Akademia Górniczo-Hutnicza w Krakowie
Wydział Fizyki i Informatyki Stosowanej
Informatyka Stosowana
Bartosz Rogowski, III rok
19 czerwca 2021

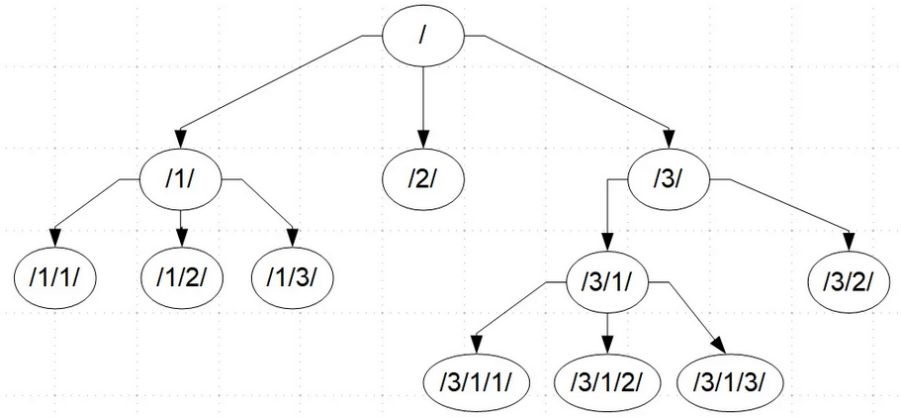
Bazy Danych 2

Przetwarzanie danych hierarchicznych (typ `hierarchyid`)

Spis treści

1 Opis problemu i opis funkcjonalności udostępnianej przez API

API (ang. *application programming interface*) zostało przygotowane tak, aby umożliwić obsługę struktury organizacyjnej firmy, która z założenia jest hierarchiczna. Dane pracowników stanowią elementy owej hierarchii, zaś sam jej mechanizm został oparty na typie `hierarchyid`, który został wprowadzony w 2008 roku w *SQL Server 2008* [1]. Reprezentacja tego typu po konwersji na tekst jest analogiczna do struktury drzewa (rys. 1)*, które posiada swój korzeń oraz potomków, a sama wartość `hierarchyid` odpowiada pojedynczemu wierzchołkowi [2].



Rysunek 1: Wizualizacja typu `hierarchyid` jako drzewa.

2 Opis typów danych oraz metod (funkcji) udostępnionych w ramach API

Dane pracowników przedsiębiorstwa są przechowywane na serwerze bazodanowym w bazie danych *Project_Hierarchy* w tabeli *Employees*. Każda kolumna nie może być pusta (`NOT NULL`).

nazwa	typ bazodanowy	typ w `C#`	znaczenie
Path	`hierarchyid UNIQUE`	`SqlHierarchyId`	typ `hierarchyid` pełniący rolę ID
Name	`varchar(100)`	`String`	imię oraz nazwisko
Position	`varchar(100)`	`String`	pełniona rola/pozycja
Start_Year	`int`	`int`	rok rozpoczęcia pracy
Salary	`numeric(7, 2)`	`float`	pensja (w dolarach)

Tabela 1: Struktura tabeli *Employees*.

Wartym wspomnienia jest fakt, że API jest wspomagane przez pomocniczą klasę `Employee`, która ma atrybuty analogiczne do tych z bazodanowej tabeli *Employees* i usprawnia proces wypisywania czy dodawania danych i zawiera jedną metodę – `public void Display()`, która służy wypisaniu danych w „ładny", sformatowany sposób (m. in. korzystając z metody `ToString()` dla zmiennej `hierarchyid`) oraz zawarciu dodatkowych informacji takich jak poziom węzła (liczony od korzenia – jako 0) czy liczba lat od rozpoczęcia pracy w firmie.

*źródło: `https://softwarehut.com/blog/tech/hierarchyid-entity-framework`; data dostępu: 18 czerwca 2021; UWAGA: Wartości w węzłach są już przekonwertowane na tekstową reprezentację za pomocą metody `ToString()`.

API jest natomiast zaimplementowane w języku `C#` jako klasa `Company` i posiada następujące:

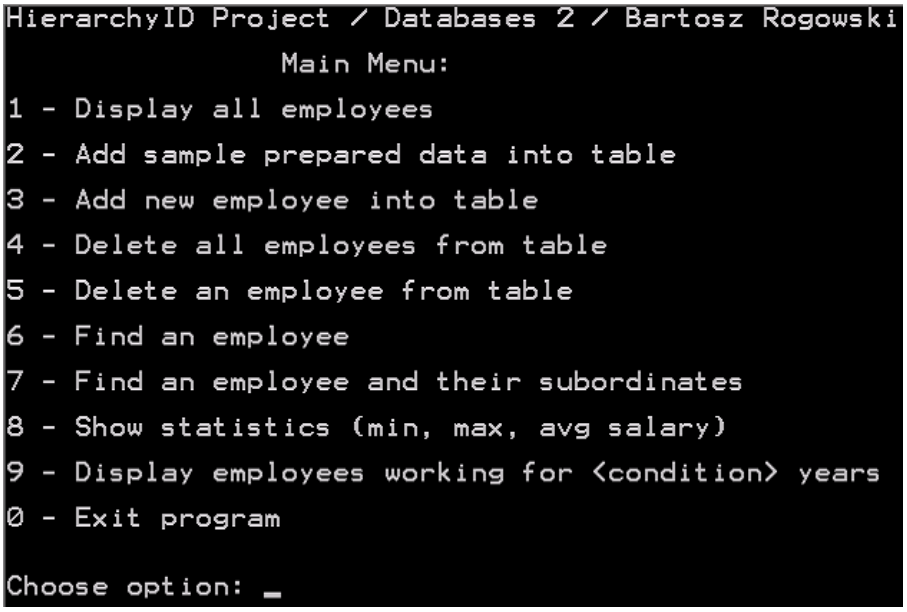
- atrybuty:
  - `private String connection_string` – zmienna określająca parametry połączenia z serwerem bazodanowym
  - `private List<Employee> employee_list` – lista przechowująca obiekty typu `Employee` (pracowników)
- metody:
  - `public List<Employee> GetEmployeeList()` – pomocnicza metoda zwracająca atrybut `employee_list`
  - `public void LoadAllEmployees()` – ładuje dane wszystkich pracowników z bazy do atrybutu `employee_list`
  - `private void ShowLegend()` – pomocnicza metoda, wyświetlająca w konsoli legendę (korzystają z niej metody wyświetlające dane pracowników)
  - `public void DisplayAllEmployees()` – wyświetla dane wszystkich pracowników
  - `public void AddNewEmployee(`
    ```
    String Path,
    String Name,
    String Position,
    int Start_Year,
    float Salary
    ```
    `)` – dodaje nowego pracownika do tabeli bazodanowej (oznaczenia jak w tab. 1)
  - `public void AddSampleData()` – dodaje do tabeli bazodanowej sztuczne przygotowane wcześniej dane tzw. *mock data*
  - `public void DeleteAllData()` – usuwa wszystkie dane z tabeli *Employees* uprzednio ostrzegając o nieodwracalności operacji
  - `public void DeleteEmployee(String Path)` – pozwala na usunięcie pracownika poprzez podanie jego hierarchii/ścieżki (np. `"/3/"`) w przypadku, gdy taki pracownik (wierzchołek) istnieje i nie ma pracowników od niego zależnych (potomków)
  - `public void FindEmployee(String Path)` – umożliwia wyszukanie i wypisanie danych pracownika poprzez podanie jego hierarchii/ścieżki (np. `"/3/"`)
  - `public void FindEmployeeWithSubordinates(String Path)` – działa analogicznie jak powyższa, z tym że wyszukiwani są także potomkowie takiego węzła (pracownicy zależni)
  - `public void DisplayStatisticsForPath(String Path)` – wyświetla statystyki (minimalna, maksymalna oraz średnia pensja) dla podanego pracownika (w tym także pracowników od niego zależnych) poprzez podanie jego hierarchii/ścieżki (np. `"/3/"`)
  - `public void DisplayStatisticsForLevel(int Level)` – metoda analogiczna do powyższej z tym, że statystyki podawane są dla pracowników (węzłów, bez potomków – „poziome” wyszukiwanie) z tego samego poziomu
  - `public void DisplayEmployeesWorkingFor(char sign, int number)` – wyświetla dane pracowników pracujących dłużej niż/krócej niż/dokładnie (specyfikacja za pomocą znaków nierówności bądź równości – kolejno: `>`/`<`/`=`) `number` lat (np. wywołanie metody z argumentami `'>', 4` wypisze pracowników pracujących dłużej niż 4 lata).

3 Opis implementacji

Metody API za każdym razem otwierają połączenie z bazą danych (co zapobiega długim połączeniom, które są mało efektywne pod względem stabilności i wydajności [3]), a następnie wykonują zapytania bazodanowe `SQL` – a więc działają bezpośrednio na rekordach tabeli – które (najczęściej) zwracają określone wyniki.

Interaktywna aplikacja konsolowa umożliwia korzystanie z omówionych w poprzednim rozdziale funkcjonalności. Główne menu aplikacji (rys. 2) umożliwia:

1. wyświetlenie wszystkich rekordów w tabeli bazodanowej
2. dodanie do tabeli bazodanowej przykładowych danych
3. dodanie nowego użytkownika (poprzez podanie każdego atrybutu)
4. usunięcie wszystkich rekordów z tabeli
5. usunięcie wskazanego poprzez ścieżkę pracownika z tabeli
6. wyszukanie danych pracownika wskazanego poprzez ścieżkę
7. wyszukanie danych pracownika wskazanego poprzez ścieżkę wraz z wszystkimi jego podwładnymi
8. wyświetlenie statystyk dotyczących pensji (minimalna, maksymalna, średnia) dla danego pracownika oraz jego podwładnych lub dla danego poziomu
9. wyświetlenie danych pracowników pracujących przez <warunek> lat
0. wyjście z programu



Rysunek 2: Zrzut ekranu z konsoli – interaktywne menu główne.

Ponadto kod został napisany tak, aby zapewnić podstawową obsługę błędów/wyjątków.

```
Legend:
        L - Level
        WS - Working since [year]
        WF - Working for [in years]

Path      L     Name                        Position                          WS       WF   Salary $
-------------------------------------------------------------------------------------------------------
/         0     Taylor Alison Swift         CEO                               2007     14      13000
/1/       1     Billie Eilish               Financial Manager                 2015      6       8000
/1/1/     2     Finneas O'Connell           Financial Senior Manager          2015      6       6000
/2/       1     Tyler Joseph                HR Manager                        2010     11       7000
/2/1/     2     Joshua Dun                  HR Assistnant Manager             2011     10       5000
/2/1/1/   3     Dan Smith                   HR Assistnant                     2012      9       3000
/2/1/2/   3     Josh Taylor                 HR Assistnant                     2018      3       2200
/3/       1     Selena Gomez                Administration Manager            2009     12       8000
/4/       1     Ebba Tove Elsa Nilsson      Technical Manager                 2013      8       7000
/4/1/     2     Alma-Sofia Miettinen        Technical Senior Manager          2016      5       6000
/4/1/1/   3     Zara Maria Larsson          Technical Assistant Manager       2016      5       4000

Press any key to go back
_
```

Rysunek 3: Zrzut ekranu z konsoli – wyświetlenie przykładowych danych użytkowników.

4 Prezentacja przeprowadzonych testów jednostkowych

Testy zostały przygotowane w oparciu o narzędzia do tworzenia testów jednostkowych w IDE *Visual Studio 2008* z pomocą oficjalnej dokumentacji *Microsoftu* [4, 5]. Pokrywają one przetestowanie praktycznie wszystkich metod z API, łącznie z przypadkami, gdzie mogą zostać rzucone wyjątki. Zrzut ekranu prezentujący poprawnie wykonane testy znajduje się na rys. 4.

Test run completed Results: 14/14 passed; Item(s) checked: 0

Result	Test Name	Project
Passed	DeleteEmployeeTest_ShouldThrowExceptionMessag	TestProject
Passed	AA_LoadAllEmployeesTest_ShouldHaveCount11	TestProject
Passed	AddNewEmployeeTest_ShouldThrowSqlException	TestProject
Passed	AddNewEmployeeTest_Right	TestProject
Passed	FindEmployeeTest_NonExistingEmployee	TestProject
Passed	FindEmployeeWithSubordinatesTest_NonExistingEm	TestProject
Passed	DisplayEmployeesWorkingForTest_ShouldHaveCour	TestProject
Passed	FindEmployeeTest_ExistingEmployee	TestProject
Passed	FindEmployeeWithSubordinatesTest_ExistingEmploy	TestProject
Passed	DisplayEmployeesWorkingForTest_ShouldHaveCour	TestProject
Passed	DisplayStatisticsForPathTest	TestProject
Passed	Z_DeleteEmployeeTest	TestProject
Passed	DisplayStatisticsForLevelTest	TestProject
Passed	DisplayEmployeesWorkingForTest_ShouldHaveCour	TestProject

Rysunek 4: Zrzut ekranu z *Visual Studio 2008* – zdane testy.

5 Podsumowanie, wnioski

Typ `hierarchyid` jest prosty w użyciu, doskonale nadaje się do danych wykazujących własności hierarchiczne oraz posiada zestaw przydatnych metod (takich jak: `GetLevel()` czy `IsDescendantOf()`), które w przypadku tworzenia własnej wersji tego typu, musiałyby zostać zaimplementowane samodzielnie. Nie mniej jednak „interfejs” ten nie jest idealny, czego przykładem może być konieczność konwersji do reprezentacji tekstowej (`ToString()`).

6 Literatura

[1] wielu autorów. *Tutorial: Using the hierarchyid Data Type*. URL: https://docs.microsoft.com/pl-pl/sql/relational-databases/tables/tutorial-using-the-hierarchyid-data-type?view=sql-server-2017. (data dostępu: 17 czerwca 2021).

[2] wielu autorów. ***hierarchyid** data type method reference*. URL: https://docs.microsoft.com/en-us/sql/t-sql/data-types/hierarchyid-data-type-method-reference?view=sql-server-ver15. (data dostępu: 17 czerwca 2021).

[3] URL: https://stackoverflow.com/questions/312702/is-it-safe-to-keep-database-connections-open-for-long-time. (data dostępu: 18 czerwca 2021).

[4] wielu autorów. *Console Unit Testing*. URL: https://docs.microsoft.com/en-us/archive/blogs/ploeh/console-unit-testing. (data dostępu: 12 czerwca 2021).

[5] wielu autorów. *Przewodnik: Tworzenie i uruchamianie testów jednostkowych dla kodu zarządzanego*. URL: https://docs.microsoft.com/pl-pl/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019. (data dostępu: 12 czerwca 2021).

W trakcie tworzenia projektu posługiwano się również materiałami dydaktycznymi przygotowanymi przez Prowadzących w ramach przedmiotu *Bazy Danych 2*.

7 Kod źródłowy

Skrypt `SQL` potrzebny do stworzenia bazy danych oraz tabeli znajduje się w głównym folderze i nosi nazwę *prepare_database.sql*. Pliki `C#` zawierające klasy *Company*, *Employee* oraz *Program* znajdują się w folderze `Rogowski_Hierarchy_Project/Rogowski_Hierarchy_Project`; natomiast klasa *UnitTests* zawierająca testy API – w `Rogowski_Hierarchy_Project/TestProject`.