Report #1

# Project informations

**Project owner: Mariusz Piątkowski**

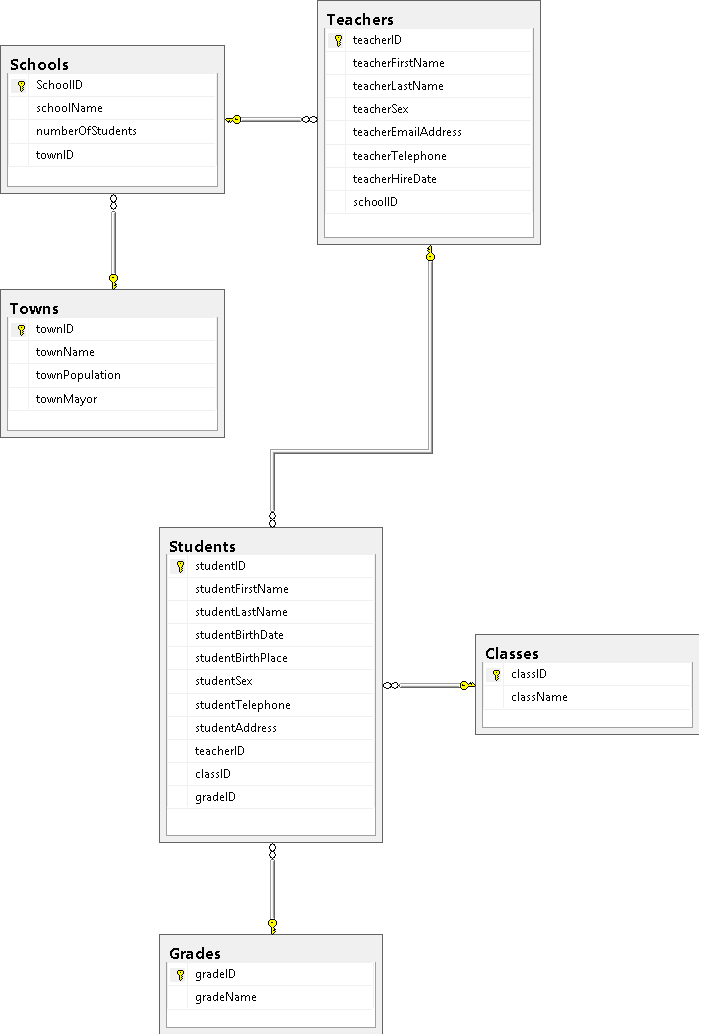
**Database name:** DatabaseProgramming2

**SQL:** SQL Server/TSQL

**Date created:** January 2016

**Repository: https://github.com/mpxx24/repo**

# Database diagram



Bigger photo in attached files and repository

# Creating database

Database represents simple Town/School database. For creating it I used the most obvious code

create database DatabaseProgramming2

Name could be better, but it’s not what it is about (I’m bad at naming things). Now, creating tables.

if not exists (select \* from sys.tables where name = 'Towns')

create table Towns(

townID int primary key identity(1,1),

townName nvarchar(15) not null,

townPopulation int,

townMayor nvarchar (32)

)

go

if not exists (select \* from sys.tables where name = 'Schools')

create table Schools(

schoolID int primary key identity(1,1),

schoolName nvarchar(20) not null,

schoolNumberOfStudents int,

townID int,

foreign key (townID) references Towns(townID) on delete cascade

)

go

Just two examples, rest can be found in attacked files or after clicking link to repository in ‘Project Informations’ tab or in attached files.

We have basic Towns table with few properties, and School table with foreign key from Towns table.

Filling tables is divided on two parts. First is creating stored procedures that generate random data.

# Creating data

Examples:

create procedure spdCreateRandomString

@maxLength as int

as

begin

declare @townName nvarchar(15)

declare @lenght int

select @townName = ''

set @lenght = cast(rand() \*(@maxLength-1)+1 as int)

while @lenght <> 0

begin

select @townName = @townName + char(round(rand() \* 26, 1) + 97)

set @lenght = @lenght - 1

end

select @townName

end

go

Procedure creates random string with given length as procedure parameter.

Or

create procedure spdCreateRandomPhoneNumber

as

begin

declare @phoneNumber int

select cast(rand()\*(999999999-100000000)+100000000 as int)

end

go

Procedure which creates random phone number (int value with length=9)

Rest of the procedures can be found in attached files or after clicking link to repository.

# Filling tables

Tables are filled with random data generated by stored procedures. Results of the procedures are passed to temporary tables, and from there to project tables.

Examples:

declare @iterator int = 1;

declare @tempCName table (tempIndex int identity(1,1), cName nvarchar(32))

while @iterator <> 1001

begin

insert @tempCName

exec spdCreateRandomString 15

set @iterator = @iterator + 1

end

insert into Classes(className)

select t.cName

from @tempCName t

OR

declare @iterator int = 1;

declare @tempSNAME table (tempIndex int identity(1,1), sName nvarchar(15))

declare @tempSNUMBEROFSTUDENTS table (tempIndex int identity(1,1), sNumberOfStudents int)

declare @tempSTOWNID table (tempIndex int identity(1,1), sTownId int)

while @iterator <> 1001

begin

insert @tempSNAME

exec spdCreateRandomString 20

insert @tempSNUMBEROFSTUDENTS

exec spdCreateRandomInt

insert @tempSTOWNID

select top 1 townID from Towns

order by newid()

set @iterator = @iterator + 1

end

insert into Schools(schoolName, numberOfStudents, townID)

select t.sName, t2.sNumberOfStudents, t3.sTownId

from @tempSNAME t

inner join @tempSNUMBEROFSTUDENTS t2 on t2.tempIndex = t.tempIndex

inner join @tempSTOWNID t3 on t3.tempIndex = t.tempIndex

select \* from @tempSNAME

select \* from @tempSNUMBEROFSTUDENTS

select \* from @tempSTOWNID

Foreign keys are generated randomly from specific tables like this:

insert @tempTable

select top 1 tableID from Table

order by newid()

More can be found in *qInsertRandomDataInTables.sql* file in attacked files/repository.