POWER QUERY

Co je **Power Query**?

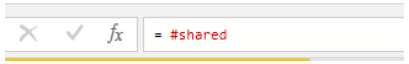
„IDE pro vývoj M“

Komponenty

› **Pás karet** – Pás obsahující nastavení a předpřipravené funkce, které Power Query samo přepíše do jazyka M pro pohodlí uživatele.
› **Dotazy** – jednoduše pojmenovaný výraz M. Alias **Queries** se mohou seskládat do skupin
› **Primitiv** – Primitiv je jednodílná hodnota, jako je například číslo, pravda/nepravda, text, datum, nebo null. Hodnotu **null** lze použít k označení absence jakýchkoli dat.
› **Seznam** – Alias **List**. Seznam je seřazená posloupnost hodnot. Jazyk M podporuje nekonečné seznamy. Seznamy definují znaky „**{**„ a „**}**“, které označují začátek a konec seznamu.
› **Záznam** – Záznam je sada polí, kdy pole je dvojice, kterou tvoří název a hodnota. Název je textová hodnota, která je v záznamu pole jedinečná.
› **Tabulka** – Tabulka je sada hodnot uspořádaná do pojmenovaných sloupců a řádků. S tabulkou lze pracovat, jako by se jednalo o seznam záznamů, nebo jako by se jednalo o záznam seznamů. **Tabulka [Pole]** (syntaxe odkazu na pole pro záznamy) vrací seznam hodnot v tomto poli. „**Tabulka {i}**“ (syntaxe přístupu k indexu seznamu) vrací záznam představující řádek tabulky.
› **Funkce** – Funkce je hodnota, která při vyvolání pomocí argumentů vytvoří novou hodnotu. Funkce se zapisují vypsáním parametrů funkce v závorkách, následovaným symbolem přechodu „**=>**“ a výrazem definující funkci. Tento výraz obvykle odkazuje na parametry pomocí jména. Existují i funkce bez parametrů.
› **Parametr** – Parametr ukládá hodnotu, kterou lze použít pro transformace. Kromě názvu parametru a hodnoty, kterou ukládá, má také další vlastnosti, které mu poskytují metadata. Nespornou výhodou parametru je to, že jde měnit i z prostředí **Power BI Service** bez nutnosti přímého zásahu do datové sady. Syntaxe parametru je jako běžný dotaz pouze to, co je zvláštní, je to metadata mají specifický formát.
› **Řádek vzorců** – Zobrazuje aktuálně načtený krok a umožňuje jeho úpravu. Defaultně je tento řádek skrytý. Jeho zobrazení se prování na kartě **Zobrazit**.
› **Nastavení dotazu** – Nastavení obsahující možnost editace jména a popisu dotazu. Zároveň obsahuje přehled všech aktuálně aplikovaných kroků. Aplikované kroky jsou proměnné definované ve výrazu let a oni jsou reprezentovány názvy proměnných.
› **Náhled na data** – Komponenta zobrazující náhled na data v aktuálně zvoleném kroku transformace.
› **Stavový řádek** – Jde o řádek umístěný na samém spodu obrazovky. Řádek obsahuje informaci o přibližném stavu řádků, sloupců a času posledního vytvoření náhledu na data. Krom těchto informací je zde informace zdroji profilace pro sloupce. Zde je možné profilaci přepnout z 1000 řádků na celou datovou sadu.

Funkce v Power Query

Znalost funkcí je vaším nejlepším pomocníkem při práci s funkčním jazykem jako je jazyk **M**.
› **Shared** – Je klíčové slovo, které načte všechny funkce (včetně nápovědy a ukázky) a enumerátory ve výsledkové sadě. Vyvolání se provede v prázdném dotazu za pomoci zápisu **=#shared**



Funkce můžeme rozdělit na dvě kategorie:
› **Předpřipravené** – Příklad: **Date.From()**
› **Vlastní** – jde o funkce, které si sami do modelu připraví sám uživatel za pomocí rozšíření zápisu o **„() =>“,** kdy do závorek se dají umístit parametry, které pro vyhodnocení funkce budou požadovány. Při použití více parametrů je nezbytné je oddělit pomocí oddělovače.

Druhy hodnot

Každý druh hodnoty je spojen s doslovnou syntaxí, sadou hodnot tohoto druhu, sadou operátorů definovaných nad touto sadou hodnot a vnitřním typem připisovaným nově vytvořeným hodnotám.

› **Null** – null
› **Logické** – true, false
› **Číslo** – 1, 2, 3, ...
› **Čas** – #time(HH,MM,SS)
› **Datum** – #date(yyyy,mm,ss)
› **Datum a Čas** – #datetime(yyyy,mm,dd,HH,MM,SS)
› **Datum, Čas a Časová zóna** – #datetimezone(yyyy,mm,dd,HH,MM,SS, 9,00)
› **Trvání** – #duration(DD,HH,MM,SS)
› **Text** – „text"
› **Binary** – #binary(„ODAKZ")
› **List** – {1, 2, 3}
› **Záznam** – [A = 1, B = 2]
› **Tabulka** – #table({sloupce},{{Obsah Řádku 1},{}...})*
› **Funkce** – (x) => x + 1
› **Typ** – type { number }, type table [A = any, B = text]
* Index prvního řádku tabulky je stejně jako u záznamů v listu **0**

Operátory

Operátorů v rámci jazyka M je celá řada ne však každý operátor se dá použít pro všechny typy hodnot.
› **Primární operátory**
› **(x)** - Prioritizace
› **x[i]** – Vyhledávání. Vrátí hodnotu ze záznamu, seznam hodnot z tabulky.
› **x{i}** – Přístup k hodnotě ve vloženém indexu. Vrátí hodnotu ze seznamu, záznam z tabulky
„Umístěním znaku „**?**“ Za operátor dojde k návratu hodnoty **null**, pokud se index nebude v seznamu nacházet“
› **x(...)** – Ohraničení funkce
› **{1 .. 10}** – Automatické vytvoření seznamu od 1 do 10
› **...** – Neimplementováno
› **Matematické operátory** – **+, -, *, /**
› **Porovnávací operátory**
› **>** , **>=** – Větší než, větší nebo rovno
› **<** , **<=** – Menší než, menší nebo rovno
› **=** , **<>** – je rovno, není rovno. Je rovno platí i pro null = null
› **Logické operátory**
› **and** – spojovací funkce „a“
› **or** – spojovací funkce „nebo“
› **not** – logická negace
› **Typové operátory**
› **as** – potvrzení kompatibility s primitivním typem nebo chybou
› **is** – ověření kompatibility s primitivním typem nebo chybou
› **Metadata** - slovo meta přiřadí metadata k hodnotě. Příklad zápisu „**x meta y**“ nebo „**x meta [name = x,...]**“
V rámci Power Query platí priorita operátorů, takže například „X + Y * Z“ bude vyhodnocováno jako „X + (Y*Z)“

Komentáře

Jazyk M podporuje **dva** verze komentářů:
› Jednořádkové komentáře – tvoří se za pomoci znaků **//** před kódem
› Zkratka: **CTRL +** ´
› Víceřádkové – tvoří se za pomoci znaků **/*** před kódem a ***/** za kódem
› Zkratka: **ALT + SHIFT + A**

Výraz let

Výraz let se používá k zachycení hodnoty z mezilehlého výpočtu v pojmenované proměnné. Tyto pojmenované proměnné jsou v rozsahu lokálního výrazu `let`. Konstrukce termínu let vypadá takto:

```
let
   název_proměnné = <výraz>
   výstupováProměnná = <funkce>(název_proměnné)
in
   výstupováProměnná
```

Když dochází k jeho vyhodnocení, tak vždy platí následující:

› Výrazy v proměnných definují nový rozsah obsahující identifikátory z produkce seznamu proměnných a musí být přítomny při hodnocení výrazů v rámci seznamu proměnných. Výrazy v seznamu proměnných se mohou navzájem odkazovat
› Všechny proměnné musí být vyhodnoceny než dojde k vyhodnocení výrazu let.
› Pokud nejsou zpřístupněny výrazy v proměnných, tak nesmí být vyhodnocovány
› Chyby, které se vyskytnou během vyhodnocení dotazu se rozšíří jako chyba do dalších provázaných dotazů.

Podmínky

I v Power Query existuje funkce „**Když**“ (**if**), která na základě vložené podmínky rozhodne zda výsledkem bude skutečnost pro pravdu nebo nepravdu.

Sémantický zápis:
if <predicate> **then** < true-expression > **else** < false-expression >
„**else** is required in M's conditional expression “

Zápis podmínky:
If x > 2 **then** 1 **else** 0
If [Month] > [Fiscal_Month] **then** true **else** false

If výraz je jediným podmíněným výrazem v M. Máte-li k testování více predikátů, musíte je spojit dohromady např:
if <predicate> **then** < true-expression >
else if <predicate **then** < false-true-expression >
else < false-false-expression >

Při vyhodnocování podmínek platí, že:

› Pokud hodnota vytvořená vyhodnocením podmínky if není logická hodnota, pak se vyvolá chyba s kódem příčiny „Expression.Error“
› Pravdivý výraz se vyhodnotí, pouze pokud se podmínka **if** vyhodnotí na hodnotu **true** s výsledkem **false** dojde k vyhodnocení nepravdivého výrazu.
› Pokud nejsou zpřístupněny výrazy v proměnných, tak nesmí být vyhodnocovány
› Chyba vzniklá během vyhodnocení podmínky se bude dále šířit buďto formou selhání celého dotazu nebo hodnotou „**Error**“ v záznamu.

Výraz try ... otherwise

Odchytávání chyb je možné například pomocí výrazu **try**. Dojde k pokusu vyhodnotit výraz za slovem **try**. Pokud při vyhodnocení dojde k chybě, tak se aplikuje výraz za slovem **otherwise**

Zápis výrazu:
try Date.From([textDate]) **otherwise** null

Vlastní funkce

Příklad zápisů vlastní funkce:
(x, y) => Number.From(**x**) + Number.From(**y**)

```
(x) =>
let
   out = Number.From(x) +
   Number.From(Date.From(DateTime.LocalNow()))
in
   out
```

Vstupní parametry do funkcí jsou dvojího typu:
› **Povinné** – Všechny běžně zapsané parametry v (). Bez těchto parametrů nejde funkci vyvolat.
› **Nepovinné** – Takový parametr může nebo nemusí fungovat vstoupit. Označte parametr jako volitelný umístěním textu před název argumentu „**Optional**“. Například **(optional x)**. Pokud si to nenastane splnění volitelného argumentu, tak buďte pro účely výpočtu stejné, ale jeho hodnota bude nulová. **Volitelné argumenty musí následovat po požadovaných argumentech..**

Argumenty lze označit pomocí `as <type>`, aby se označil požadovaný typ argumentu. Funkce vyvolá chybu typu, pokud je volána s argumenty nesprávného typu. Funkce mohou mít také jejich anotovaný návrat. Tato anotace je poskytována jako: **(x as number, y as text) as logical** => <expression>

Výstup z funkcí je velmi rozdílný. Výstupem totiž může být list, tabulka, jedna hodnota ale i další funkce. Znamená to tedy to, že jedna funkce může produkovat funkce další. Taková funkce se zapíše takto:

let first = **(x)=> () =>** let out = {1..x} in out in first

Při vyhodnocování funkcí platí, že:

› Chyby vyvolané při vyhodnocení výrazů v seznamu výrazů nebo ve výrazu funkce se bude dále šířit buďto formou selhání nebo jako hodnota „**Error**“
› Počet argumentů vytvořených ze seznamu argumentů musí být kompatibilní s formálními parametry funkce, jinak dojde k chybě s kódem příčiny „**Expression.Error**“

Rekurzivní funkce

Pro rekurzivní funkce, je nutné použít znak „**@**“ který odkazuje na funkci v rámci jejího výpočtu. Typickou rekurzivní funkcí je faktoriál. Funkci pro faktoriál je možné napsat následovně:

```
let
  Factorial = (x) =>
    if x = 0 then 1 else x * @Factorial(x - 1),
  Result = Factorial(3)
in
  Result // = 6
```

Each

Funkce lze volat proti konkrétním argumentům. Pokud je však nutné provést funkci pro každý záznam, celý list nebo celý sloupec v tabulce, je nutné připojit každé slovo do kódu. Jak název napovídá, pro každý kontextový záznam použije postup, který za ním stojí. **each** není nikdy vyžadován! Jednoduše usnadňuje definování funkce in-line pro funkce, které vyžadují funkci jako argument.

Zjednodušená syntaxe

› Each je v podstatě syntaktickou zkratkou pro deklaraci zadejte funkce pomocí jediného formálního parametru s názvem. Následující zápisy jsou proto sémanticky ekvivalent:

```
let
  Source = ...,
  addColumn = Table.AddColumn(Source, „NewName", each [field1] + 1)
in
  addColumn
```

```
let
  Source = ...,
  add1ToField1 = (_) => [field1] + 1,
  addColumn(Source,"NewName",add1ToField1)
in
```

Další zjednodušení syntaxe spočívá v tom, že holé hranaté závorky jsou zjedodušením pro přístup do pole záznamu s názvem `_`.

Query Folding

Jak název napovídá, jde o skládání. Konkrétně se kroky v Power Query skládají do jediného dotazu, který se poté implementuje proti zdroji dat. Zdroje dat, které podporují skládání dotazů, jsou prostředky, které podporují koncept dotazovacích jazyků jako zdrojů relační databáze. To znamená, že například soubor CSV nebo XML jako plochý soubor s daty rozhodně nebude Query Folding podporovat. Transformace tedy nemusí probíhat až po načtení dat, ale je možné je okamžitě připravit. Ne každý zdroj bohužel tuto funkci podporuje.
› Platné funkce
› Odebírání, Přejmenování sloupců,
› Filtrování řádků
› Seskupení, sumarizování, pivot a unpivot
› Sloučení a extrahování dat z dotazů,
› Připojení dotazů na základě stejného zdroje dat
› Přidání vlastních sloupců s jednoduchou logikou
› Neplatné funkce
› Sloučení dotazů na základě rozdílných zdrojů dat
› Přidávání sloupců s Indexem
› Změna datového typu sloupce

DEMO

› Operátory je možné kombinovat. Například takto:
› LastStep[Year]{[ID]}

*To znamená, že můžete získat hodnotu z jiného kroku na základě indexu ze sloupce

› Výroba DateKey dimenze jde například takto:

```
#table(
   type table [Date=date, Day=Int64.Type, Month=Int64.Type,
MonthName=text, Year=Int64.Type,Quarter=Int64.Type],
   List.Transform(
      List.Dates(start_date, (start_date-endd_ate),
                 #duration(1, 0, 0 ,0)),
      each {_, Date.Day(_), Date.Month(_),
Date.MonthName(_), Date.Year(_), Date.QuarterOfYear(_)}
   ))
```

Klíčová slova

and, as, each, else, error, false, if, in, is, let, meta, not, otherwise, or, section, shared, then, true, try, type, #binary, #date, #datetime, #datetimezone, #duration, #infinity, #nan, #sections, #shared, #table, #time