Язык M Power Query. Временны**е** типы

В M представлено целое семейство временны**х** типов: *date*, *time*, *datetime*, *datetimezone* и duration. У этих типов много общего, поэтому мы рассмотрим их иначе, чем типы text и number. Сначала мы представим каждый тип и рассмотрим его уникальные аспекты. Затем изучим, как различные типы в этом семействе взаимодействуют друг с другом. И наконец расскажем об общей функциональности, свойственной всем типам в семействе.[1]

Предыдущая заметка Следующая заметка

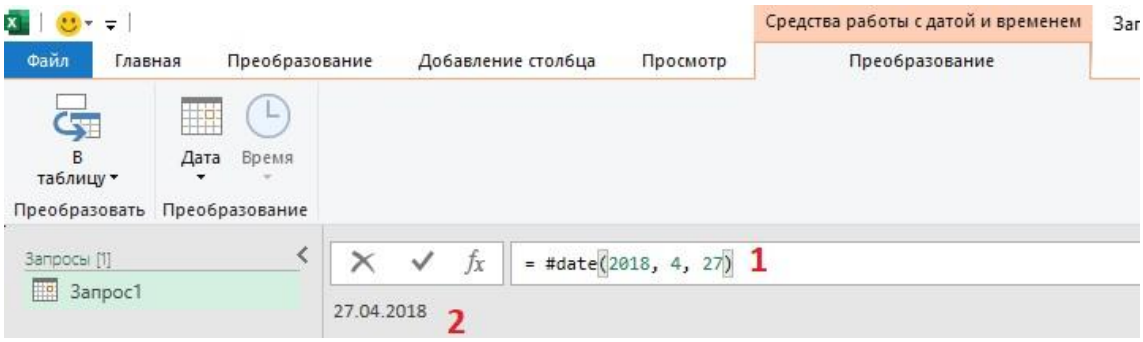

Рис. 1. Тип *date*

Дата

date можно задать, например, так…

Листинг 1[2]

```
= #date(2018, 4, 27) // год, месяц, день
```

Функция…

#date(year as number, month as number, day as number) as date

… требует указанный порядок аргументов, независимо от региональных настроек (см. цифру 1 на рис. 1). При этом на панели предварительного просмотра (см. цифру 2 на рис. 1) и при загрузке в Excel формат даты определяется региональными настройками.

Поддерживаются годы от 1 до 9999. Даты до Рождества Христова не поддерживаются.[3]

Время

Тип *time* используется для значений времени. Поддерживаются доли секунды с точностью до 100 наносекунд, или до семи знаков после запятой. Время можно задать, например, так…

Листинг 2

```
= #time(11, 15, 25)
```

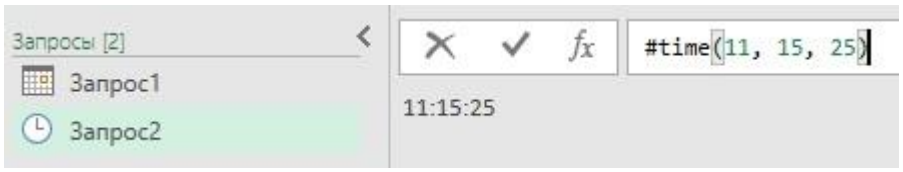

Рис. 2. Тип *time*

Обратите внимание на значки перед именами запросов. Редактор Power Query распознал тип литерала (тип значения) и вывел соответствующую пиктограмму.

Листинг 3

```
= #time(13, 0, 0.53257)
```

---

[1] Заметка написана на основе статьи Ben Gribaudo. Power Query M Primer (Part 8): Types—The Temporal Family. Если вы впервые сталкиваетесь с Power Query, рекомендую начать с Марк Мур. Power Query.

[2] Номер листинга соответствует номеру запроса в приложенном Excel файле.

[3] В этой связи может возникнуть вопрос, когда родился Иисус Христос? 01.01.00 или 01.01.01? Правильный ответ – 01.01.01. С этого дня начинается отсчет возраста Иисуса, а один год ему исполнится 01.01.02. Поэтому можно сказать, что M поддерживает даты с рождения Христа. Подробнее см. здесь.

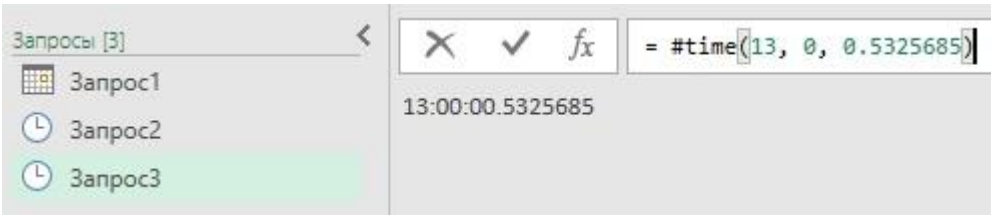

Рис. 3. Время с точностью до 100 наносекунд

Попытка добавить еще одну значащую цифру после запятой не увенчается успехом. Значение на панели предварительного просмотра не изменится.

Функция #time() возвращает ошибку, если не соблюдаются следующие условия:

- 0 ≤ час ≤ 24
- 0 ≤ минута ≤ 59
- 0 ≤ секунда < 60
- часы и минуты могут быть только целыми
- если значение часа равно 24, то минута и секунда должны быть равны 0
- если значение минут равно 60, то значение секунд должно быть равны 0

Обратите внимание, что верхняя граница секунд имеет строгое значение – менее 60, т.е. максимальное значение секунд = 59,9999999. В Excel для времени возможно переполнение. Например, 25-й час трактуется, как 1-й час следующего дня. M не поддерживает переполнение и выдает ошибку: *Expression.Error: Операция Time завершилась неудачей, поскольку результат находится вне диапазона допустимых значений*.

Подробнее о полуночи

И 00:00:00, и 24:00:00 относятся к полуночи. На циферблате часов стрелки будут расположены одинаково. Так что, в каком-то смысле, это одно и то же. Однако, иногда мы хотим различать эти значения. Когда мы используем 24:00 то подразумеваем окончание дня. А использование 00:00 для обозначения полуночи подразумевает начало дня.

Допустим, вы хотите указать, что работали с 10 вечера до полуночи. Вы могли бы сказать, что работали с 22:00 до 24:00. С другой стороны, если бы вы хотели указать, что работали с полуночи до 2 часов ночи, вы бы, сказали с 00:00 до 02:00. В этом смысле значения времени 00:00 и 24:00 различны.

В M и #time(0, 0, 0), и #time(24, 0, 0) относятся к точке на циферблате часов, где час = 0, минута = 0 и секунда = 0. Преобразование в запись вернет одно и тоже значение:

Листинг 4

```
= Time.ToRecord(#time( 0, 0, 0))
```

Листинг 5

```
= Time.ToRecord(#time(24, 0, 0))
```

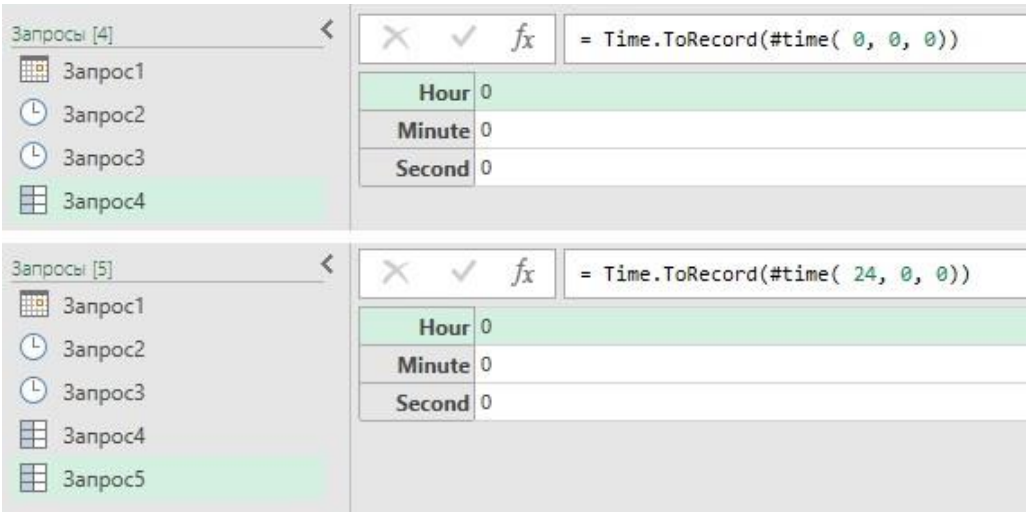

Рис. 4. Время преобразованное в запись

При этом сравнение всё же говорит, что для M это разные значения:

Листинг 6
= #time(0, 0, 0) = #time(24, 0, 0) // false

И преобразование в числа возвращает разные значения:

Листинг 7
= Number.From(#time(0,0,0)) // 0

Листинг 8
= Number.From(#time(24,0,0)) // 1

В M, если вы используете значения 00:00 и 24:00, вам нужно решить, следует ли рассматривать их как эквивалентные или разные. Затем вы можете использовать логику, которая выполнит сравнение в принятой парадигме. Если они должны быть разными, используйте простую логику как в листингах 6-8. Если их следует рассматривать как эквивалентные, вам нужно что-то более сложное, например:

Листинг 9
= Time.ToRecord(#time(0, 0, 0)) = Time.ToRecord(#time(24, 0, 0)) // true (что подтверждается рис. 4)

Дата и время

Объедините типов *date* и *time* дает тип *datetime*:

Листинг 10
= #datetime(2018, 4, 30, 15, 30, 15)

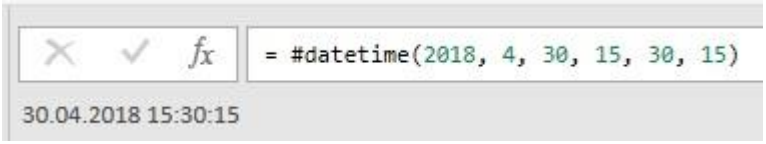

Рис. 5. Тип *datetime*

Временная часть *datetime* отличается по поведению от типа *time*. *time* поддерживает время 24 часа, 0 минут, 0 секунд; *datetime* – нет. Чтобы указать момент времени ровно через 24 часа после начала дня, установите *datetime* на начало следующего дня.

Тип datetimezone

datetimezone использует идею *datetime* и добавляет к ней часовой пояс. Часовой пояс определяется как смещение часов и минут от UTC, а не как текстовое название, что-то типа Центральноевропейское Стандартное Время.

= #datetimezone(2018, 4, 30, 15, 30 ,15, 04, 00)

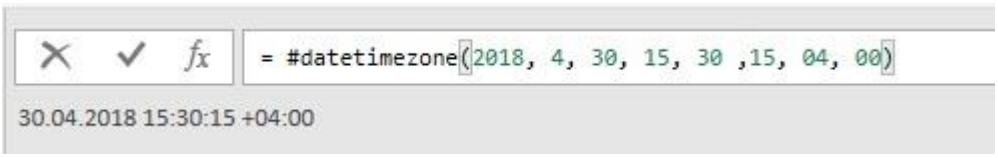

Рис. 6. Тип *datetimezone*

Область определения параметров смещения:

14 ≤ часы смещения + минуты смещения/60 ≤ 14

Любопытно, но здесь PQ поддерживает переполнение.

Рис. 7. Синтаксис *datetimezone* поддерживает переполнение в последнем параметре – минуты смещения

Смещение указано как 4 часа и 65 минут, а отображается как 5 часов 5 минут.

Продолжительность (duration)

Продолжительность представляет собой количество времени. В ней нет конкретной даты или времени. Вместо этого есть количество прошедшего времени или сколько времени осталось. Например, вы можете использовать продолжительность для указания на то, что занятие длилось 2 часа 15 минут:

=#duration(0, 2, 15, 0)

Синтаксис функции

#duration(days as number, hours as number, minutes as number, seconds as number) as duration

Как и *time*, *duration* может сохранять значения с точностью до 100 наносекунд.

В отличие от *time*, *duration* может быть положительной или отрицательной. Функция =#duration поддерживает переполнение по часам, минутам и секундам. В то время как *time* обрабатывает время продолжительностью до одного дня, *duration* может длиться чуть более 10 675 199 дней в положительном или отрицательном направлении. Некоторые аргументы функции =#duration могут быть положительными, а другие отрицательными. Функция их аккуратно суммирует, учитывая знак. Как и в *time* только секунды могут быть не целыми.

Общие черты

Теперь, когда мы познакомились со всей семьей временн**ы**х типов, давайте рассмотрим поведение и черты, которые свойственны более чем одному члену семейства.

Преобразование

Там, где это уместно, один тип может быть преобразован в другие родственные типы:

Листинг 13

= DateTime.From(#date(2019, 12, 31)) // возвращает 00:00:00 указанного дня

Рис. 8. Преобразование типа *date* в тип *datetime*

Преобразование из типа с большим количеством информации (например, *datetime*) в тип, содержащий меньше информации (например, *date* или *time*), приводит к усечению информации.

Листинг 14

= Date.From(#datetime(2000, 10, 15, 13, 25, 12)) // возвращает 15.10.2000

Листинг 15

= Time.From(#datetime(1000, 10, 15, 13, 25, 12)) // возвращает 13:25:12

Такое поведение удобно при работе со столбцом таблицы, содержащим *datetime* (или *datetimezone*), где вас интересует только дата или время. Вы просто преобразуете тип столбца в *date* (или *time*), и посторонняя информация отбрасывается.

При преобразовании в *datetimezone* M предполагает, что входное значение относится к местному часовому поясу.

Листинг 16

= DateTimeZone.From(#date(2019, 12, 31))

Рис. 9. Функция *DateTimeZone.From* добавила часовой пояс региональных настроек ПК

Аналогично, преобразование из типа *datetimezone* в другой тип даты/времени изменяет время с учетом разницы часового пояса в локальных настройках ПК и заданного в параметрах функции:

Листинг 17

= DateTime.From(#datetimezone(2022, 5, 09, 17, 29, 30, 0, 0))

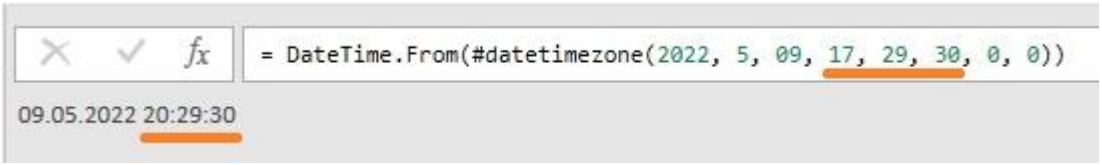

Рис. 10. При извлечении типа *datetime* из типа *datetimezone* происходит изменение времени

Функция увеличила значение времени на три часа. Это разница между значением локального часового пояса и значением часового пояса, указанного в параметрах функции #datetimezone(…, 0, 0).

Арифметика

Арифметические операторы могут использоваться с временн***ы***ми значениями.

Сложение

Если к значению даты/времени добавить значение длительности, получим значение даты/времени того же типа, что и исходное с добавленной продолжительностью:

```
= #date(2018, 6, 1) + #duration(1, 2, 0, 0)
// возвращает 02.06.18; два часа в #duration не повлияли на дату

= #time(13, 5, 25) + #duration(1, 0, 0, 25)
// возвращает 13:05:50; один день в #duration не повлияли на время

= #datetime(2022, 05, 09, 18, 13, 0) + #duration(0, -6, 0, 0)
// возвращает 09.05.2022 12:13:00

= #datetimezone(2022, 05, 09, 6, 13, 0, 3, 0) + #duration(0, -4, 0, 0)
// возвращает 09.05.2022 12:13:00 +03:00
```

Когда вы к времени добавляете продолжительность, представьте себе как стрелки на 24-часовом циферблате вращаются вперед или назад.

Поскольку сложение коммутативно, не имеет значения, стоит ли на первом месте дата/время или продолжительность.

А вот сложить дату и время не получится. Код…

```
= #date(2018, 6, 1) + #time(13, 5, 25)
```

…вернет ошибку *Expression.Error: Не удается применить оператор + к типам Date и Time*.

Неожиданно вместо знака + можно использовать оператор конкатенации. При этом не происходит объединения текстовых строк (поскольку операнды не являются текстом), а изменяется тип литерала. (Напомню литерал – это значение. В отличие от других элементов синтаксиса языка М.) Например:

```
= #date(2018, 4, 30) & #time(15, 30, 10)
// возвращает литерал с типом datetime 30.04.2018 15:30:10
```

Более того, не важен порядок литералов. Следующий код также возвращает верный результат:

```
= #time(15, 30, 10) & #date(2018, 4, 30)
// возвращает 30.04.2018 15:30:10
```

А вот сложить два значения даты/времени с одинаковым типом не получится. При этом не поможет ни знак +, ни оператор конкатенации.

Вычитание

Вычитание работает аналогично сложению. Единственное отличие – на этот раз порядок имеет значение: продолжительность может быть вычтена из даты/времени, но не наоборот.

Если вместо вычитания длительности из даты/времени вы вычитаете другую дату/время, какой тип вы получите? Правильно, длительность! Проверим это:

Листинг 18

```
let
   Subtraction = #date(2018, 8, 10) - #date(2018, 8, 5)
in
   Subtraction is duration // возвращает true
```

```
= #time(12, 0, 0) - #time(14, 0, 0) /* возвращает -2:00:00;
еще одно подтверждение, что возвращается длительность,
time не может быть отрицательным */
```

Листинг 19

```
= #datetimezone(2018, 10, 5, 16, 0, 0, 4, 0) –
   #datetimezone(2018, 10, 5, 15, 0, 0, -4, 0)
/* возвращает -7:00:00; первая функция возвращает 12:00 по Гринвичу,
вторая – 19:00 по Гринвичу, а их разность возвращает длительность -7:00:00 */
```

duration можно сложить с другой *duration*. *duration* можно вычесть из *duration*.

Умножение и деление

В отличие от даты/времени, длительность можно умножать и делить на числа.

```
= #duration(1, 5, 0, 0) * 3 // возвращает 3.15:00:00
```

```
= #duration(1, 5, 0, 0) / 3 // возвращает 0.9:40:00
```

Поскольку умножение является коммутативным, порядок аргументов не имеет значения. Что не верно для деления – порядок аргументов имеет значение. При этом делить числа на длительность в M нельзя.

*Форматирование временны**х** типов как текст*

Преобразование временно**го** значения в текст приводит к получению текста в формате по умолчанию, соответствующего локальным настройкам ПК.

```
= Text.From(#date(2010, 12, 31)) // возвращает 31.12.2010
```

```
= Text.From(#time(16, 12, 31)) /* возвращает 16:12;
 обратите внимание, что секунды не отображаются */
```

```
= Text.From(#datetime(2010, 12, 31, 15, 16, 32))
/* возвращает 31.12.2010  15:16:32;
а здесь секунды отображаются */
```

```
= Text.From(#datetimezone(2010, 12, 31, 15, 16, 32, -4, 15))
// возвращает 31.12.2010  15:16:32 -03:45
```

```
= Text.From(#duration(15, 6, 3, 25.2))
// возвращает 15.06:03:25.2000000
```

Используемые форматы по умолчанию могут отличаться в зависимости от настроек системы. Текст выше предполагает использование культуры ru-RU; если ваша система использует другие локальные настройки, ваши результаты могут отличаться.

Иногда форматы по умолчанию выглядят странно. Например в последнем примере PQ вывел все семь доступных десятичных знаков. Если вы хотите контролировать формат результата, используйте семейство функций *.ToText (Date.ToText, DateTime.ToText и т.п.). В них второй параметр управляет форматом.

```
= Date.ToText(#date(2010, 12, 31), "dddd, dd MMMM yyyy г.")
// возвращает пятница, 31 декабря 2010 г.
```

```
= Time.ToText(#time(12, 15, 18.253), "hh:mm:ss")
// возвращает 12:15:18
```

Любопытные и полезные детали использования форматирования вы найдете в спецификации упомянутого семейства функций *.ToText, например, Date.ToText. Полный список символов, используемых для описания формата, см. Custom date and time format strings. И не забывайте, что

язык M чувствителен к регистру. Так что MMM в описании формата вернет трехбуквенное имя месяца, а mmm – ошибку.

Некоторые текстовые элементы формата даты/времени отображаются по-разному в зависимости от культуры. Например, с региональными настройками en-US MMM вернет Jan, Feb, Mar, ..., а с ru-RU – янв, фев, мар, ... По умолчанию результаты отображаются в текущей культуре локальной системы. Вы можете переопределить это, указав идентификатор культуры в качестве третьего аргумента семейства функций *.ToText. Например…

Листинг 20

```
= Date.ToText(#date(2010, 12, 31), "m", "ja-JP")
```

… вернет …

Рис. 11. Дата в японской культуре

Вы можете использовать третий аргумент функции Date.ToText – *Culture*, чтобы найти, например, какой год в тайской культуре соответствует дате 01.01.2022:

Листинг 21

```
= Date.ToText(#date(2022, 01, 01), "y", "th-TH") // возвращает…
```

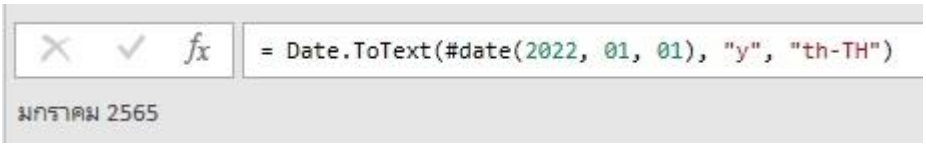

Рис. 12. Какой год в Таиланде соответствует 1 янв 2022

Excel и язык M используют коды культур в соответствии с ISO Language Code Table.

В функции Duration.ToText нет аргумента с идентификатором культуры. И это логично, так как *duration* не зависит от культуры.

Одним из способов обеспечить согласованное отображение значений в разных системах является указание используемой эталонной культуры, как мы это делали выше. Другой вариант - использовать строку формата, которая не зависит от культуры. Строка формата "o" является одной из таких строк. Её использование приводит к тому, что строка даты и времени выводится в формате, который остается неизменным независимо от текущей конфигурации системы. Например…

Листинг 22

```
= DateTime.ToText(#datetime(2018, 12, 25, 11, 50, 20), "o")
```

… вернет…

Рис. 13. Использование строки "o" в аргументе формата подавляет локальные настройки

Значения OLE Automation Date (OADate)

Все временны**е** типы M могут быть созданы из значений OLE Automation Date (OADate). Excel также использует этот формат для кодирования даты и времени.

OADate использует одно число, чтобы указать, на сколько дней, включая доли дня, дата отстоит от контрольной (вперед или назад). Значение OADate=2,5 представляет момент времени через два с половиной дня после контрольной точки, в то время как -15,75 указывает на пятнадцать и три четверти дня до контрольной точки.

Для *date*, *datetime* и *datetimezone* точкой отсчета является начало дня 30 декабря 1899 года:

= Date.From(2.75) // возвращает 01.01.1900

= DateTime.From(2.75) // возвращает 01.01.1900 18:00:00

Date.From игнорирует дробную часть OADate – по сути, она усекается. Поскольку дата ничего не знает о времени, это имеет смысл.

DateTimeZone.From значения OADate интерпретирует как относящиеся к текущему часовому поясу системы (который на моем ПК имеет смещение +3 часа):

= DateTimeZone.From(2.75) // возвращает 01.01.1900 18:00:00 +03:00

Для *Time.From* контрольная точка соответствует 00:00. Для *Time.From* допустимые значения параметра: 0 ≤ OADate < 1

= Time.From(0.75) // возвращает 18:00:00

Поскольку OADate, равный 1, недопустим, невозможно создать эквивалент #time(24, 0, 0) с помощью OADate.

Для *Duration.From* контрольная точка = 0. Допускаются положительные и отрицательные значения.

= Duration.From(0.75) // возвращает 0.18:00:00

Все временные типы могут быть преобразованы в значение OADate с помощью функции Number.From.

= Number.From(#date(2015, 12, 25)) // 42363

= Number.From(#time(11, 32, 18.5)) // 0.4807696759259259

= Number.From(#datetime(2015, 12, 25, 11, 32, 18.5)) // 42363.480769675924

= Number.From(#datetimezone(2015, 12, 25, 11, 32, 18.5, -4, 0)) // 42363.397436342595

= Number.From(#duration(35, 10, 15, 25.2)) // 35.427375

Как мы видели ранее, когда тип *datetimezone* преобразуется в значение, которое не содержит смещение часового пояса, выводимое значение корректируется относительно часового пояса локальной системы.

В следующий заметке мы обсудим оставшиеся примитивные типы языка M: *logical*, *null* и *binary*.