Power BI Desktop performance tunning

Ing. Martin Haman

MCSA: BI Reporting, MOS Master,
MCAS Master, MCP, MPS, ECDL

Martin.Haman@gmail.com

www.linkedin.com/in/MartinHaman

Osnova optimalizace

- Data load/refresh settings
- Query folding
- Optimalizace obecně
- DAX vs. Query
- DAX columns vs. DAX measures
- Diagnostika log souborů v Query
- Diagnostika DAX výpočtů

Základní dotazy

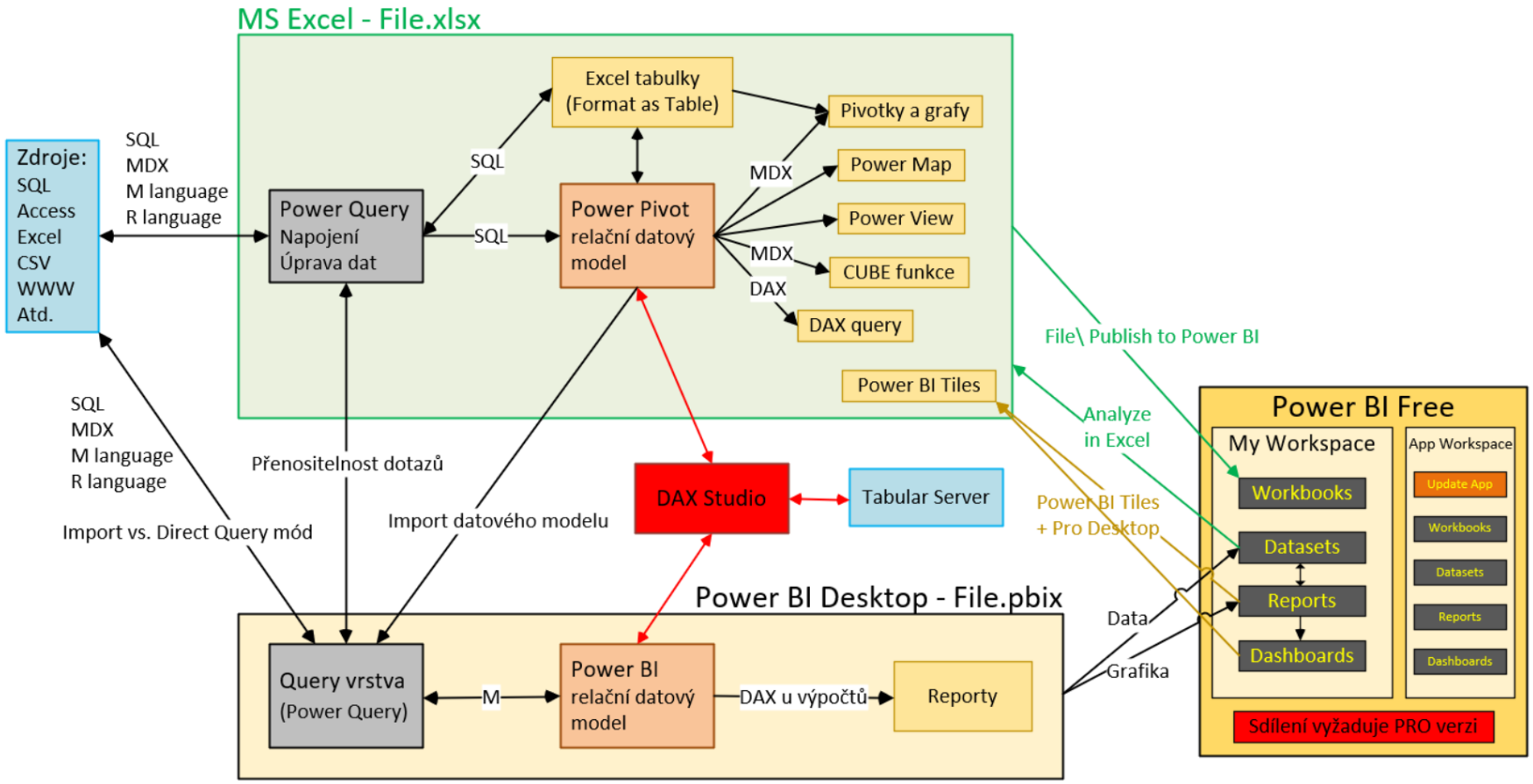
- Objemy dat?
  - Počet tabulek ve zdroji!
  - Počet sloupců v tabulkách (ovlivněno počtem tabulek)
  - Počet řádků v tabulkách
- Počet sloupců s unikátními hodnotami
- Datové typy

Pozice Power BI na trhu 2018

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms

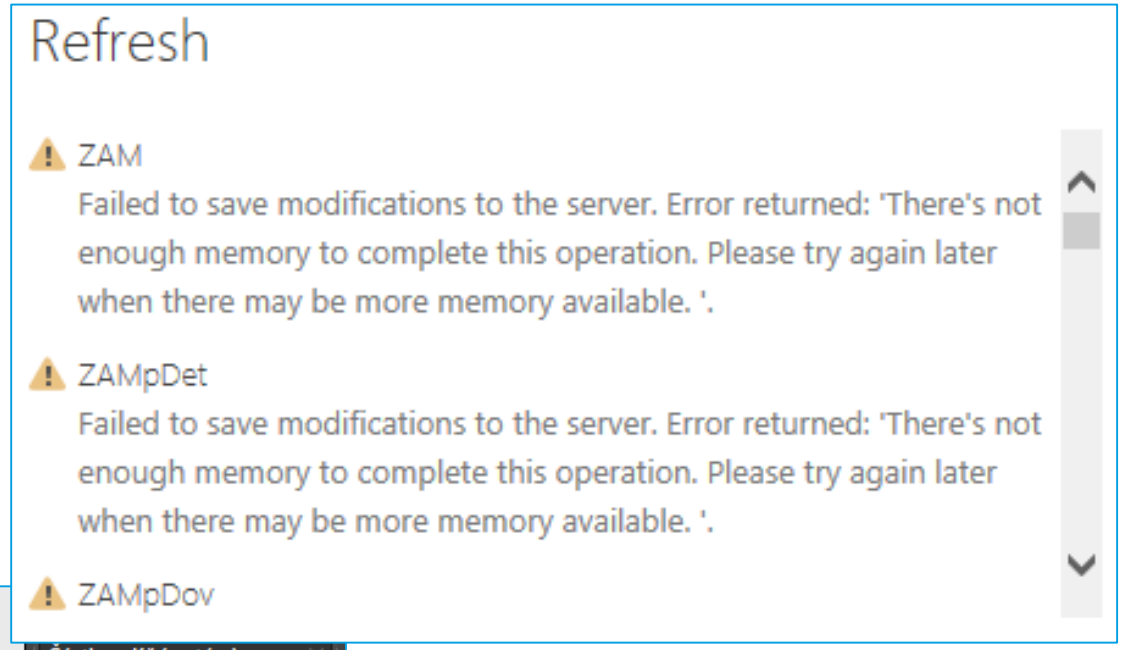
Source: Gartner (February 2018)

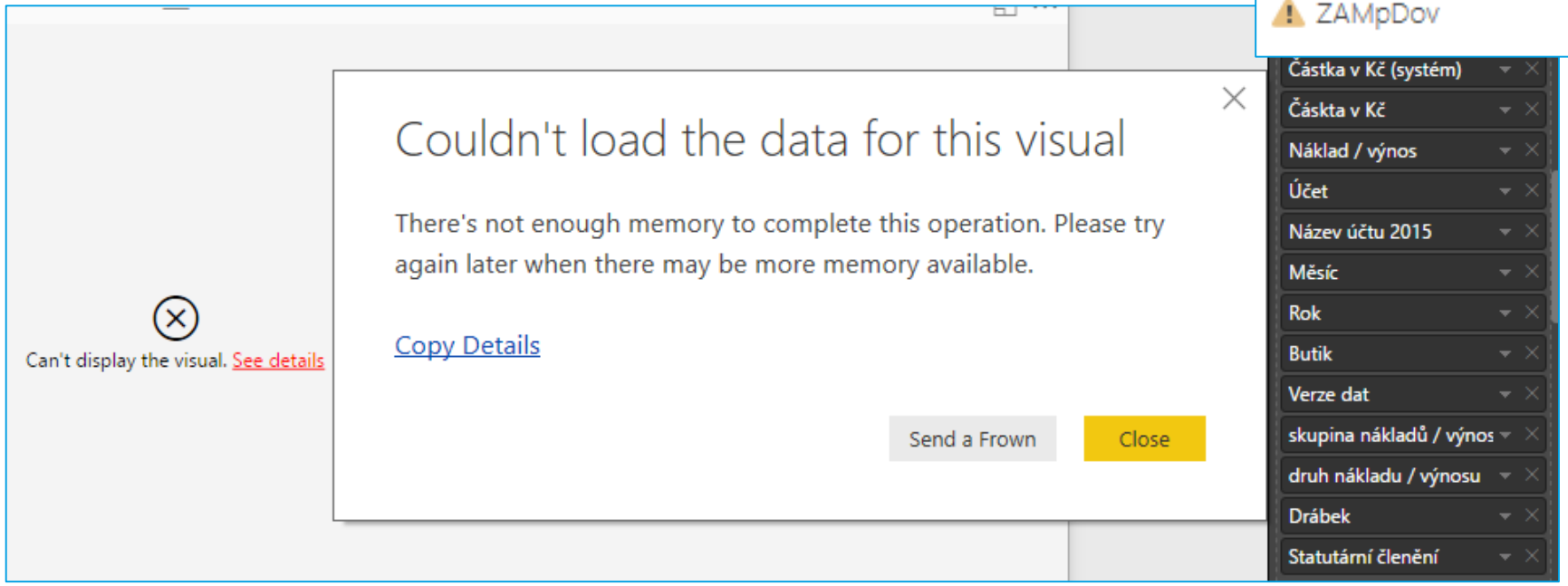
Zjednodušené schéma Power BI



Je to pomalé…

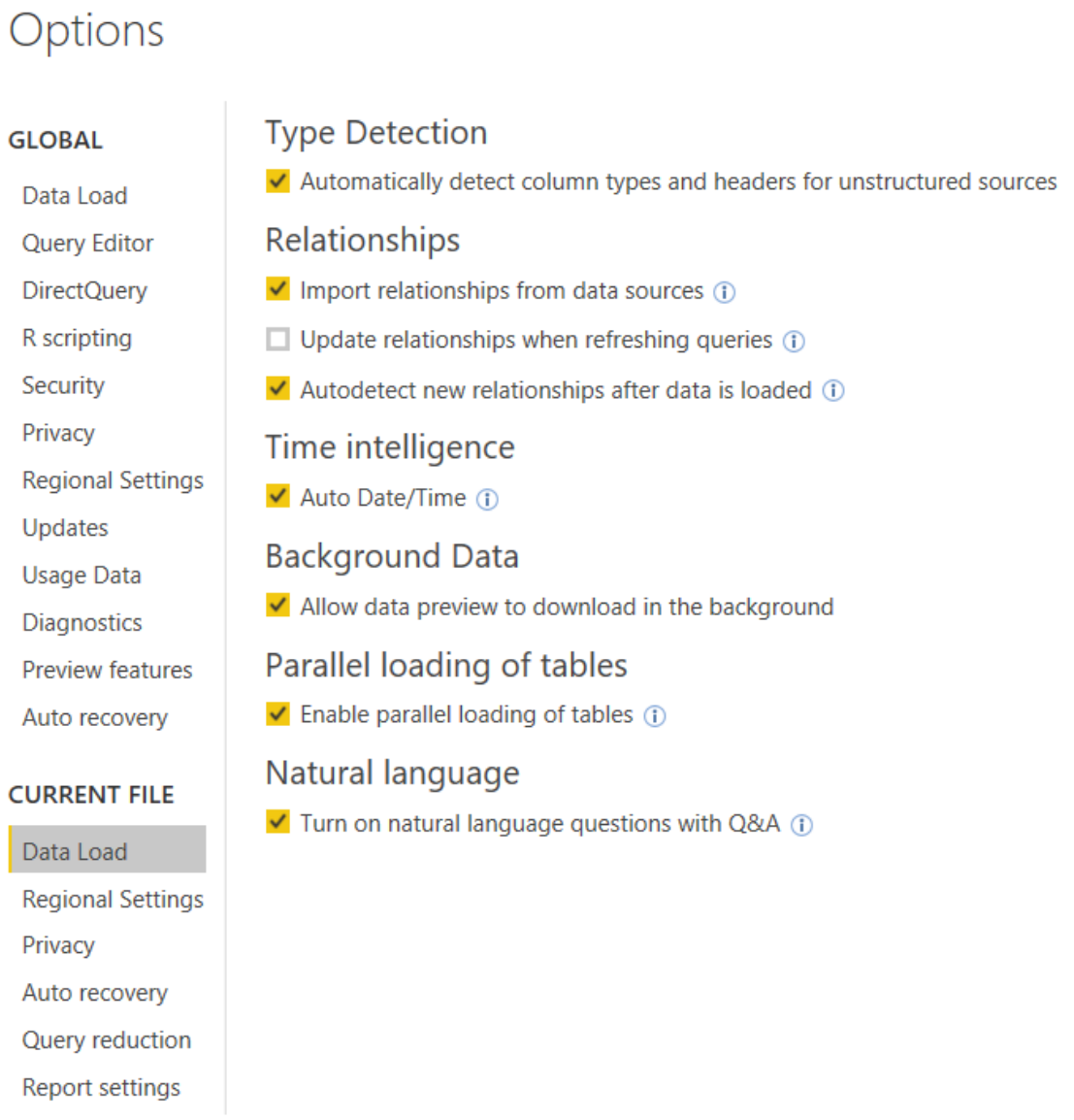
- 16BG RAM + SSD disk a stále problém?
- Máme řešení ☺





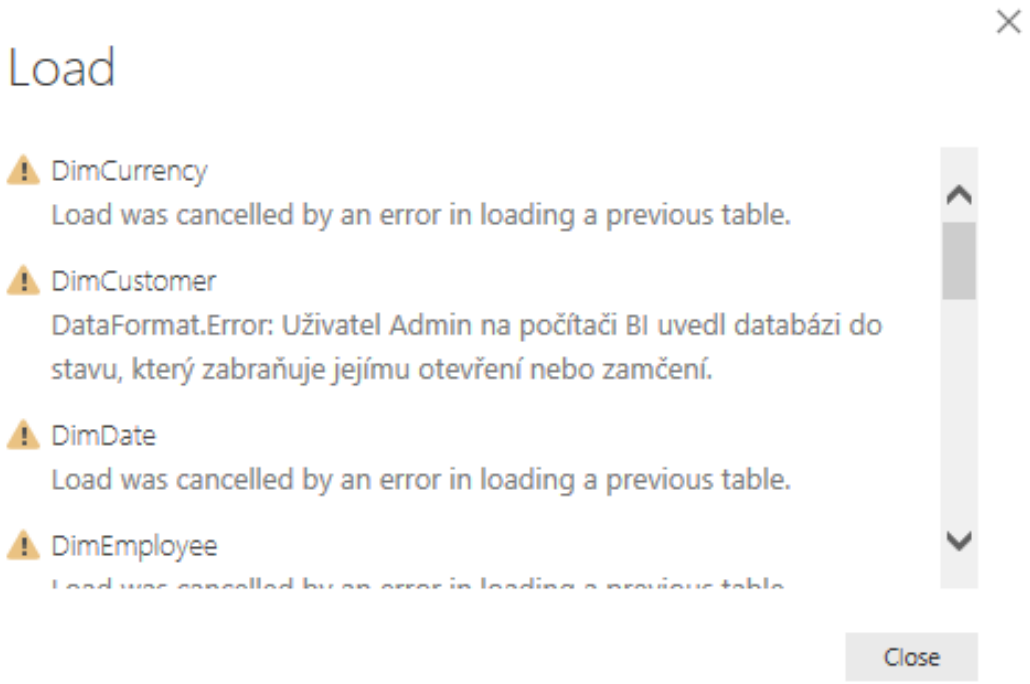
Data load settings

- Některé volby můžou pomoct (import relationships)
- Některé můžou komplikovat život
- Obecně platí čím méně, tím lépe



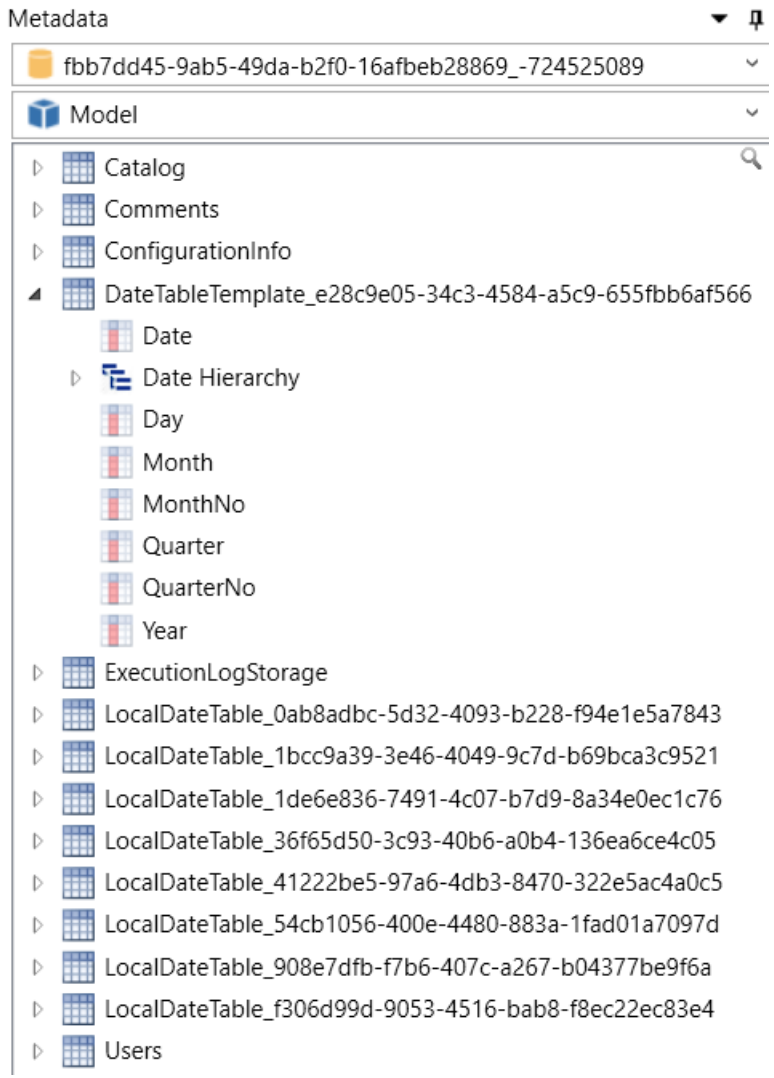
Parallel loading

- Chybové hlášení neodpovídá realitě a za základě chyby v jedné tabulce ukončí načítání ostatních
- Řešením je vypnutí paralelního načítání tabulek (bude nepatrně pomalejší), ale povede se!



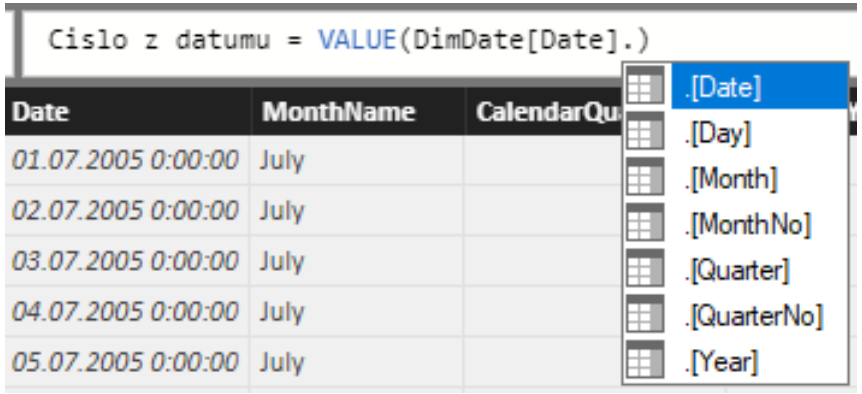
Auto Date/Time

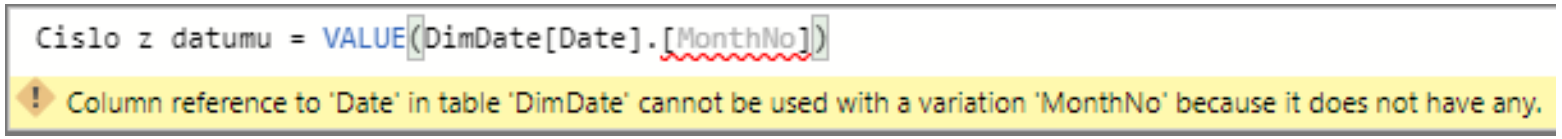
- Pokud máme hodně tabulek a v nich se objevují datumové sloupce, tak lze urychlit načítání a aktualizace dat vypnutím Auto Date/Time
- Je dobré mít v modelu vlastní kalendář pro DAX časové fce.



Auto Date/Time

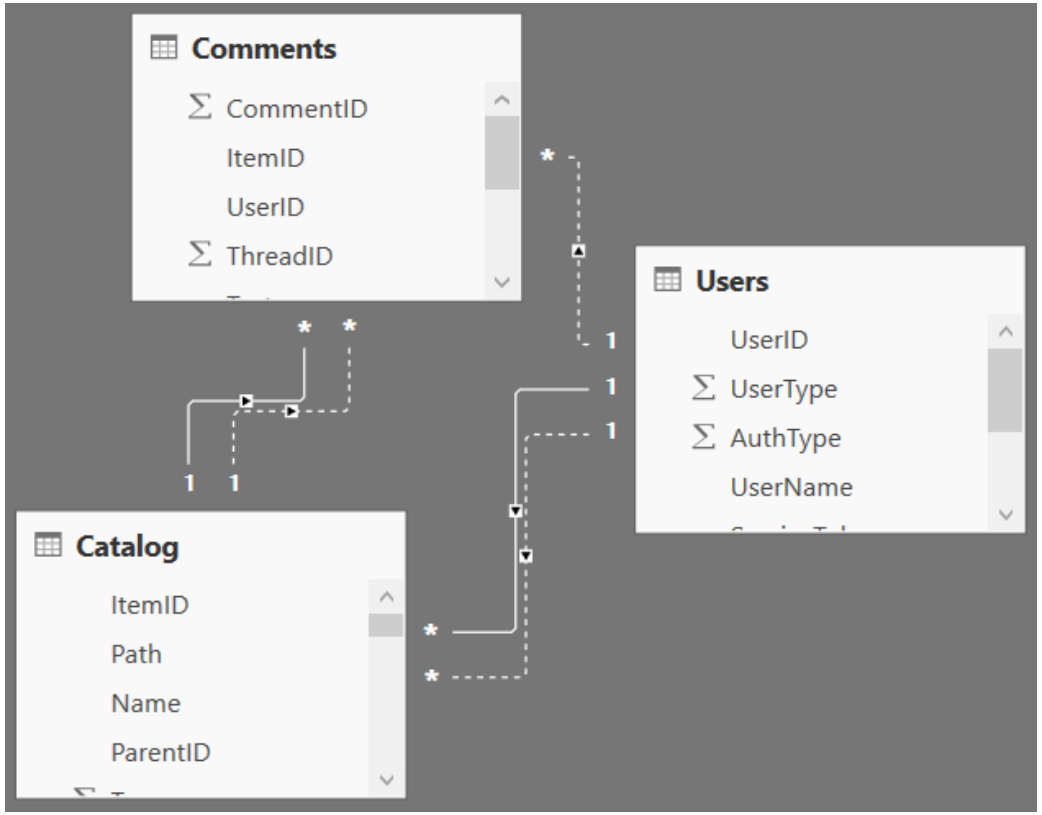
- S vypnutím volby hrozí, že nebudou fungovat některé DAX výpočty, protože zmizí datumové hierarchie na pozadí!





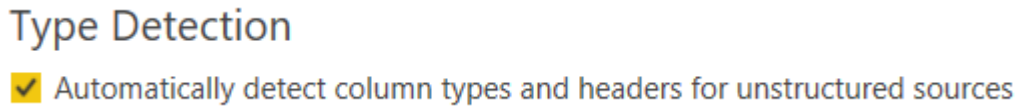
Autodetect new relationships

- Může způsobit naprostý chaos
- Kolikrát se stává, že neaktivní relace může být OK
- Je nutná pořádná kontrola modelu
- Ideální je mít vypnuto

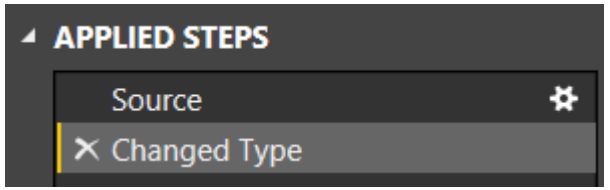


Automatically detect column types

- Automatická detekce je uplatňovaná ještě v Query okně



- Způsobuje automatický odhad datového typu sloupců zdroje na základě analýzy prvních 200 řádků



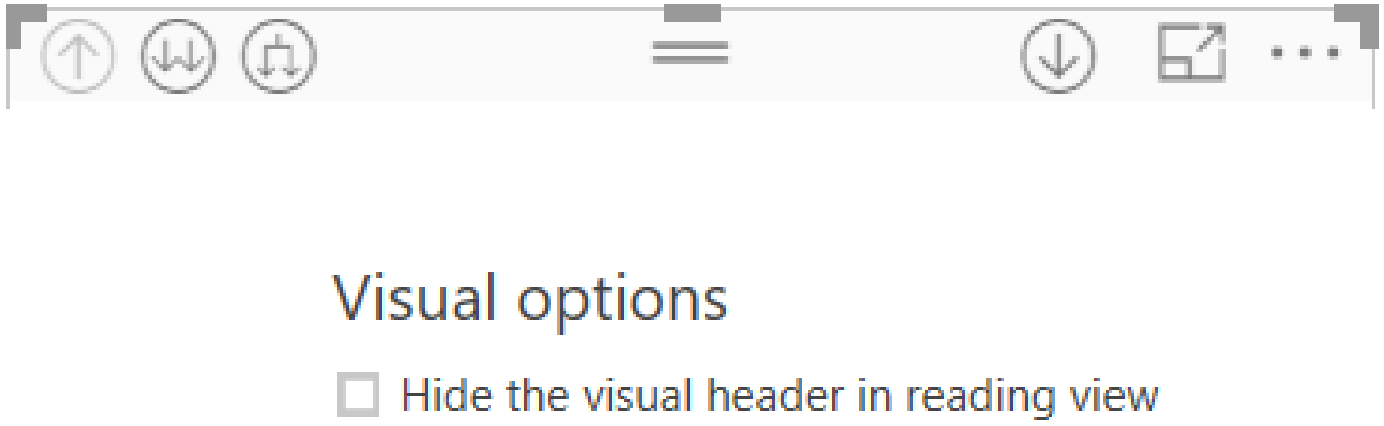
- Může zapříčinit vypnutí query folding!

Allow data preview to download in the background

- Možnost vypnutí této funkce byla dodatečně přidána vývojovým týmem
- Vypnutí výrazně redukuje množství dat, které je nutné zpracovat a ušetří RAM a procesor
- Její potenciál odhalíme při načítání velkého množství dat

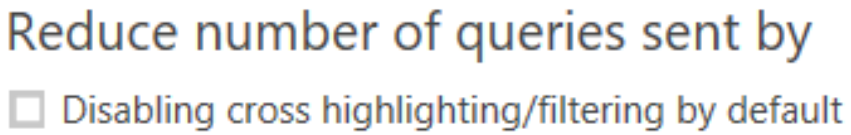
Hide the visual header in reading view

- Pokud nechceme, aby někdo maximalizoval, drilloval po publikování na web, tak máme k dispozici volbu File\ Options and settings\ Options\ Report Settings



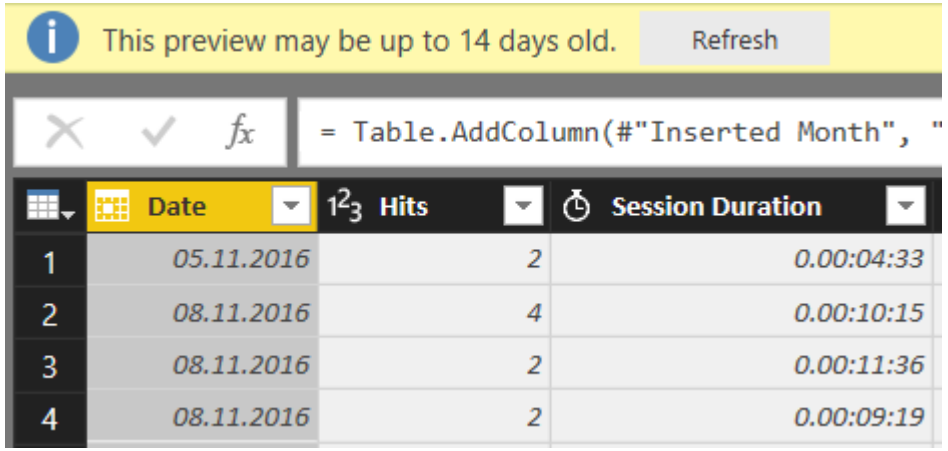
Disabling cross highlighting/filtering

- Máte desítky vizuálů na stránce a načítání je pomalé?
- Není potřeba, aby se navzájem ovlivňovaly?
- Je možné vypnout v File\ Options and settings\ Options\ Query Reduction



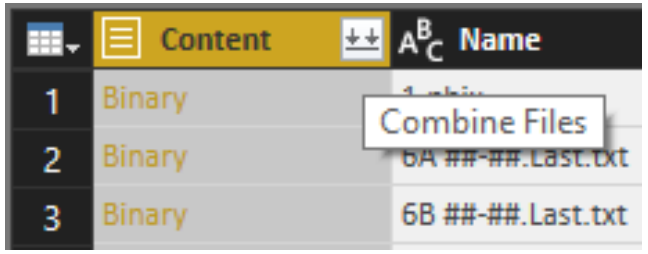
Data Cache management options

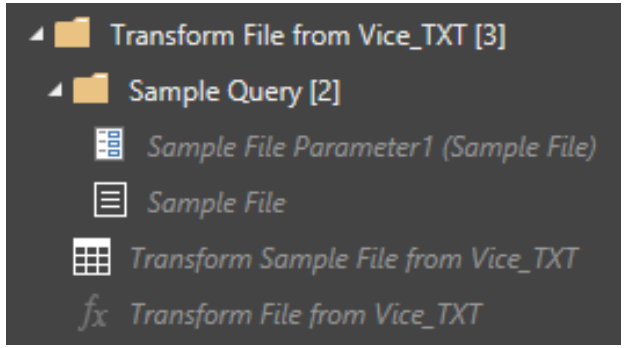
- Souvisí plně s Query editorem a zobrazením dat ze zdroje
- Pokud pracujeme pořád s jedním zdrojem, nehýbeme posuvníkem (v základu načítá prvních cca 200 řádků) a nevadí starý náhled na data, tak se cache nezvětšuje



Combine

- Pokud se někde objeví Combine…





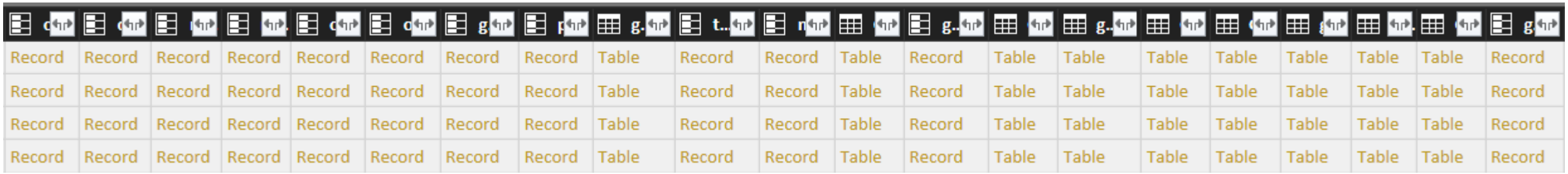
  …tak popřemýšlejte, zda neudělat jinak
- Plánovaná aktualizace reportu v Office 365 se neprokouše skrz takto vytvořenou funkci atd.

Table.AddKey

- Primární klíče v M tabulkách pomocí **Table.AddKey** (případně lze obejít přes Remove Duplicates)
- Sloučíme 2 tabulky (např. left outer join) kde první tabulka je cca 100 MB CSV, druhá tabulka je obyčejný číselník
- Při následné agregaci v takto vytvořeném dotazu dojde při aktualizacích k namnožení 100MB pro jednotlivé položky z číselníku (pokud jich je 10, tak 10* 100MB)
- Velmi pomalá aktualizace a spousta dat v RAM

„žluté“ sloupce

- Někdy si nelze nevšimnout „žlutých“ sloupců na konci dotazů (value/record strana 1, table strana N)



- Vlastně se jedná o nachystané joiny (drží v RAM)

```
#"Merged Queries" = Table.NestedJoin(
```

- Při větším množství umí opět VELMI zpomalovat

„žluté“ sloupce

- Expandování sloupce v SQL zápisu

Native Query

```
select [$Outer].[ProductKey] as [ProductKey],
    [$Outer].[ProductSubcategoryKey2] as [ProductSubcategoryKey],
    [$Outer].[ProductName] as [ProductName],
    [$Outer].[Color] as [Color],
    [$Outer].[ListPrice] as [ListPrice],
    [$Outer].[Size] as [Size],
    [$Outer].[DealerPrice] as [DealerPrice],
    [$Outer].[Class] as [Class],
    [$Outer].[Style] as [Style],
    [$Outer].[ModelName] as [ModelName],
    [$Outer].[LargePhoto] as [LargePhoto],
    [$Inner].[ProductSubcategoryName] as [ProductSubcategoryName]
from
(
    select [_].[ProductKey] as [ProductKey],
        [_].[ProductSubcategoryKey] as [ProductSubcategoryKey2],
        [_].[ProductName] as [ProductName],
        [_].[Color] as [Color],
        [_].[ListPrice] as [ListPrice],
        [_].[Size] as [Size],
        [_].[DealerPrice] as [DealerPrice],
        [_].[Class] as [Class],
        [_].[Style] as [Style],
        [_].[ModelName] as [ModelName],
        [_].[LargePhoto] as [LargePhoto]
    from [DimProduct] as [_]
) as [$Outer]
left outer join [DimProductSubcategory] as [$Inner] on ([$Outer].[ProductSubcategoryKey2] = [$Inner].[ProductSubcategoryKey])
```

```
#"Merged Queries" = Table.NestedJoin(_DimProduct,{"ProductSubcategoryKey"},DimProductSubcategory,{"ProductSubcategoryKey"},
#"Expanded DimProductSubcategory" = Table.ExpandTableColumn(#"Merged Queries", "DimProductSubcategory", {"ProductSubcategor
```

Query folding

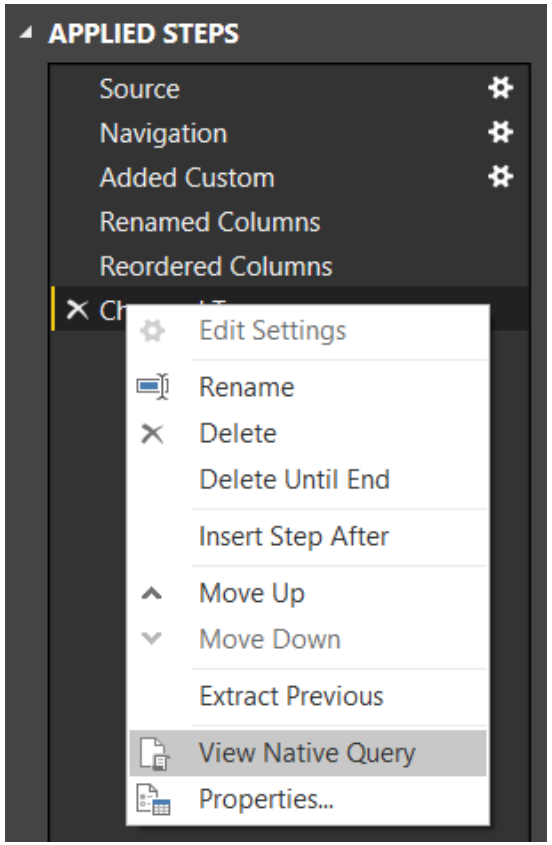
- Samotné optimalizační triky v podobě data load settings nemusejí stačit
- Lepší je dělat transformace přímo ve zdroji (na serveru) než je stahovat do cache (síťový provoz a zahlcení RAM)
- Query folding:
  Power BI > Power Query M Script > překlad do T-SQL > SQL Server Databáze
- Query folding částečný:
  Power BI > Power Query M Script > Cache Local > předklad některých operací do T-SQL > SQL Server Databáze

Query folding

- Query folding je ve výchozím nastavení zapnutý
  (v případě napojení na relační zdroj)
- Některé M operace nejsou podporovány při překladu
  do T-SQL, např.:
  Merge columns, které využívá funkce Text.Combine
  (klasika v podobě & funguje)
  Kombinování více zdrojů v rámci jednoho dotazu
- Proto je určitě lepší popřemýšlet nad postupem jednotlivých
  operací a nad provedením dané operace!

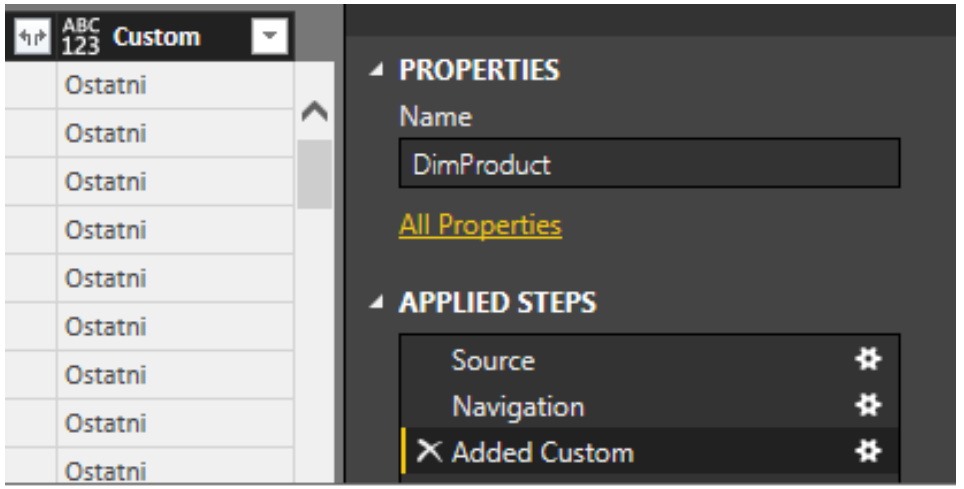
Native Query

- Zda daný krok ještě překládá nebo už ne poznáme pravým klikem na krok v Applied steps a musí být aktivní možnost „View Native Query“:



Native Query

- Ukázka konstrukce IF překládané do SQL (switch úplně dole)

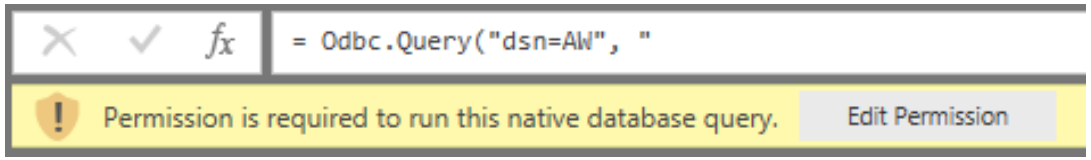


Native Query

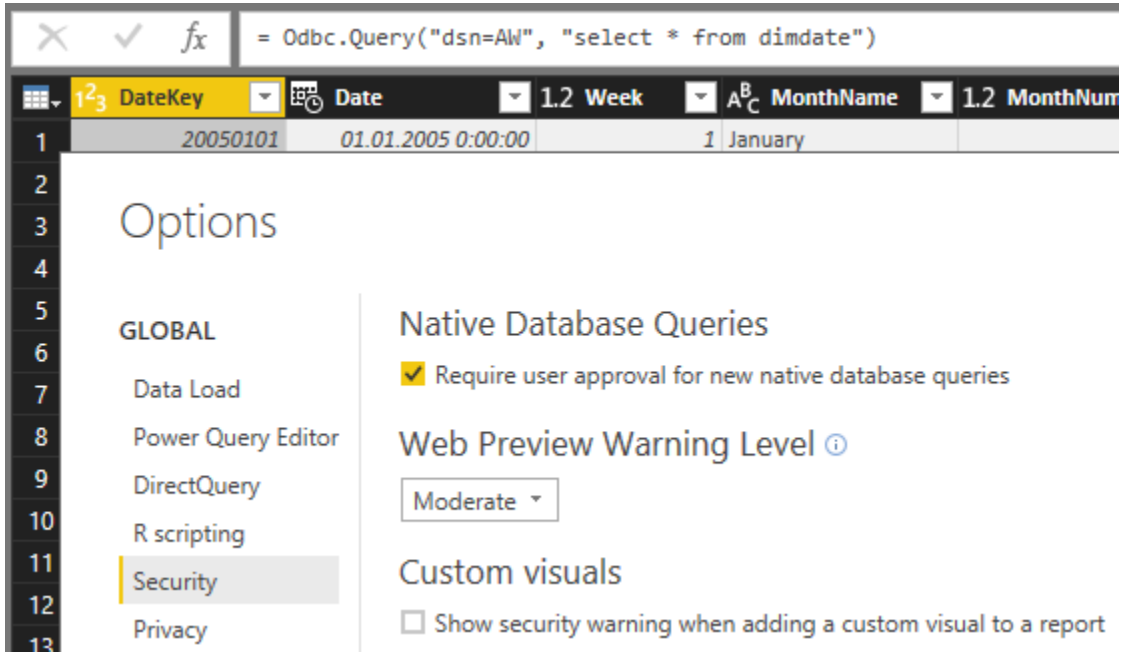
```
select [_].[ProductKey] as [ProductKey],
    [_].[ProductSubcategoryKey] as [ProductSubcategoryKey],
    [_].[ProductName] as [ProductName],
    [_].[Color] as [Color],
    [_].[ListPrice] as [ListPrice],
    [_].[Size] as [Size],
    [_].[DealerPrice] as [DealerPrice],
    [_].[Class] as [Class],
    [_].[Style] as [Style],
    [_].[ModelName] as [ModelName],
    [_].[LargePhoto] as [LargePhoto],
    switch(
        [_].[ModelName] is null, 'Ostatni',
        1, [_].[ModelName]
    ) as [Custom]
from [DimProduct] as [_]
```

Native Query

- Při tvorbě dotazů může při každé změně chtít oprávnění

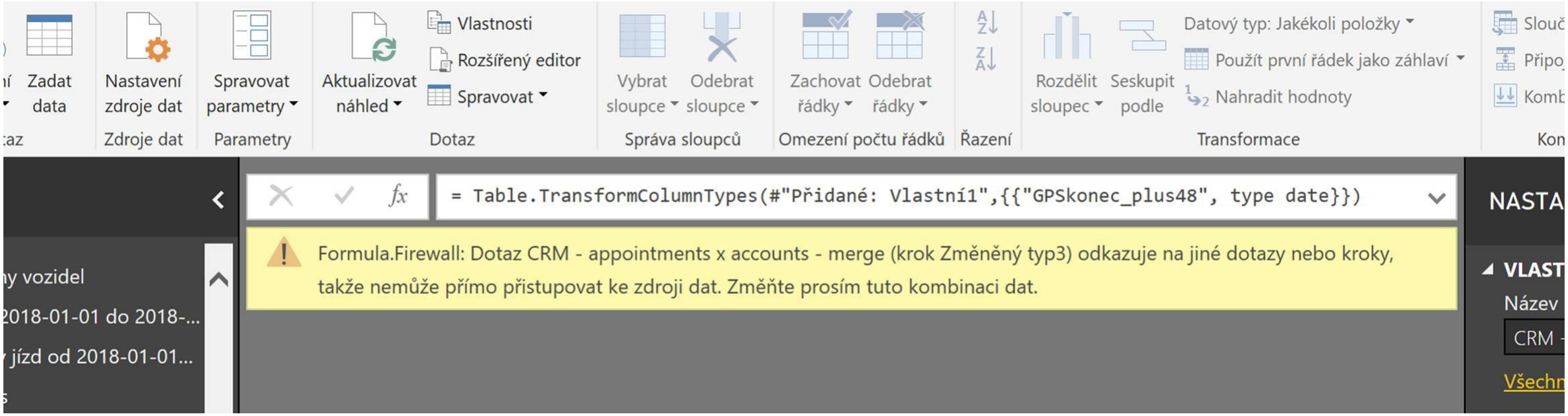


- Vypnout se dá přes Options\ Security\ Require user approval



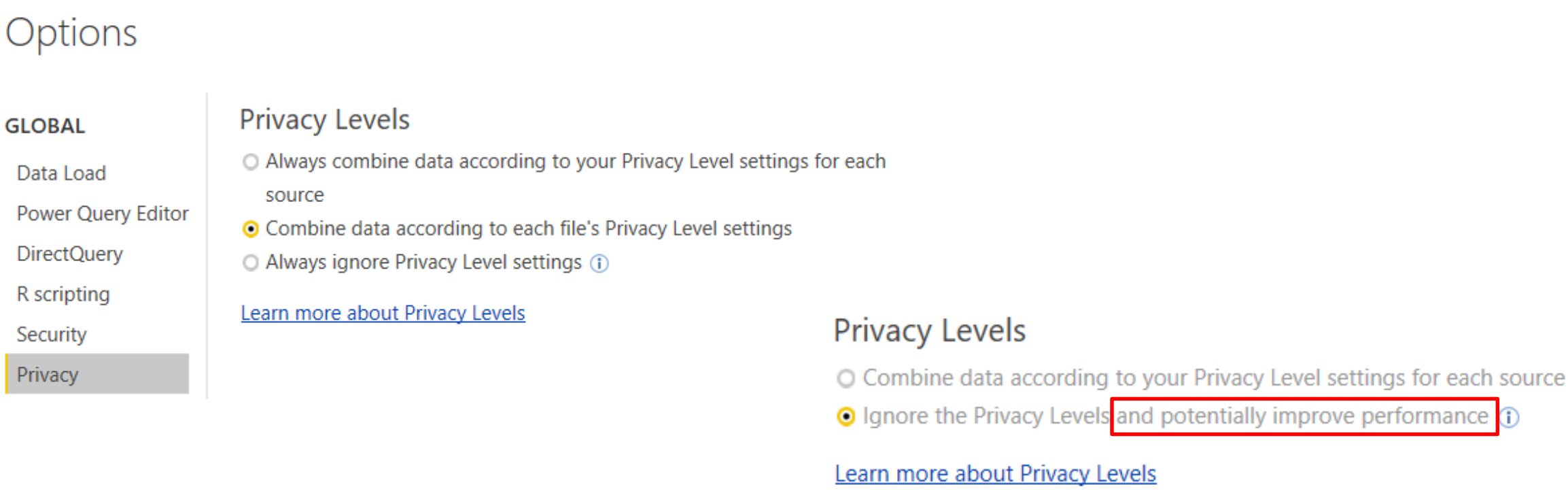
Formula.Firewall

- Může se objevit např. když děláme merge více dotazů, které jsou ze zdrojů vyžadujících nastavení úrovně soukromí



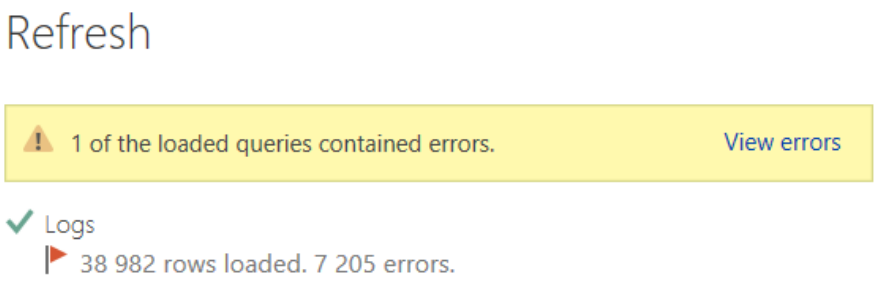
Formula.Firewall

- Na vypnutí Firewall funkce funguje „Always ignore Privacy Level settings“

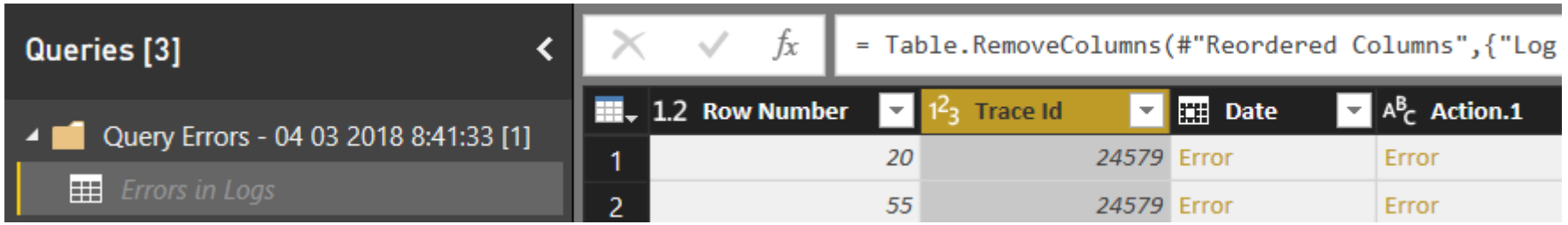


Jak na chyby?

- V průběhu načítání/aktualizace se může objevit dialog s chybou



- View errors ukáže číslo řádku s chybou
- Často problém s datovými typy v kombinaci s národním prostředím



Start aplikace

- První startuje Power BI Desktop
- Pak SQL Server Analysis Services
- Následně důležitý CefSharp.BrowserSubprocess = Chromium (PBID používá na vykreslení vizuálů)
- Mashup čte ze souboru DataMashup a potřebuje pro navigaci v datech (žluté sloupce atd.), např.:

  OData/https:\\/\\/firma.api.crm4.dynamics.com\\/api\\/data\\/v8.2\\//opportunities.{ds_probability,1}","
  Načte všechny sloupce tabulky a jejich pořadí + KeyColumn

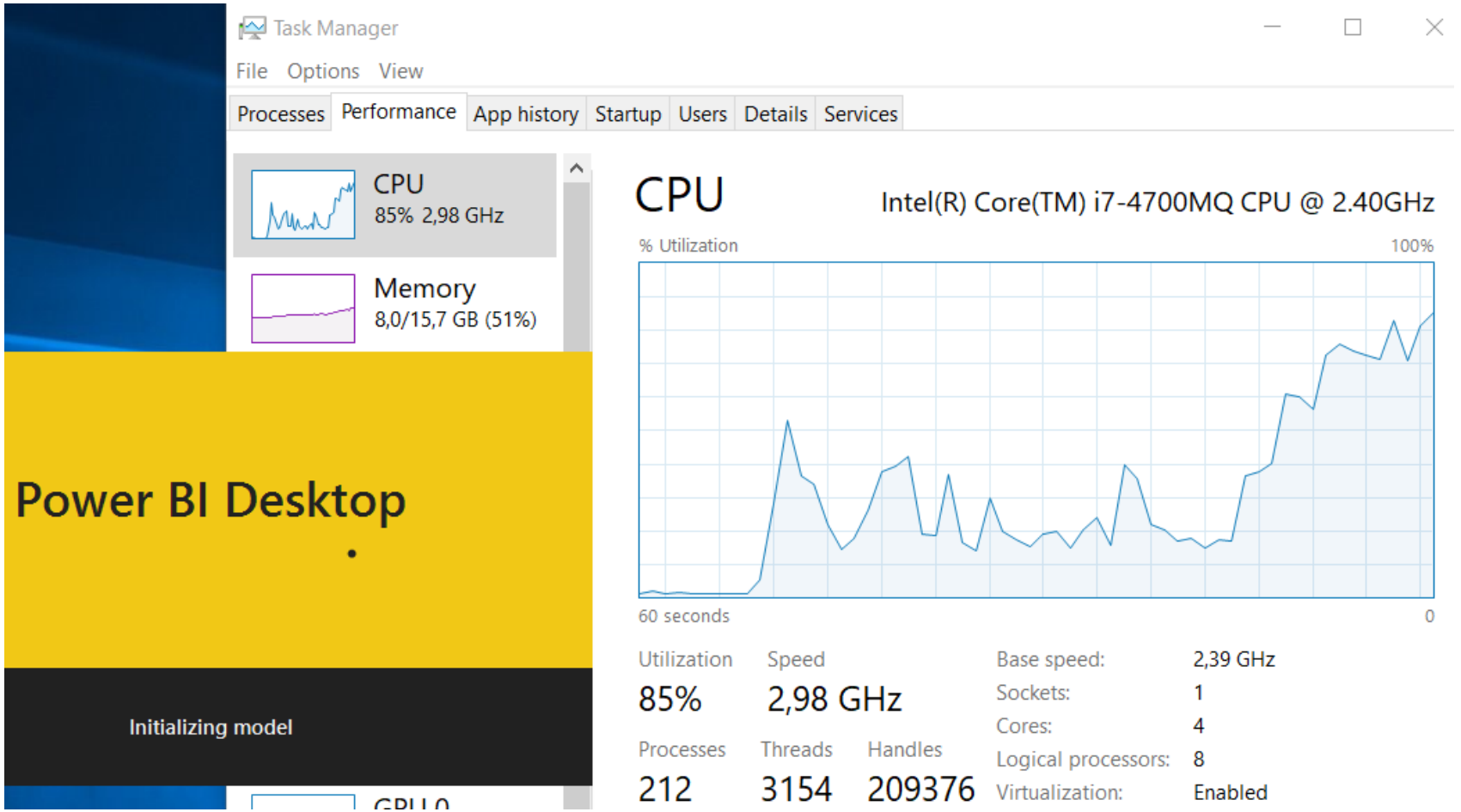
Start aplikace

- Počet Odata odkazů v Mashup souboru při napojení na 6 Dynamics tabulek mám 1605 !!!
- Z toho 928 „žlutých“ sloupců => 928 joinů, které jsou nachystané v RAM

- Máš hodně unavený počítač:
  Start Mashup je výkonově náročný a pak k tomu přidáme Chromium kvůli vykreslení (+ dopočítání measures)

Start aplikace

- Start Dynamics reportu



Optimalizace obecně

- Pokud budeme bezmyšlenkovitě používat „neomezený“ prostor datového modelu, tak můžeme na běžném kancelářském stroji rychle narazit na výkonnostní strop

Name	23% CPU	49% Memory	0% Disk	0% Network	1% GPU
Power BI Desktop (10)	20,7%	1 938,1 MB	0,6 MB/s	0 Mbps	0,4%
Microsoft Mashup Evaluation ...	19,9%	241,4 MB	0,2 MB/s	0 Mbps	0%
Microsoft Power BI Desktop	0,6%	1 029,4 MB	0,4 MB/s	0 Mbps	0,4%
CefSharp.BrowserSubprocess	0,1%	127,4 MB	0 MB/s	0 Mbps	0%
Microsoft Mashup Evaluation ...	0%	49,5 MB	0 MB/s	0 Mbps	0%
CefSharp.BrowserSubprocess	0%	32,5 MB	0 MB/s	0 Mbps	0%
Microsoft Mashup Evaluation ...	0%	52,4 MB	0 MB/s	0 Mbps	0%
Microsoft Mashup Evaluation ...	0%	52,6 MB	0 MB/s	0 Mbps	0%
CefSharp.BrowserSubprocess	0%	117,4 MB	0 MB/s	0 Mbps	0%
Microsoft SQL Server Analysis ...	0%	230,9 MB	0 MB/s	0 Mbps	0%
Console Window Host	0%	4,7 MB	0 MB/s	0 Mbps	0%

Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

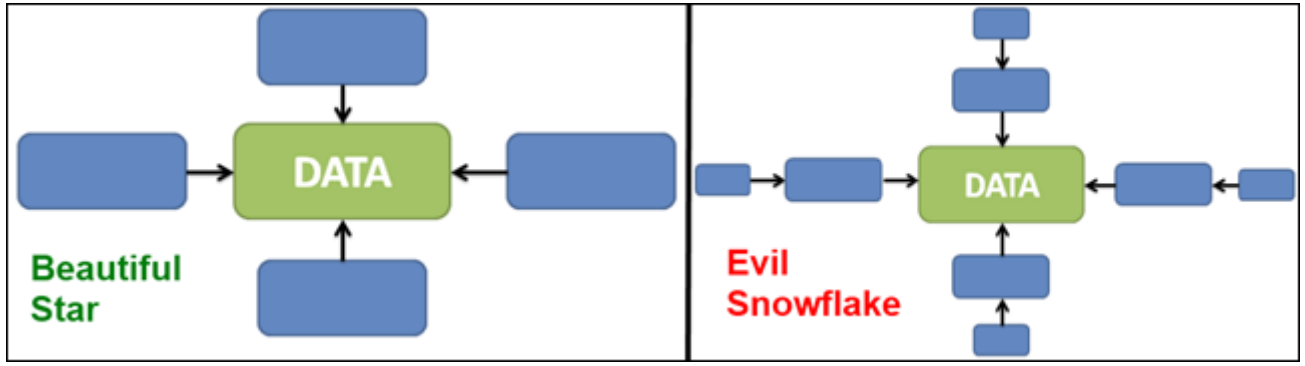
Name	85% CPU	64% Memory
Microsoft Excel	77,3%	2 390,1 MB
System	6,3%	0,1 MB

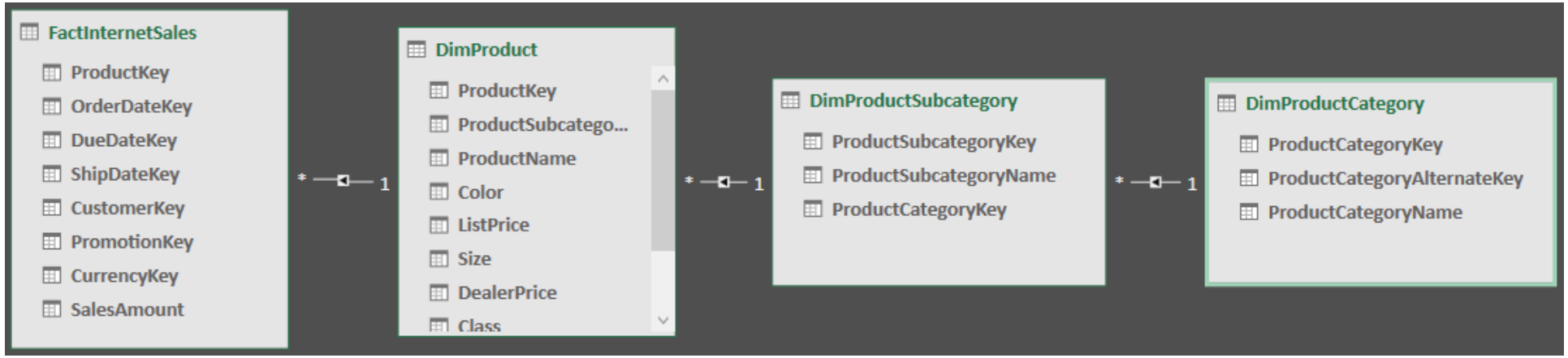
Optimalizační tipy

- Data můžeme spravovat s menšími nároky na místo a výkon počítače, když se budeme držet:
  1. 64 bit. Aplikace
  2. Používat STAR schéma
  3. Jednoznačný identifikátor řádku
  4. Limitovat počet řádků a sloupců v tabulkách
  5. Pozor na sloupce s vysokým počtem unikátních hodnot (datum a čas, desetinné čísla, ID)
  6. Používat measures místo calculated columns (viz. dále)
  7. Omezit počet slicerů v konkrétním souboru (při každém výběru musí model přepočítat vše)
  8. Tvořit slicery jen nad DIM tabulkami (FACT mají desetinné)
  9. Zakázat cross-filtering ve slicerech

Star vs. Snowflake model

- Hlavně Power Pivot nesnáší dobře velké mezitabulky



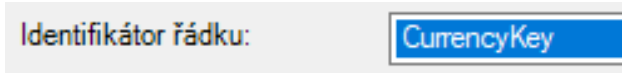


Star vs. Snowflake model

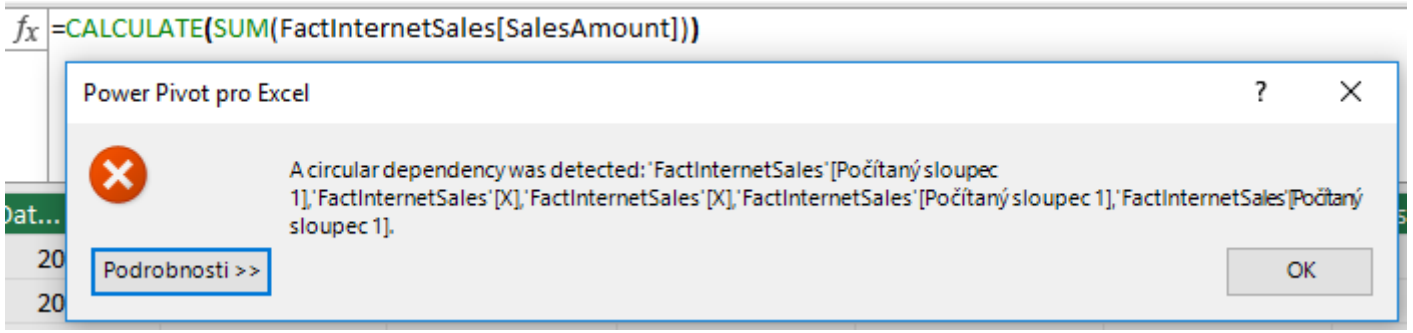
- V naší modelové databázi máte také tento problém, ale nad malým objemem dat!
- Kdybychom měli např. 3 tabulky:
  - Kategorie (1000 řádků) => Podkategorie (20000 řádků) => Produkty (2000000 řádků)
  - Když dáme v Pivotce slicer na Kategorie a použijeme data z tabulky Produkty, tak zapříčiníme pravděpodobně pád Excelu, v lepším případě neuvěřitelně nabobtná velikost souboru a vše bude velmi pomalé!
  - Řešení 1 – nepoužívat Snowflake schéma
  - Řešení 2 – RELATED (nebo PQ) dotáhne Kategorie do tabulky Produkty a slicer děláme jen nad jednou tabulkou

Jednoznačný identifikátor řádku

- Každá tabulka MUSÍ mít primární klíč nebo alespoň jedinečný identifikátor řádku v "Chování tabulky"



- Jinak jsme vystaveni problémům v případě použití measures nebo sloupců s Calculate (cyklická závislost)



- Funkce na obrázku řeší sumu všech řádků v tabulce, které mají stejné hodnoty ve všech sloupcích

Jednoznačný identifikátor řádku

- V PBID zatím tato volba chybí, ale lze nahradit upraveným zápisem druhé a další Calculate:

  CALCULATE(SUM(TabProdeje[Celkem]);
  ALLEXCEPT(TabProdeje;TabProdeje[PredchazejiciVypocetCalculate]))

- Aplikováním filtru nedojde k výběru sloupce s Calculate a tím se nezacyklíme

Omezení počtu řádků

- Máme sloupec ID, který obsahuje 100 milionů unikátních čísel >> ve VertiPaq cca 3GB
- Pokud však sloupec rozdělíme >> ve VertiPaq se dostaneme na cca 200MB
  rozdělení může být do dvou sloupců o 10000:

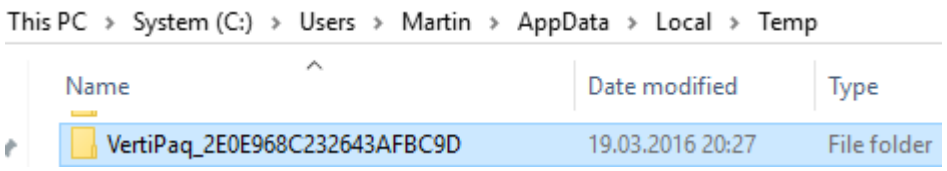
```
SELECT
    TransactionID / 10000 AS TransactionHighID,
    TransactionID % 10000 AS TransactionLowID,
    Quantity,
    Price
FROM Fact
```

Následuje složení:

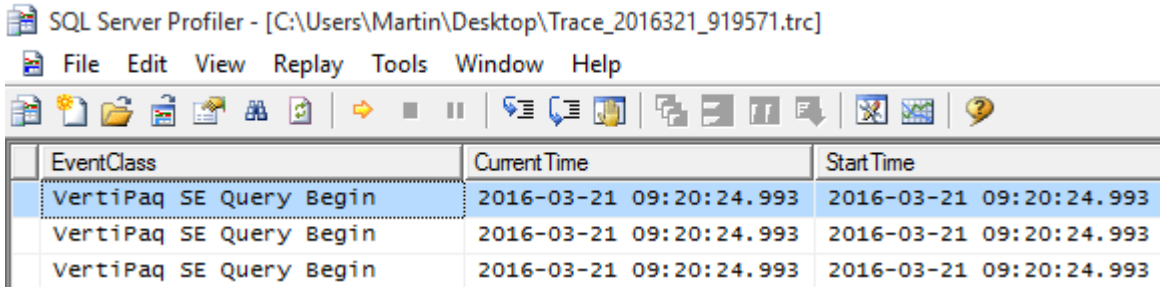
Fact[TransactionID]:=IFERROR(VALUES(Fact[TransactionHighID]) *10000 + VALUES(Fact[TransactionLowID]);BLANK())

VertiPaq engine

- Datový model oficiálně využívá „xVelocity in-memory Analysis Services“
- Pracovním názvem byl však dlouhou dobu **VertiPaq**
- Interně je proto pořád vidět VertiPaq



- Ve skutečnosti dotazovací engine vykonává VertiPaq dotazy, ne xVelocity dotazy



VertiPaq engine

- DAX fungují v SS Analysis Services, Power BI (server i local) a v Power Pivot v1 (Excel 2010), PowerPivot v2 (Excel 2013 a 2016)
- Technicky je Power Pivot lokální instance SSAS Tabular – v SSMS jsme schopni udělat „back up“ databáze (databaze.ABF) => lze přepsat na item.data
- Power Pivot je tedy SSAS, který běží přímo v Excelu
- Při načítání dat do paměti (load nebo refresh) se děje transformace na interní VertiPaq datovou strukturu

VertiPaq engine

- Postup při načítání dat (viz další snímky):
  1. Čtení dat a transformace do sloupcové struktury VertiPaq, šifrování a komprese každého sloupce
  2. Tvorba slovníků a indexů pro každý sloupec
  3. Tvorba relací
  4. Počítání a komprese všech počítaných sloupců

- Poslední dva kroky můžou být i v opačném pořadí (relace může být nad počítaným polem) (počítané pole může být závislé na relaci – RELATED)

VertiPaq engine

- Čtení dat a transformace do sloupcové struktury

ID	Jmeno	Barva	Cena
1	Kamera	Červená	112 €
2	Fotoaparát	Červená	97 €
3	Telefon	Bílá	100 €
4	Konzole	Černá	112 €
5	TV	Modrá	1 240 €
6	CD	Červená	40 €
7	Monitor	Modrá	45 €
8	DVD	Černá	120 €
9	Klávesnice	Černá	120 €
10	Myš	Červená	50 €
11	Tiskárna	Bílá	180 €
12	USB	Modrá	30 €

ID
1
2
3
4
5
6
7
8
9
10
11
12

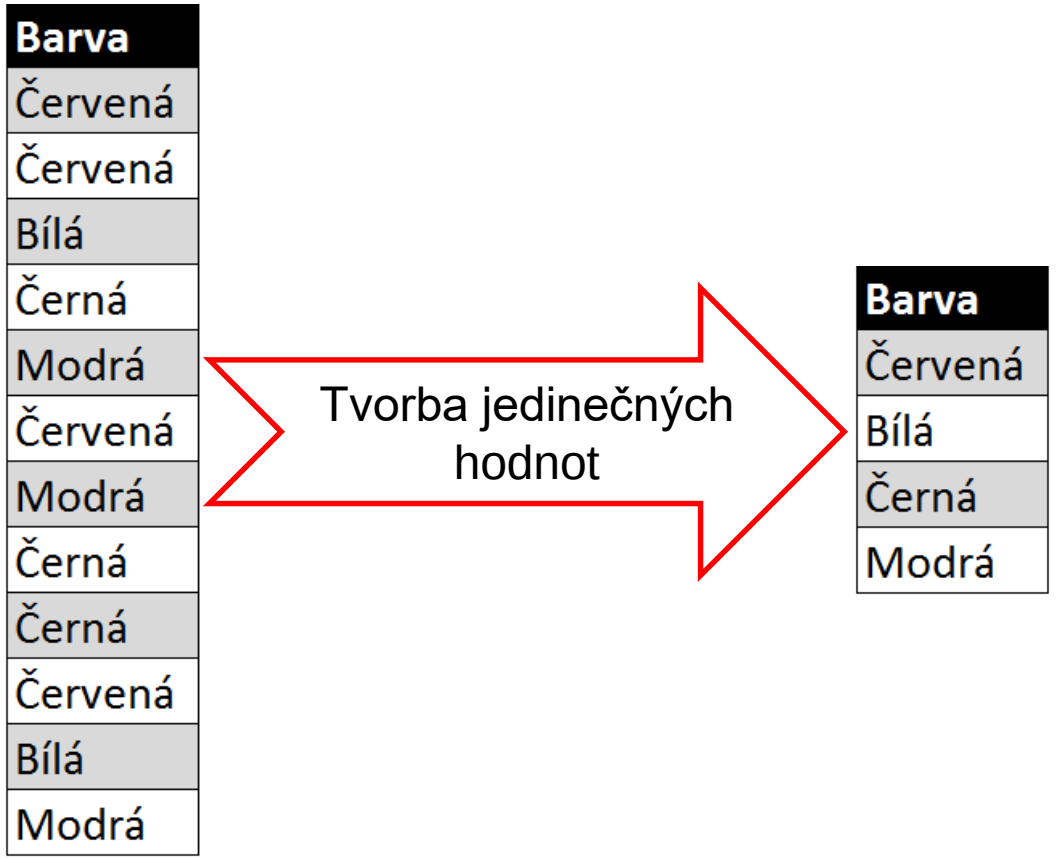
Jmeno
Kamera
Fotoaparát
Telefon
Konzole
TV
CD
Monitor
DVD
Klávesnice
Myš
Tiskárna
USB

Barva
Červená
Červená
Bílá
Černá
Modrá
Červená
Modrá
Černá
Černá
Červená
Bílá
Modrá

Cena
112 €
97 €
100 €
112 €
1 240 €
40 €
45 €
120 €
120 €
50 €
180 €
30 €

VertiPaq engine

- Pro jednotlivé sloupce z tabulky následuje tvorba slovníku jedinečných hodnot

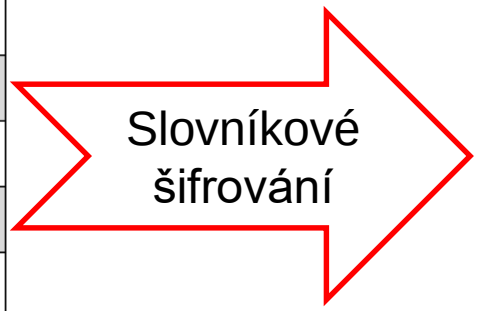


VertiPaq engine

- Nahrazení jedinečných hodnot indexy
- Všechny sloupce tak budou číselně (integer)

Barva	ID
Červená	0
Bílá	1
Černá	2
Modrá	3

Barva
Červená
Červená
Bílá
Černá
Modrá
Červená
Modrá
Černá
Černá
Červená
Bílá
Modrá

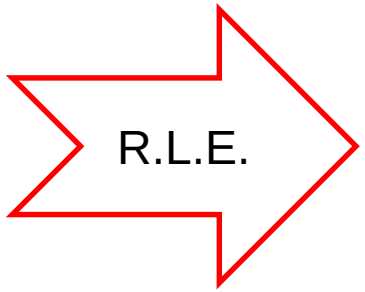


Barva ID
0
0
1
2
3
0
3
2
2
0
1
3

VertiPaq engine

- Duplicitní výskyty indexů jsou následně seřazeny

Barva ID
0
0
0
0
1
1
2
2
2
3
3
3



Run Length Encoding - máme seřazeno a víme počty jednotlivých indexů => velmi zredukován počet hodnot

Barva ID	Start	Počet
0	1	4
1	5	2
2	7	3
3	10	3

VertiPaq engine

- Výsledné řešení je paměťově úspornější a rychleji prohledatelné (start sloupec není třeba)
- Vše závisí na poměru počtu hodnot a počtu unikátních hodnot

Barva ID	Počet
0	4
1	2
2	3
3	3

VertiPaq engine

- 1.obr má 4 unikátní členy (QTR), 2. má 7232 členů

SQL Server Profiler - [C:\Users\Martin\Desktop\Trace_2016321_2051561.trc]

File Edit View Replay Tools Window Help

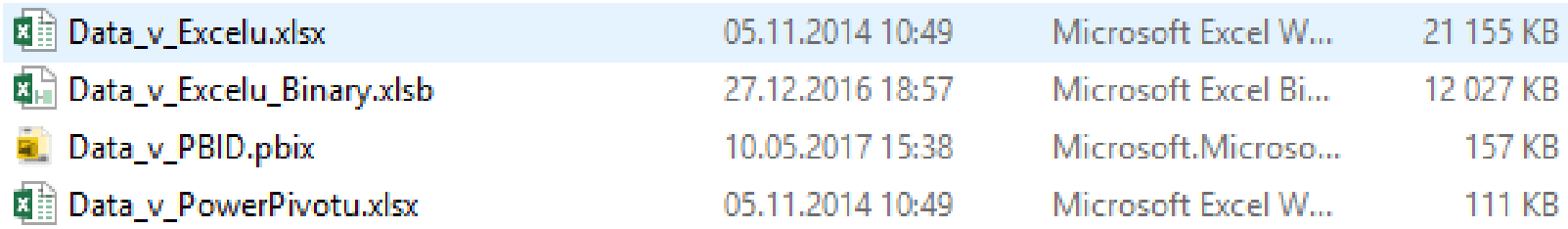
EventClass	CurrentTime	StartTime	Success	Error	Duration	CPUTime	EventSubclass
Query Begin	2016-03-21 20:52:37.346	2016-03-21 20:52:37.346					3 - DAXQuery
DAX Query Plan	2016-03-21 20:52:37.348						1 - DAX VertiPaq Logical Plan
VertiPaq SE Query Begin	2016-03-21 20:52:37.349	2016-03-21 20:52:37.348					0 - VertiPaq Scan
VertiPaq SE Query Begin	2016-03-21 20:52:37.349	2016-03-21 20:52:37.349					10 - VertiPaq Scan internal
VertiPaq SE Query End	2016-03-21 20:52:37.349	2016-03-21 20:52:37.349	1	0	0	0	10 - VertiPaq Scan internal
VertiPaq SE Query End	2016-03-21 20:52:37.349	2016-03-21 20:52:37.348	1	0	1	0	0 - VertiPaq Scan
DAX Query Plan	2016-03-21 20:52:37.349						2 - DAX VertiPaq Physical Plan
Serialize Results Current	2016-03-21 20:52:37.349						3 - Serialize SQL Rowset
Lock Released	2016-03-21 20:52:37.350						
Lock Released	2016-03-21 20:52:37.350						
Query End	2016-03-21 20:52:37.350	2016-03-21 20:52:37.346	1		4	0	3 - DAXQuery
Resource Usage	2016-03-21 20:52:37.350						
Lock Acquired	2016-03-21 20:52:37.351						

File Edit View Replay Tools Window Help

EventClass	CurrentTime	StartTime	Success	Error	Duration	CPUTime	EventSubclass
Query Begin	2016-03-21 20:52:37.606	2016-03-21 20:52:37.606					3 - DAXQuery
DAX Query Plan	2016-03-21 20:52:37.608						1 - DAX VertiPaq Logical Plan
VertiPaq SE Query Begin	2016-03-21 20:52:37.609	2016-03-21 20:52:37.609					0 - VertiPaq Scan
VertiPaq SE Query Begin	2016-03-21 20:52:37.609	2016-03-21 20:52:37.609					10 - VertiPaq Scan internal
VertiPaq SE Query End	2016-03-21 20:52:37.609	2016-03-21 20:52:37.609	1	0	0	0	10 - VertiPaq Scan internal
VertiPaq SE Query End	2016-03-21 20:52:37.609	2016-03-21 20:52:37.609	1	0	0	0	0 - VertiPaq Scan
DAX Query Plan	2016-03-21 20:52:37.610						2 - DAX VertiPaq Physical Plan
Serialize Results Current	2016-03-21 20:52:37.617						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.623						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.630						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.636						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.643						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.649						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.656						3 - Serialize SQL Rowset
Serialize Results Current	2016-03-21 20:52:37.656						3 - Serialize SQL Rowset
Lock Released	2016-03-21 20:52:37.656						
Lock Released	2016-03-21 20:52:37.656						
Query End	2016-03-21 20:52:37.656	2016-03-21 20:52:37.606	1		50	16	3 - DAXQuery
Resource Usage	2016-03-21 20:52:37.656						
Lock Acquired	2016-03-21 20:52:37.659						

VertiPaq engine v praxi

- 21MB Excel soubor v PBIX jen jako 160KB?
- Redukce může být opravdu znatelná



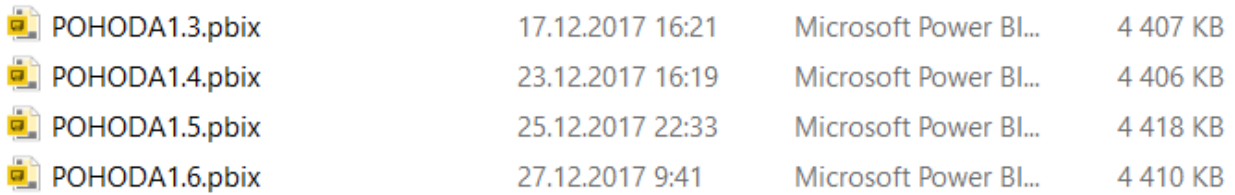
Data_v_Excelu.xlsx	05.11.2014 10:49	Microsoft Excel W...	21 155 KB
Data_v_Excelu_Binary.xlsb	27.12.2016 18:57	Microsoft Excel Bi...	12 027 KB
Data_v_PBID.pbix	10.05.2017 15:38	Microsoft.Microso...	157 KB
Data_v_PowerPivotu.xlsx	05.11.2014 10:49	Microsoft Excel W...	111 KB

DAX vs. Query vrstva

- Na 100% lze doporučit nachystat vše přímo ve zdroji
- Pokud nelze => Query okno
- Jedná se o výkonově a paměťově méně náročnou variantu než DAX
- V Query okně je dobré přemýšlet nad použitými kroky
- DAX je výhodnější používat na measures, které v Query nenachystáme

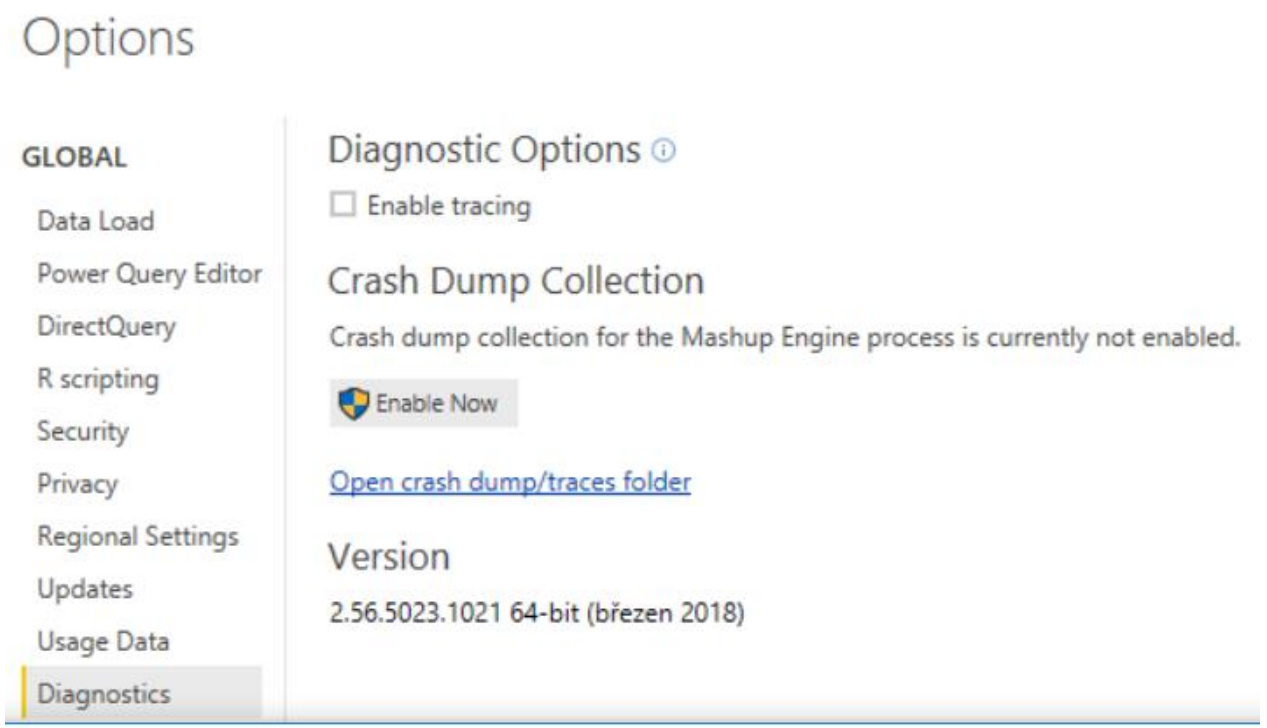
DAX columns vs. DAX measures

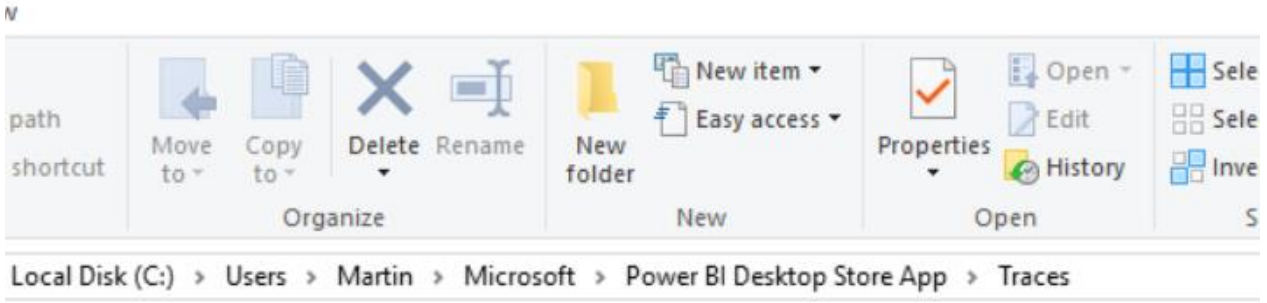
- Vše k čemu by se mohly hodit počítané sloupce lze udělat v Query okně (např. spojování, rozdělování, výjimky, atd.)
- Počítané sloupce zaberou místo v RAM
- Measures nezaberou místo v RAM a minimálně na disku a dopočítají se až v případě zobrazení samotného zápisu nebo vizuálu, kde je přítomen výpočet
- Mezi POHODA 1.3. a 1.6 je rozdíl cca 200 measures



Diagnostika LOG souborů

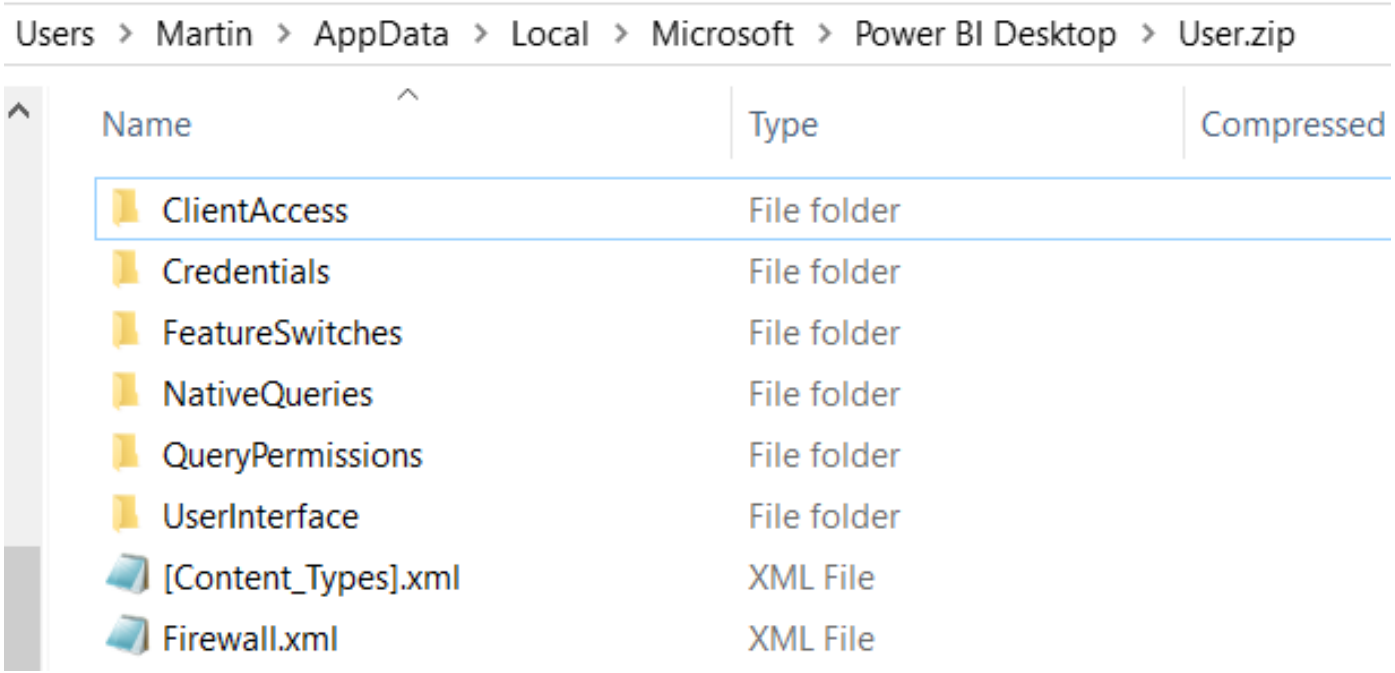
- Pokud povolíme trasování v globálním nastavení, tak ukládá soubory pro další analýzu do Traces
- Následně můžeme využít již hotového řešení pro procházení LOGů přímo v reportech





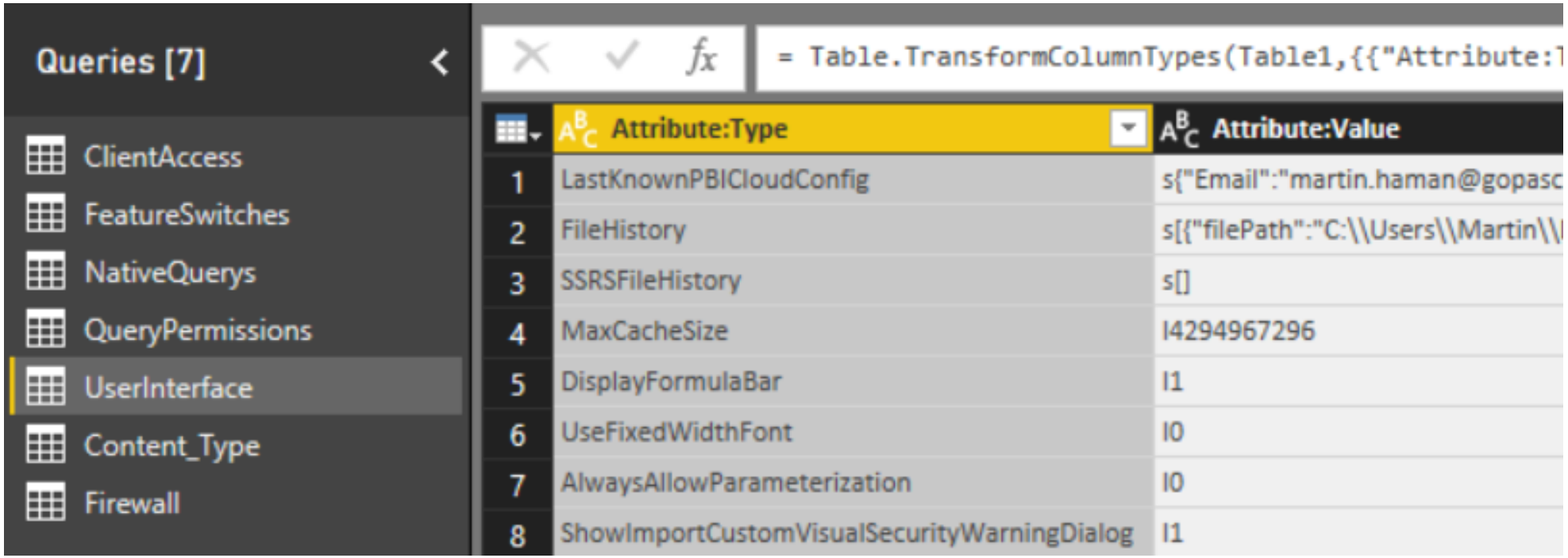
User.zip

- Veškeré informace o přístupech a credentials se ukládají lokálně do User.zip



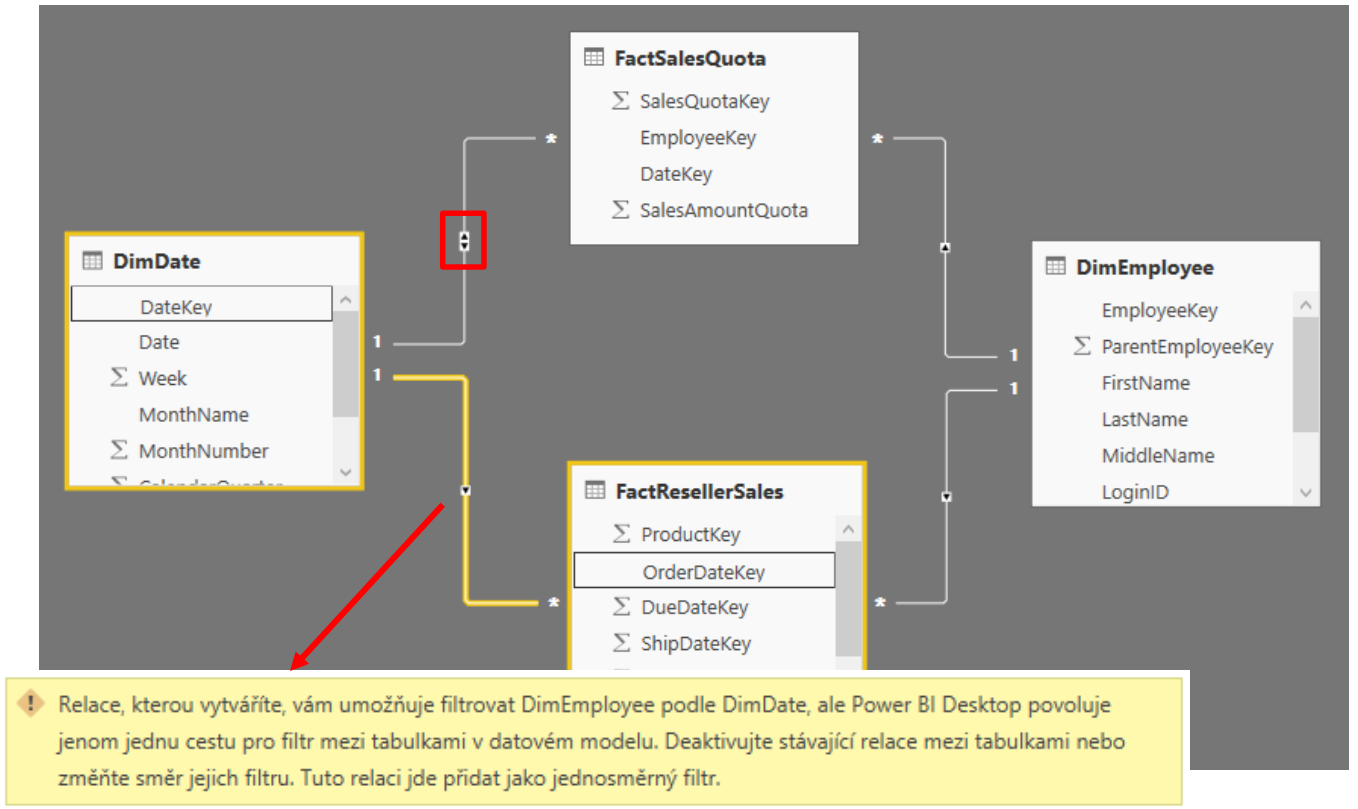
User.zip

- Vše se dá načíst jako XML soubory a následně analyzovat



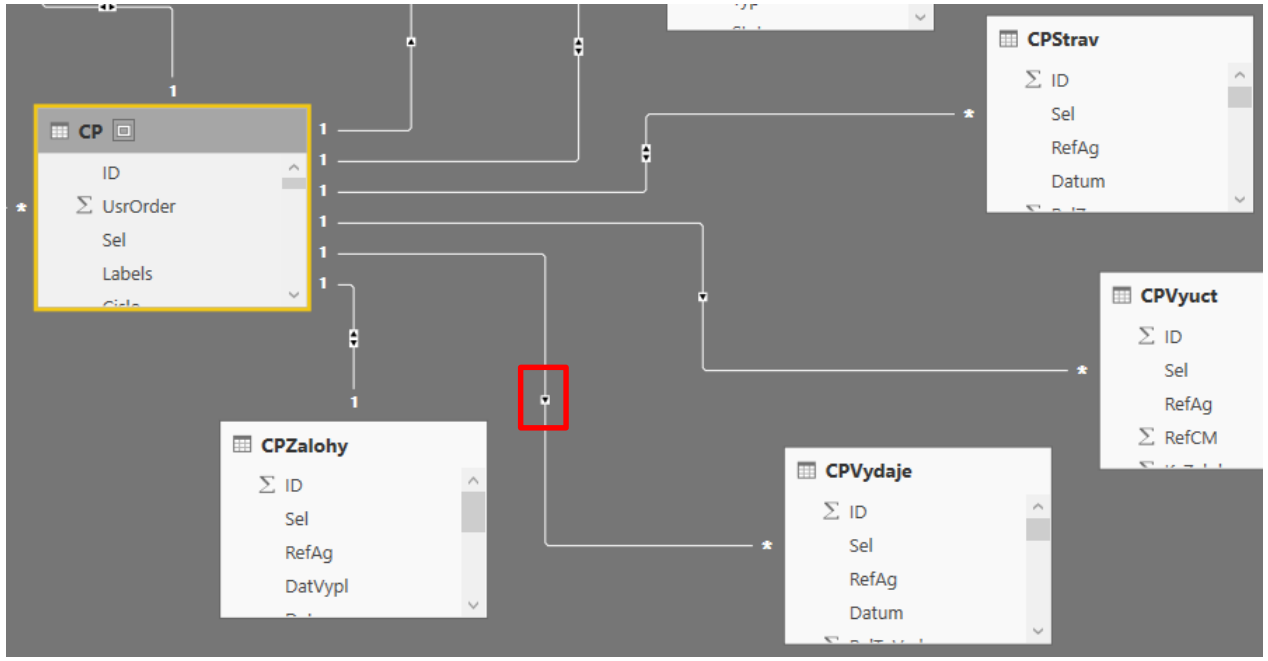
Relace a směr křížového filtru

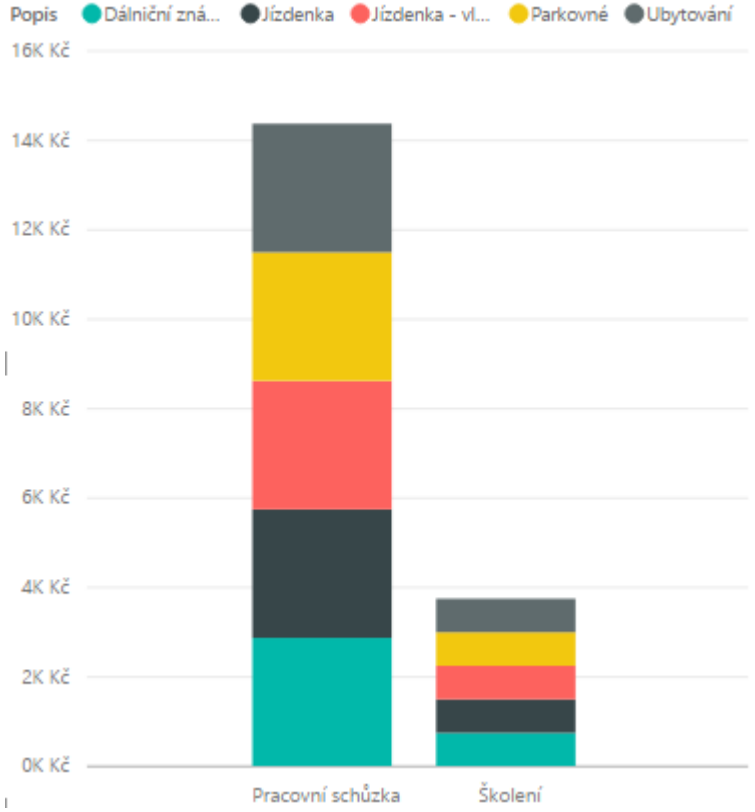
- Relace v PBID mají možnost upravit směr filtru
- Křížový filtr může být jednosměrný nebo obousměrný (odkud kam dosáhneme na data)



Směr křížového filtru - Single

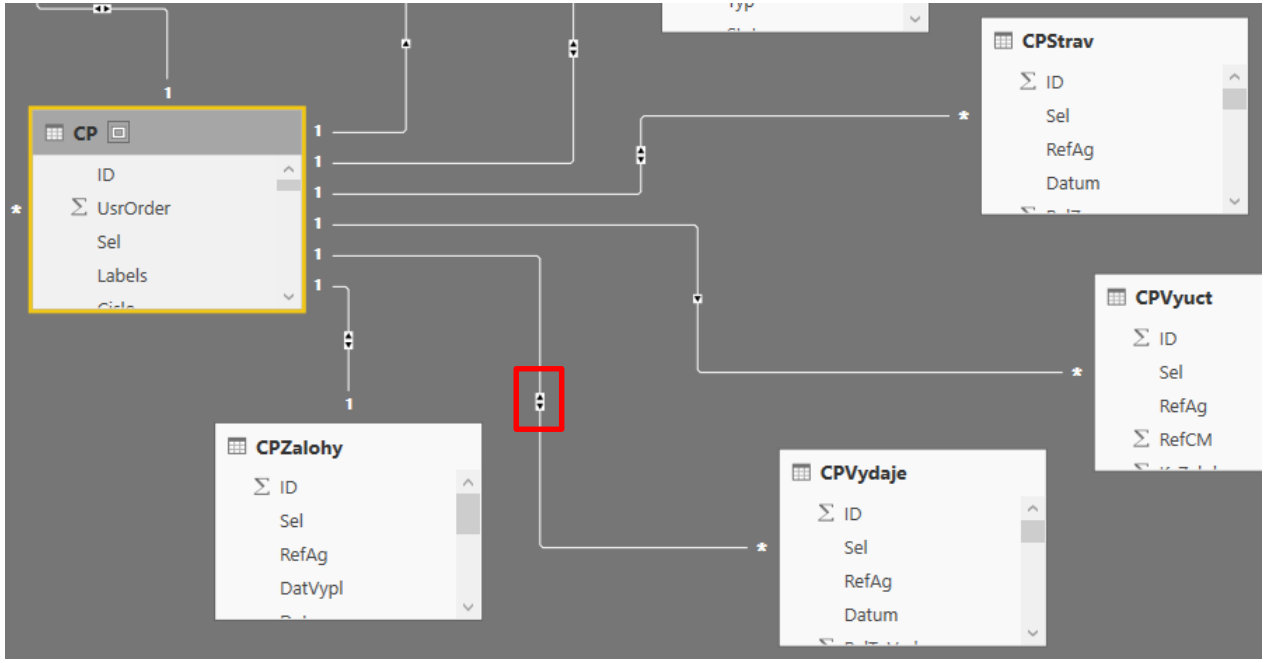
- CPStrav (finance) > CP (událost) > CPVydaje (jízdenky, atd.)

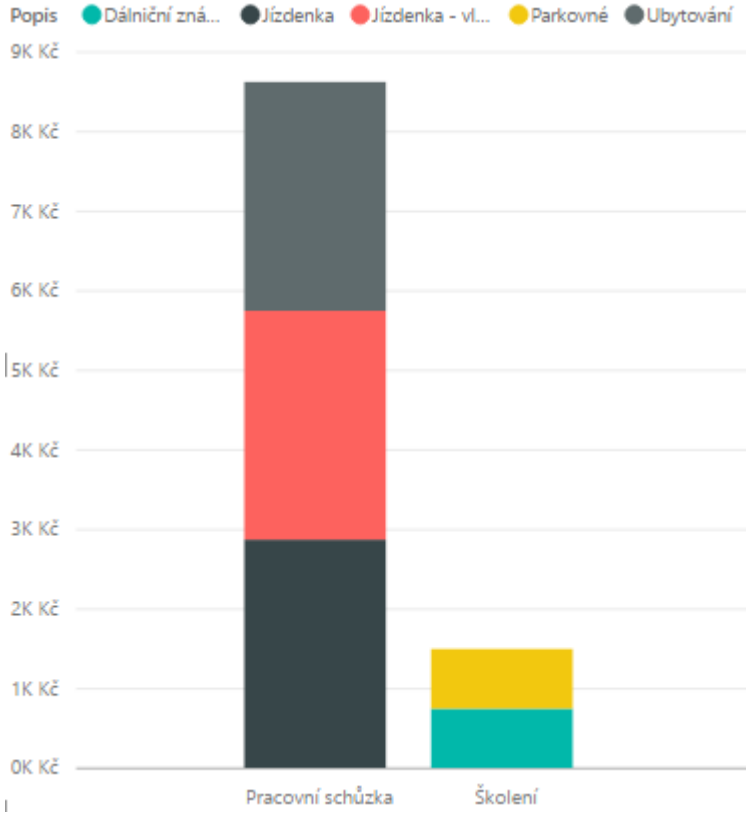




Směr křížového filtru - Both

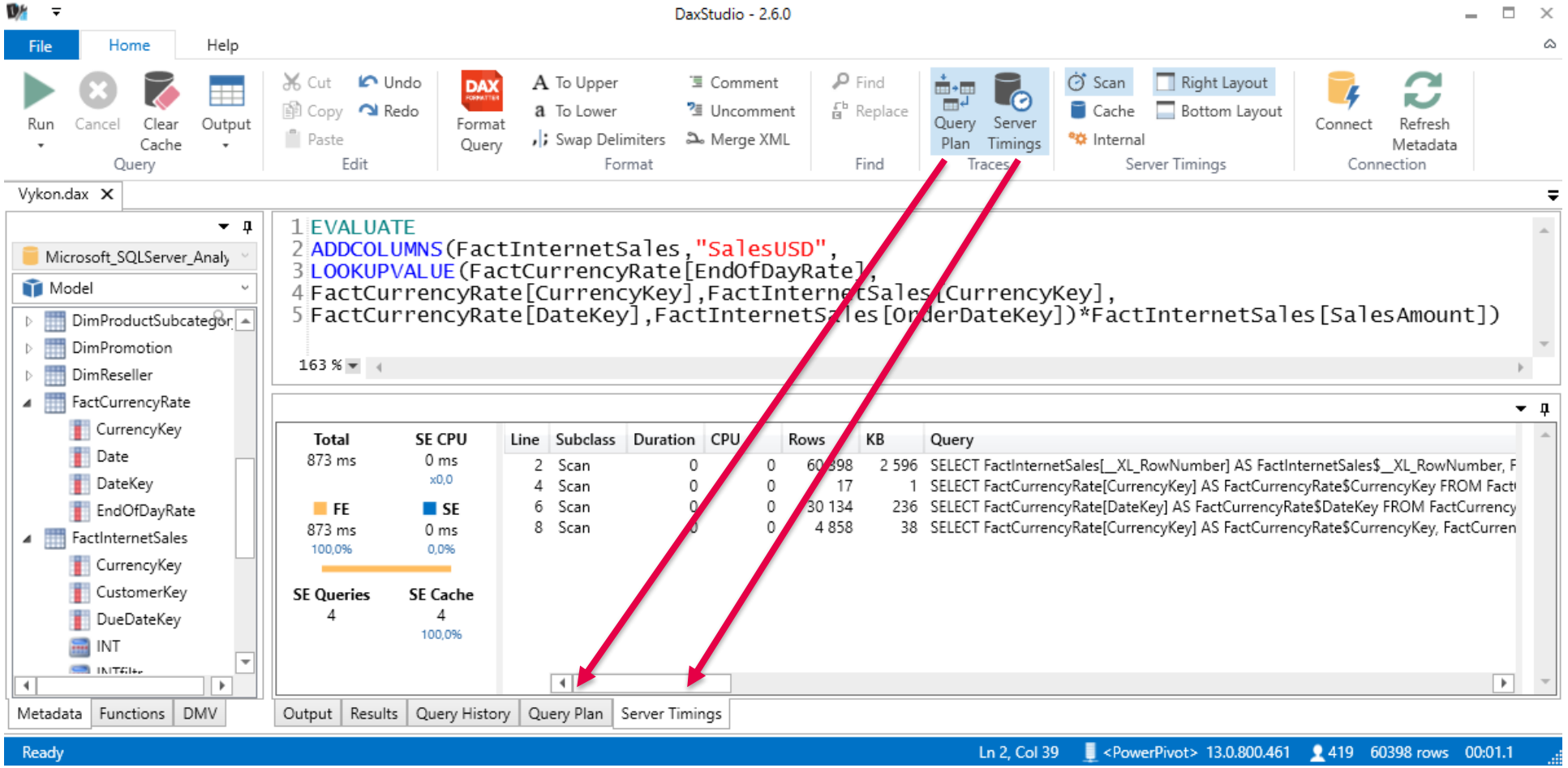
- CPStrav (finance) > CP (událost) > CPVydaje (jízdenky, atd.)





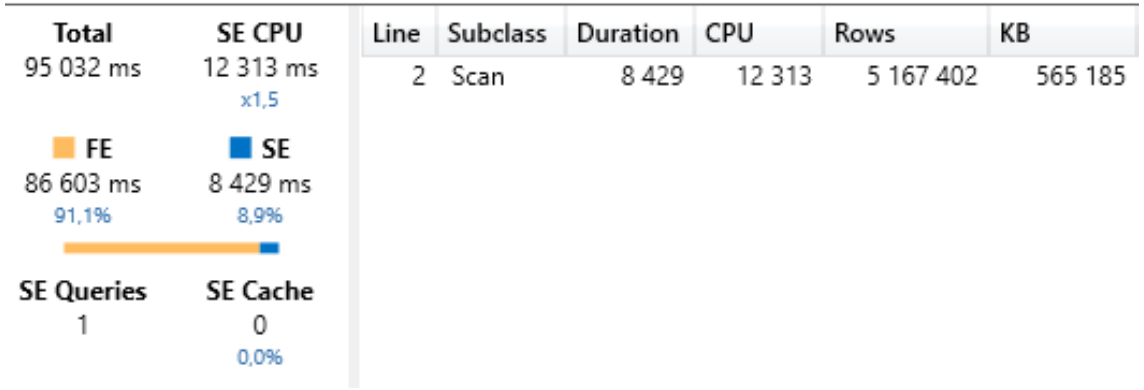
Testování výkonu DAX v DaxStudio

- První spustíme Query Plan, následně Server Timings
- Následuje tvorba výrazu a „Run“ tlačítko



Testování výkonu DAX v DaxStudio

- Legenda k výsledkům:
  Rows – počet řádků ve zdroji 5.167.402
  KB – využitá RAM 565.185 KB
  Total = FE+SE celkový čas načtení dotazu
  FE – Formula engine (vyhodnocení výpočtu)
  SE = Duration – Storage engine (uložení dotazu)
  SE CPU – jak dlouho trvalo uložení dotazu
  SE Queries – počet dotazů
  SE Cache – uložení do RAM



Testování výkonu DAX v DaxStudio

- SE Queries = 9 dotazů ; SE Cache = 7 dotazů

```
evaluate
ADDCOLUMNS(SUMMARIZE(FactInternetSales,DimDate[CalendarYear],
DimProductCategory[ProductCategoryName],
"Prodeje za internet",SUM(FactInternetSales[SalesAmount]),
"Prodeje za obchodníky",SUM(FactResellerSales[SalesAmount]),
"Celkové prodeje",SUM(FactInternetSales[SalesAmount])+SUM(FactResellerSales[SalesAmount])),
"Suma Prodejů",SUM(FactInternetSales[SalesAmount])+SUM(FactResellerSales[SalesAmount]))
```

163 %

Total	SE CPU
236 ms	32 ms
	x2,9
FE	SE
225 ms	11 ms
95,3%	4,7%
SE Queries	SE Cache
9	7
	77,8%

Line	Subclass	Duration	CPU	Rows	KB	Query
2	Scan	1	0	1	1	SELECT SUM ('FactInternetSales'[SalesAmount]) FROM 'FactInternetSales'; 'Estimated size (volume, marsha
4	Scan	0	0	1	1	SELECT SUM ('FactResellerSales'[SalesAmount]) FROM 'FactResellerSales'; 'Estimated size (volume, marshal
6	Scan	0	0	63	1	SELECT 'DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], SUM ('FactInternetSales'[Sa
8	Scan	0	0	60 391	1 888	SELECT 'DimCurrency'[CurrencyKey], 'DimCustomer'[CustomerKey], 'DimDate'[DateKey], 'DimDate'[CalendarY
10	Scan	0	16	37 306	146	DEFINE TABLE '$TTable3' := SELECT 'DimCustomer'[CustomerKey], 'DimDate'[CalendarYear], 'DimProductCate
12	Scan	0	0	18 496	3	DEFINE TABLE '$TTable4' := SELECT SIMPLEINDEXN ('$TTable3'[DimCustomer$CustomerKey]) FROM '$TTable
14	Scan	0	0	37 306	146	DEFINE TABLE '$TTable1' := SELECT '$TTable2'[DimCustomer$CustomerKey], '$TTable3'[DimDate$CalendarYea
15	VertiPaq	2	16			DEFINE TABLE '$TTable3' := SELECT 'DimCustomer'[CustomerKey], 'DimDate'[CalendarYear], 'DimProductCate
17	Scan	5	16	17	1	SELECT 'DimCurrency'[CurrencyKey], 'DimDate'[DateKey], 'DimDate'[CalendarYear], 'DimGeography'[Geograp
19	Scan	5	0	17	1	SELECT 'DimCurrency'[CurrencyKey], 'DimDate'[DateKey], 'DimDate'[CalendarYear], 'DimGeography'[Geograp

Output | Results | Query History | Server Timings | Query Plan

Testování výkonu DAX v DaxStudio

- Fyzický query plán má 77 řádků (summarize je velmi náročná)

```
evaluate
ADDCOLUMNS(SUMMARIZE(FactInternetSales,DimDate[CalendarYear],
DimProductCategory[ProductCategoryName],
"Prodeje za internet",SUM(FactInternetSales[SalesAmount]),
"Prodeje za obchodníky",SUM(FactResellerSales[SalesAmount]),
"Celkové prodeje",SUM(FactInternetSales[SalesAmount])+SUM(FactResellerSales[SalesAmount])),
"Suma Prodejů",SUM(FactInternetSales[SalesAmount])+SUM(FactResellerSales[SalesAmount]))
```

163 %

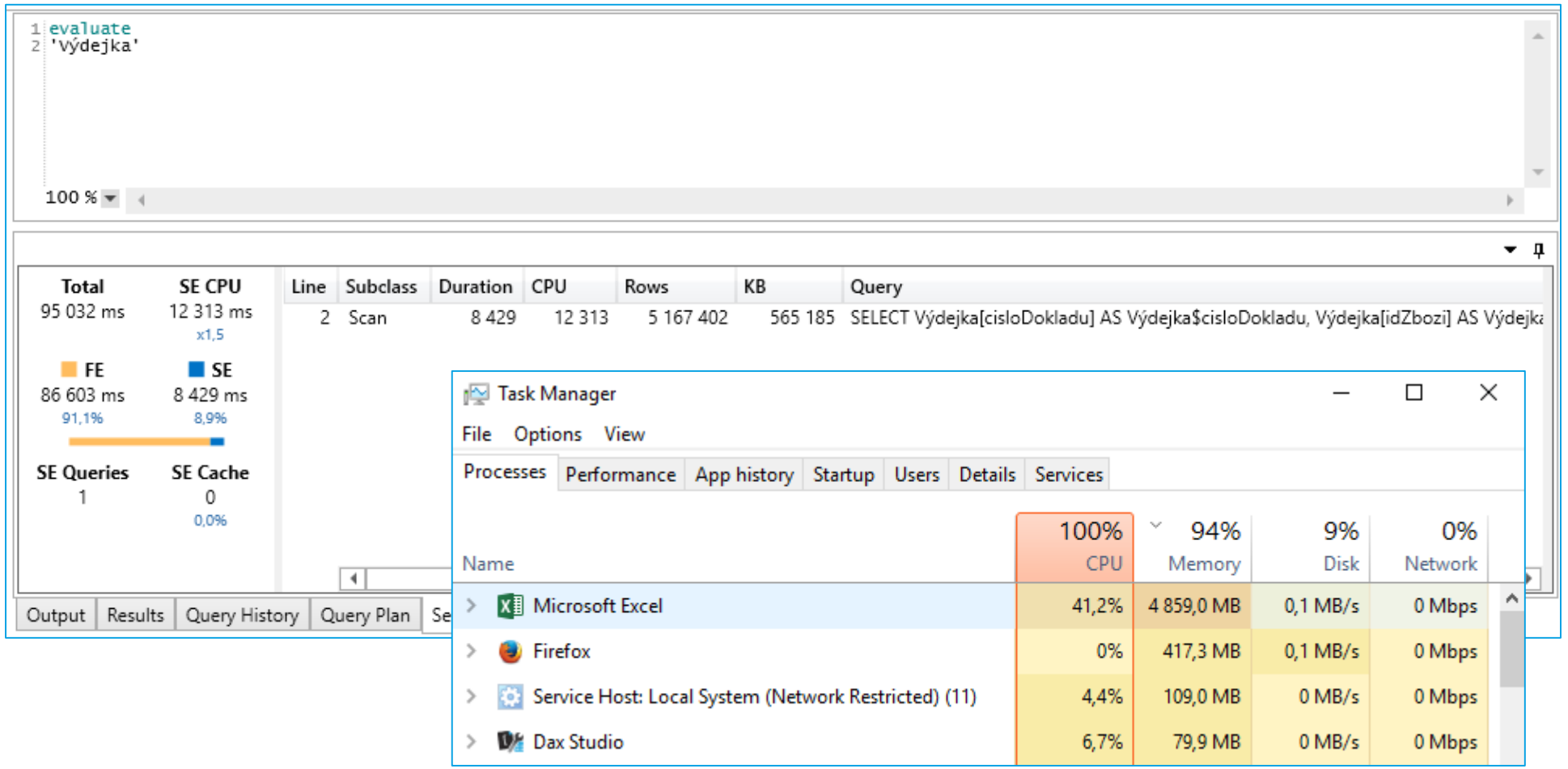
Line	Records	Physical Query Plan
1		AddColumns: IterPhyOp LogOp=AddColumns IterCols(0, 1, 2, 3, 4, 5)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], ''[Prodeje za internet], ''[Prodeje
2		AddColumns: IterPhyOp LogOp=AddColumns IterCols(0, 1, 2, 3, 4)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], ''[Prodeje za internet], ''[Prodeje
3	8	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=GroupBy_Vertipaq IterCols(0, 1)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName]) #Records=8 #I
4	8	AggregationSpool<GroupBy>: SpoolPhyOp #Records=8
5	60 391	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=GroupBy_Vertipaq IterCols(0, 1, 33, 52, 59, 74, 120, 124)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCa
6	60 391	ProjectionSpool<ProjectFusion<>>: SpoolPhyOp #Records=60391
7		VertipaqResult: IterPhyOp #FieldCols=8 #ValueCols=0
8	8	SpoolLookup: LookupPhyOp LogOp=Sum_Vertipaq LookupCols(0, 1)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName]) Currency #Records=8 #Key(
9	8	ProjectionSpool<ProjectFusion<Copy>>: SpoolPhyOp #Records=8
10		VertipaqResult: IterPhyOp #FieldCols=2 #ValueCols=1
11	6	SpoolLookup: LookupPhyOp LogOp=Sum_Vertipaq LookupCols(0, 1)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName]) Currency #Records=6 #Key(
12	6	AggregationSpool<AggFusion<Sum>>: SpoolPhyOp #Records=6

Line	Logical Query Plan
1	AddColumns: RelLogOp DependOnCols()() 0-5 RequiredCols(0, 1, 2, 3, 4, 5)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], ''[Prodeje za internet], ''[Prodeje za obch
2	AddColumns: RelLogOp DependOnCols()() 0-4 RequiredCols(0, 1, 2, 3, 4)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], ''[Prodeje za internet], ''[Prodeje za obch
3	GroupBy_Vertipaq: RelLogOp DependOnCols()() 0-125 RequiredCols(0, 1, 33, 52, 59, 74, 120, 124)('DimDate'[CalendarYear], 'DimProductCategory'[ProductCategoryName], 'DimPromotic
4	Scan_Vertipaq: RelLogOp DependOnCols()() 0-125 RequiredCols(1, 5, 51, 55, 67, 74, 88, 94)('DimCurrency'[CurrencyKey], 'DimCustomer'[CustomerKey], 'DimDate'[DateKey], 'DimDate'

Output | Results | Query History | Server Timings | Query Plan

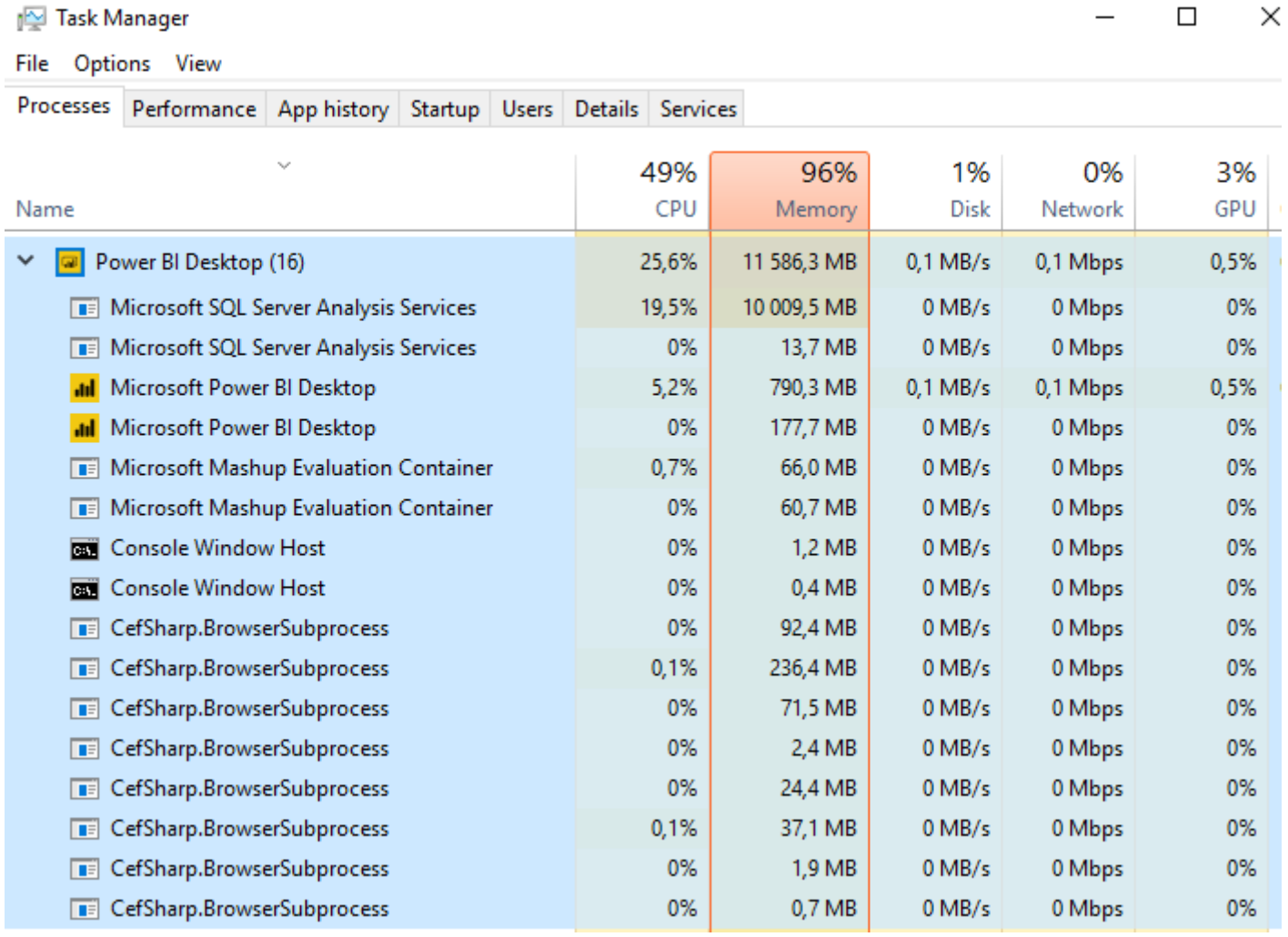
Testování výkonu v tabulce s >5M rows

- Cca 10MB na 100000 řádků



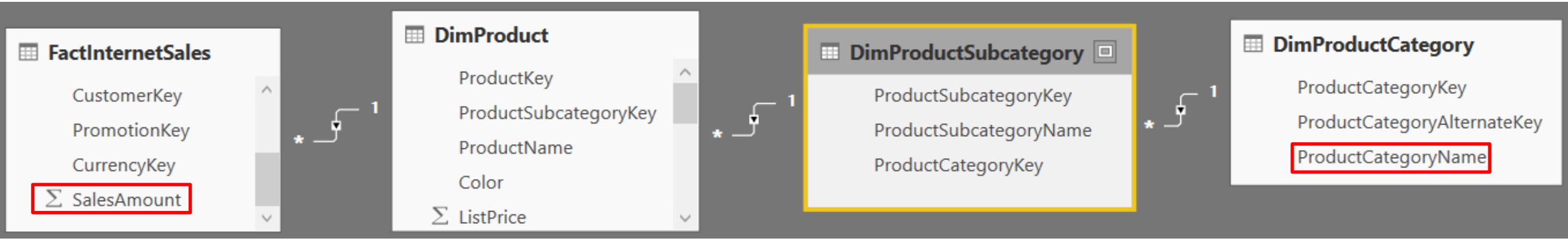
Optimalizace je základ

- Optimalizujte nebo nakupte silnější stroje a více RAM!



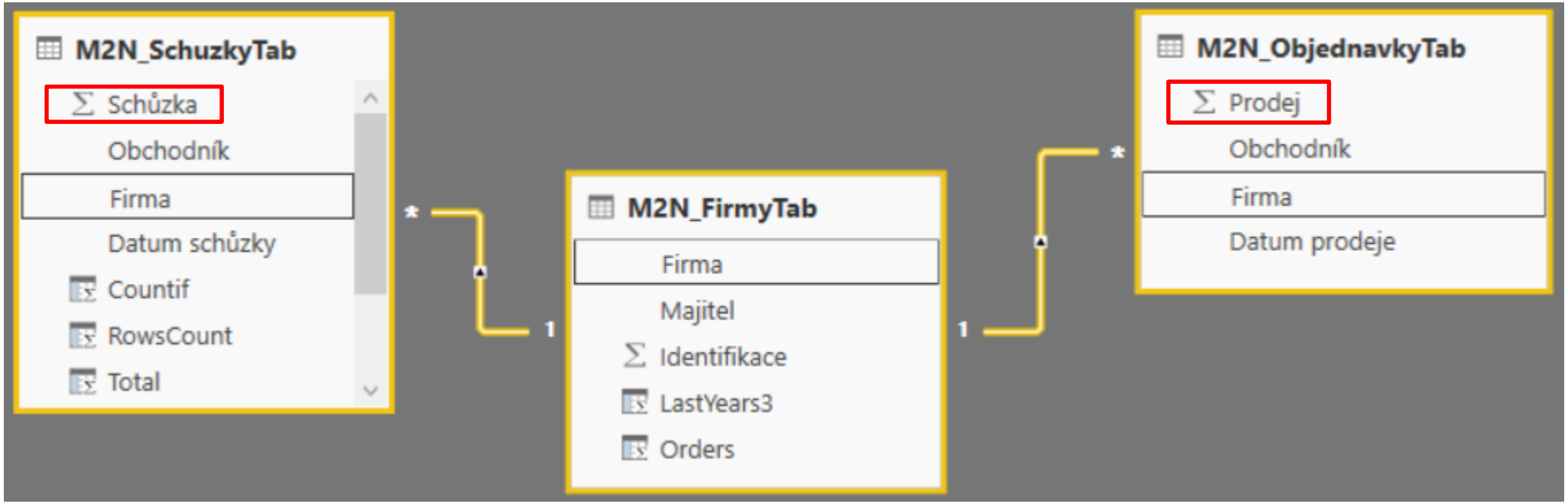
Relace M2N

- Pokud je vše N>1>N>1>N>1 tak není problém s vizuály a zobrazením správných dat
- Např. Prodej za jednotlivé kategorie



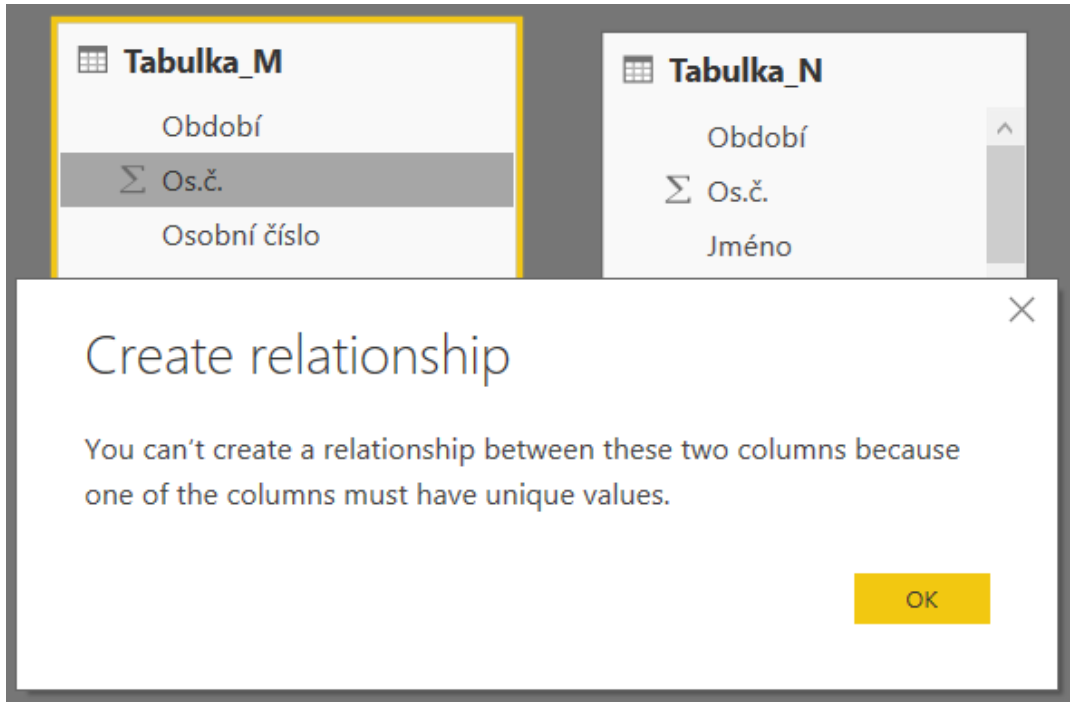
Relace M2N

- Když ovšem dojdeme do situace, která je na obrázku a potřebujeme schůzky a prodeje pro firmy?



Relace M2N

- Případně ještě o něco horší varianta, není mezitabulka



Relace M2N

- Řešení (výpočtové sloupce tvoříme v Tabulka_N):

Nejprve spočítáme kolikrát je který záznam z tabulky M v tabulce N:
Countif = CALCULATE(COUNT(Tabulka_M[Os.č.]);FILTER(Tabulka_M;Tabulka_M[Os.č.]=Tabulka_N[Os.č.]))

Po té spočítáme počet záznamů z tabulky M:
RowsCount = COUNTROWS(FILTER(Tabulka_N;Tabulka_N[Os.č.]=EARLIER(Tabulka_N[Os.č.])))

Podělíme, protože vizuál automaticky dělá každý s každým, když nemá relaci:
Result = DIVIDE(Tabulka_N[_Countif];Tabulka_N[_RowsCount])

Os.č.	_Countif	_RowsCount	Počet záznamů
1	9	9	3
2		9	
3	6	9	2
4	3	9	1
5		1	
6		1	
Total	**18**	**38**	**6**

Děkuji za pozornost.

Martin.Haman@gmail.com