IT a anatomie firmy

(DAX, Data Analysis Expressions)

(pracovní dokument)

MBI tým

VŠE Praha, 2025

!!

Poznámka: Pro velkou část příkladů a schémat v této publikaci ***byly využity školicí materiály společnosti Seyfor*** s laskavým svolením autorů těchto materiálů.

Text se zaměřuje především na ***podstatu funkcí*** a jejich využití, řešení dílčích a ***specifických aspektů*** funkcionality ponecháváme s ohledem na rychlý vývoj časté změny v produktech ***na firemních tutoriálech***.

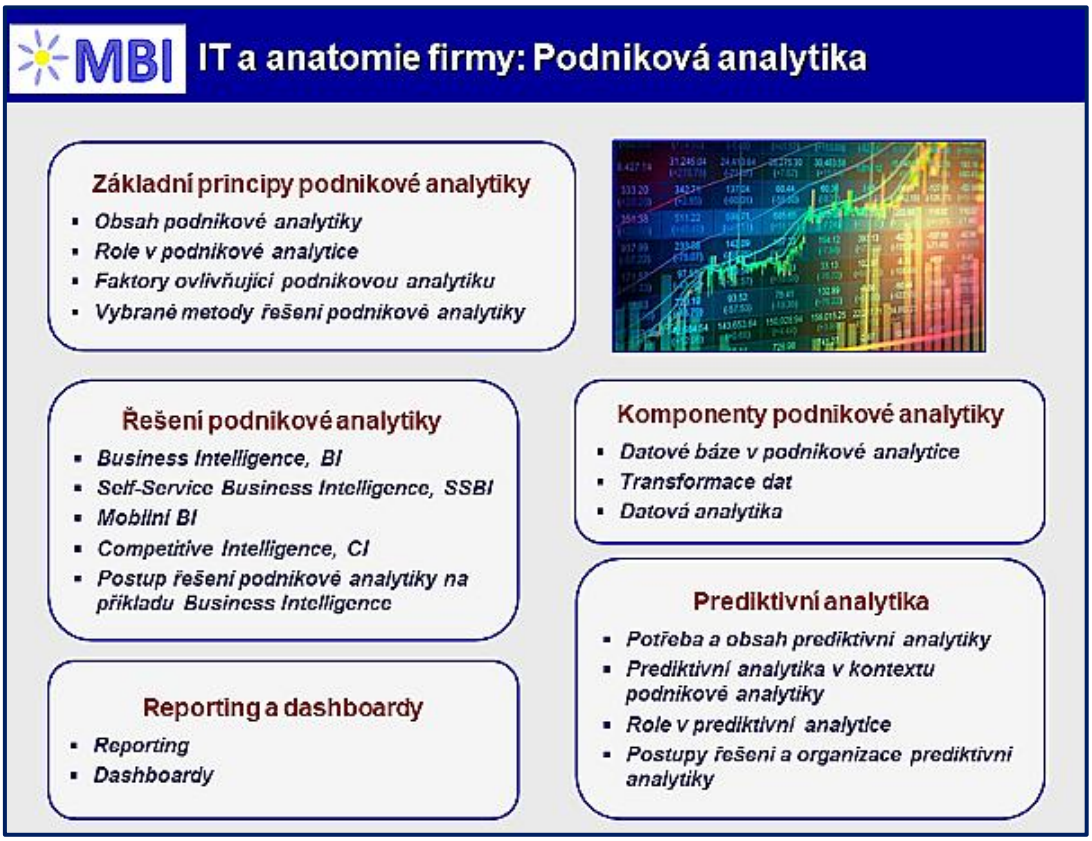
Obsah

Vazby a souvislosti s dalšími dokumenty řady „IT a anatomie firmy“

S ohledem na značný rozsah funkcionality a dalších momentů spojených s uvedenými produkty jsou tyto funkce ***obsahem i dalších publikací*** uvedené řady, které s nimi souvisejí***.***

Podniková analytika

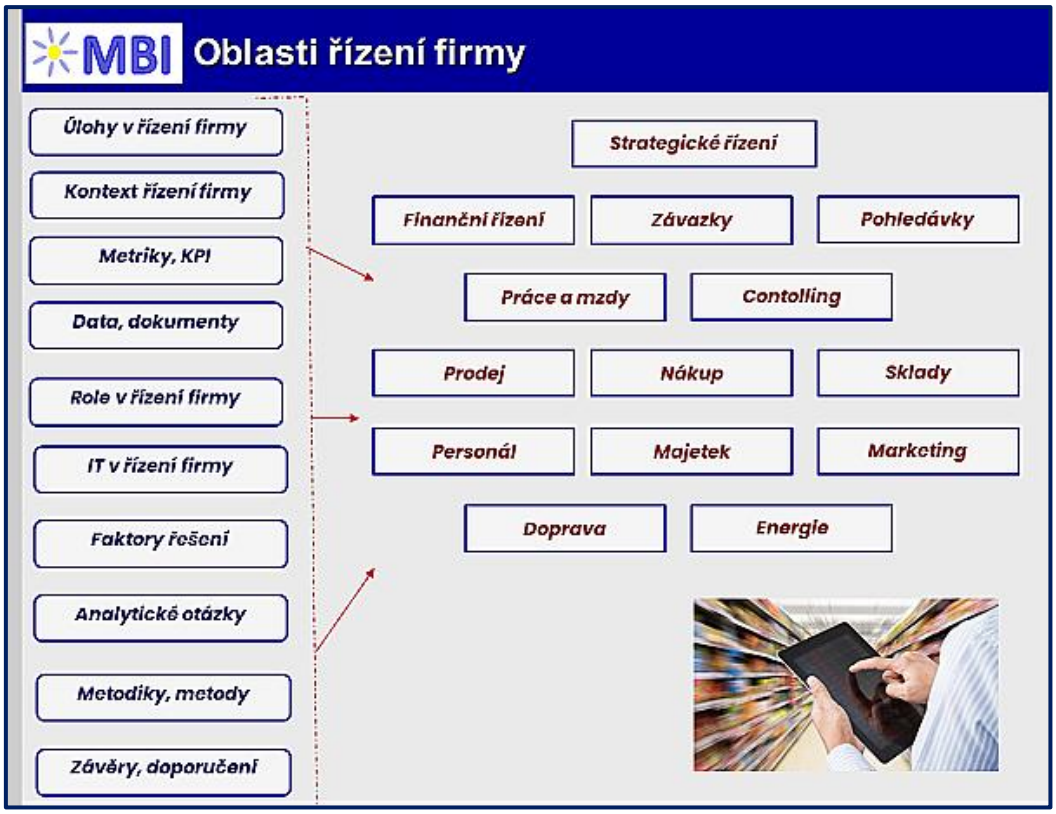
Publikace „*Podniková analytika*“ je ***základním dokumentem,*** který se snaží poskytnout ***celkový***, byť relativně stručný, ***přehled*** o principech, postupech, produktech, problémech i řešeních podnikové analytiky v praxi. Zahrnuje otázky jak „***deskriptivní analytiky***“ postavené většinou na nástrojích a přístupech business intelligence, self sertvice business intelligence nebo competitive intelligence, tak ***pokročilé, zejména prediktivní analytiky***.



Podniková analytika, struktura publikace

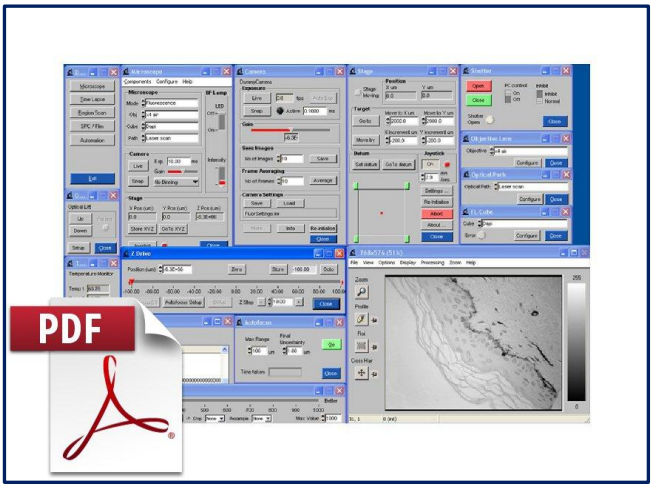
Oblasti a komponenty řízení

Publikace vymezuje obsah řízení firmy z analytického pohledu, pohledu komponent řízení, tj. jednotlivých úloh řízení, datových zdrojů, metrik, IT aplikací a dalších. Jednou ze součástí úloh jsou i analytické úlohy, tedy prostor pro uplatnění Power BI a dalších produktů, jak ukazuje další obrázek.



Oblasti a komponenty řízení, struktura textu

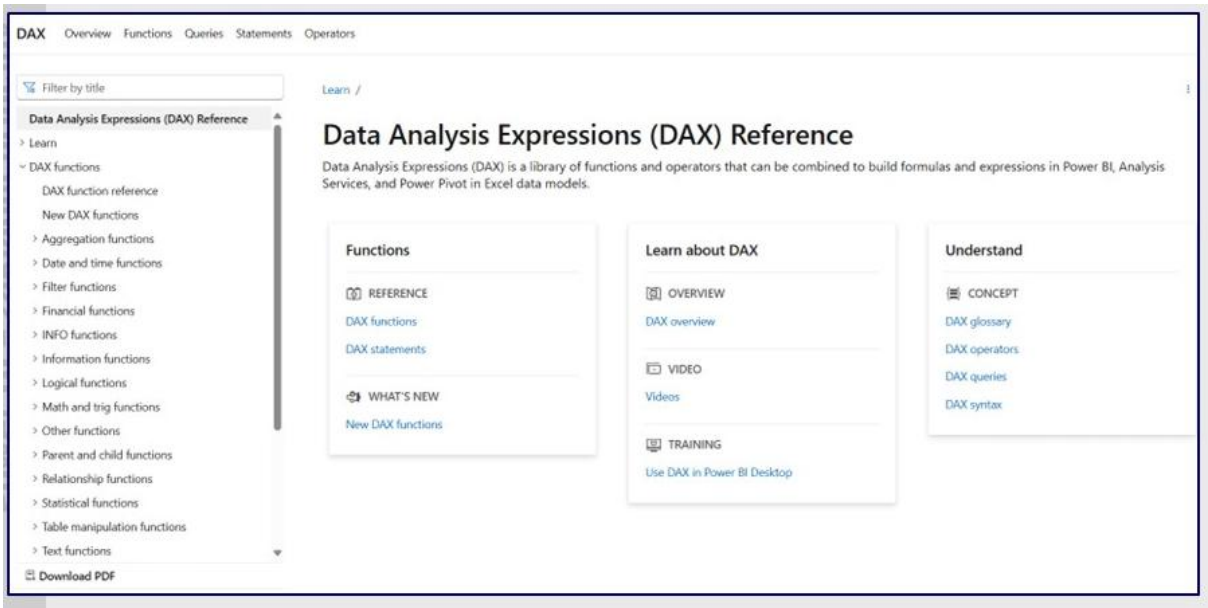
Další oddíly a kapitoly se již věnují výše uvedeným třem produktům.



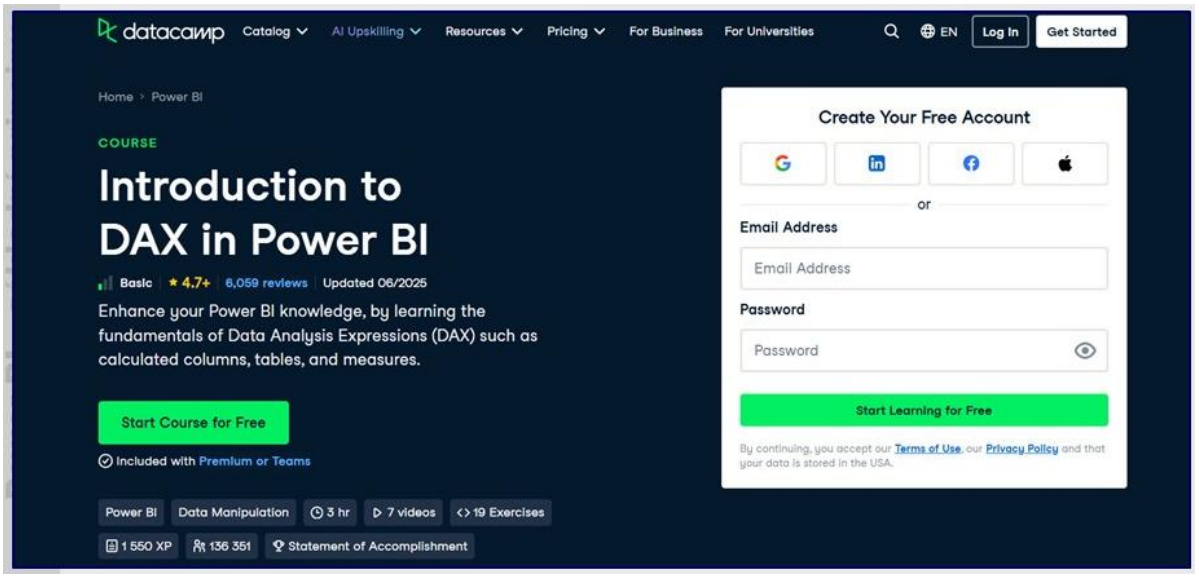
[1] Základní principy a užití jazyka DAX ***(Základní vymezení pravidel, výpočetní předpis, vytvoření kalkulovaného sloupce a vytvoření kalkulované míry)***	
[2] Konstrukce jazyka DAX	
[3] Kontext vyhodnocování výpočetních předpisů	**[4] Kalkulované sloupce a kalkulované míry**
[5] Řešení vazeb v datech	**[6] Time Intelligence**

Další text oddílu věnovaný jazyku DAX je shrnutím a prezentací principů a možností tohoto prostředku. Doplňující informace a detaily nabízejí vybrané aktuálně ***dostupné tutoriály***:

Jazyk DAX, přehled: [https://learn.microsoft.com/en-us/dax/]



DAX na systému DataCamp: [Introduction to DAX in Power BI Course | DataCamp]



1. Základní principy a užití DAX

Účelem této vstupní kapitoly je vytvořit pouze ***ve stručné formě předpoklad*** pro vytváření kalkulací s pomocí jazyka DAX, ***bez dalších detailnějších informací***. K těm se vrátíme v dalších kapitolách, zejména k vytváření základních metrik pro jednotlivé oblasti řízení jako obejm prodeje, náklady na prodej a další, na které pak naváží kalkulace na bázi Time intelligence jako YTD (*year-to-date*), YOY (*year-over-year*) a další.

DAX je dotazovací i funkční jazyk. Je to efektivní jazyk ***pro práci s multidimenzionálně organizovanými a uloženými daty***. Kromě prostředků pro vytváření analytických aplikací umožňuje uživateli lépe a detailněji pochopit některé důležité principy pro práci v multidimenzionálním prostředí, které mohou být užitečné nejen při vytváření aplikací v Power BI, ale i při návrzích aplikací realizovaných v jiných produktech.

Na druhou stranu je dobré uvést, že v souvislosti s charakterem Self Service BI aplikací a orientací na samostatnou práci koncových uživatelů, mohou být některé funkce a postupy náročnější. Proto je obsahem tohoto oddílu dokumentu určitý ***širší základ jazyka DAX,*** který tvoří centrální část řešení v PBI s tím, že je na uživateli, co nakonec pro svoji práci využije, co bude skutečně potřebovat a co nikoli. Navíc je ověřená zkušenost, která platí nejenom pro DAX, že si každý uživatel postupně ověřuje různé možnosti a funkce a tím i rozšiřuje škálu svých možností využití daného prostředku.

DAX je charakterizován jako ***funkční programovací jazyk***, což znamená, že výpočty primárně využívají funkce, které generují výsledky. Disponuje aktuálně přes ***200 funkcí*** a dál se rozvíjí. Každá kalkulace využívá jednu nebo více definovaných funkcí. Nelze ale vytvářet vlastní uživatelské funkce. Funkce ***na výstupu poskytují jednotlivé hodnoty nebo tabulky, na vstupu využívají parametry***. Funkce mohou být vnořovány („*nested*“), tedy výstup jedné funkce je vstupem pro další.

Na rozdíl od SQL jazyka neumožňuje funkce jako INSERT, UPDATE, DELETE. Data v tabulce v Power BI nelze měnit, pouze dotazovat nebo filtrovat.

1.1 Výpočetní předpis v DAX

Formulace kalkulací a dalších operací v DAX je obdobná některým základním pravidlům Excelu, na druhé straně ale má i výrazné odlišnosti**.**

Každý ***výpočetní předpis v DAX začíná rovnítkem*** „**=**“ nebo přiřazením „***:=***“, jednotlivé prvky předpisu výlučně používají pro identifikaci název tabulky a název sloupce v tabulce. Například výpočetní předpis pro vynásobení skutečného prodeje v kusech cenou za kus má následující tvar:

*= FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks]*

kde *FTQ_Prodej* je název tabulky a *Prodej_Skut_Ks* a *Cena_Ks* jsou názvy sloupců, které v kalkulacích musí být vždy uzavřeny v hranatých závorkách.

Pokud je název tabulky víceslovný, začíná-li číslicí nebo je shodný s některým rezervovaným slovem DAXu, (např. Date, SUM apod.), je třeba název uzavřít do apostrofů, například *´Data Prodeje´*.

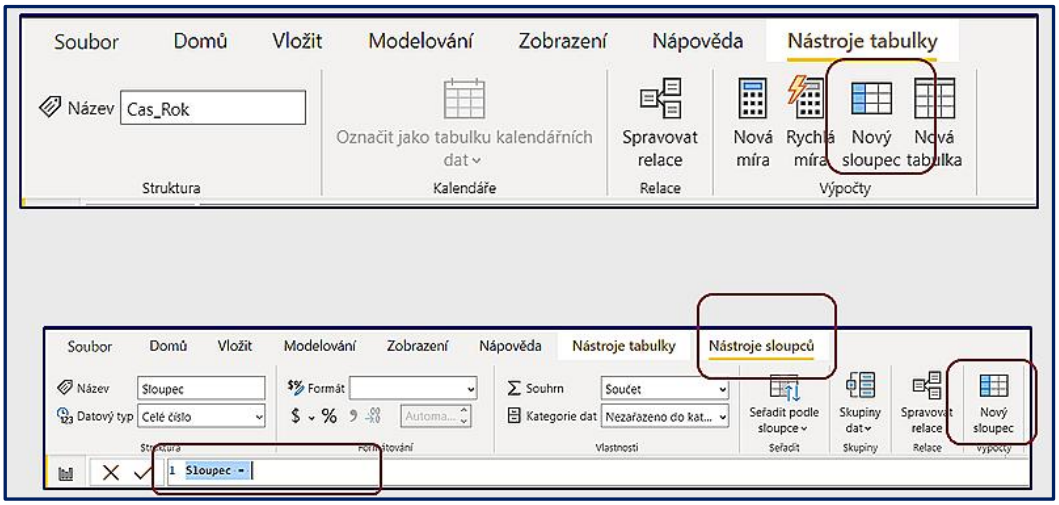
V rámci DAX se uplatňují podle daných pravidel (viz další kapitoly) tyto typy výpočtů, resp. kalkulací:

- ***kalkulovaný sloupec***, realizovaný pouze v rámci jedné (každé) řádky datové tabulky,
- ***kalkulovaná míra***, kde se výpočet uskutečňuje na úrovni celé datové tabulky,
- ***kalkulovaná tabulka,*** mohou být využity pouze v Power BI a SSAS Tabular, v určitých verzích DAX.

Podrobněji se k nim vrátíme v dalších kapitolách, nyní pouze neznačíme jejich využití.

1.2 Vytvoření kalkulovaného sloupce

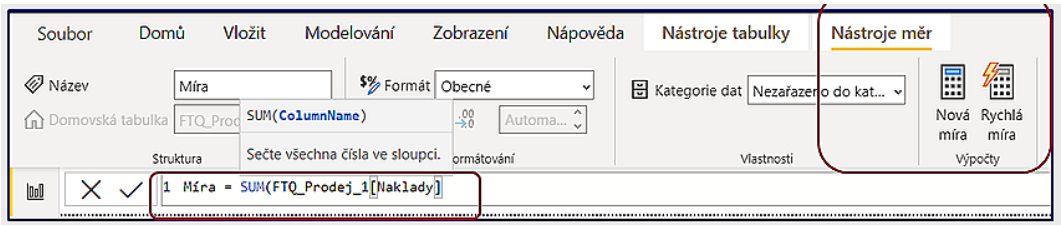
Kalkulované sloupce jsou v Power BI přidány do tabulek, v nichž jsou definovány. Volbou v menu nebo v seznamu polí tabulky ***„Nový sloupec“*** *(„New Column“)* ***se zpřístupní příkazová řádka*** a uživatel do ní zapíše příslušný příkaz DAX. Obrázek 1-1 ukazuje založení příkazu DAX pro přidání nového kalkulovaného sloupce.



Obrázek 1-1: DAX výraz pro jednoduchý výpočet kalkulovaného sloupce

1.3 Vytvoření kalkulované míry

Míra se počítá na agregační úrovni dat, tj. ***za tabulku*** nebo její podmnožinu (Obrázek 1-2):



Obrázek 1-2: Vytvoření nové kalkulované míry

Příkladem pro práci s mírou může být i celková marže z prodeje zboží, tj. celkový objem prodeje v Kč – celkové náklady na prodané zboží. K vytvoření míry se využije funkce *„Míry“.*

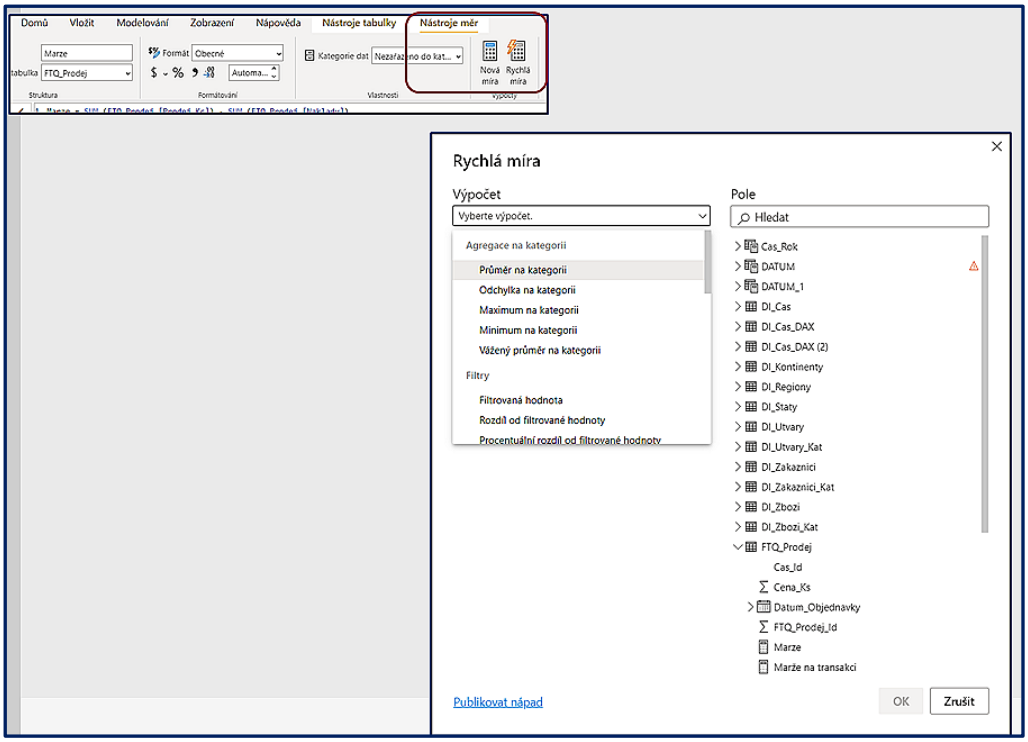
=SUM (FTQ_Prodej [Prodej_Kc]) - SUM (FTQ_Prodej [Naklady])

V každém případě je třeba, aby ***byla vybrána dimenzionální nebo faktová tabulka***, do které se příslušný sloupec nebo míra zařadí. ***V případě míry*** je zřejmé, že ***nemusí být v podstatě zařazena do žádné tabulky***, nebo by mohla vzniknout ***tabulka obsahující pouze vypočtené míry***, ale pro práci uživatelů je srozumitelnější, aby i míry byly dostupné ze seznamu polí tabulek, k nimž logicky nejvíce patří. Tak například vypočtená míra *„Marže“* logicky přísluší k prodejům zboží, je tedy zařazena mezi pole faktové tabulky *FTQ_Prodej* (*tzv. Home Table)*. Zařazení vypočteného ukazatele (míry) do domovské tabulky je viditelné a zároveň je možné ***změnit v pohledu Data*** v menu *Modeling*.

Jméno vytvořené kalkulované míry musí být v rámci celého sémantického modelu ***jedinečné***.

1.4 Podpora pro výpočet měr v Power BI – Rychlá míra

Power BI kromě přímého definování vypočtených měr zápisem příkazů v jazyce DAX (viz výše) nabízí k použití funkcionalitu ***pro rychlé vytváření nejobvykleji používaných měr*** v aplikacích BI. Tato funkcionalita je dostupná pod ***volbou Rychlá míra*** nebo na pravý klik v seznamu polí. Uživatel vybírá typ kalkulace, pole z tabulek sémantického modelu a další parametry podle typu kalkulace. Power BI ***na pozadí generuje příkaz v DAXu.***



Obrázek 1-3: Definování rychlé míry

Postup definice *rychlé míry* **je následující**. Uživatel vybere ***typ míry v poli „Výpočet“***. Potom jednoduchým tažením myší umístí datová pole, mezi nimiž jsou i všechny dosud vypočtené sloupce i ***míry z nabídky „Pole“*** a stejným způsobem určí filtr. Nakonec vybere z hodnot zvoleného pole pro filtr jednu nebo více hodnot.

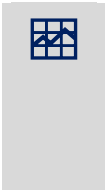
1.5 Vytvoření kalkulované tabulky

Kalkulované tabulky lze vytvořit příslušnou funkcí DAX, která na výstupu vrací tabulku nebo prostým zkopírováním jedné tabulky do nové, jak ukazuje další příklad:

```
Cas_1 = Cas
```

Tabulka *Cas_1* bude prostou a plnou kopií tabulky *Cas* s tím, že ***změny tabulky*** *Cas* se budou promítat do tabulky *Cas _1.* Na druhé straně změny v tabulce *Cas _1* původní tabulku *Cas* nezmění. To znamená, že do nové tabulky lze přidávat nové sloupce, míry, vazby, aniž by to ovlivnilo původní tabulku *Cas*.

Kalkulované tabulky mohou představovat souhrnné, resp. ***agregované hodnoty*** z původních tabulek, což pak urychlí a zpřehlední další výpočty (viz další kapitoly). Dalším jednoduchým příkladem vytvoření tabulky je vytvoření modifikované tabulky s využitím funkce *FILTER*, pouze s daty pro rok 2023.



```
Cas_23 = FILTER(
        Cas,
        'Cas' [Rok] = 2023
    )
```

2. Konstrukce jazyka DAX

Účelem kapitoly věnované konstrukci jazyka DAX je provést ***celkovou rekapitulaci*** a vymezení definovaných datových typů, operátorů a rovněž proměnných používaných pro zefektivnění vyjádření kalkulačních předpisů.

2.1 Datové typy

DAX používá **standardní datové typy**, a to:

- *Whole Number, integer.*
- *Decimal Number, Real.*
- *Fixed Decimal / Currency.*
- *Date / Time.*
- *TRUE/FALSE (Boolean).*
- *Text*.

V DAXu platí, že výsledný typ ve výrazu vychází z datových typů jednotlivých částí výrazu, např. pokud je ve výrazu použit datový typ *Date*, pak výsledek bude rovněž mít datový typ *Date*. Pokud je např. třeba zvýšit aktuální datum ve sloupci *Datum* o 3, pak pro výraz

= FTQ_Prodej [Datum] + 3

budou výsledky vypadat takto: původní: 15/5/2017, nové: 18/5/2017

20/7/2018 23/7/2018 atd.

DAX rovněž automaticky konvertuje řetězce na čísla a čísla na řetězce, ***jak to vyžaduje příslušný operátor*** (tzv. *operator overloading*), např. pro výraz ***7 & 2*** bude výsledná hodnota 72 (& je operátor konkatenace, spojování řetězců). Na druhé straně pro ***„6“ + „3“*** bude výsledek 9, řetězce jsou převedeny na čísla. Je zřejmé, že zde výsledek závisí na operátoru, nikoli na typu vstupních dat.

Pokud je třeba vygenerovat v DAX hodnotu datumu využije se k tomu funkce *DATE* s parametry rok, měsíc, den, např. DATE (2024, 11, 27).

2.2 Operátory

Operátory ve výrazech používá DAX standardní, takže dále je pouze jejich stručná rekapitulace:

- aritmetické: +, -, *, /, ^,
- porovnávací: =, <>, >, >=, <, <=,
- konkatenace: &,
- logické: AND nebo &&, OR nebo ||, IN.

Pořadí operátorů podle priorit je následující:

1. ^ (exponent),
2. – (znaménko, kladné nebo záporné),
3. * /,
4. ! (NOT),
5. + - ,
6. & (konkatenace),
7. = < > <= >= <>.

2.3 Sloupce

Data jsou ukládána ze zdrojů do sloupců tabulek s tím, že jejich ***obsah je považován za statický***, tedy nemůže být měněn, pouze při dalším vstupu nebo aktualizacích ze zdrojů.

2.4 Vazby

Vazby propojují tabulky, s tím, že se připouští ***pouze vazby 1:1 nebo 1:M***, ale pouze s využitím pouze jednoho sloupce v každé tabulce. Vazby umožňují, aby výběry řádek na základě funkce *FILTER* na jedné tabulce stejné proběhly v další tabulce s vazbou na původní tabulku. *FILTER* použitý v tabulce na straně M (many) se neuplatní v tabulce na straně 1. V opačném případě se filtr uplatní, a to i přes více propojených tabulek ale pouze ve směru 1 : M.

Obdobně se nabízejí možnosti provádět výpočty s daty ve sloupcích různých provázaných tabulek.

2.5 Hierarchie

Hierarchie jsou ***seskupení jednoho nebo více sloupců***, které vymezují jednotlivé úrovně hierarchie (pro účely *drill up a drill down*).

2.6 Poznámky

Poznámky mohou být zařazovány do výpočtů v DAXu na základě těchto znaků:

- // Text vpravo až do konce řádku je považován za poznámku.
- -- Text vpravo až do konce řádku je považován za poznámku.
- /* */ Text mezi dvojicemi znaků je považován za poznámku, i na více řádcích.

2.7 Proměnné (Variables)

Účelem proměnných v DAX je ***výpočty udělat jednodušší a některých případech i rychlejší*** a využívají se k ukládání výsledků z výrazů DAX. (Seamark, 2018). Syntaxe:

```
VAR název proměnné = výraz
RETURN výraz
```

Může být definována jedna nebo více proměnných, ale celý jejich blok musí být uzavřen klíčovým slovem *RETURN*. Název proměnné nemůže obsahovat mezery nebo být uzavřen v apostrofech nebo závorkách. *RETURN* vrací výsledek a může rovněž obsahovat výraz. Následující příklad dokumentuje naplnění proměnné:

```
VAR promenna1 = 7
VAR promenna2 = promenna1 + 3
RETURN promenna2 * 5
```

Proměnné mohou obsahovat nejen numerické hodnoty, ale i texty, např.:

```
VAR promText1= „Zjisti“
VAR promText2 = „hodnotu prodeje“
RETURN CONCATENATE (promText1, promText2)
```

Proměnné lze vnořovat, ale musí být zajištěno správné nastavení *RETURN*, jak ukazuje další příklad kalkulované míry:

```
Celkovy_prodej =
VAR Prodej1a = 200
VAR Prodej1b =
        VAR Prodej2a = Prodej1a
        VAR Prodej2b = Prodej2a + 150
        RETURN Prodej2b
RETURN Prodej1b
```

Proměnné lze využívat v různých situacích včetně kalkulovaných sloupců, kalkulovaných měr i kalkulovaných tabulek. K jejich využití se vrátíme v dalších kapitolách.

3. Kontext vyhodnocování výpočetních předpisů

Kontext vyhodnocování výpočetních předpisů a jeho pochopení představuje ***jádro pochopení*** konstrukce a využití celého jazyka DAX.

Účelem této kapitoly je objasnit principy a využití dvou tzv. kontextů, tj. ***filtr kontextu a řádkového kontextu*** a současně i jejich vliv na tvorbu ***kalkulovaných sloupců a kalkulovaných měr***.

DAX a na jeho základě definované výpočetní předpisy musí ve svém principu respektovat uspořádání a prostředí databáze Power BI, nebo SSAS Tabular a jejich uložení dat. To znamená, musí respektovat ***kontext***, v němž se definovaný předpis bude vyhodnocovat (*evaluation context*).

Pochopení kontextu vyhodnocování dat ***se promítá do dvou základních typů výpočetních předpisů*** i následně do využití nejrůznějších funkcí jazyka DAX. Kontext je ve svém základu dán systémem ukazatelů a dimenzí, resp. jejich úrovní a prvků, tedy multidimenzionálním uspořádáním dat v sémantickém modelu Power BI. Z pohledu zobrazení dat v kontingenčních tabulkách je pak kontext dán řádky a sloupci a jejich položkami, filtry a průřezy (*slicers*), na jejichž základě jsou reálná data zobrazena.

DAX rozlišuje ***dva kontexty***, a to:

- ***filtr kontext*** (***filter context***) vztahující se k souhrnným hodnotám a aktuálně nastaveným filtrům tabulky,
- ***řádkový kontext*** (***row context***) mající základ ve výpočtech a dalších operacích v rámci 1 řádky.

Většina pravidel kontextu se realizuje ***automaticky***, ale v některých případech se nabízejí možnosti. jak pomocí kontextu upravovat požadované výpočty. Kontext představuje filtrovací vrstvu dynamicky určující způsob výpočtů včetně řádkových i sloupcových součtů.

3.1 Filtr kontext

Filtr kontext je ***sada sloupcových filtrů*** využívaných u každých výpočtů určujících, které řádky tabulky vstoupí do dalších výpočtů. Platí tato pravidla (Seamark, 2018):

- každý filtr může být založen ***na jednom nebo i více sloupcích*** současně,
- je účelné se dívat ***na každou buňku (míru)*** výstupní tabulky jako na ***samostatný výpočet***,
- je rovněž účelné se dívat na kontext jako na ***kontejner***, který je buď prázdný, nebo obsahuje jeden nebo více sloupcových filtrů,
- jakmile je příslušný výpočet proveden kontejner se ***vyprázdní***.

3.1.1 Implicitní filtr kontext

Předpokládejme, že máme kalkulované míry tržeb za prodeje skupin zboží v létech (např. v milionech Kč) v následující výstupní tabulce:

	Počítače	Televize	Pračky	Součet
2020	7,3	10,5	8,1	**25,9**
2021	6,1	8,4	6,7	**21,2**
2022	8,4	12,6	9,3	**30,3**
Součet	**21,8**	**31,5**	**24,1**	**77,4**

Jednotlivé hodnoty buněk odpovídají výpočtům kalkulované míry filtrované na základě prvků sloupců a řádek. To představuje efekt ***filtr kontextu kalkulované míry***, který ***DAX implicitně doplňuje*** ke každé ze 16 buněk tabulky. To znamená každý výpočet kalkulované míry v této tabulce využívá unikátní filtr kontext.

Všechny filtry filtr kontextu jsou založené ***na logickém součinu*** (AND). Každá buňka tabulky má tak svůj filtr kontext a každý filtr kontext má svoji sadu sloupcových filtrů.

3.1.2 Explicitní filtr kontext

Explicitní filtr kontext se vztahuje k vyjádření kalkulací, kde specifikuje přidání nebo odebrání sloupcového filtru k nebo z filtr kontextu. Umožňuje upravovat, resp. měnit předpokládané způsoby výpočtu. Výsledek explicitního filtr kontextu se liší pode toho, zda je uplatněn na kalkulovaný sloupec nebo kalkulovanou míru. To dokumentuje další příklad:

Kalkulovaný sloupec:

```
Televize_Kalk_Sloupec = CALCULATE (
                        Sum (´Fakt Prodej´ [Trzby]),
                        ´Dimenze Zbozi´ [Typ] = „Televize“
                        )
```

Kalkulovaná míra:

```
Televize_Kalk_Mira = CALCULATE (
                     Sum (´Fakt Prodej´ [Trzby]),
                     ´Dimenze Zbozi´ [Typ] = „Televize“
                     )
```

Je evidentní, že kalkulační předpis je zcela identický, ale výstupy se budou vzhledem k odlišnému kontextu lišit. Výpočty probíhají na faktové tabulce *´Fakt Prodej´* a definují filtr na základě sloupce v jiné tabulce *´Dimenze Zbozi´* (vazba je definována v sémantickém modelu). Rozdíl ve výsledném zobrazení ukazuje následující tabulka.

	Tržby	Televize Kalk Sloupec	Televize Kalk Mira
Počítače	21,8		31,5
Televize	31,5	31,5	31,5
Pračky	24,1		31,5
Součet	**77,4**	**31,5**	**31,5**

Zatímco v případě výpočtu kalkulovaného sloupce je odpovídající hodnota uvedena pouze v případě položky „*Televize*“, v případě kalkulované míry se opakuje pro všechny položky zboží.

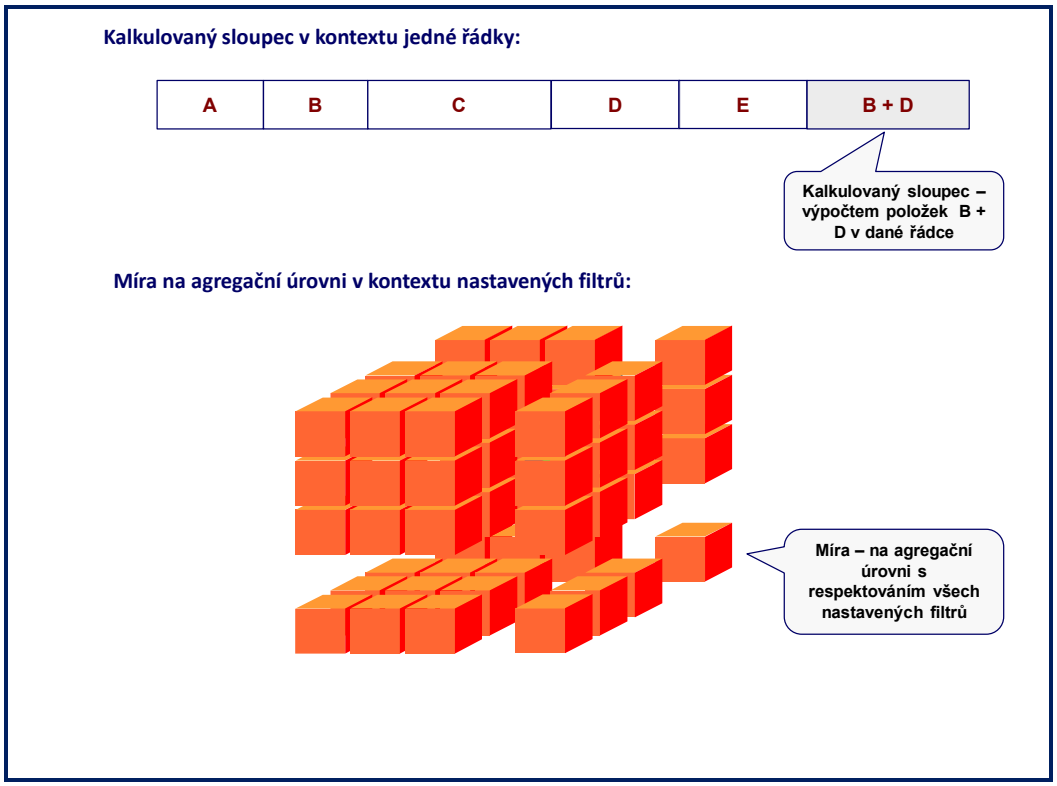
3.2 Řádkový kontext

Vedle filtr kontextu existuje i řádkový kontext a s tím související možnost v DAXu transformovat řádkový kontext na filtr kontext. Řádkový kontext představuje vyhodnocování dat ale vždy na úrovni jednotlivých řádků.

Řádkový kontext je využívaný ***při tvorbě kalkulovaných sloupců*** nebo v případě využití tzv. ***iterátorů*** (např. SUMX(), FILTERX(), viz dále), které provádějí výpočty po jednotlivých řádcích dané tabulky, přičemž jejich počet, resp. rozsah je dán původním filtr kontextem. Jako v případě filtr kontextu ***se vztahuje k výpočtům pro každou buňku*** výstupní tabulky. Pokud se např. vytváří kalkulovaný sloupec pro výstupní tabulku o 20 řádcích pak výpočet se bude opakovat 20x vždy s mírně odlišným řádkovým kontextem. Platí, že pokud jsou data do sémantického modelu uložena nebo obnovena, už nemohou být dále měněna.

Zásadní rozdíl mezi kalkulovaným sloupcem a mírou je v tom, že ***hodnoty buněk kalkulovaného sloupce*** se počítají v okamžiku fyzického vstupu nebo obnovy dat do sémantického modelu, zatímco ***hodnoty kalkulované míry*** se přepočítávají vždy při změně filtru, který ji ovlivňuje.

Kalkulované sloupce a míry, oba druhy kontextu a jejich detailní charakteristiky budou předmětem dalšího textu. Rozdíl mezi nimi dokumentuje Obrázek 3-1.



Obrázek 3-1: Rozdíl v kontextu kalkulovaného sloupce a míry

3.3 Skalární a tabulkové funkce

Většina funkcí DAX ***vrací jednu hodnotu*** na základě např. agregací nebo logiky vyhodnocení dat, jako např. COUNTROWS() pro počet řádek tabulky. Ty mohou obsahovat i informační funkce jako ISBLANK() a LOOKUPVALUE() a jsou označeny jako ***skalární funkce***.

Vedle skalárních funkcí může mnoho DAX funkcí ***vracet na výstupu i tabulku***. Tabulky, které vracejí funkce jako FILTER() nebo ALL(), se využívají jako ***parametry pro další míry*** ovlivňující jejich filtr kontext. K dispozici jsou i další tabulkové funkce jako např. TOPN(). Výsledky tabulkových funkcí v DAXu mohou být ***součástí i uživatelských reportů***, jako nejčastěji je funkce SUMMARIZECOLUMNS(). Kromě toho mohou tabulkové funkce vracet sumarizované nebo filtrované tabulky přímo do datasetu a označují se jako ***kalkulované tabulky***. S nárůstem složitosti modelů a řešení v DAXu se častěji používají i míry představující kombinace skalárních a tabulkových funkcí.

4. Kalkulované sloupce a kalkulované míry

Účelem kapitoly je vrátit se ve větším detailu ke kalkulovaným sloupcům a kalkulovaným mírám a funkcím, které jsou s nimi spojeny, a k příkladům jejich užití.

DAX umožňuje definovat kalkulované sloupce tedy vlastní výpočty v tabulkách Power BI. **Kalkulovaný sloupec** je vypočítáván **v řádkovém kontextu,** tj. v kontextu jednotlivých řádek tabulky a vytváří v tabulce nový kalkulovaný sloupec z hodnot položek a konstant dané řádky. **Kontext řádky** tak znamená, že definovaný výpočet se vždy provede právě na jedné řádce, a to pouze v jejím rozsahu, tedy v kontextu této řádky. Postupně zpracování přechází z první na další řádky v rámci celé tabulky.

Hodnoty kalkulovaného sloupce jsou vypočítávány v průběhu obnovení, resp. aktualizace tabulky a výpočet **není závislý** na aktivitách uživatele při práci s kontingenční tabulkou, zejména na nastavení filtrů.

4.1 Vytvoření kalkulovaného sloupce a řádkový kontext

K vytváření kalkulovaných sloupců doplníme ***několik poznámek***:

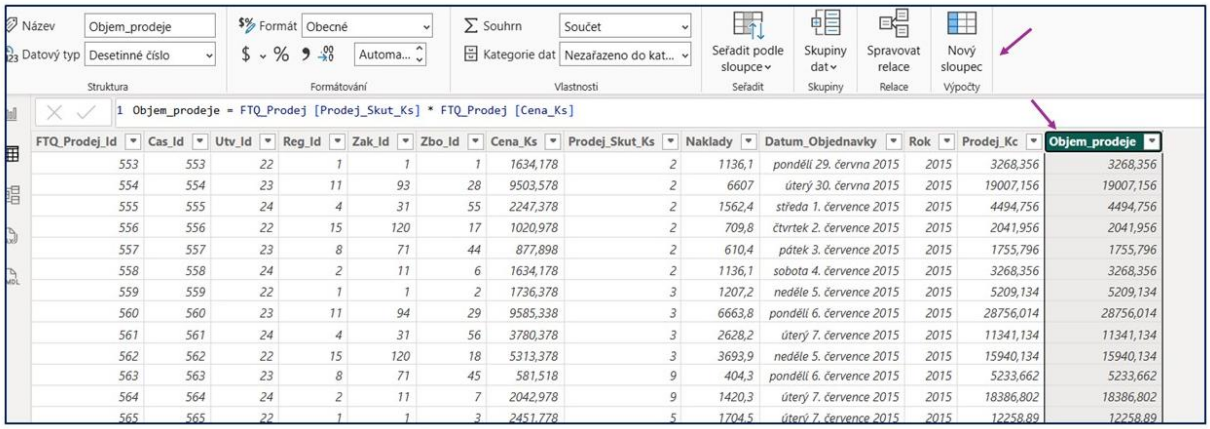
- na kalkulované sloupce se lze rovněž ***odkazovat jejich názvem***, proto je účelné využívat názvy, které co nejlépe vyjadřují obsah sloupce, resp. kalkulace,
- všechny položky (v řádcích) kalkulovaného sloupce ***využívají stejný výpočet*** (nikoli jako v listech Excelu) – pokud je nutné zařadit nějaké výjimky, pak pouze s využitím funkce *IF (),*
- výsledky výpočtů v kalkulovaném sloupci ***se ukládají do databáze Power BI***, tj. do aktualizovaného souboru *nazev_souboru.pbix* – to má pozitivní dopady na výkonnost aplikací Power Pivot,
- ***výpočty se provádějí definicí, nebo redefinicí*** kalkulačního předpisu, případně při spuštění funkce aktualizace*,* to znamená, že data v kalkulovaném sloupci jsou (oproti *Míře*, statická nemění se s použitím filtrů),
- v rámci kalkulovaných sloupců lze ***využít celou škálu funkcí***, např. =*SUM ([Naklady])* – agregovaná hodnota bude stejná v každém poli, resp. řádku sloupce, nebo =*AVERAGE ([Naklady]), =COUNT ([Naklady]), =MONTH ([Datum_Objednavky]), =YEAR ([Datum_Objednavky])* apod.

4.2 Příklady kalkulovaných sloupců

Příkladem kalkulovaného sloupce je výpočet zisku / ztráty z jednotlivých prodejů zboží:

Zisk = FTQ_Prodej [Prodej_Kc] - FTQ_Prodej [Naklady]

Dalším příkladem je kalkulovaný sloupec pro „*Objem_prodeje*“, jak ukazuje Obrázek 4-1:



FTQ_Prodej_Id	Cas_Id	Utv_Id	Reg_Id	Zak_Id	Zbo_Id	Cena_Ks	Prodej_Skut_Ks	Naklady	Datum_Objednavky	Rok	Prodej_Kc	Objem_prodeje
553	553	22	1	1	1	1634,178	2	1136,1	pondělí 29. června 2015	2015	3268,356	3268,356
554	554	23	11	93	28	9503,578	2	6607	úterý 30. června 2015	2015	19007,156	19007,156
555	555	24	4	31	55	2247,378	2	1562,4	středa 1. července 2015	2015	4494,756	4494,756
556	556	22	15	120	17	1020,978	2	709,8	čtvrtek 2. července 2015	2015	2041,956	2041,956
557	557	23	8	71	44	877,898	2	610,4	pátek 3. července 2015	2015	1755,796	1755,796
558	558	24	2	11	6	1634,178	2	1136,1	sobota 4. července 2015	2015	3268,356	3268,356
559	559	22	1	1	2	1736,378	3	1207,2	neděle 5. července 2015	2015	5209,134	5209,134
560	560	23	11	94	29	9585,338	3	6663,8	pondělí 6. července 2015	2015	28756,014	28756,014
561	561	24	4	31	56	3780,378	3	2628,2	úterý 7. července 2015	2015	11341,134	11341,134
562	562	22	15	120	18	5313,378	3	3693,9	neděle 5. července 2015	2015	15940,134	15940,134
563	563	23	8	71	45	581,518	9	404,3	pondělí 6. července 2015	2015	5233,662	5233,662
564	564	24	2	11	7	2042,978	9	1420,3	úterý 7. července 2015	2015	18386,802	18386,802
565	565	22	1	1	3	2451,778	5	1704,5	úterý 7. července 2015	2015	12258,89	12258,89

Obrázek 4-1: Příklad kalkulovaného sloupce

Dalším příkladem pro ***přidání kalkulovaného sloupce „Den název“ do tabulky DATUM*** je demonstrace ***funkce SWITCH*** pro přidání sloupce s českým názvem dne podle hodnoty pořadového čísla dne v týdnu ve sloupci „*Den v týdnu*“:

```
Den_nazev = SWITCH (DATUM [Den_v_tydnu],
                    1,"Pondělí",
                    2,"Úterý",
                    3,"Středa",
                    4,"Čtvrtek",
                    5,"Pátek",
                    6,"Sobota",
                    7,"Neděle")
```

4.3 Vytváření měr a filtr kontext

Míra, resp. *calculated measure,* nebo *počítané pole, calculated field,* se počítá na agregační úrovni dat, tj. ***za tabulku*** nebo její podmnožinu na základě ***kontextu daného filtrem*** nebo filtry (*filter context*). Data jsou tak vypočítávána na úrovni agregací (nikoli 1 řádky) a s respektováním filtrů, které ovlivňují jednotlivé buňky. *Míry* se tak nemohou vztahovat k jednotlivým řádkům. Míra se může vztahovat i k 1 nebo více kalkulovaným sloupcům. Je součástí datové sady v Power BI – je k dispozici pro všechny reporty nad datovou sadou.

Postup vyhodnocení míry je následující (Deckler, Powell, 2022):

1. ***Počáteční filtr kontext.***
   a. Zahrnuje všechny filtry použité v rámci plochy reportu, tj. filtry na úrovni reportu, stránky nebo vizuálu.
   b. Výběr řádků a sloupců tabulek reprezentuje původní nastavené filtry.
2. ***Modifikovaný filtr kontext na bázi DAXu.***
   a. Pro základní míry zůstává počáteční filtr kontext nezměněný.
   b. Pro komplexní míry a funkce CALCULATE() bude počáteční filtr kontext modifikovaný. Prostřednictvím funkce CALCULATE() bude počáteční filtr kontext zrušen, nahrazen nebo doplněn novými podmínkami filtrů.
   c. Pokud dojde ke konfliktu mezi počátečním filtr kontextem a podmínkami doplněnými do míry DAXu, pak míra v DAXu přepíše počáteční podmínky report filtru.
3. ***Vazby a cross-filtering.***

   a. S každou tabulkou s upravenými filtry v předchozích dvou krocích, se filtr kontext dále promítá do nastavených vazeb tabulek (*cross-filtering*).
   b. Ve většině případů se filtr na dimenzionální tabulce promítá do provázané faktové tabulky, na bázi jednosměrného cross-filtering (vazba 1 : M).
   c. Na druhé straně obousměrný cross-filtering dovoluje realizovat filtr kontext i na vazbě M : 1.

4. ***Logika výpočtu míry.***
   a. Výpočetní logika míry (např. pro DISTINCTCOUNT(), COUNTROWS() apod.) se vyhodnocuje pouze na vybraných řádcích a sloupcích podle filtrů.
   b. Pro základní míry se realizuje na zbylých nebo aktivních řádcích faktové tabulky, zatímco pro ostatní míry ve spojení s dimenzionálními tabulkami je výpočet ovlivněn uvedenými nastavenými filtry.

Uvedený ***postup 4 kroků se opakuje*** pro výpočet každé hodnoty míry v reportu ***nezávisle***. To samozřejmě podstatně ovlivňuje výkon zpracování reportu. To vede obvykle k tomu, že se návrháři snaží např. podstatně omezit obousměrné filtrování.

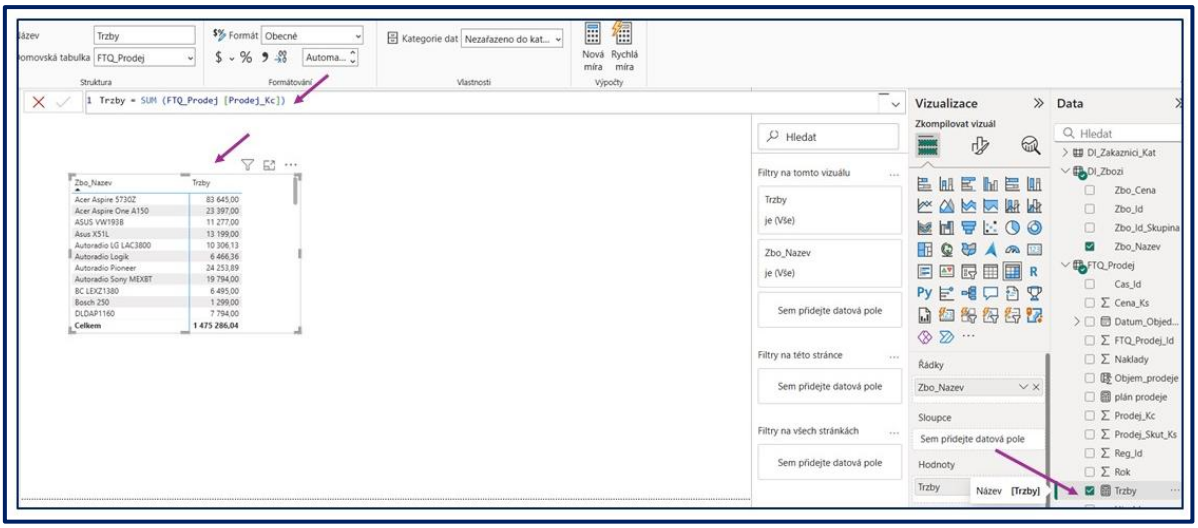
K ***vytváření a využití měr*** jsou uvedeny ***další poznámky***:

- všechny *míry* ***se nabízejí v řídících oknech***,
- ***úpravy míry se promítají*** do dalších měr, které mají na vstupu danou upravovanou míru (tyto změny mohou probíhat v několika úrovních tak, jak se využívají již vytvořené míry pro další nově vytvářené),
- míry se také označují jako ***přenosné výpočty****,* resp. *portable formulas*, protože se mohou využívat v mnoha různých situacích,
- pokud existují ***dva průřezy k různým dimenzionálním tabulkám*** v modelu, mohou filtrovat jak stejnou faktovou tabulku a také výpočet míry založené na faktové tabulce. Tyto filtry ale končí na faktové tabulce a neznamenají přímý vztah mezi dimenzionálními tabulkami.

Příklady míry:

- ***míra pro tržby*** (*funkce SUM*), viz Obrázek 4-2:

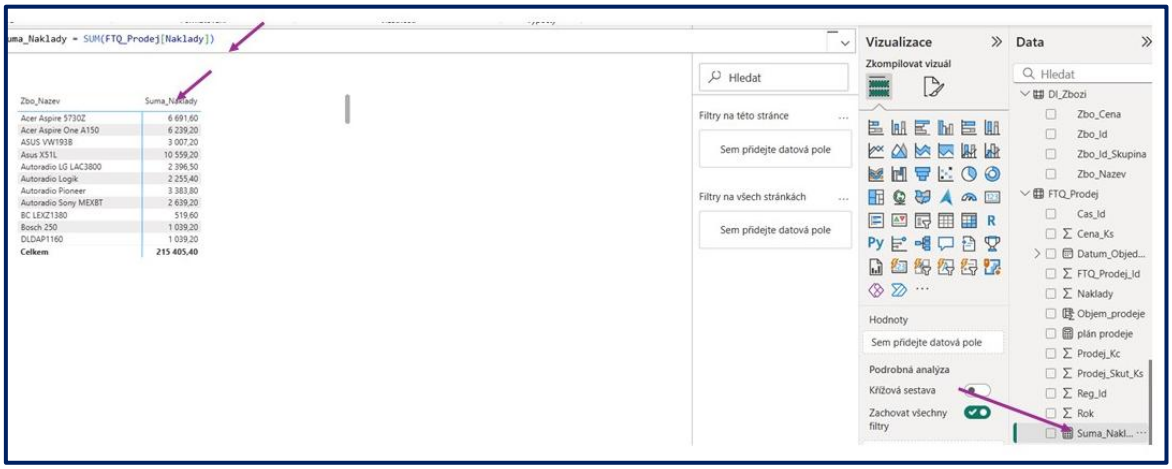
Trzby = SUM (FTQ_Prodej [Prodej_Kc])



Obrázek 4-2: Výpočet míry "Trzby"

- míra pro výpočet ***souhrnu nákladů***, viz Obrázek 4-3:

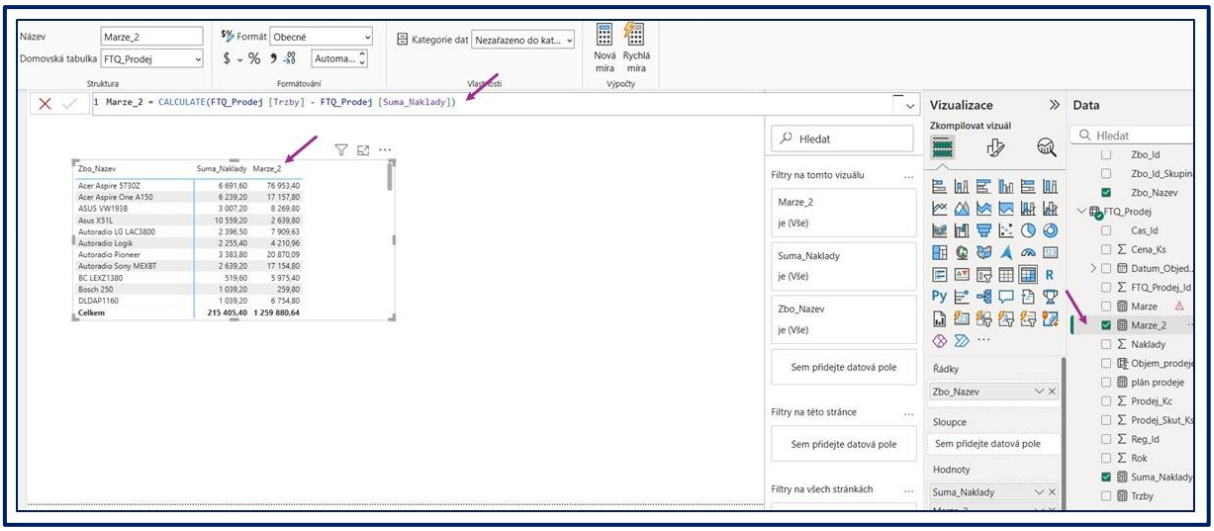
```
Suma_Naklady = SUM (FTQ_Prodej [Naklady])
```



Obrázek 4-3: Míra pro výpočet sumy nákladů

- míry jsou použity ***při výpočtu marže,*** viz Obrázek 4-4:

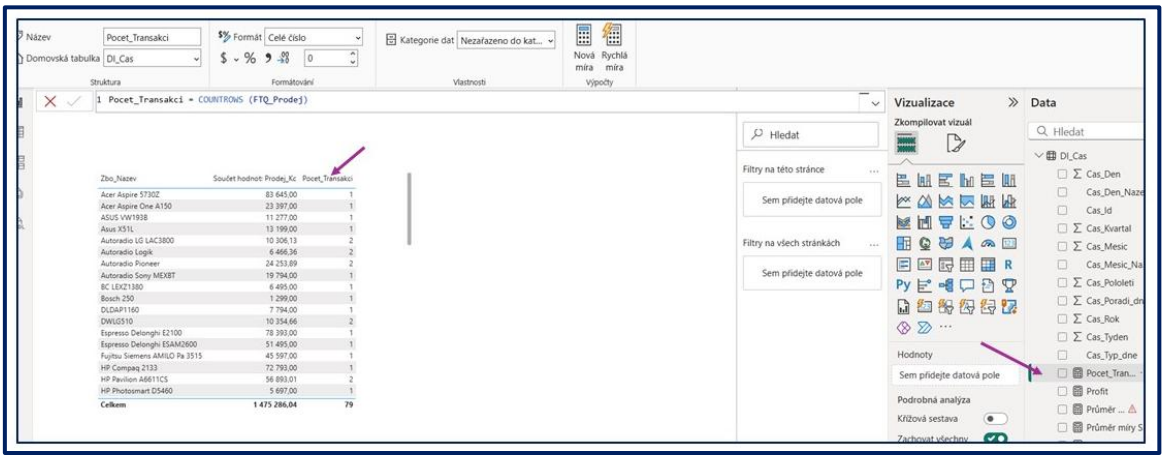
```
Marze_2 = CALCULATE (FTQ_Prodej [Trzby] - FTQ_Prodej [Suma_Naklady])
```



Obrázek 4-4: Výpočet maže

- pro výpočet ***počtu prodejních transakcí***, jednotlivých řádků lze využít následující míru, která spočítá počet řádek faktové tabulky, tedy počet prodejních transakcí, viz Obrázek 4-5:

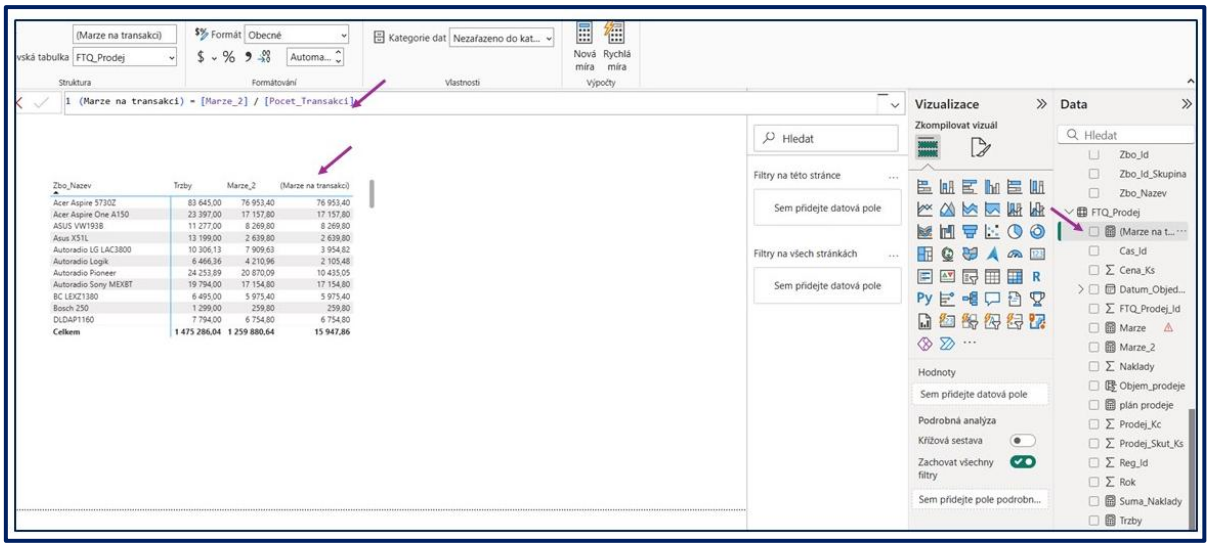
Pocet_Transakci = COUNTROWS (FTQ_Prodej),



Obrázek 4-5: Výpočet počtu prodejních transakcí

- vytvořené míry lze ***využít při definování nových měr a postupně je vkládat do sebe***. To pak umožňuje, že provedení určitě změny do jedné míry se automaticky promítne do všech dalších, kde je tato míra využita. Příkladem je ***výpočet marže na jednu obchodní transakci***, viz Obrázek 4-6

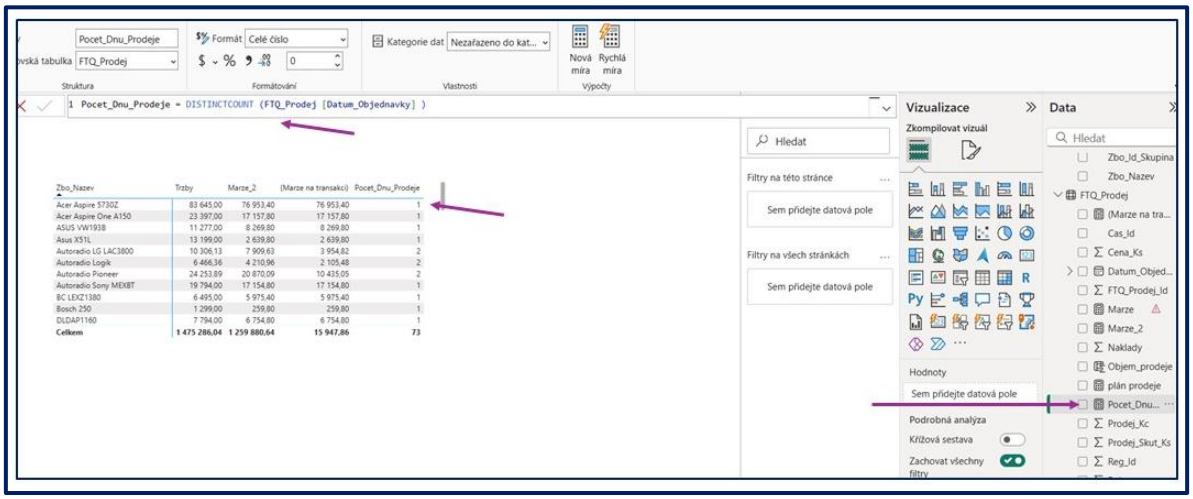
Marze_na_transakci = [Marze_2] / [Pocet_Transakci]



Obrázek 4-6: Výpočet marže na transakci

- ***pro výpočet dnů, kdy byly zpracovány objednávky*** na zboží (mohou se v přehledu prodejů opakovat) lze použít následující míru, která vypočítá počet výskytů unikátních, neopakujících se datumů objednávky:

[Pocet_Dnu_Prodeje]:= DISTINCTCOUNT (FTQ_Prodej [Datum_Objednavky])



4.4 Funkce CALCULATE()

Funkce *CALCULATE()* se považuje ***za nejdůležitější, nejužitečnější a nejkomplexnější funkci*** jazyka DAX. Podstatnou charakteristikou *CALCULATE()* je to, že umožňuje modifikovat *filtr kontext*, tedy zrušení a nastavení filtrů podle okamžité potřeby. Jinými slovy, *CALCULATE()* nastavuje nový filtr kontext a na jeho základě vyhodnocuje specifikovaný výraz. ***Syntaxe*** *CALCULATE()* je následující:

CALCULATE (výraz míry, <podmínka 1>, <podmínka 2>, …)

Pro *CALCULATE()* je ***jediný povinný parametr***, a to ten první – *výraz míry,* další – podmínky, resp. filtry jsou nepovinné a jejich počet není nijak omezen. *CALCULATE()* ***funguje následujícím způsobem***:

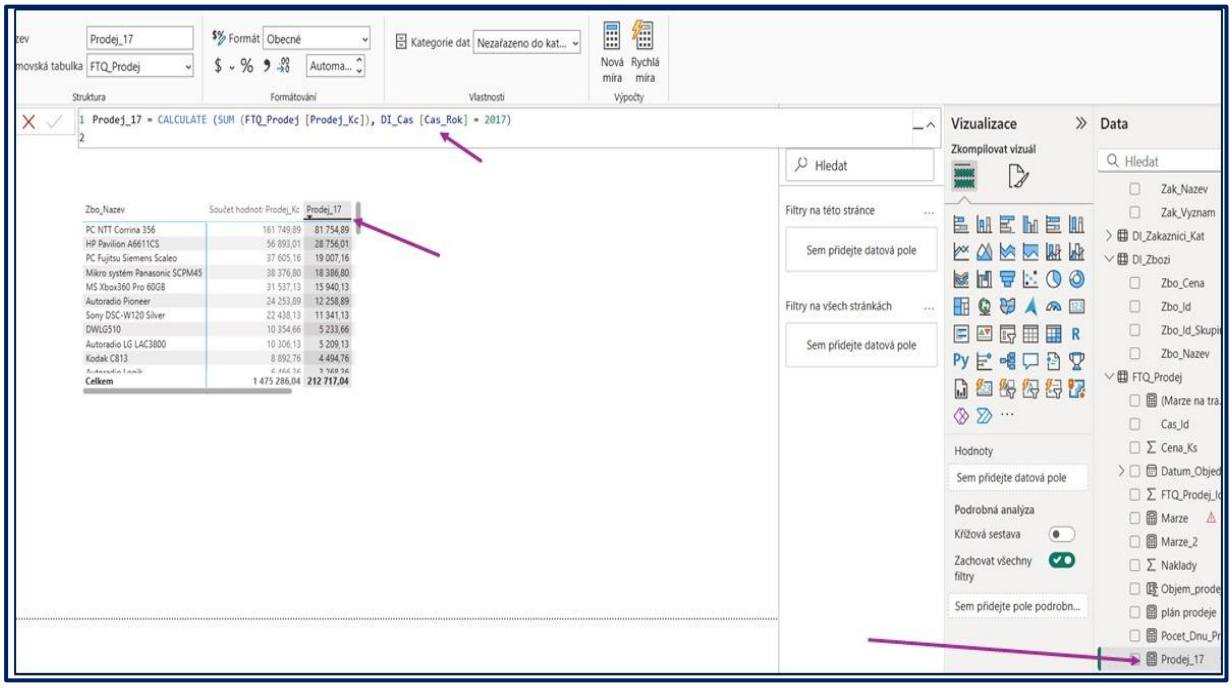
- ***převezme aktuální filtr kontext*** a vytvoří z něj kopii do nového filtr kontextu,
- ***vyhodnocuje každou podmínku*** v zadaných parametrech a pro každou z nich platí, že:
  - pokud je použit sloupec, který ještě ***nebyl*** předmětem filtru v původním kontextu, přidá tuto podmínku do nově vytvářeného filtr kontextu,
  - pokud na druhé straně je použit sloupec, který už ***byl*** předmětem filtru v původním kontextu, pak nahradí existující filtr novou zadanou podmínkou,
- všechny podmínky jsou v novém filtr kontextu vyhodnocovány ***s uplatněním logického součinu***, tedy operátoru ***AND***,
- jakmile je nový filtr kontext vytvořen, ***výraz míry je na jeho základě vyhodnocen***.

CALCULATE() rozeznává ***2 typy filtrů***:

- ***booleovské podmínky***, jako např. *DI_Zbozi [Cena] > 2000*, tyto filtry se používají na jednotlivých sloupcích – s výslednou hodnotou *Ano / Ne*, resp. *TRUE / FALSE*,
- ***seznamy hodnot prezentované ve formě tabulky***, které mají být součástí nového filtr kontextu a všechny sloupce této tabulky jsou součástí filtru.

Příklad použití funkce *CALCULATE()* pro ***výpočet hodnoty prodeje v roce 2017,*** viz Obrázek 4-7:

```
=CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Cas [Cas_Rok] = 2017)
```



Obrázek 4-7: Objem prodeje v roce 2017

Další příklady:

- Prodej zboží pouze v roce 2025 a současně za zboží s identifikátorem 35:

```
= CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Cas[Cas_Rok] = 2025, DI_Zbozi [Zbo_Id] = 35)
```

- Pokud je potřeba, je ve výrazu použít operátor pro vyjádření logického součtu, tj. ||, nebo *OR*. Operátor || je možné použít pouze pro porovnání dvou stejných položek (sloupců), např. Zbo_Id.:

=CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Zbozi [Zbo_Id] = 60 || DI_Zbozi [Zbo_Id] = 63)

4.5 CALCULATE() v řádkovém kontextu

Další podstatnou úlohou funkce CALCULATE() je ***transformace*** (v případě potřeby) ***řádkového kontextu na filtr kontext***. Předpokládejme, že potřebujeme vytvořit souhrn ceníkových cen a uložit ho do kalkulovaného sloupce (*Souhrn_Cen*) na základě zápisu:

Souhrn_Cen = SUM (DI_Zbozi [Cena])

V tomto případě se tedy ***v řádkovém kontextu vypočítá celkový souhrn ceníkových cen*** a ten se uloží do každého řádku kalkulovaného sloupce *Souhrn_Cen*. Pokud ale v rámci řádkového kontextu požadujeme souhrny ceníkových cen podle jednotlivých druhů zboží, pak použijeme v rámci řádkového kontextu funkci *CALCULATE()*, takto:

Souhrn_Cen_Calc = CALCULATE (SUM (DI_Zbozi [Cena]))

V tomto případě neobsahuje zápis *CALCULATE()* ***žádné filtry*** a tedy nemění existující filtr kontext, ale na druhé straně ***transformuje existující řádkový kontext na filtr kontext***. Uskutečňuje tak ***transformaci kontextu***. Na základě tohoto zápisu výraz *SUM (DI_Zbozi [Cena])* je vypočítáván v rámci filtr kontextu vždy pouze pro jednu řádku.

Tento princip se efektivně ***uplatňuje ve výpočtech s více provázanými tabulkami***. Jak jsme již zmínili, filtry se v případě řádkového kontextu nepromítají automaticky do dalších provázaných tabulek, zatímco pro filtr kontext platí, že se promítají automaticky ve směru vazby 1 ku M. To znamená, že transformace kontextu takové automatické promítání filtrů do vázaných tabulek zajistí, tedy např.

Souhrn_Prodeje_Calc = CALCULATE (SUM (FTQ_Prodej [Prodej_Skut_Ks]))

To znamená, že ***filtry z tabulky zboží*** (*DI_Zbozi)* se automaticky ***promítnou do tabulky faktů*** (*FTQ_Prodej*) – jde o vazbu 1 : M - a dostaneme tak výsledky odpovídající těmto filtrům a podobně i v obdobných případech.

4.6 Parametr ALL, ALLSELECTED

Funkce ***CALCULATE*(),** jak bylo již uvedeno, umožňuje ***měnit, a tedy i zrušit všechny nastavené filtry*** na určité tabulce, a to s využitím parametru *ALL*, např. … *ALL* (*DI_Zbozi [Cena])…* – zruší všechny filtry na sloupci ceny zboží. Pokud ale požadujeme tyto filtry ***zrušit kromě těch, které jsme aktuálně nastavili*** ve výstupní tabulce, pak se pro to využije parametr *ALLSELECTED*, tedy

… ALLSELECTED (DI_Zbozi [Cena])…

Funkce *CALCULATE()* má ***širokou škálu možností*** a zejména může ***operativně měnit aktuální filtr kontext a nastavovat nový*** a tím měnit prostředí pro realizaci výpočtů a dalších operací. Lze doporučit si na dílčích příkladech ověřovat tyto možnosti a postupně tak rozšiřovat i možnosti využití jazyka DAX pro složitější analytické úlohy a aplikace Self Service Business Intelligence.

4.7 Funkce CALCULATETABLE()

Podobně jako funkce CALCULATE(), která vrací skalární hodnotu výrazu, funkce CALCULATETABLE() modifikuje filtr kontext výrazu, ale vrací tabulku. To dokumentuje další příklad, kde funkce vrací tabulku zboží pouze s řádky, kde cena je vyšší než 1 500,-:

= CALCULATETABLE() (DI_Zbozi, DI_Zbozi [Cena] >1500)

4.8 Funkce IF ()

S příkladem, kde počítáme ***objem marží na jednu transakci***, tedy podíl objemu marží počtem transakcí se logicky váže další často využívaná funkce *IF ()*. Je zřejmé, že ***pokud by*** v některých případech ***byl počet transakcí roven nule***, pak by míra vykazovala chybu. Ošetření tohoto stavu je pak (jako i v jiných produktech) záležitostí funkce *IF ()*. Její ***syntaxe*** je následující:

IF (podmíněný výraz, hodnota pro splnění podmínky, hodnota pro nesplnění podmínky)

Použití funkce IF () v případě výpočtu maže na jednu transakci dokumentuje další příklad:

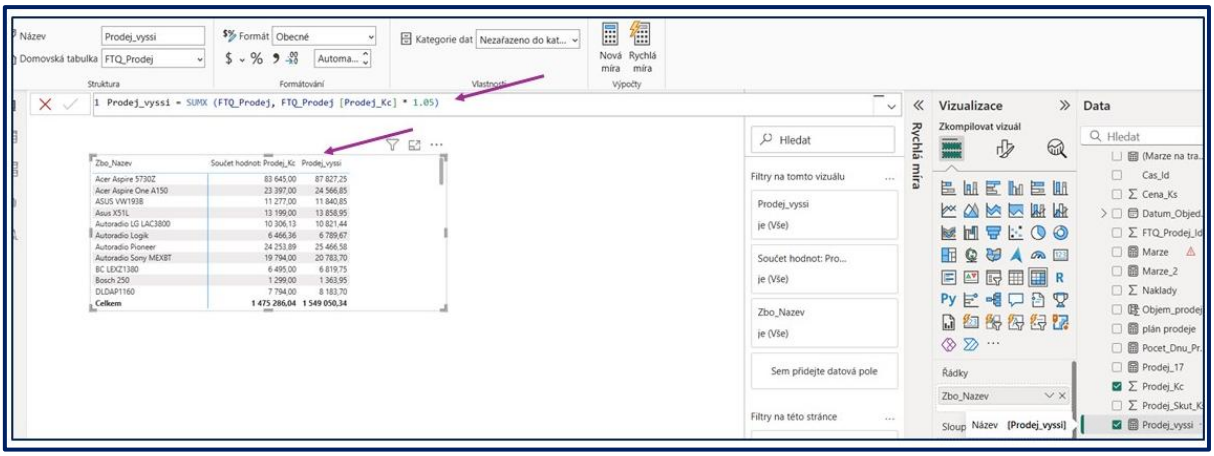
=IF ([Pocet_Transakci] = 0, BLANK(), [Marze] / [Pocet_Transakci])

Tedy, pokud je počet transakcí roven nule, vrátí výpočet míry mezeru, resp. funkci BLANK (), pokud ne, vrátí hodnotu příslušného podílu.

4.9 Iterátor

Další možností je využití řádkového kontextu s iteracemi, resp. s použitím *iterátorů*. Všechny funkce DAX končící na ***„X“*** jsou považovány za iterátory. To znamená, že vyhodnocují výpočty v každé řádce, a nakonec je agregují podle různých algoritmů. Tak např. funkce ***SUMX***, viz Obrázek 4-8:

*=SUMX (FTQ_Prodej, FTQ_Prodej [Prodej_Kc] * 1,05)*



Obrázek 4-8: Využití iterátoru pro výpočet navýšení objemu prodeje

zajistí, že v každé řádce tabulky se položka náklady zvýší o 5 % a nakonec vrátí souhrn těchto přepočítaných hodnot. Funkce *SUMX* tak využije v tabulce FTQ_Prodej řádkový kontext v rámci celé iterace, i když zápis odpovídá v tomto případě výpočtu *míry* (obdobně i dále).

Všechny ***iterátory*** v DAX fungují stejně, a to:

1) vytvoří nový řádkový kontext na tabulce definované prvním parametrem,
2) vyhodnotí druhý parametr v řádkovém kontextu, tj. pro každou řádku tabulky,
3) vytvoří agregaci z hodnot vytvořených v průběhu druhého kroku, pokud to specifikuje iterátor. Některé iterátory (*FILTER, ADDCOLUMS*) tento krok neprovádějí.

Další iterátory představuje následující přehled:

- AVERAGEX: aritmetický průměr hodnot z vyhodnocených výrazů v řádcích tabulky.
- COUNTX: počet hodnot, které jsou výsledkem vyhodnocených výrazů v každé řádce tabulky.
- GEOMEANX: geometrický průměr hodnot z vyhodnocených výrazů v řádcích tabulky.
- MAXX: maximální numerická hodnota z vyhodnocených výrazů v řádcích tabulky.
- MEDIANX: medián z vyhodnocených výrazů v řádcích tabulky.
- MINX: minimální numerická hodnota z vyhodnocených výrazů v řádcích tabulky.
- …

Dalším ***typem iterátorů*** je funkce ***FILTER***, která v řádkovém kontextu prochází tabulku (řádku po řádce) a ***vrací novou tabulku*** s řádky odpovídajícími zadané podmínce, např.:

```
=FILTER (DI_Zbozi, DI_Zbozi [Zbo_Cena] > 3000)
```

To znamená, že funkce vrátí tabulku zboží, jejichž cena je vyšší než 3000. Pokud je např. třeba ***zjistit počet prodaných zboží s prodejní cenou vyšší než 1500***, pak bude výraz vypadat takto:

```
Pocet_zbozi := COUNTROWS (FILTER (FTQ_Prodej, FTQ_Prodej [Cena_Ks] > 1500))
```

V případě, že by při použití došlo ***ke konfliktu filtrů***, takže by výše uvedená funkce nedávala žádné výsledky, používá se ***parametr ALL***, který zajistí, že se ***aktuální filtr kontext ignoruje***, např.:

```
Pocet_zbozi = COUNTROWS (FILTER ALL (FTQ_Prodej, FTQ_Prodej [Cena_Ks] > 1500))
```

4.10 CALCULATE () a využití FILTER ()

Z předchozího textu také vyplynulo, že nastavené filtry se promítají ***prostřednictvím vazeb*** z jedné tabulky do další, a to vždy ***ve směru od 1 ku M*** (ve vazbách *1 : M*). Totéž ***platí i pro filtry nastavené pomocí CALCULATE()***, tj. např. od *DI_Zbozi_Kat* (kategorie zboží), přes *DI_Zbozi_Skupina* – *DI_Zbozi* – až *FTQ_Prodej* (faktovou tabulku).

Podstatným pravidlem na druhé straně je to, že současně lze nastavovat jeden filtr (jednu podmínku) ***pouze na jednom sloupci***, např. potřebujeme vytvořit míru pro objem prodeje (*Prodej_Skut_Ks*), a to pouze za zboží, kde ceníková cena zboží (*Cena*) je vyšší, než trojnásobek standardních nákladů na jednotku zboží (*Zbo_Naklady*):

```
Profitabilni_Prodej =
        CALCULATE (
           SUM (FTQ_Prodej  [Prodej_Skut_Ks] ),
           DI_Zbozi [Cena]  >  DI_Zbozi [Zbo_Naklady] * 3 )
```

Uvedený zápis míry ***povede k chybě***, protože podmínka, tedy druhý parametr pracuje se 2 sloupci najednou. Cesta, jak řešit toto omezení, je ***použít variantu seznamu hodnot***, tj. funkci *FILTER*. V tomto případě pak bude zápis vypadat takto:

```
Profitabilni_Prodej =
        CALCULATE (
            SUM (FTQ_Prodej  [Prodej_Skut_Ks] ),
            FILTER ( DI_Zbozi, DI_Zbozi [Cena]  >  DI_Zbozi [Zbo_Naklady] * 3 ))
```

Pokud se ve funkci FILTER() používá v podmínce komplexní míra, pak je dobré s takovou funkcí nakládat opatrně. V řádkovém komplexu je míra řádku po řádce vyhodnocována a v případě velmi rozsáhlých tabulek to může znamenat výrazné snížení výkonu a prodloužení doby odezvy.

4.11 Cílové hodnoty KPI

KPI (*Key Performance Indicator*) vizuál v PBI porovnává hodnotu míry indikátoru ve vztahu ke specifikované cílové hodnotě, což může být také míra. Tato cílová míra může být kalkulována na základě současné míry. Příkladem může být cílová hodnota objemu prodeje zvýšené o 5 %, např.:

```
Cilovy_Prodej = (FTQ_Prodej  [Prodej_Skut_Ks] ) * 1,05
```

V některých případech může být vytvořena a zařazena do datasetu specifická tabulka se sloupci s cílovými hodnotami prodeje, nákladů apod., a to na požadované úrovni granularity časové dimenze (roky, kvartál, měsíc apod.). Míry pro cílové hodnoty mohou být zpřístupněny s pomocí funkce např. LOOKUPVALUE() , která vrací skalární hodnoty daného sloupce.

4.12 Další analytické funkce založené na DAX

Poslední část kapitoly je věnována několika komplexním funkcím založeným na možnostech jazyka DAX.

4.12.1 Seskupování dat, banding

V podnikových analýzách je často potřeba ***sdružit velká množství dat do definovaných skupin***. Příkladem mohou být analýzy prodeje podle intervalů cen (velmi nízká, nízká atd.). Hodnoty prodejů zboží jsou např. v tabulce *FTQ_Prodej*. Předpokladem pro řešení ***funkce bandingu*** je vytvoření speciální

tabulky, např. *Band_Tab* pro definování cenových intervalů, např. (použijeme zde standardního termínu *band*):

Band_Id	Band_Nazev	Min_Cena	Max_Cena
1	Velmi nízká	0	50
2	Nízká	51	150
3	Střední	151	500
4	Vysoká	501	1000
5	Velmi vysoká	1001	99999

Tabulka se stane ***součástí sémantického modelu***. Problém je ale v tom, že nelze definovat její vazbu k uvedené tabulce faktů *FTQ_Prodej*, neboť v ní žádný odpovídající klíč k *Band_Id* není. Řešení je ve ***vytvoření kalkulovaného sloupce*** *Band_Prodej* v rámci tabulky faktů na základě následujícího předpisu:

```
Band_Prodej = CALCULATE (
        VALUES (Band_Tab [Band_Id] ),
         FILTER (
                    Band_Tab, Band_Tab [Min_Cena] <= FTQ_Prodej [Cena]
                   && Band_Tab [Max_Cena] >  FTQ_Prodej [Cena] )
                  )
```

4.12.2 Vytvoření pořadí, *ranking*

Vytvoření ***pořadí různých objektů*** podle stanovených hodnot je rovněž častou součástí reportů a analýz. Předpokládejme, že požadujeme ***zjistit pořadí zboží podle objemů prodeje***. Pro tento účel slouží funkce *RANKX*, která má ***charakter iterátoru*** a využívá dva parametry – název tabulky a výraz. Funguje tak, že zpracovává výraz pro každou řádku tabulky a třídí výsledky. Pro daný příklad má předpis pro míru následující tvar:

```
Poradi_Nakladu = RANKX (ALLSELECTED (Dl_Zbozi),  [Suma_Prodeje] )
```

Předpokládáme zde předem vytvořenou míru *Suma_Prodeje*. Parametr *ALLSELECTED* zajišťuje, zrušení všech filtrů, kromě nastavených na kontingenční tabulce, což je předpoklad pro správné číslování pořadí.

4.12.3 Růst počtu zákazníků

Účelem bude sledovat nárůst počtu zákazníků v jednotlivých létech. Na počátku je třeba definovat novou základní hodnotu zvoleného ukazatele, tj. počet aktivních zákazníků:

```
Aktivni_Zakaznici =
        DISTINCTCOUNT (FTQ_Prodej [Zak_Id] )
```

V dalším kroku se zvolí ***základna, resp. základní hodnota*** (*base measure*):

```
2024_Zakaznici =
        CALCULATE ( [Aktivni_Zakaznici], FTQ_Prodej [Rok] = 2024 )
```

Ve třetím kroku se bude počítat ***procentuální nárůst zákazníků*** vzhledem k roku 2024

```
Aktivni_Zakaznici_Rust :=
        DIVIDE ( [Aktivni_Zakaznici] - [2024_Zakaznici], [2024_Zakaznici] )
```

5. Řešení vazeb

Výpočty v DAX se provádějí i ***na vzájemně provázaných tabulkách***. ***Účelem*** kapitoly je vytvořit představu o tom, jak lze výpočty při různých typech vazeb realizovat.

Úlohy Business Intelligence i Self Service Business Intelligence zahrnují 2 typy tabulek – faktové (*data tables*) a dimenzionální (*lookup tables*). Zatímco ***faktové tabulky*** představují průběh, obsah a výsledky byznys procesů, ***dimenzionální tabulky*** vyjadřují podstatné charakteristiky všech faktorů, resp. dimenzí, které tyto procesy ovlivňují a podle kterých je účelné je analyzovat.

Většina úloh se může realizovat automaticky ***na základě standardních vazeb*** tabulek definovaných v sémantickém modelu. ***Nikoli*** ale ***ve všech případech***. K tomu se používá ***řada funkcí***, zejména:

- GENERATE,
- CROSSJOIN,
- NATURALINNERJOIN,
- NATURALLEFTOUTERJOIN,
- UNION,
- LOOKUPVALUE.

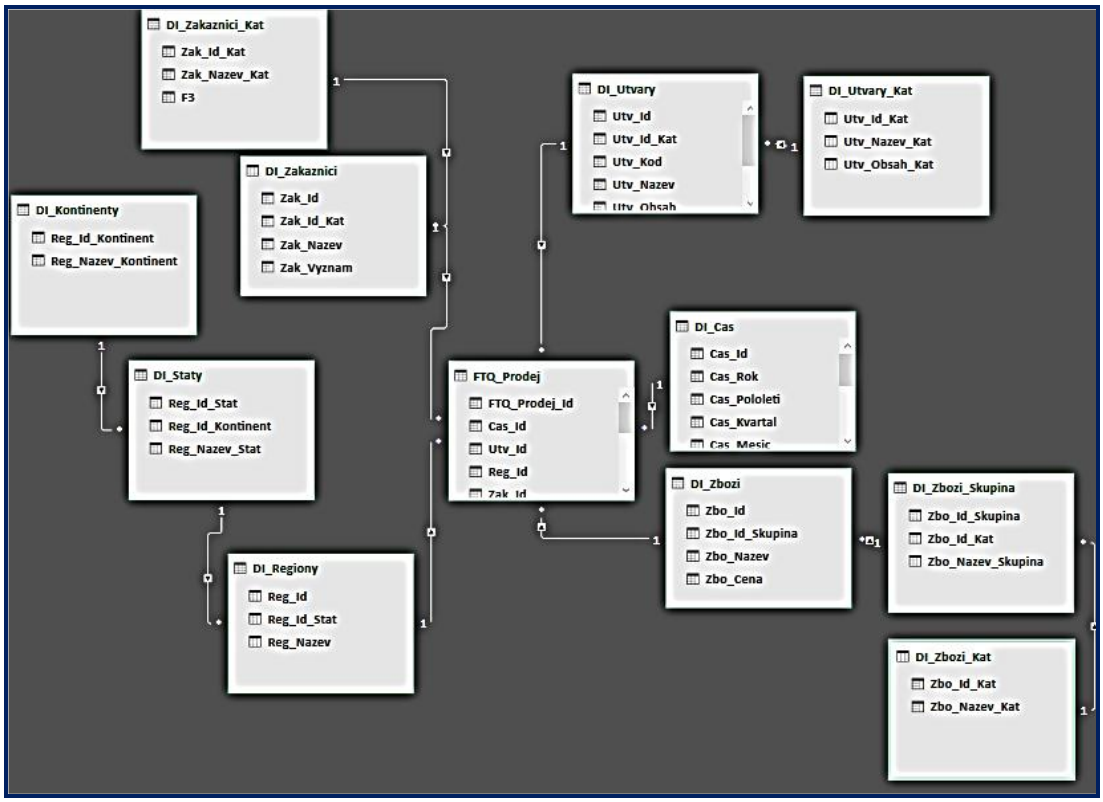
5.1 Řešení ve standardních vazbách

Dimenzionální tabulky se váží na faktové tabulky ***ve dvou schématech (STAR a SNOWFLAKE).*** Pro tyto tabulky a jejich vazby platí ***následující pravidla***:

- vazby mohou být jen ***1 : 1***, nebo ***1 : M***,
- u každé tabulky může být pro vazbu využit ***pouze 1 sloupec***,
- pro vyjádření vazby může být použit ***pouze operátor „=“,***
- ***nemůže být*** použita vazba na stejnou tabulku („***selfjoins***“),
- ***je třeba nedefinovat vazby mezi faktovými tabulkami navzájem***, ale pouze prostřednictvím sdílených dimenzionálních tabulek,
- není dobré vytvářet ***„násilně“ kombinované faktové tabulky*** z několika základních (*flatten tables*), které pak snižují přehlednost řešení a omezují analytické možnosti,
- na druhé straně je efektivním řešením v případě potřeby ***kombinovat využití měr vázaných na různé faktové tabulky*** v jedné výstupní tabulce.

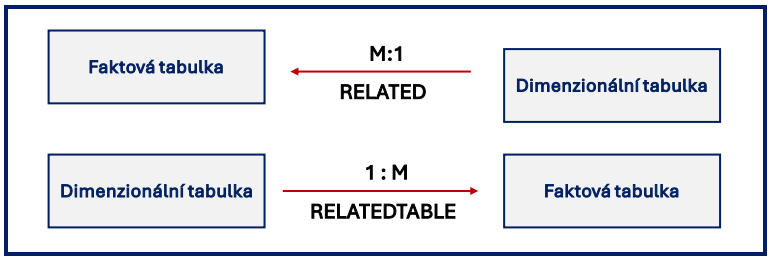
5.1.1 Řádkový kontext s vazbami tabulek

Pokud se ***zpracování v řádkovém kontextu*** bude vztahovat na 2 nebo více vzájemně provázaných tabulek pak je třeba využít parametrů ***RELATED*** nebo ***RELATEDTABLE***. Vazby v modelu pro další příkladu ukazuje Obrázek 5-1:



Obrázek 5-1: Příklad vazeb pro řádkový kontext s vazbami tabulek

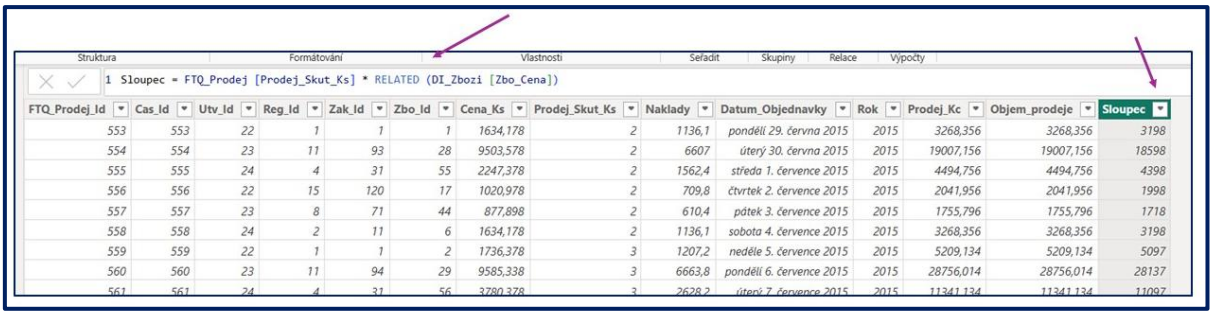
Předpokládejme, že chceme vytvořit nový kalkulovaný sloupec s hodnotou prodeje vypočítané ze skutečného množství prodaného zboží (*Prodej_Skut_Ks*) v tabulce faktů *FTQ_Prodej* a ceníkové ceny (*Cena*) v dimenzionální tabulce *DI_Zbozi*. Zobrazení příkladu vazeb ukazuje Obrázek 5-1. V tomto případě ***je podstatná kardinalita a směr vazby,*** tedy pro *FTQ_Prodej : DI_Zbozi* je kardinalita vazby *M : 1*. Tuto situaci dokumentuje Obrázek 5-2 nahoře.



Obrázek 5-2: Řešení vazeb v řádkovém kontextu s funkcemi RELATED a RELATEDTABLE

Pak lze požadovaný ***kalkulovaný sloupec*** v řádkovém kontextu zapsat následujícím výrazem, viz Obrázek 5-3:

```
= FTQ_Prodej [Prodej_Skut_Ks] * RELATED (DI_Zbozi [Cena])
```



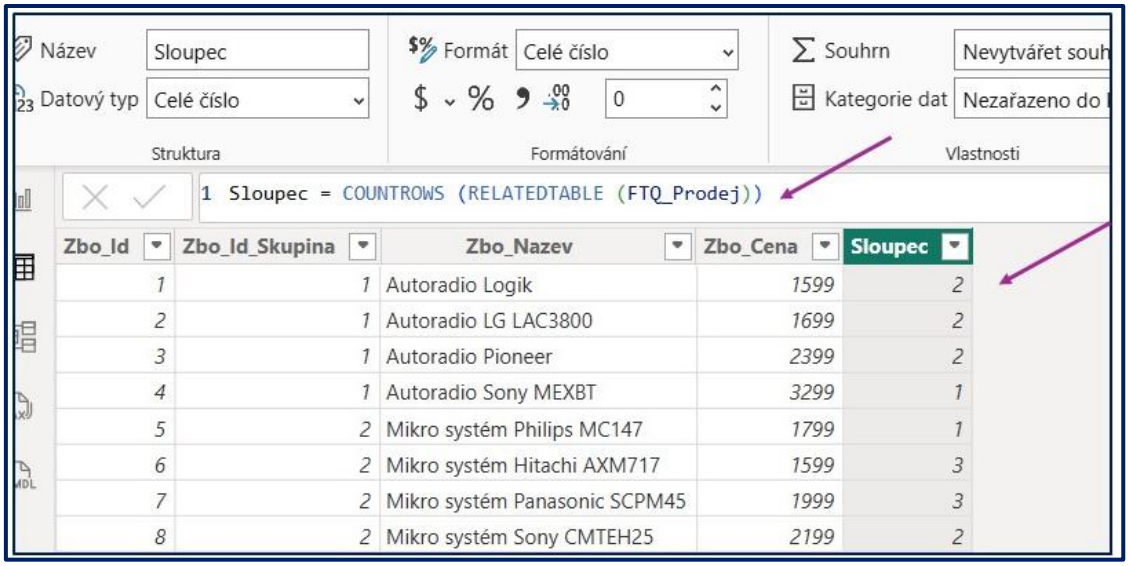
FTQ_Prodej_Id	Cas_Id	Utv_Id	Reg_Id	Zak_Id	Zbo_Id	Cena_Ks	Prodej_Skut_Ks	Naklady	Datum_Objednavky	Rok	Prodej_Kc	Objem_prodeje	Sloupec
553	553	22	1	1	1	1634,178	2	1136,1	pondělí 29. června 2015	2015	3268,356	3268,356	3198
554	554	23	11	93	28	9503,578	2	6607	úterý 30. června 2015	2015	19007,156	19007,156	18598
555	555	24	4	31	55	2247,378	2	1562,4	středa 1. července 2015	2015	4494,756	4494,756	4398
556	556	22	15	120	17	1020,978	2	709,8	čtvrtek 2. července 2015	2015	2041,956	2041,956	1998
557	557	23	8	71	44	877,898	2	610,4	pátek 3. července 2015	2015	1755,796	1755,796	1718
558	558	24	2	11	6	1634,178	2	1136,1	sobota 4. července 2015	2015	3268,356	3268,356	3198
559	559	22	1	1	2	1736,378	3	1207,2	neděle 5. července 2015	2015	5209,134	5209,134	5097
560	560	23	11	94	29	9585,338	3	6663,8	pondělí 6. července 2015	2015	28756,014	28756,014	28137
561	561	24	4	31	56	3780,378	3	2628,2	úterý 7. července 2015	2015	11341,134	11341,134	11097

Obrázek 5-3: Řešení vazeb v řádkovém kontextu s funkcí RELATED

Parametr ***RELATED*** v tomto případě ***funguje, protože řádkový kontext je na straně vazby many, „M“,*** tedy tabulky *FTQ_Prodej*. Pokud by řádkový kontext byl použit na straně vazby *jeden*, *„1“,* tedy *DI_Zbozi*, pak by výraz nefungoval, protože k jedné hodnotě sloupce *Cena*, by bylo více hodnot ve sloupci *Prodej_Skut_Ks* . To znamená, kalkulovaný sloupec by se nevytvořil.

Pokud ale řádkový kontext na straně vazby „1“ je třeba, pak se k tomu využije parametr ***RELATEDTABLE***. To je např. v situaci, kdy je potřeba ***spočítat počty prodejů za jednotlivé druhy zboží,*** tedy řádkový kontext se vztahuje k tabulce *DI_Zbozi*, tedy na straně „1“ v relaci k tabulce *FTQ_Prodej*. ***RELATEDTABLE vrací*** v řádkovém kontextu, tedy postupně ***pro každý řádek zboží, tabulku*** odpovídajících prodejů v tabulce *FTQ_Prodej*. Počty prodejů tak lze spočítat (jako součást tabulky *DI_Zbozi*) a vytvořit nový kalkulovaný sloupec, a to na základě tohoto výrazu:

=COUNTROWS (RELATEDTABLE (FTQ_Prodej))



Zbo_Id	Zbo_Id_Skupina	Zbo_Nazev	Zbo_Cena	Sloupec
1	1	Autoradio Logik	1599	2
2	1	Autoradio LG LAC3800	1699	2
3	1	Autoradio Pioneer	2399	2
4	1	Autoradio Sony MEXBT	3299	1
5	2	Mikro systém Philips MC147	1799	1
6	2	Mikro systém Hitachi AXM717	1599	3
7	2	Mikro systém Panasonic SCPM45	1999	3
8	2	Mikro systém Sony CMTEH25	2199	2

Obrázek 5-4: Řešení vazeb s využitím funkce RELATEDTABLE

Je dobré zdůraznit, že ***RELATED i RELATEDTABLE mohou tímto způsobem procházet celý řetěz*** provázaných tabulek, jejich počet není limitován. Jediné pravidlo je v tom, že ***vazby mezi tabulkami musí mít kardinalitu stejného typu***, tedy M : 1 nebo 1 : M a musí být definovány ***ve stejném směru***.

Posledním příkladem je ukázka ***použití neaktivní vazby mezi tabulkami ve výpočtu míry***, tedy využití ***funkce USERELATIONSHIP***. Power BI umožňuje definovat více vztahů mezi dvěma tabulkami. Aktivní však může být pouze jeden. ***Vztahy nemají názvy***, pro jejich označení je třeba použít názvy příslušných

atributů tabulek na obou stranách vztahu. Aby bylo možné pracovat s počtem dodaných kusů zboží podle data dodání ve vztahu k dimenzi času, je třeba pro výpočet dodaných počtů kusů použít neaktivní vztah. Funkce *USERELATIONSHIP* v DAXu aktivuje vztah mezi tabulkami *DI_CAS* a *FTQ_Prodej* provázáním přes atributy *DI_Cas [Cas_Datum]* a *FTQ_Prodej [Datum_dodani]*, a to právě vždy jen v době použití dané míry:

```
Počet_dodanych = CALCULATE (FTQ_Prodej [Prodano ks],
                    USERELATIONSHIP (FTQ_Prodej [Datum_Dodani], DI_Cas [Cas_Datum])
                                    )
```

5.1.2 Filtr kontext s vazbami tabulek

V předchozí části jsme se zabývali využitím řádkového kontextu a vytvářením kalkulovaných sloupců v prostředí více provázaných tabulek. Na tomto místě bude obsahem ***řešení měr a filtr kontextu*** rovněž v prostředí několika tabulek s definovanými vazbami (viz Obrázek 5-1).

Pro uplatnění filtr kontextu zde platí ***následující pravidlo***: Filtr aplikovaný na určitou tabulku se automaticky ***promítá do všech provázaných tabulek***, které jsou ***na straně „M“*** (*many*) ve vazbách *1 : M* (*one-to-many*). Na druhé straně, pokud je tabulka na straně *„1“* vazby *M : 1,* automatické promítnutí filtru ze strany *„M“* do *„1“* provázané tabulky neproběhne.

Jako příklady využijeme tyto míry a předpokládejme, že filtr bude nastavený na *DI_Zbozi* (např. Cena = 1000):

```
Pocet_Prodeju = COUNTROWS (FTQ_Prodej)
Pocet_Zbozi   = COUNTROWS (DI_Zbozi)
Pocet_Zakazniku = COUNTROWS (DI_Zakaznici)
```

Filtr na tabulce DI_Zbozi se promítne takto:

1) ***Pocet_Prodeju*** – filtr je definován na straně *„1“* vazby *1 : M*, tedy se z tabulky *DI_Zbozi* ***promítne*** do tabulky *FTQ_Prodej* a *Pocet_Prodeju* tím ***bude ovlivněn***,
2) ***Pocet_Zbozi*** – filtr je definován na vlastní tabulce a do výsledku *Pocet_Zbozi* se ***promítne*** a výsledek ***bude ovlivněn***,
3) ***Pocet_Zakazniku*** – filtr (na *DI_Zbozi,* tj. Cena = 1000, viz výše) se ***nepromítne***, protože tabulka *DI_Zakaznici* je na straně *„1“* vazby *M : 1* (*FTQ_Prodej : DI_Zakaznici*), a to i v případě, že by filtr byl nastaven přímo na *FTQ_Prodej*, výsledek nebude ovlivněn.

Určitým ***řešením posledního příkladu*** je ***funkce VALUES***. Ta vrací tabulku o pouze 1 sloupci obsahujícím všechny hodnoty odpovídající aktuálnímu filtru. Předchozí příklad by bylo možné takto ***modifikovat***:

```
Pocet_Zakazniku = COUNTROWS (VALUES (FTQ_Prodej [Zak_Id] )
```

Jak je patrné, funkce počítá ***počty zákazníků z tabulky faktů*** (nikoli *DI_Zakaznici*) s respektováním promítnutého filtru z tabulky *DI_Zbozi* do tabulky faktů *FTQ_Prodej*.

Je dobré zdůraznit, že (obdobně jako v řádkovém kontextu) ***promítnutí filtrů proběhne i v řetězu tabulek s vazbami stejného typu a ve stejném směru***.

5.2 Parametrické nepropojené tabulky

Jak název napovídá, DAX umožňuje vytvářet ***speciální tabulky, které obvykle slouží jako parametry*** pro přepočty hodnot ukazatelů podle aktuální situace. Tyto tabulky ***nejsou propojené vazbou*** k žádné jiné tabulce sémantického modelu, protože neexistuje ani klíčová položka, na jejímž základě by bylo možné vazbu vytvořit.

Příkladem může být ***tabulka kursů korun za Euro*** pro přepočty cen zboží. Tabulka na obrázku (Obrázek 5-5) představuje možné kursy Kč vzhledem k Euru, která bude dále ***sloužit jako parametrická pro přepočty.*** Je samozřejmé, že jde pouze o příklad a lze do ní doplnit množství dalších hodnot kursu.

[Kc_Euro] f_x

	Kc_Euro	Přidat sloupec
1	25,51	
2	25,59	
3	25,65	
4	25,72	
5	25,86	
6	25,95	
7	26,09	
8	26,15	
9	26,21	
10	26,25	
11	26,31	

Obrázek 5-5: Parametrická tabulka, kurs Kč za Euro

Tato tabulka bude pak při vytváření výstupních tabulek sloužit jako nabídka kursů v rámci sliceru. V dalším kroku vytvoříme *míru* maximální hodnoty kursu, viz Obrázek 5-6:

Korun_za_Euro = MAX (CZK_EUR [Kc_Euro])

Popisky řádků	Průměr Cena_Ks	Korun_za_Euro
Acer Aspire 5730Z	16729	26,31
Acer Aspire One A150	7799	26,31
ASUS VW193B	3759	26,31
Asus X51L	13199	26,31
Autoradio LG LAC3800	1699	26,31
Autoradio Logik	1599	26,31
Autoradio Pioneer	2399	26,31
Autoradio Sony MEXBT	3299	26,31
BC LEXZ1380	1299	26,31
Bosch 250	1299	26,31
Canon Pixma MP190	1799	26,31
DES1005D	389	26,31
DLDAP1160	1299	26,31
DWLG510	569	26,31
Epson Stylus SX100	1199	26,31

Obrázek 5-6: Vytvoření míry – maximální hodnota kursu

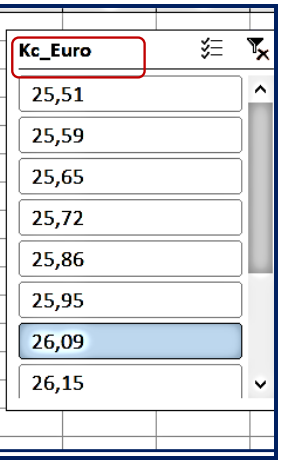
Tato ***míra*** bude poskytovat jednak ***maximální hodnotu na kursovním lístku***, nebo bude sloužit jako ***hodnota pro přepočet*** ceny zboží.

V dalším kroku vytvoříme míru pro přepočet průměrné ceny zboží podle zvoleného aktuální kursu Kč k Euru:

Cena_Euro = [Průměr Cena_Ks] / [Korun_za_Euro]

Míra Korun_za_Euro funguje v tomto případě tak, že pokud ve sliceru vybereme jednu hodnotu, např. 26,09, pak se do výpočtu doplní tato hodnota, pokud nevybereme žádnou hodnotu nahradí se maximální, tedy 26,31, viz Obrázek 5-7

Popisky řádků	Průměr Cena_Ks	Cena_Euro
Acer Aspire 5730Z	16729	5 770,83
Acer Aspire One A150	7799	2 690,34
ASUS VW193B	3759	1 296,70
Asus X51L	13199	4 553,12
Autoradio LG LAC3800	1699	586,09
Autoradio Logik	1599	551,59
Autoradio Pioneer	2399	827,56
Autoradio Sony MEXBT	3299	1 138,02
BC LEXZ1380	1299	398,31
Bosch 250	1299	398,31
Canon Pixma MP190	1799	551,63
DES1005D	389	119,28
DLDAP1160	1299	398,31
DWLG510	569	174,47



Obrázek 5-7: Přepočet průměrné ceny zboží podle aktuálního kursu Eura

Další podkapitoly se věnují funkcím, ***kdy nelze využívat standardní vazby***.

5.3 Funkce CROSSJOIN

Funkce CROSSJOIN má následující ***syntaxi***:

CROSSJOIN (<tabulka1>, <tabulka2> [,<tabulka n >])

Výsledkem je tabulka na bázi karteziánského součinu, kde ke každé řádce tabulky 1 jsou přiřazeny všechny řádky tabulky2 atd. k dalším tabulkám. To znamená, že pokud první tabulka má 15 řádků a druhá tabulka 10 řádků, pak výsledná tabulka bude mít 150 řádků.

5.4 Funkce GENERATE

Funkce GENERATE má následující ***syntaxi***:

GENERATE (<tabulka1>, <tabulka2>)

Výsledkem je rovněž tabulka na bázi karteziánského součinu, kde ke každé řádce tabulky 1 jsou přiřazeny všechny řádky tabulky2. Parametry musí být dvě rozdílné tabulky. Pokud se předpokládá využití také funkce FILTER, pak je nutné použít funkce GENERATE a nikoli CROSSJOIN

Příklad: vytvořit kalkulovanou míru pro zjištění počtu řádků výsledné tabulky z funkce GENERATE:

Pocet_radku = COUNTROWS (GENERATE (´TabulkaX´, ´TabulkaY´))

Pokud chceme např. ponechat ve výsledné tabulce pouze řádky, kde je shoda mezi tabulkami u určitého atributu, např. „*Cena*“ a ty spočítat, pak se využije funkce GENERATE spolu s funkcí FILTER:

```
Počet_radku = COUNTROWS (GENERATE (´TabulkaX´,
                    FILTER (´TabulkaY´,
                                ´TabulkaX´ [Cena] = ´TabulkaY´ [Cena])
                                        )
                )
```

6. Summarizing, Aggregating

Účelem této kapitoly je ukázat, jak lze ***generovat souhrnné tabulky*** na bázi existujících tabulek. Souhrnné (sumární) tabulky mohou výrazně ***zvýšit výkon*** při práci s Power BI, zejména u velkých vstupních tabulek. Pro účely filtrování na vstupních datech je účelné souhrnnou tabulku propojit s původní.

Pro vytváření souhrnných tabulek jsou k dispozici zejména ***tyto funkce***:

- *SUMMARIZE,*
- *SUMMARIZECOLUMNS,*
- *GROUPBY.*

S uvedenými funkcemi jsou spojeny ***tyto možnosti***:

- vytvářejí kalkulované tabulky nebo mohou být součástí vytváření kalkulované míry,
- mohou specifikovat sloupce pro funkci „*group by*“, pokud specifikovány nejsou vrací se tabulka o jednom řádku,
- pokud je použit jeden sloupec pro „*group by*“, pak výsledná tabulka obsahuje tolik řádků, kolik je unikátních hodnot v daném sloupci,
- sloupce pro řešení souhrnů mohu existovat ve více tabulkách, pokud jsou správně nadefinovány jejich vazby.

6.1 Funkce SUMMARIZE

Syntaxe funkce je tato:

SUMMARIZE (<tabulka>, <groupBy_jmenosloupce1> …, <jmeno1>, <vyraz1>…..)

Kde:

- <tabulka> je jméno jakékoli tabulky v modelu, která může obsahovat i jiné kalkulované tabulky. Může to být i funkce, která vrací tabulku. V rámci funkce *SUMMARIZE* lze vytvořit pouze jednu souhrnnou tabulku,
- <groupBy_jmenosloupce1> je jméno jakéhokoli sloupce v tabulce <tabulka> nebo sloupce z tabulek propojených s touto tabulkou. Lze takto definovat i 2., 3. další parametry. Pokud jsou tyto parametry, resp. sloupce vynechány, výsledkem bude souhrnná tabulka o 1 řádku,
- <jmeno1> je jméno agregovaného sloupce,
- <vyraz1> je výraz pro agregaci.

Další příklad vytváří souhrnnou tabulku tržeb podle jednotlivých let a druhů zboží. S tím souvisí:

- k tabulce lze vytvářet vazby k ostatním tabulkám,
- do tabulky lze doplňovat kalkulované sloupce i míry,
- z tabulky lze vytvářet vizuály, využívat filtry,
- tabulku lze použít jako vstup do další funkce *SUMMARIZE* a vytvořit tak vyšší úroveň agregace

Příklad:

```
Souhrnna_tabulka_trzeb =
SUMMARIZE (
   -- faktová tabulka –
   'FTQ_Prodej'
   -- sloupce pro Group by –
    DI_Cas [Cas_Rok],
    DI_Zbozi [Zbo_Nazev])
    -- agregovaný sloupec –
    „Suna tržeb“, SUM ('FTQ_Prodej' [Trzby])
    )
```

Výsledná tabulka:

Rok	Zboží	Suma tržeb
	Počítače	150 280,-
2021	Počítače	300 500,-
2022	Počítače	450 270,-
2023	Počítače	350 900,-

Jak je patrné z tabulky je v prvním řádku zleva mezera. To znamená, že ve faktové tabulce 'FTQ_Prodej' v položce *Cas_Trzby* není žádná hodnota nebo hodnoty neodpovídají žádné z hodnot sloupce *Cas_Rok* v dimenzionální tabulce *DI_Cas*. Pokud je potřeba zjistit takové záznamy, lze k tomu využít funkci *FILTER*:

```
Tabulka_chybejicich_dat =
   FILTER (
    'FTQ_Prodej',
     ISBLANK (
       RELATED (DI_Zbozi [Zbo_Nazev])
      )
   )
```

6.1.1 SUMMARIZE s alternativní vazbou

Pokud je potřeba vytvořit sumarizovanou tabulku nikoli podle roku realizace tržeb, ale podle roku dodávek, tedy s jinou než aktivní vazbou mezi faktovou a dimenzionální tabulkou času, je možné využít již výše zmíněnou funkci USERELATIONSHIP, viz příklad:

```
Souhrnna_tabulka_trzeb =
```

```
CALCULATETABLE
 SUMMARIZE (
      'FTQ_Prodej'
      DI_Cas [Cas_Rok],
      DI_Zbozi [Zbo_Nazev])
    „Suma tržeb“, SUM ('FTQ_Prodej' [Trzby])
   ),
USERELATIONSHIP(
        'FTQ_Prodej' [Cas_Rok]_Dodani],
         DI_Cas [Cas_Rok],
)
```

6.1.2 SUMMARIZE s filtrem

V případě, že je třeba zjistit souhrnné hodnoty tržeb za jednotlivé roky, ale pouze za region Ostrava, změní se příklad následujícím způsobem:

```
Souhrnna_tabulka_trzeb_za_region_Ostrava =
SUMMARIZE (
     FILTER ( 'FTQ_Prodej', RELATED DI_Region [Reg_Nazev]) = „Ostrava“),
      DI_Cas [Cas_Rok],
      DI_Zbozi [Zbo_Nazev])
    „Suma_trzeb“, SUM ('FTQ_Prodej' [Trzby])
   )
```

6.2 Funkce SUMMARIZECOLUMNS

Syntaxe funkce je tato:

SUMMARIZECOLUMNS (<groupBy_jmenosloupce1>…, <filtrovanatabulka>…, <jmeno1>, <vyraz1>…..)

Kde:

- <groupBy_jmenosloupce1> je jméno sloupce pro realizaci agregací „group by“, pokud jsou použity i další tyto parametry, jde o další úrovně agregací,
- <filtrovanatabulka> je jméno tabulky vzniklé na základě definovaného filtru. Pokud tento parametr není uveden, funguje funkce adekvátně jako *SUMMARIZE,*
- <jmeno1> je jméno agregovaného sloupce,
- <vyraz1> je výraz pro agregaci.

Příklad 1:

```
Souhrnna_tabulka_trzeb =
 SUMMARIZECOLUMNS (
       DI_Cas [Cas_Rok],
       DI_Zbozi [Zbo_Nazev])
     „Suma tržeb“, SUM (‘FTQ_Prodej‘ [Trzby])
    )
```

Příklad 2:

```
Souhrnna_tabulka_trzeb =
 SUMMARIZECOLUMNS(
       DI_Cas [Cas_Rok],
       DI_Zbozi [Zbo_Nazev])
 FILTER(
     ALL(
          DI_Region [Reg_Nazev] ),
          DI_Region [Reg_Nazev]) = „Ostrava“
          )
     „Suma_trzeb“, SUM (‘FTQ_Prodej‘ [Trzby])
    )
```

6.3 Funkce GROUP BY

Syntaxe funkce je tato:

```
GROUPBY (<tabulka>, <groupBy_jmenosloupce1> …, <jmeno1>, <vyraz1>….. )
```

Funkce je obdobná jako *SUMMARIZE.* Rozdíl je v tom, že funguje pouze s využitím iterátorů, např. SUMX, AVERAGEX apod.

Příklad:

```
Souhrnna_tabulka_trzeb_na_zaklade_GROUPBY =
 GROUPBY (
       'FTQ_Prodej',
       DI_Cas [Cas_Rok],
       DI_Zbozi [Zbo_Nazev]),
     „Suma_trzeb“, SUMX (CURRENTGROUP(), ‘FTQ_Prodej‘ [Trzby])
    ).
```

6.3.1 Funkce CURRENTGROUP

Funkce iterátorů vyžadují jako parametr tabulku. V případě funkce GROUPBY je tato tabulka reprezentovaná funkcí CURRENTGROUP(). Ta poskytuje pro iterátor tabulku odpovídající faktové tabulce *FTQ_Prodej.* To dokumentuje i následující příklad s výpočtem tržeb podle prodaných kusů a ceny.

Příklad:

```
Souhrnna_tabulka_trzeb_na_zaklade_GROUPBY =
 GROUPBY (
      ‘FTQ_Prodej ‘,
      DI_Cas [Cas_Rok],
      DI_Zbozi [Zbo_Nazev]),
     „Suma tržeb“, SUMX(
            CURRENTGROUP(),
           ‘FTQ_Prodej‘ [Kusy] *   ‘FTQ_Prodej‘ [Cena] )
            ).
```

7. Time Intelligence

Účelem této kapitoly je prezentovat úlohy s využitím dimenze času a s pomocí jazyka DAX. Jde zejména o sledování časového vývoje vybraných ukazatelů, hodnocení nárůstu ukazatelů k určitému datu, meziroční porovnání a porovnání hodnot ukazatelů za různá časová období apod.

Analytické úlohy na bázi Business Intelligence a obdobně SSBI pracují v naprosté většině s dimenzí času. To umožňuje důležité analytické funkce jako např. ***sledování vývoje firemních ukazatelů, meziroční porovnání jejich hodnot, klouzavé ukazatele*** a další a velmi široké spektrum nejrůznějších funkcí spojených s časem. Souhrnně se celá tato analytická oblast označuje jako ***Time Intelligence***. Účelem této podkapitoly je objasnit základní princip takové funkcionality a ukázat alespoň některé častěji používané případy.

7.1 Základní principy time intelligence

Zcela obecný základ spojený s uplatněním časové dimenze byl obsažen již v předchozím textu. Vstupním předpokladem je navržení a ***naplnění časové dimenze*** a je dobré k ní doplnit několik poznámek:

- ***Struktura tabulky časové dimenze*** by měla obsahovat kromě údaje o čase i případné další potřebné údaje pro analýzy a podnikový reporting (např. plné názvy měsíců a dnů, určení pracovních dnů atd.).
- Pro ***identifikaci jednotlivých období***, většinou dní je k dispozici primární klíč nebo v definované struktuře, např. 20170101). Tento klíč také slouží k propojení s faktovými, případně některými dalšími tabulkami.
- Vedle primárního klíče se pro většinu funkcí DAX spojených s časem používá i ***plné datum, označované jako FullDate***, které musí být datového typu. Je možné použít i jiný název, ale s ohledem na četnost použití tohoto označení i v literatuře zůstáváme v tomto případě u anglického názvu, v daném kontextu ji označujeme jako *DI_Cas_DAX*.
- ***Zdrojem pro časovou dimenzi*** je buď databázová tabulka ***podnikového kalendáře*** spravovaná často v centrálních databázích, nebo manuálně vytvořená, resp. vygenerovaná tabulka časové dimenze. Výhodou centrálně spravovaného podnikového kalendáře je obvykle jeho systematická aktualizace a použití stejných algoritmů pro některé speciální části dimenze. Otázkou je však někdy jeho dostupnost a kvalita.
- Je nutné ověřit, aby ***rozsah časové dimenze*** pokrýval časový rozsah dat faktových tabulek včetně i potřebné rezervy do budoucnosti.

V některých situacích se u analytických úloh ***nevyhneme vytvoření více tabulek pro časovou dimenzi***. To je v případech, kdy je třeba současně rozlišovat různé časové okamžiky. Například je třeba rozlišit datum objednání zboží, datum požadovaného termínu dodání zboží a datum skutečného dodání zboží. Pro většinu časových funkcí DAX je nezbytné specifikovat, že právě daná tabulka má charakter časové dimenze, a právě daná položka má význam plného data.

7.2 Vygenerování dimenzionální tabulky času

Příklad komplexního výrazu v DAXu patří také do oblasti Time intelligence, pro vygenerování nové dimenzionální tabulky času s názvem *DATUM* se všemi potřebnými sloupci a pokrývající dny kalendáře pro všechny datumy objednávek zboží ve faktové tabulce *FTQ_Prodej*.

```
DATUM = ADDCOLUMNS
    ( CALENDAR ( DATE ( YEAR ( MIN ( 'FTQ_Prodej' [Datum objednávky] ) ), 1, 1 ),
                 DATE ( YEAR ( MAX ( 'FTQ_Prodej'[Datum_objednavky] ) ), 12, 31 )),
      "DateAsInteger", FORMAT ( [Date],  "YYYYMMDD" ),
```

```
        "Rok", YEAR ( [Date] ), "Měsíc číslo", FORMAT ( [Date], "MM" ),
        "Rok/měsíc", FORMAT ( [Date], "yyyy/mm" ),
        "Měsíc v Roce",  FORMAT ( [Date], "yyyy/mmm" ),
        "Měsíc Zkratka", FORMAT ( [Date], "mmm" ),
        "Měsíc", FORMAT ( [Date], "mmmm" ),
        "Den v týdnu", WEEKDAY ( [Date],2),
        "Den", FORMAT ( [Date], "dddd" ),
        "Čtvrtletí", "Q" & FORMAT ( [Date], "Q" ),
        "Čtvrtletí v roce", FORMAT ( [Date], "YYYY" ) & "/Q" & FORMAT ( [Date], "Q"
)    )
```

Funkce ***ADDCOLUMNS přidá sloupce s uvedenými názvy*** a vygeneruje tolik řádků, kolik je dnů mezi 1. lednem nejmenšího roku a 31. prosincem nejvyššího roku určeného datem objednávky v tabulce *FTQ_Prodej*, jak je definováno funkcí *CALENDAR*. ***Minimální datum*** definuje výraz:

```
DATE ( YEAR ( MIN ( 'FTQ_Prodej'[Datum objednávky] ) ), 1, 1 )
```

maximální datum pak výraz:

```
DATE ( YEAR ( MAX ( 'FTQ_Prodej'[Datum objednávky] ) ), 12, 31 )
```

Den (pořadí) v týdnu definuje výraz (argument 2 říká, že má týden začínat pondělkem, pokud argument chybí nebo je = 1, týden začíná nedělí):

```
"Den_v_tydnu", WEEKDAY ( [Date],2)
```

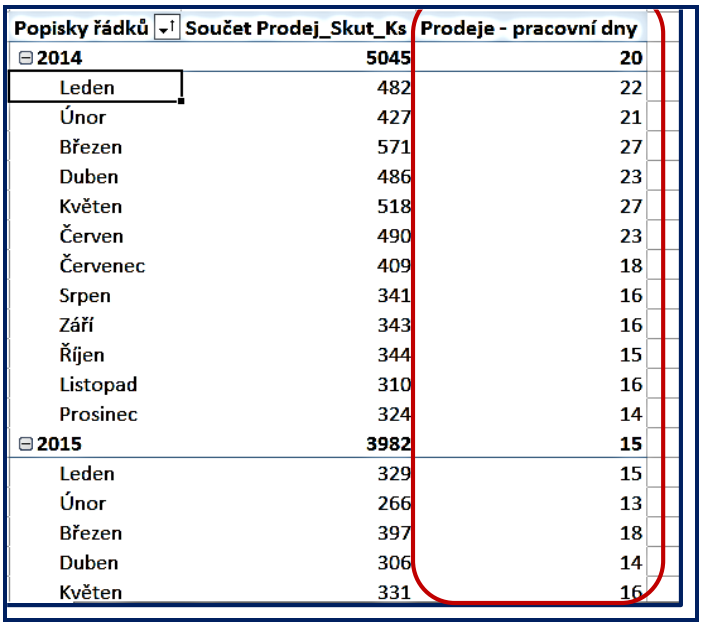
7.3 Ukazatelé pro pracovní dny

Častým případem funkcí s časovou dimenzí, jsou ***výpočty ukazatelů rozlišujících pracovní dny a svátky*** (např. objem prodeje v pracovních dnech oproti svátkům apod.). K tomu je třeba ***identifikovat v tabulce***, zda jde o svátek, či pracovní den, kde sloupec *Cas_Typ_Id* rozlišuje pracovní dny jako „1“ a svátky „0“.

Jako příklad požadujeme ***zjistit průměrné denní počty prodaných kusů zboží v pracovních dnech***. Výpočetní předpis pro míru pro tento účel bude mít následující tvar:

```
=SUM (FTQ_Prodej [Prodej_Skut_Ks] ) / SUM (DI_Cas_DAX [Cas_Typ_Id] )
```

kde souhrnné hodnoty prodaných kusů (*FTQ_Prodej [Prodej_Skut_Ks]*) se dělí počtem pracovních dnů v roce a v měsíci *(DI_Cas_DAX [Cas_Typ_Id])*. Výsledkem je definování míry „*Prodeje – pracovní dny“* a tabulka s průměrnými denními prodeji, viz Obrázek 7-1:



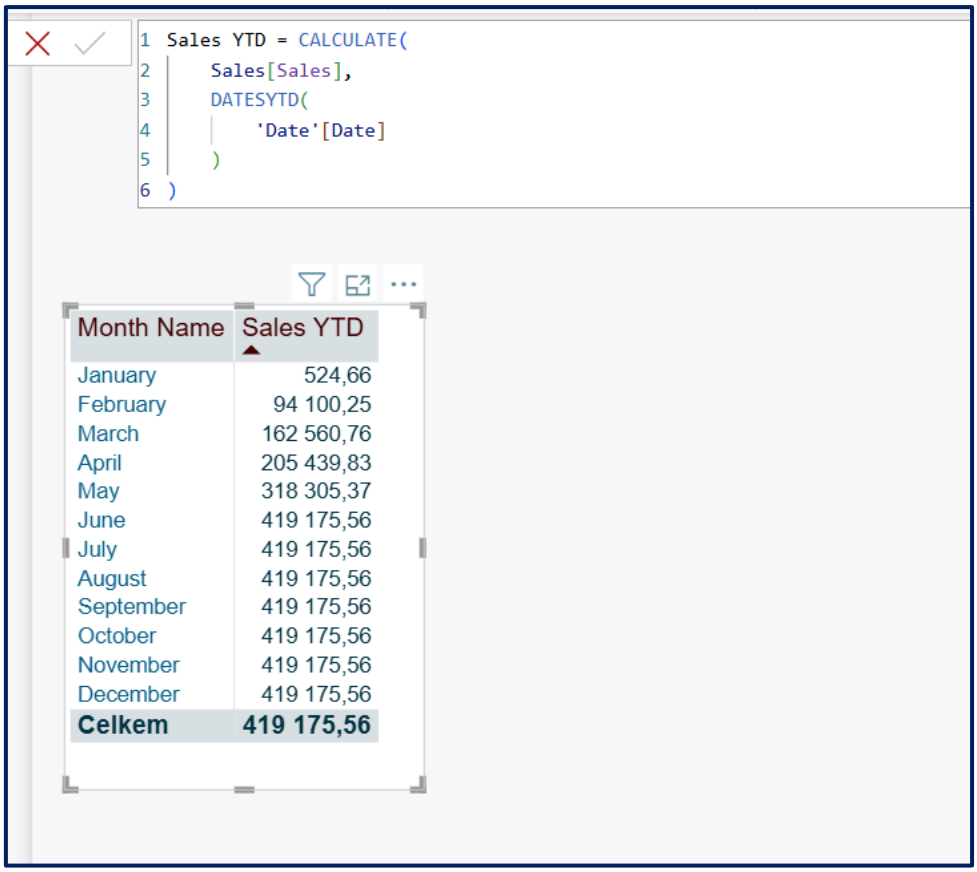
Popisky řádků	Součet Prodej_Skut_Ks	Prodeje - pracovní dny
2014	**5045**	**20**
Leden	482	22
Únor	427	21
Březen	571	27
Duben	486	23
Květen	518	27
Červen	490	23
Červenec	409	18
Srpen	341	16
Září	343	16
Říjen	344	15
Listopad	310	16
Prosinec	324	14
2015	**3982**	**15**
Leden	329	15
Únor	266	13
Březen	397	18
Duben	306	14
Květen	331	16

Obrázek 7-1: Průměrné prodeje v pracovních dní dle měsíců a roků

7.4 Funkce time intelligence – YTD, QTD, MTD

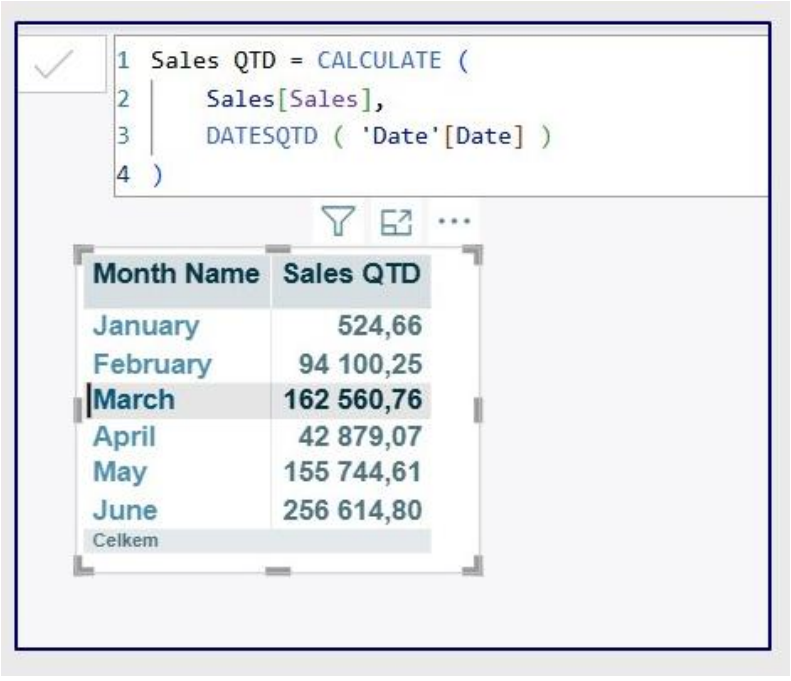
Často je v analýzách různých trendů a sezónních výkyvů nutné ***sledovat hodnoty ukazatelů v průběhu času. Jednou z nejčastějších kalkulací jsou ty, které jsou označeny jako YTD*** *(year-to-date)* zobrazující ***postupný nárůst hodnot od aktuálního data k začátku roku***. Obdobný princip platí pro kalkulace ***QTD*** *(quarter-to-date)* pro nárůst hodnot k začátku kvartálu a ***MTD*** *(month-to-date)* pro nárůst hodnot k začátku měsíce.

Příklad dokumentující funkci YTD představuje Obrázek 7-2



Obrázek 7-2: Uplatnění funkce year-to-date

Další příklad dokumentuje funkci QTD, Obrázek 7-3



Obrázek 7-3: Příklad funkce QTD

Dále předpokládejme požadavek na ***sledování nárůstu nákladů pro kvartály a měsíce*** vzhledem k začátku roku. K tomu se využívá funkce ***TOTALYTD*** a výpočetní předpis pro novou míru je následující, viz Obrázek 7-4:

=TOTALYTD (SUM (FTQ_Prodej [Naklady]), DI_Cas_DAX [FullDate])

Popisky řádků	Součet Prodej_Skut_Ks	Prodeje - pracovní dny	Naklady YTD
2014	**5045**	**20**	**1984264,3**
Leden	482	22	289420,8
Únor	427	21	449749,5
Březen	571	27	615059,7
Duben	486	23	773850,9
Květen	518	27	986188
Červen	490	23	1218019,7
Červenec	409	18	1421592,3
Srpen	341	16	1540746
Září	343	16	1648124,5
Říjen	344	15	1740276,9
Listopad	310	16	1822128,2
Prosinec	324	14	1984264,3
2015	**3982**	**15**	**1704670,8**
Leden	329	15	161702,1
Únor	266	13	327123,5
Březen	397	18	464421,4
Duben	306	14	575846,1
Květen	331	16	756987,3

List2 | List1

Obrázek 7-4: Objem nákladů k počátku roku (YTD)

V některý případech je ale třeba sledovat ***nárůst hodnot nikoli ke standardnímu začátku roku, ale např. k začátku fiskálního roku*** a jeho termín tak musíme doplnit do výpočetního předpisu míry, např.:

=TOTALYTD (SUM (FTQ_Prodej [Naklady]), DI_Cas_DAX [FullDate], „30-06“)

Kromě funkce *TOTALYTD* nabízí DAX v této souvislosti i ***řadu dalších užitečných funkcí***, jako jsou *STARTOFYEAR, ENDOFYEAR, PREVIOUSYEAR, NEXTYEAR, DATESYTD, OPENINGBALANCEYEAR, CLOSINGBALANCEYEAR.*

U většiny těchto funkcí ponecháváme s ohledem na rozsah na čtenáři, aby si je vyzkoušel i s pomocí funkcí nápovědy.

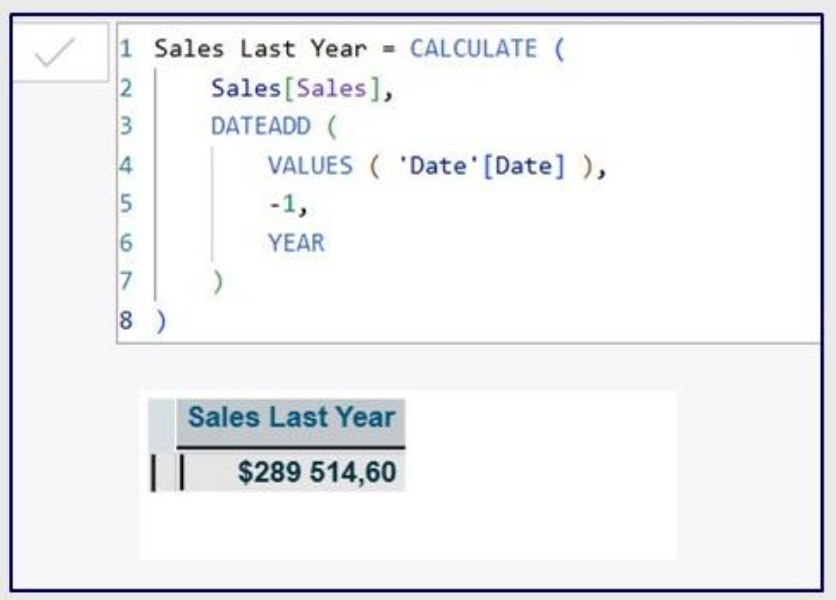
Je účelné ještě doplnit, že realizaci funkce *TOTALYTD* lze řešit ***prostřednictvím univerzální funkce CALCULATE().*** V tomto případě by adekvátní zápis pro výše uvedenou míru vypadal následujícím způsobem:

= CALCULATE (SUM (FTQ_Prodej [Naklady]), DATESYTD (DI_Cas_DAX [FullDate]))

Funkce DATESYTD vrací všechny datumy z tabulky kalendáře, resp. časové dimenze, a to ***na základě aktuálně nastaveného filtr kontextu***. To znamená, že funkce *CALCULATE()* zpracovává data pro YTD s respektováním aktuálně nastaveného filtru.

7.5 Sledování hodnot za minulé roky

Podnikové reporty a analytické tabulky často vyžadují hodnoty ukazatelů jak za aktuální období, resp. rok, tak ***za odpovídající období v minulých létech*** pro účely meziročních porovnání, vývojových trendů apod. Příklad tržeb za minulý rok prezentuje Obrázek 7-5:



Obrázek 7-5: Objem tržeb za minulý rok

V dalším případě požadujeme takové informace za minulá období pro sledování podnikových nákladů. Předpis pro míru bude např. vypadat takto, viz Obrázek 7-6:

=CALCULATE (SUM (FTQ_Prodej [Naklady]),SAMEPERIODLASTYEAR (DI_Cas_DAX [FullDate]))

Popisky řádků	Součet Prodej_Skut_Ks	Naklady YTD	Naklady minuly rok
2014	**5045**	**1984264,3**	
Leden	482	289420,8	
Únor	427	449749,5	
Březen	571	615059,7	
Duben	486	773850,9	
Květen	518	986188	
Červen	490	1218019,7	
Červenec	409	1421592,3	
Srpen	341	1540746	
Září	343	1648124,5	
Říjen	344	1740276,9	
Listopad	310	1822128,2	
Prosinec	324	1984264,3	
2015	**3982**	**1704670,8**	**1984264,3**
Leden	329	161702,1	289420,8
Únor	266	327123,5	160328,7
Březen	397	464421,4	165310,2
Duben	306	575846,1	158791,2
Květen	331	756987,3	212337,1

Obrázek 7-6: Náklady v aktuálním a minulém roce

Funkce *CALCULATE()* změní filtr s použitím funkce *SAMEPERIODLASTYEAR*(), která ***bude vracet soubor datumů z předchozího roku***. Tato funkce je specifickou verzí obecnější funkce *DATEADD()*, která navíc využívá počtu a typu období, která se mají zpětně sledovat. Ekvivalentní zápis k předchozímu bude vypadat takto:

```
=CALCULATE (SUM (FTQ_Prodej [Naklady] ),
    DATEADD (DI_Cas_DAX [FullDate], -1, YEAR ))
```

V některých případech je ale ***třeba porovnávat pouze celkovou hodnotu za předchozí rok***, nikoli jednotlivé dílčí, tedy za kvartály nebo měsíce. K tomu slouží funkce ***PARALLELPERIOD()***, která je obdobná funkci *DATEADD(),* ale vrací celé období specifikované třetím parametrem, namísto jednotlivých dílčích období. To znamená, že následující míra bude vždy obsahovat celkový objem nákladů za určený minulý rok:

```
=CALCULATE (SUM (FTQ_Prodej [Naklady] ),
    PARALLELPERIOD (DI_Cas_DAX [FullDate], -1, YEAR ))
```

7.6 Klouzavé ukazatele

Častou součástí podnikových reportů a analýz jsou klouzavé ukazatele. Příkladem je ***klouzavý roční souhrn*** (***moving annual total, MAT***), který sleduje souhrnné hodnoty za posledních 12 měsíců (označuje se také *LTM, last twelve month*). Např. MAT pro leden 2017 bude dán souhrnem hodnot od února roku 2016 do ledna roku 2017 apod. Příklad takové funkce dokumentuje následující zápis:

```
Klouzavy_souhrn_hodnot =CALCULATE (
        SUM (FTQ_Prodej [Naklady]),
            DATESBETWEEN
                (DI_Cas_DAX [FullDate],
```

```
NEXTDAY (SAMEPERIODLASTYEAR
        (LASTDATE (DI_Cas_DAX [FullDate] ))),
        LASTDATE(DI_Cas_DAX [FullDate] )
) )
```

V rámci tohoto předpisu je použita ***funkce DATESBETWEEN***, která ***vrací data ze sloupce uvedeného mezi dvěma uvedenými datumy***, a to mezi prvním a posledním dnem požadovaného intervalu. Poslední den se získá na základě funkce *LASTDATE*, která vrací poslední datum ze zadaného sloupce (*FullDate*) a vždy přitom respektuje aktuální filtr kontext. Od tohoto data se získá první den intervalu požadavkem na následující den funkcí *NEXTDAY* z odpovídajícího posledního data o rok předtím získaného funkcí *SAMEPERIODLASTYEAR*, viz Obrázek 7-7:

Popisky řádků	Naklady minuly rok	Klouzavy souhrn hodnot
⊟2014		1984264,3
Leden		289420,8
Únor		449749,5
Březen		615059,7
Duben		773850,9
Květen		986188
Červen		1218019,7
Červenec		1421592,3
Srpen		1540746
Září		1648124,5
Říjen		1740276,9
Listopad		1822128,2
Prosinec		1984264,3
⊟2015	1984264,3	1704670,8
Leden	289420,8	1856545,6
Únor	160328,7	1861638,3
Březen	165310,2	1833626
Duben	158791,2	1786259,5
Květen	212337,1	1755063,6

Obrázek 7-7: Klouzavý souhrn hodnot

7.7 Další funkce

K běžným dalším funkcím v tomto kontextu patří **průměry**. Např. pokud zjišťujeme průměrné náklady na prodaný výrobek, lze vytvořit následující míru:

```
Prumerne_naklady = SUM (FTQ_Prodej [Naklady] ) / SUM (FTQ_Prodej [Prodej_Skut_Ks] )
```

Takto získanou míru lze pak dále použít v dalších výpočtech, např. při porovnání průměrných nákladů oproti minulému roku, např.:

```
Prumerne_naklady_LY = CALCULATE
        ([Prumerne naklady ] , SAMEPERIODLASTYEAR (DI_Cas_DAX [FullDate] ) )
```

Standardní operací v reportingu jsou metriky Time intellifgence, jako například ***rozdíly hodnot aktuálního a minulého roku***. Např. objem nákladů v minulém roce je dán předpisem:

```
Naklady_LY = CALCULATE
    (SUM (FTQ_Prodej [Naklady]),SAMEPERIODLASTYEAR (DI_Cas_DAX [FullDate]))
```

Rozdíl hodnot k minulému roku, označuje se jako (*YOY, year-over-year*) je následující:

```
YOY_Naklady = SUM (FTQ_Prodej [Naklady] )  - [Naklady_MinRok])
```

v podílovém vyjádření, pak:

```
YOY_Podil_Naklady =
        IF  ( [Naklady_LY ] = 0,
                BLANK (),
                    [YOY_Naklady] / [Naklady_LY ] )
```

8. Závěr

Jazyk DAX je pro řešení a využití analytických úloh velmi důležitý prostředek. Vztahuje se jak na úlohy SSBI, zejména ve vztahu na Power BI, tak i na řešení komplexní BI úloh, např. v prostředí MS SQL Serveru. Je velmi účelné, pokud analytik principy a jednotlivé funkce a možnosti ovládá, neboť tak ve větším rozsahu může připravovat zadání analytických úloh, které budou přesněji odpovídat potřebám jednotlivých sfér řízení firmy. Např. s využitím funkcionality Time Intelligence lze sledovat vývojové trendy v obchodu, marketingu finančních zdrojích atd.

9. Zdroje

ASPIN, A., 2016: *Pro Power BI Desktop*. Apress. Staffordshire. 2016. ISBN 978-1-4842-1804-4.

BI4DYNAMICS, 2017. Sales – Top 30 customer table Report. (online). (29.08.2017). Dostupné z: http://www.bi4dynamics.com/business-intelligence-for-microsoft-dynamics-nav/content/

BSC Designer, 2017. Sales Business Unit Scorecard. online. Strategy Maps and KPIs. (online). (29.08.2017). Dostupné z: https://www.webbsc.com/s/sales-kpis.

CANVASJS, 2013. JavaScript Range Column & Range Bar Charts. (online). canvasjs.com. Dostupné z: https://canvasjs.com/javascript-range-column-range-bar-chart/.

CIMLER, P., ZADRAŽILOVÁ, D. a kol., 2007: *Retail management*. Praha, Management Press, 2007. ISBN: 978-80-7261-167-6

COLLIE,R.,SINGH, A., 2016: Power Pivot and Power BI, Holy Macro Books, 2016

CZSO, 2016. Tab. 02.05 Investice na ochranu životního prostředí (1989-2015). (online). czso.cz. Vydáváme. Česká republika od roku 1989 v číslech – 2016. https://www.czso.cz/csu/czso/ceska-republika-od-roku-1989-v-cislech-w0i9dxmghn#03

CZSO, 2017a. Česká republika: hlavní makroekonomické ukazatele. Vydáváme. Časové řady. czso.cz. (online). (03.07.2017). Dostupné z: https://www.czso.cz/csu/czso/hmu_cr.

CZSO, 2017b. Graf 3 Ceny bytů - ČR (index, 2010 = 100). Ceny bytů. Vydáváme. czso.cz. (online). (07.07.2017). Dostupné z: https://www.czso.cz/csu/czso/ceny_bytu.

CZSO, 2017c. Tab. 7 Stravování a pohostinství (CZ-NACE 56). Vydáváme. Obchod, pohostinství, ubytování - časové řady - Základní finanční ukazatele - čtvrtletní - Klasifikace NACE Rev. 2 (CZ-NACE) (online). www.czso.cz. (13.08.2017). Dostupné z: https://www.czso.cz/csu/czso/1-malzfu_b.

CZSO, 2017d. Peněžní vydání domácností podle počtu vyživovaných dětí. Veřejná databáze. czso.cz. (online). (13.08.2017). Dostupné z: https://vdb.czso.cz/vdbvo2/faces/index.jsf?page=vystup-objekt&pvo=ZUR10&z=T&f=TABULKA&katalog=30847&c=v3~8__RP2011&&str=v389.

Denver.edu, 2017. Line Graph, Bar Graph, Pie Chart and Scatter Plot. University of Denver. (online). (13.08.2017). Dostupné z: http://www.du.edu/ifs/help/use-online/repeated/general/graph/linebar.html.

ECKERSON, W., 2006. Deploying Dashboards and Scorecards [online]. (29.08.2017). Dostupné z:<http://www.businessobjects.com/pdf/products/performancemanageme nt/wp_tdwi_deploying_dashboards_and_scorecards.pdf>.

ECKERSON, W. 2010. Performance Dashboards: Measuring, Monitoring, and Managing Your Business, 2. vydání. Wiley. 336 stran. ISBN: 978-0-470-58983-0.

ECKERSON, W., W., 2006: Performance Dashboards. New Jersey, Johh Wiley & Sons 2006..

ENGLISH, L. P., 2003: *Improving Data Warehouse and Business Information Quality: Methods for reducing costs and increasing profits*. New York, John Wiley & Sons 2003. ISBN 0-471-25383-9

FEW, S., 2006. Information Dashboard Design: The Effective Visual Communication of Data. Sebastopol, O'Reilly Media. ISBN 978-0-596-10016-2.

FEW, S., 2012. Show Me the Numbers: Designing Tables and Graphs to Enlighten. 2nd edition. Burlingame, AnalyticsPress. ISBN 978-0-970-60197-1.

FEW, S., 2013. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring. 2nd edition. Burlingame, AnalyticsPress. ISBN 978-1-938-37700-6.

GALLAGHER, J., 2015. Antibiotic surge revealed by seasonal maps. bbc.com. News. (online). (03.07.2017). Dostupné z: http://www.bbc.com/news/health-34790038.

GAPMINDER, 2017. Life expectancy, years. (online). Gapminder.org. (13.08.2017). Dostupné z: http://www.gapminder.org/tools/#_chart-type=bubbles.

HARINATH, S., PIHLGREN, R., LEE, D.G., SIRMON, J, BRUCKNER, R.M., 2012: *Microdsoft SQL Server 2012. Analysis Services with MDX and DAX.* John Wiley and Sons, Inc., Indianopolis, 2012. ISBN 978-1-118-10110-0.

HURSMAN, A., 2010. Effective-dashboard-design-why-your-baby-is-ugly. (online). (13.08.2017). Dostupné z: https://www.slideshare.net/hursman/effective-dashboard-design-why-your-baby-is-ugly.

IMHOFF, C., WHITE, C., 2011: Self-Service Business Intelligence: Empowering Users to Generate Insights. Renton, WA : The Data Warehousing InstituteTM, 2011.

INETSOFT.com, 2017. Information about Scorecards and Scorecard Examples. (online). (29.08.2017). Dostupné z: https://www.inetsoft.com/info/information_about_scorecards_and_scorecard_examples/

INMON, B., 202: *Building the Data Warehouse*. Indianopolis, John Wiley and Sons 2002.

JOTHIGANESH, S., 2017. 3Dscatterplot. (online). (13.08.2017). Dostupné z: http://www.jothiganesh.com/category/technical/.

KIMBALL, R., ROSS, M., 2010: Relentlessly Practical Tools for Data Warehousing and Business Intelligence. Wiley Publishing, Inc. 2010. ISBN 978-0-470-56310-6.

KIMBALL, R., CASERTA, J., 2004: *The Data Warehouse ETL Toolkit*. Indianopolis, John Wiley and Sons 2004. ISBN 0-764-56757-8

KIMBALL, R., ROSS, M., 2002: *The Data Warehouse Toolkit, The Complete Guide to Dimensional Modelling*. Boston, John Wiley 2002. ISBN 0-471-20024-7

MICROSOFT, 2013: Power Pivot: Výkonné analýzy a modelování dat v Excelu. MICROSOFT. Office - Office.com [online]. 2013 [cit. 2014-01-02]. Dostupné z: http://office.microsoft.com/cs-cz/excel-help/power-pivot-vykonne-analyzy-a-modelovani-dat-v-excelu-HA102837110.aspx

OFFICE 1: Typy funkcí jazyka DAX. Office.com [online]. 2013 [cit. 2014-03-14]. Dostupné z: http://office.microsoft.com/cs-cz/excel-help/typy-funkci-jazyka-dax-HA102836089.aspx

OFFICE 2: Perspektivy v Power Pivotu. Office.com [online]. 2013 [cit. 2014-03-14]. Dostupné z: http://office.microsoft.com/cs-cz/excel-help/perspektivy-v-power-pivotu-HA102837427.aspx

OFFICE 3: Filtrování a zvýrazňování v Power View. Office.com [online]. 2013 [cit. 2014-03-22]. Dostupné z: http://office.microsoft.com/cs-cz/excel-help/filtrovani-a-zvyraznovani-v-power-view-HA102834776.aspx

KELLE ONEAL, BEYENETWORK, 2012, on-line: http://www.b-eye-network.com/blogs/oneal/archives/2012/02/what_is_the_dif.php

PROVOST, F., FAWCETT, T., 2013: Data Science for Business. What You Need to Know About Data Mining and Data-Analytic Thinking. O´Reilly Media. Sebastopol. 2013. ISBN: 978-1-449-36132-7.

RUSSO, M., FERRARI, A., 2013: *Microsoft Excel 2013: Building Data Models with PowerPivot.* California, O´Reilly Media, Inc., 2013. ISBN: 978-0-7356-7634-3.

RUSSO, M., FERRARI, A., 2011: *PowerPivot for Excel 2010. Give Your Data Meaning.* Redmond, Microsoft Press, 2011. ISBN: 978-0-7356-5058-0.

SEAMARK, P.: Beginning DAX with Power BI. Apress, 2018. ISBN: 978-1-4842-3477-8

SIEGEL, E., 2016: Predictive Analytics, The power to predict who will click, buy, lie, or die. John Wiley&Sons. New Jersey. 2016. ISBN: 978-1-119-14567-7.

SPOFFORD, G., 2001: *MDX Solutions with Microsoft SQL Server Analysis Services*. New York, John Wiley, 2001. ISBN: 0-471-40046-7.

TANKERSLY, B., 2017. Using Dashboards For a Real-Time View to Productivity. Intuit QuickBooks. (online). (29.08.2017). Dostupné z: https://www.firmofthefuture.com/content/using-dashboards-for-a-real-time-view-to-productivity/.

TURLEY, P., BRUCKNER, B., SILVA, T., WITHEE, K., PAISLEY, G., 2012: *Microdsoft SQL Server 2012. Reporting Services.* John Wiley and Sons, Inc., Indianopolis, 2012. ISBN 978-1-118-10111-7.

WEXLER, S., SHAFFER, J., COTGREAVE, A., 2017: The Big Book of Dashboards. John Wiley&Sons. New Jersey. 2017. ISBN: 978-1-119-28271-6.