnection conn = Connect(Username, password);  SqlCommand command = new SqlCommand($"INSERT INTO {tables[1]} VALUES ('{values[0]}','{values[1]}','{values[2]}')" +  $"INSERT INTO {tables[2]} VALUES({values[3]},{values[4]})", conn);  conn.Open();  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  conn.Close();  }  public static void Delete(string ID)  {  List<string> tables = getTables();  SqlConnection conn = Connect(Username, password);  SqlCommand command = new SqlCommand($"DELETE FROM {tables[2]} WHERE ID = {ID}" +  $"DELETE FROM {tables[1]} WHERE ID = {ID}", conn);  conn.Open();  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  conn.Close();  }  public static List <int> GetID()  {  List<string> tables = getTables();  List<int> ID = new List<int>();  SqlConnection conn = Connect(Username, password);  SqlCommand command = new SqlCommand($"select {tables[1]}.ID from {tables[1]},{tables[2]} Where {tables[1]}.ID = {tables[2]}.ID", conn);  conn.Open();  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  ID.Add(Convert.ToInt32(reader["ID"]));  }  }  conn.Close();  return ID;  }  public static DataTable GetRow(int id)  {  SqlConnection conn = Connect(Username, password);  DataTable dataTable = new DataTable();  List<string> tables = getTables();  using (conn)  {  using (SqlCommand cmd = new SqlCommand($"Select \* from {tables[1]},{tables[2]} where {tables[1]}.ID = {tables[2]}.ID AND {tables[1]}.ID = {id}"))  {  cmd.Connection = conn;  using (SqlDataAdapter sda = new SqlDataAdapter(cmd))  {  DataTable dt = new DataTable();  sda.Fill(dataTable);  }  }  }   return dataTable;  }  public static bool EditUser(List<string> Values)  {  string commandstring = $"ALTER LOGIN {Values[0]} WITH PASSWORD = '{Values[2]}'; ALTER LOGIN {Values[0]} WITH NAME = [{Values[1]}];" +  $"USE fødselsrate\_2017; ALTER USER {Values[0]} WITH NAME = [{Values[1]}]; ";  if (!AlreadyInSpecificRole(Values))  {  commandstring += $"ALTER ROLE {Values[3]} ADD MEMBER {Values[1]};";  if (Values[3] == "Adminusers")  {  commandstring += "ALTER ROLE db\_owner ADD MEMBER {Values[1]}; use master; ALTER SERVER ROLE Admins ADD MEMBER " + Values[1];  }  }   SqlConnection conn = new SqlConnection();  conn.ConnectionString =  String.Format($"Data Source={ServerName};") +  String.Format($"Initial Catalog=master;") +  String.Format($"User id={Username};") +  String.Format($"Password={password};");   SqlCommand command = new SqlCommand(commandstring, conn);  conn.Open();  try {  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  }catch  {  return false;  }      conn.Close();  return true;  }   public static bool AlreadyInSpecificRole(List<string>values)  {  string commandstring = "SELECT role.name as rolename,bruger.name as username FROM sys.database\_role\_members ,sys.database\_principals as role, sys.database\_principals as bruger where role.type = 'R' and role\_principal\_id = role.principal\_id and member\_principal\_id = bruger.principal\_id or bruger.type = 'S'and member\_principal\_id = bruger.principal\_id and role\_principal\_id = role.principal\_id";   SqlConnection conn = Connect(Username, password);  SqlCommand command = new SqlCommand(commandstring, conn);  conn.Open();  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  if (reader["rolename"].ToString() == values[3] && reader["username"].ToString() == values[0]) return true;  }  }  return false;  }  public static bool CreateUser(List<string> Values)  {  string ekstra = "";  if (Values[3] == "Adminusers")  ekstra = $"ALTER ROLE db\_owner ADD MEMBER {Values[1]}; USE master; ALTER SERVER ROLE Admins ADD MEMBER " + Values[1];   SqlConnection conn = new SqlConnection();  conn.ConnectionString =  String.Format($"Data Source={ServerName};") +  String.Format($"Initial Catalog=master;") +  String.Format($"User id={Username};") +  String.Format($"Password={password};");  SqlCommand command = new SqlCommand($"USE master CREATE LOGIN {Values[1]} WITH PASSWORD = '{Values[2]}'; " +  $"USE fødselsrate\_2017; CREATE USER {Values[1]} FOR LOGIN {Values[1]}; " +  $"ALTER ROLE {Values[3]} ADD MEMBER {Values[1]}; " + ekstra, conn);  conn.Open();  try  {  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  }  catch  {  conn.Close();  return false;  }  conn.Close();   return true;  }  public static void DeleteUser(string User)  {  SqlConnection conn = new SqlConnection();  conn.ConnectionString =  String.Format($"Data Source={ServerName};") +  String.Format($"Initial Catalog=master;") +  String.Format($"User id={Username};") +  String.Format($"Password={password};");  SqlCommand command = new SqlCommand($"use fødselsrate\_2017; DROP USER {User}; USE master DROP LOGIN {User}; ", conn);  conn.Open();  try  {  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  }  }  }  catch   {     }      conn.Close();  }  }  class TablesData  {    }  class User  {  public bool Edit { get; set; }  public bool Delete { get; set; }  public bool Insert { get; set; }  public bool CredentialsOK { get; set; }  public bool ALTER { get; set; }  public User(bool edit, bool delete, bool insert, bool alter)  {  Edit = edit;  Delete = delete;  Insert = insert;  ALTER = alter;  }   } } |

### deleteform.cs(sletning af rækker)

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class deleteform : Form  {  public deleteform()  {  InitializeComponent();  }   private void textBox1\_TextChanged(object sender, EventArgs e)  {   }   private void delete\_btn\_Click(object sender, EventArgs e)  {  if (ID\_select.SelectedItem == null)  {  string message = "You have not selected anything";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }   Program.Delete(ID\_select.SelectedItem.ToString());  this.Close();  }   private void cancel\_btn\_Click(object sender, EventArgs e)  {  this.Close();  }   private void deleteform\_Load(object sender, EventArgs e)  {  List<int> IDs = Program.GetID();  foreach (int i in IDs)  {  ID\_select.Items.Add(i);  }  }   private void ID\_select\_SelectedIndexChanged(object sender, EventArgs e)  {   }  } } |

### editform.cs(redigering af tabeller)

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class editform : Form  {  public bool Selected { get; set; }    public editform()  {  InitializeComponent();  }   private void editform\_Load(object sender, EventArgs e)  {  List<int> IDs = Program.GetID();  foreach(int i in IDs)  {  ID\_select.Items.Add(i);  }  }   private void Create\_btn\_Click(object sender, EventArgs e)  {  if (!Selected)  {  string message = "You have not selected anything";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  DataTable selected = Program.GetRow(Convert.ToInt32(ID\_select.SelectedItem));  List<string> values = new List<string>();  values.Add(selected.Rows[0].Field<int>(selected.Columns[0]).ToString());  values.Add(Land\_txtbox.Text);  values.Add(Verdensdel\_txtbox.Text);  values.Add(Verdensdel2\_txtbox.Text);  values.Add(rang\_txtbox.Text);  values.Add(fødselsrate\_txtbox.Text);  Program.Update(values);  this.Close();  }   private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)  {  }   private void Select\_btn\_Click(object sender, EventArgs e)  {  if (ID\_select.SelectedItem == null)  {  string message = "You have not selected anything";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }   DataTable selected = Program.GetRow(Convert.ToInt32(ID\_select.SelectedItem));  Land\_txtbox.Text = selected.Rows[0].Field<string>(selected.Columns[1]);  Verdensdel\_txtbox.Text = selected.Rows[0].Field<string>(selected.Columns[2]);  Verdensdel2\_txtbox.Text = selected.Rows[0].Field<string>(selected.Columns[3]);  rang\_txtbox.Text = selected.Rows[0].Field<int>(selected.Columns[5]).ToString();  fødselsrate\_txtbox.Text = selected.Rows[0].Field<double>(selected.Columns[6]).ToString();  Selected = true;  }   private void rang\_txtbox\_KeyPress(object sender, KeyPressEventArgs e)  {  if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))  {  e.Handled = true;  }   }   private void fødselsrate\_txtbox\_KeyPress(object sender, KeyPressEventArgs e)  {  if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&  (e.KeyChar != '.'))  {  e.Handled = true;  }   if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))  {  e.Handled = true;  }  }  } } |

### User\_edit.cs(Redigering, oprettelse, og sletning af brugere)

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class User\_edit : Form  {   public User\_edit()  {  InitializeComponent();  }   private void textBox1\_TextChanged(object sender, EventArgs e)  {   }   private void label3\_Click(object sender, EventArgs e)  {   }   private void textBox2\_TextChanged(object sender, EventArgs e)  {   }   private void User\_Load(object sender, EventArgs e)  {  List<string> Users = Program.getUsers();  User\_box.Items.Add("New User");  foreach(string i in Users)  {  User\_box.Items.Add(i);  }  List<string> Roles = Program.getRoles();  foreach(string i in Roles)  {  if (i.Contains("db\_") || i.Contains("public")) continue;  Group\_box.Items.Add(i);  }  }   private void Create\_btn\_Click(object sender, EventArgs e)  {  if (User\_box.SelectedItem == null)  {  string message = "You have not selected a User";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  else if (User\_box.SelectedItem.ToString() != "New User")  {  string message = "You have to select New User";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  else if (Group\_box.SelectedItem == null)  {  string message = "You have not selected a group";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  List<string> values = new List<string>();  values.Add(User\_box.SelectedItem.ToString());  values.Add(Username\_text.Text);  values.Add(Password\_text.Text);  values.Add(Group\_box.SelectedItem.ToString());  bool done = Program.CreateUser(values);  if (done) this.Close();  else  {  string message = "The user do aldready exist";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }   }   private void Edit\_btn\_Click(object sender, EventArgs e)  {  if (User\_box.SelectedItem == null)  {  string message = "You have not selected anything";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  else if (User\_box.SelectedItem.ToString() == "New User")  {  string message = "You cannot select New User";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  else if (Group\_box.SelectedItem == null)  {  string message = "You have not selected a group";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  List<string> values = new List<string>();  values.Add(User\_box.SelectedItem.ToString());  values.Add(Username\_text.Text);  values.Add(Password\_text.Text);  values.Add(Group\_box.SelectedItem.ToString());  bool done = Program.EditUser(values);  if (done) this.Close();  else  {  string message = "The user do aldready exist or is already in that group ";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  }   private void User\_box\_SelectedIndexChanged(object sender, EventArgs e)  {  if (User\_box.SelectedItem != null && User\_box.SelectedItem.ToString() == "New User") return;  if (User\_box.SelectedItem != null) Username\_text.Text = User\_box.SelectedItem.ToString();  }   private void Delete\_btn\_Click(object sender, EventArgs e)  {  if (User\_box.SelectedItem == null)  {  string message = "You have not selected anything";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  else if (User\_box.SelectedItem.ToString() == "New User")  {  string message = "You cannot select New User";  string caption = "Error Detected";  MessageBoxButtons buttons = MessageBoxButtons.OK;  DialogResult result;   result = MessageBox.Show(message, caption, buttons);  return;  }  Program.DeleteUser(User\_box.SelectedItem.ToString());  this.Close();    }  } } |

### New Form.cs(opretning af rækker)

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class New\_Form : Form  {    public New\_Form()  {  InitializeComponent();  }   private void button2\_Click(object sender, EventArgs e)  {  this.Close();  }   private void label1\_Click(object sender, EventArgs e)  {   }   private void Rang\_label\_Click(object sender, EventArgs e)  {   }   private void Create\_btn\_Click(object sender, EventArgs e)  {   Program.New\_values[0] = Land\_txtbox.Text;  Program.New\_values[1] = Verdensdel\_txtbox.Text;  Program.New\_values[2] = Verdensdel2\_txtbox.Text;  Program.New\_values[3] = rang\_txtbox.Text;  Program.New\_values[4] = fødselsrate\_txtbox.Text;  Program.Create(Program.New\_values);  this.Close();  }   private void rang\_txtbox\_TextChanged(object sender, EventArgs e)  {    }   private void rang\_txtbox\_KeyPress(object sender, KeyPressEventArgs e)  {  if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))  {  e.Handled = true;  }  }   private void fødselsrate\_txtbox\_KeyPress(object sender, KeyPressEventArgs e)  {  if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&  (e.KeyChar != '.'))  {  e.Handled = true;  }   // only allow one decimal point  if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))  {  e.Handled = true;  }  }  } } |

### Graph.cs

|  |
| --- |
| using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms;  namespace GeoTema\_Program {  public partial class graph : Form  {  public graph()  {  InitializeComponent();  }   private void graph\_Load(object sender, EventArgs e)  {  DataTable Data = Program.Refresh(Program.getTables());  for (int i = 0; i < Data.Rows.Count; i++)  {  Rang.ChartAreas["ChartArea"].AxisX.Interval = 1;  Rang.Series["Rang"].Points.AddXY(Data.Rows[i][1], Convert.ToInt32(Data.Rows[i][5]));  Rang.Series["Fødselsrate"].Points.AddXY(Data.Rows[i][1], Convert.ToDouble(Data.Rows[i][6])/100);  }  }  } } |

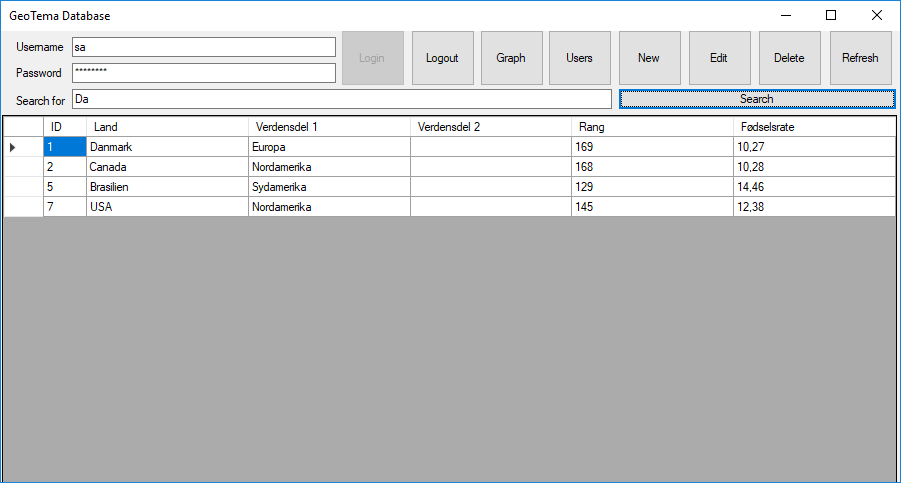
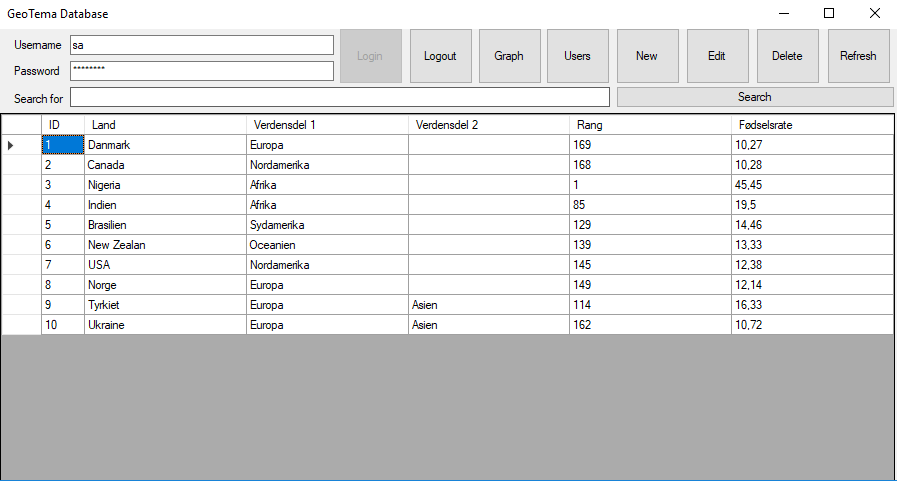
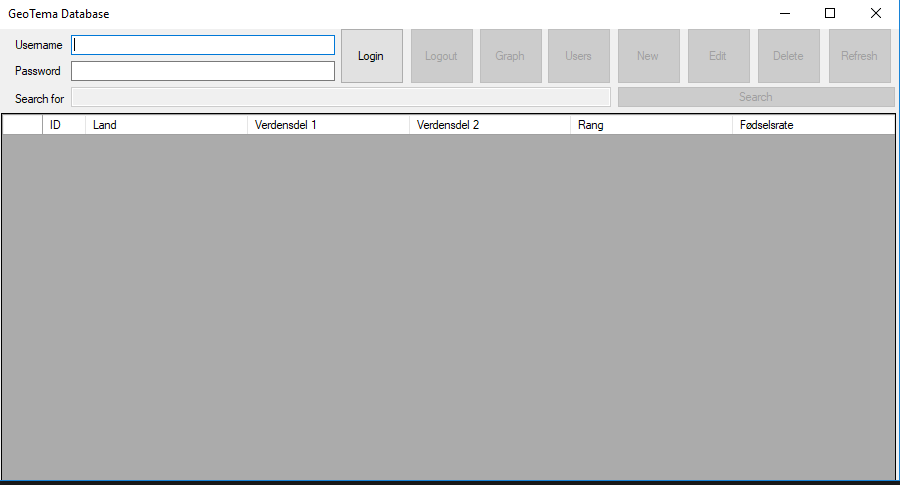
## 

## 

## C# program billeder

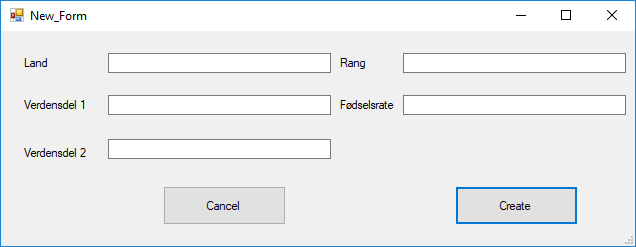
### Index

Billede til venstre er når der ikke er nogen logget ind og det til højre viser at der er en der er logget ind. Det nederste billede viser hvordan det ser ud når der er blevet søgt efter noget.



### New Form.cs(Ny række i tabel)

Det her billede viser hvordan det ser ud når man vil lave en ny række.

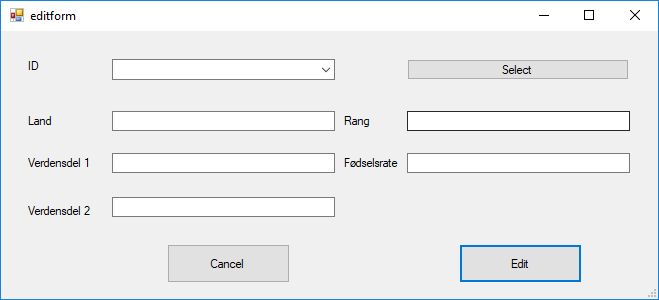


### 

### 

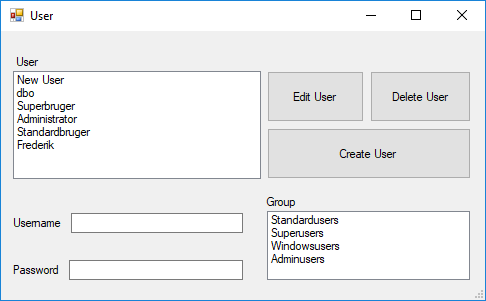
### editform.cs(redigering af tabeller)

Billedet her viser hvordan det ser ud når man vil redigere en række i tabellen.



### User\_edit.cs(Redigering, oprettelse, og sletning af brugere)

Billedet her viser hvordan bruger redigering ser ud.

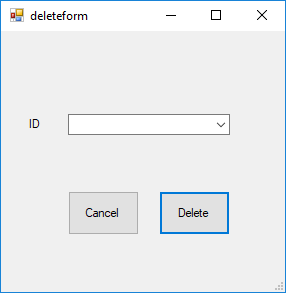


### 

### 

### deleteform.cs

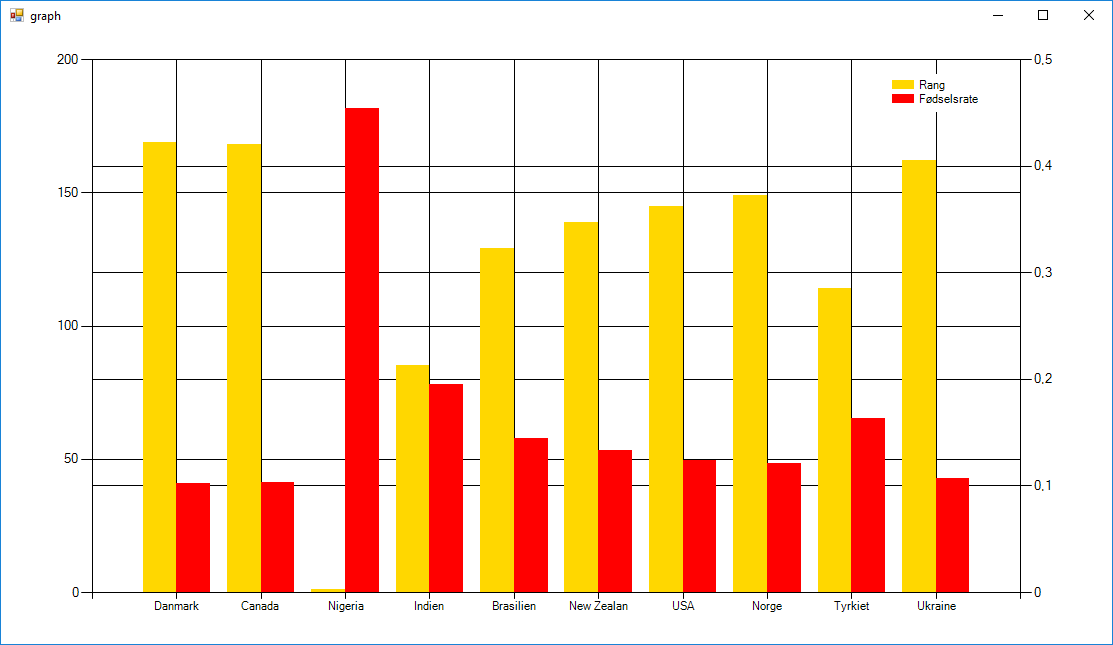
Det her billede viser hvordan deleteform ser ud.



### 

### Graph.cs

Det her billede viser grafen.



### 