Justyna Pyrcz

DAX w Power BI

PODSTAWY

Podręcznik z ćwiczeniami

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/daxpob
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresem:
https://ftp.helion.pl/przyklady/daxpob.zip

ISBN: 978-83-289-1608-1

Printed in Poland.

Spis treści

ROZDZIAŁ 3.

Funkcje agregujące i iteracyjne

Funkcje agregujące SUM, AVERAGE, COUNT, MIN, MAX

Tworząc miarę `BWRezerwacji`, użyliśmy funkcji `SUM`, która miała za zadanie zsumowanie wartości z kolumny `BWartość`. `SUM` to zwykła funkcja agregująca, która przyjmuje jako argument jedno odwołanie do kolumny i na tej jednej kolumnie dokonuje obliczenia. DAX ma kilka innych funkcji agregujących, takich jak `AVERAGE` (oblicza średnią arytmetyczną wartości) czy `COUNT` (zlicza wartości), które również wymagają jako argumentu jednego odwołania do kolumny.

Zatem jeśli będziemy chcieli obliczyć średnią wartość rezerwacji, zdefiniujemy prostą miarę (formuła 3.1).

FORMUŁA 3.1. Formuła miary obliczającej średnią wartość rezerwacji

```
SrWRezerwacji = AVERAGE(Rezerwacje[BWartość])
```

a jeśli potrzebne będzie zliczenie wartości w tej kolumnie, formuła będzie równie oczywista (formuła 3.2).

FORMUŁA 3.2. Formuła miary zwracającej liczbę wartości w kolumnie BWartość

```
LWRezerwacji = COUNT(Rezerwacje[BWartość])
```

Podobnie wygląda użycie funkcji `MAX` i `MIN`, jednak w nowych wersjach Power BI mają one dwa tryby działania. Można za ich pomocą agregować dane z jednej kolumny, czyli np. dzięki mierze pokazanej w formule 3.3 dowiemy się, jaka jest największa wartość rezerwacji.

FORMUŁA 3.3. Formuła miary zwracającej największą wartość rezerwacji

```
MaxWRezerwacji = MAX(Rezerwacje[BWartość])
```

Możemy także za pomocą tych funkcji porównać dwie wartości skalarne, np. w wyniku obliczenia miary pokazanej w formule 3.4 otrzymamy większą z dwóch wartości — średniej i maksymalnej rezerwacji.

FORMUŁA 3.4. Formuła miary zwracającej większy z wyników dwóch miar

```
PorownanieMaxWRiSrWR = MAX([MaxWRezerwacji],[SrWRezerwacji])
```

A w kolumnie kalkulowanej zdefiniowanej formułą 3.5 uzyskamy w każdym wierszu większą ze znajdujących się w nim liczb z kolumn `Cena 1 noc` i `BWartość`.

FORMUŁA 3.5. Formuła kolumny kalkulowanej zwracająca większą z wartości z tego samego wiersza

```
PorownanieCena1NiBWartość =
MAX(Rezerwacje[Cena 1 noc],Rezerwacje[BWartość])
```

Rzecz jasna, wyniki ostatnich dwóch formuł są oczywiste i zbyteczne w analizie, ale nie o wyniki tu chodzi, lecz o zaprezentowanie funkcji z wykorzystaniem tych danych, które mamy w tabelach.

Funkcje iteracyjne agregujące SUMX, AVERAGEX, COUNTX, MINX, MAXX

Opisane uprzednio funkcje agregujące są proste w użyciu, ale trzeba pamiętać, że ich rolą jest agregowanie wyłącznie wartości znajdujących się w istniejącej kolumnie. Nie sprawdzą się w sytuacji, w której nie będziemy chcieli tworzyć kolumny z wartościami pojedynczych rezerwacji (takiej jak `BWartość`), lecz od razu uzyskać miarę obliczającą łączną wartość rezerwacji z wykorzystaniem danych z kolumn `Liczba nocy` i `Cena 1 noc`. Nie sprawdzi się w tej sytuacji również miara z formułą identyczną z tą z kolumny kalkulowanej `BWartość`.

Zobaczmy, jakie są skutki takich prób.

Jeśli w definicji miary umieścimy działanie mnożenia liczby nocy przez cenę jednostkową (formuła 3.6):

FORMUŁA 3.6. Próba utworzenia miary zakończona BŁĘDEM

```
Miara = Rezerwacje[Liczba nocy]*Rezerwacje[Cena 1 noc]
```

to w efekcie uzyskamy komunikat o błędzie:

```
Nie można określić pojedynczej wartości kolumny „Liczba nocy" w tabeli „Rezerwacje".
Może się tak zdarzyć, gdy formuła miary odwołuje się do kolumny zawierającej wiele
wartości bez określania agregacji, takiej jak min, max, count lub sum, w celu
uzyskania pojedynczego wyniku.
```

Przyczyną błędu jest to, że w formule znajdują się odwołania do kolumn, które powinny zwrócić pojedyncze wartości z konkretnego wiersza (te wartości miałyby być przez siebie pomnożone), czyli jest to składnia właściwa dla kontekstu wiersza. Miara jest obliczana w kontekście filtra, zatem potrzebuje agregacji wielu wartości, aby w wyniku zwrócić jedną. Stąd w treści komunikatu o błędzie znalazła się wzmianka o potrzebie określenia agregacji.

Niezależnie od komunikatu o błędzie, który wyjaśnia problem od strony technicznej, w formule brakuje też logiki. Nie chcemy przecież definiować obliczenia, które mnożyłoby wartości z jednego wiersza, lecz takie, które pomnoży te wartości we wszystkich wierszach tabeli i zsumuje wyniki tych działań.

Użyjmy więc agregacji i umieśćmy formułę mnożenia wewnątrz funkcji `SUM` (formuła 3.7).

FORMUŁA 3.7. Zakończona BŁĘDEM próba umieszczenia wyrażenia arytmetycznego w funkcji SUM

```
Miara = SUM(Rezerwacje[Liczba nocy]*Rezerwacje[Cena 1 noc])
```

Znów otrzymujemy komunikat o błędzie:

```
Funkcja SUM akceptuje jako argument tylko odwołanie do kolumny.
```

Tym razem błąd wynika ze znanych nam już ograniczeń funkcji agregujących.

Widzimy więc, że próby wykorzystania używanych wcześniej formuł i funkcji nie dają pożądanego efektu. Musimy uciec się do innego rozwiązania i tu przyjdą nam z pomocą funkcje agregujące iteracyjne, które iterują tabelę, czyli „wchodzą" do każdego jej wiersza, aby wykonać w nim określoną operację, a wyniki operacji wykonanej we wszystkich wierszach zagregować. Funkcje, o których mowa, to: `SUMX`, `AVERAGEX`, `COUNTX`, `MAXX`, `MINX`. Wszystkie mają podobną składnię:

```
FunkcjaX(Tabela, Wyrażenie)
```

Pierwszym ich argumentem jest tabela, która ma być iterowana, czyli przeglądana wiersz po wierszu, a drugim wyrażenie, czyli formuła, która ma być wykonana w każdym z tych wierszy. To, w jaki sposób wyniki tej formuły będą agregowane, zależy od użytej funkcji — `SUMX` wyniki z wierszy zsumuje, a `AVERAGEX` zwróci ich średnią arytmetyczną.

W naszym przypadku, aby obliczyć łączną wartość rezerwacji, użyjemy funkcji `SUMX`, której pierwszym argumentem będzie tabela `Rezerwacje`, a drugim iloczyn liczby nocy i ceny jednostkowej. Zmodyfikujmy zatem wcześniej utworzoną miarę `BWRezerwacji` (formuła 3.8).

FORMUŁA 3.8. Formuła miary z funkcją iteracyjną

```
BWRezerwacji =
SUMX(Rezerwacje,Rezerwacje[Liczba nocy]*Rezerwacje[Cena 1 noc])
```

Jak widać w raporcie (rysunek 3.1), wyniki nowo skonstruowanej, niekorzystającej z kolumny kalkulowanej `BWartość` miary nie zmieniły się, czyli osiągnęliśmy zamierzony efekt.

Kategoria	Liczba elementów Nr rezerwacji	Suma elementów BWartość	Suma elementów BWartość2	BWRezerwacji
komfort	324	558 450	558 450	558 450
luksus	208	328 040	328 040	328 040
standard	1028	1 745 590	1 745 590	1 745 590
standard plus	678	1 135 770	1 135 770	1 135 770
Suma	**2238**	**3 767 850**	**3 767 850**	**3 767 850**

RYSUNEK 3.1. Zmodyfikowana miara BWRezerwacji w podziale na kategorie.

Sprawdziliśmy w ten sposób poprawność obliczenia wykonanego za pomocą funkcji `SUMX`, ale ponieważ działanie iteratorów nie zawsze jest dla nas całkowicie jasne, przeanalizujmy ten przypadek jeszcze raz. Weźmy pod uwagę miarę `BWRezerwacji` obliczoną w kontekście filtra `komfort`.

Pamiętamy, że miara wymaga kontekstu filtra i w nim jest obliczana. Kiedy użyliśmy w mierze funkcji SUM z kolumną Rezerwacje[BWartość], działanie filtra z kontekstu było jasne — w kolumnie BWartość poprzez filtr komfort działający na tabelę Rezerwacje były widoczne tylko wartości rezerwacji pokojów komfortowych (jak widać na rysunku 3.1, jest ich 324), więc tylko te wartości zostały zsumowane. Czy w przypadku nowej miary z użyciem funkcji SUMX filtr z kontekstu działa tak samo? Choć może na pierwszy rzut oka tego nie widać, to odpowiedź brzmi: tak, filtr tak samo działa na tabelę Rezerwacje (pierwszy argument funkcji), więc podczas iteracji brane są pod uwagę tylko 324 wiersze z rezerwacjami pokojów kategorii komfort i tylko w tych 324 wierszach wykonywana jest formuła umieszczona w drugim argumencie funkcji. Warto w tym miejscu zauważyć, że formuła ta (Rezerwacje[Liczba nocy] * Rezerwacje[cena 1_noc]) jest obliczana w kontekście wiersza — każdego wiersza przefiltrowanej tabeli, dlatego możemy w niej użyć odwołań do kolumn bez żadnych agregacji.

Omówiony przykład dotyczył obliczenia, w którym wykorzystaliśmy kolumny z tej samej tabeli. Jak wyglądałaby ta miara, gdybyśmy chcieli do obliczenia wartości użyć ceny z tabeli Pokoje? Różniłaby się wyrażeniem umieszczonym w drugim argumencie funkcji SUMX — to byłoby to samo wyrażenie, którego użyliśmy w formule kolumny kalkulowanej BWartość2 (formuła 2.6). Zdefiniujmy więc tę miarę w sposób pokazany w formule 3.9.

FORMUŁA 3.9. Formuła miary z iteratorem i wyrażeniem odwołującym się do kolumn z dwóch tabel

```
BWRezerwacji2 =
SUMX(Rezerwacje,Rezerwacje[Liczba nocy]*RELATED(Pokoje[Cena]))
```

Zwróćmy uwagę na to, że w mierze BWRezerwacji2 użyliśmy funkcji RELATED, o której wiemy, że działa w kontekście wiersza. Mogliśmy tak zrobić, bo mimo że miara jest obliczana w kontekście filtra (filtrowana jest tabela będąca pierwszym argumentem funkcji), to wyrażenie stanowiące drugi argument funkcji SUMX jest obliczane w kontekście wiersza (możemy powiedzieć, że z każdego wiersza iterowanej tabeli Rezerwacje sięgamy po powiązaną z nim cenę do tabeli Pokoje).

Wiemy już, jak używać funkcji iteracyjnej SUMX. W podobny sposób używamy innych iteratorów. Na przykład, jeśli chcemy obliczyć średnią arytmetyczną wartości rezerwacji, zdefiniujemy miarę pokazaną w formule 3.10.

FORMUŁA 3.10. Formuła miary z iteracyjną funkcją AVERAGEX

```
SrWRezerwacji =
AVERAGEX(Rezerwacje,Rezerwacje[Liczba nocy]*Rezerwacje[Cena 1 noc])
```

a jeśli interesuje nas największa wartość rezerwacji, miara będzie wyglądała jak w formule 3.11.

FORMUŁA 3.11. Formuła miary z iteracyjną funkcją MAXX

```
MaxWRezerwacji =
MAXX(Rezerwacje,Rezerwacje[Liczba nocy]*Rezerwacje[Cena 1 noc])
```

WSKAZÓWKA Jeśli nie mamy pewności co do sposobu konstruowania funkcji iteracyjnej — nie wiemy, która tabela powinna być iterowana lub jakie wyrażenie należy określić w jej drugim argumencie, pomocne może się okazać przemyślenie obliczenia tak, jak gdyby miało być zdefiniowane w kolumnie. Czyli warto sprowadzić problem do pytań: jakie działanie i w której tabeli należy wykonać. Odpowiadając sobie na te pytania, znajdujemy argumenty funkcji. Tabela będzie tą, która ma być poddana iteracji, czyli zostanie użyta jako pierwszy argument, a działanie tym, które należy wykonać w każdym wierszu tej tabeli, czyli powinno zostać umieszczone w drugim argumencie funkcji.

Kiedy chcieliśmy obliczyć wartość poszczególnych rezerwacji, jasne było, że aby to zrobić, należy dodać kolumnę do tabeli `Rezerwacje`, a nie do żadnej innej. Więc to tabela `Rezerwacje` jest właściwą do umieszczenia w pierwszym argumencie funkcji iteracyjnej.

Oczywiste było również działanie definiowane w nowej kolumnie — dla każdej rezerwacji mnożyliśmy liczbę nocy przez cenę pokoju. Ta formuła działała poprawnie w kolumnie i tej formuły użyliśmy w drugim argumencie funkcji iteracyjnej.

UWAGA **Stosowanie miar w drugim argumencie funkcji iteracyjnej.**

Użycie miary w iteracji wydłuża czas wykonywania operacji.

Użycie miary w iteracji w tabeli, której wiersze nie są unikalne, daje błędne wyniki. Dlatego nie należy umieszczać miar w iteracjach tabel, co do których nie mamy pewności, że zawierają i będą zawierały wyłącznie unikalne wiersze. Przykładem tabeli, co do której nie możemy mieć takiej pewności, jest tabela faktów `Rezerwacje`.

Więcej informacji na ten temat znajduje się w podrozdziale „Przejście kontekstu” w rozdziale 5.

Funkcja DISTINCTCOUNT

Funkcja `DISTINCTCOUNT` również jest funkcją agregującą, ale nie ma swojego odpowiednika wśród iteratorów, dlatego omawiam ją osobno.

Funkcja ta służy do zliczania odrębnych wartości w kolumnie.

Jeśli na przykład będziemy chcieli się dowiedzieć, ile pokojów z całego hotelu było kiedykolwiek rezerwowanych, to będziemy musieli zliczyć pokoje znajdujące się w kolumnie `Pokój` w tabeli `Rezerwacje`. Ale użycie funkcji `COUNT` nie będzie tu właściwe, ponieważ `COUNT` zlicza wszystkie wartości z kolumny, każde ich powtórzenie, czyli zwróci liczbę rezerwacji, a nie pokojów. Odpowiednią w tym przypadku funkcją będzie `DISTINCTCOUNT`, która policzy każdą występującą w kolumnie wartość, czyli każdy pokój, tylko raz.

Sprawdźmy to i utwórzmy dwie miary — jedną z funkcją `COUNT` (formuła 3.12), a drugą z `DISTINCTCOUNT` (formuła 3.13).

FORMUŁA 3.12. Formuła miary z funkcją COUNT zliczającą wszystkie wartości w kolumnie

```
LPokoje = COUNT(Rezerwacje[Pokój])
```

FORMUŁA 3.13. Formuła miary z funkcją DISTINCTCOUNT zliczającą wartości odrębne

```
LPokoje_odrebne = DISTINCTCOUNT(Rezerwacje[Pokój])
```

Umieśćmy nowe miary w raporcie w tabeli z kategoriami w wierszach.

Tabela pokazuje wyniki obliczenia nowych miar dla poszczególnych kategorii (rysunek 3.2). Miara `LPokoje`, w której użyliśmy funkcji `COUNT`, daje ten sam wynik, który otrzymaliśmy, licząc wartości w kolumnie `Nr rezerwacji`. Miara `LPokoje_odrebne` z funkcją `DISTINCTCOUNT` pokazuje zaś liczbę hotelowych pokojów, które były rezerwowane przez klientów.

Kategoria	Liczba elementów Nr rezerwacji	LPokoje	LPokoje_odrebne
komfort	324	324	21
luksus	208	208	13
standard	1028	1028	55
standard plus	678	678	46
Suma	**2238**	**2238**	**135**

RYSUNEK 3.2. Miary zliczające wszystkie wartości i wartości odrębne

Powtórka

Przeczytaj poniższe stwierdzenia i określ, które z nich są prawdziwe, a które fałszywe.

1. Argumentem funkcji agregującej `AVERAGE` może być odwołanie do jednej kolumny lub wyrażenie arytmetyczne. (P/F)
2. `MINX` i `MAXX` są funkcjami iteracyjnymi. (P/F)
3. Tabela określona w pierwszym argumencie iteratora musi zawierać wszystkie kolumny użyte w wyrażeniu znajdującym się w drugim argumencie funkcji. (P/F)
4. Formuła umieszczona w drugim argumencie iteratora jest obliczana w kontekście filtra. (P/F)
5. Funkcja `COUNT` zwraca liczbę odrębnych wartości w kolumnie. (P/F)

Ćwiczenia

Przejrzyj zawartość wszystkich tabel w modelu danych oraz ich opisy (wprowadzenie w rozdziale 1.) i nie korzystając z dotychczas zdefiniowanych miar lub kolumn, przygotuj miary obliczające:

1. Łączną liczbę nocy, na które zarezerwowano pokoje.
2. Najniższą cenę pokoju za jedną noc.
3. Liczbę dni, w których dokonano jakichkolwiek rezerwacji.
4. Łączną wartość opłaty dodatkowej dokonanych rezerwacji.
5. Łączną wartość rezerwacji wraz z opłatą dodatkową.
6. Średnią liczbę nocy, na które rezerwowano pokoje.
7. Średnią wartość opłaty dodatkowej z rezerwacji.
8. Najwyższą wartość opłaty dodatkowej z rezerwacji.

Poznaj język, za pomocą którego analitycy sprawiają, że dane są posłuszne, a raporty ciekawsze!

Język DAX (ang. Data Analysis Expressions) został stworzony przez firmę Microsoft, by umożliwić wykonywanie efektywnych obliczeń na danych przechowywanych w wielu tabelach i przetwarzanych w Power BI, Excel Power Pivot czy SQL Server Analysis Services. Możliwości DAX-a są naprawdę ogromne, osoba, która go opanuje, zyska potężnego sprzymierzeńca w pracy z wielkimi zbiorami danych ukrytymi pod wyświetlanymi w raportach wykresami czy tabelami.

Ten podręcznik wyjaśnia podstawy działania języka DAX i stosowania go w Power BI. Przystępnie napisany, uczy obsługi prostych modeli danych i korzystania z szeregu podstawowych, choć nieoczywistych funkcji. Ich opanowanie pozwala zacząć samodzielnie tworzyć ciekawe analizy, a równocześnie oswoić się z DAX-em i modelem danych.

Dzięki lekturze książki i wykonaniu zawartych w niej ćwiczeń dowiesz się między innymi:

- Czym jest model danych
- Do czego służą kolumny kalkulowane, miary i co to są konteksty wykonania
- Jak działają funkcje agregujące i iteracyjne
- Jak używać CALCULATE — najważniejszej funkcji języka DAX
- Do czego można użyć funkcji tablicowych
- Jak w prosty sposób tworzyć dynamiczne komunikaty

Justyna Pyrcz

Specjalizuje się w analizie danych. Z Power BI i języka DAX korzysta w codziennej pracy analityka. Jest również doświadczoną trenerką, prowadzi szkolenia i konsultacje z zakresu analizy danych z użyciem wymienionych narzędzi.

Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-1608-1

9 788328 916081

Cena: 39,90 zł