

IT a anatomie firmy

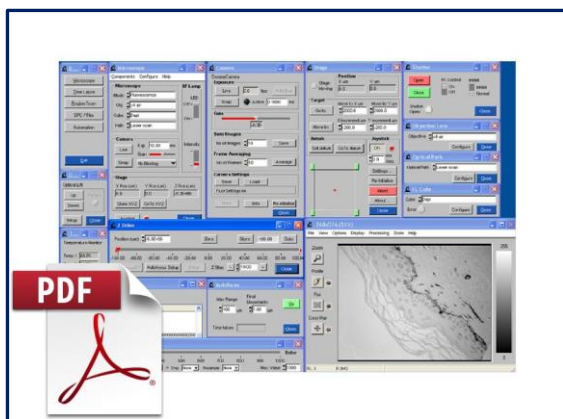
Jazyk DAX

(pracovní dokument)



MBI tým

VŠE Praha, 2023



[2] Základní principy a užití jazyka DAX

(základní vymezení pravidel, výpočetní předpis, základy vytvoření kalkulovaného sloupce a vytvoření kalkulované míry)

[3] Konstrukce jazyka DAX

(datové typy, operátory v DAX, proměnné (variables))

[4] Kontext vyhodnocování výpočetních předpisů

(filtr kontext – implicitní a explicitní, řádkový kontext, vlivy na kalkulované sloupce a míry)

[5] Kalkulované sloupce a kalkulované míry

(vytvoření kalkulovaného sloupce a příklady, kalkulované míry a příklady, funkce CALCULATED, funkce IF, iterátor, funkce CALCULATE s funkcí FILTER, výpočty na hierarchiích prvků, specifické analytické funkce)

[6] Řešení vazeb v datech

(řešení úloh s využitím standardních vazeb v datovém modelu, řádkový a filtr kontext s vazbami tabulek, funkce CROSSJOIN, funkce GENERATE)

[7] Time intelligence

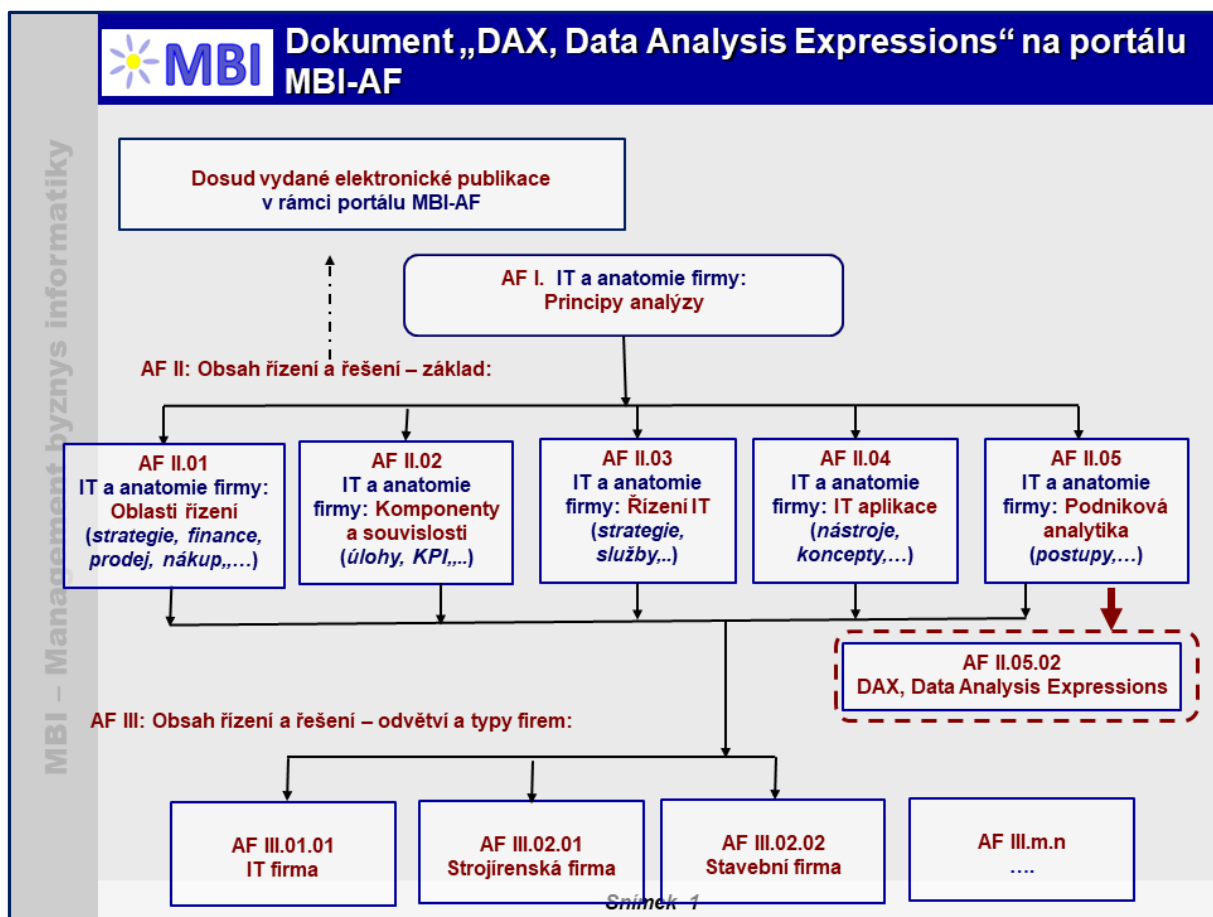
(principy time intelligence, generování dimenzionální tabulky času, ukazatele pro pracovní dny, funkce YTD, QTD, MTD, sledování hodnot za minulé roky, klouzavé ukazatele, specifické funkce)



DAX (Data Analysis Expressions) je výkonný programovací jazyk implementovaný v prostředí **Power BI**, **SSAS** a **Power Pivot** pro pokročilé vytváření BI a SSBI aplikací.

Účelem dokumentu je vymezit podstatné funkce jazyka, vysvětlit tzv. kontext vytváření a provádění kalkulací a na příkladech ukázat specifické funkce a postupy uplatňované v DAX.

Dokument představuje doplnění k základnímu dokumentu orientovanému na podnikovou analytiku: „AF_II_05_Podnikova_Analytika.pdf“. Jeho místo ve struktuře portálu MBI-AF ukazuje další obrázek:



Obsah:

1.	<i>Úvod</i>	6
1.1	Podniková analytika	6
1.2	Power BI, desktop	7
1.3	Oblasti řízení firmy	8
1.4	Komponenty řízení firmy	8
2.	<i>Základní principy a užití DAX</i>	9
2.1	Výpočetní předpis v DAX	9
2.2	Vytvoření kalkulovaného sloupce	10
2.3	Vytvoření kalkulované míry	10
2.4	Podpora pro výpočet měř v Power BI – Rychlá míra	11
3.	<i>Konstrukce jazyka DAX</i>	12
3.1	Datové typy	12
3.2	Operátory	12
3.3	Proměnné (Variables)	13
4.	<i>Kontext vyhodnocování výpočetních předpisů</i>	14
4.1	Filtr kontext	14
4.1.1	Implicitní filtr kontext	14
4.1.2	Explicitní filtr kontext	15
4.2	Řádkový kontext	15
5.	<i>Kalkulované sloupce a kalkulované míry</i>	17
5.1	Vytvoření kalkulovaného sloupce	17
5.2	Příklady kalkulovaných sloupců	18
5.3	Kalkulované míry	18
5.4	Funkce CALCULATE()	20
5.5	CALCULATE() v řádkovém kontextu	21
5.6	Parametr ALLSELECTED	21
5.7	Funkce IF ()	21
5.8	Iterátor	22
5.9	CALCULATE () a využití FILTER ()	23
5.10	Využití hierarchií prvků dimenzí	23
5.10.1	Výpočty v hierarchiích prvků	24
5.11	Další analytické funkce založené na DAX	24
5.11.1	Seskupování dat, <i>banding</i>	24
5.11.2	Vytvoření pořadí, <i>ranking</i>	24
5.11.3	Růst počtu zákazníků	25
6.	<i>Řešení vazeb</i>	26
6.1	Řešení ve standardních vazbách	26

6.1.1	Řádkový kontext s vazbami tabulek.....	26
6.1.2	Filtr kontext s vazbami tabulek	28
6.2	Parametrické nepropojené tabulky.....	29
6.3	Funkce CROSSJOIN.....	29
6.4	Funkce GENERATE	30
7.	<i>Time Intelligence</i>.....	31
7.1	Základní principy time intelligence.....	31
7.2	Vygenerování dimenzionální tabulky času	31
7.3	Ukazatele pro pracovní dny	32
7.4	Funkce YTD, QTD, MTD	32
7.5	Sledování hodnot za minulé roky	33
7.6	Klouzavé ukazatele	34
7.7	Další funkce	34
8.	<i>Závěr</i>	36
9.	<i>Zdroje</i>	37

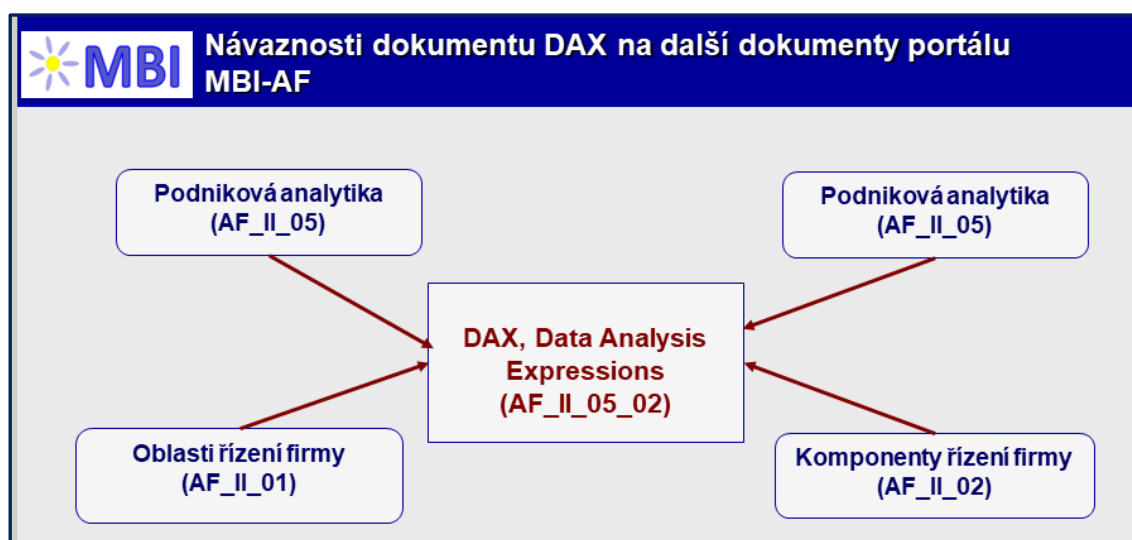
1. Úvod



DAX (*Data Analysis Expressions*) je programovací jazyk využívaný a implementovaný v prostředí **Power BI**, **SSAS** a **Power Pivot** pro pokročilé vytváření BI a SSBI aplikací. DAX je **schopen pracovat s daty uloženými v tabulkách datového modelu**, resp. databáze Power BI a SSAS Tabular model, Power Pivot **Má vlastní syntaxi, systém funkcí** a další součásti.

Účelem kapitoly je **charakterizovat celý dokument a jeho návaznosti** na ostatní dokumenty na portálu MBI-AF.

Dokument **charakterizuje jazyk DAX**, jeho syntaxi, pravidla, možnosti využití. Na druhé straně je dobré se na řešení úloh s využitím DAX dívat **v širším kontextu**, zejména z pohledu nároků úloh byznysu, ve kterých má být využíván. Stejně tak je účelné respektovat pravidla a přístupy k řešení **celé podnikové analytiky** a zejména samotného produktu **Power BI**. **Vazby** na tyto další dokumenty prezentuje Obrázek 1-1. V dalším textu uvedeme jejich stručnou charakteristiku.



Obrázek 1-1: Vazby dokumentu na ostatní dokumenty v rámci MBI-AF


1.1 Podniková analytika

Dokument „*Podniková analytika*“ je **základním dokumentem**, na který tento dokument navazuje. Snaží se poskytnout **celkový**, byť relativně stručný, **přehled** o principech, postupech, produktech, problémech i řešeních podnikové analytiky v praxi. Zahrnuje otázky jak „**základní analytiky**“ postavené většinou na nástrojích a přístupech business intelligence, self service business intelligence nebo competitive intelligence, tak **pokročilé analytiky**.



Odkaz na dokument: „AF_II_05_Podniková_analytika.pdf“.

Strukturu dokumentu představuje Obrázek 1-2:

 Podniková analytika	
[B]] Obsah a principy podnikové analytiky	
[C]] Nástroje a řešení pro základní podnikovou analytiku	
[D]] Komponenty podnikové analytiky	[E]] Reporting a vizualizace dat
[F]] Pokročilá podniková analytika – nástroje, řešení	
[G]] Data pro podnikovou analytiku	[H]] Podniková analytika na velkých datech
[I]] Podniková analytika a cloud computing	[J]] Řízení podnikové výkonnosti
[K]] Doporučené postupy v řešení podnikové analytiky	

Obrázek 1-2: Podniková analytika, obecně

1.2 Power BI, desktop

DAX, jak bylo uvedeno se využívám v prostředí Power BI, SSAS, Power Pivot, ale lze říci, že Power BI je zde primární. Proto vazbu na dále uvedený dokument pokládáme za základní:



Odkaz na dokument: „AF_II_05_01_Power BI.pdf“.

<p>[A] Power BI: předpoklady, vstupy</p> <p><i>(principy self service business intelligence, postup řešení úloh SSBI, datové modelování, dimenzionální modelování,)</i></p>
<p>[B] Principy a charakteristiky Power BI (PBI)</p> <p><i>(principy a komponenty PBI, instalace PBI, Postup vytvoření datové sady a aplikace, vytvoření uživatelského rozhraní – reporty a vizuály, filtrování, průřezy, zvýrazňování, a drilování, Power BI Service, mobilní aplikace PBI, Režim přímého dotazování – Direct Query)</i></p>
<p>[C] Uplatnění Power BI podle oblastí řízení firmy</p> <p><i>(strategie firmy, finanční řízení, obchod,...)</i></p>

Obrázek 1-3: Struktura dokumentu Power BI

1.3 Oblasti řízení firmy

Při řešení jednotlivých aplikací a uplatnění DAX je nezbytné potřeby těchto oblastí řízení v rámci firmy identifikovat a analyzovat. Ty jsou obsahem dokumentu, jehož **odkaz a strukturu** představuje další část.



Odkaz na dokument: „AF_II_01_Oblasti.pdf“.

[1] Strategické řízení firmy				
[2] Finanční řízení	[3] Závazky	[4] Pohledávky	[5] PAM	[6] Controlling
[7] Prodej	[8] Nákup	[9] Sklady		[10] Personál
[11] Majetek	[12] Marketing	[13] Doprava		[14] Energie

Obrázek 1-4: Oblasti řízení firmy

1.4 Komponenty řízení firmy

Dokumentace jednotlivých komponent řízení, **zejména metrik, dimenzí a datových zdrojů a metodik a metod** je relativně rozsáhlá, a proto je zde vyčleněna do zvláštního dokumentu. Ty představují podklady pro řešení aplikací. Odkaz a struktura dokumentu jsou:



Odkaz na dokument: „AF_II_02_Komponenty.pdf“.

[1] Úlohy	
[2] Metriky, ukazatelé	[3] Analytické dimenze
[4] Data, dokumenty	[5] Role
[6] Faktory: firemní prostředí	[7] Faktory: řízení a organizace
[8] Metodiky a metody řízení firmy	[9] Metodiky a metody řízení IT
[10] Metodiky a metody řešení IT	

Obrázek 1-5: Komponenty řízení firmy a jejich souvislosti

Další text dokumentu poskytuje **základní informace o jazyku DAX**. Navazuje a vychází z původní publikace (Stanovská a další, 2018). Příklady se vesměs váží k DAX v prostředí Power BI, pouze ve výjimečných případech k Power Pivot.

Na druhé straně se i tento jazyk velmi rychle a silně vyvíjí, a proto i tento dokument **podléhá aktualizacím** a současně se zde odvoláváme na různé **tutoriály společnosti Microsoft**.

2. Základní principy a užití DAX



Účelem této vstupní kapitoly je vytvořit pouze **ve stručné formě předpoklad** pro vytváření kalkulačí s pomocí jazyka DAX, **bez dalších detailnějších informací**. K těm se vrátíme v dalších kapitolách.

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 dokumentu určitý **širší základ jazyka DAX** 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 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, nebo vybírat je, pouze dotazovat nebo filtrovat.

2.1 Výpočetní předpis v DAX

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

Každý **výpočetní předpis v DAX začíná rovnítkem „=“** nebo přiřazením „:=“, jednotlivé prvky předpisu výlučně používají pro identifikaci názvu tabulky a názvu sloupce v tabulce. Na příklad 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 kalkulačích musí být vždy uzavřeny v hranatých závorkách.

kde *FTQ_Prodej* je název tabulky a *Prodej_Skut_Ks* a *Cena_Ks* jsou názvy sloupců, které v kalkulačích 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 Prodej'*.

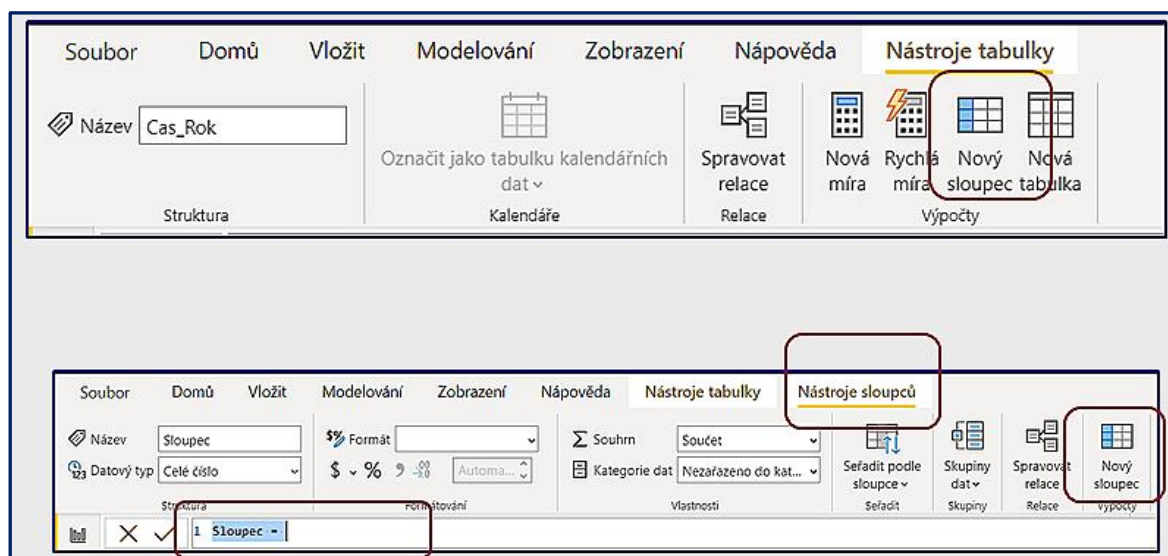
V rámci DAX se uplatňují podle daných pravidel (viz další kapitoly) dva typy výpočtů, resp. kalkulačí:

- **kalkulovaný sloupec**, realizovaný pouze v rámci jedné (každé) řádky datové tabulky,
- **kalkulovaná míra**, kde se výpočet uskutečňuje na úrovni celé datové tabulky,
- **kalkulované tabulky**, 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.

2.2 Vytvoření kalkulovaného sloupce

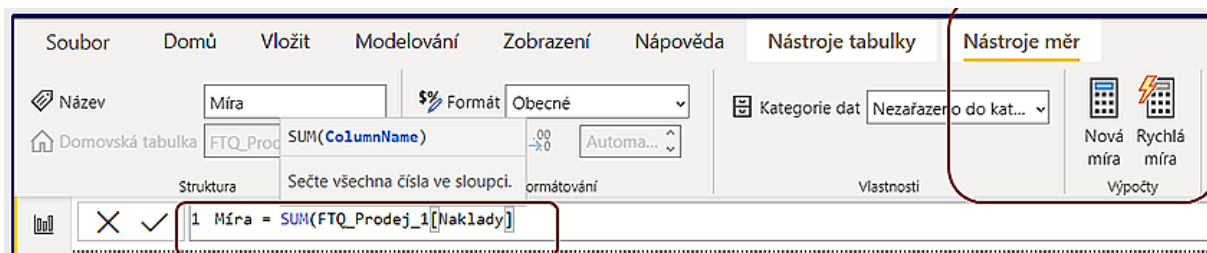
Kalkulované sloupce jsou v Power BI přidány do tabulek, v nichž jsou definovány. Volbou v menu nebo v seznamu polí tabulky **Nový sloupec se zpřístupní příkazová řádka** a uživatel do ní запиše příslušný příkaz DAX. Obrázek 2-1 ukazuje založení příkazu DAX pro přidání nového kalkulovaného sloupce.



Obrázek 2-1: DAX výraz pro jednoduchý výpočet kalkulovaného sloupce

2.3 Vytvoření kalkulované míry

Míra, resp. *measure*, nebo **kalkulovaná míra**, resp. *počítané pole*, *calculated field*, se počítá na agregační úrovni dat, tj. **za tabulku** nebo její podmnožinu. Data jsou tak vypočítávána na úrovni agregací (nikoli 1 řádky) a s respektováním filtrů, které ovlivňují jednotlivé buňky.



Obrázek 2-2: 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*.



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

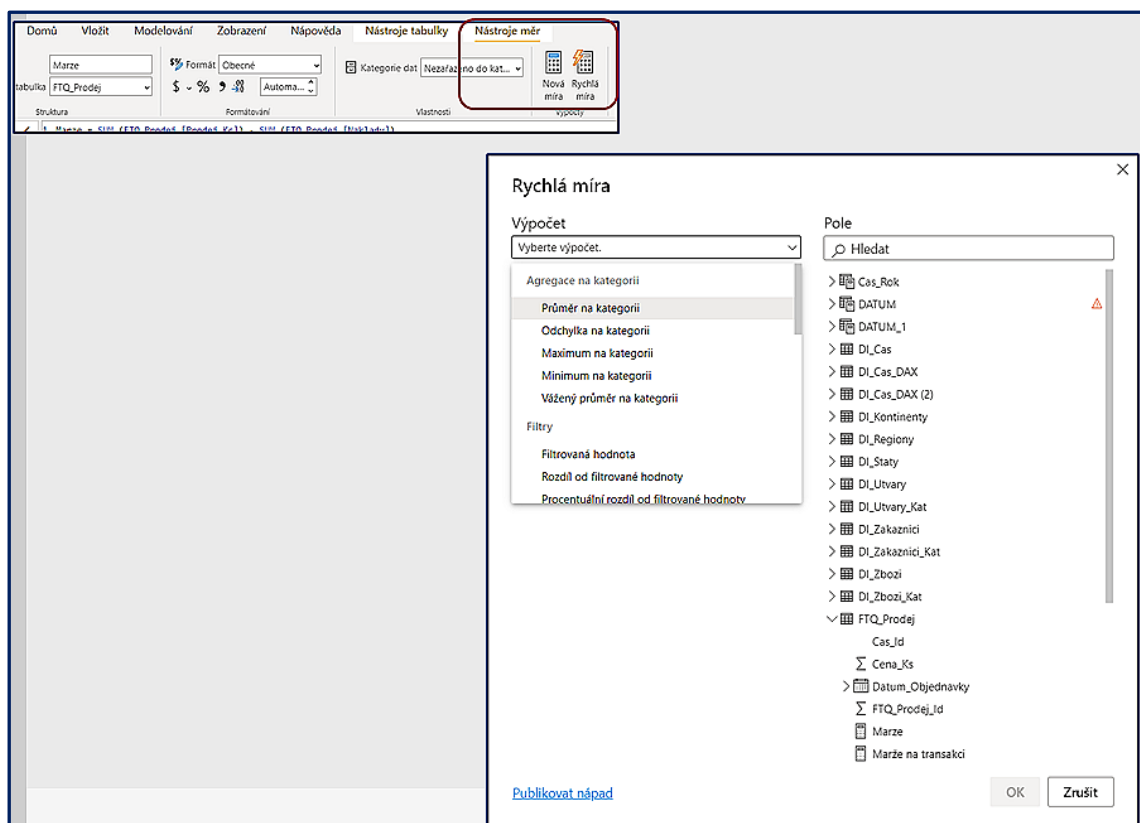
V každém případě je třeba, aby **byla vybrána dimenzionální nebo faktová tabulka**, do které se příslušný sloupec nebo míra zařadí. **V případě míry** je zřejmé, že **nemusí být v podstatě zařazena do žádné tabulky**, nebo by mohla vzniknout **tabulka obsahující pouze vypočtené míry**, ale pro práci uživateli je srozumitelnější, aby i míry byly dostupné ze seznamu polí tabulek, k nimž logicky nejvíce patří. Tak například vypočtená míra Marže logicky přísluší k prodejem zboží, je tedy zařazena mezi

poli 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 datového modelu **jedinečné**.

2.4 Podpora pro výpočet měř v Power BI – Rychlá míra

Power BI kromě přímého definování vypočtených měř zápisem příkazů v jazyce DAX (viz výše) nabízí k použití funkcionalitu **pro rychlé vytváření nejobvykleji používaných měř** v aplikacích BI. Tato funkcionalita je dostupná pod **volbou Rychlá míra** (Obrázek 2-3 **Chyba! Nenalezen zdroj odkazů.**, nahore) nebo na pravý klik v seznamu polí. Uživatel vybírá typ kalkulace, pole z tabulek datového modelu a další parametry podle typu kalkulace. Power BI **na pozadí generuje příkaz v DAXu**.



Obrázek 2-3: Definování rychlé míry

Postup definice **rychlé míry je následující**. Uživatel vybere **typ míry v poli Výpočet**. Potom jednoduchým tažením myši 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.

Další doplňující informace ke kalkulovaným sloupcům a mírám jsou doplněny v další kapitolách, zejména v kapitole 4 věnované kontextu v jazyce DAX

3. 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.

3.1 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 DAX 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 konkaténace, 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.

3.2 Operátory

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

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

Pořadí operátorů podle priorit je následující:

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

3.3 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 * 10
```

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



```
VAR promText1 = „Zjistí“
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.

4. 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 kalkulovalých měř.**

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 datovém 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.

4.1 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** 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**.

4.1.1 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.

4.1.2 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 podle 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 datovém 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.

4.2 Řádkový kontext

Řádkový kontext je využíván **při tvorbě kalkulovaných sloupců** nebo v případě využití tzv. **iterátorů** (viz dále). Jako v případě filtr kontextu se vztahuje k výpočtům pro každou buňku výstupní tabulky. Pokud se např. vytváří kalkulovaný sloupec pro tabulku o 20 řádcích pak výpočet se bude opakovat 20x vždy s mírně odlišným řádkovým kontextem.

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 datového modelu, zatímco **hodnoty kalkulované míry** se přepočítávají vždy při změně filtru, který ji ovlivňuje.

Platí, že pokud jsou data do datového modelu uložena nebo obnovena, už nemohou být dále měněna.

5. Kalkulované sloupce a kalkulované míry



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

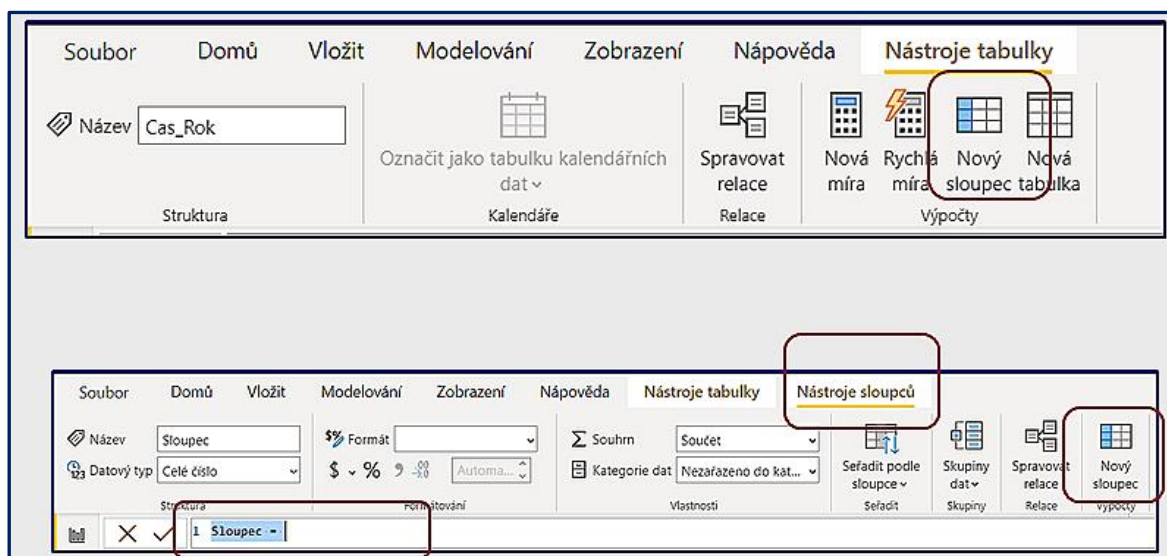
DAX umožňuje definovat kalkulované sloupce tedy vlastní výpočty v tabulkách Power BI. **Kalkulovaný sloupec** je vypočítáván **v řádkovém kontextu**, tj. v kontextu jednotlivých řádek tabulky a vytváří v tabulce nový kalkulovaný sloupec z hodnot položek a konstant dané řádky. **Kontext řádky** tak znamená, 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.

- **Kalkulovaný sloupec** je vypočítáván převážně **v kontextu jednotlivých řádek** tabulky (*row context*), tj. vytváří se v tabulce nový (kalkulovaný sloupec) z hodnot položek, konstant v rámci jedné řádky a odpovídajících řádek v tabulkách majících s tabulkou vztah.
- **Míra se počítá na agregační úrovni dat, tj. za tabulku** nebo podmnožinu jejích řádek a musí respektovat kontext každé buňky, včetně **nastavených filtrů a průřezů**.

5.1 Vytvoření kalkulovaného sloupce

Kalkulované sloupce jsou v Power BI přidány do tabulek, v nichž jsou definovány. Volbou v menu nebo v seznamu polí tabulky **se zpřístupní příkazová řádka** a uživatel do ní запиše příslušný příkaz DAX:



Obrázek 5-1: DAX výraz pro jednoduchý výpočet kalkulovaného sloupce

Příkazový řádek pro zápis výrazů DAX se zpřístupní také v případě volby *Nová míra* nebo *Nový sloupec* v seznamu polí tabulek modelu. Nabídka je dostupná na pravé tlačítko myši.

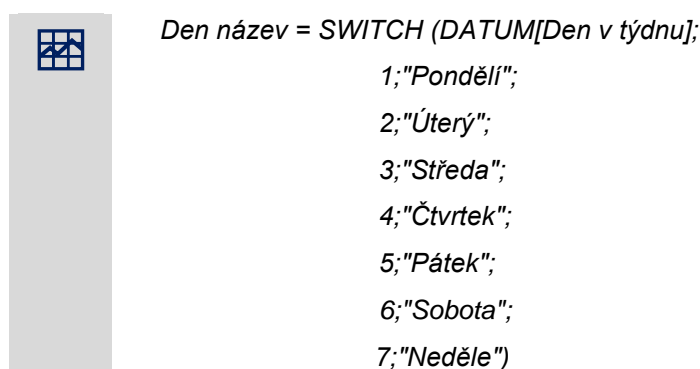
K vytváření kalkulovaných sloupců doplníme **několik poznámek**:

- 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.

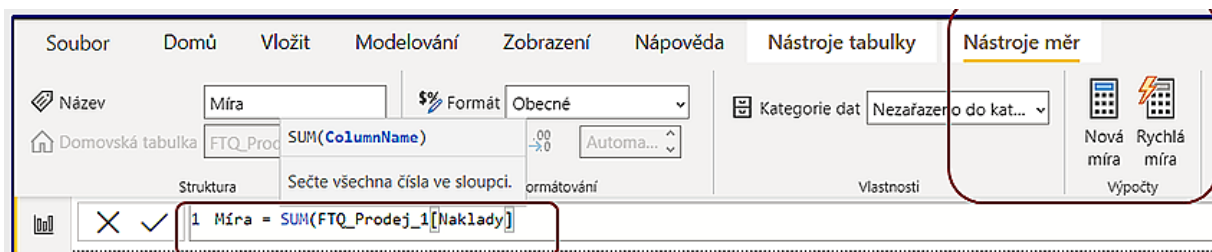
5.2 Příklady kalkulovaných sloupců

Příkladem pro **přidání kalkulovaného sloupce Den název do tabulky DATUM** je demonstrace **funkce SWITCH** pro přidání sloupce s českým názvem dne dle hodnoty pořadového čísla dne v týdnu ve sloupci *Den* v *týdnu*:



5.3 Kalkulované míry

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. Příklad vytvoření míry ukazuje Obrázek 2-2:



Obrázek 5-2: Vytvoření nové kalkulované míry

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

- všechny *míry se nabízejí v řídicích oknech* výstupních tabulek,
- **úpravy míry se promítají** do dalších měř, 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,
- míry **se vypočítávají a vyhodnocují na zdrojových datech**, nikoli na datech výstupní tabulky (*pivot table*). Na příklad *Celkový součet* není suma polí nad danou položkou v kontingenční tabulce, ale pro kontrolu je třeba zobrazit data zdrojové tabulky,
- **každá buňka** pro míru je vypočítávána **nezávisle**,
- **Identifikují se „souřadnice“** pro danou míru, tj. prvky dimenzí odpovídajících řádku, sloupce a filtru a na tomto základě se provádějí výpočty příslušné míry.

Příkladem pro práci s mírou může být i celková marže z prodeje:



$\text{=SUM (FTQ_Prodej [Prodej_Kc]) - SUM (FTQ_Prodej [Naklady])}$

Další příklady vytváření kalkulovaných měr jsou v dalším přehledu:

- **výpočet míry pro náklady** (v tomto případě je použit **iterátor SUMX**):



$\text{Náklady} = \text{SUMX (FTQ_Prodej; FTQ_Prodej [Náklady prodeje])}$

- **míry pro tržby** (funkce **SUM**):



$\text{Tržby} = \text{SUM (FTQ_Prodej [Cena prodeje])}$

- obě vypočtené míry jsou následně použity **při výpočtu marže**:



$\text{Marže} = \text{CALCULATE(FTQ_Prodej [Tržby] - FTQ_Prodej [Náklady])}$

- pro výpočet počtu prodejních transakcí, jednotlivých řádků lze využít následující míru, která spočítá počet řádek faktové tabulky, tedy počet prodejních transakcí



$\text{[Pocet_Transakci]} := \text{COUNTROWS (FTQ_Prodej),}$

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



$\text{Marže na transakci} = \text{[Marže] / [Pocet_Transakci]}$

- **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 objednávek:



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

5.4 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 presentované ve formě tabulky**, představují seznamy hodnot, 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 2016**:



=CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]); DI_Cas [Cas_Rok] = 2016)

Další příklady:

- Náklady na zboží pouze v roce 2016 a současně za zboží s identifikátorem 35:



= CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]); DI_Cas[Cas_Rok] = 2016;
DI_Zbozi[Zbo_Id] = 35)

- Pokud je potřeba 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)

5.5 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 (*Suma_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.

5.6 Parametr 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.

5.7 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.

5.8 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**:



*=SUMX (FTQ_Prodej; FTQ_Prodej [Naklady] * 1,05)*

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, ADDCOLUMNS) tento krok neprovádějí.

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 podle regionů**, 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))
```

5.9 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 ))
```

5.10 Využití hierarchií prvků dimenzí

Hierarchické struktury prvků dimenzí patří k základním principům Business Intelligence aplikací, neboť umožňují uživateli se v nich pohybovat po takové podrobnosti informací, které momentálně potřebuje.

Hierarchie prvků tak představují předdefinované cesty, kterými se uživatel může dostávat ke svým datům, a proto musí být tyto cesty navrhovány a implementovány velmi zodpovědně. Hierarchie mohou být vyjádřeny v jedné tabulce nebo ve více propojených tabulkách, jak bylo uvedeno již uvedeno v souvislosti se schématy *STAR* a *SNOWFLAKE*.

Druhou možností je vyjádření hierarchie v několika propojených tabulkách na bázi *SNOWFLAKE*. V této souvislosti je dobré upozornit i na možnost, jak s pomocí DAX transformovat v hierarchii data několika propojených tabulek do jedné a realizovat tak i jejich spojení s využitím funkce *RELATED*. Předpokládejme hierarchii v dimenzi zboží založenou na vazbách tabulek kategorie zboží (*DI_Zbozi_Kat*) - skupiny zboží (*DI_Zbozi_Skupina*) - jednotlivé položky zboží (*DI_Zbozi*). Pokud bude účelné promítnout některé údaje, jako např. název z vyšších úrovní hierarchie do nižší a současně i propojit tabulky v jedné, tedy v *DI_Zbozi*, pak lze použít následujícího zápisu pro vytvoření nového kalkulovaného sloupce:



```
DI_Zbozi [Skupina_Zbozi] := RELATED (DI_Zbozi_Skupina [Zbo_Nazev_Skupina] )  
DI_Zbozi [Kategorie_Zbozi] := RELATED (DI_Zbozi_Kat [Zbo_Nazev_Kat] )
```

5.10.1 Výpočty v hierarchiích prvků

Častým případem je promítání různých typů výpočtů do pivot tabulek založených na hierarchiích prvků dimenzí. Výpočty lze na hierarchiích prvků dimenzí realizovat i pomocí jazyka DAX a zejména funkce *CALCULATE()*. Míry pro hierarchie prvků na základě funkce *CALCULATE()* jsou součástí celého datového modelu a jsou použitelné kdykoli. Na druhé straně má tato druhá varianta několik nevýhod:

- vytvoření kompletních výpočtů na hierarchiích prvků je relativně složité a dosti pracné,
- výpočty na bázi DAX jsou v tomto případě méně flexibilní, a pokud se hierarchie změní, pak výpočty mohou poskytovat chybné výsledky.

5.11 Další analytické funkce založené na DAX

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

5.11.1 Seskupování dat, *banding*

V podnikových analýzách je často potřeba **sdužit 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 datového 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] )
)
```

5.11.2 Vytvoření pořadí, *ranking*

Vytvoření **pořadí různých objektů** podle stanovaných hodnot je rovněž častou součástí reportů a analýz. Předpokládejme, že požaduje **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 (DI_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.

5.11.3 Růst počtu zákazníků

Účelem bude sledovat nárůst počtu zákazníků v jednotlivých letech. 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*):



2014_Zakaznici :=
CALCULATE ([Aktivni_Zakaznici]; FTQ_Prodej [Rok] = 2014)

Ve třetím kroku se bude počítat **procentuální nárůst zákazníků** vzhledem k roku 2014



Aktivni_Zakaznici_Rust :=
DIVIDE ([Aktivni_Zakaznici] - [2014_Zakaznici]; [2014_Zakaznici])

6. 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 datovém modelu. **Nikoli** ale **ve všech případech**. K tomu se používá **řada funkcí**, zejména:

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

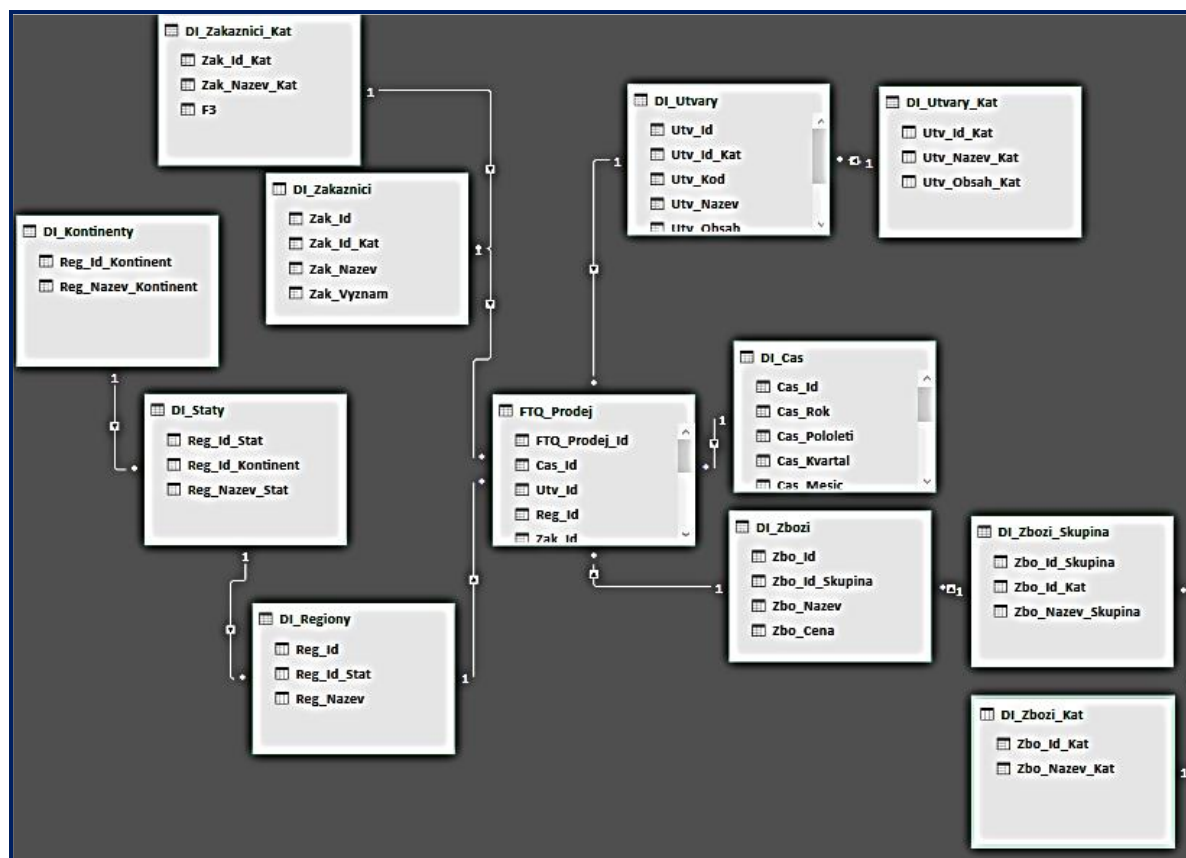
6.1 Řešení ve standardních vazbách

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*“),
- **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 měř vázaných na různé faktové tabulky** v jedné výstupní tabulce.


6.1.1 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**. 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 6-1. V tomto případě **je podstatná kardinalita a směr vazby**, tedy pro *FTQ_Prodej : DI_Zbozi* je kardinalita vazby **M : 1**.



Obrázek 6-1: Příklad vazeb pro řádkový kontext s vazbami tabulek

Pak lze požadovaný **kalkulovaný sloupec** v řádkovém kontextu zapsat následujícím výrazem:

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

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 by ale řádkový kontext na straně vazby „1“ byl 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))`

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 dle 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:



```
Pocet dodanych = CALCULATE (FTQ_Prodej [Prodano ks];
                             USERELATIONSHIP (FTQ_Prodej [Datum_Dodani]; DI_Cas [Cas_Datum])
                             )
```

6.1.2 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 měř a filtr kontextu** rovněž v prostředí několika tabulek s definovanými vazbami (viz Obrázek 6-1).

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 obsahující 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**.

6.2 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 datového 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 6-2) představuje možné kursy Kč vzhledem k Euro, 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 6-2: 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:



$$\text{Korun_za_Euro} := \text{MAX} (\text{CZK_EUR} [\text{Kc_Euro}])$$

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 aktuálního kursu Kč k Euro:



$$\text{Cena_Euro} := [\text{Průměr Cena_Ks}] / [\text{Korun_za_Euro}]$$

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.

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

6.3 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.

6.4 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 funkci GENERATE a nikoli CROSSJOIN

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



Pocet radků = 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:



```
Pocet radků = COUNTROWS (GENERATE ('TabulkaX',  
                                     FILTER ('TabulkaY',  
                                             'TabulkaX' [Cena] = 'TabulkaY' [Cena])  
                                     )  
)
```

7. Time Intelligence



Účelem této kapitoly je presentovat 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 nejrozličnějších 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 případně další potřebné údaje pro analýzy a podnikový reporting (např. plné názvy měsíců a dnů, určení pracovních dnů atd.). Pozn.: název tabulky je *DI_CAS_DAX*, abychom vyjádřili její pracovní charakter pro úlohy Time Intelligence v prostředí DAX.
- Pro **identifikaci jednotlivých období**, většinou dní je k dispozici primární klíč (např. jako umělý klíč na obrázku, 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.
- **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 **nevýhneme vytvoření více tabulek pro časovou dimenzi**. To je v případech, kdy je třeba současně rozlišovat různé časové okamžiky. Na příklad 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 datумы 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 objednávky] ) ); 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" );
"MěsícZkratka"; FORMAT ( [Date]; "mmm" );
"Měsíc"; FORMAT ( [Date]; "mmmm" );
"Den v týdnu"; WEEKDAY ( [Date]-1 );
"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 )
```

7.3 Ukazatele 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 prodejmi.

7.4 Funkce 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.

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:



```
=TOTALYTD ( SUM ( FTQ_Prodej [Naklady] ); DI_Cas_DAX [FullDate] )
```


V některých 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 datумы 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. V našem případě požadujeme takové informace za minulá období pro sledování podnikových nákladů. Předpis pro míru bude např. vypadat takto:



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

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 daty**, 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**.

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:



```
Prum_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.:



```
Prum_Naklady_MinRok := CALCULATE (
    [Prum_Naklady] ; SAMEPERIODLASTYEAR (DI_Cas_DAX [FullDate] ) )
```

Standardní operací v reportingu jsou **rozdíly hodnot aktuálního a minulého roku**. Např. objem nákladů v minulém roce je dán předpisem:



```
Naklady_MinRok := 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:



$YOY_Podil_Naklady :=$
 $IF([Naklady_MinRok] = 0;$
 $BLANK();$
 $[YOY_Naklady] / [Naklady_MinRok])$

8. Závěr

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/performancemanagement/wp_tdw_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., 2006: Performance Dashboards. New Jersey, John 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: *Microsoft SQL Server 2012. Analysis Services with MDX and DAX*. John Wiley and Sons, Inc., Indianapolis, 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., 2002: *Building the Data Warehouse*. Indianapolis, 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*. Indianapolis, 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: *Microsoft SQL Server 2012. Reporting Services*. John Wiley and Sons, Inc., Indianapolis, 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.

