Функция Table.FromList M Power Query

Недавно столкнулся с неожиданной ошибкой в работе функции *Table.FromList()* языка M Power Query. Простой код…

Запрос1[1]

```
let
   source = Table.FromList({10, 20, 30, 40, 50})
in
   source
```

… не работает, и возвращает ошибку:

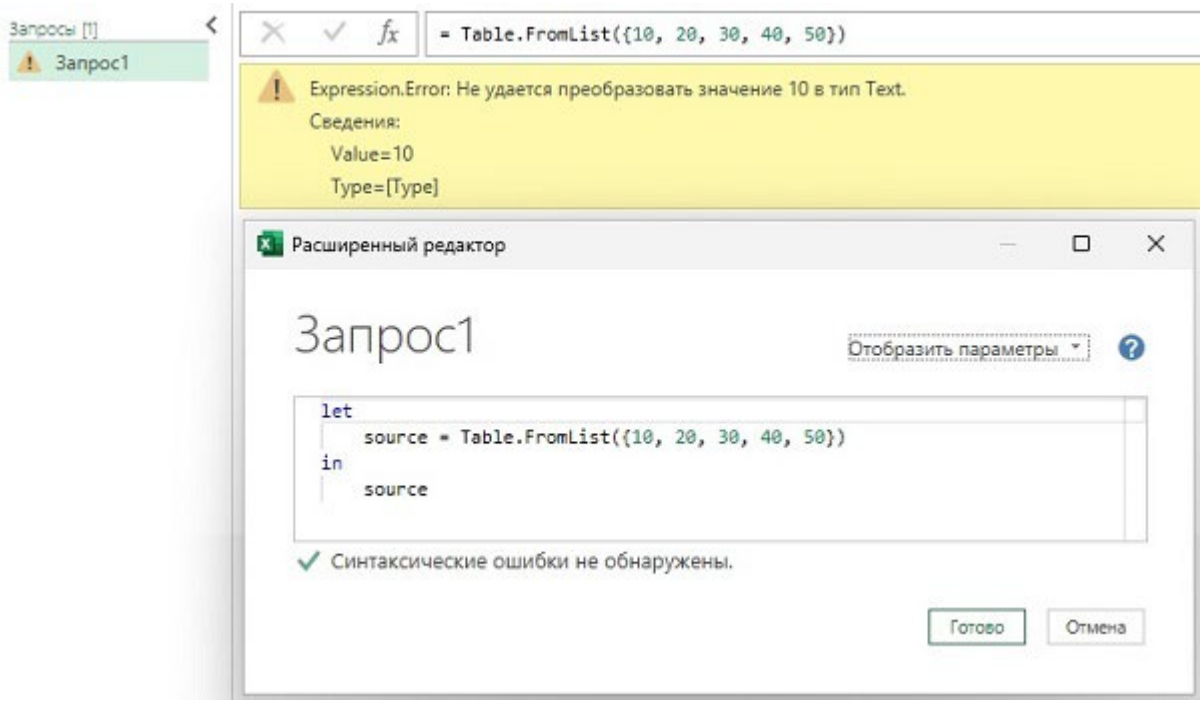


Рис. 1. Неожиданное поведение функции Table.FromList()

Интересно, а что будет, если подсунуть движку M логические значения или списки!?

Запрос2

```
let
   source = Table.FromList({false, true, true, false})
in
   source
```

Запрос3

```
let
   source = Table.FromList({{1, 2}, {2, 3}})
in
   source
```

Аналогичные ошибки. Похоже, Table.FromList() ожидает, что список состоит из текстовых элементов. Следующий код отрабатывает корректно:

Запрос4

```
let
   source = Table.FromList({"1", "2", "3", "4"})
in
   source
```

[1] Номер в заметке соответствует номеру запроса в приложенном Excel-файле.

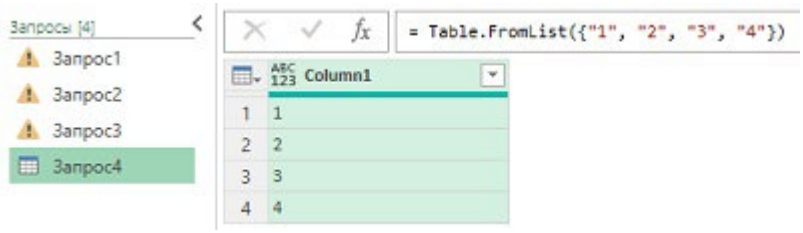


Рис.2. Текстовые элементы списка работают корректно

Официальная справка Microsoft сообщает, что функция имеет один обязательный и четыре необязательных параметра:

```
Table.FromList(
   list as list,
   optional splitter as nullable function,
   optional columns as any,
   optional default as any,
   optional extraValues as nullable number
) as table
```

Далее сказано, что функция *Table.FromList()* преобразует список *list* в таблицу путем применения заданной функции разбиения *splitter* к каждому элементу в списке. **По умолчанию предполагается, что список представляет собой список текстовых значений, разделенных запятыми** (выделение моё). Необязательный параметр *columns* может иметь следующие значения: число столбцов, список имен столбцов или тип таблицы. При необходимости также можно указать *default* и *extraValues*.

Обязательный параметр Список

Продолжим эксперименты. Что если предложить движку в качестве разделителя пробел без опциональных параметров!?

Запрос5

```
let
   source = Table.FromList({"1" "2" "3" "4"})
in
   source
```

… не работает:

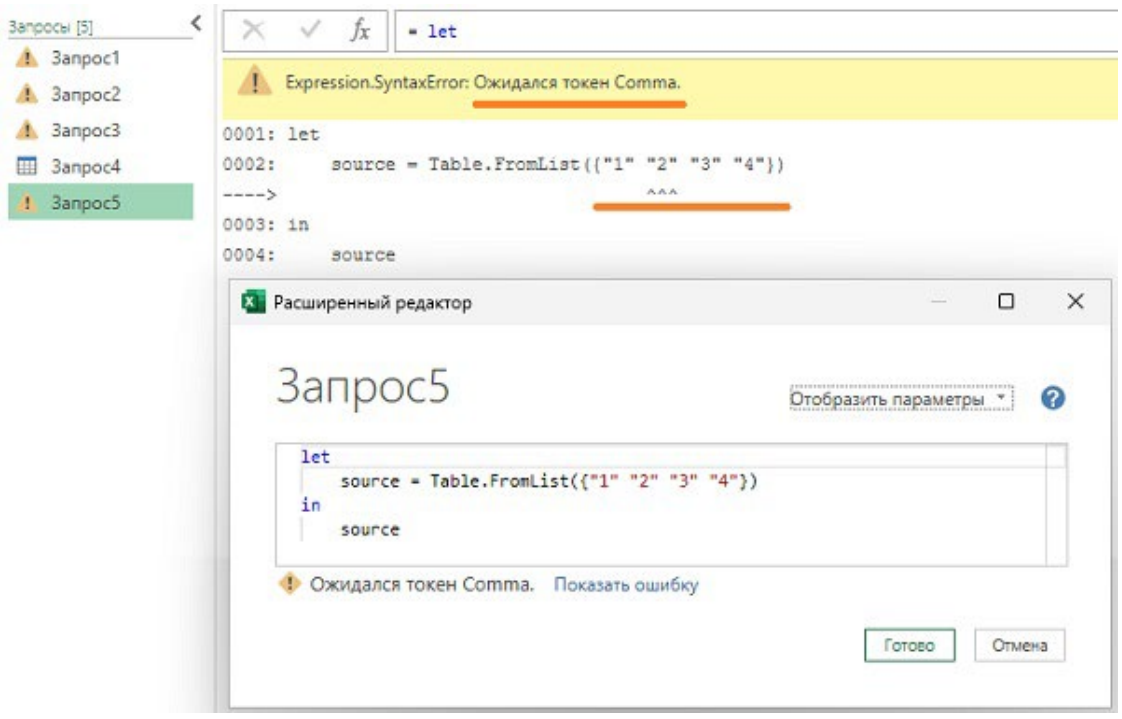


Рис. 3. Разделитель пробел не работает

Как и сказано в документации Microsoft, по умолчанию *Table.FromList()* ожидает запятую в качестве разделителя.

> Итак, Table.FromList() с одним параметром – списком – работает только, с *текстовыми элементами списка, разделенными запятыми*.

Скопируйте следующий код, и вставьте его в Расширенный редактор:

Запрос6

```
let
   tbl_1 = Table.FromList({"A,B,C","D,E,F"}),
   tbl_2 = Table.FromList({"A B C","D E F"}),
   tbl_3 = Table.FromList({"A;B;C","D;E;F"}),
   tbl_4 = Table.FromList({"A,B,C","D,E","F,G,H,I"}),
   tbl_5 = Table.FromList({"A,B,C,D","E,F,G","H,I"})
in
   tbl_5
```

Если в редакторе PQ вы последовательно выберите шаги запроса, сможете увидеть в каком виде возвращаются таблицы.

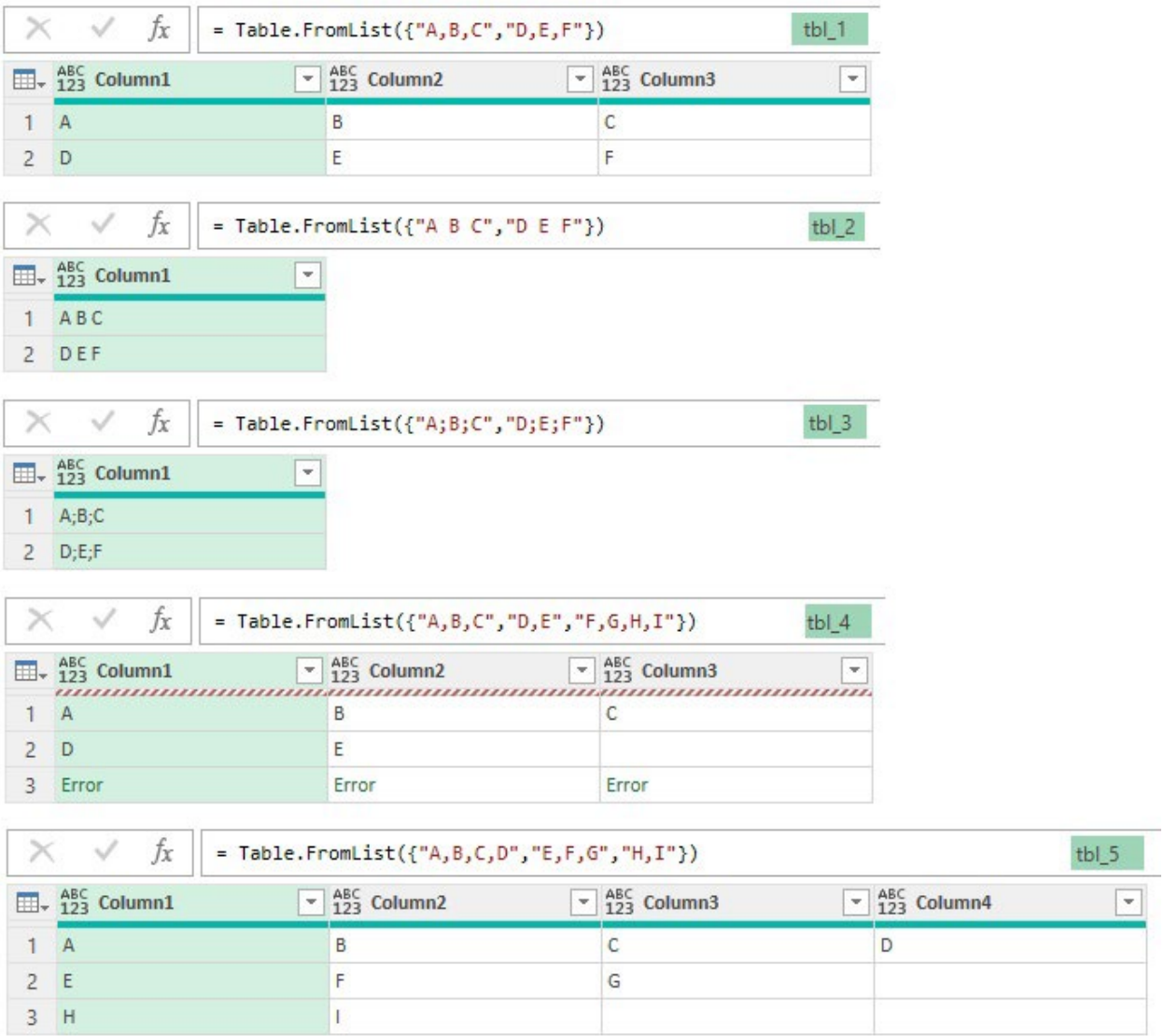


Рис. 4. Особенности работы функции Table.FromList() с одним параметром

Первые три списка содержат по два текстовых элемента, 4-й и 5-й – по три элемента. Количество элементов списка задает число строк таблицы. Число столбцов определяется тем, на сколько частей разделяется каждый элемент. tbl_1 разбивается на три столбца, поскольку внутри каждого элемента списка используется разделитель по умолчанию – запятая. tbl_2 и tbl_3 представлены

одним столбцом, так как внутри элементов списка нет разделителя по умолчанию – запятой. tbl_4 и tbl_5 демонстрируют еще одну особенность. Список преобразуется в таблицу с числом столбцов, на которое разбивается первый элемент списка. В tbl_4 первый элемент "A,B,C" разбивается на три столбца, а третий элемент "F,G,H,I", который хотел бы разбиться на четыре столбца, не знает, как это сделать, и возвращает ошибку. tbl_5 отражается без ошибок, потому что элемент списка с максимальным после разбиения числом частей является первым.

Обратите также внимание на имена столбцов: *Column1*, *Column2*, ... Поскольку мы не указали список имен столбцов в параметре *columns* функции *Table.FromList()*, движок M присвоил имена по умолчанию.

Второй параметр Splitter – функция-разделитель

Второй опциональный параметр функции *Table.FromList() splitter* является функцией, выполняющей разделение внутри элементов списка. В качестве *splitter* может выступать одна из стандартных библиотечных функций M, пользовательская функция или *null*.

Посмотрим, как отрабатывает преобразование списка в таблицу интерфейс редактора Power Query. Создайте пустой запрос и введите список:

Запрос7

```
let
   Источник = {1, 2, 3, 4}
in
   Источник
```

В редактора Power Query пройдите *Средства для списков –> Преобразование –> В таблицу*:

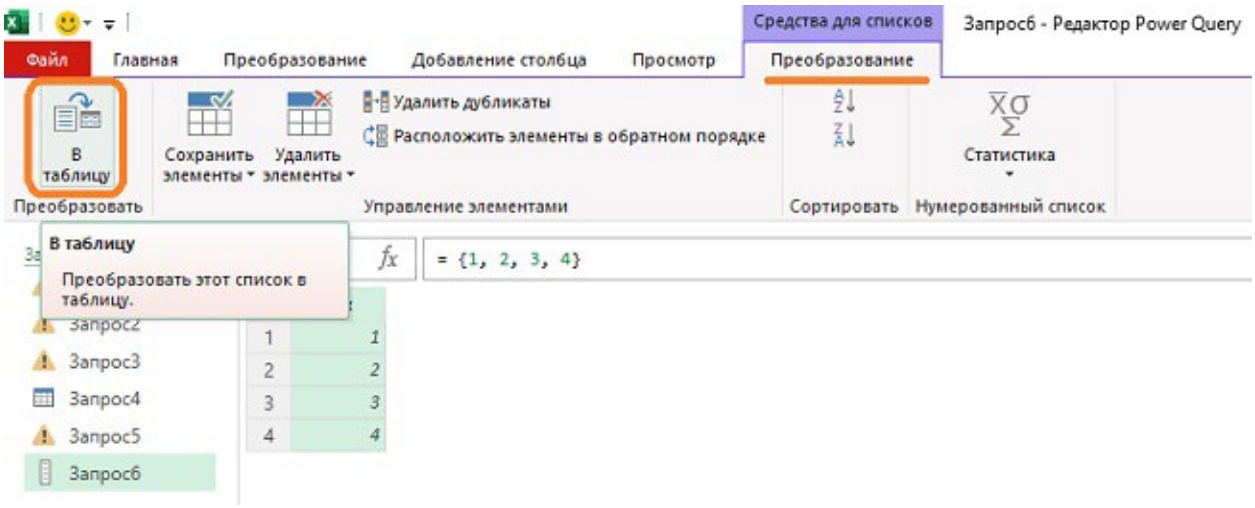


Рис. 5. Преобразование списка в таблицу

В открывшемся окне оставьте параметры по умолчанию:

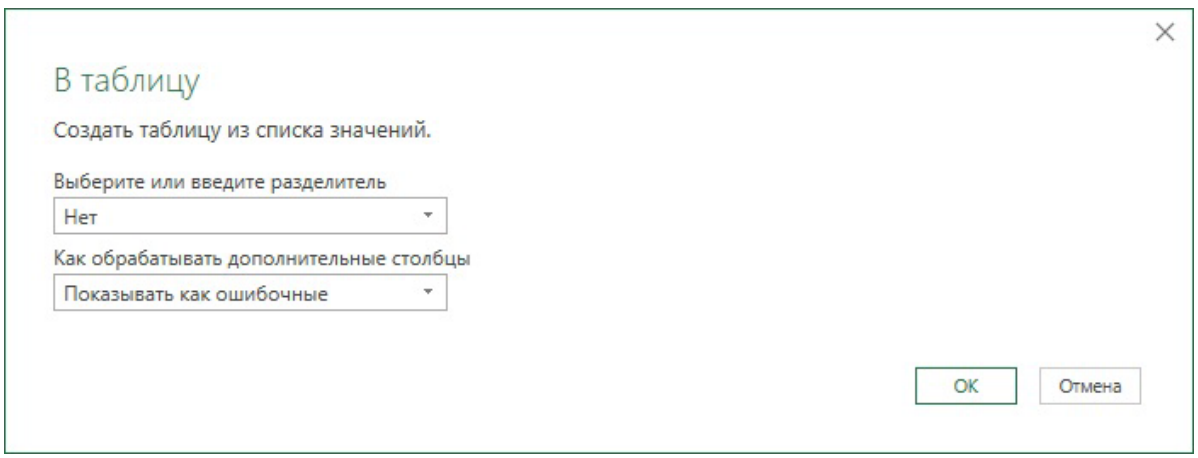


Рис. 6. Параметры преобразования в таблицу

Нажмите Ok. PQ преобразует список в таблицу, и отразит код операции в строке формул:

= Table.FromList(Источник, Splitter.SplitByNothing(), null, null, ExtraValues.Error)

Можно видеть, что интерфейс управляет двумя параметрами функции Table.FromList() – *splitter* и *extraValues*. Параметры *columns* и *default* получают значения по умолчанию – *null*, и в интерфейсе их изменить не получиться.

Разделители стандартной библиотеки M

	A	B
1	Имя	Описание
2	Splitter.SplitByNothing	Возвращает функцию, которая не разбивает текст, возвращая свой аргумент как единый список элементов.
3	Splitter.SplitTextByCharacterTransition	Возвращает функцию, которая разделяет текст на текстовый список в соответствии с переходом от одного типа символов к другому.
4	Splitter.SplitTextByAnyDelimiter	Возвращает функцию, которая разделяет текст, используя любой поддерживаемый разделитель.
5	Splitter.SplitTextByDelimiter	Возвращает функцию, которая будет разбивать текст в соответствии с разделителем.
6	Splitter.SplitTextByEachDelimiter	Возвращает функцию, которая разбивает текст каждым разделителем по очереди.
7	Splitter.SplitTextByLengths	Возвращает функцию, которая разбивает текст в соответствии с указанными длинами.
8	Splitter.SplitTextByPositions	Возвращает функцию, которая разбивает текст в соответствии с указанными позициями.
9	Splitter.SplitTextByRanges	Возвращает функцию, которая разбивает текст в соответствии с указанными диапазонами.
10	Splitter.SplitTextByRepeatedLengths	Возвращает функцию, которая разбивает текст на текстовый список в цикле, после каждого отрезка указанной длины.
11	Splitter.SplitTextByWhitespace	Возвращает функцию, которая разбивает текст в соответствии с пробелами.

Рис. 7. Функции-разделители стандартной библиотеки M

Для удобства обзора функций и констант языка M я вывел их все на лист Excel, используя…

Запрос8
#shared

См. лист *Библиотека*, приложенного Excel-файла.

Splitter.SplitByNothing

Не имеет аргументов. Не разбивает текст, возвращает элементы списка, как есть. Основное отличие от *null* или отсутствия аргумента в функции *Table.FromList()* в том, что разделитель *Splitter.SplitByNothing* справляется с нетекстовыми элементами списка.

Запрос9

```
let
   tbl_1 = Table.FromList({10, 20, 30, 40, 50}), // возвращает ошибку
   tbl_2 = Table.FromList({10, 20, 30, 40, 50}, null), // возвращает ошибку
   tbl_3 = Table.FromList({10, 20, 30, 40, 50}, Splitter.SplitByNothing()), // возвращает таблицу
   tbl_4 = Table.FromList({false, true, true, false}, Splitter.SplitByNothing()),
      // возвращает таблицу, сравни с Запрос2
   tbl_5 = Table.FromList({{1, 2}, {3, 4}}, Splitter.SplitByNothing())
      // возвращает таблицу, сравни с Запрос3
in
   tbl_5
```

Splitter.SplitTextByDelimiter

Разбивает текст на список в соответствии с указанным разделителем. Имеет два аргумента:

Splitter.SplitTextByDelimiter(

```
    delimiter as text,
    optional quoteStyle as nullable number
) as function
```

Опциональный аргумент quoteStyle

quoteStyle может принимать два значения. Официальная документации весьма лаконична:

QuoteStyle.Type

Определение

Указывает стиль кавычек.

Допустимые значения

Имя	Значение	Описание
QuoteStyle.None	0	Выбор символов кавычек не имеет значения.
QuoteStyle.Csv	1	Символы кавычек указывают начало строки в кавычках. Вложенные кавычки обозначаются двумя символами кавычек.

Применяется к

- Функции разделения

Рис. 8. Справка Microsoft по *QuoteStyle.Type*

Описание Microsoft является, как минимум странным… Эксперименты показывают, что выбор *QuoteStyle.Type* влияет на два аспекта:

- как движок реагирует на разделители внутри кавычек;
- как отражаются экранированные кавычки.

Напомню. В языке M Power Query кавычки " (двойные кавычки) используются для обозначения текстовых литералов. Они всегда используются парами: первые кавычки начинают текстовую строку, вторые – её завершают. Если нужно использовать сам знак кавычек, как часть текстовой строки, кавычки нужно экранировать.

Например,

"Это ""текст"" с экранированными кавычками".

В этом примере двойные кавычки " внутри строки экранированы удвоением. Это указывает, что они являются частью текстового значения, а не символами, обозначающими начало и конец строки.

При отображении текстовой строки кавычки пропадают, а при отображении экранированных кавычек, они преобразуются в обычные ""–>".

Запрос10

```
let
    tbl_1 = "tbl_1 - это простой текст",
    tbl_2 = "tbl_2 - это ""текст"" с экранированными кавычками"
in
    tbl_2
```

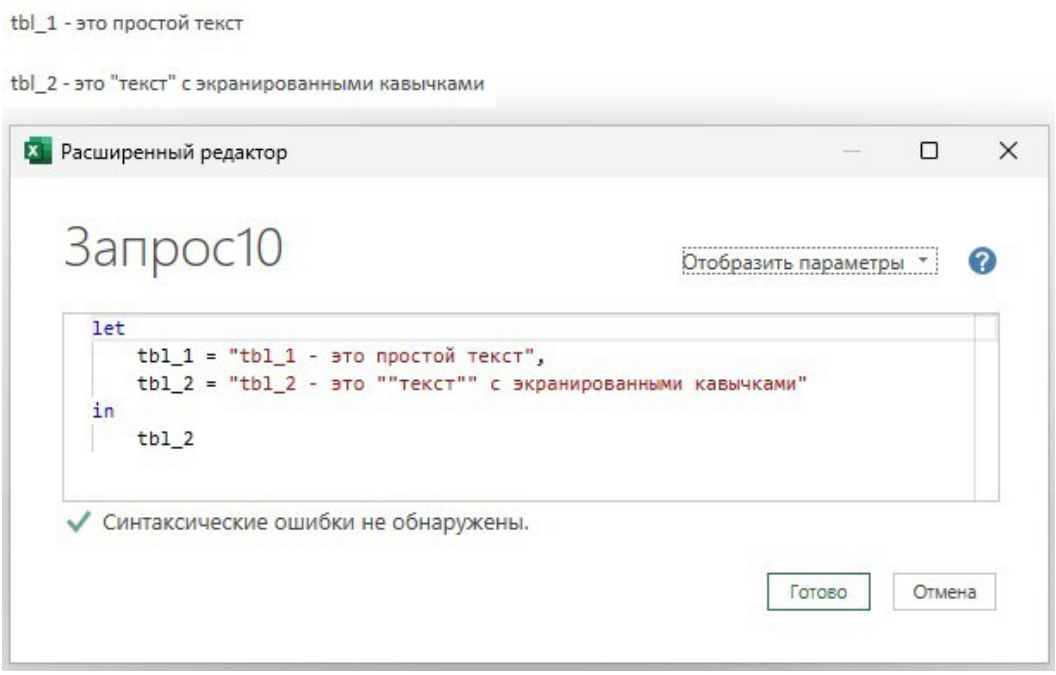


Рис. 9. Кавычки в коде и на экране предварительного просмотра

Итак в языке M *QuoteStyle.Type* может принимать два значения:

- *QuoteStyle.None* или 0: кавычки не влияют на разбор текста; все разделители сыграют свою роль; экранированные кавычки преобразуются в обычные двойные;
- *QuoteStyle.Csv* или 1: разделители внутри экранированных кавычек считаются элементами текстовой строки и не участвуют в разделении текста; сами кавычки (и обычные ", и экранированные "") при разборе пропадают.

Запрос11

```
let
   tbl_1 = Table.FromList({"a,b,c,d"}, Splitter.SplitTextByDelimiter(",", QuoteStyle.None)),
   tbl_2 = Table.FromList({"a,b,c,d"}, Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv)),
   tbl_3 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter(",", QuoteStyle.None)),
   tbl_4 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv))
in
   tbl_4
```

Рис. 10. Влияние параметра QuoteStyle.Type на обработку кавычек

В отсутствие экранированных кавычек (tbl_1 и tbl_2) параметры *QuoteStyle.None* и *QuoteStyle.Csv* ведут себя одинаково. При наличии экранированных кавычек (tbl_3) *QuoteStyle.None* «видит» разделитель внутри экранированных кавычек, и разбивает текст на 4 столбца. *QuoteStyle.Csv* (tbl_4) считает разделитель внутри экранированных кавычек элементом текста, и разбивает строку на три столбца.

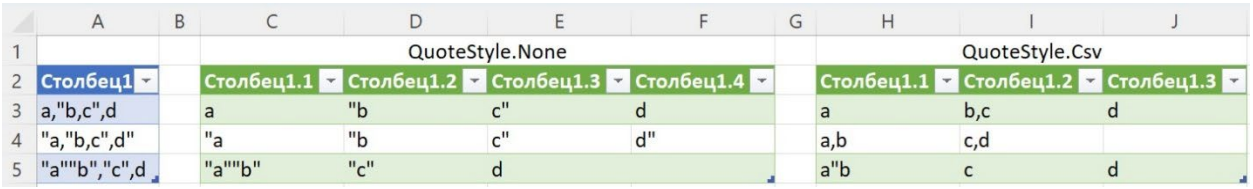
Разделитель может состоять из нескольких символов. Два последовательных разделителя в разделяемом тексте приводят к созданию одной пустой ячейки. Если разделитель стоит в начале текста, создается одна пустая ячейка:

Запрос12

```
let
   tbl_1 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter("a,", QuoteStyle.None)),
   tbl_2 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter("a,", QuoteStyle.Csv)),
   tbl_3 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter("b,", QuoteStyle.None)),
   tbl_4 = Table.FromList({"a,""b,c"",d"}, Splitter.SplitTextByDelimiter("b,", QuoteStyle.Csv))
in
   tbl_4
```

Рис. 11. Разделитель из нескольких символов и в начале текста

Учтите, что при импорте текста, например, из Excel, внешние и экранированные кавычки не нужны. Наличие таких кавычек будет говорить о том, что сам текст содержит кавычки. Сравните:



	A	B	C	D	E	F	G	H	I	J
1			QuoteStyle.None					QuoteStyle.Csv		
2	Столбец1		Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4		Столбец1.1	Столбец1.2	Столбец1.3
3	a,"b,c",d		a	"b	c"	d		a	b,c	d
4	"a,"b,c",d"		"a	"b	c"	d"		a,b	c,d	
5	"a""b","c",d		"a""b"	"c"	d			a"b	c	d

Рис. 12. Импорт текста из таблицы Excel

Здесь синяя таблица – исходная, импортируемая в PQ. А две зеленых – результат разбиения исходной таблицы на столбцы с помощью функции *Table.SplitColumn* и функции-разделителя *Splitter.SplitTextByDelimiter* с параметрами *QuoteStyle.None* и *QuoteStyle.Csv*.

Запрос13

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица1"]}[Content],
   #"Разделить столбец по разделителю" = Table.SplitColumn(
      Источник,
      "Столбец1",
      Splitter.SplitTextByDelimiter(",", QuoteStyle.None)
```

```
    )
in
    #"Разделить столбец по разделителю"
```

Splitter.SplitTextByAnyDelimiter

В справке Microsoft указано, что функция разбивает текст по всем указанным в списке разделителям. Имеет один обязательный и два необязательных параметра:

```
Splitter.SplitTextByAnyDelimiter(
    delimiters as list,
    optional quoteStyle as nullable number,
    optional startAtEnd as nullable logical
) as function
```

Опциональный параметр startAtEnd

Новый параметр *startAtEnd* по умолчанию равен *false*, и поиск разделителей стартует с начала текста. Если *startAtEnd = true*, поиск разделителей ведется с конца текста. Поскольку в разбиении участвует любой разделитель из списка, мне представляется, что параметр *startAtEnd* для функции *Splitter.SplitTextByAnyDelimiter()* не играет никакой роли…

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			startAtEnd = false							startAtEnd = true					
2	Столбец1		A	B	C	D	E	F		A	B	C	D	E	F
3	a;b,c_d,e,f;g		a	b	c_d	e	f	g		a	b	c_d	e	f	g
4	a,b,c_d;e,f;g		a	b	c_d	e	f	g		a	b	c_d	e	f	g

Рис. 13. Функция-разделитель *Splitter.SplitTextByAnyDelimiter*

Запрос15

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Таблица7"]}[Content],
    #"Разделить столбец по разделителю" = Table.SplitColumn(
        Источник,
        "Столбец1",
        Splitter.SplitTextByAnyDelimiter(
            {",",";"},
            QuoteStyle.Csv,
            true
        ),
    {"A", "B", "C", "D", "E", "F"})
in
    #"Разделить столбец по разделителю"
```

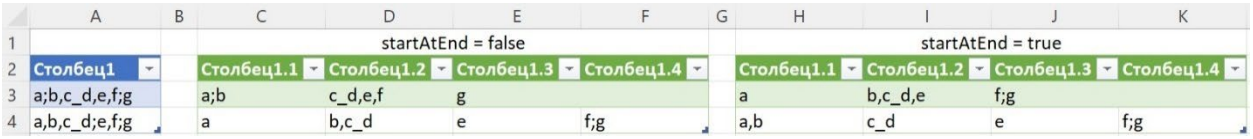
Splitter.SplitTextByEachDelimiter

В справке Microsoft указано, что функция разбивает текст по всем заданным разделителям в последовательности.

```
Splitter.SplitTextByEachDelimiter(
    delimiters as list,
    optional quoteStyle as nullable number,
    optional startAtEnd as nullable logical
) as function
```

Как обычно официальная документация слишком лаконична, а два приведенных в документации примера, как назло подобраны так, что не проясняют ситуацию.

На самом деле разделители используются последовательно в том порядке, как указаны в списке. Например, список разделителей {",", ";", ","} говорит, что в тексте произойдет разделение максимум на 4 столбца: сначала по первой найденной запятой, далее по точке с запятой, потом по вновь найденной запятой. Для этой функции становится понятным наличие параметра *startAtEnd*. Теперь направление разбора становится важным!



	Столбец1	Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4	Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4
		startAtEnd = false				startAtEnd = true			
3	a;b,c_d,e,f;g	a;b	c_d,e,f	g		a	b,c_d,e	f;g	
4	a,b,c_d;e,f;g	a	b,c_d	e	f;g	a,b	c_d	e	f;g

Рис. 14. Функция-разделитель *Splitter.SplitTextByEachDelimiter*

Запрос17

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица10"]}[Content],
   #"Разделить столбец по разделителю" = Table.SplitColumn(
      Источник,
      "Столбец1",
      Splitter.SplitTextByEachDelimiter(
         {",",";",","},
         QuoteStyle.Csv,
         false
      ), 4
   )
in
   #"Разделить столбец по разделителю"
```

Splitter.SplitTextByLengths

В справке Microsoft указано, что функция разбивает текст на список по каждой указанной длине.

```
Splitter.SplitTextByLengths(
   lengths as list,
   optional startAtEnd as nullable logical
) as function
```

Здесь описание функции и примеры в документах Microsoft адекватны)) Если текст длиннее суммы элементов в списке разделителей, часть текста после разделения будет потеряна.

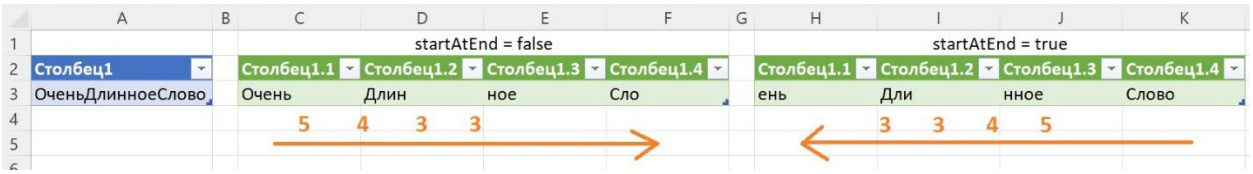


Рис. 15. Разделение текстовой строки по числу символов

Запрос19

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица13"]}[Content],
   #"Разделить столбец по положению" = Table.SplitColumn(
      Источник,
      "Столбец1",
      Splitter.SplitTextByLengths(
         {5, 4, 3, 3},
         false
      )
   )
in
   #"Разделить столбец по положению"
```

Splitter.SplitTextByPositions

Функция разбивает текст на текстовый список по всем указанным позициям.

```
Splitter.SplitTextByPositions(
   positions as list,
   optional startAtEnd as nullable logical
) as function
```

И здесь Microsoft хорошо справился с описанием и примерами. Помните, что счет позиций в текстовой строке начинается с нуля. Изучите примеры разбиения строки AABBBCCCCDDDDD:

	A	B	C	D	E	F	G
1		A A B B B C C C C D D D D D					
2		0 1 2 3 4 5 6 7 8 9 10 11 12 13					
3							
4	Список разбиения	startAtEnd	Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4	Столбец1.5
5	{0, 2, 5, 9}	false	AA	BBB	CCCC	DDDDD	
6	{1, 2, 5, 9, 11}	false	A	BBB	CCCC	DD	DDD
7	{0, 2, 5, 9}	true	AABBB	CCCC	DDD	DD	
8	{1, 2, 5, 9, 11}	true	AAB	BB	CCCC	DDD	D

Рис. 16. Разбиение по позициям

Запрос21

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица16"]}[Content],
   #"Разделить столбец по позициям" = Table.SplitColumn(
      Источник,
      "Столбец1",
      Splitter.SplitTextByPositions(
         {0, 2, 5, 9},
         false
      )
   )
in
   #"Разделить столбец по позициям"
```

Splitter.SplitTextByRanges

Разбивает текст на текстовый список по заданным значениям смещения и длины. Длина *null* указывает, что все остальные входные данные должны быть включены. Я использую две версии Power Query. В одной *null* работает, в другой выдает ошибку – *Не удается преобразовать значение null в тип Number.*

```
Splitter.SplitTextByRanges(
   ranges as list,
   optional startAtEnd as nullable logical
) as function
```

В Excel, когда мы используем функцию ПСТР (A1; 5; 3) мы переходим к пятому символу, а затем берем его и следующие два символа:

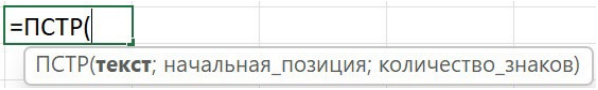


Рис. 17. Параметры функции ПСТР в Excel

Аналогично работает *Splitter.SplitTextByRanges*. Функция принимает аргумент, который выглядит например так: { { 0, 2 } , { 3, 1 } }. Каждая пара значений определяет положение и число символов, как и ПСТР(). Изучите приведенные ниже примеры:

	A	B	C	D	E	F
1	Текст	Список смещений и длин	startAtEnd		Текст.1	Текст.2
2	codelimiter	{{0, 4}, {2, 10}}	false		code	delimiter
3	RedmondWA?98052	{{0, 5}, {6, 2}}	true		WA	98052
4	RedmondWA?98052	{{0, 5}, {6, null}}	true		RedmondWA	98052

Рис. 18. Разбиение по диапазонам

Обратите внимание, что подстроки могут перекрываться.

Splitter.SplitTextByRepeatedLengths

Разбивает текст на текстовый список в цикле, после каждого отрезка указанной длины.

```
Splitter.SplitTextByRepeatedLengths(
   length as number,
   optional startAtEnd as nullable logical
) as function
```

Эта функция похожа на *Splitter.SplitTextByLengths*, за исключением того, что все длины имеют одинаковое значение. Например…

```
Splitter.SplitTextByRepeatedLengths(3)
```

… разделит текст на группы по три символа. В последнем столбце символов может быть меньше.

	A	B	C	D	E	F	G
1	Текст	length (знаков)	startAtEnd		Текст.1	Текст.2	Текст.3
2	12345678	3	false		123	456	78
3	12345678	3	true		12	345	678
4	12345678	0	false				

Рис. 19. Разбиение на одинаковое число символов

Splitter.SplitTextByWhiteSpace

Разбивает текст на текстовый список по пробелам.

```
Splitter.SplitTextByWhitespace(
   optional quoteStyle as nullable number
) as function
```

Аналогична функции *Splitter.SplitTextByDelimiter*, когда разделителем является пробел. При quoteStyle.Type = quoteStyle.Csv пробелы внутри экранированных кавычек не служат разделителями:

	A	B	C	D	E	F	G
1			QuoteStyle.None				
2	Столбец1		Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4	Столбец1.5
3	Это простой текст с пробелами		Это	простой	текст	с	пробелами
4	Это текст с" экранированными "пробелами		Это	текст	с"	экранированными	"пробелами
5							
6			QuoteStyle.Csv				
7	Столбец1		Столбец1.1	Столбец1.2	Столбец1.3	Столбец1.4	Столбец1.5
8	Это простой текст с пробелами		Это	простой	текст	с	пробелами
9	Это текст с" экранированными "пробелами		Это	текст	с экранированными пробелами		

Рис. 20. Разделение текстовой строки по пробелам

В дополнение к обычному пробелу (код ASCII 32), пробелами считается довольно много иных символов. Они подробно рассмотрены в документации Microsoft. Вот наиболее интересные:

- табуляция, код символа ASCII 9, или Escape-последовательность #(tab),
- перевод строки, код символа ASCII 10 или #(lf),
- возврат каретки, код символа ASCII 13 или #(cr).

Точный набор символов, которые считаются разделителями пробелов, может различаться в зависимости от версии Power Query или среды приложения.

Splitter.SplitTextByCharacterTransition

Разделяет текст на текстовый список в соответствии с переходом от одного типа символов к другому. Параметры *before* и *after* могут быть либо списком символов, либо функциями, которые принимают символы и возвращают значения *true* или *false*. Если символ из первого списка стоит перед каким-либо символом из второго списка, текст будет разделен в этом месте. Вместо символов из списков *before* и *after* можно использовать логические значения. Разделение сработает если оба значения будут *true.*

```
Splitter.SplitTextByCharacterTransition(
   before as anynonnull,
```

```
    after as anynonnull
) as function
```

Интерфейс редактора Power Query использует несколько предустановленных возможностей функции *Splitter.SplitTextByCharacterTransition*:

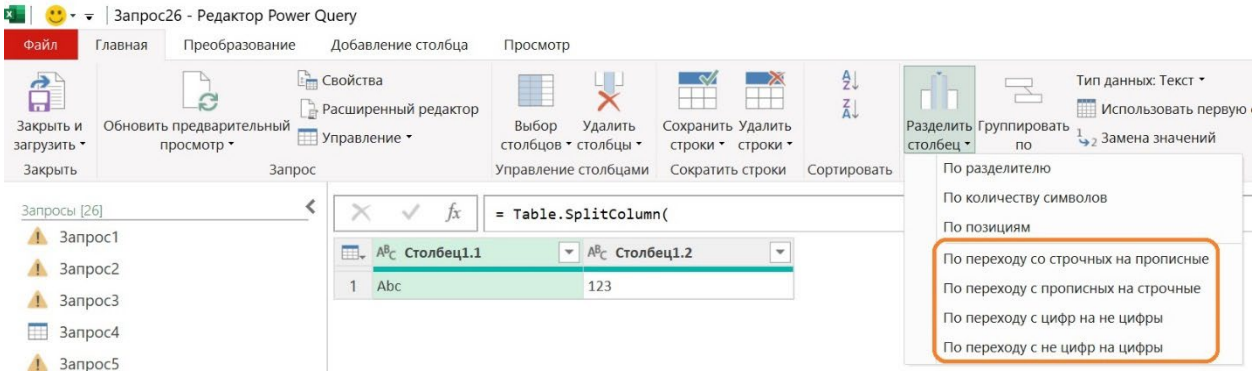


Рис. 21. Разделители в интерфейсе редактора Power Query

Поэкспериментируйте с этими четырьмя опциями. Посмотрите, какой код формирует каждая из них. Например, для *перехода с не цифр на цифры*:

Запрос26

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица29"]}[Content],
   #"Разделить столбец по переходам символов" = Table.SplitColumn(
      Источник,
      "Столбец1",
      Splitter.SplitTextByCharacterTransition(
         (c) => not List.Contains({"0".."9"}, c),
         {"0".."9"}
      ),
      {"Столбец1.1", "Столбец1.2"})
in
   #"Разделить столбец по переходам символов"
```

Здесь параметр *before* определяется пользовательской функцией. Её часть – библиотечная функция *List.Contains()* – указывает, содержит ли список {"0".."9"} значение *c*. Если значение найдено в списке, возвращается *true*; в противном случае — *false*. Оператор *not* переворачивает логическое значение. Таким образом, аргумент *before* вернет *true*, если оцениваемый символ не цифра, и *false,* если цифра. Параметр *after* = {"0".."9"}. Разделитель сработает, если после не цифры идет цифра. Ровно то, что нам и требовалось. Пример такого разделения см. (1) на рис. 22.

	A	B	C	D	E	F	G	H	I	J
1										
2	Текст	Befor	After		Столбец	Столбец	(1)			
3	Abc123	не цифра	цифра		Abc	123				
4										
5	Текст	Befor	After		Текст.1	Текст.2	Текст.3	Текст.4	Текст.5	(2)
6	ABAACBBCB	{"A", "B"}	{"B", "C"}		A	BAA	CB	B	CB	
7										
8	Текст	Befor	After		Текст.1	Текст.2	Текст.3	Текст.4		
9	AABBBCCDD	<= "B"	> "B"		AABBB	CCDD				(3)
10	BBAFFF	<= "B"	> "B"		BBA	FFF				
11	AAAGGGBBFF	<= "B"	> "B"		AAA	GGGBB	FF			

Рис. 22. Разделение по переходу от одних символов к другим

Разделение в (2) происходит каждый раз, когда левый символ равен *A* или *B*, а правый – *B* или *C*. Разделение в (3) происходит на основе пользовательской функции:

Запрос28

```
let
   Источник = Excel.CurrentWorkbook(){[Name="Таблица33"]}[Content],
   #"Удаленные столбцы" = Table.RemoveColumns(Источник,{"Befor", "After"}),
   #"Разделить столбец по переходам символов" = Table.SplitColumn(
      #"Удаленные столбцы",
      "Текст",
      Splitter.SplitTextByCharacterTransition(
         each if _ <= "B" then true else false,
         each if _ > "B" then true else false
      )
   )
in
   #"Разделить столбец по переходам символов"
```

Если левый символ <= "B", а правый > "B", текстовая строка будет разделена. Учитывайте, что коды *ПРОПИСНЫХ* букв меньше кода *строчных*.

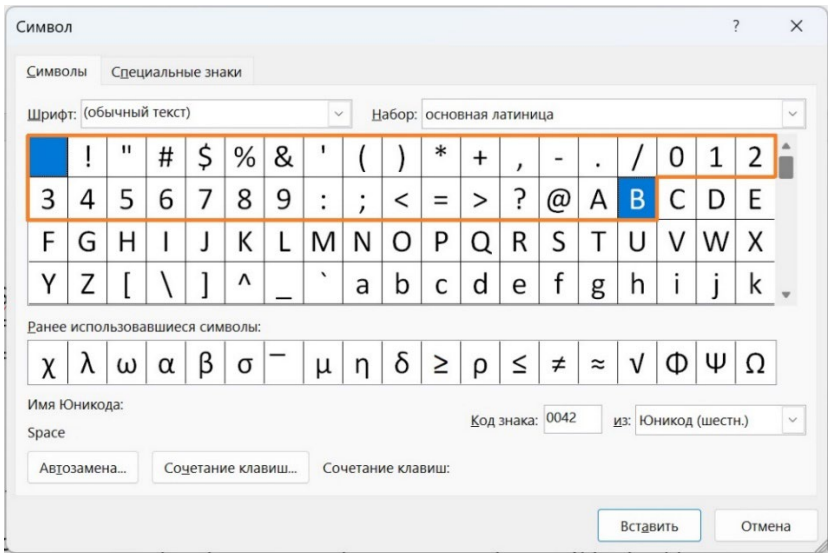


Рис. 23. Символы в порядке возрастания их кодов

Функция Record.FieldValues в роли разделителя

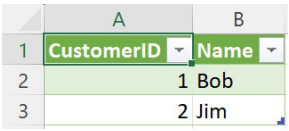
Если список состоит из записей, то в качестве разделителя можно использовать функцию *Record.FieldValues*. В документации Microsoft сказано, что функция *Record.FieldValues* возвращает список значений полей записи, и имеет синтаксис:

```
Record.FieldValues(record as record) as list
```

Запрос29

```
let
   Источник = Table.FromList(
      {
         [CustomerID = 1, Name = "Bob"],
         [CustomerID = 2, Name = "Jim"]
      },
      Record.FieldValues,
      {"CustomerID", "Name"}
   )
in
   Источник
```

Запрос 29 вернет таблицу, состоящую из значений полей записи:



	A	B
1	CustomerID	Name
2	1	Bob
3	2	Jim

Рис. 24. Функция-разделитель *Record.FieldValues*

Опциональные параметры columns, default и extraValues

Документация Microsoft описывает эти параметры предельно лаконично. Необязательный *columns* может иметь следующие значения: число столбцов, список имен столбцов или тип таблицы. При необходимости также можно указать *default* и *extraValues*.

Понимай, как хочешь!

Мы рассмотрим эти параметры совместно, так как они взаимоувязаны.

columns может быть:

- числом – количество столбцов, на которые будут разбиты элементы списка,
- списком имен столбцов,
- *type table []*; запись содержащая пары полей: имя столбца и тип столбца.

Аргумент *columns* по умолчанию равен *null*.

Если после разделения элементов списка их недостаточно, чтобы заполнить все предоставленные им столбцы, *default* задаст значение, которое будет возвращено. По умолчанию *default* = *null*.

Если после разбиения элементов списка у нас недостаточно столбцов для их размещения, аргумент *extraValues* определит, что будет отображаться:

- *ExtraValues.Ignore* – разделенные элементы заполнят все предоставленные столбцы, а те элементы, для которых столбцов не хватило, будут отброшены (проигнорированы),
- *ExtraValues.List* – последний столбец будет содержать все избыточные значения в виде списка,
- *ExtraValues.Error* – все столбцы для такой строки будут содержать ошибку.

Значением по умолчанию является *ExtraValues.Error*. Часто это полезно, поскольку обращает внимание, что столбцов недостаточно. *ExtraValues.Ignore* «прячет» ошибку. Значения выглядят нормально, и нехватку столбцом можно не заметить.

Указание числа столбцов

В следующем коде элементы списка {"abcd","abc","abcde"} разделяются на три или четыре столбца:

Запрос30

```
let
   Источник = {"abcd","abc","abcde"},
   tbl_1 = Table.FromList(Источник, Splitter.SplitTextByRepeatedLengths(1, false), 4, "н/д",
ExtraValues.Ignore),
   tbl_2 = Table.FromList(Источник, Splitter.SplitTextByRepeatedLengths(1, false), 3, "н/д",
ExtraValues.List),
   tbl_3 = Table.FromList(Источник, Splitter.SplitTextByRepeatedLengths(1, false), 4, "н/д",
ExtraValues.Error)
in
   tbl_3
```

Источник – список из трех элементов, т.е. таблица будет содержать три строки. С помощью функции *Splitter.SplitTextByRepeatedLengths(1, false)* каждый элемент списка разделяется по одному символу. Изучите результаты Запроса 30:

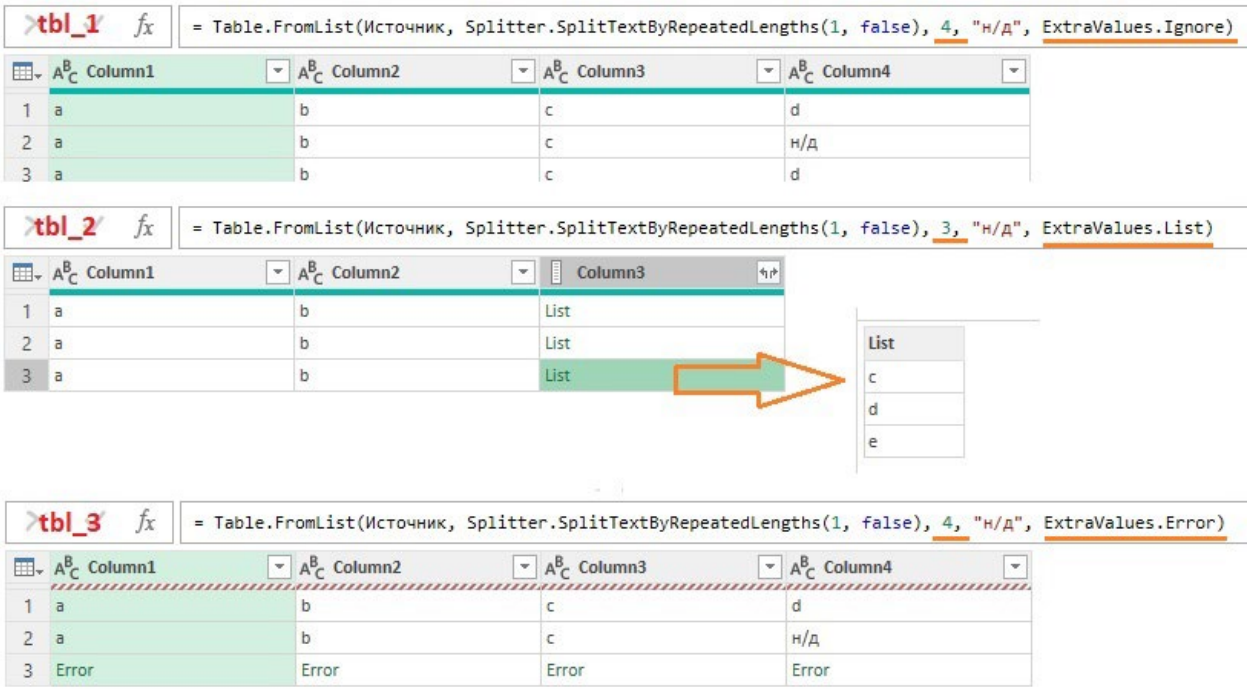


Рис. 25. Параметр *columns* – число

Как заметил buchlotnik, число столбцов не обязательно константа – его можно и вычислить. Например…

Запрос30а

```
let
   Источник = {{"q"},{"a","s"},{"z","x","c"}},
   B_CSV = Table.FromList(
      Источник,
      (x)=>x,
      List.Max(Источник,null,List.Count)
   )
in
   B_CSV
```

… вернет таблицу с 3 столбцами. Источник состоит из трех списков, больше всего элементов в последнем списке, число элементов – три.

Указание имен столбцов

Имена столбцов перечисляются, как текстовые литералы элементы списка. В Запросе 31 имена столбцов {"A", "B", "C", "D"}

Запрос31

```
let
   Результат = Table.FromList(
      {"a b c d","a b null","a b c d e"},
      Splitter.SplitTextByWhitespace(),
      {"A", "B", "C", "D"}
   )
in
   Результат
```

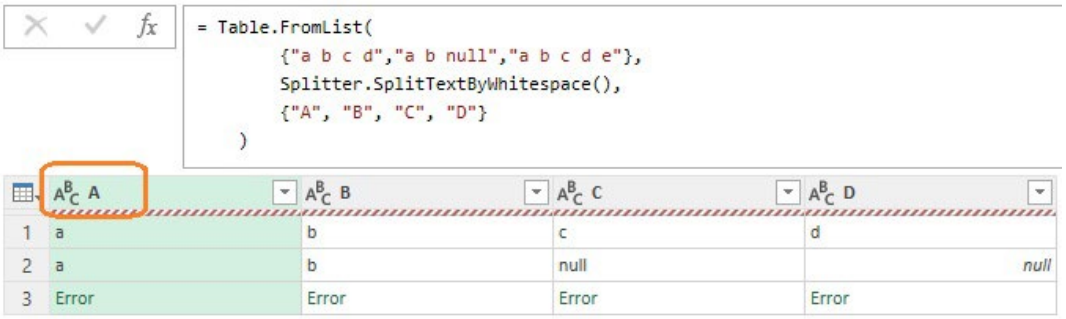


Рис. 26. Параметр *columns* – список имен столбцов

Указание имен и типов столбцов

Чтобы задать имена и типы столбцов, в аргументе *columns* используется синтаксис

type table [Имя1 = type1, Имя2 = type2, …]

Например:

Запрос33

```
let
   Таблица = Table.FromList(
      {"ПК, 30 000", "Ноутбук, 50 000"},
      Splitter.SplitTextByDelimiter(","),
      type table [Наименование = text, Цена = number]
   )
in
   Таблица
```

Сравните, простое именование столбцов (слева) и именование столбцов с одновременным заданием типов:

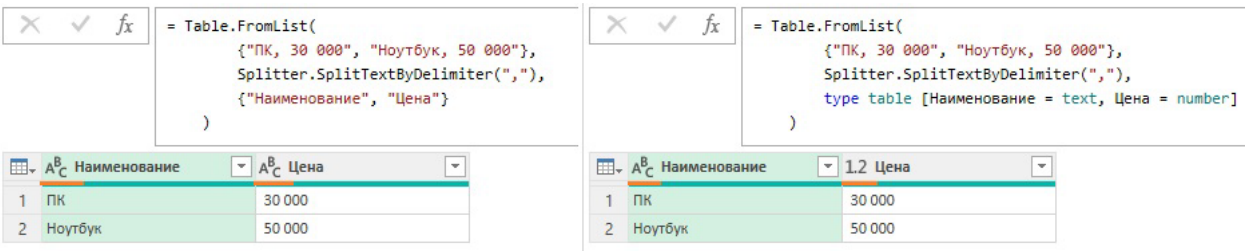


Рис. 27. Параметр *columns* – запись с именами и типами столбцов

Дополнение от 1 июня 2023 г. Как указано в документации Microsoft, в качестве разделителя может выступать пользовательская функция. А buchlotnik в своем телеграмм-канале приводит несколько примеров таких функций.

Запрос34

```
let
   Источник = {"a;b;c;1","d;e;f;2","g;h;i;3"},
   B_CSV = Table.FromList(
      Источник,
      (x)=>Text.Split(x,";")
   )
in
   B_CSV
```

Здесь разделить – пользовательская функция (x)=>Text.Split(x,";"). Она берет на вход по одному элементу списка, и разделяет его с помощью библиотечной функции Text.Split(). Также будет работать синтаксис (_)=>Text.Split(_,";").

Запрос37

```
let
   Источник = {"abc1","def2","ghi3"},
```

```
    B_CSV = Table.FromList(
        Источник,
        Text.ToList
    )
in
    B_CSV
```

И не спрашивайте меня, почему здесь достаточно простого синтаксиса *Text.ToList*, а не строгого *(x)=>Text.ToList(x)* или *(_)=>Text.ToList(_).*

В комментариях buchlotnik предложил решение проблемы, озвученной в начале заметки. С ошибкой функции *Table.FromList()* при обработке списка чисел {10, 20, 30, 40, 50} справится пользовательская функция (x)=>x. Сравните два варианта её использования:

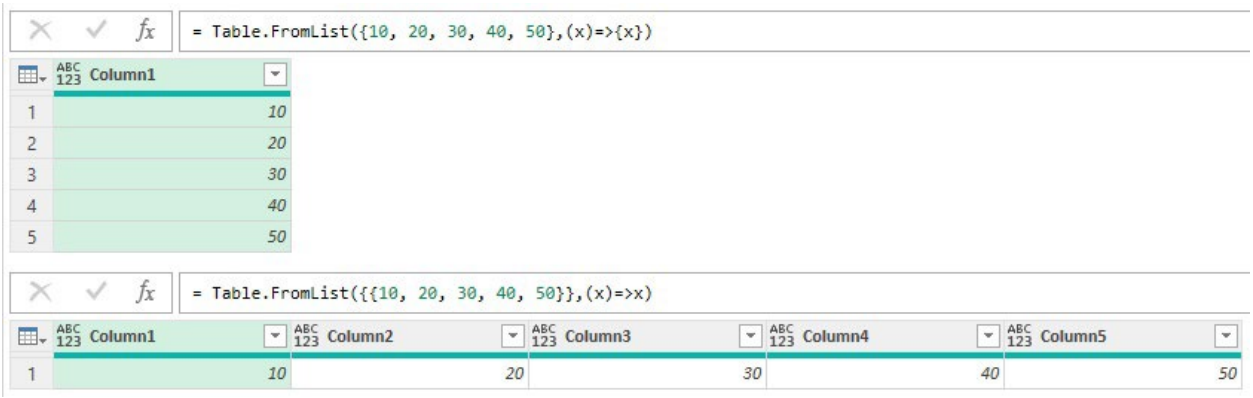


Рис. 28. Разделитель – пользовательская функция (x)=>x и (x)=>{x}

Пользовательскую функцию можно дополнить параметром *type table*:

Запрос38

```
let
    Источник = {"a;b;c;1","d;e;f;2","g;h;i;3;4;5"},
    B_CSV = Table.FromList(
        Источник,
        (_)=>Text.Split(_,";"),
        type table [а=text,б=text,в=text,г=number,д=number,е=number]
    )
in
    B_CSV
```

Сама по себе эта возможность уже обсуждалась выше. Любопытно другое. В Запросе 38 типы столбцов установлены в соответствии с типами данных, а вот сами значения остались текстовыми. Обратите внимание, на рис. 29 числа отформатированы влево, как текст:

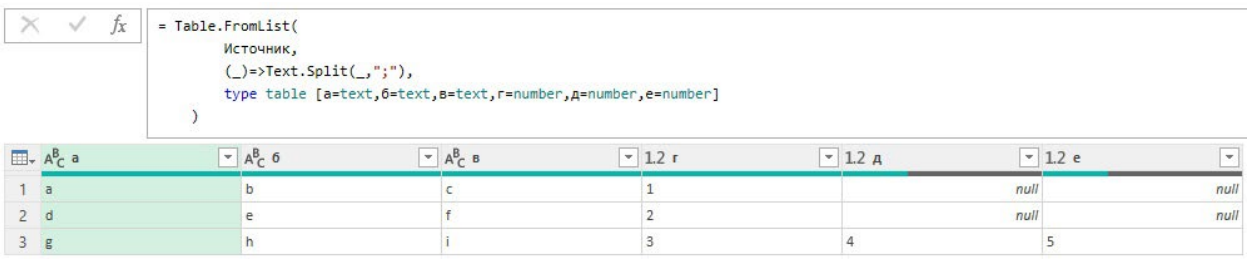


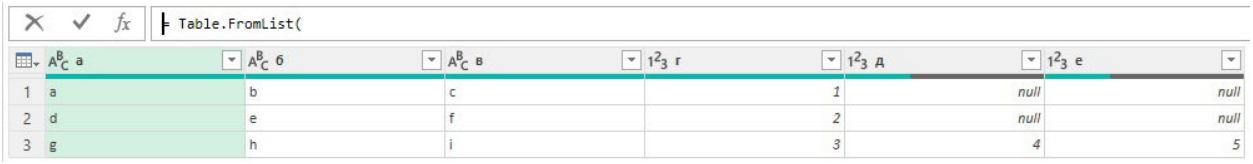
Рис. 29. Тип столбцов – численный, а сами значения текстовые

Поскольку разделитель может быть любой функцией, возвращающей список, buchlotnik предлагает более сложную пользовательскую функцию, которая не только разделит элементы списка по разделителю, но и преобразует числа как текст в настоящие числа:

Запрос39

```
let
```

```
  Источник = {"a;b;c;1","d;e;f;2","g;h;i;3;4;5"},
  B_CSV = Table.FromList(
    Источник,
    (x)=>List.Transform(
      Text.Split(x,";"),
      (y)=>try Number.From(y) otherwise y
    ),
    type table
    [
      а = text,
      б = text,
      в = text ,
      г = Int64.Type,
      д = Int64.Type,
      е = Int64.Type
    ]
  )
in
  B_CSV
```



	а	б	в	г	д	е
1	a	b	c	1	null	null
2	d	e	f	2	null	null
3	g	h	i	3	4	5

Рис. 30. Тип значений и столбцов – число

Здесь числа в столбцах выключены вправо. А тип десятичных чисел *number* заменен на целые *Int64.Type*.

Использованные материалы

Форум MrExcel.com. https://www.mrexcel.com/board/threads/table-fromlist.1204740/

Форум Microsoft Power BI Community. https://community.powerbi.com/t5/Power-Query/How-to-create-table-from-list-while-specifying-column-name-as/td-p/2249421

Rick de Groot. Create Tables from Scratch in Power Query M (40+ Examples). https://gorilla.bi/power-query/creating-tables/

Chris Webb. An In-Depth Look At The Csv.Document M Function. https://blog.crossjoin.co.uk/2018/03/09/an-in-depth-look-at-the-csv-document-m-function/

Splitter functions in Power Query. https://bizkapish.com/power-query/splitter-functions-in-power-query/

Запросы Chat GPT. https://chat.openai.com/