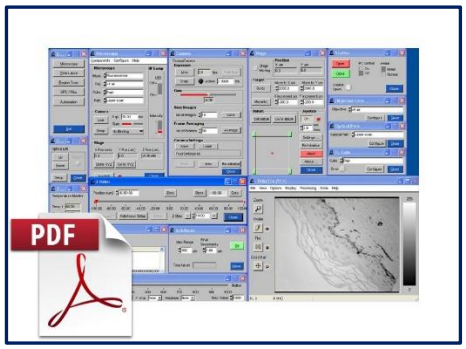
IT a anatomie firmy

(DAX, Data Analysis Expressions)

(pracovní dokument)

MBI tým

VŠE Praha, 2026

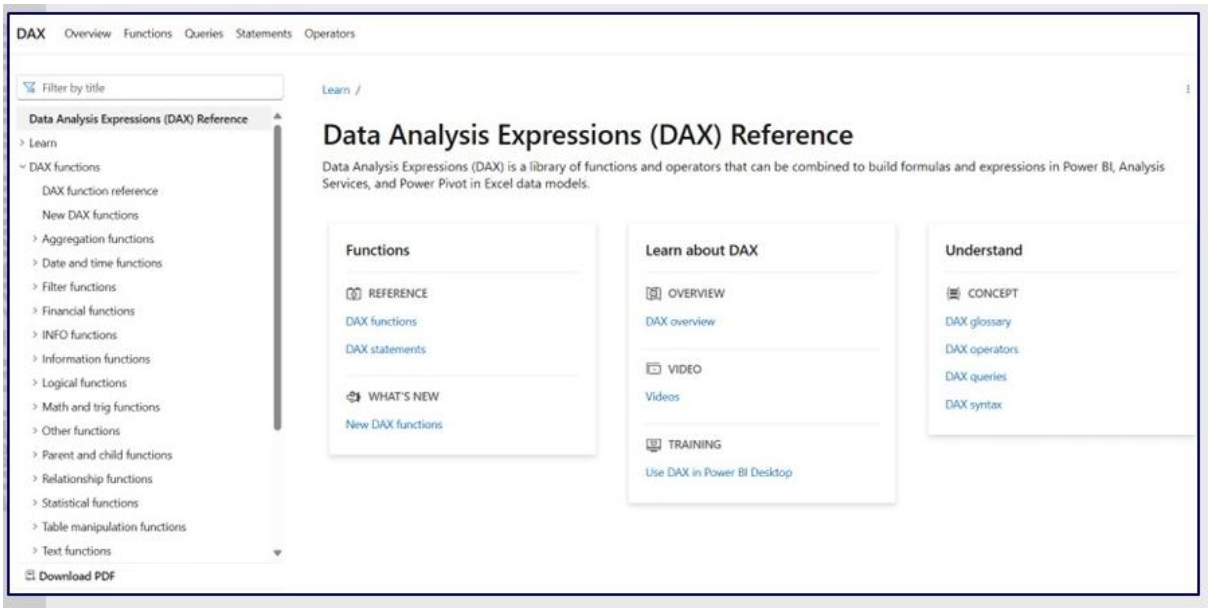


!!

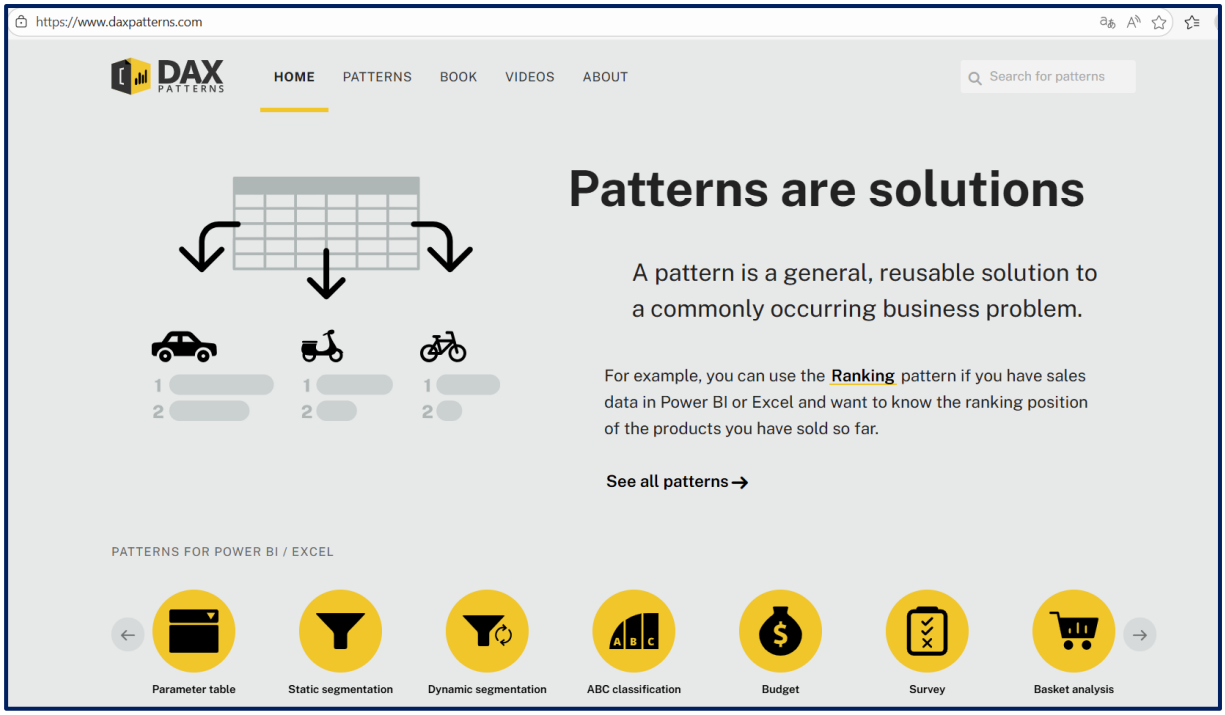
Poznámka: Pro velkou část příkladů a schémat v této publikaci ***byly využity školicí materiály společnosti Seyfor*** s laskavým svolením autorů těchto materiálů.

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***.

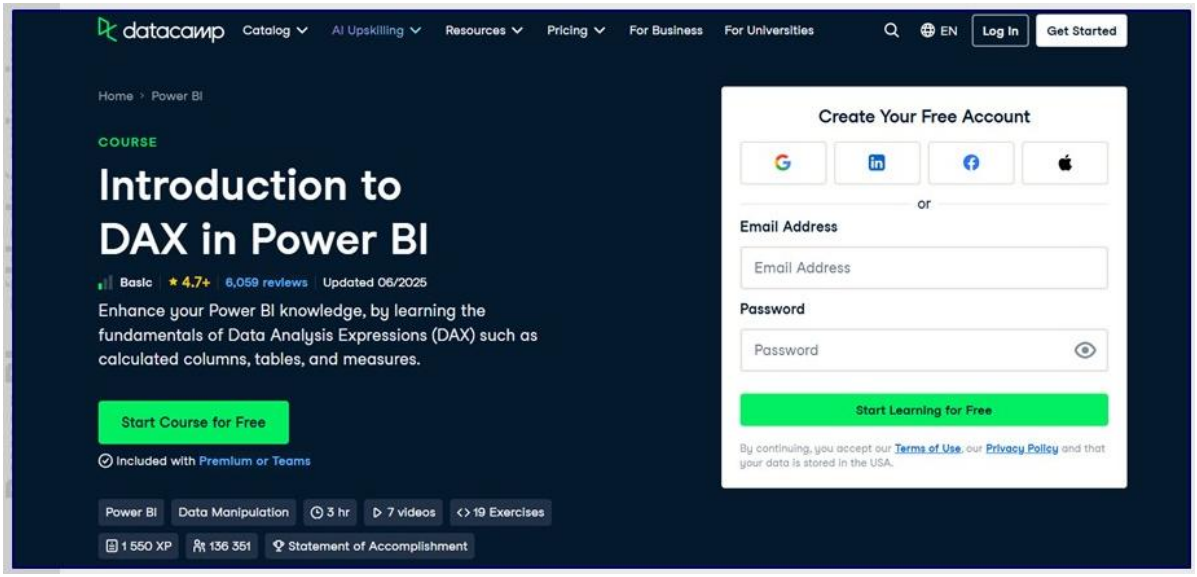
Jazyk DAX, přehled: [https://learn.microsoft.com/en-us/dax/]



FERRARI, A., RUSSO, M.: [DAX Patterns]



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



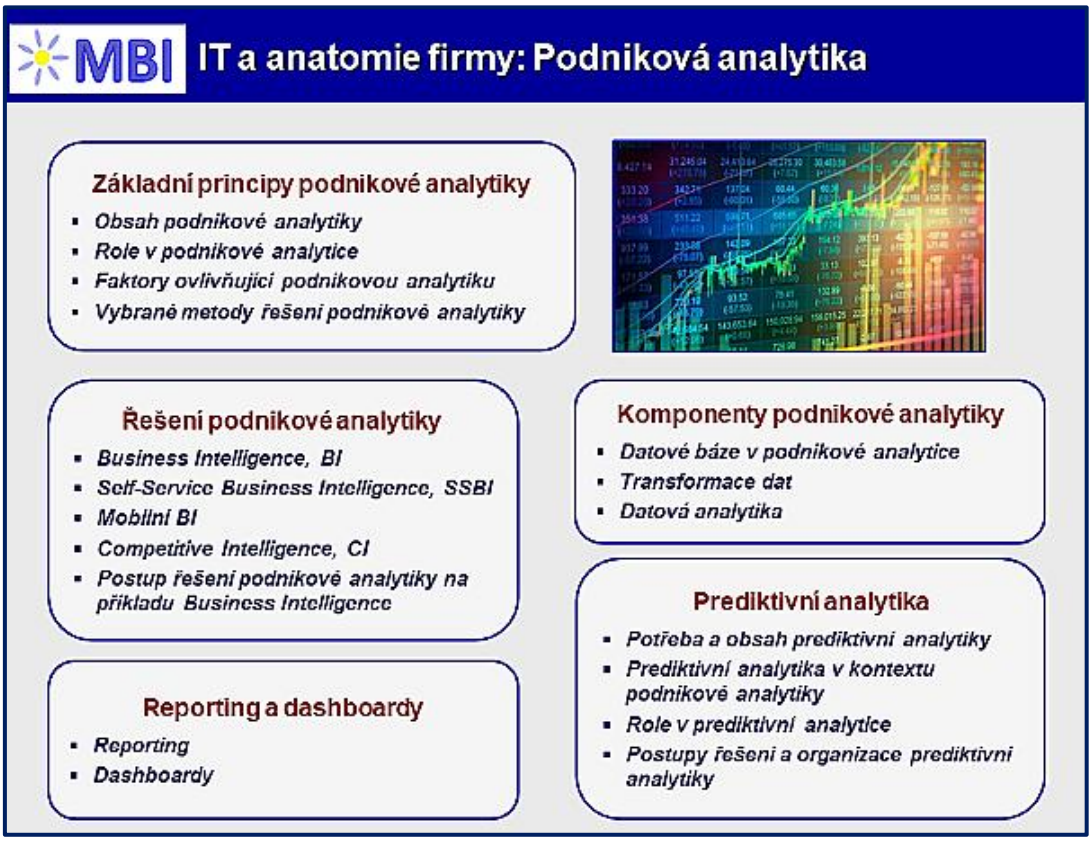
Obsah

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 dokumentů*** uvedené řady, které s nimi souvisejí***.***

Podniková analytika

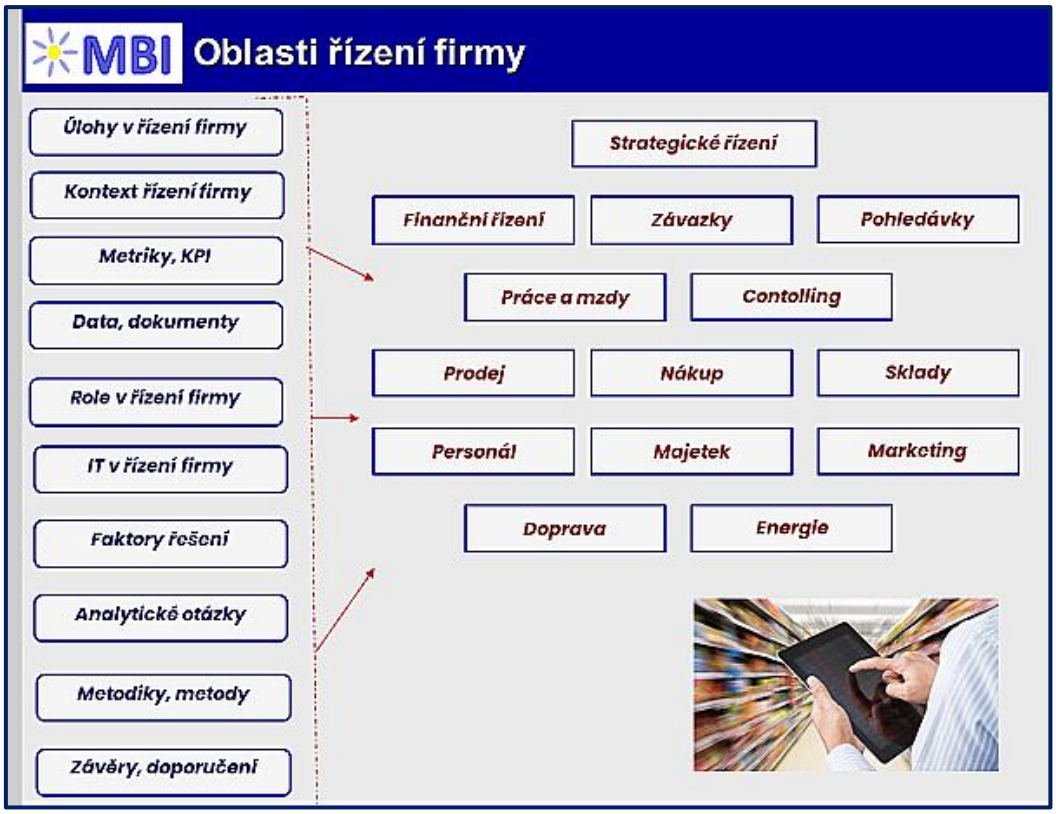
Publikace „*Podniková analytika*“ je ***základním dokumentem,*** který se snaží poskytnout ***celkový***, byť relativně stručný, ***přehled*** o principech, postupech, produktech, problémech i řešeních podnikové analytiky v praxi. Zahrnuje otázky jak „***deskriptivní analytiky***“ postavené většinou 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

Dokument 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.

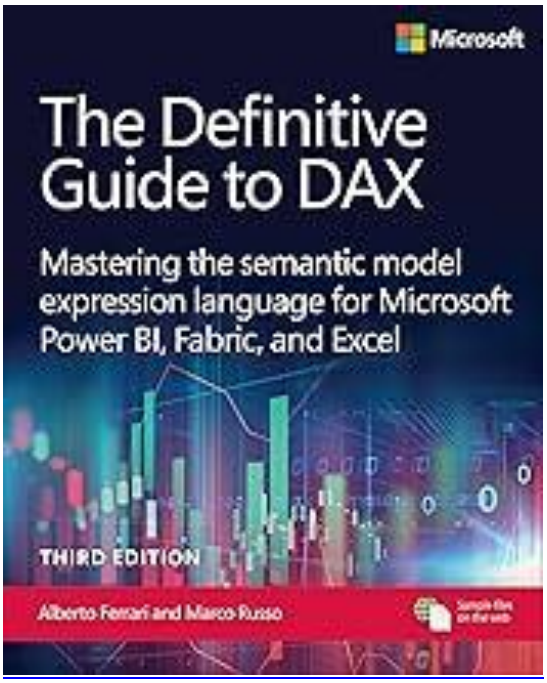


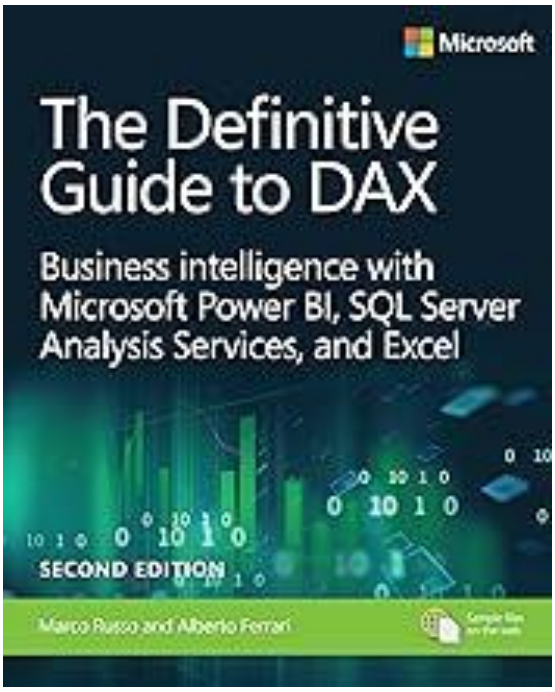
Oblasti a komponenty řízení, struktura textu

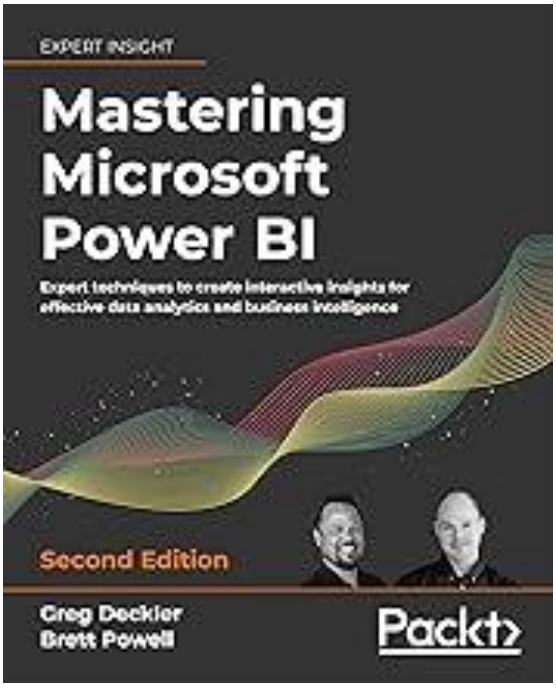
Hlavní zdroje dokumentu

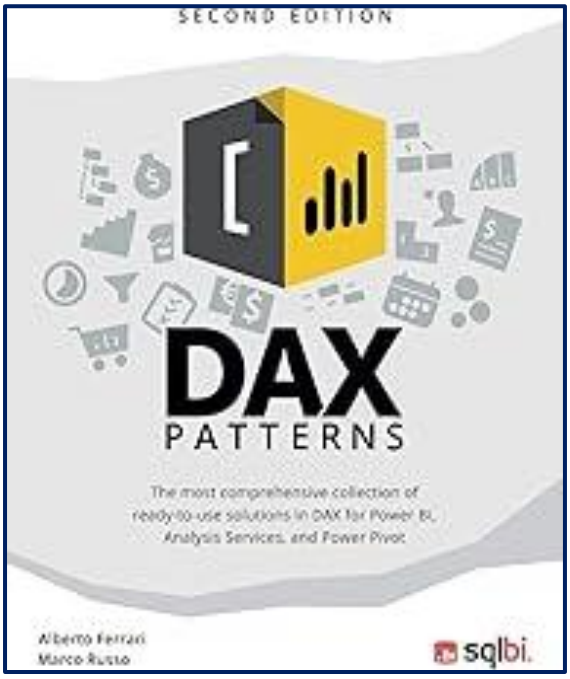
Přehled literatury je jako obvykle v závěru dokumentu. Na tomto místě uvádíme hlavní zdroje pro přípravu a aktualizaci dokumentu:

- FERRARI, A., RUSSO, M.: The Definitive Guide to DAX: Mastering the semantic model expression language for Microsoft Power BI, Fabric, and Excel (Business Skills). Pearsons Education, Inc. 2026. ISBN 978-0-13-824472-9
- FERRARI, A., RUSSO, M.: The Definitive Guide to DAX. Pearsons Education, Inc. 2020. ISBN 13-978-1-5093-0697-8
- FERRARI, A., RUSSO, M.: DAX Patterns. SQLBI, Corp., 2020. ISBN 978-1-7353652-0-6.
- DECKLER, G., POWELL, B.: Mastering Microsoft Power BI. Packt, Birmingham, 2022. ISBN: 978-1-80181-148-4.
- SEAMARK, P.: Beginning DAX with Power BI. Apress, 2018. ISBN 978-1-4842-3476-1.









Data pro příklady v dalším textu

Na tomto místě uvádíme příklady dat faktové tabulky a několika dimenzionálních tabulek, které budou využity dále v některých příkladech demonstrujících funkce a možnosti jazyka DAX. Data jsou, na rozdíl od praxe velmi zjednodušená tak, aby se s jejich pomocí uživatel v příkladech rychle orientoval bez zbytečných detailů. Do tohoto základu jsme zařadili:

- Faktovou tabulku FTQ_Prodej.
- Dimenzionální tabulky ve formátu STAR, a to:
  - času DI_Cas,
  - regionů DI_Regiony,

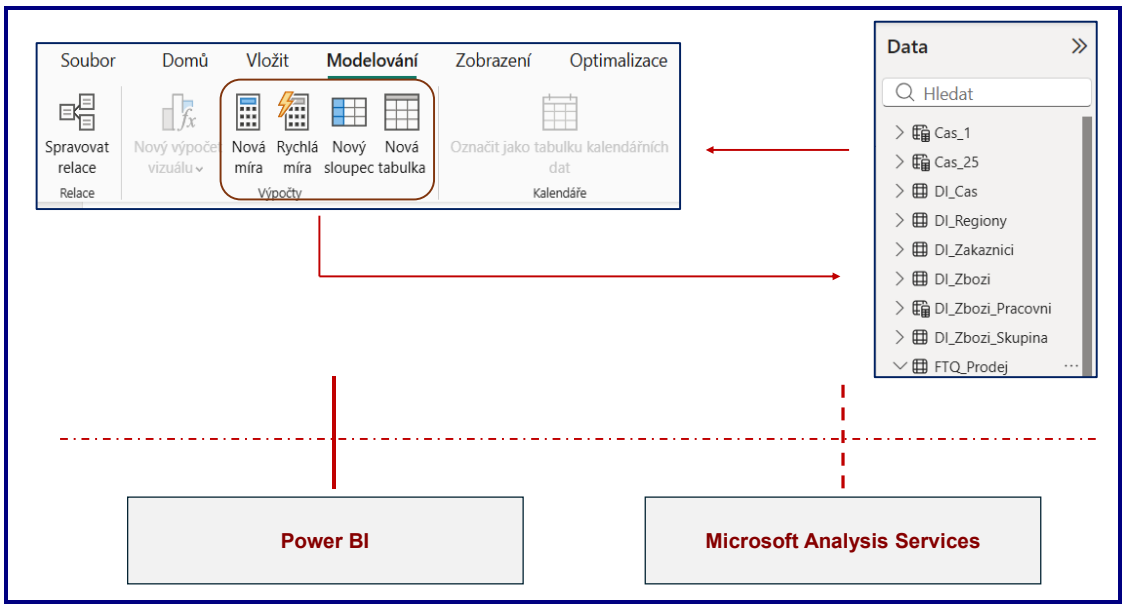
- zákazníků DI_Zakaznici,
- zboží DI_Zbozi.

Detailní přehled struktur data a vlastní data obsahuje Příloha 1: Data pro dokumentační příklady

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 vytvořený již v roce 2010. Je to efektivní jazyk ***pro práci s multidimenzionálně organizovanými a uloženými daty***, využívající sémantické modely. Kromě prostředků pro vytváření analytických aplikací umožňuje uživateli lépe a detailněji pochopit některé důležité principy pro práci v multidimenzionálním prostředí, tj. s využitím ***v prostředí Power BI, v Microsoft Analysis Services*** a v různých nástrojích, které jsou vesměs založeny interním databázovém jádru Tabular. DAX čerpá obvykle ze zdrojů, resp. tabulek datasetu vykonává na nich požadované operace a vrací je to datasetu v podobě nových měr, resp. rychlých měr, nových sloupců v tabulkách nebo celých nových tabulek. Tento základní princip dokumentuje Obrázek 1-1:



Obrázek 1-1: Výchozí princip DAX

V roce 2015 se DAX stal součástí systému ***Power BI*** a od roku 2023 tvoří jádro platformy ***Microsoft Fabric***, kde plní klíčovou roli v řešení sémantických modelů. Existuje rozdíl mezi datovým a sémantickým modelem. Datový model vyjadřuje fyzické tabulky a jejich vazby, sémantický model popisuje logické entity a jejich vazby. V Power BI představuje sémantický model datový model doplněný o míry a kalkulace zjednodušující následně komplexní dotazy uživatele. Sémantický model obsahuje i mnoho metadat, která popisují, jak prezentovat data v hierarchiích, jak třídit data ve sloupcích apod. (Ferrari, Russo, 2026).

DAX je relativně jednoduchý jazyk odlišný od většiny ostatních jazyků. Na druhou stranu ale využívá některé nové koncepty, jako kontext vyhodnocování (*evaluation context*), iterace a změna kontextu (*context transition*). Je tedy dobré uvést, že zejména 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. Primárním cílem je zde ***definovat míry*** (*measures*) a proto se DAX také označuje jako „***measure-driving language“***. Míry respektují filtry s ohledem na požadavky reportu a musí tak poskytovat relevantní výsledky nad takto definovanými podmnožinami dat, tedy report ovlivňuje chování míry. Pro celkové pochopení jazyka lze doporučit portál na adrese https://dax.guide .

DAX 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.

DAX může být charakterizován základními použitými koncepty, které budou předmětem dalších částí dokumentu, a to:

- kontextem vyhodnocování měr a kalkulací (*evaluation context*) a promítání do nich filtrů,
- iterátory (*iterators*),
- změnami kontextu (*context transition*),
- rozšířenými tabulkami a propojením dat (*expanded tables and data lineage*).

Funkce a jejich kombinace dokumentuje určité konstrukce a formy pokročilého řízení firmy, jako jsou např.:

- vytváření kalkulací a výsledných metrik v multidimenzionálním prostředí dat,
- operace a kalkulace na hierarchických strukturách objektů řízení a odpovídajících datech,
- podpora řízení a monitorování vývoje firmy na bázi tzv. „time intelligence“,
- řešení operací s určeným omezeným nebo komplexním kontextem řešení.

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. Výpočetní předpis dokumentuje Příklad 1-1:

*= 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.

Příklad 1-1: Skutečný prodej v kusech x cena za kus

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´*.

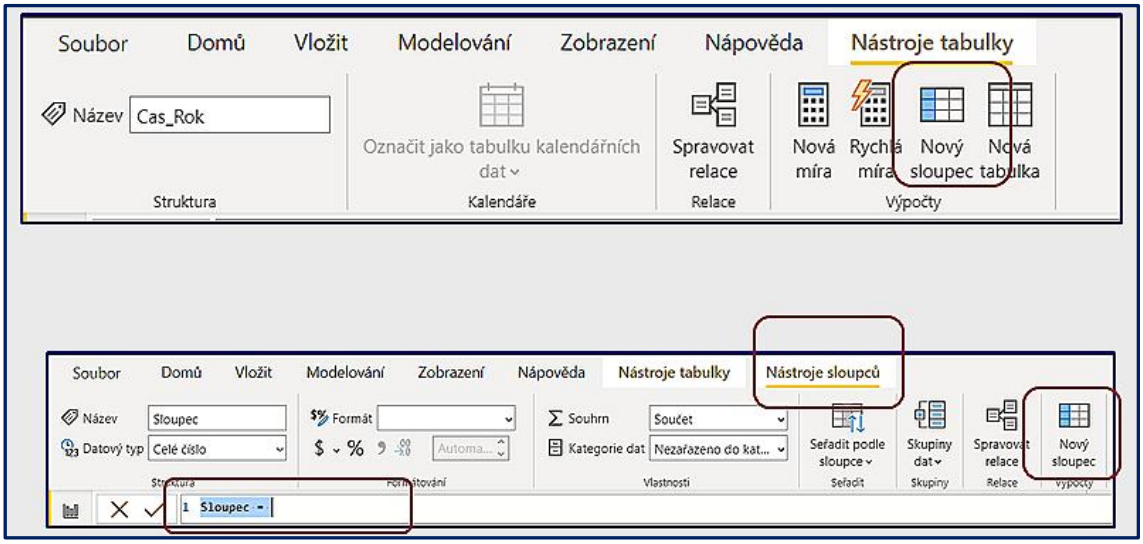
V rámci DAX se uplatňují podle daných pravidel (viz další kapitoly) tyto typy výpočtů, resp. kalkulací:

- ***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 1-2 ukazuje založení příkazu DAX pro přidání nového kalkulovaného sloupce.



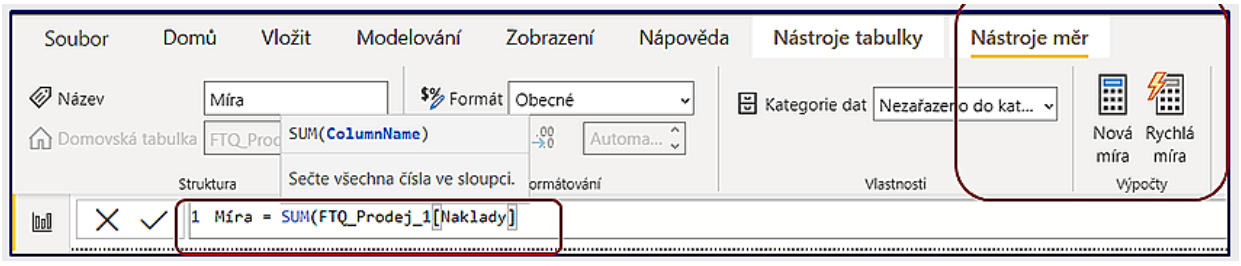
Obrázek 1-2: DAX výraz pro jednoduchý výpočet kalkulovaného sloupce

1.3 Vytvoření kalkulované míry

Míra se počítá na agregační úrovni dat, tj. ***za tabulku*** nebo její podmnožinu. Práci s mírou ukazuje Příklad 1-2, tj. celkový objem prodeje v Kč – celkové náklady na prodané zboží. K vytvoření míry se využije funkce *„Míry“.* a (Obrázek 1-3):

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

Příklad 1-2: Celková marže z prodeje zboží



Obrázek 1-3: Vytvoření nové kalkulované míry

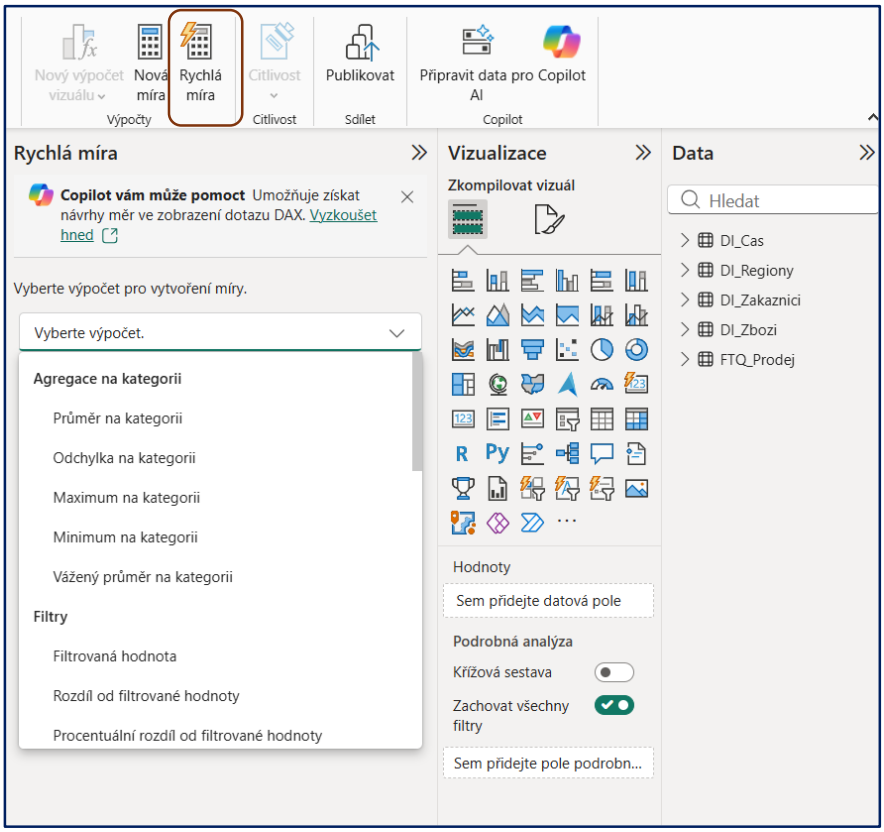
V každém případě je třeba, aby ***byla vybrána dimenzionální nebo faktová tabulka***, do které se příslušný sloupec nebo míra zařadí. ***V případě míry*** je zřejmé, že ***nemusí být v podstatě zařazena do žádné tabulky***, nebo by mohla vzniknout ***tabulka obsahující pouze vypočtené míry***, ale pro 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 míry musí být v rámci celého sémantického modelu ***jedinečné***. Na rozdíl od sloupce tabulky však ***identifikace nesmí obsahovat jméno tabulky***.

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 1-4: 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.

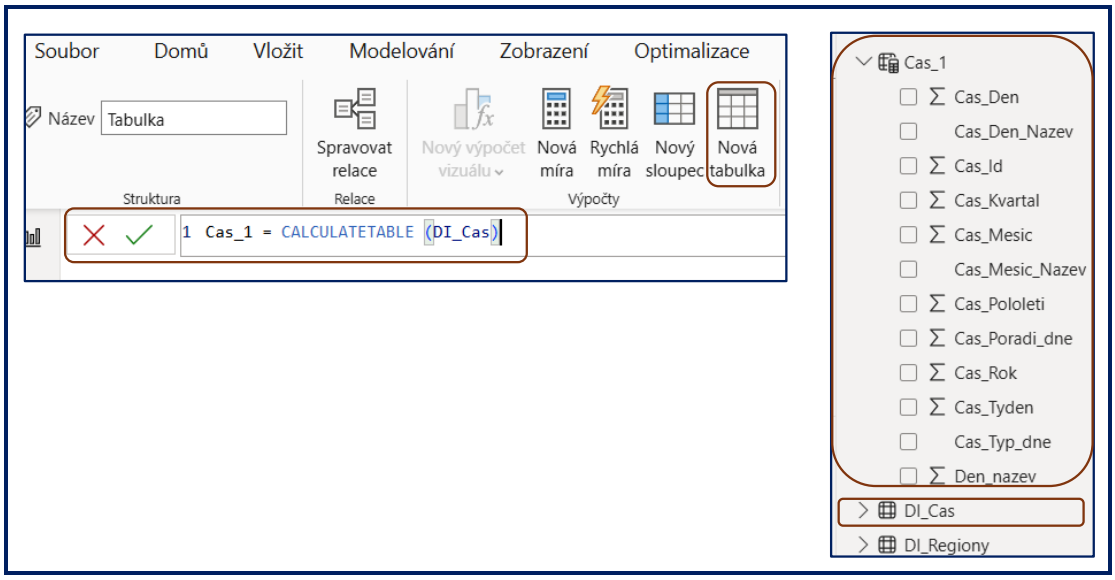
1.5 Vytvoření kalkulované tabulky

Kalkulované tabulky lze vytvořit příslušnou funkcí DAX, která na výstupu vrací tabulku nebo prostým, zkopírováním jedné tabulky do nové, jak ukazuje Příklad 1-3:

```
Cas_1 = CALCULATETABLE (DI_Cas)
```

Příklad 1-3: Vytvoření kalkulované tabulky

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

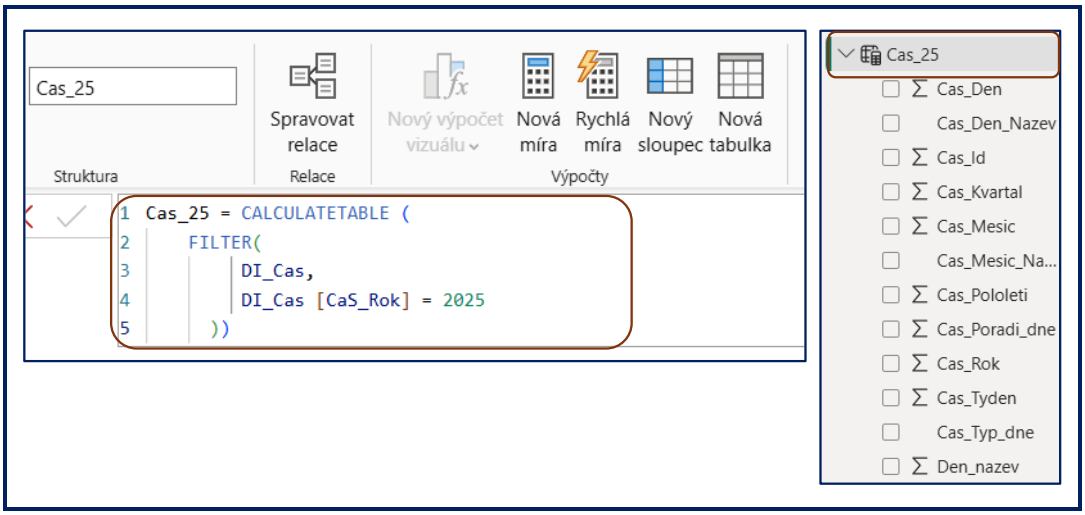


Obrázek 1-5: Vytvoření kopie tabulky DI_Cas

Kalkulované tabulky mohou představovat data vybraná na základě filtrů, což pak urychlí a zpřehlední další výpočty, jak ukazuje Příklad 1-4 a Obrázek 1-6

```
Cas_25 = CALCULATETABLE (
    FILTER (
        DI_Cas,
        DI_Cas [Cas_Rok] = 2025
            )
)
```

Příklad 1-4: Vytvoření tabulky s využitím funkce FILTER s daty pro rok 2025



Obrázek 1-6: Kalkulovaná tabulka s využitím filtru

1.6 Pracovní závěry

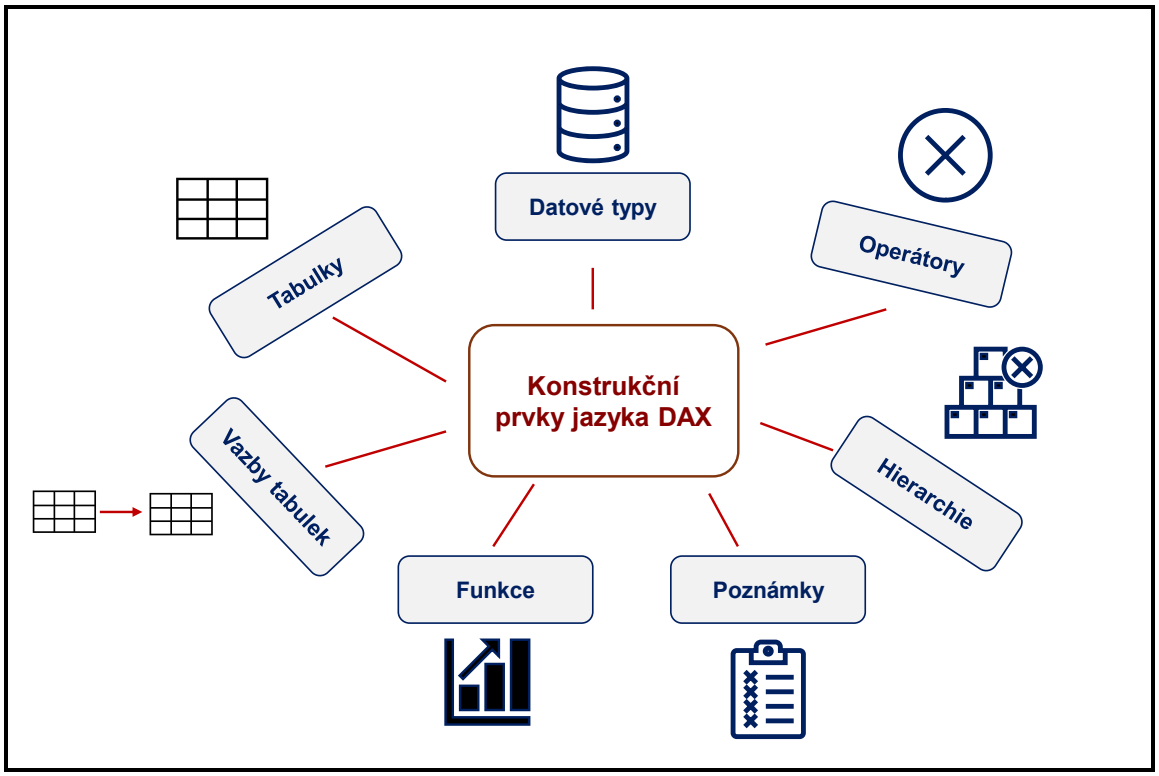
Z kapitoly vyplývají následující ***závěry***:

- V rámci DAXu ***typy 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.***
- Příklad výpočetního předpisu je následující:
  - *= FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks].*
- DAX nabízí funkcionalitu ***pro rychlé vytváření nejobvykleji používaných měr,*** a to volbou *Rychlá míra.*

2. Konstrukce jazyka DAX

Účelem kapitoly věnované konstrukci jazyka DAX je provést ***celkovou rekapitulaci*** a vymezení definovaných datových typů, operátorů a dalších.

Jako každý jazyk nebo složitý systém se skládá z konstrukčních prvků s jejichž pomocí jsou vytvářeny výpočetní nebo operační předpisy. Přehled hlavních těchto prvků dokumentuje Obrázek 2-1¨a další text této kapitoly.



Obrázek 2-1: Konstrukční prvky jazyka DAX

2.1 Datové typy

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

- *Integer* – celé číslo, 64-bitová hodnota.
- *Decimal* – desetinné číslo, s dvojitou přesností a pohyblivou řádovou čárkou.
- *Currency* – měna, ukládá data s pevnou řádovou čárkou, 4 desetinná čísla.
- *DateTime* – datum a čas, interně s pohyblivou řádovou čárkou, celá část odpovídá počtu dní počínaje 30.12.1899 a desetinná část představuje hodiny, minuty, sekundy konvertované na destinnou část dne. Vedle toho existují dva další datové typy *Date* a *Time*. Pokud je třeba použít část „den“, je třeba využít funkce *TRUNC()* pro oddělení desetinné čárky.
- *Boolean* –True / False, kde True odpovídá hodnotě 1 a False hodnotě 0.
- *String* – textový řetězec v Unicode, každý znak je kódován v 16 bitech. Porovná řetězců nerozlišuje velká a má písmena, není „*case sensitive*“.
- *Variant* – pro případy, kdy výraze může vracet různé datové typy v závislosti na definovaných podmínkách.

- *Binary* – binární hodnota pro obrázky a nestrukturovaná data.

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. + - ,
5. & (konkatenace),
6. = < > <= >= <>.

Logické operátory:

- && – AND, logický součin,
- || – OR, logický součet,
- IN – obsahuje prvek ze seznamu,
- ! (NOT) – Booleovská negace.

2.3 Tabulky, řádky, 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. Každý sloupec má určitý datový typ. Řádky v tabulce se chápou jako záznamy (*records*).

Pokud se v DAXu použijí anonymní tabulky se zadanými hodnotami, uzavírají se do složených závorek, Příklad 2-1:

{ (“Philips”), (“Samsung”), (“Siemens”) }

Příklad 2-1: Anonymní tabulka s hodnotami

Uvedená konstrukce je často spojována s operátorem IN, který kontroluje přítomnost příslušné řádky v tabulce Příklad 2-2:

```
DI_Zbozi [ Zbo_Znacka ] IN { ( “Philips” ),  ( “Samsung” ), ( “Siemens”) }
```

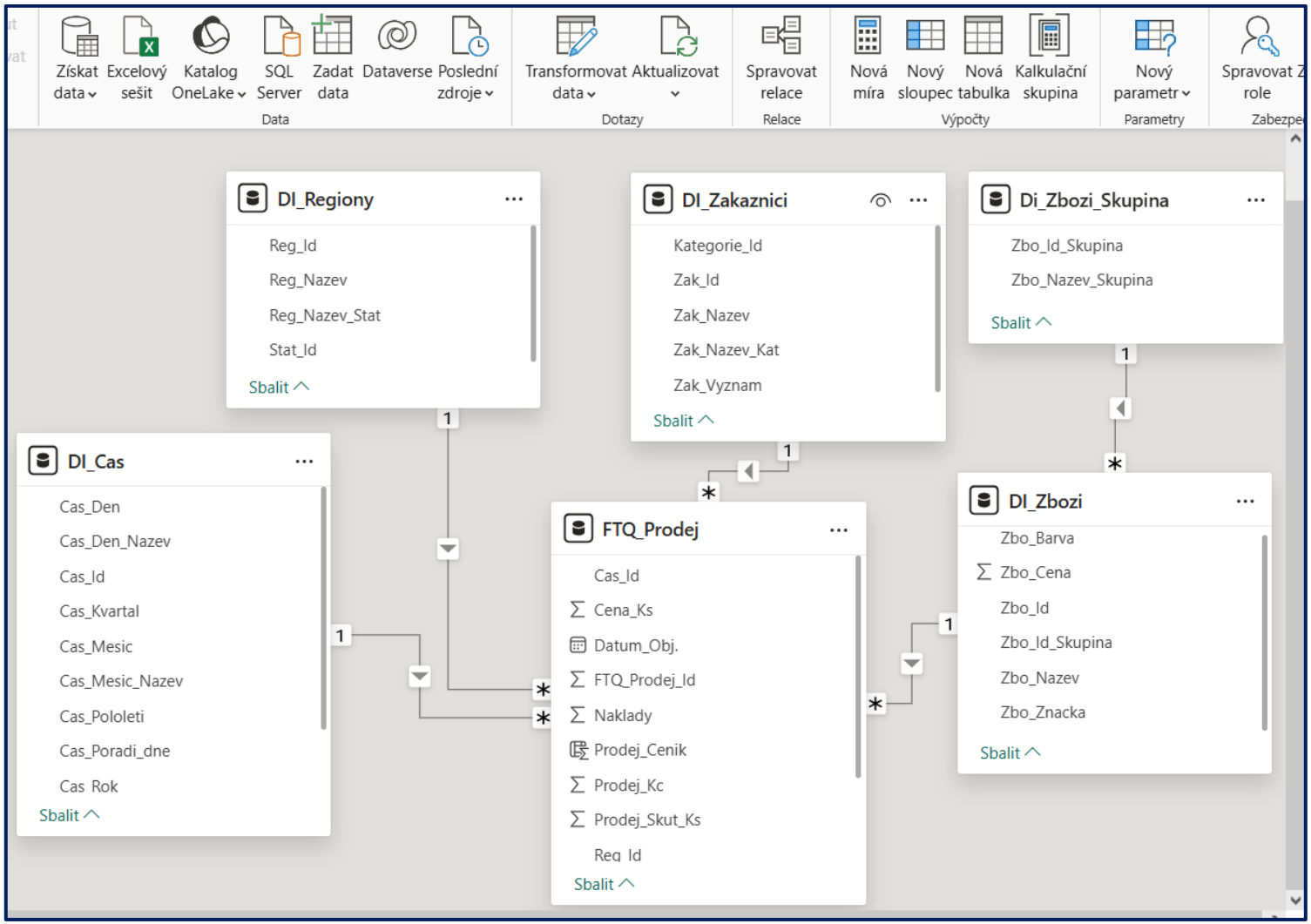
Příklad 2-2: Využití operátoru IN v anonymní tabulce

2.4 Hierarchie

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

2.5 Vazby tabulek

Vazby propojují tabulky, s tím, že se připouští ***pouze vazby 1:1 nebo 1:M***, ***resp. M:1***, ale pouze s využitím pouze jednoho sloupce v každé tabulce. Tabulky s vazbami přestavují sémantický model, jak ukazuje Obrázek 2-2:



Obrázek 2-2: Sémantický model v Power BI

K obrázku uvedeme několik poznámek:

- ***Sloupce na straně „1“*** musí obsahovat ***unikátní hodnoty*** (nesmí se opakovat) a nesmí obsahovat mezery.
- ***Sloupce na straně „M“*** mohou obsahovat opakované hodnoty a velmi často to tak je.
- Vazby mohou vytvářet ***řetězce tabulek***, např. *DI_Zbozi – DI_Zbozi_Skupina,* viz Obrázek 2-2 vpravo.
- V každé vazbě se šipkami identifikuje i ***orientace filtrování***, a to ve dvou variantách: ***jednosměrné a obousměrné*** (*cross-filter direction*).
- ***Jednosměrné*** filtrování je zobrazeno ***jednoduchou šipkou***, téměř u všech vazeb. Např. ve vazbě *DI_Zakaznici – FTQ_Prodej* ve vazbě 1 : M jde o jednosměrný filtr a hodnoty prodeje jsou filtrovány přes jednotlivé zákazníky apod. V opačném případě M:1 se filtr neuplatní.
- ***Obousměrné*** filtrování (*bidirectional*) je zobrazeno dvojitou šipkou, jako je tomu u vazby *DI_Cas – FTQ_Prodej.* V tomto případě se filtr uplatní i pro vazbu M:1. Toto filtrování se však v praxi příliš nedoporučuje.
- Pro otázky datového modelování a řešení vazeb lze doporučit tutoriály na adrese https://www.sqlbi.com .

2.6 Funkce jazyka DAX

Jazyk DAX poskytuje široké spektrum funkcí. ***Účelem*** této podkapitoly je pouze vytvořit základní přehled a vymezení skupin i dílčích funkcí pro jejich specifikace a využití v dalších kapitolách. Do podkapitoly jsou zařazeny vybrané funkce rozdělené ***do následujících skupin*** (Ferrari, Russo, 2026):

- Agregační funkce.
- Logické funkce.
- Informační funkce.
- Matematické funkce.
- Textové funkce.
- Konverzní funkce.
- Datové a časové funkce.
- Relační funkce.
- Komplexní funkce
- Funkce Time Intelligence

Přehled a základní vymezení funkcí obsahuje Příloha 2: Přehled funkcí DAX podle skupin a současně lze doporučit portál na adrese https://dax.guide .

2.7 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. Tyto poznámky však na rozdíl od jednořádkových poznámek nejsou používané často.

Nedoporučuje se uvádět poznámky u kalkulovaných sloupců, měr a kalkulovaných tabulek.

2.8 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- V DAXu platí, že výsledný typ ve výrazu vychází z datových typů jednotlivých částí

výrazu.

- Vazby propojují tabulky, s tím, že se připouští ***pouze vazby 1:1 nebo 1:M***, ***resp. M:1***, ale pouze s využitím pouze jednoho sloupce v každé tabulce.
- Hierarchie jsou ***seskupení jednoho nebo více sloupců***, které vymezují jednotlivé úrovně hierarchie.
- ***Nejpodstatnější*** a nejčastěji využívané se v praxi ukazují ***agregační a logické*** funkce.
- Pro ***práci s texty*** je k dispozici několik funkcí pro ***výběry dat z textových řetězců*** a další textové operace.
- ***Informační funkce*** jsou většinou zaměřeny na identifikaci chyb v konstrukcích zápisů v jazyce DAX.
- Významné místo ve využití mají ***relační funkce*** umožňující zpracovávat data z různých vzájemné provázaných tabulek.
- ***Poznámky*** dokumentují operace v DAXu a mohou být zařazovány do výpočtů v DAXu na základě znaků: //, --, /* */.
- Účelem proměnných v DAX je ***výpočty udělat jednodušší a některých případech i rychlejší*** a využívají se k ukládání výsledků z výrazů DAX.

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. Proměnné nemohou být definovány jako globální proměnné, tedy na úrovni celého kódu modelu v DAXu.

Využívají se také zejména v případech, kde se výrazy opakují a zvyšují současně čitelnost kódu. (Seamark, 2018).

3.1 Podstata proměnných

Syntaxe:

VAR název proměnné = výraz

RETURN výraz

K principu proměnných ***několik poznámek***:

- Může být definována jedna nebo více proměnných, ale celý jejich blok musí být uzavřen klíčovým slovem *RETURN*.
- *RETURN* vrací výsledek a může rovněž obsahovat výraz, Příklad 3-1.
- Název proměnné nemůže obsahovat mezery nebo být uzavřen v apostrofech nebo závorkách.
- Proměnná je lokální v rámci výrazu, kde je definována. Globální definice proměnných neexistuje.
- Pokud má být proměnná vypočítána, je počítána pouze jednou. Při opakovaném použití ve výrazu se již nepočítá, využívá předem získané hodnoty.

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

Příklad 3-1: Naplnění proměnné

Proměnné mohou obsahovat nejen numerické hodnoty, ale i texty, Příklad 3-2:

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

Příklad 3-2: Proměnné VAR s texty

Proměnné lze vnořovat, ale musí být zajištěno správné nastavení *RETURN*, Příklad 3-3:

```
´Celkovy prodej´ =
VAR Prodej1a = 200
VAR Prodej1b =
        VAR Prodej2a = Prodej1a
```

```
        VAR Prodej2b = Prodej2a + 150
        RETURN Prodej2b
RETURN Prodej1b
```

Příklad 3-3: Vnořování proměnných při výpočtu míry

```
VAR ´Objem prodeje´= SUM (FTQ_Prodej [Prodej_Kc])
VAR ´Objem nakladu´ = SUM (FTQ_Prodej [Naklady])
VAR ´Hruba marze´ = ´Objem prodeje´ - ´Objem nakladu´
RETURN
    ´Hruba marze´ / ´Objem prodeje´
```

Příklad 3-4: Využití proměnných s mírami

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

3.2 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Účelem proměnných v DAX je ***výpočty udělat jednodušší a některých případech i rychlejší*** a využívají se k ukládání výsledků z výrazů DAX.
- V rámci kódu může být definována ***jedna nebo více proměnných***, ale takový blok musí být uzavřen klíčovým slovem ***RETURN***.
- Proměnná je ***lokální*** v rámci výrazu, kde je definována

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 kalkulovaných měr***.

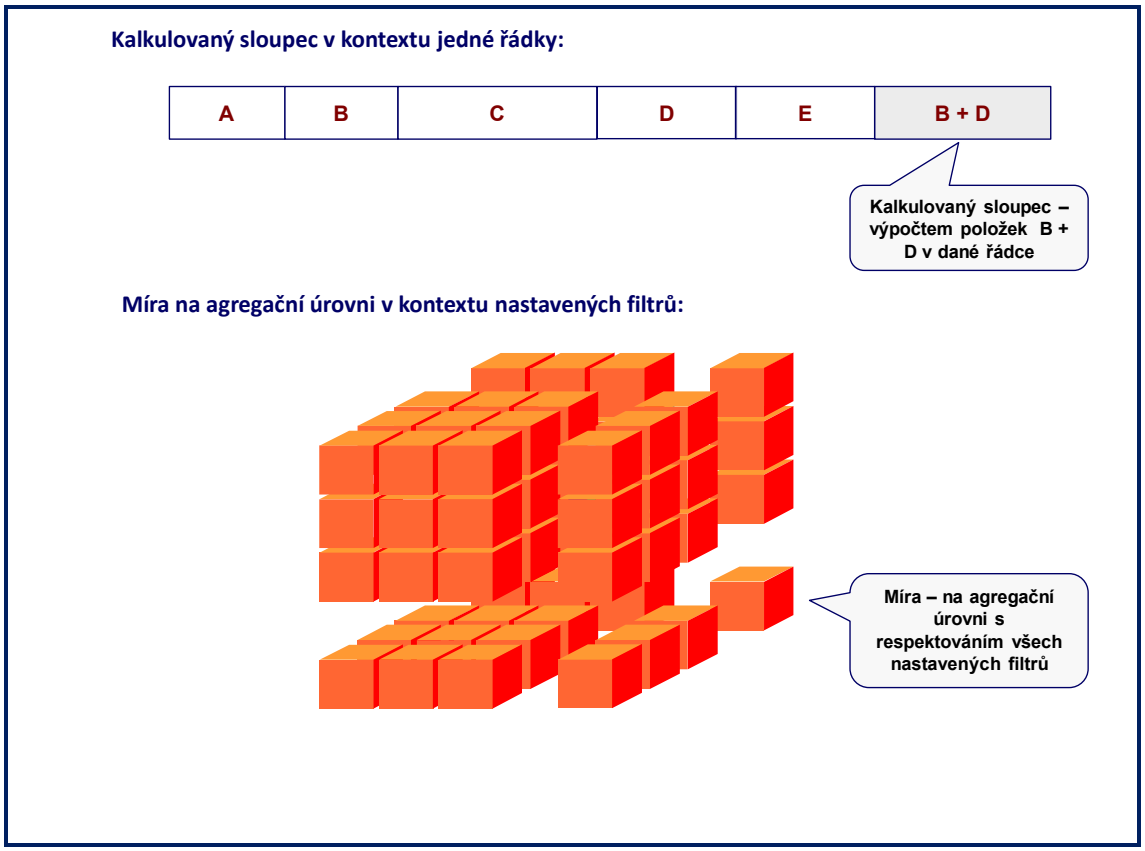
DAX a na jeho základě definované výpočetní předpisy musí ve svém principu respektovat uspořádání a prostředí databáze Power BI, nebo SSAS Tabular a jejich uložení dat. To znamená, musí respektovat ***kontext****,* v němž se definovaný předpis bude vyhodnocovat (*evaluation context*).

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, většinou je základem kalkulovaných měr.
- ***řádkový kontext* (*row context*)** mající základ ve výpočtech a dalších operacích v rámci 1 řádky a obvykle je základem vytváření kalkulovaných sloupců.

Kalkulované sloupce a míry, oba druhy kontextu a jejich detailní charakteristiky budou předmětem dalšího textu. Rozdíl mezi nimi dokumentuje Obrázek 4-1.



Obrázek 4-1: Rozdíl v kontextu kalkulovaného sloupce a míry

Filtr kontext tak filtruje data, zatímco řádkový kontext tabulku skenuje a na řádcích vykonává operace, resp. výpočty. ***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.

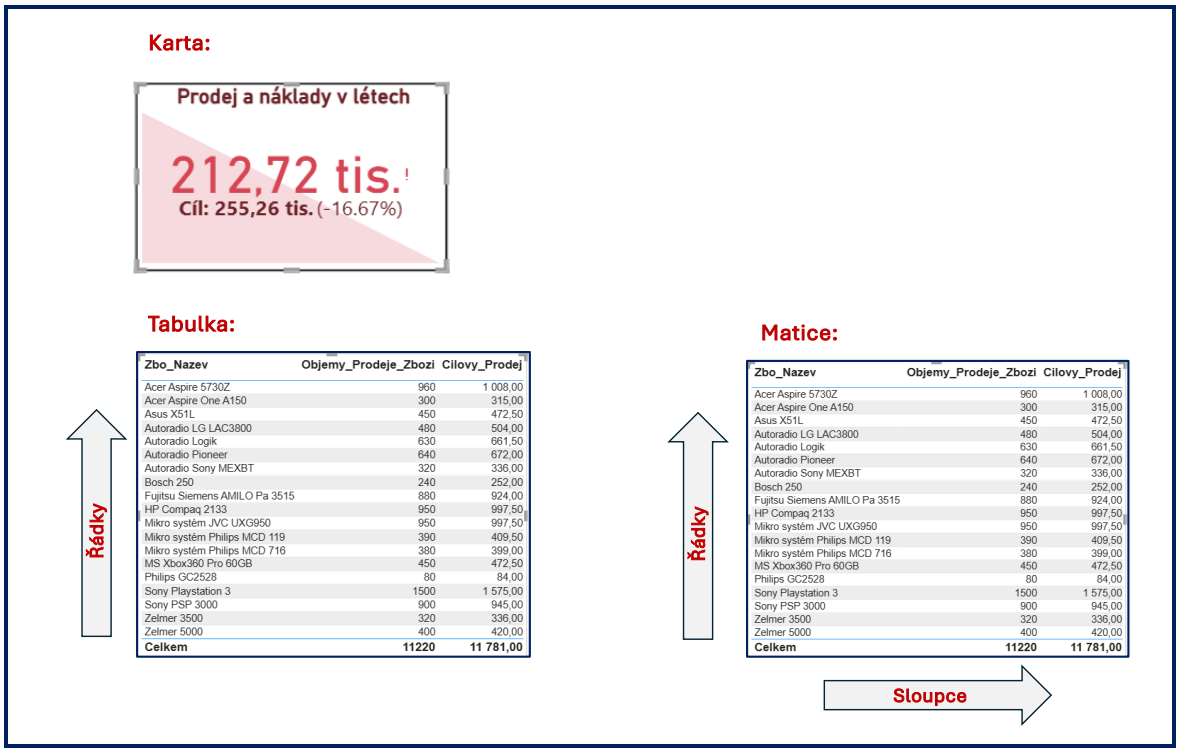
(Ferrari, Russo, 2026) uvádí, že skutečná síla DAX není ani tak ve využití jednotlivých funkcí, ale v kvalifikovaném a efektivním využití jak filtr, tak řádkového kontextu.

4.1 Filtr kontext

Jako podstatný vstup pro pojetí filtr kontextu je vymezení os a souřadnic bodů na vizuálu Power BI.

4.1.1 Filtr kontext na bázi souřadnic prvku

Vizuály Power BI mohou mít žádnou, jednu nebo dvě osy, Obrázek 4-2:



Zbo_Nazev	Objemy_Prodeje_Zbozi	Cilovy_Prodej
Acer Aspire 5730Z	960	1 008,00
Acer Aspire One A150	300	315,00
Asus X51L	450	472,50
Autoradio LG LAC3800	480	504,00
Autoradio Logik	630	661,50
Autoradio Pioneer	640	672,00
Autoradio Sony MEXBT	320	336,00
Bosch 250	240	252,00
Fujitsu Siemens AMILO Pa 3515	880	924,00
HP Compaq 2133	950	997,50
Mikro systém JVC UXG950	950	997,50
Mikro systém Philips MCD 119	390	409,50
Mikro systém Philips MCD 716	380	399,00
MS Xbox360 Pro 60GB	450	472,50
Philips GC2528	80	84,00
Sony Playstation 3	1500	1 575,00
Sony PSP 3000	900	945,00
Zelmer 3500	320	336,00
Zelmer 5000	400	420,00
Celkem	**11220**	**11 781,00**

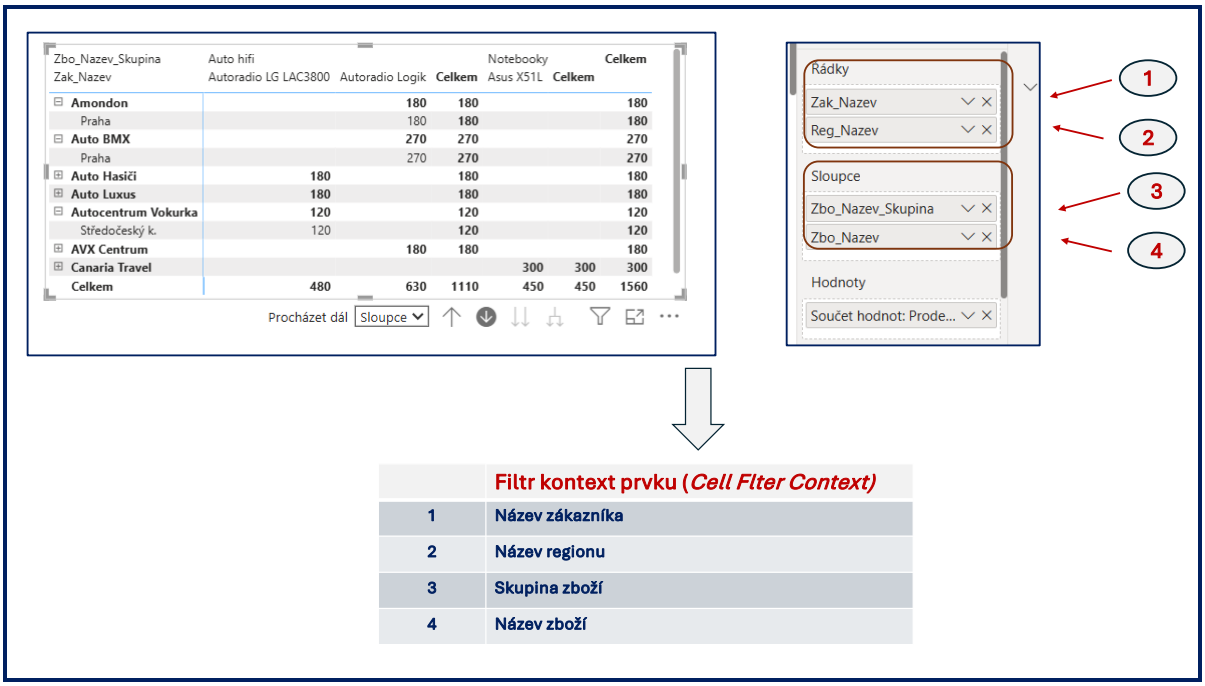
Obrázek 4-2: Osy na vizuálech Power BI

Z obrázku je patrné, že:

- karta nepoužívá žádnou osu,
- tabulka pracuje s jednou osou (řádky),
- matice má dvě osy (řádky a sloupce).

Na tomto základě je evidentní, že je možné definovat souřadnice pro každý prvek vizuálu. Pro kartu jde pouze o jeden prvek. V případě tabulky je osa jedna, založená na řádcích. Matice má osy 2, tedy řádky a sloupce.

Obrázek 4-3 dokumentuje matici v rámci Power BI a souřadnice (*coordinates*) jednotlivých prvků na bázi dvou os (řádků, sloupců). Souřadnice prvků jsou i základem pro formulování ***filtr kontextu prvků*** (*Cell Filter Context*). Ten je však pouze jednou součástí celého filtr kontextu.



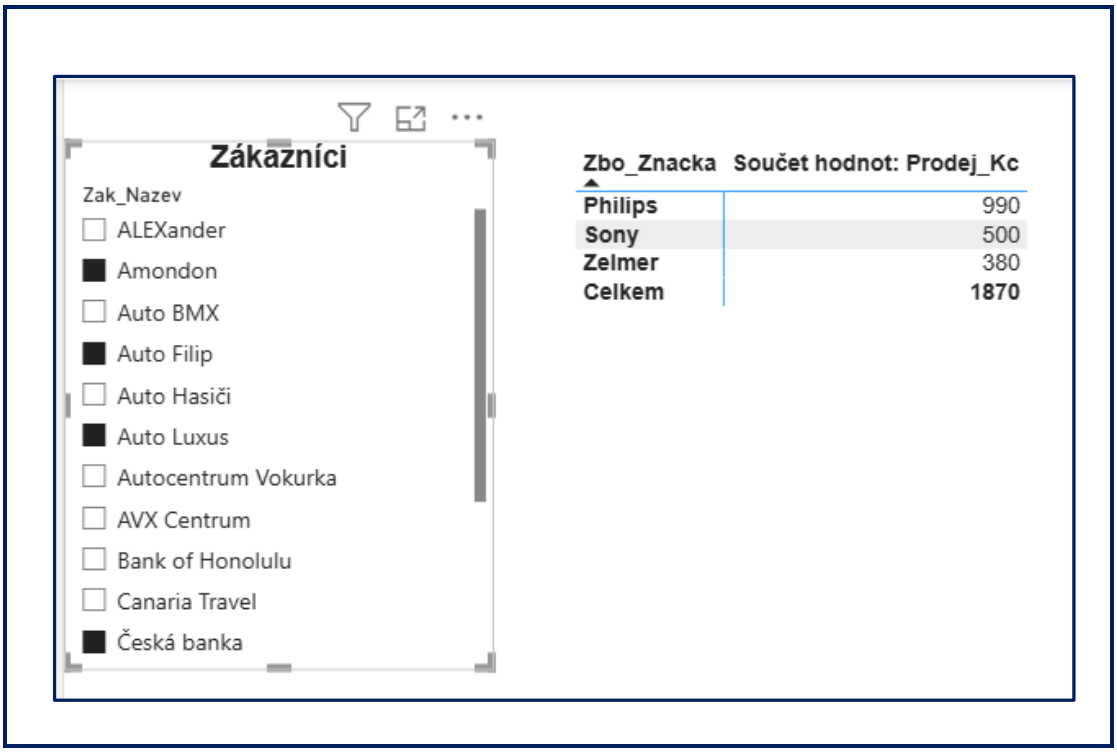
	Filtr kontext prvku (*Cell FIter Context*)
1	Název zákazníka
2	Název regionu
3	Skupina zboží
4	Název zboží

Obrázek 4-3: Souřadnice prvku v rámci matice Power BI

4.1.2 Princip filtr kontextu

Filtr kontext je ***sada sloupcových filtrů*** využívaných u každých výpočtů určujících, které řádky tabulky vstoupí do dalších výpočtů. Platí tato pravidla (Seamark, 2018, Ferrari, Russo, 2026):

- každý filtr může být založen ***na jednom nebo i více sloupcích*** současně,
- je účelné se dívat ***na každou buňku (míru)*** výstupní matice jako na ***samostatný výpočet*** s tím, že pro každou buňku je aktivní určitý, speciální filtr kontext,
- filtr kontext je ***definován*** primárně *souřadnicemi* každé buňky v matici,
- filtr kontext ***určuje výpočet*** pro každou buňku,
- filtr kontext ***ovlivní i další kritéria***, např. obsah průřezu v kombinaci se značkou zboží, jak dokumentuje Obrázek 4-4,



Obrázek 4-4: Filtr kontext určený obsahem průřezu

- specifický filtr kontext ***dílčí agregace***, k niž se váží pouze jim odpovídající souřadnice, např. na obrázku (Obrázek 4-3) pro první řádek - „Amondon“ zahrnuje filtr kontext pouze název zákazníka, název skupiny zboží a zboží, nikoli region,
- každý ***vizuál má svůj filtr kontext***, který je ovlivněn i ostatními vizuály v reportu, s nimiž je daný vizuál v interakci,
- 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***.

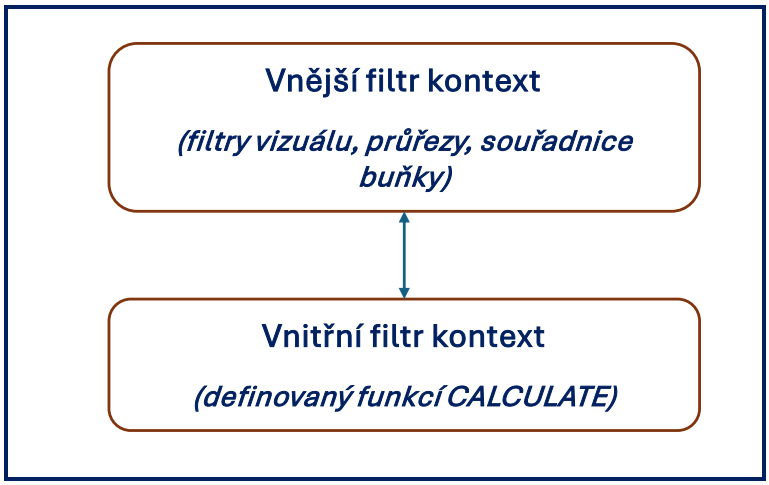
4.1.3 Filtr kontext na základě funkce CALCULATE()

Kromě výše uvedených možností nastavení filtr kontextu, existuje další na bázi využití funkce CALCULATE(). Ta umožňuje pro výpočet (v prvním parametru funkce), nastavit výběrová kritéria, která se stávají součástí filtr kontextu, podrobněji kapitola 8. Uvedenou možnost dokumentuje další příklad:

```
Prodej_Zak = CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Zakaznici [Zak_Nazev] = „Amondon“)
```

Příklad 4-1: Příklad CALCULATE s výběrem zákazníka

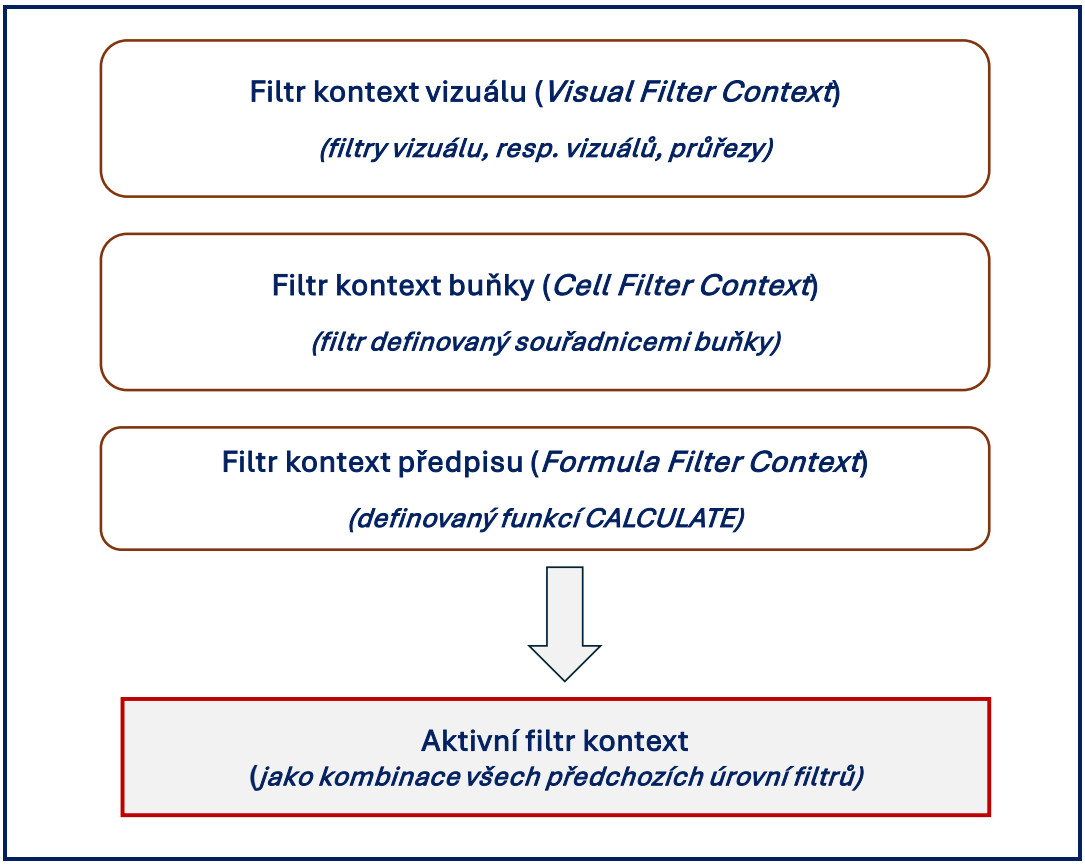
Filtr kontext na bázi funkce CALCULATE() se označuje jako vnitřní filtr kontext (*inner filter context*) na rozdíl od vnějšího filtr kontextu (*outer filter context*) definovaného ještě před provedením funkce CALCULATE(). Jejich vztah dokumentuje Obrázek 4-5.



Obrázek 4-5: Vnější a vnitřní filtr kontext

4.1.4 Výsledný aktivní filtr kontext

Výsledný aktivní filtr kontext pro buňky je dán kombinací všech tří úrovní, na kterých se filtr kontext vymezuje, jak dokumentuje Obrázek 4-6. Výsledný filtr kontext je pak ovlivněn pravidly, která jsou spojena s funkcí CALCULATE() a ke kterým se vrátíme v kapitole 8.



Obrázek 4-6: Kombinace kontext filtrů a výsledný aktivní filtr kontext

4.2 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 tak zvyšují obsazení kapacity paměti. Ale v případě režimu Direct Query nebo kompozitního modelu se hodnoty kalkulovaného sloupce počítají v průběhu provádění dotazu a paměť tak nezabírají. Na druhé straně dobu odezvy dotazu prodlužují.

4.3 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.

4.4 Pracovní závěry

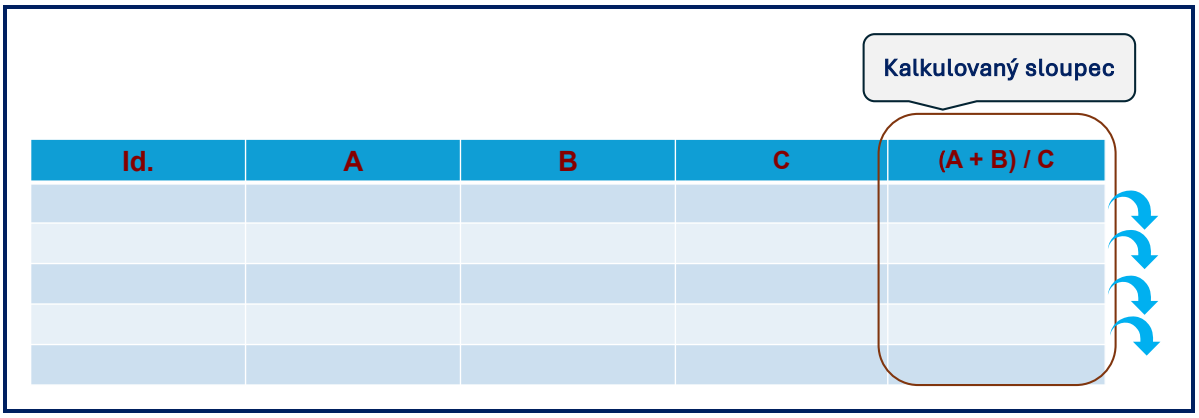
Z kapitoly vyplývají následující ***závěry***:

- Existují dva základní kontexty vyhodnocování dat: filtr kontext a řádkový kontext.
- Filtr kontext filtruje data na úrovni modelu (datasetu), řádkový kontext skenuje tabulku a zpracovává jednotlivé řádky.
- DAX vytváří řádkový kontext automaticky při realizaci kalkulovaného sloupce.
- Řádkový kontext se také aktivuje programově při užití iterátorů. Každý iterátor definuje řádkový kontext.
- Řádkové kontexty lze vnořovat do sebe, pokud se realizují na jedné tabulce.
- S řádkovým kontextem je účelné využívat proměnné (*VAR*) pro uchování hodnot ze zpracování v rámci řádkového kontextu.
- Filtr kontext se aktivuje, pokud se v řešení používají datové položky v řádcích, sloupcích průřezech a filtrech.
- Filtr kontext se může aktivovat i programově při použití funkce *CALCULATE()* (viz dále).
- Filtr kontext filtruje sémantický model s využitím vazeb mezi tabulkami a s respektováním směru filtrování. Vesměs se filtruje ve směru kardinality 1 k M. V případě, že je nastaveno obousměrné filtrování (*BOTH*), lze filtrovat i ve směru M ku 1.

5. Kalkulované sloupce (Calculated Columns)

Účelem kapitoly je vrátit se ve větším detailu ke kalkulovaným sloupcům 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, jak dokumentuje Obrázek 5-1:



Obrázek 5-1: Princip kalkulovaného sloupce

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.

5.1 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 datasetu Power BI***, tj. do aktualizovaného souboru *nazev_souboru.pbix* – to má pozitivní dopady na výkonnost aplikací Power BI,
- ***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.,
- kalkulované sloupce lze využít i ***pro definování vazeb*** mezi tabulkami,

5.2 Příklady kalkulovaných sloupců

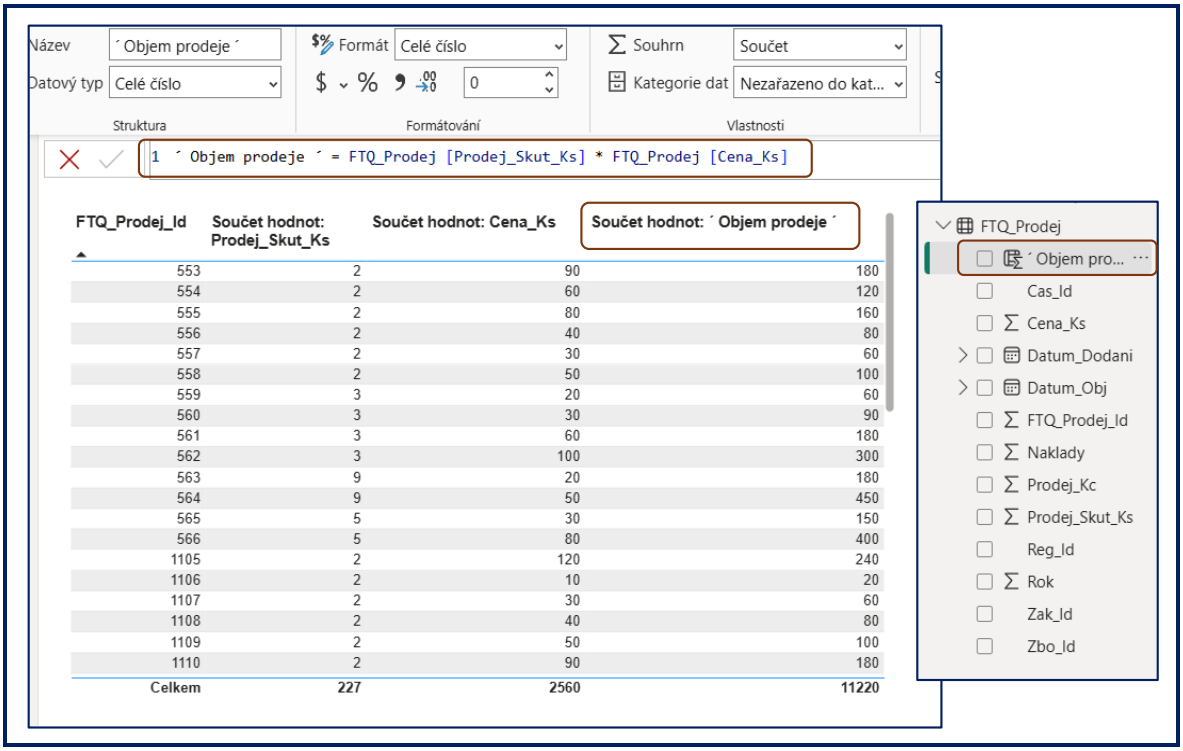
Příkladem kalkulovaného sloupce je ***výpočet zisku*** / ztráty z jednotlivých prodejů zboží:

```
Zisk = FTQ_Prodej [Prodej_Kc] - FTQ_Prodej [Naklady]
```

Příklad 5-1: Výpočet kalkulovaného sloupce zisku / ztráty

```
´ Objem prodeje ´ = FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks]
```

Příklad 5-2: Kalkulovaný sloupec pro objem prodeje v aktuálních cenách



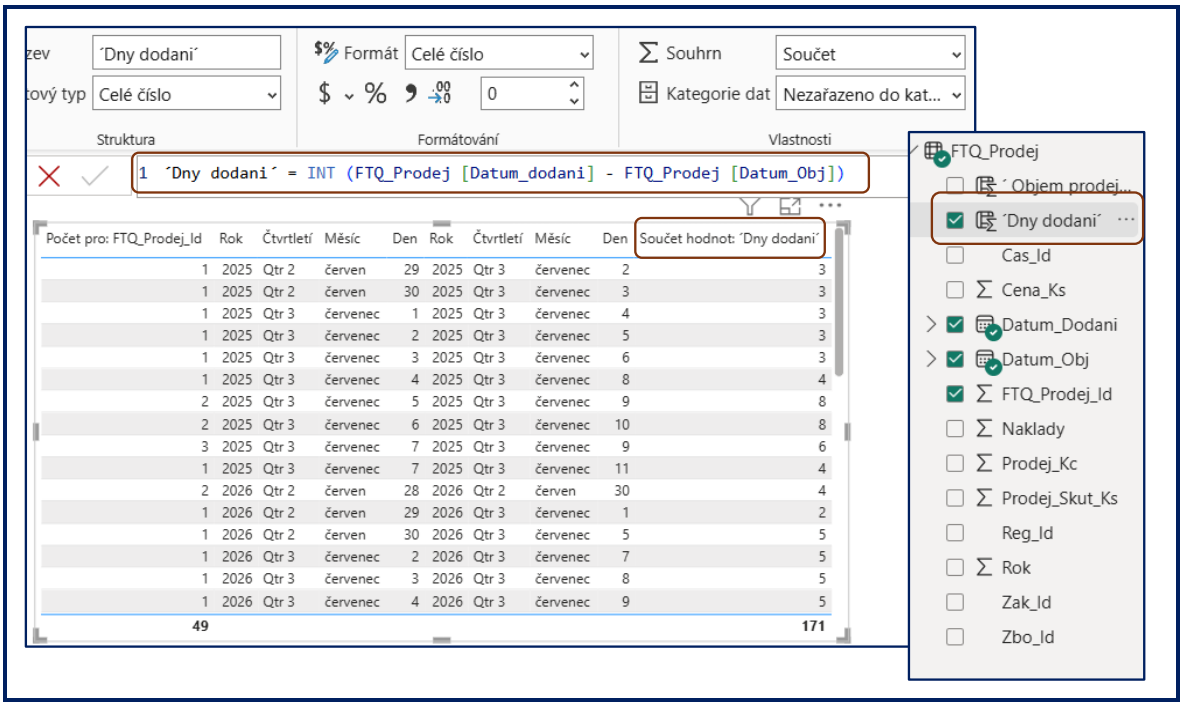
Obrázek 5-2: Příklad kalkulovaného sloupce

Dále je třeba např. zjistit ***počet dní dodání zboží od objednání***. V tomto případě mají obě položky formát datumu (*Date*) a výstupem by bylo zase datum. Abychom získali počet dnů dodání zboží je nutné konvertovat výsledek na celočíselnou hodnotu funkcí *INT*.

```
´Dny dodani´ = INT (FTQ_Prodej [Datum_dodani] - FTQ_Prodej [Datum_Obj])
```

Příklad 5-3: Počet dní dodání zboží od objednání

Řešení dokumentuje Obrázek 5-3:



Obrázek 5-3: Zobrazení hodnot počtu dní dodávek

Dalším příkladem, za předpokladu, že v tabulce nemáme název dne, 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 (DI_Cas [Cas_Den_v_Tydnu],
                    1,"Pondělí",
                    2,"Úterý",
                    3,"Středa",
                    4,"Čtvrtek",
                    5,"Pátek",
                    6,"Sobota",
                    7,"Neděle")
```

Příklad 5-4: funkce SWITCH pro přidání sloupce s názvem dne

5.3 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Kalkulované sloupce ***zabírají v datovém modelu větší prostor***, a tedy velkých tabulek je lepší se jim vyhnout.
- Na druhé straně je účelné použít kalkulované sloupce, pokud se mají kalkulované hodnoty stát ***obsahem filtrů*** nebo mají být součástí průřezů (*slicers*).
- Výhodou kalkulovaných sloupců jsou případy, kdy výpočet měr by byl časově náročný, a tedy výpočet kalkulovaného sloupce v době aktualizace datasetu by byl efektivnější. Na druhé straně řešení problému by se naskýtalo s použitím iterátorů (kapitola 11).
- Kalkulované sloupce se používají také v případech, kdy je třeba kategorizovat

textová nebo numerická data, např. při určování rozsahu hodnot u věkových kategorií (21 - 30, 31 – 40 apod.).

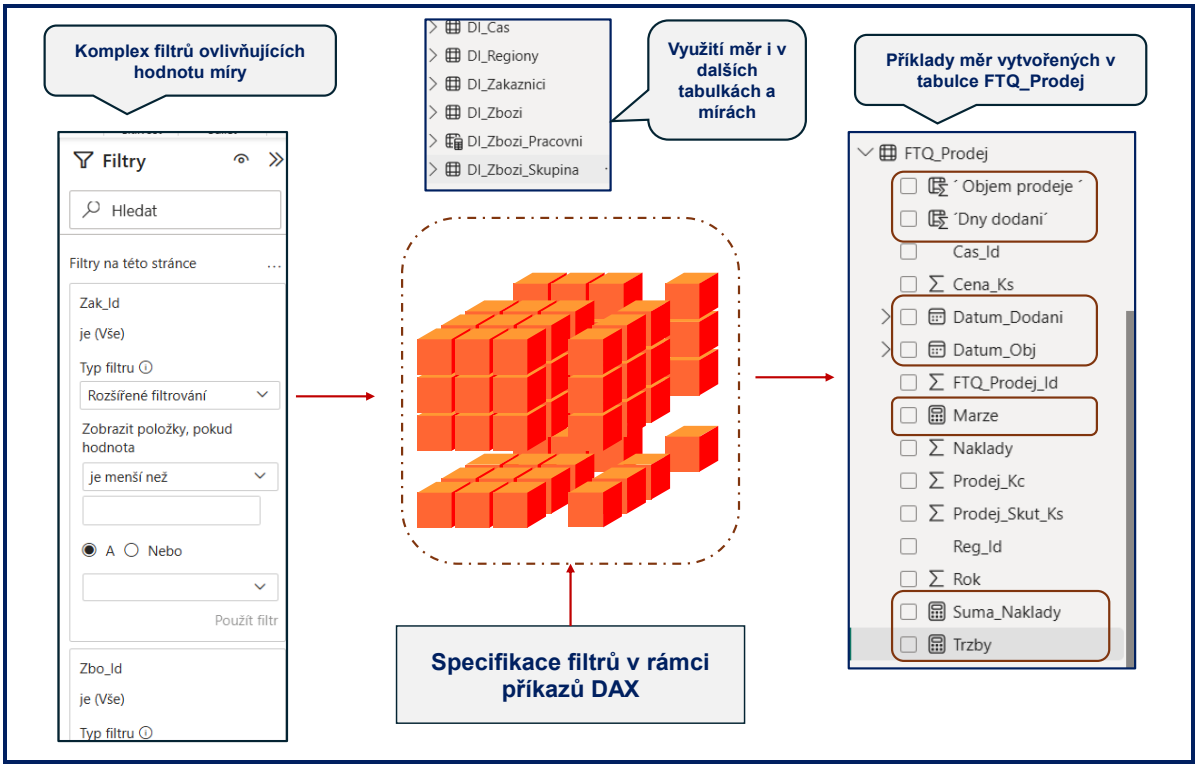
6. Míry (Measures)

Účelem kapitoly je podstatu míry a možnosti funkce, které jsou s nimi spojeny, a příklady jejich užití.

Míra, resp. *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*).

6.1 Principy míry a filtr kontext

Data jsou tak vypočítávána ***na úrovni více řádků, resp. agregací*** (nikoli 1 řádky) a s respektováním filtrů, které ovlivňují jednotlivé buňky. *Míry* se tak nemohou vztahovat k jednotlivým řádkům. Míra se může ale vztahovat i k 1 nebo více kalkulovaným sloupcům. Je součástí datové sady v Power BI – je k dispozici pro všechny reporty nad datovou sadou, Obrázek 6-1:



Obrázek 6-1: Princip míry

K ***vytváření a využití měr*** shrneme ***další poznámky***:

- Míra musí být ***definována v rámci tabulky***. Na druhé straně nemusí být reálně vázána s tabulkou a lze ji přenášet od jedné tabulky ke druhé, bez ztráty její funkcionality.
- Míra se realizuje ***na agregacích dat*** definovaných aktuálním kontextem. Pro agregace dat DAX disponuje řadou agregačních funkcí, agregátorů, nejběžnější je funkce SUM().
- Všechny míry, resp. jejich identifikace ***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,
- Power BI používá ve sloupcích tabulky nebo matice ***implicitní míry*** (*implicit measures*), tedy automaticky vytvořené agregace, průměry apod., jak ukazuje Obrázek 6-2 dole. Ta ale ne vždy zaručuje pnou kontrolu na kalkulací. Lepší varianta je použít ***explicitní míru*** (*explicit measures*), která je definovaná explicitně pro danou agregaci, resp. kalkulaci.

FTQ_Prodej_Id	Součet hodnot: Prodej_Skut_Ks	Součet hodnot: Cena_Ks	Součet hodnot: ´ Objem prodeje ´
553	2	90	180
554	2	60	120
555	2	80	160
556	2	40	80
557	2	30	60
558	2	50	100
559	3	20	60
560	3	30	90
561	3	60	180
562	3	100	300
563	9	20	180
564	9	50	450
565	5	30	150
566	5	80	400
1105	2	120	240
1106	2	10	20
1107	2	30	60
1108	2	40	80
1109	2	50	100
1110	2	90	180
Celkem	227	2560	11220

Obrázek 6-2: Implicitní míra

- Použití explicitní míry je nezbytné např. v situacích, kdy v jednotlivých řádcích ***zjišťujeme procentuální hodnoty***, např. procento marže. Jejich agregace musí být vytvořena na základě explicitní míry, tedy výpočtu procenta na bázi agregovaných hodnot jednotlivých sloupců.

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.

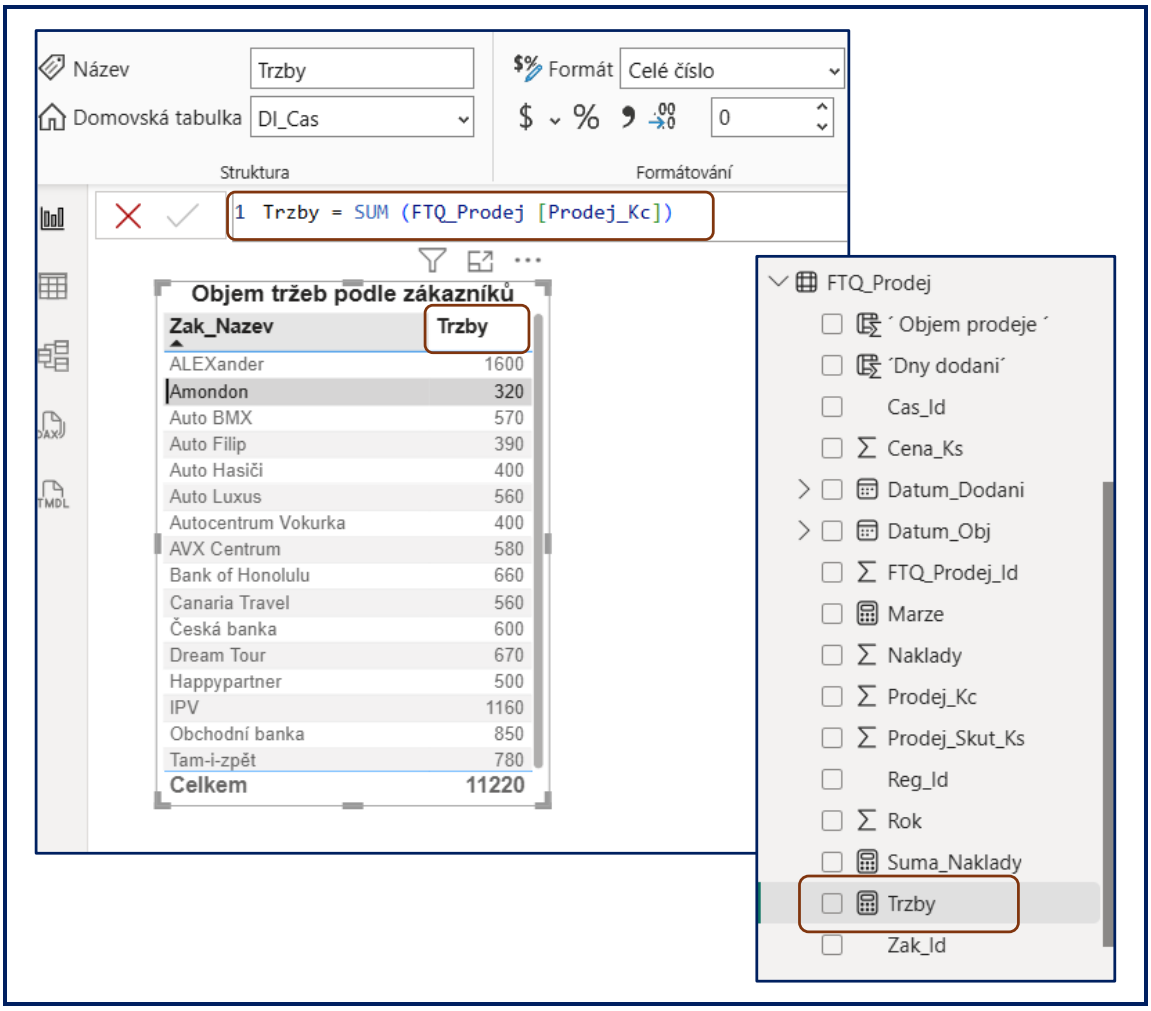
a. Výpočetní logika míry (např. pro DISTINCTCOUNT(), COUNTROWS() apod.) se vyhodnocuje pouze na vybraných řádcích a sloupcích podle filtrů.
b. Pro základní míry se realizuje na zbylých nebo aktivních řádcích faktové tabulky, zatímco pro ostatní míry ve spojení s dimenzionálními tabulkami je výpočet ovlivněn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

6.2 Příklady výpočtů míry:

```
Trzby = SUM (FTQ_Prodej [Prodej_Kc])
```

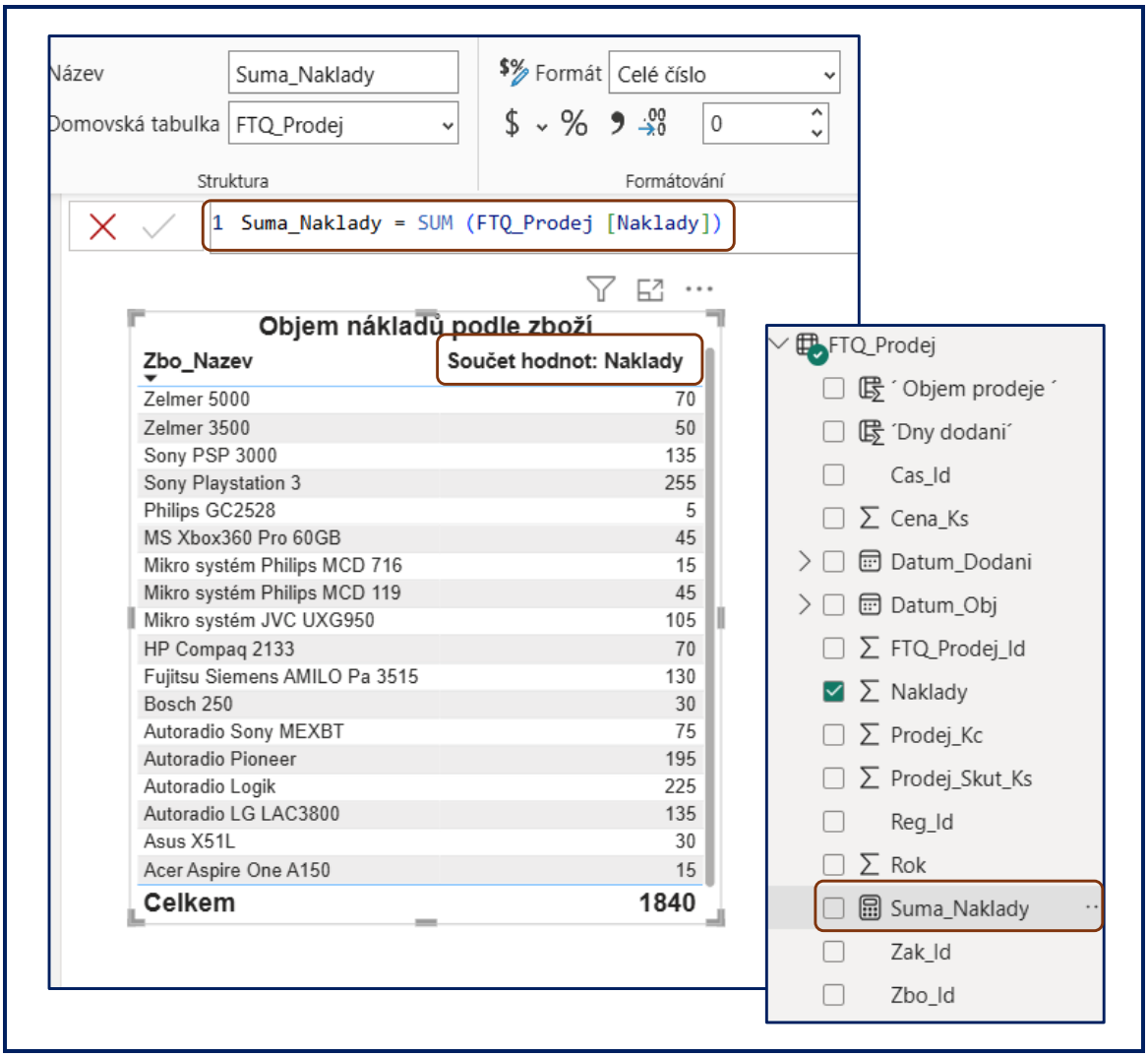
Příklad 6-1: funkce SUM pro míru tržeb



Obrázek 6-3: Výpočet míry "Trzby"

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

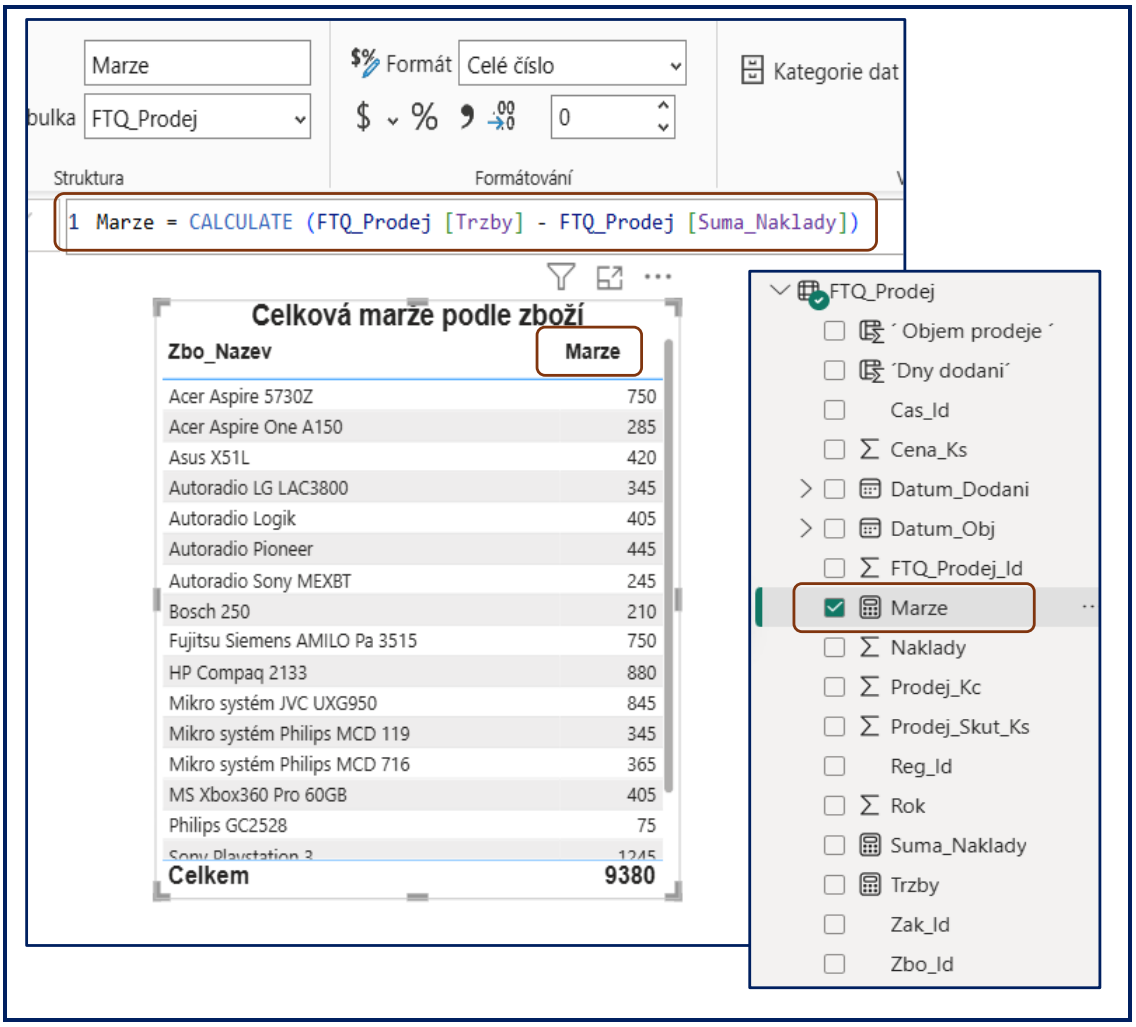
Příklad 6-2: Výpočet souhrnu nákladů



Obrázek 6-4: Míra pro výpočet sumy nákladů

Marze = CALCULATE (FTQ_Prodej [Trzby] - FTQ_Prodej [Suma_Naklady])

Příklad 6-3: Míra při výpočtu marže



Obrázek 6-5: Míry při výpočtu maže

Procento marže se získá s využití předchozí míry, tedy:

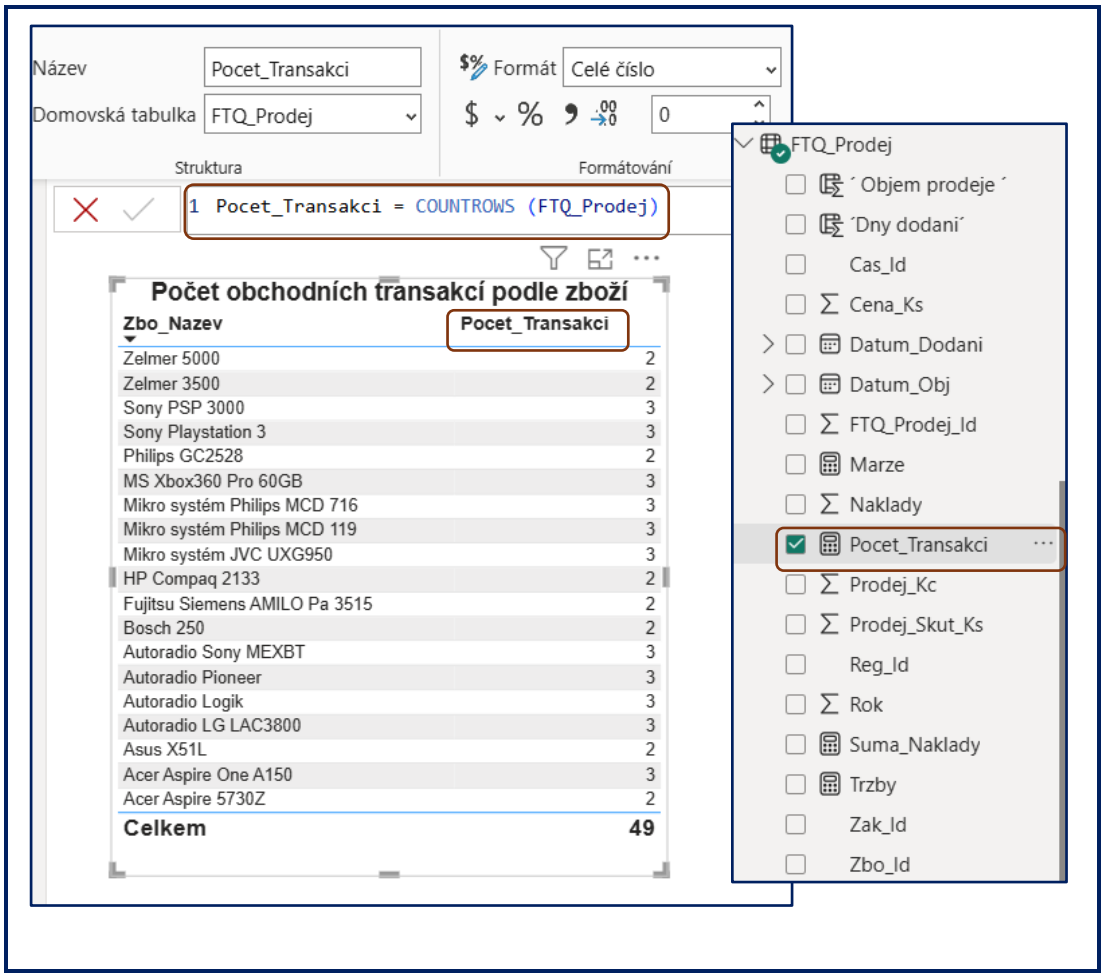
```
Procento_Marze = CALCULATE [Marze] / (FTQ_Prodej [Trzby])
```

Příklad 6-4: Míra při výpočtu procenta marže

Pro výpočet ***počtu prodejních transakcí***, jednotlivých řádků lze využít následující míru, která spočítá počet řádek faktové tabulky, tedy počet prodejních transakcí, viz Obrázek 6-6:

```
Pocet_Transakci = COUNTROWS (FTQ_Prodej)
```

Příklad 6-5: Výpočet počtu prodejních transakcí

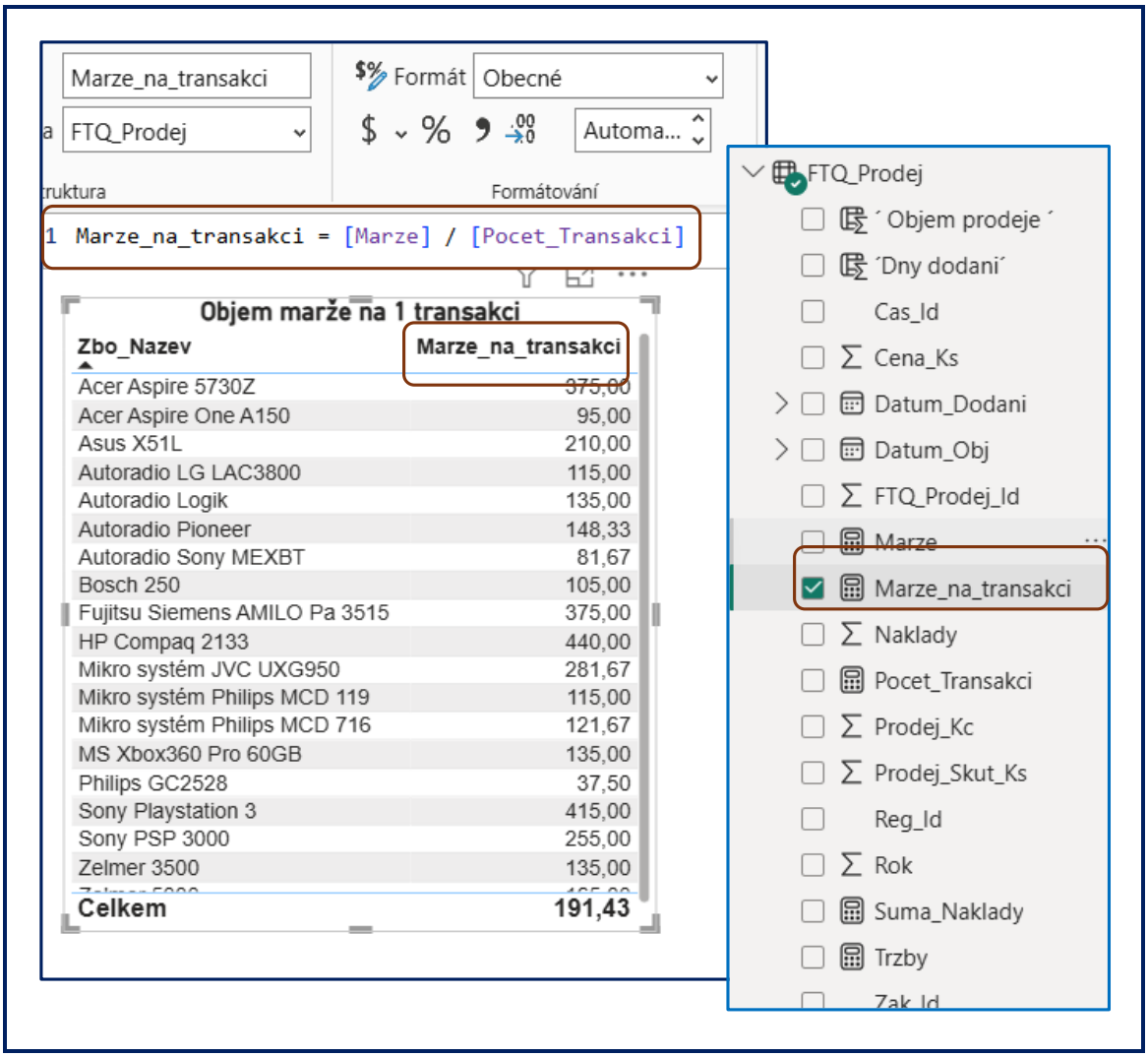


Obrázek 6-6: Výpočet počtu prodejních transakcí

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 6-7

```
Marze_na_transakci = [Marze] / [Pocet_Transakci]
```

Příklad 6-6: Vytvořené míry při definování nových měr

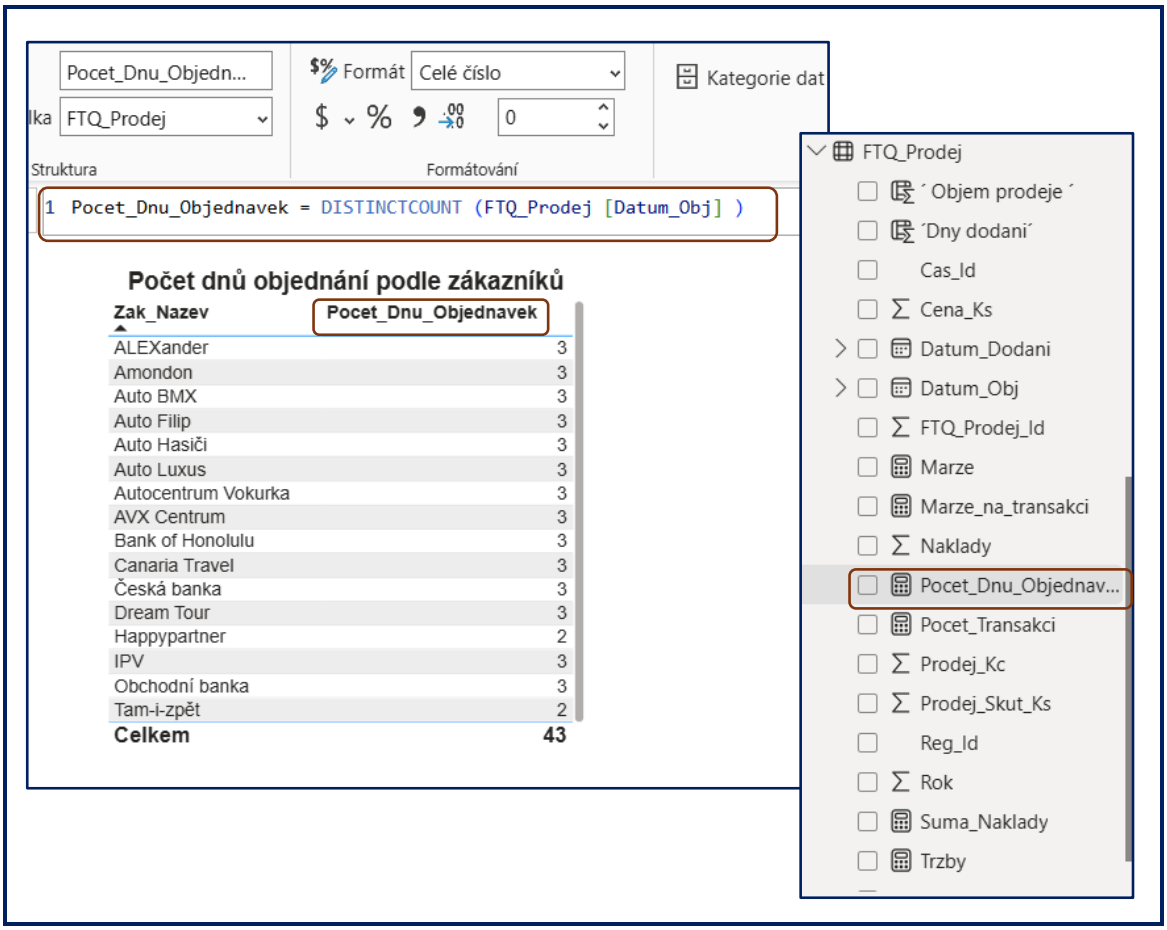


Obrázek 6-7: Výpočet marže na transakci

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_Objednavek = DISTINCTCOUNT (FTQ_Prodej [Datum_Obj] )
```

Příklad 6-7: Výpočet dnů, kdy byly zpracovány objednávky



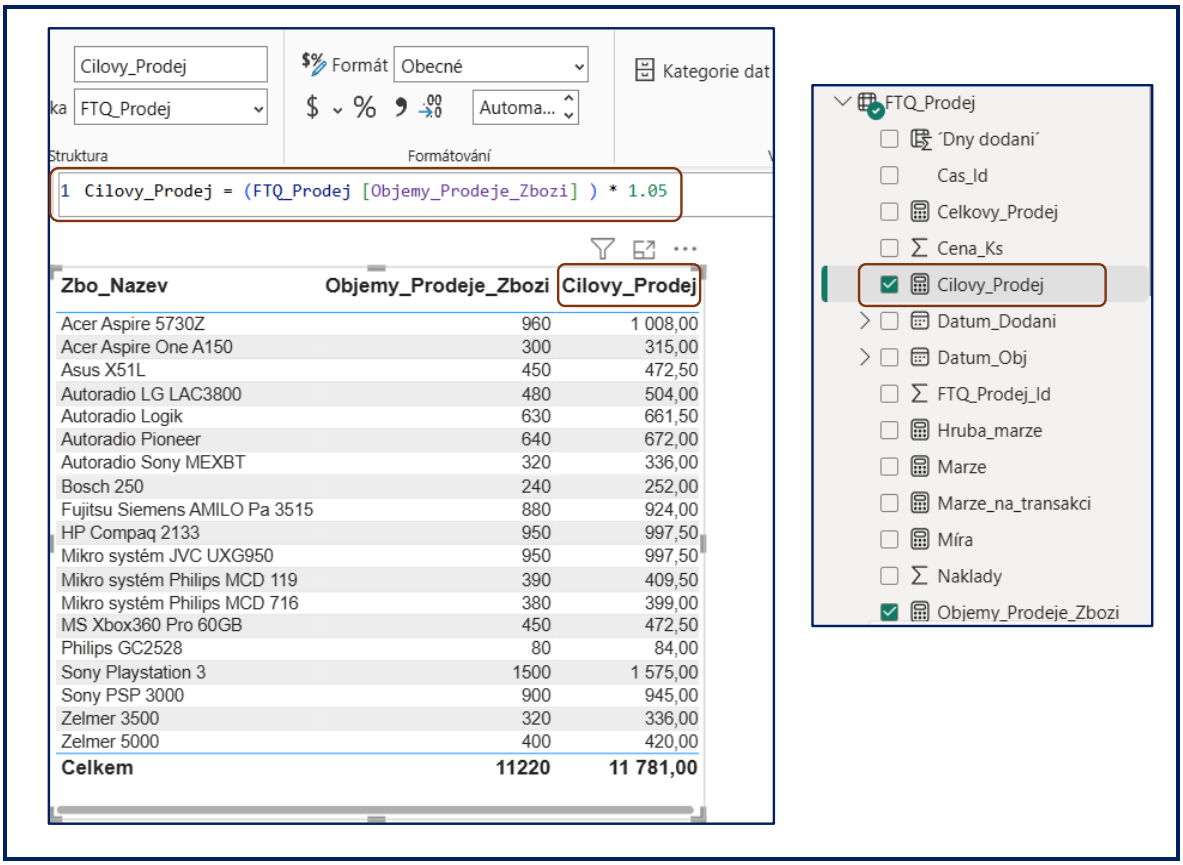
Obrázek 6-8: Počet dnů objednání zboží podle zákazníků

6.3 *Míry 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.

```
Cilovy_Prodej = (FTQ_Prodej [Objemy_Prodeje_Zbozi] ) * 1.05
```

Příklad 6-8: Cílová hodnota objemu prodeje zvýšené o 5 %



Obrázek 6-9: Určení cílové hodnoty objemu prodeje zboží

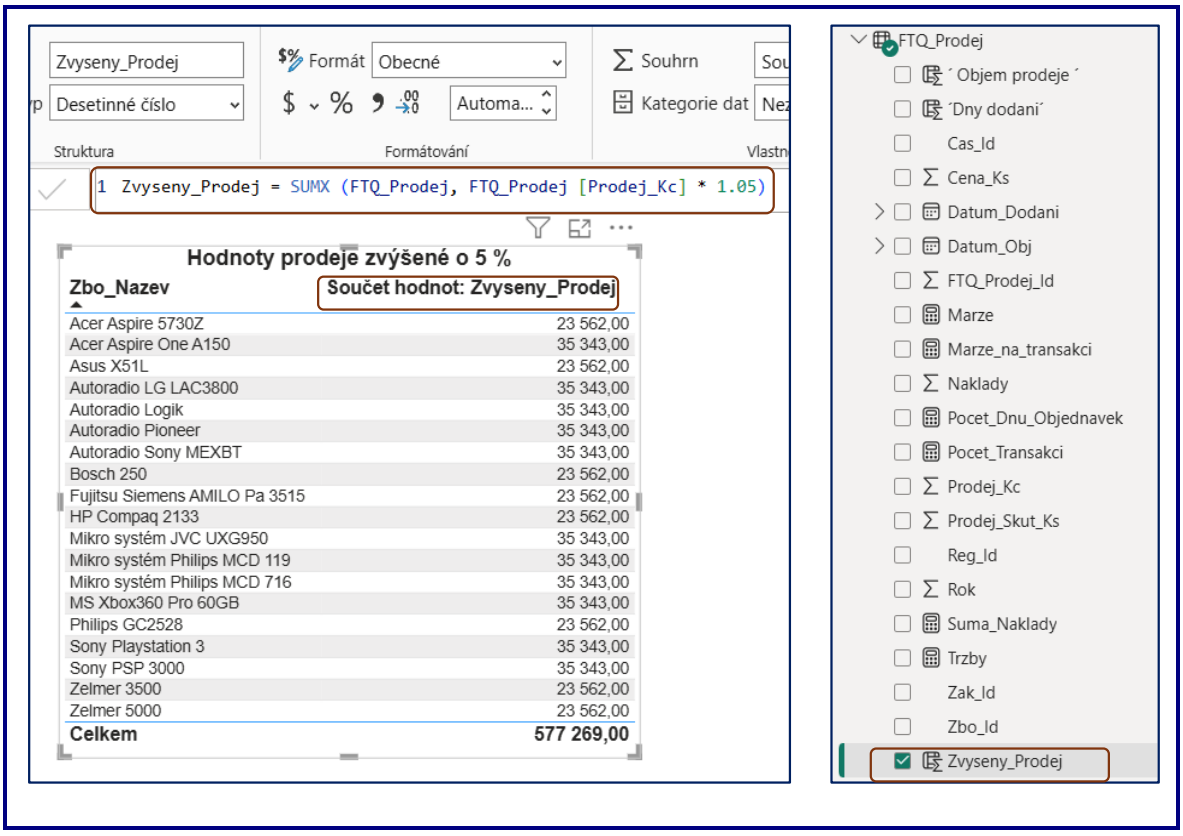
V některých případech může být vytvořena a zařazena do datasetu specifická tabulka se sloupci s cílovými hodnotami prodeje, nákladů apod., a to na požadované úrovni granularity časové dimenze (roky, kvartál, měsíc apod.). Míry pro cílové hodnoty mohou být zpřístupněny s pomocí funkce např. *LOOKUPVALUE()*,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

6.4 Iterátory, podstata

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***. Podrobněji: kapitola 10.

```
Zvyseny_Prodej =SUMX (FTQ_Prodej, ([Prodej_Kc] * 1.05))
```

Příklad 6-9: Využití řádkového kontextu s použitím iterátorů



Obrázek 6-10: Využití iterátoru SUMX pro nový kalkulovaný sloupec

6.5 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Analytik musí ***definovat kalkulovaný sloupec*** v těchto případech:
  - Pokud je třeba umístit kalkulované výsledky do jednotlivých řádků nebo průřezu, nebo použít kalkulovaný sloupec jako podmínku ve filtru v dotazu DAXu.
  - Pokud je třeba výraz přesně vázat k jednotlivým řádkám.
  - Pokud je třeba hodnoty nebo texty kategorizovat, jako např. věkové kategorie (15 – 18, 19 – 25 apod.).
- Analytik ***definuje míry*** v případech:
  - Při zpracování kalkulací, které musí reflektovat aktuální filtry určované uživatelem.
  - Při zpracování souhrnných a zejména poměrových hodnot za celou tabulku.
  - V případě možnosti vytvářet kalkulovaný sloupec nebo míru, doporučuje se použít míru s ohledem na její flexibilitu. Kalkulované sloupce je tak dobré omezit jen na případy, kdy je to nutné,

7. 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.

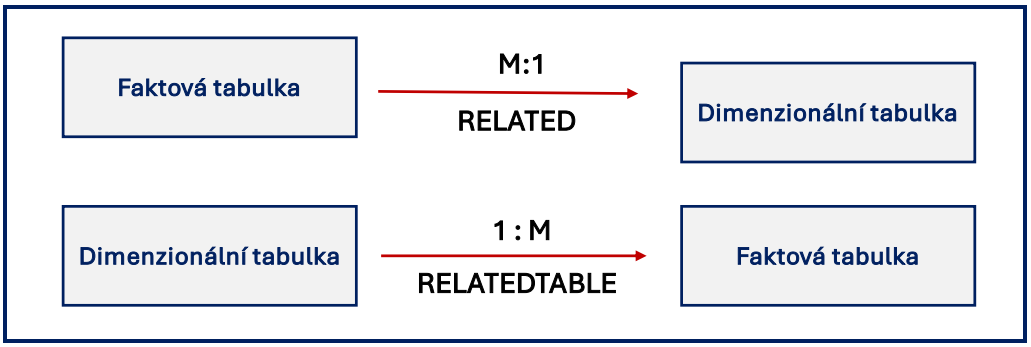
7.1 Řešení ve standardních vazbách

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.

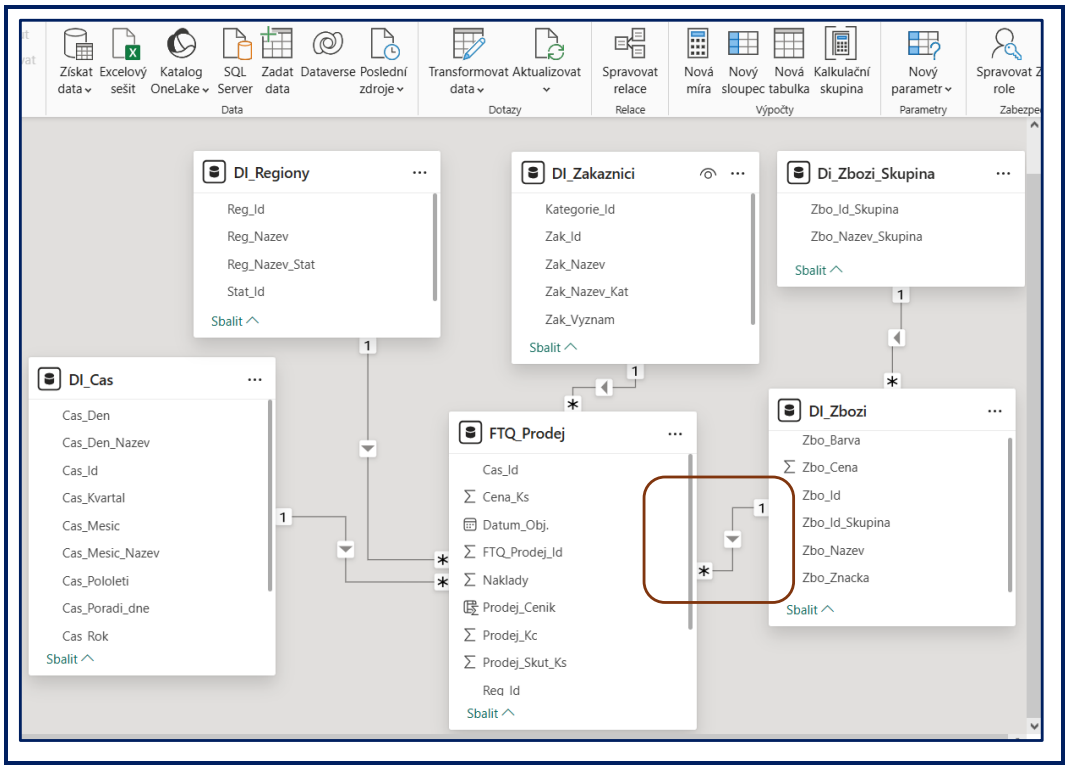
7.1.1 Řádkový kontext s vazbami tabulek

Pokud se ***zpracování v řádkovém kontextu*** bude vztahovat na 2 nebo více vzájemně provázaných tabulek pak je třeba využít parametrů ***RELATED*** nebo ***RELATEDTABLE***. Pro další příklady využijeme sémantický model na obrázku Obrázek 2-2.



Obrázek 7-1: Řešení vazeb v řádkovém kontextu s funkcemi RELATED a 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*. V tomto případě ***je podstatná kardinalita a směr vazby,*** tedy pro *FTQ_Prodej : DI_Zbozi* je kardinalita vazby *M : 1*, Obrázek 7-2:

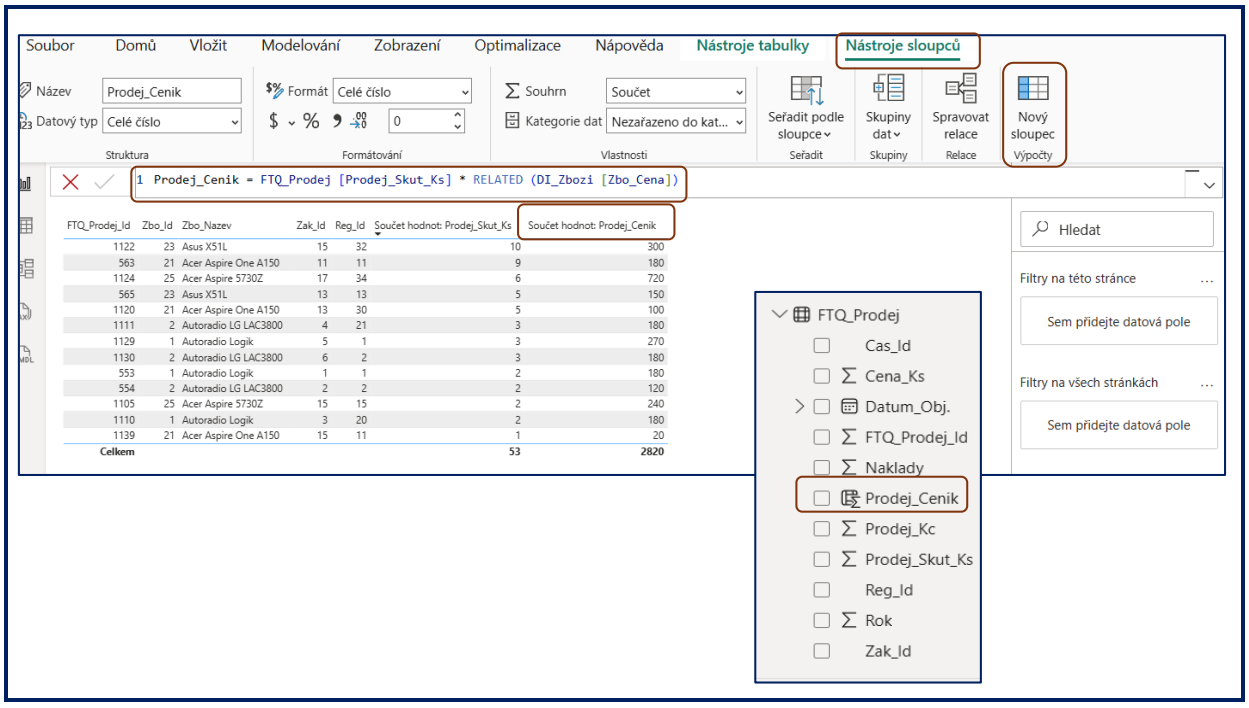


Obrázek 7-2: Funkce RELATED: vazba faktové a dimenzionální tabulky

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

```
Prodej_Cenik =
        FTQ_Prodej [Prodej_Skut_Ks] * RELATED (DI_Zbozi [Zbo_Cena])
```

Příklad 7-1: Kalkulovaný sloupec v řádkovém kontextu



Obrázek 7-3: 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 *Zbo_Cena*, by bylo více hodnot ve sloupci *Prodej_Skut_Ks* . To znamená, kalkulovaný sloupec by se nevytvořil, jak ukazuje Obrázek 7-2:

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***.

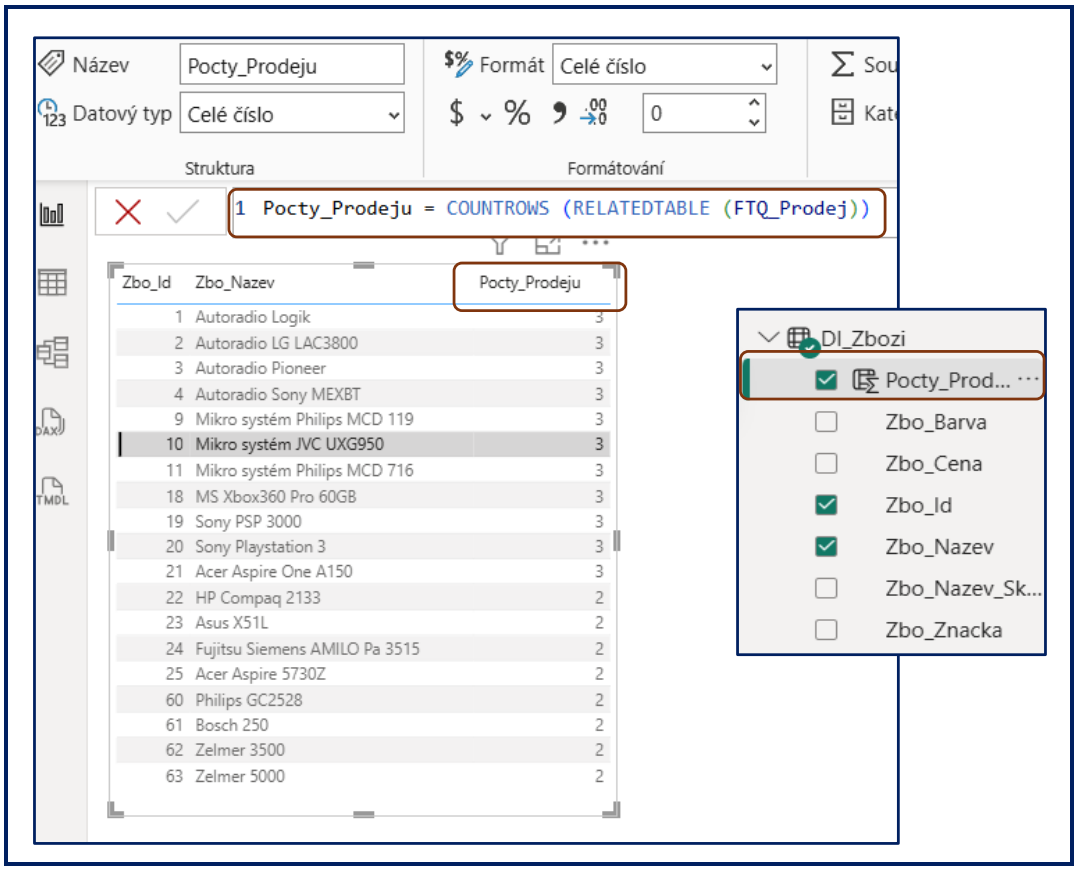
Předpokládejme nyní, oproti předchozím příkladům, že informace o zboží jsou uloženy ve třech propojených tabulkách na principu SNOWFLAKE (*DI_Zbozi*, – *DI_Zbozi_Skupina).* Parametr *RELATED* tak funguje i na bázi několika propojených tabulek, jako v případě vazby faktové tabulky *FTQ_Prodej* ke zmíněným propojeným tabulkám *DI_Zbozi*, – *DI_Zbozi_Skupina.* To dokumentuje Obrázek 7-4.

Obrázek 7-4: Funkce RELATED na několika propojených tabulkách

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 (Příklad 7-2):

```
=COUNTROWS (RELATEDTABLE (FTQ_Prodej))
```

Příklad 7-2: Počty prodejů jako kalkulovaný sloupec tabulky DI_Zbozi

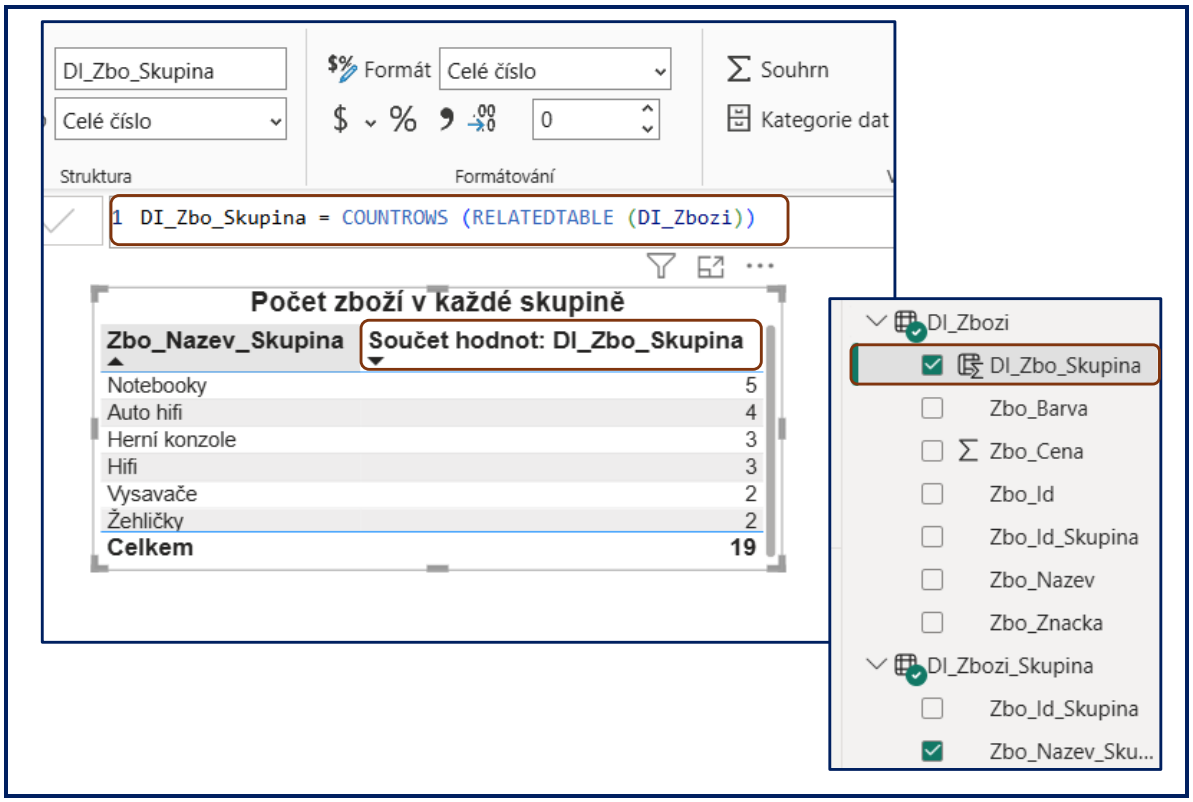


Obrázek 7-5: Řešení vazeb s využitím funkce RELATEDTABLE

Nebo obdobně, pokud chceme zjistit kolik zboží je v každé skupině s kalkulovaným sloupcem v tabulce *DI_Zbozi,* to ukazuje Příklad 7-3 a Obrázek 7-6:

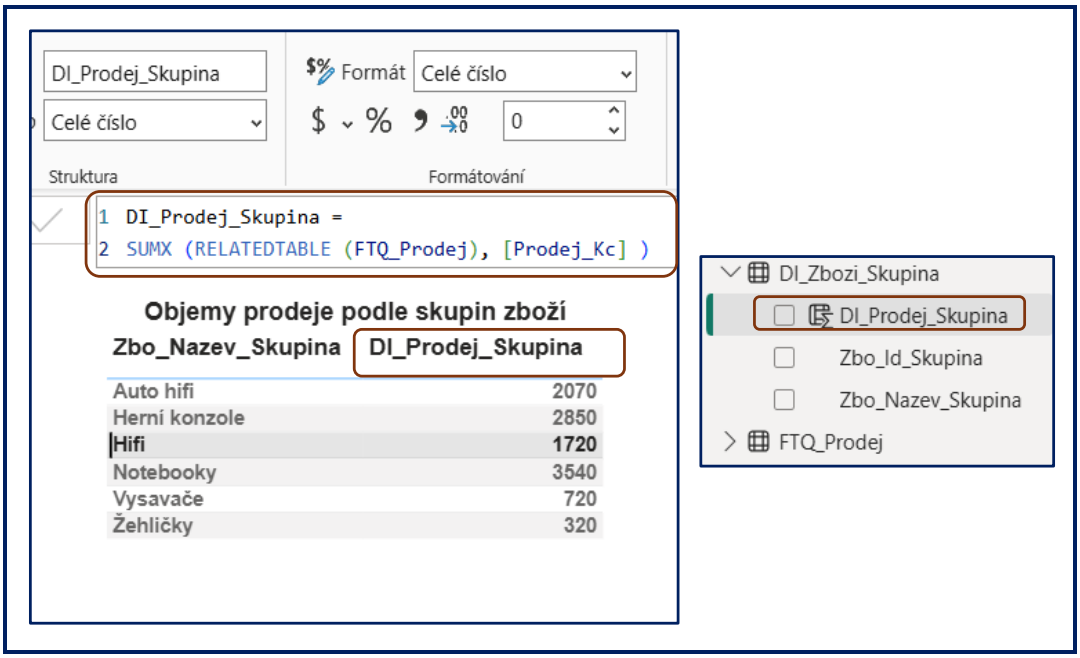
```
DI_Zbo_Skupina  = COUNTROWS (RELATEDTABLE (DI_Zbozi))
```

Příklad 7-3: Počet zboží v každé skupině – kalkulovaný sloupec v tabulce DI_Zbozi



Obrázek 7-6: Počet zboží v každé skupině – kalkulovaný sloupec v tabulce Dl_Zbozi

Funkce RELATEDTABLE se často využívá ve spojení s iterátory, jako v případě, kde požadujeme zjistit objem prodeje pro každou skupinu a vytvořit proto v tabulce *Dl_Zbozi_Skupina* kalkulovaný sloupec (Příklad 7-4)

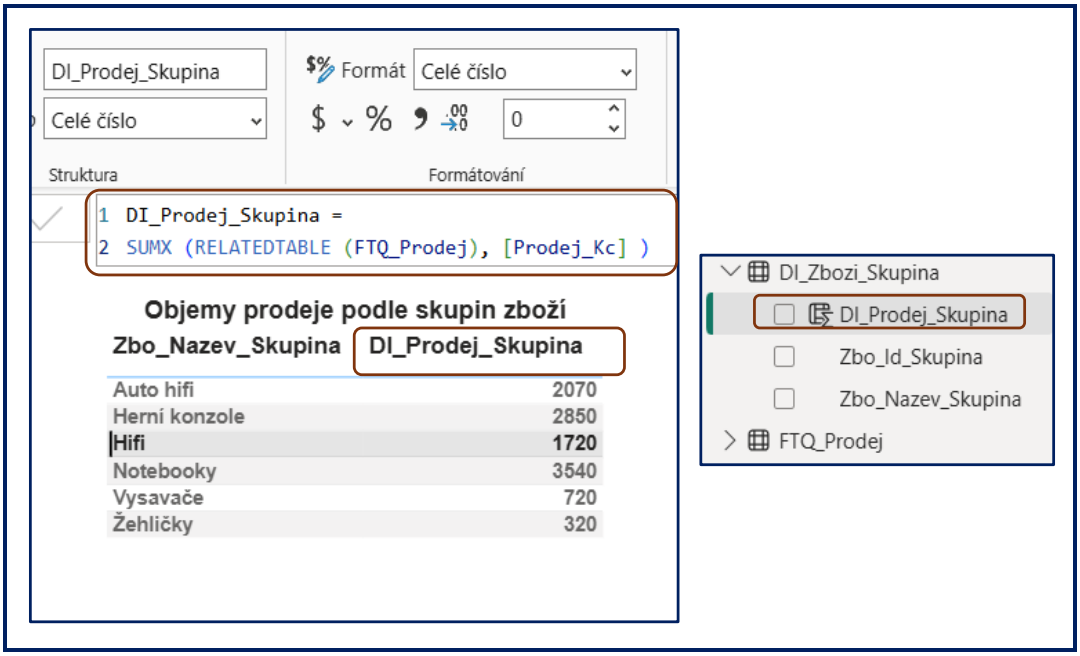


Obrázek 7-7: Využití iterátoru s funkcí RELATEDTABLE

	Dl_Prodej_Skupina =

```
SUMX (RELATEDTABLE (FTQ_Prodej), [Prodej_Kc] )
```

Příklad 7-4: Funkce RELATEDTABLE ve spojení s iterátory



Obrázek 7-7: Využití iterátoru s funkcí RELATEDTABLE

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 objednání 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*

7.1.2 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.

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. *Zbo_Cena* = 50 (Příklad 7-5)

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

Příklad 7-5: Výpočet míry s filtrem tabulky DI_Zbozi: Zbo_Cena = 50

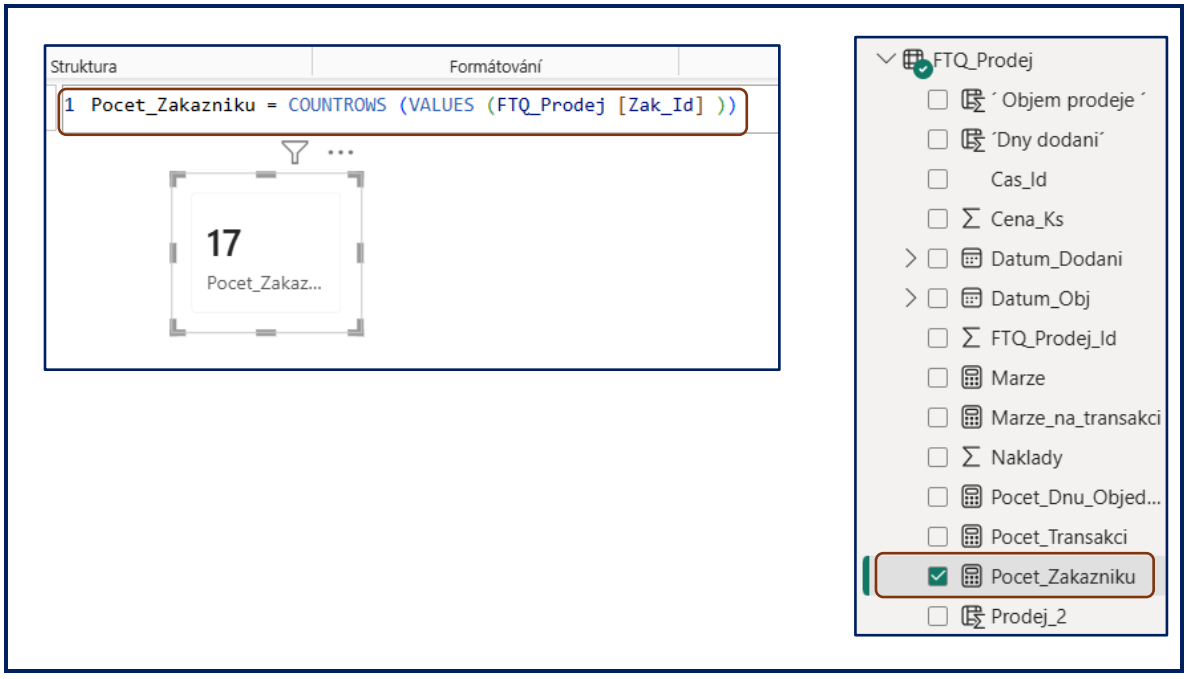
Filtr na tabulce DI_Zbozi se promítne takto:

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***, ukazuje Příklad 7-6:

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

Příklad 7-6: Funkce VALUES



Obrázek 7-8: Počet zákazníků z FTQ_Prodej s využitím funkce VALUES

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***.

7.2 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Pokud se ***zpracování v řádkovém kontextu*** bude vztahovat na 2 nebo více vzájemně provázaných tabulek pak je třeba využít parametrů ***RELATED*** nebo

RELATEDTABLE.

- ***RELATED i RELATEDTABLE mohou 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***.
- ***Pokud je řádkový kontext na straně vazby „1“*** je třeba, pak se k tomu využije parametr ***RELATEDTABLE***.
- ***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*).

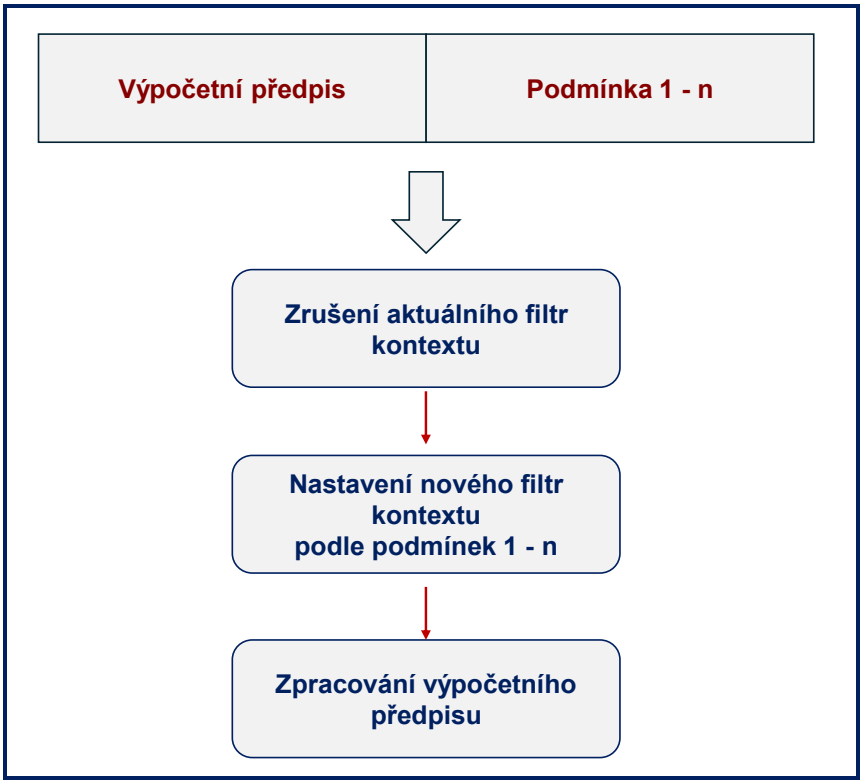
8. Funkce CALCULATE()

Funkce *CALCULATE()* se považuje ***za nejdůležitější, nejužitečnější a nejkomplexnější funkci*** jazyka DAX.

Účelem této kapitoly je charakterizovat funkce *CALCULATE()* ve větším detailu a se všemi podstatnými parametry.

(Zdroj: Ferrari, Russo, 2026).

Podstatnou charakteristikou *CALCULATE()* je to, že vytváří nový filtr kontext, resp. 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.



Obrázek 8-1: Základní princip CALCULATE ()

8.1 Možnosti užití funkce CALCULATE ()

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. ***filtr argumenty*** 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*** (tzv. filtr argument) 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***,
- když provedení funkce ***CALCULATE() skončí*** nový filtr kontext se zruší a jako aktivní se vrátí původní filtr kontext.

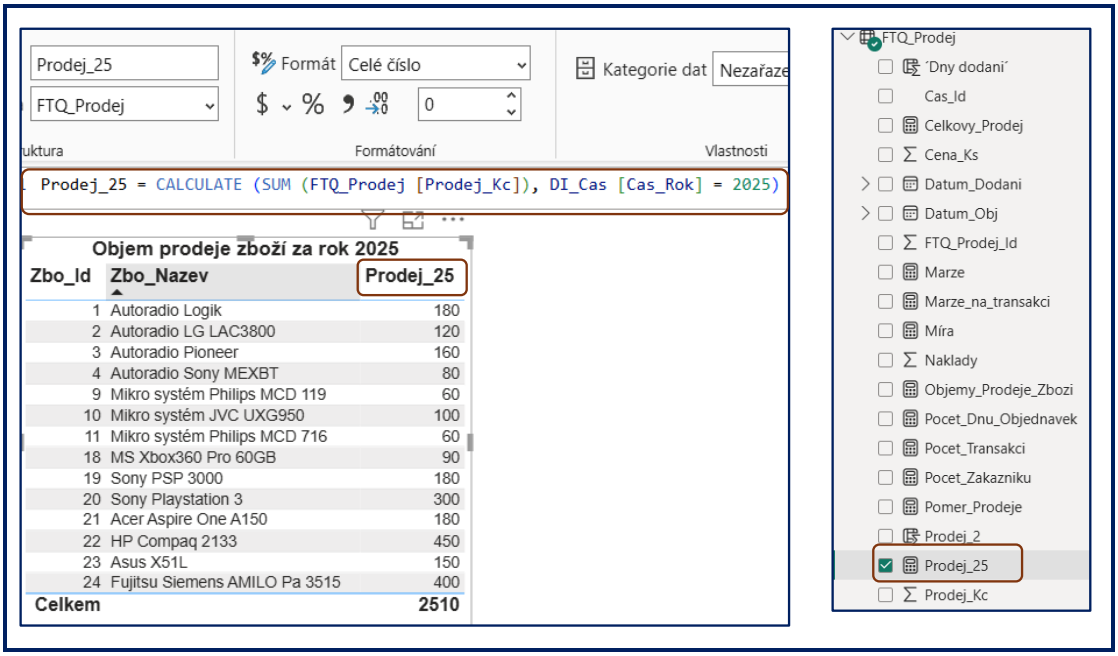
CALCULATE() rozeznává ***2 typy filtrů***:

- ***booleovské podmínky***, jako např. *DI_Zbozi [Cena] > 2000*, tyto filtry se používají na jednotlivých sloupcích – s výslednou hodnotou *Ano / Ne*, resp. *TRUE / FALSE*. V tomto případě DAX booleovské podmínky následně transformuje na tabulky hodnot,
- ***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()* ukazuje Obrázek 8-2:

```
Prodej_25 = CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Cas [Cas_Rok] = 2025)
```

Příklad 8-1: Výpočet hodnoty prodeje v roce 2025

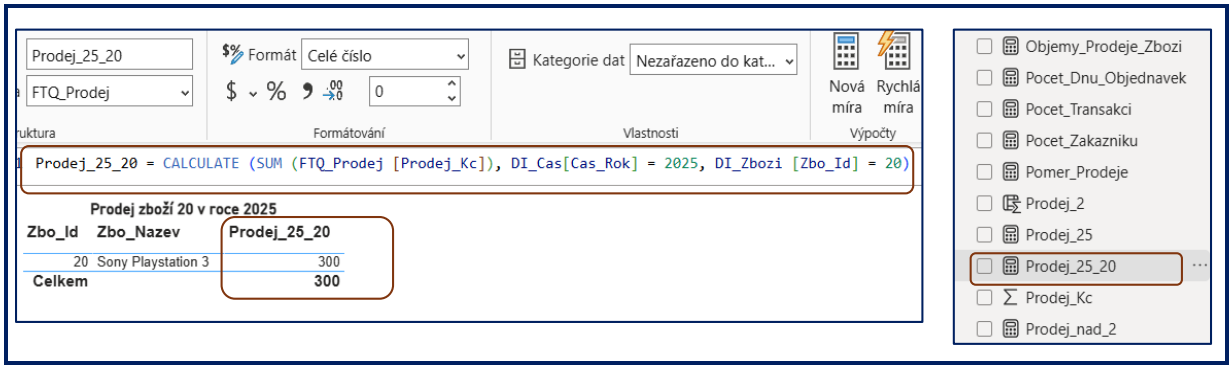


Obrázek 8-2: Objem prodeje zboží v roce 2025

8.1.1 Další příklady

```
Prodej_25_20 = CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Cas[Cas_Rok] = 2025, DI_Zbozi [Zbo_Id] = 20)
```

Příklad 8-2: Prodej zboží pouze v roce 2025 a současně s klíčem 20

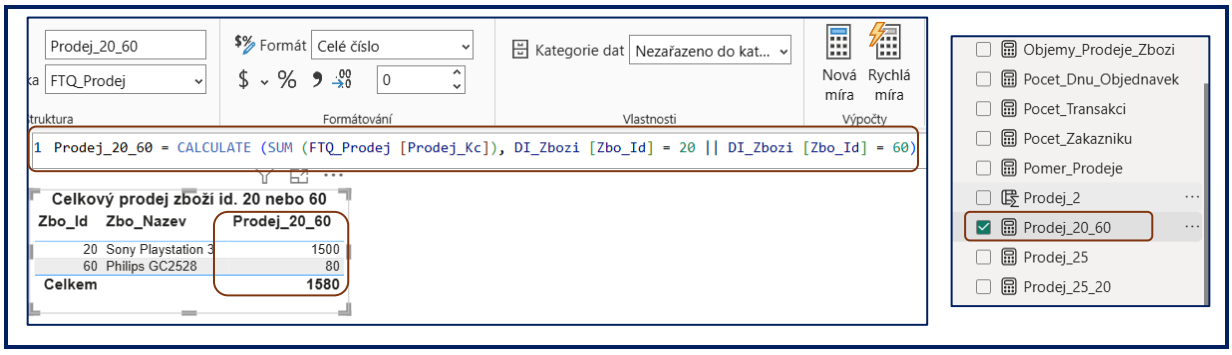


Obrázek 8-3: Prodej zboží 20 pouze v roce 2025

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.:

```
Prodej_20_60 = CALCULATE (SUM (FTQ_Prodej [Prodej_Kc]), DI_Zbozi [Zbo_Id] = 20 || DI_Zbozi [Zbo_Id] = 60)
```

Příklad 8-3: Operátor pro vyjádření logického součtu ||, nebo OR

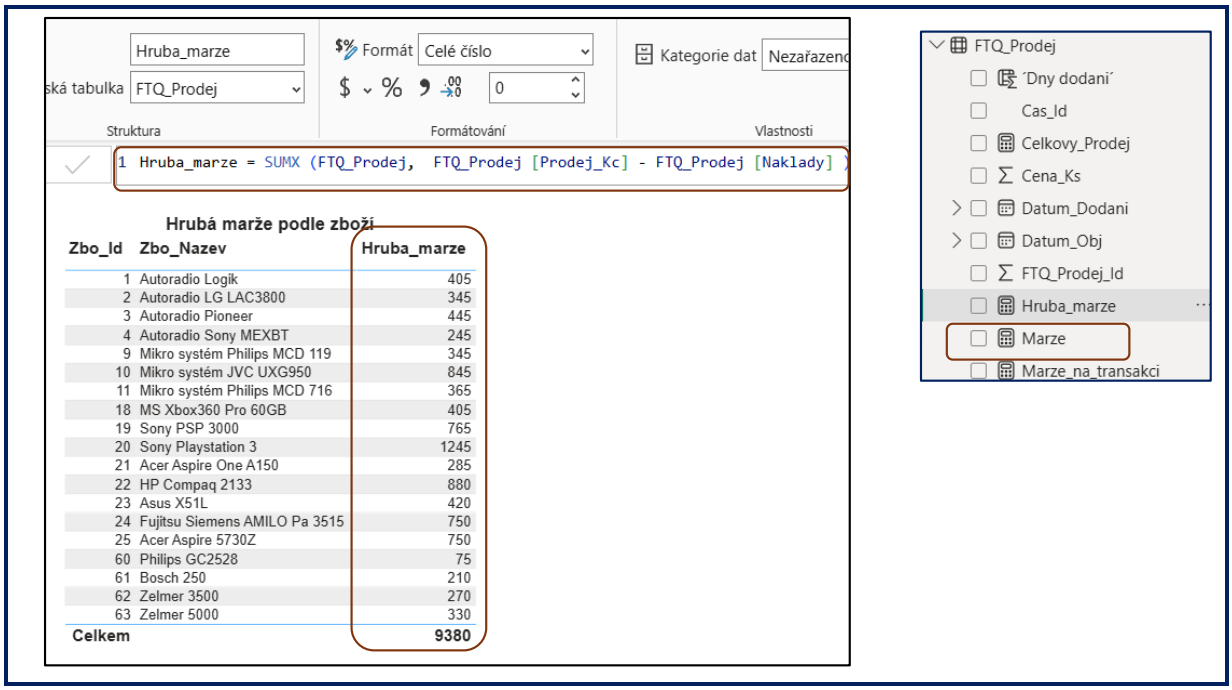


Obrázek 8-4: Vyjádření logického součtu ||, nebo OR

S využitím CALCULATE() lze velmi ***efektivně využívat dříve vytvořených měr v nově vytvořeném filtr kontextu.*** Např. potřebujeme zjistit hrubou marži pro zboží typu „Sony“. To dokumentuje Příklad 8-4 a Příklad 8-5. V prvním kroku vytvoříme míru pro hrubou marži:

```
Hruba_marze = SUMX (FTQ_Prodej,  FTQ_Prodej [Prodej_Kc] - FTQ_Prodej [Naklady] )
```

Příklad 8-4: Hrubá maže jako rozdíl objemu prodeje zboží a nákladů prodeje

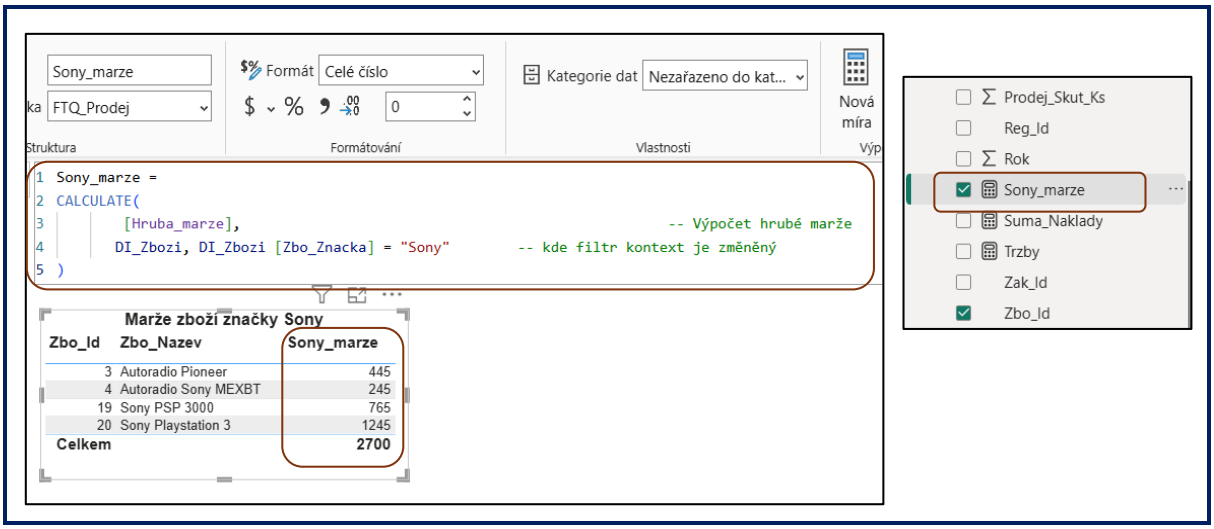


Obrázek 8-5: Hrubá marže podle zboží

V druhém kroku zjišťujeme pomocí funkce *CALCULATE()* hrubou marži pro zboží značky „Sony“.

```
Sony_marze =
CALCULATE(
    [Hruba_marze],                              -- Výpočet hrubé marže
    DI_Zbozi, DI_Zbozi [Zbo_Znacka] = „Sony“    -- kde filtr kontext je změněný
)
```

Příklad 8-5: Výpočet hrubé marže pro zboží typu Sony



Obrázek 8-6: Marže pro zboží značky Sony

Filtr argument je tabulka, resp. seznam hodnot, které budou patrné pro sloupce v průběhu vyhodnocení výrazu,

8.2 CALCULATE() v řádkovém kontextu

Další podstatnou úlohou funkce CALCULATE() je ***změna, resp. transformace*** (v případě potřeby) ***řádkového kontextu na filtr kontext***. Pro tento případ použijeme pro dokumentaci výpočet souhrnu ceníkových cen *ZboCena* v dimenzionální tabulce *Dl_Zbozi*.

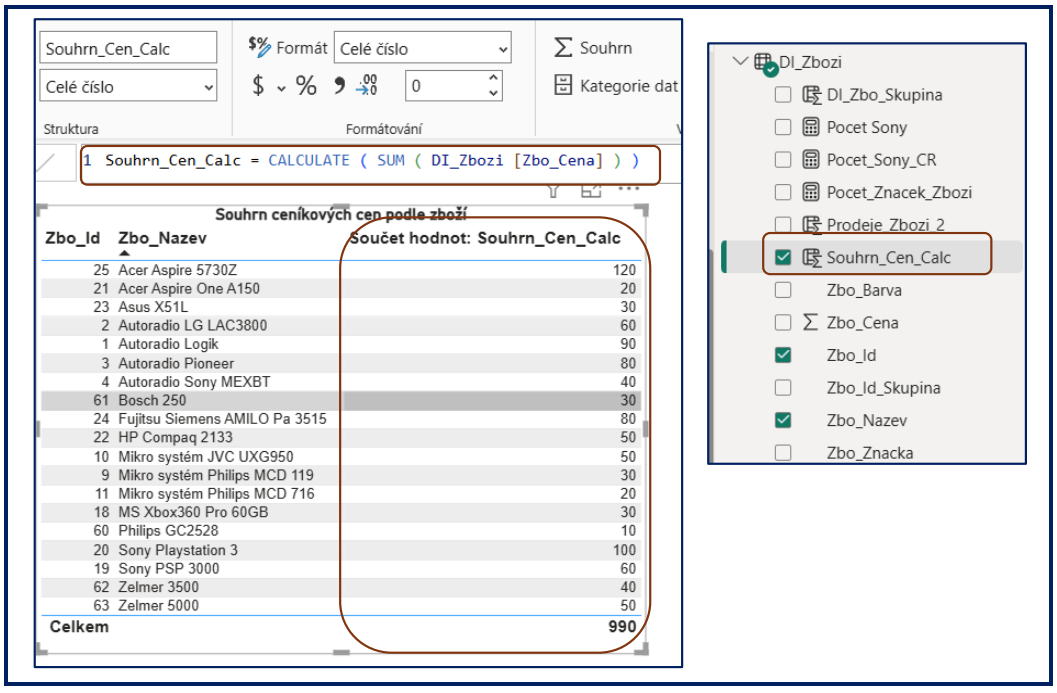
```
Souhrn_Cen = SUM ( Dl_Zbozi [Zbo_Cena] )
```

Příklad 8-6: Souhrn ceníkových cen v kalkulovaném sloupci (Souhrn_Cen)

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 ( Dl_Zbozi [Zbo_Cena] ) )
```

Příklad 8-7: Souhrny ceníkových cen podle jednotlivých druhů zboží



Obrázek 8-7: Souhrn ceníkových cen podle zboží

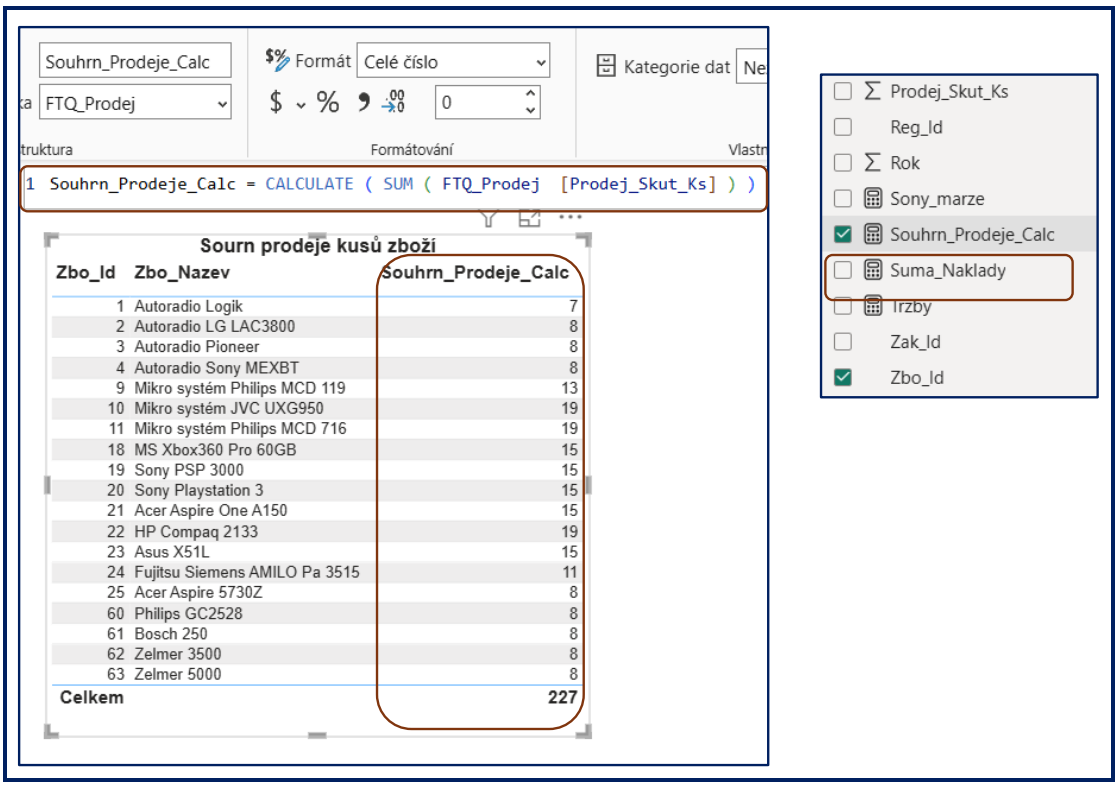
V tomto případě neobsahuje zápis *CALCULATE()* ***žádné filtry*** a tedy nemění existující filtr kontext, ale na druhé straně ***transformuje existující řádkový kontext na filtr kontext***. Uskutečňuje tak ***transformaci kontextu***. Na základě tohoto zápisu výraz *SUM (Dl_Zbozi [Zbo_Cena])* je vypočítáván 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] ) )
```

Příklad 8-8: Transformace kontextu pro automatické promítání filtrů do vázaných tabulek



Sourn prodeje kusů zboží

Zbo_Id	Zbo_Nazev	Souhrn_Prodeje_Calc
1	Autoradio Logik	7
2	Autoradio LG LAC3800	8
3	Autoradio Pioneer	8
4	Autoradio Sony MEXBT	8
9	Mikro systém Philips MCD 119	13
10	Mikro systém JVC UXG950	19
11	Mikro systém Philips MCD 716	19
18	MS Xbox360 Pro 60GB	15
19	Sony PSP 3000	15
20	Sony Playstation 3	15
21	Acer Aspire One A150	15
22	HP Compaq 2133	19
23	Asus X51L	15
24	Fujitsu Siemens AMILO Pa 3515	11
25	Acer Aspire 5730Z	8
60	Philips GC2528	8
61	Bosch 250	8
62	Zelmer 3500	8
63	Zelmer 5000	8
Celkem		227

Obrázek 8-8: Souhrn projede zboží v kusech

To znamená, že ***filtry z tabulky zboží*** (*Dl_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.

8.3 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* (*Dl_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 (Dl_Zbozi [Cena] )…
```

Příklad 8-9: Parametr ALLSELECTED pro zrušení filtrů kromě aktuálně nastavených

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.

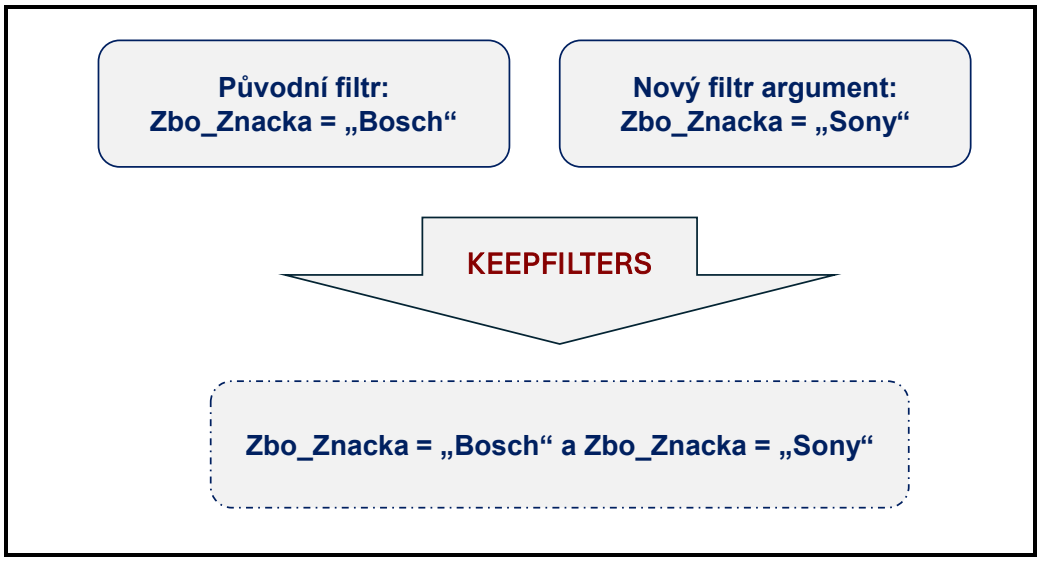
8.4 Funkce KEEPFILTERS ()

Zatím jsme vycházeli z předpokladu, že filtr argumenty *CALCULATE()* přepsaly všechny existující filtry na daném stejném sloupci. Pokud nechceme přepsat existující filtry, lze k tomu použít funkci *KEEPFILTERS* jako druhý modifikátor funkce *CALCULATE()*. Ta na místo přepsání existujícího filtru na sloupci, pouze přidá k němu nový filtr. Jinými slovy zachová vnější filtr v rámci vnitřního filtr kontextu (ze zápisu CALCULATE). To dokumentuje následující příklad.

```
Prodej_Sony_1 =
    CALCULATE(
      [Celkovy_Prodej],
      KEEPFILTERS ('DI_Zbozi' [Zbo_Znacka] = "Sony" )
)
```

Příklad 8-10: Uplatnění funkce KEEPFILTERS

Přepokládejme, že původní filtr je nastaven takto: *('DI_Zbozi' [Zbo_Znacka] = Bosch").* Výsledek operace dokumentuje Obrázek 8-9:



Obrázek 8-9: Princip funkce KEEPFILTERS

Ve výsledku se pak uplatnění takto postaveného filtru promítne např. do tabulky takto (Sony s hodnotou, Bosch při dané kombinaci filtrů jako BLANK):

Bosch	
Sony	150

KEEPFILTERS() není ve skutečnosti modifikátor funkce *CALCULATE(),* ale pouze filtr argumentu. Modifikuje tak způsob, jak bude filtr argument uplatněn v novém filtr kontextu. To ukazuje další příklad:

```
…..
      KEEPFILTERS ( DI_Zbozi [Zbo_Znacka] = „Sony“ )
….
```

Příklad 8-11: Uplatnění modifikátoru *KEEPFILTERS*

Použití *KEEPFILTERS ()* znamená, že uvedený filtr „Sony“ nepřepíše původní filtr, ale přidá ho do nového filtr kontextu. Nemění tak sémantiku celé *CALCULATE ()* funkce.

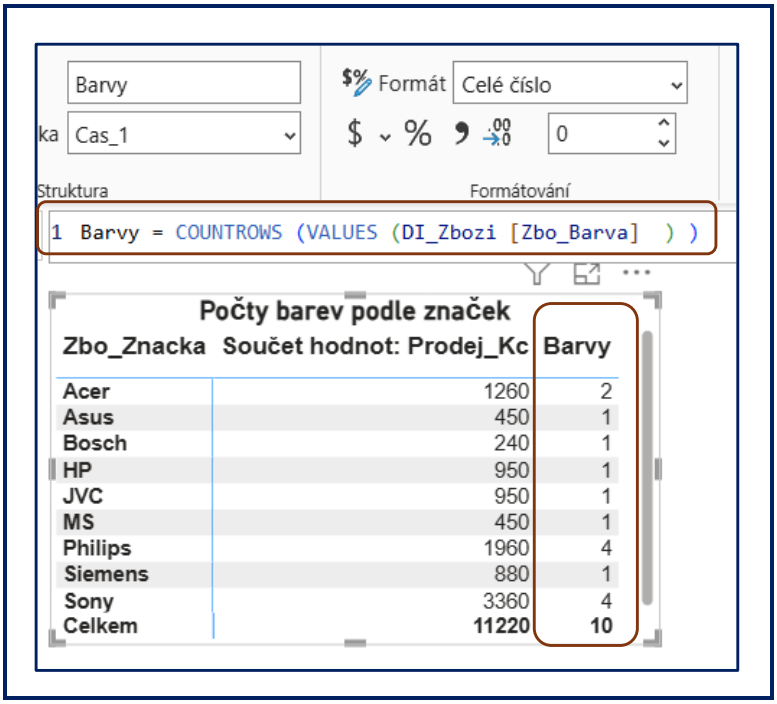
8.5 Funkce VALUES ()

Tabulková funkce VALUES () může sloužit jako alternativa k funkci KEEEPFILTERS (). Vrací hodnoty sloupce, které jsou aktuálně viditelné na základě filtr kontextu. Např. pokud je třeba zjistit počet barev zboží pro určitý produkt, lze zadat počet řádek ve výsledku funkce VALUES. K tomu se uplatní filtr kontext prvku (*Cell Filter Context*), jak ukazuje další příklad:

```
Barvy = COUNTROWS (VALUES (´DI_Zbozi´ [Zbo_Barva]  ) )
```

Příklad 8-12: Uplatnění funkce VALUE ()

Řešení dokumentuje Obrázek 8-10:



Obrázek 8-10: Využití funkce VALUE pro zjištění počtu barev

Funkce *VALUE ()* je velmi účelné i ve spojení s funkcí *CALCULATE ()* protože umožňuje přístup pouze k hodnotám sloupce, které jsou viditelné jako vnější filtr kontext.

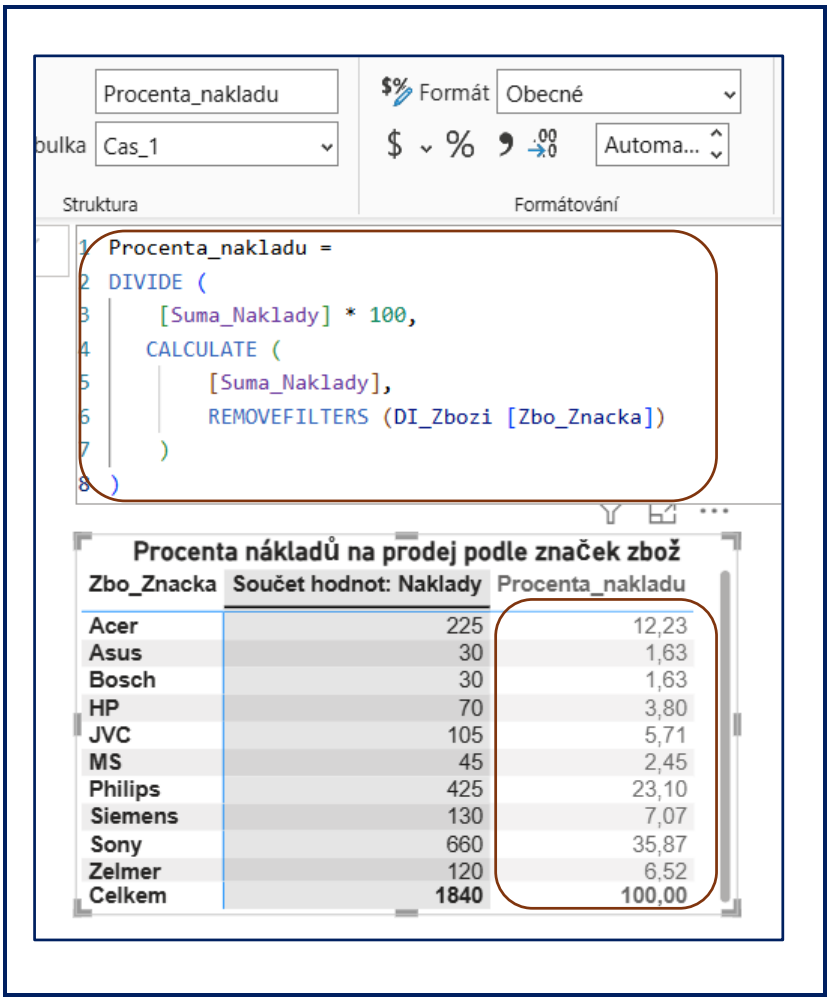
8.6 Funkce REMOVEFILTERS ()

Funkce *REMOVEFILTERS ()* řeší situace, kdy je třeba odstranit filtr bez náhrady nějakým jiným. To je v případě výpočtů procent, např. při zjišťování procentního podílu nákladů na celkovém objemu nákladů podle značek zboží, jak ukazuje další příklad:

```
Procenta_nakladu =
DIVIDE (
    [Suma_Naklady] * 100 ,
   CALCULATE (
       [Suma_Naklady] ,
       REMOVEFILTERS (DI_Zbozi [Zbo_Znacka]  )
    )
)
```

Příklad 8-13: Uplatnění funkce REMOVEFILTERS ()

Řešení dokumentuje Obrázek 8-11:



Obrázek 8-11: Funkce REMOVEFILTERS pro výpočet procent nákladů na zboží

REMOVEFILTERS () odstraňuje jakékoli filtry ze sloupce nebo z celé tabulky.

8.7 Filtrování jednoho sloupce

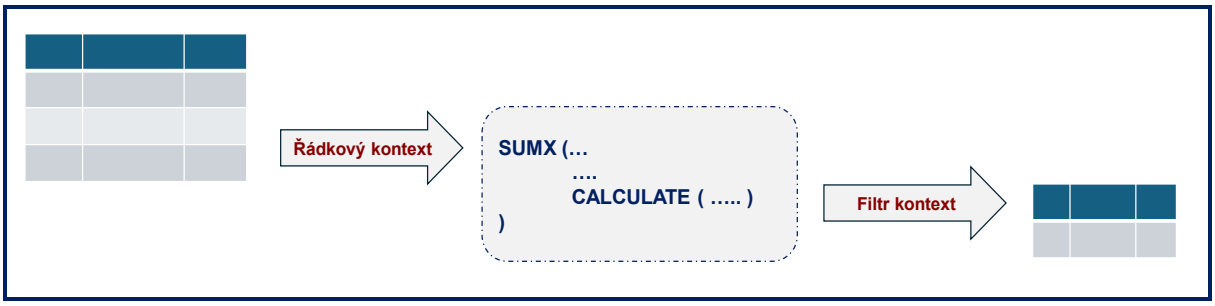
V souvislosti s filtrováním jednoho sloupce v rámci CALCULATE () je dobré připomenout, že v rámci jednoho výrazu lze použít k jednomu sloupci více podmínek, např. chceme zjistit celkový prodej v cenovém rozpětí 30–100, kde příkaz může vypadat takto:

```
Prodej_30_100 =
    CALCULATE (
      [Celkovy_Prodej],
      FTQ_Podej [Cena_Ks ]   >= 30 && FTQ_Podej [Cena_Ks ]   <= 100,
    )
```

Příklad 8-14: Uplatnění více podmínek k jednomu sloupci

8.8 Změna kontextu (Context transition)

Podstata změny kontextu je v tom, že funkce *CALCULATE()* může pracovat pouze s filtr kontextem. Tedy pokud výraz začíná funkcí odpovídající řádkovému kontextu, např. *SUMX()* následovaná funkcí *CALCULATE()*, pak pro *CALCULATE()* musí dojít ke změně řádkového kontextu na filtr kontext. To znamená, že vytváří filtr pro každý sloupec zpracovávané (iterované) tabulky. To ukazuje Obrázek 8-12.



Obrázek 8-12: Princip změny kontextu

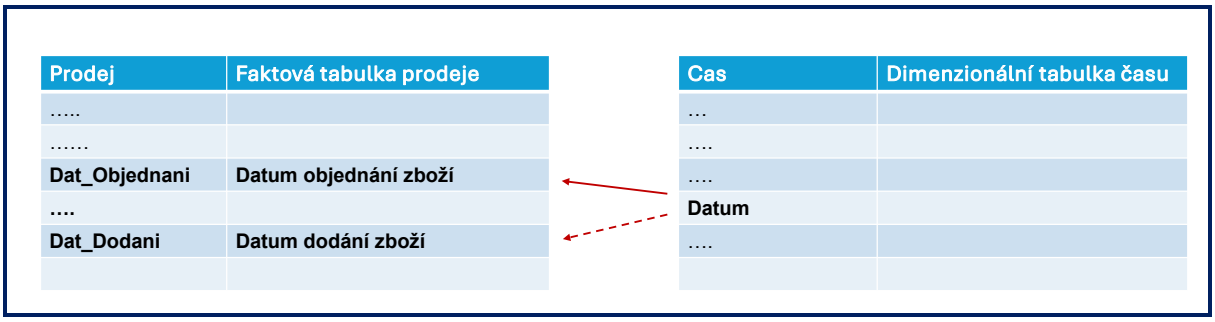
8.9 Modifikace funkce CALCULATE()

Funkce CALCULATE() je postavena vedle základních pravidel i na konceptu modifikátorů (*CALCULATE modifier*). K nim patří zejména:

- *USERELATIONSHIP*,
- *CROSSFILTER*,
- *ALL* (viz kapitola 8.3),
- *KEEPFILTERS* (viz kapitola 8.4).
- *ALLSELECTED*.

8.9.1 USERELATIONSHIP

V řešení sémantických modelů platí, že mezi dvěma tabulkami může být více vazeb, z toho ale pouze jedna z nich může být aktivní. Obvykle uváděným příkladem takové situace je vazba mezi faktovou tabulkou, např. prodeje a tabulkou času. V tabulce prodeje mohou být uvedeny čas objednání a čas dodání zboží. Oba údaje mohou představovat vazby k tabulce času. To schematicky ukazuje Obrázek 8-13:



Obrázek 8-13: Princip několika vazeb a USERELATIONSHIP

Z obrázku vyplývá, že vazba objednání zboží na „Datum“ v dimenzionální tabulce je aktivní, zatímco vazba dodání zboží na „Datum“ je neaktivní. Pokud je třeba řešit čas dodání zboží je třeba tuto vazbu aktivovat po dobu příslušného výpočtu, např.:

```
Objem_Dodaneho_Zbozi =
    CALCULATE(
      [Objem_Prodeje],
      USERELATIONSHIP ( Prodej [Dat_Dodani], Cas [Datum] )
)
```

Příklad 8-15: Uplatnění modifikátoru USERELATIONSHIP

Vazba *Prodej [Dat_Dodani] - Cas [Datum]* je aktivována pouze po dobu výpočtu, zatímco vazba *Prodej [Dat_Objednani] - Cas [Datum]* je deaktivována. To platí pouze po dobu výpočtu, po ukončení se opět aktivuje původní vazba.

USERELATIONSHIP() nevytváří nový filtr, nepředstavuje filtr argument. Je to modifikátor funkce *CALCULATE(),* pouze mění způsob, jak ostatní filtry v modelu budou použity. Platí přitom, že každý nejprve se využije modifikace a teprve potom jakýkoli filtr argument. To znamená, že filtr argumenty se provedou až na modifikované verzi modelu.

8.9.2 *CROSSFILTER*

Modifikátor *CROSSFILTER()* je obdobný jako *USERELATIONSHIP()*, ale zajišťuje dvě operace - může změnit směr (cross-filter) vazby mezi tabulkami a může deaktivovat vazbu. *CROSSFILTER()* pracuje se dvěma parametry určujícími sloupce ve vazbě a třetím, který může být *NONE, ONEWAY nebo BOTH*. Další příklad dokumentuje specifikuje použití obou směrné vazby mezi uvedenou faktovou a dimenzionální tabulkou:

```
…..
      CROSSFILTER ( FTQ_Prodej [Zbozi_Id], DI_Zbozi [Zbozi_Id], BOTH )
….
```

Příklad 8-16: Uplatnění modifikátoru *CROSSFILTER*

Obdobně jako *USERELATIONSHIP()* nevytváří nové filtry, pouze mění strukturu, resp. charakteristiky vztahů.

8.9.3 ALL

ALL() je primárně tabulkovou funkcí (podkapitola 9.4). Na druhé straně funguje jako modifikátor při použití jako filtr argument ve funkci *CALCULATE()*, s tím, že sémantika je v tomto případě odlišná. To znamená, že odstraňuje filtry, např.:

```
…..
CALCULATE (
    [Objem_Prrodeje],
   ALL DI_Zbozi [Zbo_Znacka] )
….
```

Příklad 8-17: Uplatnění modifikátoru *ALL*

V tomto případě ALL odstraňuje všechny filtry z daného filtr argumentu *CALCULATE()*.

8.9.4 ALLSELECTED

Obdobně modifikátor *ALLSELECTED()* nepůsobí zde jako tabulková funkce (podkapitola 9.4), ale jako modifikátor funkce filtr argumentu ve funkci *CALCULATE().* Obnovuje tak filtr kontext uvedeného sloupce na základě filtru pro aktuální vizuál, jak ukazuje příklad:

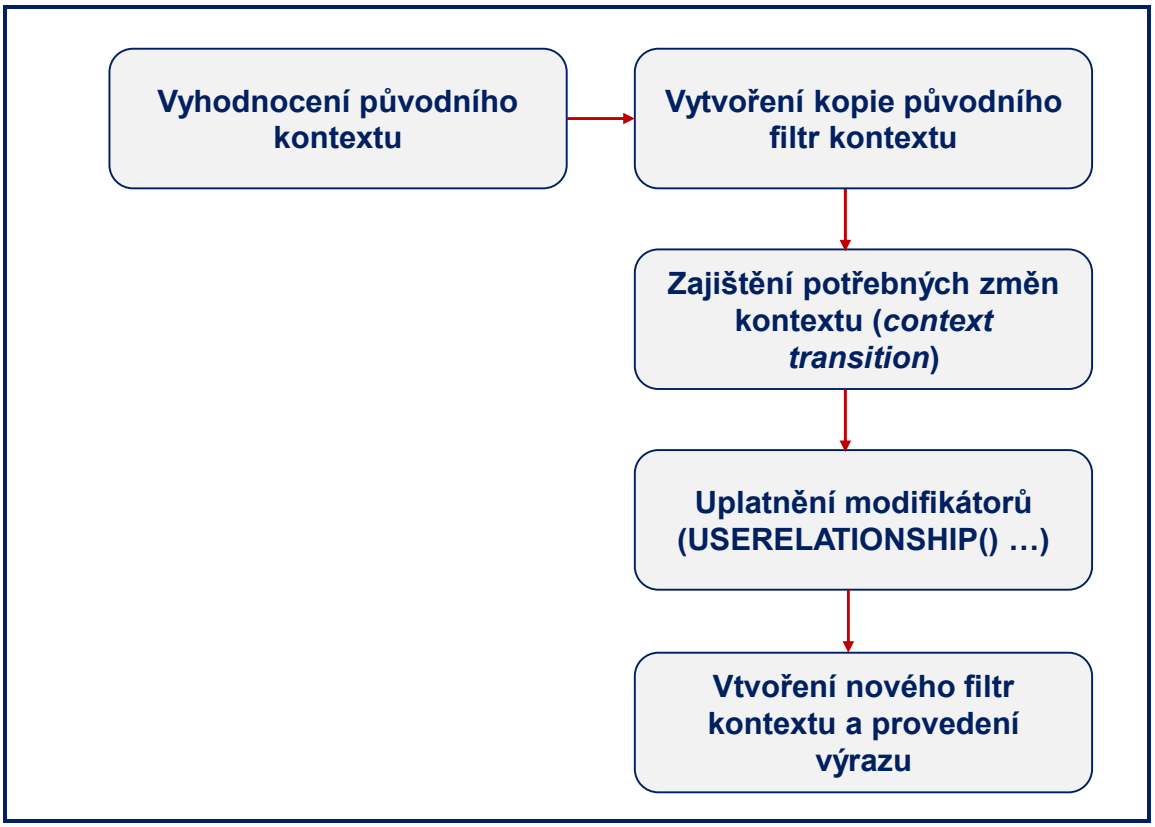
```
…..
CALCULATE (
    [Objem_Prrodeje],
   ALLSELECTED DI_Zbozi [Zbo_Znacka] )
….
 )
```

Příklad 8-18: Uplatnění modifikátoru *ALLSELECTED*

8.10 Postup realizace funkce CALCULATE()

Celkový postup realizace funkce *CALCULATE()* dokumentuje Obrázek 8-14 a zahrnuje:

- Převezme a vyhodnotí ***aktuální filtr kontext.***
- Z aktuálního filtr kontextu ***vytvoří kopii*** pro přípravu nového filtr kontextu.
- Provede potřebné ***změny filtr kontextu*** na základě principů „*context transition*“.
- Promítne modifikátory do filtr kontextu, jako např. *USERELATIONSHIP(), CROSSFILTER(), KEEPFILTERS()* atd.
- ***Vyhodnotí každou podmínku*** (tzv. filtr argument) v zadaných parametrech ***s uplatněním logického součinu***, tedy operátoru ***AND***.
- Vytvoří ***nový filtr kontext a provede výpočet výrazu*** a následně nový filtr kontext se zruší a jako aktivní se vrátí původní filtr kontext.



Obrázek 8-14: Postup realizace CALCULATE ()

8.11 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- *CALCULATE()* je to, že ***vytváří nový filtr kontext***, resp. umožňuje modifikovat *filtr kontext*, tedy zrušení a nastavení filtrů podle okamžité potřeby,
- V předpisu *CALCULATE()* je ***jediný povinný parametr***, a to ten první – *výraz míry,* další – podmínky, resp. ***filtr argumenty*** jsou nepovinné a jejich počet není omezen.
- Nový filtr kontext může obsahovat filtr kontext, pokud původně jsou součástí i řádkové kontexty převedou se na filtr kontext ***na bázi změny kontextu*** (*context transition*).
- *CALCULATE ()* pracuje ***se 3 druhy parametrů***:
  - ***Výraz***, který bude vyhodnocen v novém filtr kontextu.
  - ***Několik filtr argumentů***, které jsou základem pro vytvoření nového filtr kontextu. Každý filtr argument může být spojen s modifikátorem.
  - ***Sada modifikátorů***, které mohou měnit model nebo strukturu filtr kontextu.
- *CALCULATE ()* je funkce spojená s několika dalšími funkcemi, zejména:
  - *KEEPFILTERS* místo přepsání existujícího filtru na sloupci, pouze přidá k němu nový filtr.
  - VALUES () vrací hodnoty sloupce, které jsou aktuálně viditelné na základě filtr kontextu.
  - *REMOVEFILTERS ()* řeší situace, kdy je třeba odstranit filtr bez náhrady nějakým jiným.
- *CALCULATE ()* využívá tyto hlavní modifikátory:

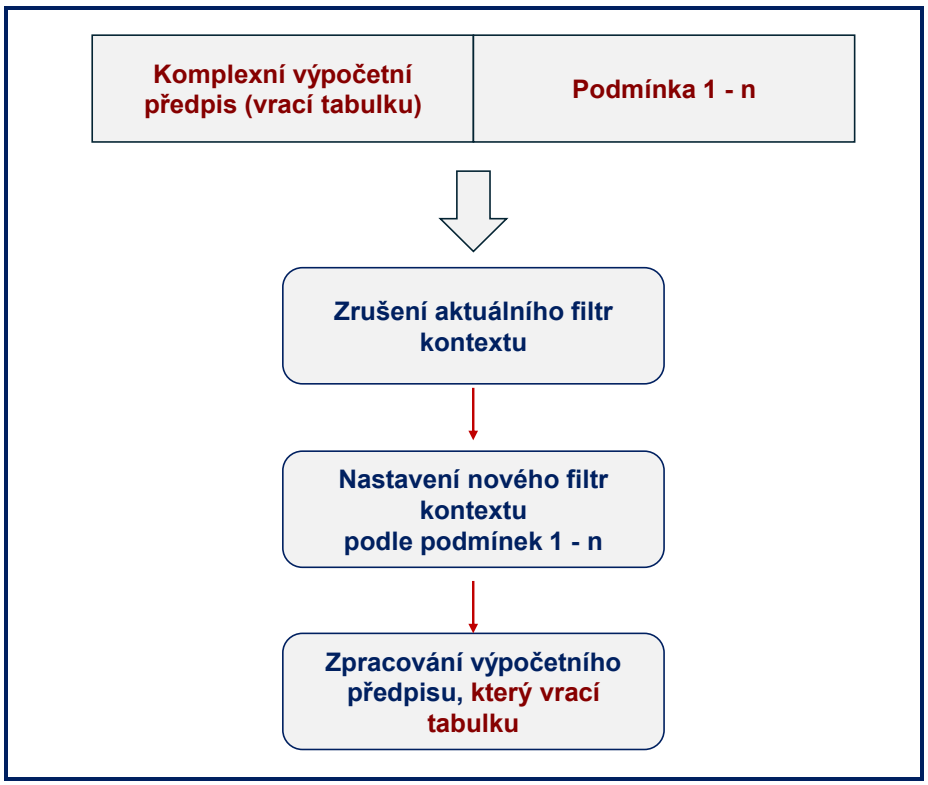
- *USERELATIONSHIP ()* – nevytváří nový filtr, nepředstavuje filtr argument, pouze mění způsob, jak ostatní filtry v modelu budou použity.
- *CROSSFILTER ()* je obdobný jako *USERELATIONSHIP ()*, ale zajišťuje dvě operace – může změnit směr (cross-filter) vazby mezi tabulkami a může deaktivovat vazbu.
- *ALL ()* – odstraňuje filtry.
- *ALLSELECTED ()* – obnovuje filtr kontext uvedeného sloupce na základě filtru pro aktuální vizuál.

9. Tabulkové funkce a funkce CALCULATETABLE()

Tabulkové funkce (*Table functions*) jsou standardní funkce DAX, které namísto jedné hodnoty vrací tabulku. Tabulkové funkce jsou využívané v dotazech DAXu a při pokročilých kalkulacích vyžadujících iterace přes několik tabulek. Základní funkcí je *CALCULATETABLE ()* a ***účelem*** této kapitoly je specifikovat její možnosti použití a uvést i několik adekvátních příkladů.

(Zdroj: Ferrari, Russo, 2026).

Jak již bylo uvedeno výrazy, které vracejí jednu hodnotu se označují jako skalární výrazy, v nichž se pro výpočty používají obvykle tabulky. Vedle těchto výrazu se pracuje s výrazy vracející tabulku jako finální výsledek nebo mezivýsledek složitější kalkulace. Základní princip funkce CALCULATETABLE() dokumentuje Obrázek 9-1. Podobně jako funkce CALCULATE(), která vrací skalární hodnotu výrazu, ***funkce CALCULATETABLE() modifikuje filtr kontext výrazu, ale vrací tabulku***.

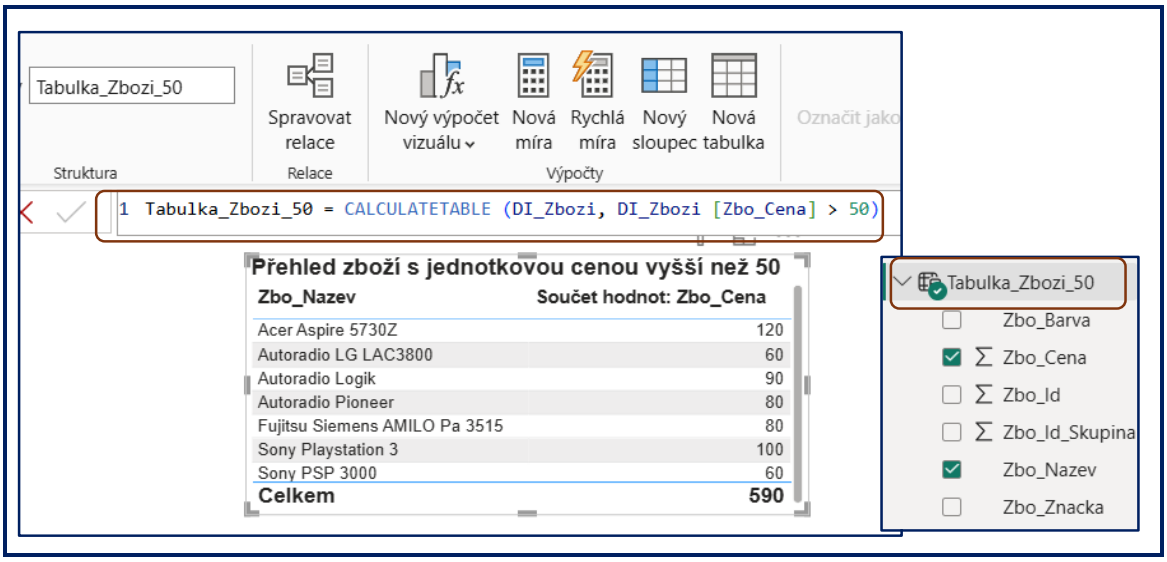


Obrázek 9-1: Princip CALCULATETABLE ()

Uvedený princip dokumentuje i další příklad, kde funkce vrací tabulku zboží pouze s řádky, kde cena je vyšší než 50,-:

```
Tabulka_Zbozi_50 = CALCULATETABLE (DI_Zbozi, DI_Zbozi [Zbo_Cena] > 50)
```

Příklad 9-1: Funkce CALCULATETABLE() vrací tabulku zboží s cenou vyšší než 1 500,-

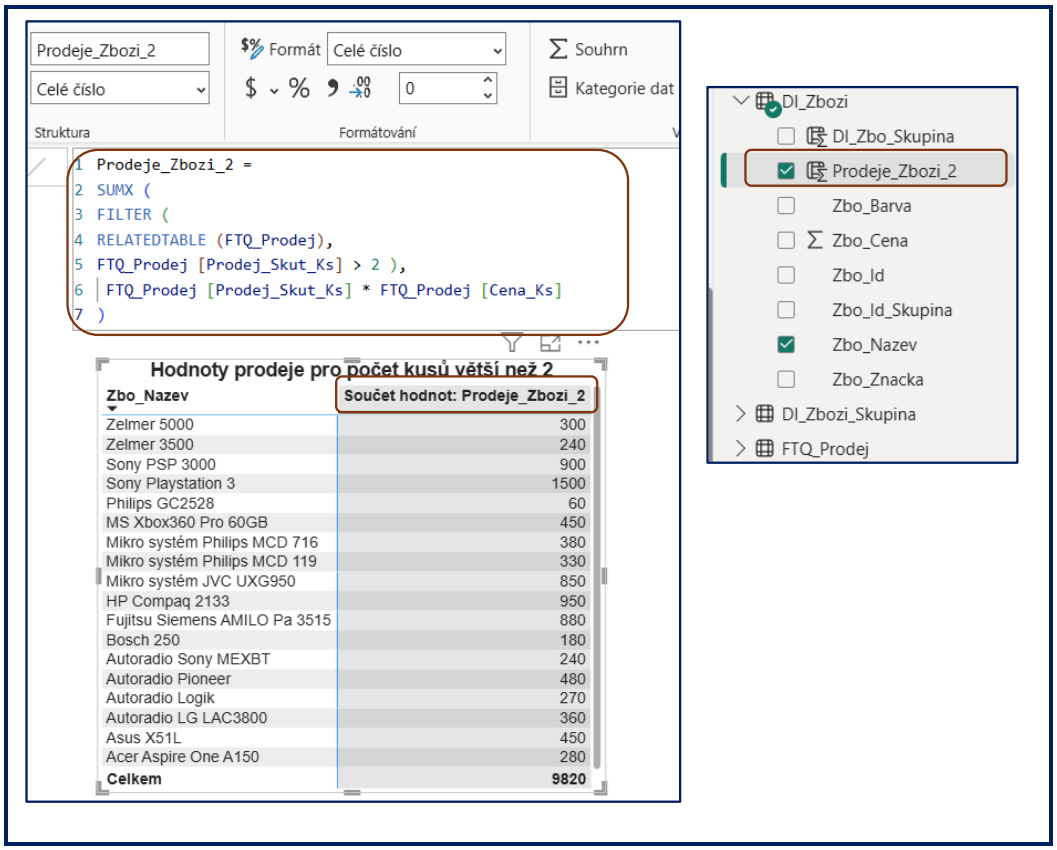


Obrázek 9-2: Využití funkce CALCULATETABLE() pro novou tabulku zboží

V kalkulovaném sloupci nebo v iteraci lze využít funkce *RELATEDTABLE* pro získání všech řádků provázané tabulky. V tabulce *DI_Zbozi* máme vytvořit kalkulovaný sloupec s hodnotami prodeje položek zboží pro počet kusů vyššími než 2. Zápis je následující:

```
Prodeje_Zbozi_2  =
SUMX (
        FILTER (
                RELATEDTABLE (FTQ_Prodej),
                FTQ_Prodej [Prodej_Skut_Ks] > 2 ),
        FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks]
)
```

Příklad 9-2: Kalkulovaný sloupec s hodnotami prodeje vyššími pro počet kusů větší než 2



Obrázek 9-3: Hodnoty prodeje pro počet kusů větší než 2

V tomto smyslu nelze použít výsledek tabulkové funkce jako hodnotu míry nebo kalkulovaného sloupce, neboť ty předpokládají výraz jako skalární hodnotu. Na druhé straně ale se přiřazuje výsledek tabulkové funkce kalkulované tabulce.

9.1 Funkce EVALUATE

Funkce *EVALUATE* zajišťuje ***zobrazení výsledků tabulkového výrazu***. Dotaz v DAXu (*DAX Query*) je výraz v DAXu, který vrací tabulku využitou ve funkci EVALUATE, která má relativně komplexní syntaxi:

Syntax:

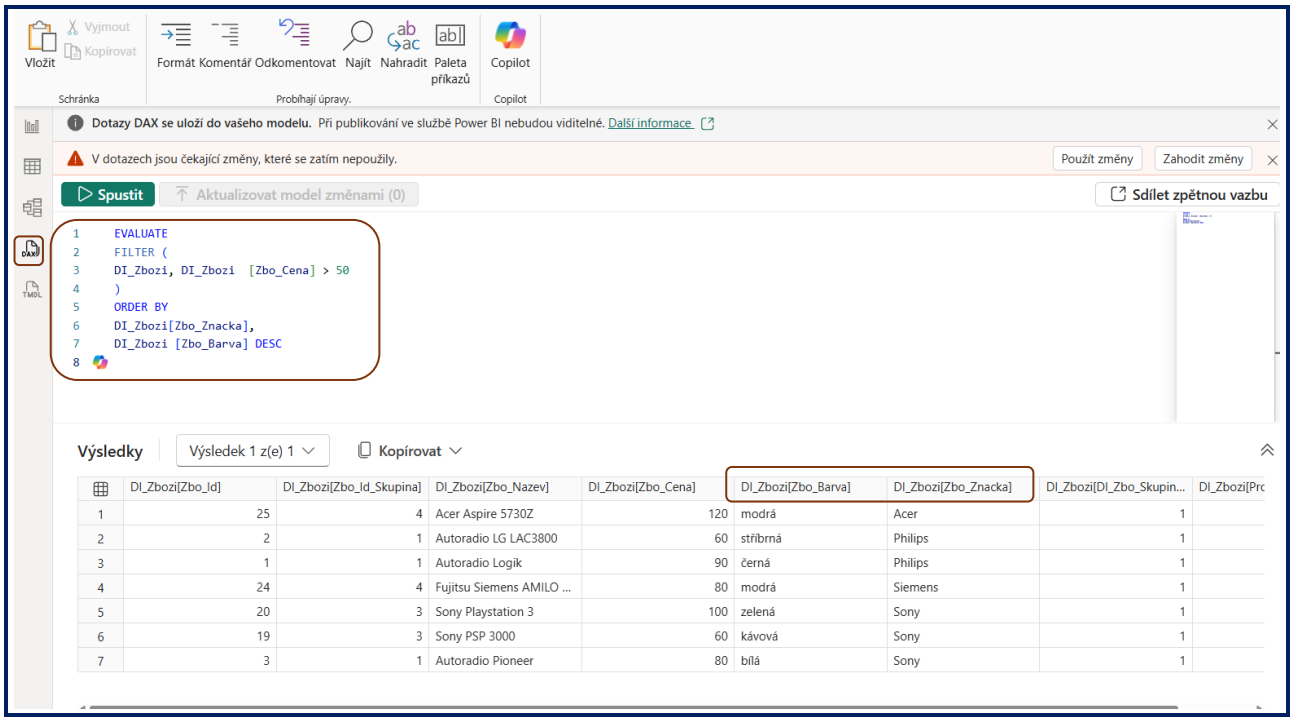
[DEFINE { MEASURE < jméno_tabulky > [< jméno >] = < výraz > }]
EVALUATE < jméno_tabulky >
[ORDER BY { < výraz > [{ ASC | DESC }] } [, ….]]

Jak je patrné většina částí syntaxe je volitelná. Příklad relativně jednoduchého využití EVALUATE je uveden následně, kde se zobrazí přehled položek zboží, jejichž cena je vyšší než 50, a je setříděný podle typu zboží vzestupně a dále podle barvy sestupně, jak ukazuje Příklad 9-3 a Obrázek 9-4

EVALUATE
FILTER (
´DI_Zbozi´, ´DI_Zbozi´ [Zbo_Cena] > 50
)

```
ORDER BY
        ´DI_Zbozi´ [Zbo_Typ],
        ´DI_Zbozi´ [Zbo_Barva] DESC
```

Příklad 9-3: EVALUATE zobrazí přehled položek zboží, jejichž cena je vyšší než 50



Obrázek 9-4: Zobrazení dotazu na bázi funkce EVALUATE

9.2 Funkce FILTER

Další z podstatných tabulkových funkcí Je *FILTER*, její ***syntax*** je následující.

```
FILTER ( < jméno_tabulky >, < podmínka > )
```

FILTER zajišťuje

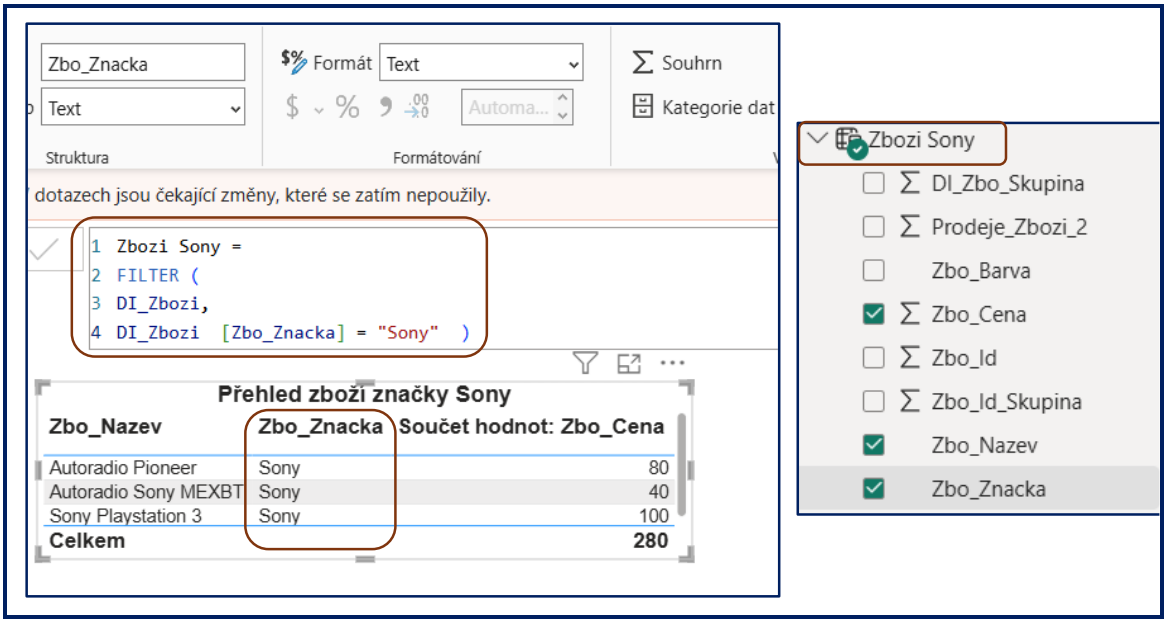
- jako výsledek vrací všechny řádky tabulky, které odpovídají zadané podmínce,
- chová se současně jako tabulková funkce a iterátor,
- vyhodnocuje tabulku řádku po řádce vzhledem k podmínce, tedy iteruje tabulku.

9.2.1 Další příklady:

```
Zbozi Sony =
FILTER (
        DI_Zbozi,
```

```
        DI_Zbozi  [Zbo_Znacka] = „Sony“
)
```

Příklad 9-4: Kalkulovaná tabulka s přehledem zboží značky „Sony“

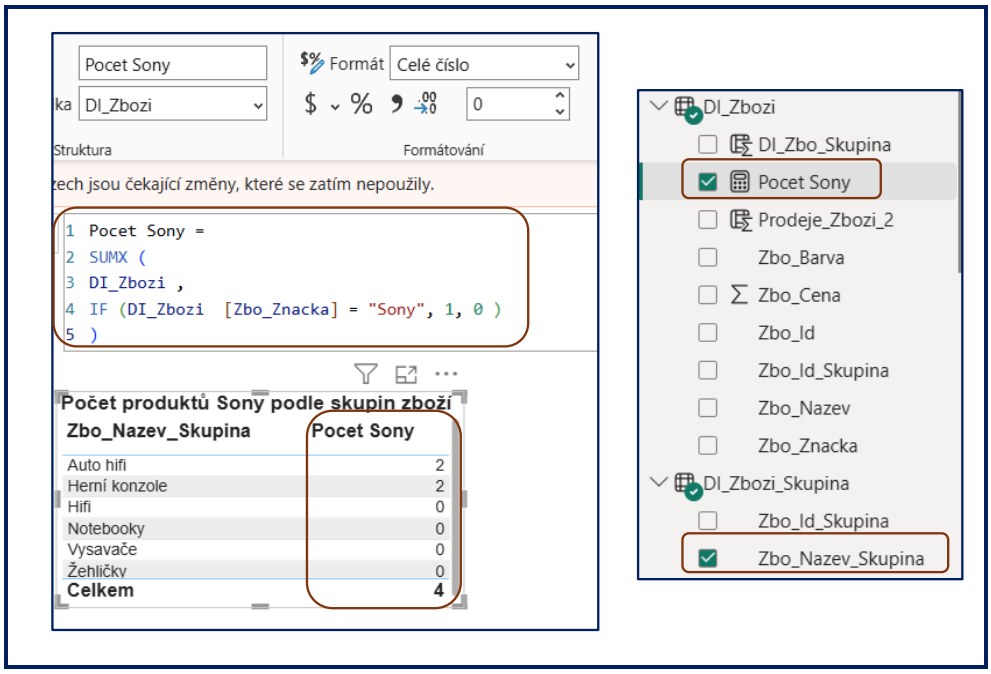


Obrázek 9-5: Kalkulovaná tabulka se zbožím značky Sony

Potřebujeme opět zjistit počet produktů typu Samsung, možností je využití funkce *IF()* v rámci iterátoru *SUMX(),* jak ukazuje Příklad 9-5. Vnitřní funkce IF() vrací hodnotu 1 nebo 0 podle toho, zda jde o typ Sony a vrací počet zboží tohoto typu.

```
Pocet Sony :=
SUMX (
        DI_Zbozi ,
        IF (DI_Zbozi  [Zbo_Znacka] = „Sony“, 1, 0 )
)
```

Příklad 9-5: Využití funkce IF() při zjišťování počtu zboží typu Sony

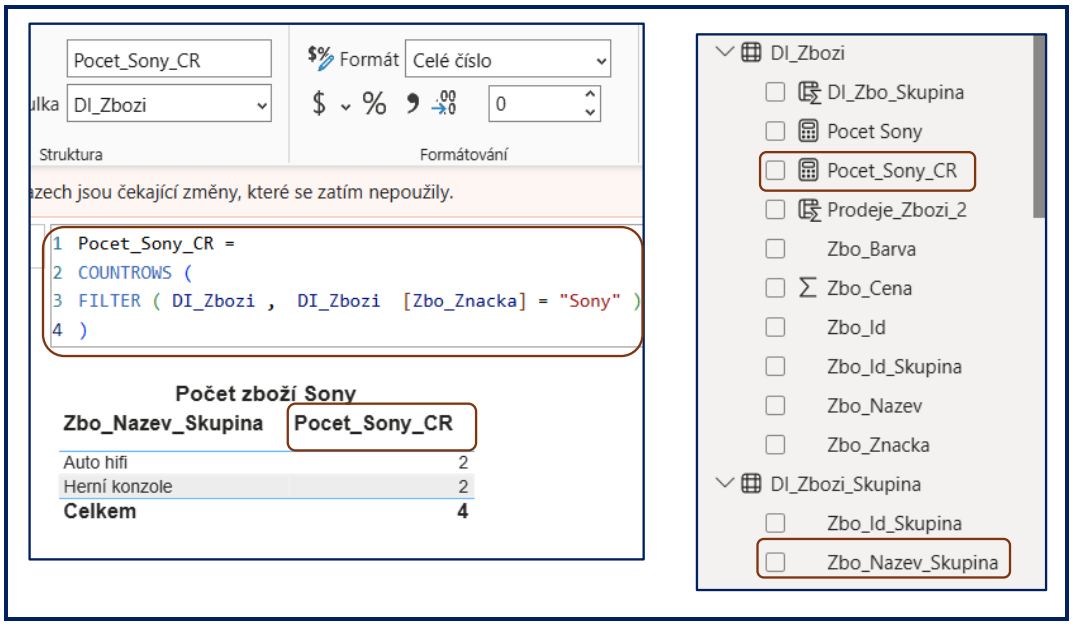


Obrázek 9-6: Počet produktů Sony podle skupin zboží

Lepší možností implementace předchozího případu je využití funkce *COUNTROWS()* ve spojení s funkcí *FILTER()*.

```
Pocet_Sony :=
COUNTROWS (
        FILTER ( DI_Zbozi ,  DI_Zbozi  [Zbo_Znacka] = „Sony“ )
)
```

Příklad 9-6: Využití funkce *COUNTROWS()* při zjišťování počtu zboží typu Sony

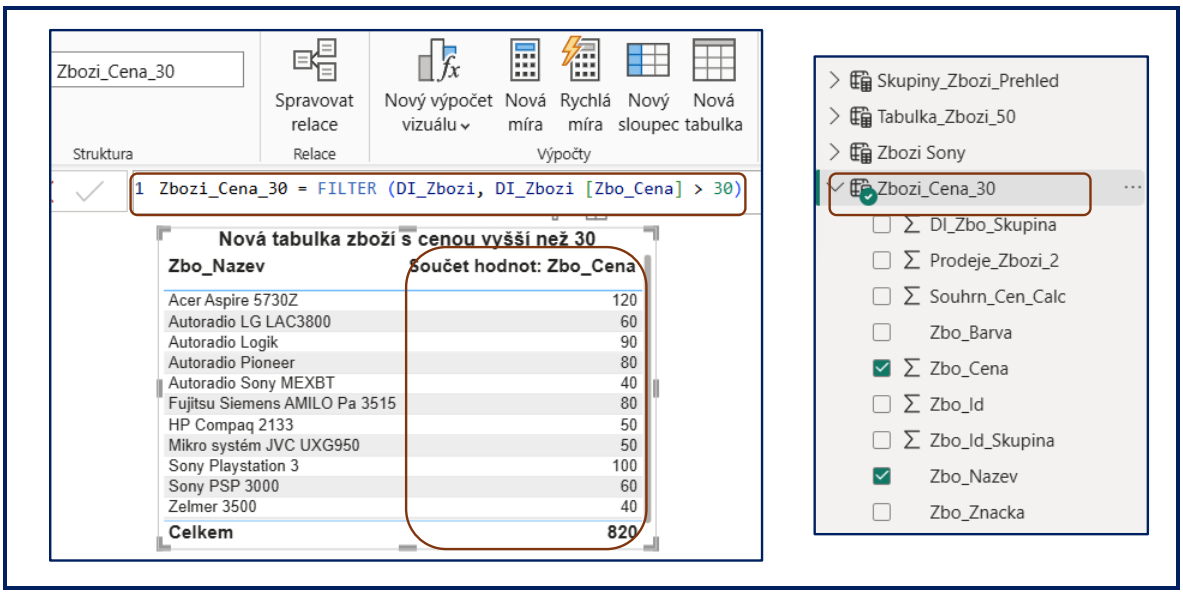


Obrázek 9-7: Počet zboží s využitím funkce COUNTROWS

Dalším příkladem je vytvoření nové tabulky na bázi dimenzionální tabulky *DI_Zbozi* se souhrnem ceníkových cen vyšším než 30, jak dokumentuje Příklad 9-8 a

```
Zbozi_Cena_30 :=
        FILTER ( DI_Zbozi ,  DI_Zbozi  [Zbo_Cena] > 30 )
```

Příklad 9-7: Vytvoření nové tabulky zboží s cenou vyšší než 30

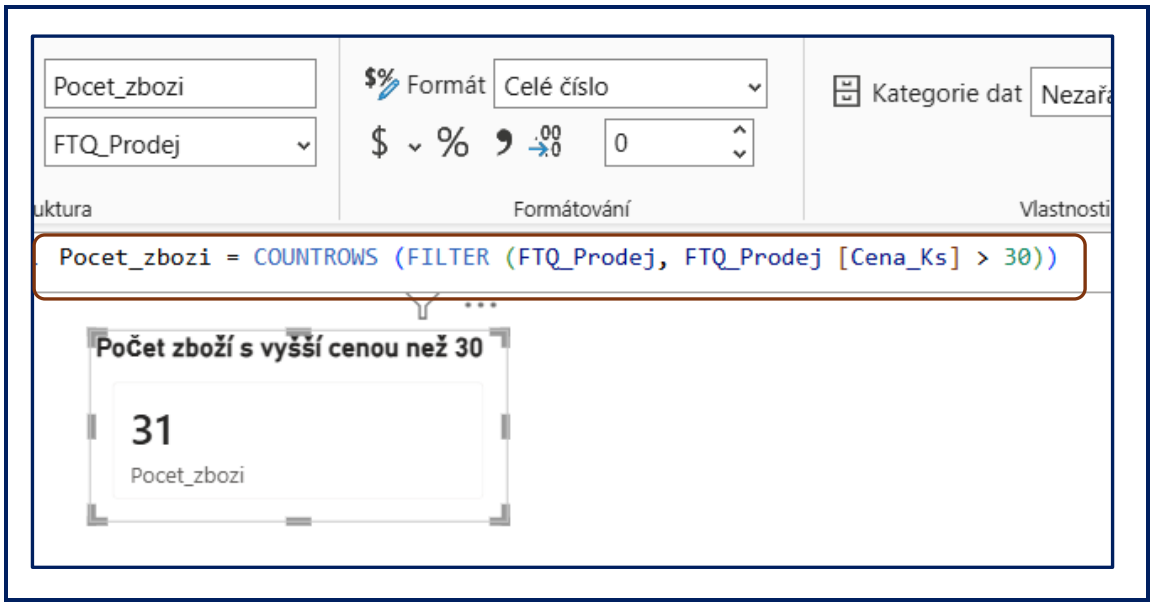


Obrázek 9-8: Vytvoření nové tabulky se součtem ceníkových cen vyšším než 30

Další příklad dokumentuje využití funkcí *COUNTROWS()* a *FILTER()* pro zjištění počtu zboží s cenou za kus vyšší než 30 ve faktové tabulce *FTQ_Prodej.*

```
Počet_zbozi =
        COUNTROWS ( FILTER (FTQ_Prodej ,  FTQ_Prodej [Cena_Ks] > 30 )
```

Příklad 9-8: Počet zboží s cenou za kus vyšší než 30 ve faktové tabulce



Obrázek 9-9: Zjištění počtu zboží s cenou vyšší než 30 ve faktové tabulce

9.3 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 *Dl_Zbozi_Kat* (kategorie zboží), přes *Dl_Zbozi_Skupina* – *Dl_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*):

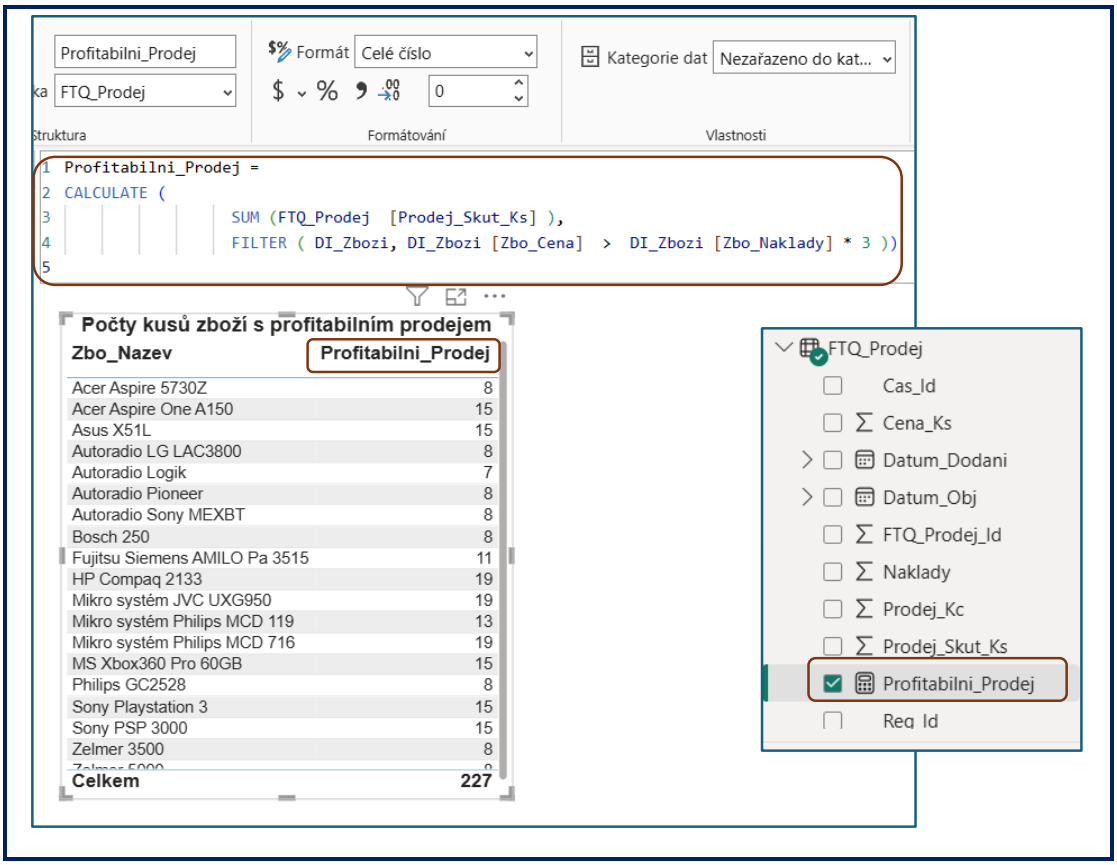
```
Profitabilni_Prodej =
        CALCULATE (
           SUM (FTQ_Prodej  [Prodej_Skut_Ks] ),
           Dl_Zbozi [Zbo_Cena]  >  Dl_Zbozi [Zbo_Naklady] * 3 )
```

Příklad 9-9: Chyba: podmínka pracuje se 2 sloupci najednou.

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 ( Dl_Zbozi, Dl_Zbozi [Zbo_Cena]  >  Dl_Zbozi [Zbo_Naklady] * 3 ))
```

Příklad 9-10: Použití seznamu hodnot a funkce FILTER



Obrázek 9-10: Funkce CALCULATE() s využitím FILTER

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.

9.4 Funkce ALL, ALLEXCEPT

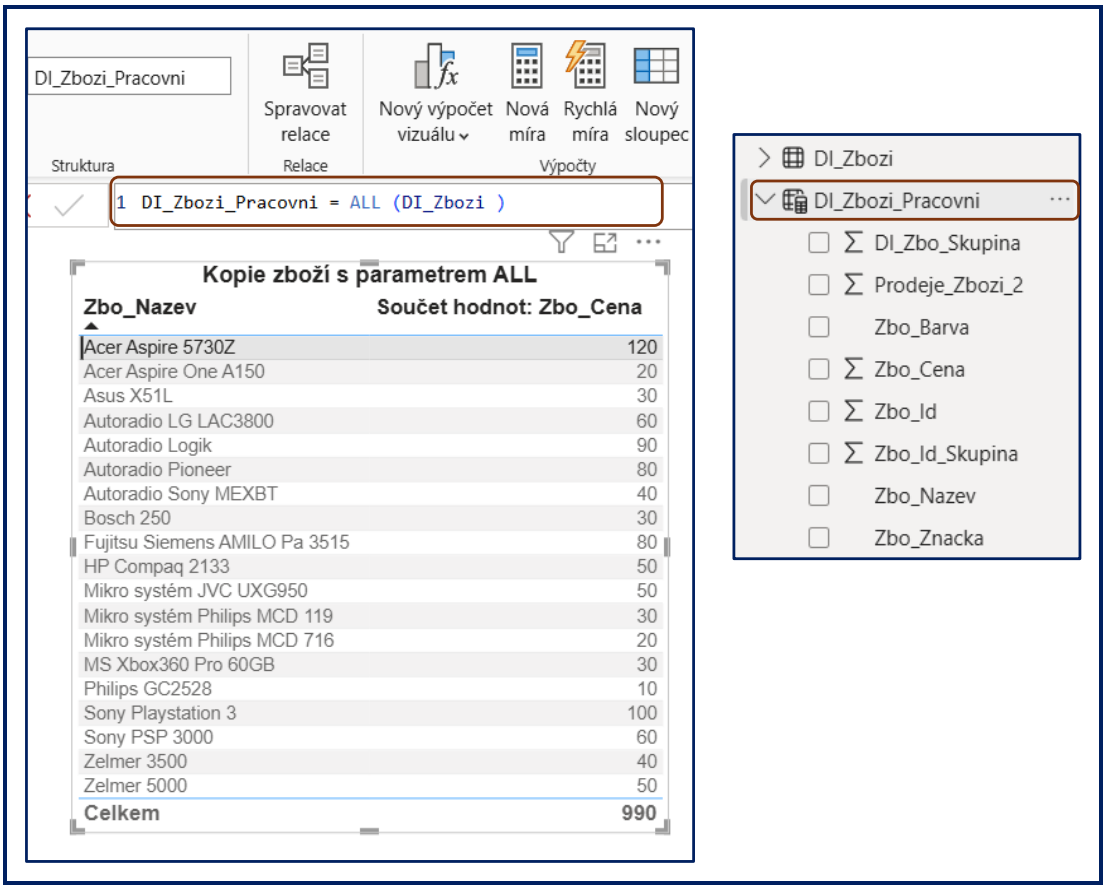
Pro ***uvolnění nastavených filtrů*** pro další operace slouží několik funkcí, zejména ALL, ALLEXCEPT, ALLSELECTED, ALLNOBLANKROW, ALLCROSSFILTERED. V této podkapitole se zaměříme na pouze na 2 uvedené v názvu.

Funkce ALL vrací všechny řádky tabulky nebo všechny hodnoty jednoho nebo více sloupců, ruší nastavené filtry, v závislosti na parametrech. Je použitelná při řešení kalkulovaných měr, zejména při kalkulacích procent nebo poměrů, protože ruší všechny filtry nastavené v reportu.

9.4.1 Další příklady:

DI_Zbozi_Pracovni = ***ALL*** *(DI_Zbozi)*

Příklad 9-11: Vytvoření kopie tabulky „Zboží“



Obrázek 9-11: Vytvoření kopie tabulky s parametrem ALL

Pokud se funkce *ALL* použije na tabulku s jedním sloupcem, pak vrací tabulku pouze unikátními hodnotami.

V případě, že ***se zjišťují procentní podíly prodeje zboží***, tedy objem prodeje vzhledem k celkovému prodeji, je třeba nejprve zjistit jednotlivé objemy prodeje podle druhů zboží (Příklad 9-12), následně celkový objem prodeje s vyloučením filtrů (Příklad 9-13), a potom lze provést výpočet podílů prodeje (Příklad 9-14).

```
Objemy_Prodeje_Zbozi =
SUMX (
        FTQ_Prodej,
        FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks]
)
```

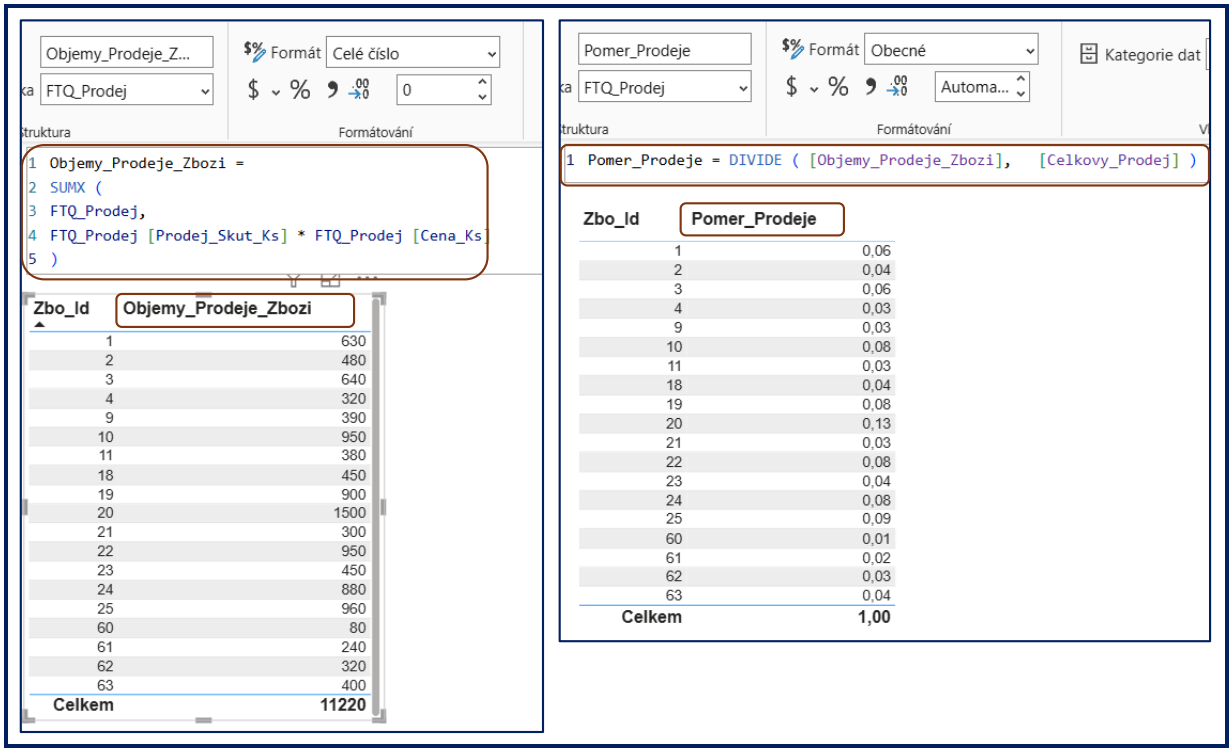
Příklad 9-12: Výpočet jednotlivých objemů prodeje podle druhů zboží

```
Celkovy_Prodej :=
SUMX (
        ALL ( FTQ_Prodej ),
        FTQ_Prodej [Prodej_Skut_Ks] * FTQ_Prodej [Cena_Ks]
)
```

Příklad 9-13: Výpočet celkové hodnoty prodeje zboží s vyloučením filtrů

```
Pomer_Prodeje := DIVIDE ( [Objemy_Prodeje_Zbozi],   [Celkovy_Prodej] )
```

Příklad 9-14: Výpočet poměru jednotlivých hodnot prodeje k celkovému objemu

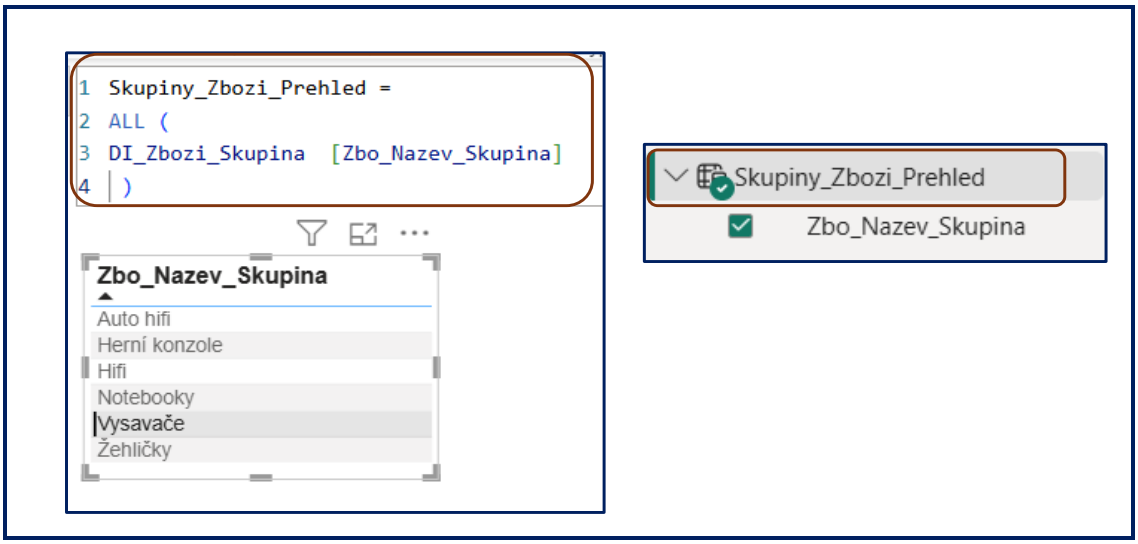


Obrázek 9-12: Výpočet poměrů prodeje jednotlivých zboží na celkovém prodeji

V následujícím příkladu je třeba získat celkový přehled skupin zboží, tedy s vyloučením filtrů:

```
Skupiny_Zbozi_Prehled =
ALL (
        DI_Zbozi_Skupina  [Zbo_Nazev_Skupina]
)
```

Příklad 9-15: Vytvoření přehledu skupin zboží



Obrázek 9-13: Přehled skupin zboží s paramerem ALL

Pokud je třeba do operace, resp. výpočtu zahrnout část ale ne všechny sloupce tabulky, využívá se funkce *ALLEXCEPT*. V zápisu funkce se uvádí tabulka s přehledem sloupců, které je třeba z výpočtu vyřadit. *ALLEXCEPT()* vrací tabulku s přehledem kombinací hodnot ze zbývajících sloupců tabulky:

```
ALLEXCEPT ( DI_Zbozi´, DI_Zbozi [Zbo_Cena], DI_Zbozi [Zbo_Znacka], )
```

Příklad 9-16: Vytvoření tabulky „Zboží“ s vyloučením ceny a typu zboží

9.5 VALUES a DISTINCT funkce

Pokud se funkce *ALL* použije na tabulku s jedním sloupce, pak vrací tabulku pouze unikátními hodnotami. Obdobně další dvě funkce vracejí unikátní hodnoty, a to *VALUES* a *DISTINCT*. Rozdíl mezi nimi je v tom, jak nakládají s prázdnými řádkami v tabulce.

VALUES vrací v případě míry počet pouze unikátních viditelných hodnot s respektováním aktuálního filtru včetně prázdné řádky. V případě kalkulovaného sloupce nebo kalkulované tabulky se filtr standardně neuplatňuje:

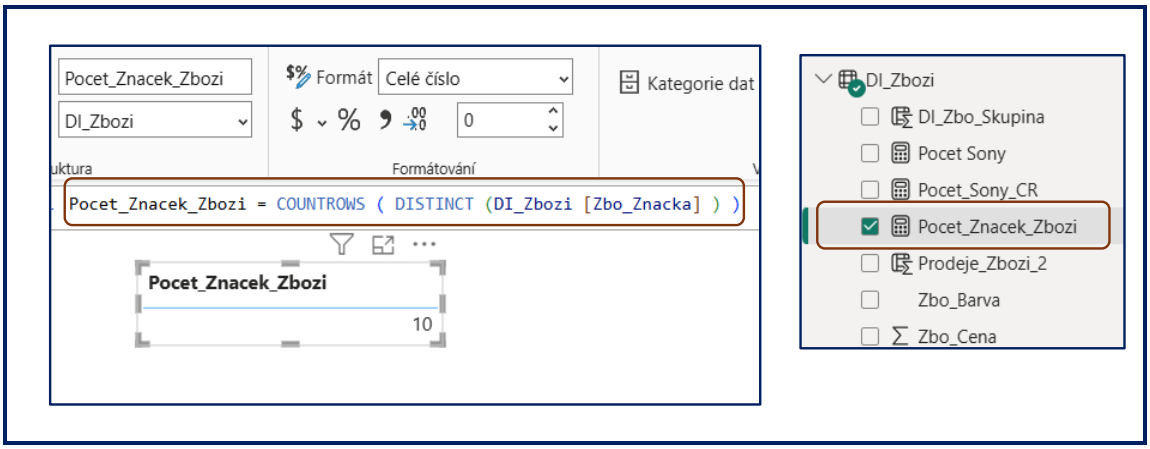
```
Pocet_Znacek_Zbozi =  COUNTROWS ( VALUES (DI_Zbozi [Zbo_Znacka] ) )
```

Příklad 9-17: Výpočet počtu řádek unikátních značek zboží včetně prázdných řádek

Funkce *DISTINCT* funguje obdobně jako VALUES, ale do výpočtu nezařazuje prázdné řádky (které mohou vzniknout při chybách ve vazbách mezi tabulkami):

```
Pocet_Znacek_Zbozi :=  COUNTROWS ( DISTINCT (DI_Zbozi [Zbo_Znacka]  ))
```

Příklad 9-18: Výpočet počtu řádek unikátních značek zboží bez prázdných řádek



Obrázek 9-14: Počet řádek unikátních značek zboží bez prázdných řádek

9.6 Funkce ALLSELECTED

ALLSELECTED je velmi komplexní tabulková funkce a využívá se, pokud je třeba získat seznam hodnot tabulky nebo sloupce, které jsou viditelné v daném reportu a respektují filtry mimo daný vizuál. Příkladem může být aktuálně výpočet poměrů objemu prodeje zboží nikoli vzhledem k celkovému prodeji (jako v případě *ALL*).

9.7 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- V souvislosti s tabulkovými funkcemi se obvykle využívají funkce *FILTER*, *ALL*, *ALLEXCEPT*, *VALUES*, *DISTINCT* a ALLSELECTED.
- Tabulkové funkce vrací obvykle modifikované tabulky oproti původní.
- *FILTER* jako výsledek vrací všechny řádky tabulky, které odpovídají zadané podmínce.
- Funkce ALL vrací všechny řádky tabulky nebo všechny hodnoty jednoho nebo více sloupců, ruší nastavené filtry.
- *VALUES* a *DISTINCT* vrací tabulku pouze unikátními hodnotami.
- *ALLSELECTED* se užívá, pokud je třeba získat seznam hodnot, které jsou viditelné v daném reportu a respektují filtry mimo daný vizuál.

10. Funkce IF()

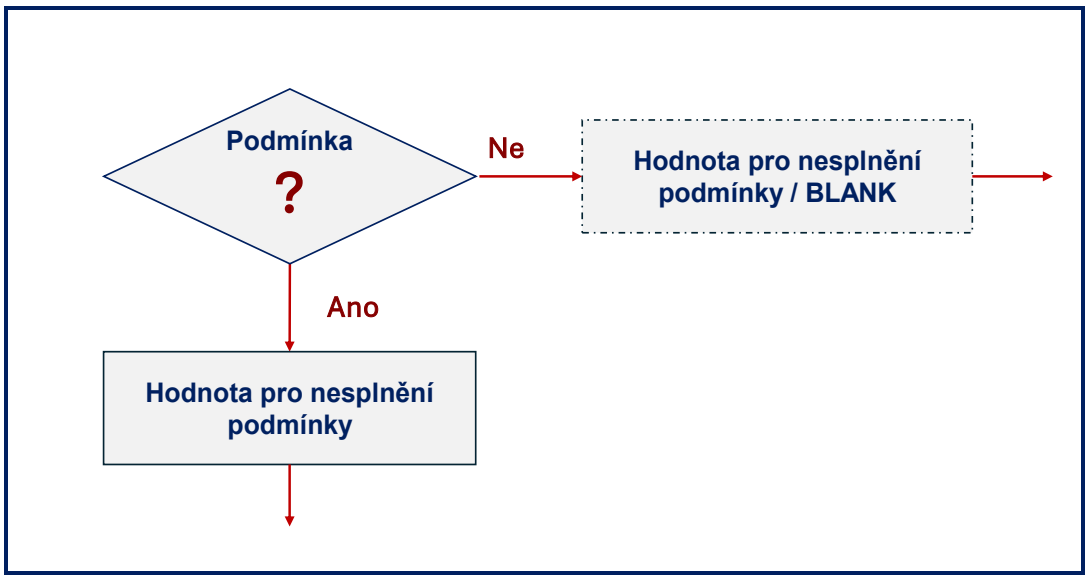
Funkce IF() patří, jako u většiny jazyků, k základním funkcím, resp. příkazům pro určování podmínek a jim odpovídajících akcí nebo cest. ***Účelem*** kapitoly je vymezit funkce IF() a ukázat možnosti využití.

(Zdroj: Ferrari, Russo, 2026).

Jak v úvodu je zmíněno, IF() patří ke standardním funkcím programovacích prostředků. Její ***syntaxe*** je následující:

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

Funkce IF() obsahuje 3 parametry, z nichž první dva jsou povinné, třetí je nepovinný a default hodnota je BLANK(). Přestože jde o běžnou funkci i její princip dokumentuje Obrázek 10-1:



Obrázek 10-1: Princip funkce IF ()

10.1 Příklady 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 ()*.

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

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

Příklad 10-1: Funkce IF () výpočtu marže na jednu transakci

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

10.2 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Funkce IF() je jednou ze základních funkcí sloužících pro větvení algoritmu.
- Funkce IF() obsahuje 3 parametry, z nichž první dva jsou povinné, třetí je nepovinný.
- Třetí nepovinný parametr představuje při neuplatnění default hodnotu BLANK().

11. Iterátory

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.

Účelem této kapitoly je specifikovat možnosti iterátorů a jejich využití v kalkulacích DAXu.

(Zdroj: Ferrari, Russo, 2026).

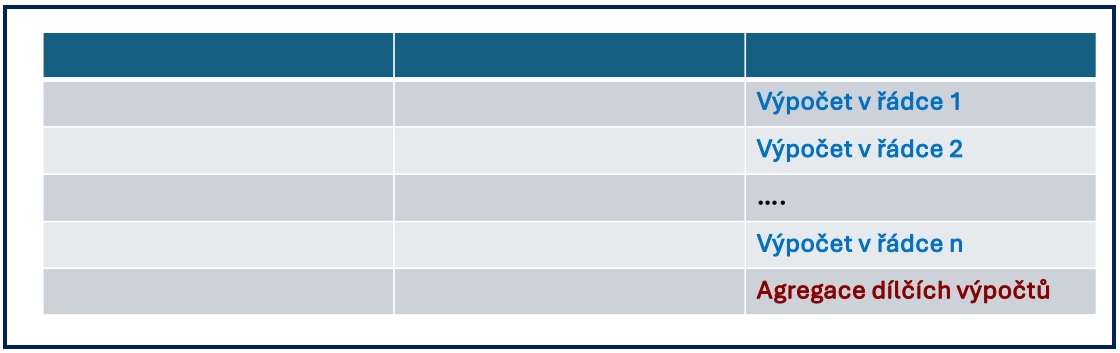
Iterátory mají vždy minimálně ***dva parametry***:

- první parametr identifikuje tabulku, která má být předmětem iterace,
- druhý parametr specifikuje výraz, který má být vyhodnocován na každé řádce.

K tomu ***několik poznámek***:

- Na závěr iterátor agreguje dílčí výpočty podle typu iterátoru.
- Pro iterátory je charakteristické, že jsou relativně rychlé.
- Základní funkce, jako např. SUM(), jsou pouze zkráceným zápisem funkce iterátoru, tedy SUMX(). Provádějí se ale naprosto stejně.

Způsob provedení iterátorů dokumentuje Obrázek 11-1:



Obrázek 11-1: Princip iterátorů

Pro další vysvětlení zopakujeme následující příklad:

```
Zvyseny_Prodej =SUMX (FTQ_Prodej, ([Prodej_Kc] * 1.05))
```

Příklad 11-1: Využití řádkového kontextu s použitím iterátorů

SUMX zajistí, že v každé řádce tabulky se položka náklady zvýší o 5 % a nakonec vrátí souhrn těchto přepočítaných hodnot (viz Obrázek 6-10). 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.
4) Některé iterátory, jako např. RANKX() používají více než dva výše uvedené parametry.

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.
- …

V dalším textu se zaměříme na některé podstatné charakteristiky a častější použití iterátorů.

11.1 Kardinalita iterátorů

Jeden z podstatných konceptů spojených s iterátory je stanovení jejich kardinality. Ta představuje ***počet řádek, které mají být iterovány***. Pokud např. v příkladu (Příklad 11-1) je počet řádek tabulky *FTQ_Prodej* je 1 000, pak kardinalita daného iterátoru je 1 000. Při vyjádření kardinality se však méně používá hodnota (1 000), ale spíše forma, že jde o „*kardinalitu faktové tabulky FTQ_Prodej“*. V každém případě informace o kardinalitě slouží pro řešení výkonnosti dotazů.

Ke kardinalitě doplníme další poznámku:

- V případě, že jde o vnořené (nestované) iterátory, pak kardinalita představuje kombinaci, resp. součin obou nebo více kardinalit, jako např.:

```
…
=SUMX (
        DI_Zbozi,
        SUMX (
                RELATEDTABLE (FTQ_Prodej ),
                ….
        )
)
```

Příklad 11-2: Kardinalita vnořených iterátorů (Zdroj: Ferrari, Russo, 2026, upraveno)

Pokud jde o vnější iterátor *DI_Zbozi*, pak jde o kardinalitu dimenzionální tabulky DI_Zbozi. Vnitřní iterátor *FTQ_Prodej* iteruje pouze ty řádky, které mají vazbu na příslušné zboží. Tomu odpovídá i kardinalita ve vztahu k tabulce *FTQ_Prodej.*

Při použití iterátorů je potřeba vždy určit tyto charakteristiky v následujícím pořadí:

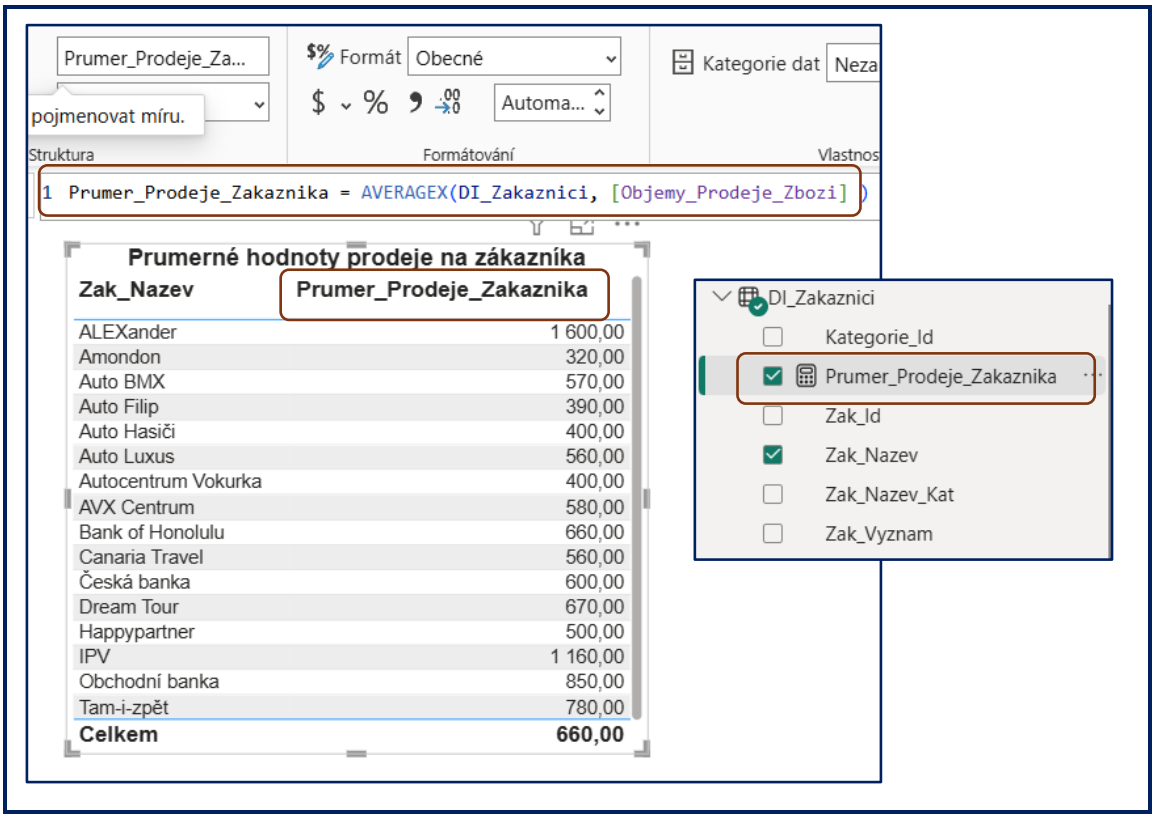
1) granularitu dat, na které se mají výpočty realizovat,
2) výpočetní výraz, který se má na dané granularitě provádět,
3) způsob použité agregace.

Pro dokumentaci použijeme následující příklad:

```
Prumer_Prodeje_Zakaznika :=
AVERAGEX (DI_Zakaznici, [ Objemy_Prodeje_Zbozi ] )
```

Příklad 11-3: Průměr objemu prodeje zboží na zákazníka

V daném případě je granularita jednotlivý „*zákazník*“, výpočetní výraz je „*objem prodeje zboží*“ a způsob agregace se průměr. Řešení ukazuje Obrázek 11-2:



Obrázek 11-2: Průměr objemu prodeje na zákazníka

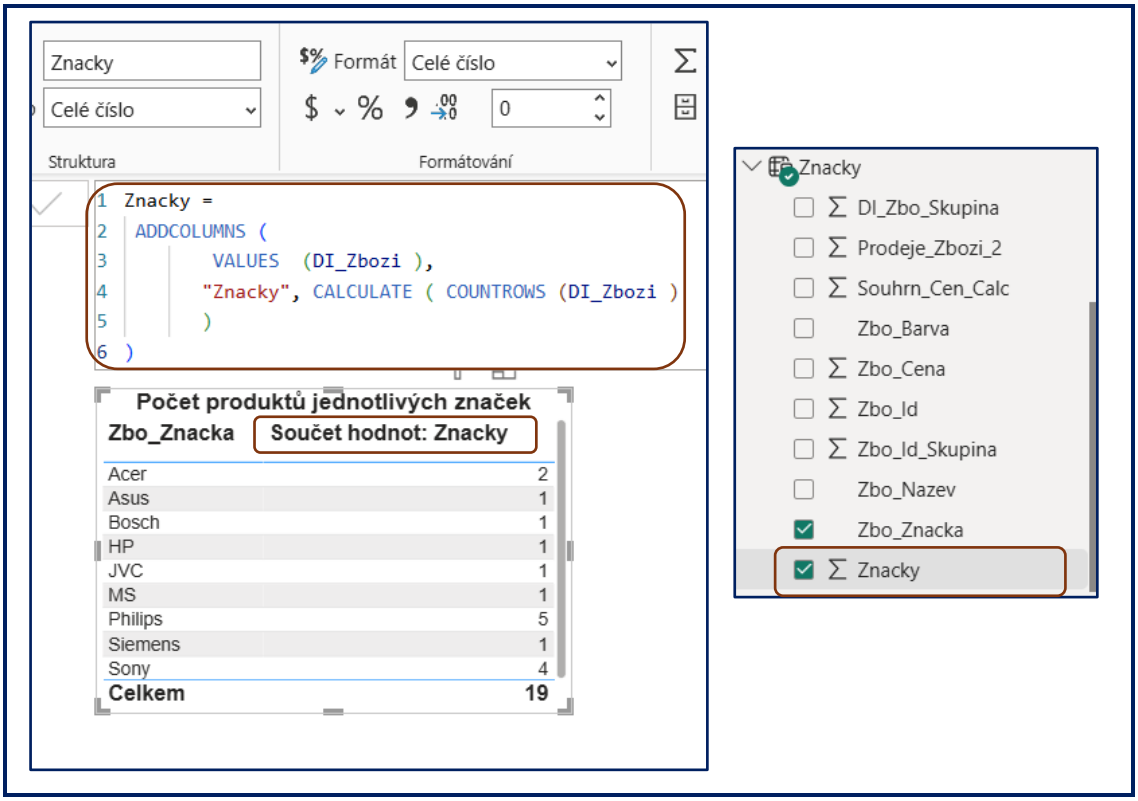
11.2 Iterátory vracející tabulku

Vedle iterátorů, které vracejí např. agregovanou hodnotu výrazu, jsou iterátory, které vracejí tabulku, jako kombinaci původní zdrojové tabulky a dalšího sloupce na základě zadaného výrazu v rámci řádkového kontextu. K tomu slouží zejména funkce *ADDCOLUMNS()* a *SELECTCOLUMNS()*. Pro ilustraci použijeme následující příklad, kde do tabulky Znacky na základě funkce *ADDCOLUMNS()* doplníme sloupec s počtem řádků, resp. produktů pro jednotlivé značky:

```
Znacky =
 ADDCOLUMNS (
       VALUES  (DI_Zbozi ),
      "Znacky", CALCULATE ( COUNTROWS (DI_Zbozi )
      )
)
```

Příklad 11-4: Využití funkce ADDCOLUMNS () pro přidání sloupce

Příklad dále dokumentuje Obrázek 11-3:



Obrázek 11-3: Doplnění sloupce do tabulky na základě funkce ADDCOLUMNS()

Zatímco funkce *ADDCOLUMNS()* vrací všechny sloupce původní tabulky (DI_Zbozi) do nové tabulky, jak ukazuje Obrázek 11-3 v tabulce „*Znacky*“, pak funkce *SELECTCOLUMNS()* vrací pouze vybranou podmnožinu sloupců původní tabulky specifikovanou v příkazu, jak ukazuje dále zjednodušený příklad (v daném případě sloupec *Zbo_Znacka*):

```
Znacky_1 =
 SELECTCOLUMNS (
      VALUES  (DI_Zbozi ),
     "Znacka", DI_Zbozi [ Zbo_Znacka ] ,
     "Znacky", CALCULATE ( COUNTROWS (DI_Zbozi )
     )
)
```

Příklad 11-5: Využití funkce SELECTCOLUMNS () pro přidání sloupce pouze k vybraným sloupcům

11.3 Funkce RANKX()

Funkce *RANKX()* se využívá pro zjištění pořadí vybrané hodnoty podle stanoveného způsobu třídění. Syntax je následující:

RANKX(
<tabulka>,
<výraz>,
[<hodnota>],
[<způsob řazení>],
[<způsob řešení shod>]
)

Kde význam parametrů je následující:

- tabulka – název skutečné tabulky která je iterována, nebo ALL() nebo ALLSELECTED () pro odstranění filtrů
- výraz – podle kterého se vytváří pořadí, např. SUM(), AVERAGE(), COUNT() apod.
- hodnota – pro hodnotu aktuálního řádku, která má být jiná než výraz,
- způsob řazení – ASC / DESC – vzestupně sestupně,
- způsob řešení shod – vynechat / nevynechat.

Princip a provádění funkce *RANKX()* obsahuje Obrázek 11-4:

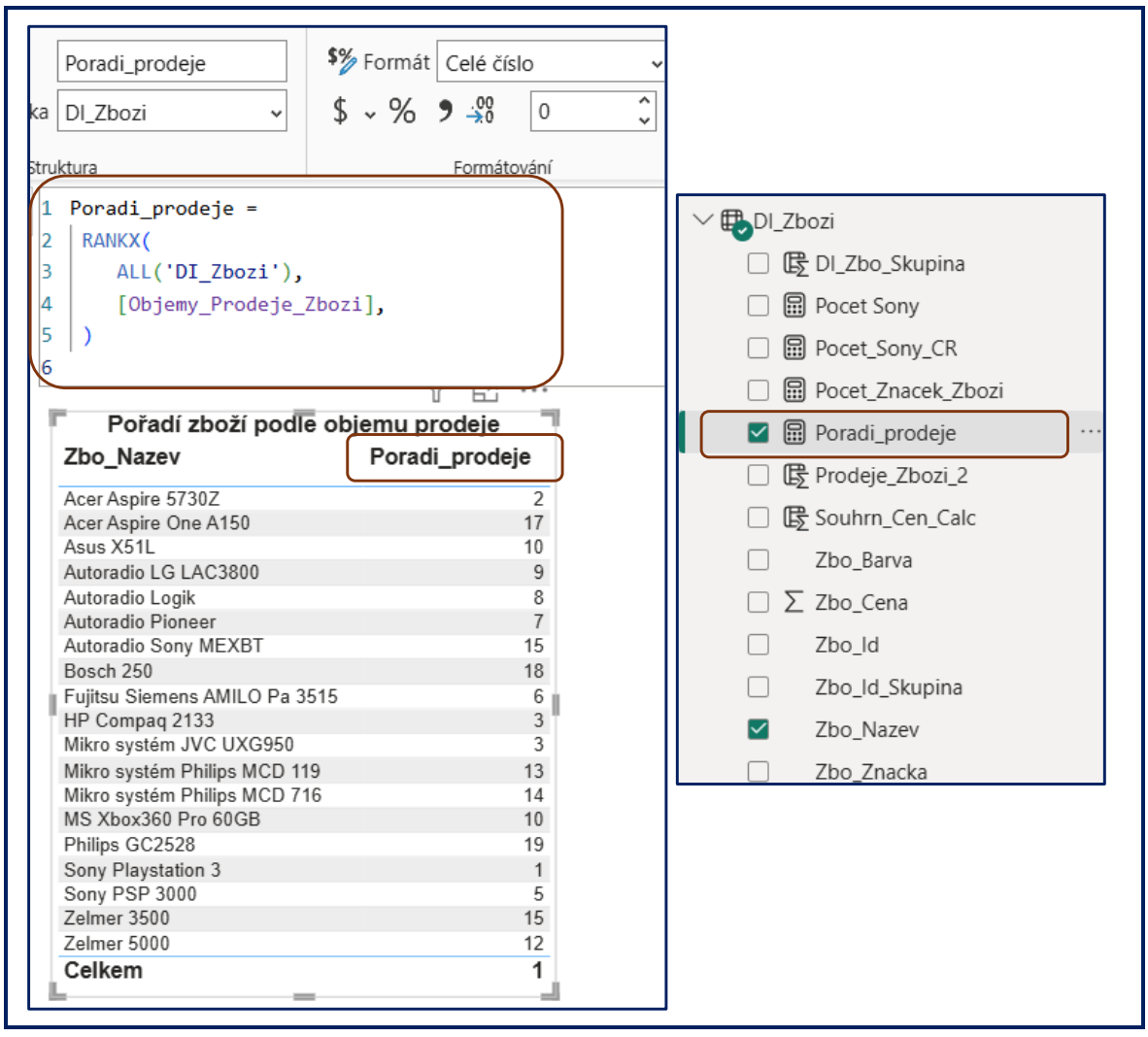
Obrázek 11-4: Postup provedení funkce RANKX()

Použití funkce RANKX() dokumentuje následující příklad:

```
Poradi_prodeje =
 RANKX(
   ALL('DI_Zbozi'),
   [Objemy_Prodeje_Zbozi],
 )
```

Příklad 11-6: Využití funkce SELECTCOLUMNS () pro přidání sloupce pouze k vybraným sloupcům

Realizaci funkce RANKX() ukazuje Obrázek 11-5:



Pořadí zboží podle objemu prodeje

Zbo_Nazev	Poradi_prodeje
Acer Aspire 5730Z	2
Acer Aspire One A150	17
Asus X51L	10
Autoradio LG LAC3800	9
Autoradio Logik	8
Autoradio Pioneer	7
Autoradio Sony MEXBT	15
Bosch 250	18
Fujitsu Siemens AMILO Pa 3515	6
HP Compaq 2133	3
Mikro systém JVC UXG950	3
Mikro systém Philips MCD 119	13
Mikro systém Philips MCD 716	14
MS Xbox360 Pro 60GB	10
Philips GC2528	19
Sony Playstation 3	1
Sony PSP 3000	5
Zelmer 3500	15
Zelmer 5000	12
Celkem	**1**

Obrázek 11-5: Určení pořadí zboží podle objemu prodeje

11.4 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Pro iterátory je charakteristické, že jde o funkce DAX končící na **„*X*“**.
- Existují dva druhy iterátorů, v prvním případě jde o jednodušší výpočty na základě řádkového kontextu, ve druhém případě obvykle o změnu kontextu pro komplexnější výpočty.
- V souvislosti s iterátory je podstatné stanovení jejich kardinality, která

představuje ***počet řádek, které mají být iterovány***.

- Některé iterátory vracejí tabulku, jako kombinaci původní zdrojové tabulky a dalšího sloupce na základě zadaného výrazu v rámci řádkového kontextu. K tomu slouží zejména funkce *ADDCOLUMNS ()* a *SELECTCOLUMNS ()*.
- Funkce *RANKX ()* se využívá pro ***zjištění pořadí*** vybrané hodnoty podle stanoveného způsobu třídění.

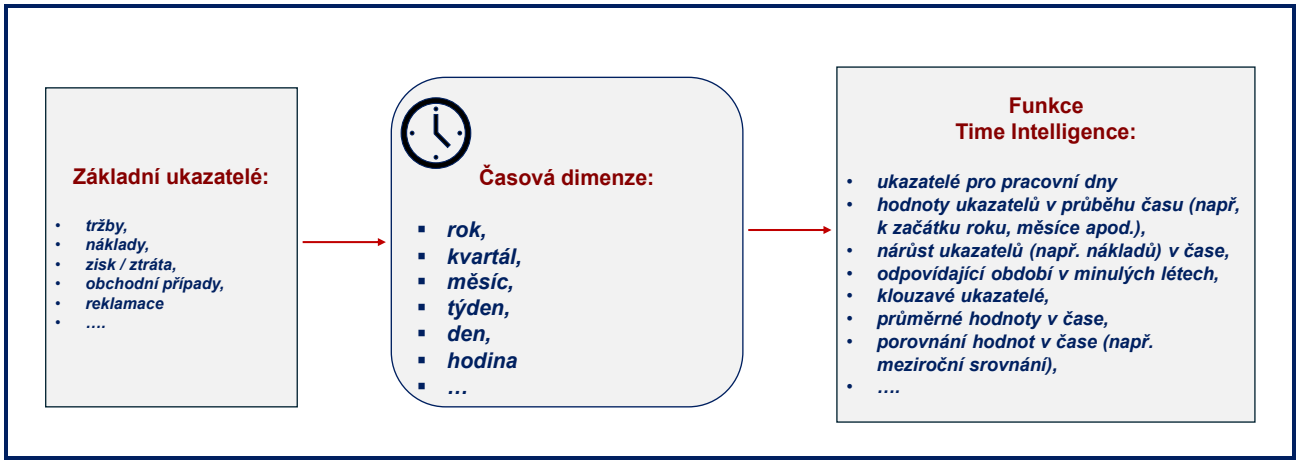
12. Time Intelligence

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.

12.1 Základní principy time intelligence

Základní princip a hlavní funkce time intelligence dokumentuje Obrázek 12-1



Obrázek 12-1: Principy a funkce 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.

12.2 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"
    )   )
```

Příklad 12-1: Vygenerování dimenzionální tabulky času s názvem DATUM se všemi potřebnými sloupci

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 )
```

Příklad 12-2: Zjištění minimálního datumu

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

Příklad 12-3: Zjištění maximálního datumu

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)
```

Příklad 12-4: Zjištění pořadí dne v týdnu

12.3 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“.

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

Příklad 12-5: Průměrné denní počty prodaných kusů zboží v pracovních dnech

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 12-2:

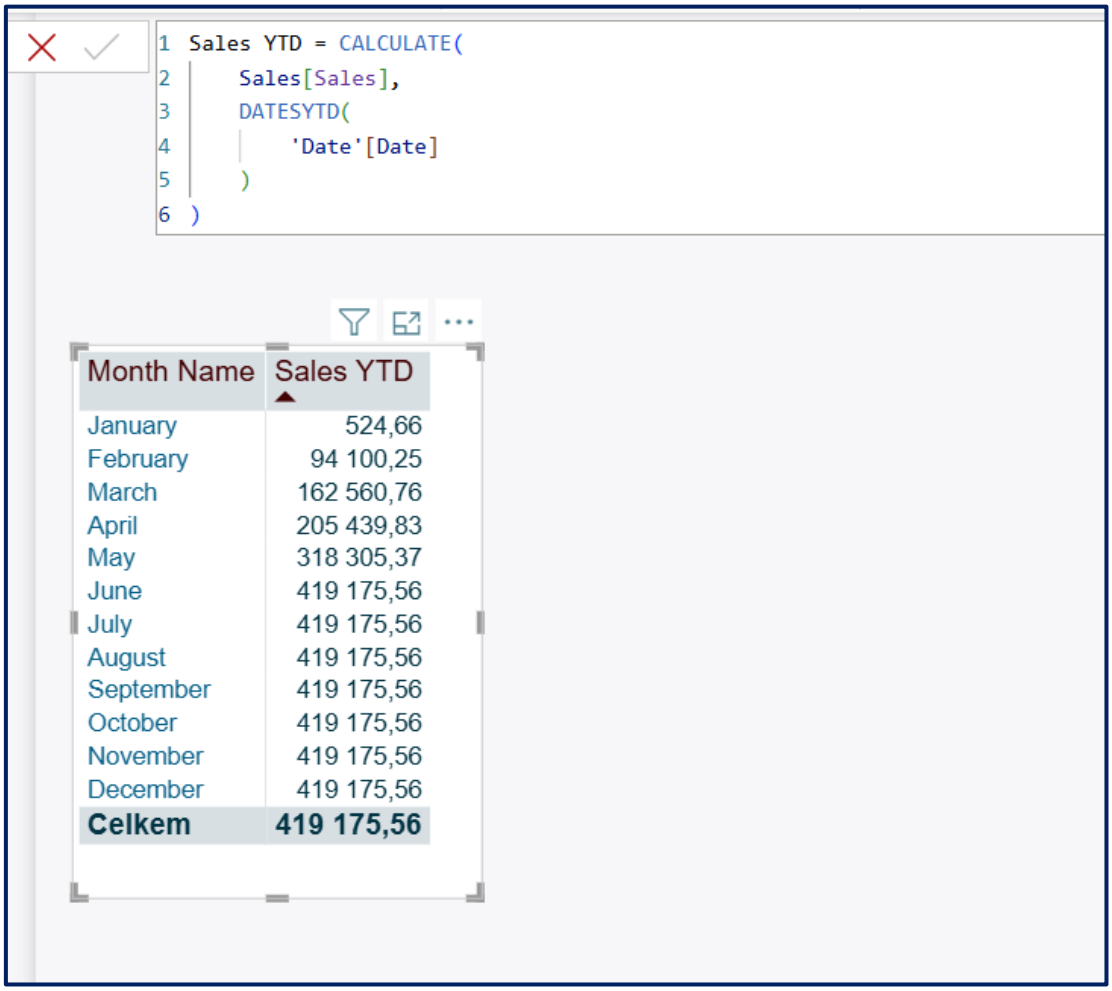
Popisky řádků	Součet Prodej_Skut_Ks	Prodeje - pracovní dny
⊟2014	5045	20
Leden	482	22
Únor	427	21
Březen	571	27
Duben	486	23
Květen	518	27
Červen	490	23
Červenec	409	18
Srpen	341	16
Září	343	16
Říjen	344	15
Listopad	310	16
Prosinec	324	14
⊟2015	3982	15
Leden	329	15
Únor	266	13
Březen	397	18
Duben	306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

Obrázek 12-2: Průměrné prodeje v pracovních dní dle měsíců a roků

12.4 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.

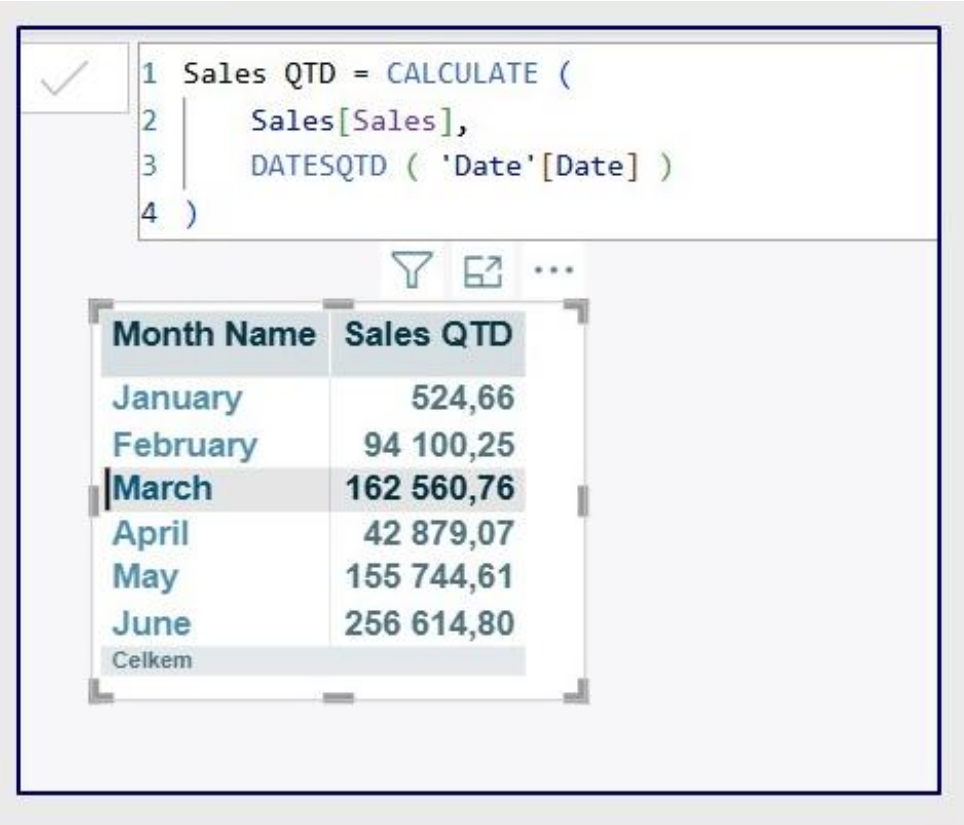
Příklad dokumentující funkci YTD představuje Obrázek 12-3



Month Name	Sales YTD
January	524,66
February	94 100,25
March	162 560,76
April	205 439,83
May	318 305,37
June	419 175,56
July	419 175,56
August	419 175,56
September	419 175,56
October	419 175,56
November	419 175,56
December	419 175,56
Celkem	**419 175,56**

Obrázek 12-3: Uplatnění funkce year-to-date

Další obrázek dokumentuje funkci QTD, Obrázek 12-4



Obrázek 12-4: 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 12-5:

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

Příklad 12-6: Nárůst nákladů pro kvartály a měsíce vzhledem k začátku roku

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 12-5: 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“ )
```

Příklad 12-7: Nárůst hodnot k začátku fiskálního roku

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:

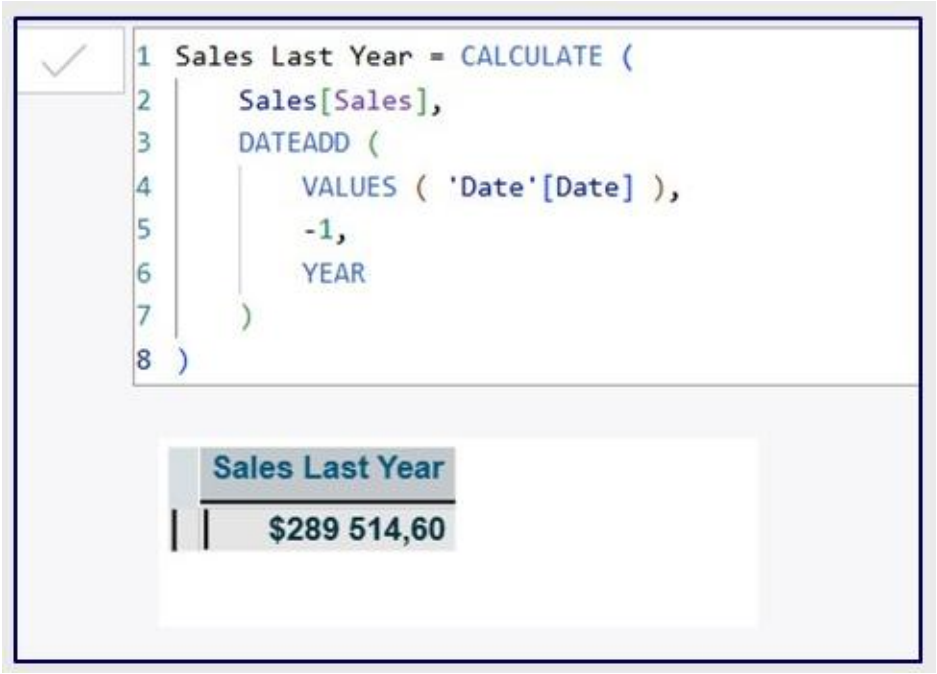
```
= CALCULATE (SUM (FTQ_Prodej [Naklady] ), DATESYTD (DI_Cas_DAX [FullDate] ) )
```

Příklad 12-8: CALCULATE() pro nárůst hodnot k začátku fiskálního roku

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.

12.5 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 12-6:



Obrázek 12-6: 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 12-7:

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

Příklad 12-9: Sledování podnikových nákladů za minulá období

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 12-7: 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 ))
```

Příklad 12-10: Funkce DATEADD() využívá počtu a typu sledovaných období

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 ))
```

Příklad 12-11: Funkce PARALLELPERIOD() pro celkový objem nákladů za určený minulý rok

12.6 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] )
    ) )
```

Příklad 12-12: Klouzavý roční souhrn sleduje souhrnné hodnoty za posledních 12 měsíců

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 12-8:

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 12-8: Klouzavý souhrn hodnot

12.7 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] )
```

Příklad 12-13: Průměrné náklady na prodaný výrobek

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] ) )
```

Příklad 12-14: Porovnání průměrných nákladů oproti minulému roku

Standardní operací v reportingu jsou metriky Time intellifgence, jako například ***rozdíly hodnot aktuálního a minulého roku***.

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

Příklad 12-15: Objem nákladů v minulém roce

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

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

Příklad 12-16: Rozdíl nákladů k minulému roku

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

Příklad 12-17: Rozdíl nákladů k minulému roku v podílovém vyjádření

12.8 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Základem je ***tabulka časové dimenze*** obsahující kromě údaje o čase i případné další potřebné údaje.
- Pro ***identifikaci období*** je k dispozici primární klíč nebo v definované struktuře, např. 20170101). Vedle primárního klíče se pro většinu funkcí DAX spojených s časem používá i ***plné datum (FullDate)***, které musí být datového typu.
- Podstatné je zajistit, aby ***rozsah časové dimenze*** pokrýval časový rozsah dat faktových tabulek včetně i potřebné rezervy do budoucnosti.
- K základním funkcím time intelligence patří:
    - výpočty ukazatelů rozlišujících pracovní dny a svátky,
    - funkce pro sledování hodnot ukazatelů v průběhu času, jako např. kalkulací jsou ty, které jsou označeny jako YTD *(year-to-date)* pro postupný nárůst hodnot od aktuálního data k začátku roku,
    - sledování hodnot ukazatelů jak za aktuální období, resp. rok, tak za odpovídající období v minulých létech,
    - klouzavé ukazatele, např. klouzavý roční souhrn (moving annual total,

MAT), který sleduje souhrnné hodnoty za posledních 12 měsíců,
- průměrné hodnoty v čase,
- porovnání ukazatelů za období, např. meziroční srovnání,
- a další

13. Další analytické funkce založené na DAX

Část kapitoly je věnována několika komplexním funkcím založeným na možnostech jazyka DAX.. ***Účelem*** této kapitoly je charakterizovat tyto specifické funkce ve větším detailu a se všemi podstatnými parametry.

(Zdroj: Ferrari, Russo, 2026).

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

13.1 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] )
                  )
```

Příklad 13-1: Vytvoření kalkulovaného sloupce Band_Prodej v rámci tabulky faktů

13.2 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_Prodeje = RANKX (ALLSELECTED (Dl_Zbozi),  [Suma_Prodeje] )
```

Příklad 13-2: Funkce *RANKX* pro pořadí zboží podle objemů 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.

13.3 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] )
```

Příklad 13-3: Zjištění počtu aktivních zákazníků

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

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

Příklad 13-4: Určení základní hodnoty

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

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

Příklad 13-5: Procentuální nárůst zákazníků k roku 2024

13.4 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Smyslem ***seskupování dat (banding)*** je sdružovat velká množství dat do definovaných skupin. Předpokladem je zde vytvoření speciální tabulky, která bude součástí sémantického modelu.
- Zpracování ***pořadí různých objektů*** podle stanovených hodnot. K tomu se užívá funkce *RANKX*, která má ***charakter iterátoru*** a využívá dva parametry – název tabulky a výraz.
- Další z častých funkcí je sledování zvyšování počtu zákazníků v jednotlivých létech, kdy na počátku je třeba definovat novou základní hodnotu zvoleného ukazatele ve vztahu k zákazníkům.

14. Další funkce s respektováním vazeb

Kapitola 5 prezentovala operace DAX ***na vzájemně provázaných tabulkách,*** a to v základní formě (*RELATED a RELATEDTABLE*). ***Účelem*** této kapitoly je doplnit zmíněnou úvodní kapitolu i o další možnosti realizace operací na vzájemně provázaných tabulkách.

14.1 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 14-1) 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 14-1: 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 14-2:

```
Korun_za_Euro = MAX (CZK_EUR [Kc_Euro])
```

Příklad 14-1: Míra maximální hodnoty kursu

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 14-2: Vytvoření míry – maximální hodnota kursu

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

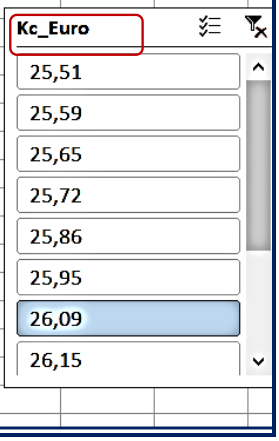
V dalším kroku vytvoříme míru pro přepočet průměrné ceny zboží podle zvoleného aktuálního kursu Kč k Euru:

```
Cena_Euro = [Průměr Cena_Ks] / [Korun_za_Euro]
```

Příklad 14-2: Průměrná cena zboží podle aktuálního kursu Kč k Euru

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 14-3

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 14-3: Přepočet průměrné ceny zboží podle aktuálního kursu Eura

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

14.2 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.

14.3 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

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

Příklad 14-3: Míra pro zjištění počtu řádků výsledné tabulky z funkce GENERATE

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])
                                        )
                )
```

Příklad 14-4: Tabulka s řádky se shodou mezi tabulkami u atributu, např. „Cena“

14.4 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- Součástí řešení mohou být i ***speciální tabulky,*** které obvykle slouží ***jako parametry*** pro přepočty hodnot ukazatelů podle aktuální situace. Tyto tabulky ***nejsou propojené vazbou*** k žádné jiné tabulce. Mohou sloužit jako specifický obsah v rámci sliceru.
- Funkce ***CROSSJOIN*** vytváří tabulku na bázi karteziánského součinu, kde ke

každé řádce tabulky 1 jsou přiřazeny všechny řádky tabulky2 atd.

- Funkce ***GENERATE*** vytváří rovněž tabulku na bázi karteziánského součinu, kde ke každé řádce tabulky 1 jsou přiřazeny všechny řádky tabulky2. Parametry musí být dvě rozdílné tabulky.

15. Zpracování dotazů

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.

15.1 Funkce SUMMARIZE

Funkce SUMMARIZE umožňuje vytvořit pouze jednu souhrnnou tabulku. K tabulce lze vytvářet vazby k ostatním tabulkám, lze do ní podle potřeby doplňovat kalkulované sloupce i míry.

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

```
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])
    )
```

Příklad 15-1: Souhrnná tabulka tržeb podle jednotlivých let a druhů zboží

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])
       )
    )
```

Příklad 15-2: Záznamy s chybějícími hodnotami

15.1.1 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],
)
```

Příklad 15-3: Sumarizovaná tabulka podle roku dodávek

15.1.2 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])
   )
```

Příklad 15-4: Souhrnné hodnoty tržeb za jednotlivé roky za region Ostrava

15.2 Funkce SUMMARIZECOLUMNS

Funkce ***SUMMARIZECOLUMNS*** definuje jméno sloupce pro realizaci agregací na bázi funkce „group by“.

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.

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

Příklad 15-5:

```
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])
 )
```

Příklad 15-6:

15.3 Funkce GROUP BY

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

Syntaxe funkce je tato:

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

```
Souhrnna_tabulka_trzeb_na_zaklade_GROUPBY =
GROUPBY (
```

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

Příklad 15-7:

15.3.1 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.*

```
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] )
         ).
```

Příklad 15-8: Výpočet tržeb podle prodaných kusů a ceny

15.4 Pracovní závěry

Z kapitoly vyplývají následující ***závěry***:

- ***Funkce SUMMARIZE*** umožňuje vytvořit pouze jednu souhrnnou tabulku. K tabulce lze vytvářet vazby k ostatním tabulkám, lze do ní podle potřeby doplňovat kalkulované sloupce i míry
- Funkce ***SUMMARIZECOLUMNS*** definuje jméno sloupce pro realizaci agregací na bázi funkce „group by“.
- Funkce ***GROUP BY*** je obdobná jako *SUMMARIZE.* Rozdíl je v tom, že funguje pouze s využitím iterátorů, např. SUMX, AVERAGEX apod.

16. 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.

17. Příloha 1: Data pro dokumentační příklady

17.1 Prodej zboží (FTQ_Prodej)

Faktová tabulka *FTQ_Prodej* obsahuje hodnoty různých ukazatelů prodeje zboží a klíče dimenzionálních tabulek (*FK, Foreign Key*).

Identifikátor	Název	Poznámka
FTQ_Prodej_Id	Jednoznačné id. záznamu faktové tabulky	PK
Cas_Id	Id. časového údaje, kdy dochází k prodeji, resp. k dodání zboží	FK
Reg_Id	Id. regionu, kde se realizuje prodej	FK
Zak_Id	Id. zákazníka, kterému bylo zboží prodáno	FK
Zbo_Id	Id. prodaného zboží	FK
	Ukazatelé	
Cena_Ks	Cena za kus, resp. jednotku zboží	
Prodej_Skut_Ks	Skutečný počet kusů prodaného zboží	
Naklady	Objem nákladů na prodej zboží	
Datum_Obj	Datum vystavení objednávky na zboží	
Datum_Dodani		
Rok	*Rok prodeje zbožíí*	
Prodej_Kc	Objem prodeje zboží v Kč	
Datum_dodani	Datum dodání zboží zákazníkovi	

Příklady dat:

FTQ_Prodej_Id	Cas_Id	Reg_Id	Zak_Id	Zbo_Id	Cena_Ks	Prodej_Skut_Ks	Naklady	Datum_Obj	Datum_Dodani	Rok	Prodej_Kc
553	553	1	1	1	90,00	2	75,00	29.06.2025	02.07.2025	2025	180
554	554	2	2	2	60,00	2	45,00	30.06.2025	03.07.2025	2025	120
555	555	3	3	3	80,00	2	65,00	01.07.2025	04.07.2025	2025	160
556	556	4	4	4	40,00	2	25,00	02.07.2025	05.07.2025	2025	80
557	557	5	5	9	30,00	2	15,00	03.07.2025	06.07.2025	2025	60
558	558	6	6	10	50,00	2	35,00	04.07.2025	08.07.2025	2025	100
559	559	7	7	11	20,00	3	5,00	05.07.2025	09.07.2025	2025	60
560	560	8	8	18	30,00	3	15,00	06.07.2025	10.07.2025	2025	90
561	561	9	9	19	60,00	3	45,00	07.07.2025	11.07.2025	2025	180
562	562	10	10	20	100,00	3	85,00	05.07.2025	09.07.2025	2025	300
563	563	11	11	21	20,00	9	5,00	06.07.2025	10.07.2025	2025	180
564	564	12	12	22	50,00	9	35,00	07.07.2025	09.07.2025	2025	450
565	565	13	13	23	30,00	5	15,00	07.07.2025	09.07.2025	2025	150
566	566	14	14	24	80,00	5	65,00	07.07.2025	09.07.2025	2025	400
1105	375	15	15	25	120,00	2	105,00	28.06.2026	30.06.2026	2026	240
1106	376	16	16	60	10,00	2	-5,00	28.06.2026	30.06.2026	2026	20
1107	377	17	17	61	30,00	2	15,00	29.06.2026	01.07.2026	2026	60
1108	378	18	1	62	40,00	2	25,00	30.06.2026	05.07.2026	2026	80
1109	379	19	2	63	50,00	2	35,00	02.07.2026	07.07.2026	2026	100
1110	380	20	3	1	90,00	2	75,00	03.07.2026	08.07.2026	2026	180
1111	381	21	4	2	60,00	3	45,00	04.07.2026	09.07.2026	2026	180
1112	382	22	5	3	80,00	3	65,00	05.07.2026	10.07.2026	2026	240
1113	383	23	6	4	40,00	3	25,00	06.07.2026	11.07.2026	2026	120
1114	384	24	7	9	30,00	3	15,00	07.07.2026	12.07.2026	2026	90
1115	385	25	8	10	50,00	9	35,00	08.07.2026	13.07.2026	2026	450
1116	386	26	9	11	20,00	9	5,00	09.07.2026	10.07.2026	2026	180

FTQ_Prodej_Id	Cas_Id	Reg_Id	Zak_Id	Zbo_Id	Cena_Ks	Prodej_Skut_Ks	Naklady	Datum_Obj	Datum_Dodani	Rok	Prodej_Kc
1117	387	27	10	18	30,00	5	15,00	10.07.2026	11.07.2026	2026	150
1118	388	28	11	19	60,00	5	45,00	11.07.2026	12.07.2026	2026	300
1119	389	29	12	20	100,00	5	85,00	12.07.2026	13.07.2026	2026	500
1120	390	30	13	21	20,00	5	5,00	13.07.2026	14.07.2026	2026	100
1121	391	31	14	22	50,00	10	35,00	14.07.2026	15.07.2026	2026	500
1122	392	32	15	23	30,00	10	15,00	15.07.2026	16.07.2026	2026	300
1123	393	33	16	24	80,00	6	65,00	16.07.2026	23.07.2026	2026	480
1124	394	34	17	25	120,00	6	105,00	17.07.2026	24.07.2026	2026	720
1125	395	35	1	60	10,00	6	10,00	18.07.2026	25.07.2026	2026	60
1126	396	36	2	61	30,00	6	15,00	19.07.2026	26.07.2026	2026	180
1127	397	37	3	62	40,00	6	25,00	20.07.2026	27.07.2026	2026	240
1128	398	38	4	63	50,00	6	35,00	21.07.2026	28.07.2026	2026	300
1129	399	1	5	1	90,00	3	75,00	22.07.2026	29.07.2026	2026	270
1130	400	2	6	2	60,00	3	45,00	23.07.2026	25.07.2026	2026	180
1131	401	3	7	3	80,00	3	65,00	24.07.2026	26.07.2026	2026	240
1132	402	4	8	4	40,00	3	25,00	25.07.2026	27.07.2026	2026	120
1133	403	5	9	9	30,00	8	15,00	26.07.2026	28.07.2026	2026	240
1134	404	6	10	10	50,00	8	35,00	27.07.2026	29.07.2026	2026	400
1135	405	7	11	11	20,00	7	5,00	28.07.2026	30.07.2026	2026	140
1136	406	8	12	18	30,00	7	15,00	29.07.2026	01.08.2026	2026	210
1137	407	9	13	19	60,00	7	45,00	30.07.2026	02.08.2026	2026	420
1138	408	10	14	20	100,00	7	85,00	31.07.2026	03.08.2026	2026	700
1139	409	11	15	21	20,00	1	5,00	01.08.2026	04.08.2026	2026	20

17.2 Čas (DI_Cas)

Časová dimenze slouží pro sledování vývoje jednotlivých ukazatelů, např. vývoje nákladů, a to např. v absolutních hodnotách nákladů, nebo v poměrových ukazatelích. Konkrétní strukturu dimenze Čas lze zvolit podle aktuální potřeby.

Identifikátor	Název	Poznámka
Cas_Id	Id. časového údaje	PK
Cas_Rok	Číslo roku	např. 2025
Cas_Pololeti	Číslo pololetí	1 - 2
Cas_Kvartal	Číslo kvartálu	1 - 4
Cas_Mesic	Číslo měsíce	1 - 12
Cas_Tyden	Číslo týdne	1 - 5
Cas_Den	Číslo dne	1 – 31
Cas_Datum	Celé datum ve standardní formě, např. rrrr mm dd	
Cas_Mesic_Nazev	Název měsíce	
Cas_Den_Nazev	Název dne	
Cas_Poradi_dne	Pořadové číslo den od začátku roku	
Cas_Typ_dne	Typ dne: Pracovní den, Víkend, Svátek	
Cas_Den_v_Tydnu	Pořadové číslo dne v týdnu	

Příklady dat:

Cas_Id	Cas_Rok	Cas_Pololeti	Cas_Kvartal	Cas_Mesic	Cas_Tyden	Cas_Den	Cas_Mesic_Nazev	Cas_Den_Nazev	Cas_Poradi_dne	Cas_Typ_dne
553	2025	2	3	7	27	1	Červenec	Úterý	182	Pracovní
554	2025	2	3	7	27	2	Červenec	Středa	183	Pracovní
555	2025	2	3	7	27	3	Červenec	Čtvrtek	184	Pracovní
556	2025	2	3	7	27	4	Červenec	Pátek	185	Pracovní
557	2025	2	3	7	27	5	Červenec	Sobota	186	Svátek
558	2025	2	3	7	27	6	Červenec	Neděle	187	Svátek
559	2025	2	3	7	28	7	Červenec	Pondělí	188	Pracovní
560	2025	2	3	7	28	8	Červenec	Úterý	189	Pracovní
561	2025	2	3	7	28	9	Červenec	Středa	190	Pracovní
562	2025	2	3	7	28	10	Červenec	Čtvrtek	191	Pracovní
563	2025	2	3	7	28	11	Červenec	Pátek	192	Pracovní
564	2025	2	3	7	28	12	Červenec	Sobota	193	Svátek
565	2025	2	3	7	28	13	Červenec	Neděle	194	Svátek
566	2025	2	3	7	29	14	Červenec	Pondělí	195	Pracovní
375	2026	2	3	7	29	15	Červenec	Úterý	196	Pracovní
376	2026	2	3	7	29	16	Červenec	Středa	197	Pracovní
377	2026	2	3	7	29	17	Červenec	Čtvrtek	198	Pracovní
378	2026	2	3	7	29	18	Červenec	Pátek	199	Pracovní
379	2026	2	3	7	29	19	Červenec	Sobota	200	Svátek
380	2026	2	3	7	29	20	Červenec	Neděle	201	Svátek
381	2026	2	3	7	30	21	Červenec	Pondělí	202	Pracovní
382	2026	2	3	7	30	22	Červenec	Úterý	203	Pracovní
383	2026	2	3	7	30	23	Červenec	Středa	204	Pracovní
384	2026	2	3	7	30	24	Červenec	Čtvrtek	205	Pracovní
385	2026	2	3	7	30	25	Červenec	Pátek	206	Pracovní

Cas_Id	Cas_Rok	Cas_Pololeti	Cas_Kvartal	Cas_Mesic	Cas_Tyden	Cas_Den	Cas_Mesic_Nazev	Cas_Den_Nazev	Cas_Poradi_dne	Cas_Typ_dne
386	2026	2	3	7	30	26	Červenec	Sobota	207	Svátek
387	2026	2	3	7	30	27	Červenec	Neděle	208	Svátek
388	2026	2	3	7	31	28	Červenec	Pondělí	209	Pracovní
389	2026	2	3	7	31	29	Červenec	Úterý	210	Pracovní
390	2026	2	3	7	31	30	Červenec	Středa	211	Pracovní
391	2026	2	3	7	31	31	Červenec	Čtvrtek	212	Pracovní
392	2026	2	3	8	31	1	Srpen	Pátek	213	Pracovní
393	2026	2	3	8	31	2	Srpen	Sobota	214	Svátek
394	2026	2	3	8	31	3	Srpen	Neděle	215	Svátek
395	2026	2	3	8	32	4	Srpen	Pondělí	216	Pracovní
396	2026	2	3	8	32	5	Srpen	Úterý	217	Pracovní
397	2026	2	3	8	32	6	Srpen	Středa	218	Pracovní
398	2026	2	3	8	32	7	Srpen	Čtvrtek	219	Pracovní
399	2026	2	3	8	32	8	Srpen	Pátek	220	Pracovní
400	2026	2	3	8	32	9	Srpen	Sobota	221	Svátek

17.3 Regiony (DI_Regiony)

Identifikátor	Název	Poznámka
Reg_Id	Id. regionu	PK
Reg_Nazev_Stat	Název státu	
Reg_Nazev	Název regionu	

Příklady dat:

Reg_Id	Stat_Id	Reg_Nazev_Stat	Reg_Nazev
1	1	ČR	Praha
2	1	ČR	Středočeský k.
3	1	ČR	Západočeský k.
4	1	ČR	Karlovarský k.
5	1	ČR	Jihočeský k.
6	1	ČR	Severočeský k.
7	1	ČR	Východočeský k
8	1	ČR	Pardubický k.
9	1	ČR	Vysočina
10	1	ČR	Brno
11	1	ČR	Jihomoravský k.
12	1	ČR	Ostrava
13	1	ČR	Severomoravsý k.
14	1	ČR	Moravsko Slezský k.
15	1	ČR	Olomoucký k.
16	2	Slovensko	Bratislava
17	2	Slovensko	Západoslovenský k.
18	2	Slovensko	Žilina
19	2	Slovensko	Martin
20	2	Slovensko	Středoslovesnký k.
21	2	Slovensko	Trnava
22	2	Slovensko	Bánská Bystrica
23	2	Slovensko	Košice
24	2	Slovensko	Východoslovenský k.
25	3	Ukrajina	Kijev
26	3	Ukrajina	Západní Ukajina
27	3	Ukrajina	Doneck
28	3	Ukrajina	Střední Ukrajina
29	3	Ukrajina	Východní Ukrajina
30	4	Německo	Berlin
31	4	Německo	Mnichov
32	4	Německo	Bavorsko
33	4	Německo	Bonn
34	4	Německo	Hessensko
35	4	Německo	Dražďany
36	4	Německo	Sasko
37	4	Německo	Porúří
38	4	Německo	Hamburk

17.4 Zákazníci (Dl_Zakaznici)

Identifikátor	Název	Poznámka
Zak_Id	Jednoznačná identifikace zákazníka	PK
Zak_Nazev_Kat	Plný název kategorie zákazníků	FK
Zak_Nazev	Plný název zákazníka	
Zak_Vyznam	Význam zákazníka pro obchodní aktivity podniku	

Příklady dat:

Zak_Id	Kategorie_Id	Zak_Nazev_Kat	Zak_Nazev	Zak_Vyznam
1	1	Automobily	Amondon	Významný zákazník
2	1	Automobily	Autocentrum Vo-kurka	Standardní zákazník
3	1	Automobily	AVX Centrum	Významný zákazník
4	1	Automobily	Auto Luxus	Významný zákazník
5	1	Automobily	Auto BMX	Významný zákazník
6	1	Automobily	Auto Hasiči	Standardní zákazník
7	1	Automobily	Auto Filip	Standardní zákazník
8	2	Banky	Bank of Honolulu	Standardní zákazník
9	2	Banky	Česká banka	Významný zákazník
10	2	Banky	Obchodní banka	Významný zákazník
11	2	Banky	Velká hypoteční	Významný zákazník
12	2	Banky	IPV	Standardní zákazník
13	3	Cestovní kanceláře	Dream Tour	Významný zákazník
14	3	Cestovní kanceláře	ALEXander	Významný zákazník
15	3	Cestovní kanceláře	Canaria Travel	Významný zákazník
16	3	Cestovní kanceláře	Happypartner	Standardní zákazník
17	3	Cestovní kanceláře	Tam-i-zpět	Standardní zákazník

17.5 Zboží (Dl_Zbozi)

Identifikátor	Název	Poznámka
Zbo_Id	Id. zboží	PK
Zbo_Id_Skupina	Id. skupiny zboží	
Zbo_Nazev	Název zboží	
Zbo_Cena	Cena zboží	
Zbo_Barva	Barva, barevná úprava zboží	
Zbo_Znacka	Značka zboží, např. „*Samsung*“	

Příklady dat:

Zbo_Id	Zbo_Id_Skupina	Zbo_Nazev	Zbo_Cena	Zbo_Barva	Zbo_Znacka
1	1	Autoradio Logik	90,00	černá	Philips
2	1	Autoradio LG LAC3800	60,00	stříbrná	Philips
3	1	Autoradio Pioneer	80,00	bílá	Sony
4	1	Autoradio Sony MEXBT	40,00	modrá	Sony

Zbo_Id	Zbo_Id_Skupina	Zbo_Nazev	Zbo_Cena	Zbo_Barva	Zbo_Znacka
9	2	Mikro systém Philips MCD 119	30,00	kávová	Philips
10	2	Mikro systém JVC UXG950	50,00	zelená	JVC
11	2	Mikro systém Philips MCD 716	20,00	černá	Philips
18	3	MS Xbox360 Pro 60GB	30,00	hnědá	MS
19	3	Sony PSP 3000	60,00	kávová	Sony
20	3	Sony Playstation 3	100,00	zelená	Sony
21	4	Acer Aspire One A150	20,00	černá	Acer
22	4	HP Compaq 2133	50,00	stříbrná	HP
23	4	Asus X51L	30,00	bílá	Asus
24	4	Fujitsu Siemens AMILO Pa 3515	80,00	modrá	Siemens
25	4	Acer Aspire 5730Z	120,00	modrá	Acer
60	5	Philips GC2528	10,00	červená	Philips
61	5	Bosch 250	30,00	šedá	Bosch
62	6	Zelmer 3500	40,00	žlutá	Zelmer
63	6	Zelmer 5000	50,00	hnědá	Zelmer

17.6 Skupina zboží (Dl_Zbozi_Skupina)

Identifikátor	Název	Poznámka
Zbo_Id_Skupina	Id. zboží	PK
Zbo_Nazev_Skupina	Název skupiny zboží	

Příklady dat:

Zbo_Id_Skupina	Zbo_Nazev_Skupina
1	Auto hifi
2	Hifi
3	Herní konzole
4	Notebooky
5	Žehličky
6	Vysavače

18. Příloha 2: Přehled funkcí DAX podle skupin

18.1 Agregační funkce

Agregační funkce zajišťují agregace hodnot ve sloupci tabulky a vrací jednu hodnotu.

	Agregace numerických sloupců
SUM	Zajišťuje agregaci specifikovaných hodnot. Např.: *´Objem prodeje´ := SUM (FTQ_Prodej [Prodej_Kc])*
AVERAGE	Aritmetický průměr z hodnot specifikovaných výběrem. Např.: *´Prumer naklady´ := AVERAGE (FTQ_Prodej [Naklady])*
MIN	Zjišťuje minimální hodnotu ze specifikovaných hodnot. Podporuje také textové hodnoty.
MAX	Zjišťuje maximální hodnotu ze specifikovaných hodnot. Podporuje také textové hodnoty.
	Počty řádků nebo hodnot
COUNT	Počet dat všech typů, které nejsou prázdné a kromě Boolean.
COUNTBLANK	Počet prázdných prvků sloupce (mezer nebo prázdných řetězců)
COUNTROWS	Počet řádků tabulky. Jako parametr předpokládá jméno tabulky, nikoli sloupce.
DISTINCTCOUNT	Počet unikátních hodnot (bez duplicit) včetně mezer.
DISTINCTCOUNTNOB-LANK	Počet unikátních hodnot (bez duplicit) bez mezer.
AVERIGEA, COUNTA, MINA a MAXA.	Pracují s hodnotami typu Boolean, kde TRUE se konvertuje na hodnotu 1 a FALSE na 0. Textové hodnoty jsou vždy konvertovány na 0. Textové hodnoty jsou vždy konvertovány na 0. Použití těchto funkcí se však nedoporučuje.
	Sumarizační funkce
SUMMARIZE	
SUMMARIZECOLUMNS	
GROUP BY	
CURRENTGROUP	

18.2 Iterátory

Iterátory představují funkce, které agregují výsledky výrazů oproti agregacím základních hodnot (v předchozí kapitole).

FILTER	
ADDCOLUMS	
SUMX	Suma hodnot z vyhodnocených výrazů v řádcích tabulky. Ve výkonu mezi SUM a SUMX není rozdíl.
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.

18.3 Logické funkce

Využívají se v případech, kdy je třeba do výrazu vložit logickou podmínku, např. při rozlišení výpočtu na základě testované hodnoty, nebo v případě určení akce na základě chybové podmínky,

AND	Logický součin.
FALSE	Není pravda, neplatná hodnota.
IF	Podmínka (Jestliže platí…, pak…).
IFERROR	Jestliže došlo k chybě. Např.: *FTQ_Prodej [Objem_prodeje] = IFERROR (FTQ_Prodej [Prodej_ks] * FTQ_Prodej [Cena], BLANK())*
NOT	Negace.
TRUE	Je pravda, platná hodnota.
OR	Logický součet.
SWITCH	Přepínač. Např.: *´DI_Zbozi´ [Barva_Spec] =* *SWITCH (* *´DI_Zbozi´ [Zbo_Barva]* *„B“, „Bílá“* *„C“, „Černá“* *„M“, „Modrá“* *„Jiná“* *)*

18.4 Informační funkce

Používají se, pokud je třeba analyzovat typ výrazu. Všechny funkce vrací hodnotu typu Boolean a mohou být využity v jakémkoli logickém výrazu.

ISBLANK	Existuje mezera.
ISERROR	Pokud došlo k chybě. Např.: *FTQ_Prodej [Spravna_cena] = NOT ISERROR (VALUE (FTQ_Prodej [Cena]))*
ISLOGICAL	Hodnota je logická.
ISNONTEXT	Hodnota není textová.
ISNUMBER	Hodnota je numerická.
ISTEXT	Hodnota je textová.

18.5 Matematické funkce

Sada matematických funkcí je obdobná Excelu s využitím v běžných matematických výrazech.

ABS	Absolutní hodnota
EXP	
FACT	
LN, LOG, LOG10	Logaritmus
MOD	
PI	
POWER	Mocnina
QUOTIENT	Celočíselná hodnota podílu dvou čísel
SIGN	
SQRT	Odmocnina.
RANDBETWEEN, RAND	Generuje náhodná čísla

ROUNDDOWN, FLOOR, TRUNC	Zaokrouhlení hodnot dolů
ROUNDUP, CELING	Zaokrouhlení hodnot nahoru
MROUND, ROUND	Zaokrouhlení hodnot

18.6 Textové funkce

Funkce jsou účelné při operaci s texty a výběrech dat z textových řetězců

CONCATENATE	
CONCATENATEX	
EXACT	
FIND	
FIXED	
FORMAT	
LEFT	
LEN	
LOWER	
MID	
REPLACE	
REPT	
RIGHT	
SEARCH	
SUBSTITUTE	
TRIM	
UPPER	
VALUE	

18.7 Konverzní funkce

I když DAX konvertuje data podle operátorů výrazu, je k dispozici několik funkcí pro explicitní konverze datových typů.

CURRENCY	Transformuje hodnotu výrazu do typu CURRENCY.
INT	Transformuje hodnotu výrazu do typu Integer.
DATE	Transformuje datum do typu DateTime.
TIME	Transformuje čas do typu DateTime
VALUE	Transformuje textový řetěze do numerického formátu.
FORMAT	Transformuje numerickou hodnotu na textový řetězec. Např.: *= FORMAT (DATE (2026, 02, 10), „yyyyy mm dd“)* Na „2026 Feb 10“
DATEVALUE	Konvertuje datumy v různých formátech (Evropský,Americký,..)

18.8 Datové a časové funkce

Zahrnuje běžné časové funkce a jednoduché transformace do a z typu DateTime.

DATE	
DATEVALUE	
DAY	

EDATE	
EOMONTH	
HOUR	
MINUTE	
MONTH	
NOW	
SECOND	
TIME	
TIMEVALUE	
TODAY	
WEEKDAY	
WEEKNUM	
YEAR	
YEARFRAC	

18.9 Relační funkce

RELATED	Realizace vazby 2 nebo více vzájemně provázaných tabulek s kardinalitou M : 1. Ref.: 7.1
RELATEDTABLE	Realizace vazby 2 nebo více vzájemně provázaných tabulek s kardinalitou 1: M. Ref.: 7.1
CROSSJOIN	Ref.: 14.2
GENERATE	Ref.: 14.3

18.10 Funkce komplexního charakteru

CALCULATE	Nastavuje nový filtr kontext a na jeho základě vyhodnocuje specifikovaný výraz. Ref.: 8, 9.3
CALCULATE TABLE	Modifikuje filtr kontext výrazu, vrací tabulku.

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.

FERRARI, A, RUSSO, M.: The Definitive Guide to DAX: Mastering the semantic model expression language for Microsoft Power BI, Fabric, and Excel (Business Skills). Pearsons Education, Inc. 2026. ISBN 978-0-13-824472-9

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.