**Глава 22. Продвинутая условная логика Power Query**

Это продолжение перевода книги Кен Пульс и Мигель Эскобар. Язык М для Power Query. Главы не являются независимыми, поэтому рекомендую читать последовательно.

[Предыдущая глава](http://baguzin.ru/wp/?p=21375) [Содержание](http://baguzin.ru/wp/?p=20852#content) Следующая глава

В [главе 18](http://baguzin.ru/wp/?p=21280) вы узнали, как перенести в Power Query возможности функций Excel ЕСЛИ() и ЕСЛИОШИБКА(). Перенос других функций из библиотеки условной логики Excel сложнее. Он основан на использовании списков и пользовательских функций.



Рис. 22.1. Исходные данные

Иногда вам будет нужно отфильтровать данные по нескольким критериям. Откройте файл Multi-Column Logic.xlsx (см. рис. 22.1).

### Репликация функции Excel ИЛИ()

Допустим вам нужно отфильтровать строки для которых Inventory Item = Talkative Parrot ИЛИ Sold By = Fred. Если это данные Таблицы Excel, можно использовать расширенный фильтр или добавить столбец фильтрации:



Рис. 22.2. Столбец фильтрации на основе логического ИЛИ

Посмотрим, что можно сделать в Power Query. Удалите столбец Filter. Выберите любую ячейку в Таблице. Пройдите по меню *Данные* –> *Из таблицы/диапазона*. В редакторе Power Query щелкните правой кнопкой мыши столбец *Date* –> *Тип изменения* –> *Дата*. Поскольку вы не можете фильтровать данные без потери части записей, вам нужно добавить пользовательский столбец и применить формулу для проверки каждой строки.

*Добавление столбца* –> *Настраиваемый столбец*. Назовите столбец *Match* (поиск, совпадение). Вспомните, в [главе 18](http://baguzin.ru/wp/?p=21280) был описан синтаксис условной логики Power Query:

=if <logical test> then <result> else <alternate result>

К сожалению, в Power Query нет функции ИЛИ(). Напомню, чтобы обратиться к списку функций, кликните на ссылку [Сведения о формулах Power Query](https://docs.microsoft.com/ru-ru/powerquery-m/power-query-m-function-reference) в нижней части диалогового окна *Настраиваемый столбец*. Вы окажитесь на странице сайта Microsoft с обзором всех функций Power Query. В категории [List functions](https://docs.microsoft.com/ru-ru/powerquery-m/list-functions) можно найти функцию [List.AnyTrue](https://docs.microsoft.com/ru-ru/powerquery-m/list-anytrue), которая возвращает ИСТИНА, если хоть одно выражение списка истинно. В документации по функции приведено два примера:

Определяет, является ли хотя бы одно из выражений в списке {true, false, 2 > 0} истинным…

List.AnyTrue({true, false, 2>0})

… и возвращает true.

Определяет, является ли хотя бы одно из выражений в списке {2 = 0, false, 2 < 0} истинным…

List.AnyTrue({2 = 0, false, 2 < 0})

… и возвращает false.

Попробуем использовать эту функцию в качестве теста в пользовательском столбце:

=if List.AnyTrue

(

{[Inventory Item]="Talkative Parrot",[Sold By]="Fred"}

)

then "Meets Criteria!"

else "No Match"

Не забудьте разделить критерии запятыми, а список критериев окружить фигурными скобками, потому что функция List.AnyTrue() в качестве параметра требует список. Нажав Ok, вы увидите, что тест возвращает корректные результаты:



Рис. 22.3. Аналог функции Excel ИЛИ() в Power Query – List.AnyTrue()

Отфильтруйте столбец *Match*. Измените имя запроса – *pqOR*. *Главная* –> *Закрыть и загрузить*. Power Query возвращает таблицу, содержащую в общей сложности 88 строк из исходных 332. Если столбец *Match* вам не нужен, его можно удалить. Поскольку Power Query обрабатывает команды последовательно, удаление столбца *Match* не приведет к снятию фильтра:



Рис. 22.4. Отобранные по критерии ИЛИ записи в Таблице на листе Excel

### Репликация функции Excel И()

Для этих целей в Power Query есть функция List.AllTrue(). Эта функция возвращает истинное значение только в том случае, если каждый логический тест возвращает истинное значение. В Excel щелкните правой кнопкой мыши запрос *pqOR* –> *Дублировать*. Переименуйте запрос – *pqAND*. Если вы удалили столбец *Match*, удалите этот шаг, чтобы вернуть столбец *Match* в запрос. Щелкните значок шестеренки рядом с шагом *Добавлен пользовательский столбец* (для редактирования формулы). Заменить *List.AnyTrue* на *List.AllTrue*. Выберите шаг Строки с примененным фильтром. Осталось лишь шесть строк:



Рис. 22.5. Использование функции *List.AllTrue()* для отбора записей

Хотя в этих примерах мы явно отфильтровали данные на основе столбца *Match*, самое замечательное в функциях *List.AnyTrue* и *List.AllTrue* заключается в том, что вы можете помечать записи без фильтрации. Это добавит вам гибкости при построении более сложной логики, с возможностью сохранения всех исходных данных (чего нельзя получить просто фильтруя столбцы).

### Репликация функции Excel ВПР()

Репликация зависит от того, какая версия ВПР/VLOOKUP вам нужна. При поиске точного совпадения репликация может быть получена простым объединением двух таблиц (см. [главу 9](http://baguzin.ru/wp/?p=21076)). Репликация приблизительного соответствия ВПР() требует довольно сложной логики (подробнее о функции ВПР в Excel см. [Билл Джелен. Всё о ВПР: от первого применения до экспертного уровня](http://baguzin.ru/wp/?p=12467)). В примере вы не будете создавать сценарий Power Query с нуля но увидите как он работает. Откройте файл *Emulating VLOOKUP.xlsx*. В нем есть две таблицы:



Рис. 22.6. Таблица подстановки



Рис. 22.7. Таблица данных

Столбцы B:D таблицы данных содержат функции VLOOKUP() соответствующие заголовкам столбцов. Каждый столбец ищет значение, из столбца A для этой строки в таблице подстановки. Столбцы B и D возвращают значение из столбца 2 (G) таблицы подстановки, а столбец C – из столбца 3 (Н). Также обратите внимание, что столбцы B и C возвращают приблизительные совпадения, поскольку четвертый параметр функции VLOOKUP = True или опущен. Столбец D запрашивает точное совпадение (четвертый параметр = False), в результате чего все записи возвращают #N/A, за исключением последней строки.

Давайте поместим сценарий Power Query в файл, а затем посмотрим, как он реплицирует функцию VLOOKUP() Excel. В проводнике Windows кликните на файл *pqVLOOKUP.txt* Он откроется в Блокноте. Выделите и скопируйте в буфер все содержимое файла. Вернитесь в Excel. *Данные* –> *Получить данные* –> *Из других источников* –> *Пустой запрос* –> *Расширенный редактор*. Выделите всю заготовку кода в окне. Ctrl+V (вставив из буфера ранее скопированный код). Нажмите кнопку *Готово.* Переименуйте функцию *pqVLOOKUP. Главная* –> *Закрыть и загрузить* (функции сохраняются только в режиме подключения).

При работе с функцией вам понадобится указатель на таблицу подстановки BandingLevels. Выберите любую ячейку в ней –> *Данные* –> *Из таблицы/диапазона*. *Главная* –> *Закрыть и загрузить...* –> *Только создать подключение.*

Теперь всё готово, чтобы посмотреть, как это работает. Удалите из таблицы данных (DataTable) все формулы Excel (ячейки В3:D10). Выберите любую ячейку в таблице *DataTable* –> *Данные* –> *Из таблицы/диапазона.* Щелкните правой кнопкой мыши столбец *Values* –> *Удалить другие столбцы*:



Рис. 22.8. Запрос готов к использованию функции *pqVLOOKUP*

Чтобы проверить, работает ли функция PQ VLOOKUP для вас, вы можете попробовать повторить следующую формулу: =VLOOKUP ([Values], BandingLevels, 2, true)

Для этого можно выполнить следующие действия:

*Добавление столбца* –> *Настраиваемый столбец*. Назовите столбец *2,True*. Используйте формулу:

=pqVLOOKUP([Values],BandingLevels,2,true)



Рис. 22.9. Репликация VLOOKUP() с четвертым параметром равным true

Снова перейдите на вкладку *Добавление столбца* –> *Настраиваемый столбец*. Назовите столбец *3,default*. Используйте формулу:

=pqVLOOKUP([Values],BandingLevels,3)



Рис. 22.10. Репликация VLOOKUP() с опущенным четвертым параметром (по умолчанию = true, приблизительное совпадение)

Теперь определите точное совпадение со вторым столбцом таблицы подстановки. *Добавление столбца* –> *Настраиваемый столбец*. Назовите его *2,false*. Используйте формулу:

=pqVLOOKUP([Values],BandingLevels,2,false)



Рис. 22.11 Репликация VLOOKUP() с точным совпадением

Несмотря на то, что вы можете использовать эту функцию для эмуляции точного соответствия VLOOKUP(), лучше этого не делать, а воспользоваться объединением таблиц. Завершите запрос. *Главная* –> *Закрыть и загрузить*.

Вы должны знать об одном незначительном отличии между функцией VLOOKUP() Excel и pqVLOOKUP Power Query: значение #N/A, возвращаемое pqVLOOKUP, на самом деле является текстом, а не значением ошибки. Поскольку истинную ошибку #N/A в Power Query вернуть нельзя.

### Понимание функции pqVLOOKUP

Взгляните на код:

(lookup\_value as any, table\_array as table, col\_index\_number as number,

optional approximate\_match as logical ) as any =>

let

/\*Provide optional match if user didn’t \*/

matchtype =

if approximate\_match = null

then true

else approximate\_match,

/\*Get name of return column \*/

Cols = Table.ColumnNames(table\_array),

ColTable = Table.FromList(Cols, Splitter.SplitByNothing(), null,

null, ExtraValues.Error),

ColName\_match = Record.Field(ColTable{0},"Column1"),

ColName\_return = Record.Field(ColTable{col\_index\_number - 1},

"Column1"),

/\*Find closest match \*/

SortData = Table.Sort(table\_array,

{{ColName\_match, Order.Descending}}),

RenameLookupCol =

Table.RenameColumns(SortData,{{ColName\_match, "Lookup"}}),

RemoveExcess = Table.SelectRows(

RenameLookupCol, each [Lookup] <= lookup\_value),

ClosestMatch=

if Table.IsEmpty(RemoveExcess)=true

then "#N/A"

else Record.Field(RemoveExcess{0},"Lookup"),

/\*What should be returned in case of approximate match? \*/

ClosestReturn=

if Table.IsEmpty(RemoveExcess)=true

then "#N/A"

else Record.Field(RemoveExcess{0},ColName\_return),

/\*Modify result if we need an exact match \*/

Return =

if matchtype=true

then ClosestReturn

else

if lookup\_value = ClosestMatch

then ClosestReturn

else "#N/A"

in Return

Код довольно длинный и сложный, и он использует множество трюков, но основная методология следующая:

1. Втяните таблицу подстановки в Power Query.
2. Отсортируйте ее по убыванию по первому столбцу.
3. Удалите все записи, превышающие искомое значение.
4. Верните значение в запрошенном столбце таблицы данных для первой оставшейся записи, если не указано точное соответствие.
5. Если было указано точное соответствие, проверьте, соответствует ли возврат. Если это так, верните значение. Если это не так, верните #N/A.

Для каждого параметра функции в первой строке кода объявлен тип данных. Это делается для предотвращения проблем, если случайно в таблице подстановки заголовки будут числами.

Переменная approximate\_match определена как необязательная (optional); пользователь может игнорировать ее.

Переменная matchtype проверяет, был ли указан тип соответствия. Если он был указан, именно он присваивается переменной matchtype, если не был указан (approximate\_match равен null), то присваивается значение true.

Имя возвращаемого столбца извлекается путем просмотра заголовков столбцов таблицы, разбиения их на список записей и извлечения записи, индекс которой соответствует запрошенному столбцу (на 1 меньше, так как отсчет начинается с 0).

Данные сортируются в порядке убывания, в зависимости от столбца для поиска. Все записи, превышающие запрошенное значение, удаляются (путем выбора всех строк, где значение меньше или равно искомому значению).

Если строк не осталось, возвращается #N/А, если есть хотя бы одна строка, возвращается первая запись в столбце поиска. Этот результат может быть позже проверен, чтобы увидеть, соответствует ли он искомой записи (что важно для точного соответствия).

Затем вычисляется приблизительное значение соответствия, даже если было запрошено точное соответствие. Если в наборе данных нет строк, сохраняется результат #N/A; в противном случае из возвращаемого столбца извлекается ближайшее значение.

Последний тест проверяет тип запрошенного соответствия. Если это приблизительное совпадение, то возвращается самое близкое совпадение (которое может быть #N/A). Если тип соответствия был точным, код вернет #N/A вместо ближайшего соответствия, если значение столбца подстановки не соответствует точно искомому значению.

### Репликация функции SWITCH() Power Pivot

Power Pivot имеет функцию SWITCH(), которая позволяет выполнять логику с несколькими условиями. Функция ищет указанное значение индекса и возвращает соответствующий результат. Эта функция проще, чем несколько уровней вложенных операторов ЕСЛИ(), поэтому полезно реплицировать функцию SWITCH() в Power Query.

Синтаксис этой функции в PowerPivot выглядит следующим образом:

=SWITCH(expression,value\_1,result\_1,[value\_2,result\_2],…,[Else])

Допустим, вы выставляете счета клиентам на основе кодированного шаблона, где каждый символ что-то значит. Например, в счете MP010450***S***P девятый символ может принимать следующие значения:

E = Employee, S = Yacht Club, N = Non-Taxable, R = Restricted, I = Inactive, L = Social, M = Medical, U = Regular

В Excel вы можете построить формулу с многократными вложениями оператора ЕСЛИ или использовать ВПР(). В Power Pivot, это намного проще с функцией SWITCH():

=SWITCH([Column],"E","Employee","S","Yacht Club", "N","Non-Taxable","R","Restricted","I","Inactive", "L","Social","M","Medical","U","Regular","Undefined")

### Построение функции Power Query SWITCH()

Откройте файл *Emulating SWITCH.xIsx*. *Данные* –> *Получить данные* –> *Из других источников* –> *Пустой запрос*. Назовите запрос – *fnSWITCH*. *Главная* –> *Расширенный редактор*. Введите код:

(input) =>

let

 values = {

 {result\_1, return\_value\_1},

 {input, "Undefined"}

 },

 Result = List.First(List.Select(values, each \_{0}=input)){1}

in

 Result

Ключевые части этого кода:

* *result\_1* – это первая из возможностей, которую вы можете передать в функцию
* *return\_value\_1* – это значение, которое вернет функция, если первое значение = *result\_1*
* Если вам нужно больше значений, вы добавляете запятую после строки {result\_1, return\_value\_1} и вставляете строки {result\_2, return\_value\_2}, {result\_3, return\_value\_3} и т.д.
* Вы можете добавить столько значений, сколько вам нужно
* Если в предоставленном списке нет переданного значения функция вернет текст *Undefined* (это часть Else конструкции SWITCH).

Используя эту структуру, вы можете изменить функцию *fnSWITCH* для нашего сценария:

(input) =>

let

values = {

{"E", "Employee"},

{"S", "SCYC"},

{"N", "Non-Taxable"},

{"R", "Restricted"},

{"I", "Inactive"},

{"L", "Social"},

{"M", "Medical"},

{"U", "Regular"},

{input, "Undefined"}

},

Result = List.First(List.Select(values, each \_{0}=input)){1}

in

Result

Вы не ограничены поиском отдельных символов. Вы можете также искать значения (числа) или более длинные текстовые строки. Просто убедитесь, что ваши параметры всегда вводятся парами между фигурными скобками и имеют запятую в конце строки.

Когда вы закончите вносить изменения в *Расширенном редакторе*, нажмите кнопку *Готово.* *Главная* –> *Закрыть и загрузить*. Теперь вы можете использовать функцию *fnSWITCH* для извлечения типа выставленного счета из каждой записи. Выберите любую ячейку в Таблице на листе *Customers* –> *Данные* –> *Из таблицы/диапазона*. Данные загрузятся в Power Query:



Рис. 22.12. Исходные данные

Для того, чтобы использовать функцию *fnSWITCH* вам сначала нужно извлечь девятый символ из столбца *BillingCode*. *Добавление столбца* –> *Настраиваемый столбец.* Назовите его *Customer Type*. Введите формулу:

=fnSWITCH(Text.Range([BillingCode],8,1))

Помните, что вам нужно начать извлечение текста с символа 8, чтобы получить девятый символ, потому что Power Query начинает отсчет с нуля. Измените имя запроса – *Billing*. Результаты почти идеальны:



Рис. 22.13. Результаты работы функции *fnSWITCH*

Символ *u* не был упомянут в кодах соответствия, и запрос вернул значение *Undefined* (напомню, что Power Query чувствителен к регистру). Если вы хотите чтобы регистр не влиял на результат, вам нужно изменить запрос (но не функцию *fnSWITCH*) и подать на вход функции *Text.Range* текст в верхнем регистре. На листе Excel в области *Запросы и подключения* щелкните правой кнопкой мыши запрос *Billing* –> *Изменить*. Выберите шаг *Добавлен пользовательский объект*. Нажмите значок шестеренки. Измените формулу:

= fnSWITCH(Text.Upper(Text.Range([BillingCode],8,1)))

Нажмите Ok. *Главная* –> *Закрыть и загрузить*:



Рис. 22.14. Тип счета определяется правильно, даже при использовании строчных букв