

# **Przetwarzanie własnych agregatów CLR UDA.**

Bazy danych 2

Przemysław Rewiś

21 czerwca 2022

## Założenia projektu

Celem projektu jest opracowanie i implementacja własnego API. Ma ono obsługiwać zestaw pięciu własnych agregatów CLR UDA, z których dwa będą w wersji rozszerzonej. Opracowane API wykorzystane zostanie w aplikacji terminalowej napisanej w C# wykonującej testy jednostkowe dla udostępnionych funkcjonalności. Połączenie z bazą danych w przygotowanej na potrzeby testów aplikacji zrealizowane zostanie z wykorzystaniem interfejsu ADO.NET. W ramach projektu należy również przygotować odpowiedni zestaw danych umożliwiających poprawną weryfikację agregatów.

## Wstęp teoretyczny

Funkcje agregujące wykonują obliczenia na zbiorze wartości i zwracają pojedynczą wartość. Microsoft SQL Server obsługuje tylko wbudowane funkcje agregujące takie jak SUM lub MAX, które działają na zestawie wejściowych wartości skalnych i generują z tego zestawu pojedynczą wartość zagregowaną. Integracja programu SQL Server ze środowiskiem uruchomieniowym języka wspólnego (CLR) programu Microsoft .NET Framework umożliwia deweloperom tworzenie niestandardowych funkcji agregujących w kodzie oraz udostępnianie tych funkcji np. w języku Transact-SQL. Wyróżniamy dwa rodzaje CLR UDA – proste i zaawansowane.

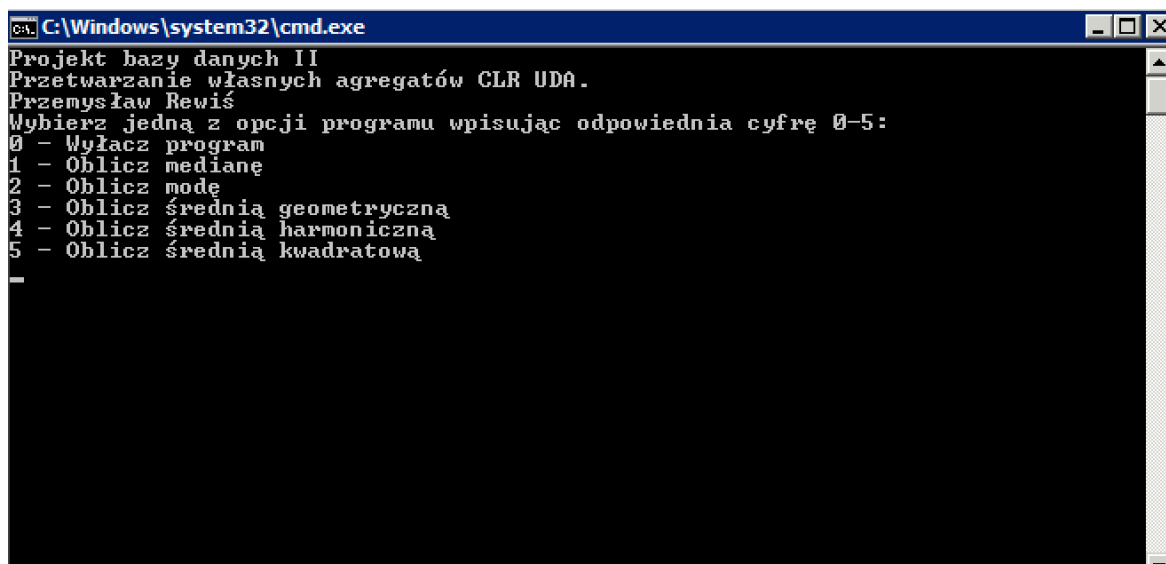
## Opis funkcjonalności API

Projekt składa się z bazy danych oraz programów napisanych w języku C#

Opracowane API będzie implementowało poniższe funkcje agregujące:

- Mediana (agregat w wersji rozszerzonej)
- Moda (agregat w wersji rozszerzonej)
- Średnia geometryczna (agregat w wersji podstawowej)
- Średnia harmoniczna (agregat w wersji podstawowej)
- Średnia kwadratowa (agregat w wersji podstawowej)

Po uruchomieniu aplikacji, w konsoli wyświetla się opis i instrukcja:



```
C:\Windows\system32\cmd.exe
Projekt bazy danych II
Przetwarzanie własnych agregatów CLR UDA.
Przemysław Rewiś
Wybierz jedną z opcji programu wpisując odpowiednią cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
```

Instrukcja informuje użytkownika o opcjach dostępnych do wykorzystania w ramach aplikacji. Należy wprowadzić na standardowe wejście programu cyfrę, odpowiadającą konkretnej opcji. Jeżeli użytkownik wpisze cokolwiek innego, niż cyfrę z przedziału 0-5, na ekran wypisze się komunikat informujący o błędzie, po czym ponownie wypisze instrukcję:

```
C:\Windows\system32\cmd.exe
Projekt bazy danych II
Przetwarzanie własnych agregatów CLR UDA.
Przemysław Rewiś
Wybierz jedną z opcji programu wpisując odpowiednią cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
9
Wpisz poprawny numer opcji
Wybierz jedną z opcji programu wpisując odpowiednią cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
-
```

W przypadku wpisania 0 na standardowe wejście, aplikacja przerywa swoje działanie. Z kolei jeżeli wpisana przez użytkownika cyfra będzie z przedziału 1-5, aplikacja zapisze wybór i przejdzie do kolejnego etapu aplikacji. Po poprawnym wybraniu opcji, wyświetla się dalsza część instrukcji:

```
C:\Windows\system32\cmd.exe
Przemysław Rewiś
Wybierz jedną z opcji programu wpisując odpowiednią cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
9
Wpisz poprawny numer opcji
Wybierz jedną z opcji programu wpisując odpowiednią cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
1
Wybierz na czym ma zostać użyty agregat poprzez wpisanie odpowiedniej cyfry 0-3:
0 - Powrót
1 - Wzrost
2 - Waga
3 - IQ
```

Ten etap jest analogiczny do poprzedniego, z tą różnicą że tym razem użytkownik wybiera atrybut, na którym chce zastosować wybraną wcześniej opcję. Dostępne wybory wypisane są na zrzucie ekranu, a w przypadku innego wyboru wyświetli się komunikat o błędzie i ponownie druga część instrukcji. Z kolei w przypadku powodzenia, na ekran wypisany zostanie wynik. Następnie, aplikacja wróci do pierwotnego stanu aplikacji, tak aby bez

wyłączenia i ponownego uruchomienia aplikacji możliwe było ponowne wybranie interesującej nas opcji.

```

C:\Windows\system32\cmd.exe
Wybierz jedną z opcji programu wpisując odpowiednia cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową
1
Wybierz na czym ma zostać użyty agregat poprzez wpisanie odpowiedniej cyfrę 0-3:
0 - Powrót
1 - Wzrost
2 - Waga
3 - IQ
1
Mediana wzrostu uczniów w szkole wyniosła:
175
Wybierz jedną z opcji programu wpisując odpowiednia cyfrę 0-5:
0 - Wyłącz program
1 - Oblicz medianę
2 - Oblicz modę
3 - Oblicz średnią geometryczną
4 - Oblicz średnią harmoniczną
5 - Oblicz średnią kwadratową

```

## Opis typów danych

W bazie znajdują się dwie tabele:

tabela UCZNIOWIE z informacjami o uczniach uczęszczających na zajęcia w przykładowej szkole:

UCZNIOWIE	
UczenID (PK)	int
Wzrost (w centymarach)	int
Waga (w kilogramach)	float
IQ	int

tabela TEST stworzona w celu wykonania testów jednostkowych sprawdzających poprawność działania agregatów.

TEST	
TESTID (PK)	int
testv	float

## Opis metod udostępnionych w ramach API

Plik *Program.cs* zawiera kod, który obsługuje standardowe wejście i wyjście w konsoli oraz połączenie z bazą danych. W pliku znajduje się klasa *Program*, która zawiera publiczną statyczną metodę *Main()* obsługującą interakcję z użytkownikiem. Innymi metodami klasy "Program" są *wypiszDane()* oraz *wpiszOpcje()*, które wypisują na ekran instrukcje dotyczące odpowiednio agregatów i atrybutów, na których ma działać agregat. Podstawowe CLR UDA (*Format.Native*) wymagały zdefiniowania czterech metod:

- *Init()* - metoda uruchamiana tylko raz na początku, która inicjalizuje dane,
- *Accumulate()* - metoda wywoływana jest przy przetwarzaniu każdego wiersza, pozwalając na dołączanie przekazanych danych,
- *Merge()* - metoda wywoływana, gdy SQL Server zdecyduje, by wykorzystać przetwarzanie równoległe do zakończenia tworzenia agregatu,
- *Terminate()* - końcowa metoda UDA, która jest wywoływana po przetworzeniu wszystkich wierszy i połączeniu wszystkich agregatów utworzonych przy przetwarzaniu równoległym, a następnie zwraca końcowy wynik z agregatu do silnika zapytania.

Rozszerzone CLR UDA oprócz zdefiniowania powyższych metod wymagały również napisania metod:

- *Read()* - metoda do wczytywania zserializowanych danych,
- *Write()* - metoda do zapisywania zserializowanych danych.

## Opis implementacji

Baza danych została skonfigurowana przy pomocy skryptów napisanych w języku SQL oraz środowiska SQL Server Management Studio. Aplikacja została napisana w języku C# przy użyciu zintegrowanego środowiska programistycznego firmy Microsoft Visual Studio 2008. Dostęp do bazy danych z poziomu aplikacji napisanej w C# zrealizowany został dzięki odpowiednim komponentom znajdujących się w przestrzeni nazw *System.Data.SqlClient*.

## Testy jednostkowe

Projekt zawiera 5 testów jednostkowych:

- MedianaTest
- ModaTest
- SredniaGeometrycznaTest
- SredniaHarmonicznaTest
- SredniaKwadratowaTest

Każdy test jest osobną metodą i sprawdza poprawność działania jednego z agregatów porównując wynik wywołania agregatu na tabeli *TEST* z wynikami obliczonymi ręcznie, na tych samych danych. Wyniki przeprowadzonych testów jednostkowych są następujące:

Administrator@MSSQLSERVER024 20 Run Debug				
Test run completed Results: 5/5 passed; Item(s) checked: 0				
	Result	Test Name	Project	Error Message
	Passed	MedianaTest	Tests	
	Passed	ModaTest	Tests	
	Passed	SredniaGeometrycznaTe	Tests	
	Passed	SredniaHarmonicznaTes	Tests	
	Passed	SredniaKwadratowaTes	Tests	

## Kod źródłowy

Kod źródłowy znajduje się w katalogu z projektem i składa się z:

- *Database.sql* - skrypt SQL, który zawiera polecenia konfigurujące bazę danych
- katalog *Tests* zawierający pliki obsługujące testy jednostkowe
- katalog *ConsoleApplication* zawierający pliki obsługujące aplikację terminalową
- katalog *Agregaty* zawierający pliki obsługujące tworzenie agregatów

## Podsumowanie

Projekt pokazuje w jaki sposób możemy definiować własne agregaty korzystając z możliwości środowiska uruchomieniowego języka wspólnego (CLR) programu Microsoft .NET Framework. Powinniśmy pamiętać, iż operacje wykonywane za pomocą Frameworka są znacznie bardziej kosztowne i czasochłonne, niż alternatywna operacja w Transact-SQL. Przy użyciu CLR powinniśmy najpierw sprawdzić, czy korzyści wynikające z łatwiejszego zakodowania są na tyle duże, aby zrezygnować z SQL.



# Literatura

- <https://docs.microsoft.com/pl-pl/>
- [https://newton.fis.agh.edu.pl/~antek/read\\_pdf.php?file=BD2\\_L09\\_CLR.pdf](https://newton.fis.agh.edu.pl/~antek/read_pdf.php?file=BD2_L09_CLR.pdf)
- <https://www.centrumxp.pl/Publikacja/20-Uzycie-NET-CLR-w-SQL-Server-2005>