

Bazy Danych 2

Dokumentacja projektowa

Przetwarzanie własnych agregatów CLR UDA

Autor: Szymon Telega

Nr. indeksu: 297924

Uczelnia: Akademia Górniczo-Hutnicza im. Stanisława

Wydział: Fizyki i Informatyki Stosowanej

Kierunek: Informatyka Stosowana

Rok: 3.

Opis problemu

Celem projektu było opracować API oraz jego implementacji obsługującej wybrany zestaw własnych agregatów CLR UDA. Należało przygotować 5 agregatów, z czego przynajmniej 2 w wersji rozszerzonej. Dodatkowo, należało przygotować odpowiedni zestaw danych oraz testy jednostkowe, umożliwiające weryfikację przygotowanych agregatów.

Projekt składa się z bazy danych oraz programów napisanych w języku C#. W bazie znajdują się:

- tabela "Cracow" z informacjami dotyczącymi warunków pogodowych w Krakowie w 2019 roku. Tabela składa się z 3 atrybutów - dzień, temperatura w stopniach celsjusza w danym dniu oraz opady deszczu w milimetrach na metr kwadratowy w danym dniu. Wypełniona jest ona autentycznymi danymi dla każdego dnia tego roku.
- tabela "TestTable" stworzona w celu wykonania na nich testów jednostkowych sprawdzających poprawność działania agregatów. Składa się z atrybutów "ID" oraz "Value" i zawiera jedynie 4 rekordy.
- 5 agregatów pozwalających wyciągnięcie statystyk z tabel:
 - 1) "GeoMean" - Agregat obliczający średnią geometryczną
 - 2) "Median" - Agregat obliczający medianę
 - 3) "Mode" - Agregat obliczający dystrybuantę
 - 4) "RMS" - Agregat obliczający średnią kwadratową
 - 5) "TruncatedMean" - Agregat obliczający średnią ucinaną (ucina 10% maksymalnych i minimalnych wartości)

Opis funkcjonalności API

Projekt jest aplikacją konsolową, która dzięki prostemu interfejsowi pozwala na wykorzystanie stworzonych w ramach projektu agregatów.

Po uruchomieniu aplikacji, w konsoli wyświetla się krótki opis aplikacji:

```
Szymon Telega - Projekt

Projekt przedstawia stworzone przeze
mnie CLR-UDA. W bazie danych znajduje
sie 5 funkcji agregujacych:
1. Mediana
2. Srednia geometryczna
3. Srednia kwadratowa
4. Srednia ucinana
5. Dominanta

W bazie znajduje sie rowniez tabela
zawierajaca dane dotyczace pogody w
Krakowie w 2019 roku. Sklada sie ona
z nastepujacych atrybutow:
1. Dzień
2. Temperatura <w danym dniu>
3. Opady deszczu <w danym dniu>
```

Zaraz po widocznym opisie, wyświetla się instrukcja obsługi:

```
Prosze wybrac agregat ktory ma zostac
uzyty (nalezy wpisac cyfre):
0. Przerwij
1. Mediana
2. Srednia geometryczna
3. Srednia kwadratowa
4. Srednia ucinana
5. Dominanta
6. Srednia arytmetyczna

Funkcja agregujaca 'sr arytmetyczna'
nie zostala zdefiniowana przeze mnie,
pojawiła sie tu w celu porownania
wynikow z innych funkcji agregujacych.
```

Instrukcja informuje użytkownika o agregatach, dostępnych do użycia w ramach aplikacji. Aby kontynuować, należy wprowadzić na standardowe wejście programu cyfrę, odpowiadającą konkretnemu krokowi. Jeżeli użytkownik wpisze cokolwiek innego, niż cyfrę z przedziału 0-6, na ekran wypisze się komunikat informujący o błędzie, po czym ponownie wypisze instrukcję:

```
10
Prosze wybrac cyfre z zakresu 0-6
```

```
Prosze wybrac agregat ktory na zostac
uzyty (nalezy wpisac cyfre):
0. Przerwij
1. Mediana
2. Srednia geometryczna
3. Srednia kwadratowa
4. Srednia ucinana
5. Dominanta
6. Srednia arytmetyczna

Funkcja agregujaca 'sr arytmetyczna'
nie zostala zdefiniowana przeze mnie,
pojawiła sie tu w celu porownania
wynikow z innych funkcji agregujacych.
```

W przypadku wpisania "0" na standardowe wejście, aplikacja przerywa swoje działanie. Z kolei jeżeli wpisana przez użytkownika cyfra będzie z przedziału 1-6, aplikacja zapisze wybór i przejdzie do kolejnego etapu aplikacji. Każda z tych cyfr przypisana jest do odpowiednich agregatów, tak jak przedstawione jest to na zrzucie ekranu. Po poprawnym wybraniu opcji, wyświetla się dalsza część instrukcji:

```
Teraz prosze wybrac atrybut na ktorym
ma zostac uzyty agregat(nalezy wpisac
cyfre):
0. Powrot do wyboru agregatu
1. Temperatura
2. Opady deszczu
```

Ten etap jest analogiczny do poprzedniego, z tą różnicą że tym razem użytkownik wybiera atrybut, na którym chce zastosować wybrany wcześniej agregat. Dostępne wybory wypisane są na zrzucie ekranu, a w przypadku wyboru opcji nieokreślonej w instrukcji, analogicznie wyświetli się komunikat o błędzie i ponownie druga część instrukcji. Z kolei w przypadku powodzenia, na ekran wypisane zostaną wybrane przez użytkownika opcje oraz finalny opis żądania wraz z wynikiem. Następnie, aplikacja wróci do pierwotnego stanu aplikacji, tak aby bez wychodzenia i ponownego uruchamiania aplikacji możliwe było ponowne użycie agregatów.

```
2
5 2
Dominanta z opadów deszczu w Krakowie w 2019r. wyniosła:
0,9mm na metr kwadratowy

: Prosze wybrac agregat ktory ma zostac :
: uzyty (należy wpisac cyfry):          :
: 0. Przerwij                           :
: 1. Mediana                            :
: 2. Średnia geometryczna                :
: 3. Średnia kwadratowa                  :
: 4. Średnia ucinana                     :
: 5. Dominanta                           :
: 6. Średnia arytmetyczna                :
:                                       :
: Funkcja agregująca 'sr arytmetyczna'   :
: nie została zdefiniowana przeze mnie,  :
: pojawiła się tu w celu porównania       :
: wyników z innymi funkcjami agregującymi. :
:
```

Opis typów danych oraz metod udostępnionych w ramach API

Część aplikacji napisanej w języku C# można podzielić na 3 grupy:

- Plik "Program.cs"- program obsługujący standardowe wejście i wyjście w konsoli, oraz połączenie z bazą danych. W pliku znajduje się klasa "Program", w której główną metodą jest metoda "Main()", która obsługuje ona interakcję z użytkownikiem. Innymi wartymi uwagi metodami klasy "Program" są:
 - OpenConnection() - metoda nawiązująca połączenie z bazą danych
 - PrintAggOptions() i PrintDataOptions() - metody wypisujące na ekran instrukcje dotyczące odpowiednio agregatów i atrybutów na których ma działać agregat.
 - PrintResult() - metoda uruchamiana, gdy użytkownik wybierze już opcje dla których chce przetestować agregat, na których podstawie metoda zwraca się do bazy danych o odpowiednie zasoby i wypisuje wynik zapytania.
- Agregaty - odpowiednio zaprogramowane klasy, które po zbudowaniu utworzyły w bazie danych agregaty. Klasy te składają się z następujących metod:
 - Init() - metoda uruchamiana tylko raz, na początku. Inicjalizuje dane.
 - Accumulate() - metoda zbierająca potrzebne dane z każdego z rekordów tabeli.
 - Merge() - metoda wywoływana, gdy SQL Server zdecyduje, by wykorzystać przetwarzanie równoległe do zakończenia tworzenia agregatu
 - Terminate() - metoda wykonywana na końcu, po przetworzeniu wszystkich wierszy. Wykonuje ostatnie operacje na danych i zwraca końcowy wynik.

- Read() - metoda do wczytywania serializowanych danych (tylko w wersji rozszerzonej agregatów tj. "Median", "Mode" oraz "TruncatedMean")
- Write() - metoda do zapisywania serializowanych danych (również tylko w wersji rozszerzonej agregatów)
- Testy jednostkowe - testy wykonane na tabeli "TestTable" z bazy danych sprawdzające działanie agregatów na prostych przykładach. Więcej informacji o testach jednostkowych projektu znajduje się w zakładce "Testy Jednostkowe"

Opis implementacji

Aplikacja została napisana w języku C# przy użyciu edytora Microsoft Visual Studio 2008. Z kolei baza danych została skonfigurowana przy pomocy skryptów napisanych w języku SQL oraz środowiska SQL Server Management Studio. 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

W ramach projektu dostępnych jest 5 testów jednostkowych. Każdy z nich sprawdza poprawność działania jednego z agregatów, porównując wynik wywołania agregatu na tabeli "TestTable" z wynikami obliczonymi ręcznie, na tych samych danych. Testy są metodami w ciele klasy "UnitTests" znajdującej się w pliku "UnitTests.cs". Każdy test opisany jest znacznikiem "TestMethod". Nazwy testów są intuicyjne, zawierają w sobie nazwę agregatu z dopiskiem "Test":

- GeoMeanTest
- MedianTest
- ModeTest
- RMSTest
- TruncatedMeanTest

Zaimplementowane testy wykonują się poprawnie:

Test run completed Results: 5/5 passed; Item(s) checked: 0					
	Result	Test Name	Project	Error Message	
<input type="checkbox"/>	Passed	MedianTest	Tests		
<input type="checkbox"/>	Passed	TruncatedMeanTest	Tests		
<input type="checkbox"/>	Passed	RMSTest	Tests		
<input type="checkbox"/>	Passed	GeoMeanTest	Tests		
<input type="checkbox"/>	Passed	ModeTest	Tests		

Kod źródłowy

Kod źródłowy znajduje się w katalogu z projektem i składa się z 4 katalogów:

- Aggregats - pliki obsługujące tworzenie agregatów w języku C#
- ConsoleApplication - pliki obsługujące główny program napisany w języku C#
- SqlScripts - skrypty SQL, które skonfigurowały bazę danych
- Tests - pliki obsługujące testy jednostkowe

Podsumowanie

Projekt pokazuje jak w prosty sposób można z poziomu aplikacji napisanej w języku C# połączyć się z bazą danych i wykonywać na niej nietrywialne operacje, do jakich z pewnością można zaliczyć definiowanie własnych agregatów. Zwraca to uwagę na możliwości jakie udostępnia nam środowisko CLR.

Bibliografia

- https://newton.fis.agh.edu.pl/~antek/read_pdf.php?file=BD2_L09_CLR.pdf
- <https://docs.microsoft.com>
- <https://stackoverflow.com/>