Язык M Power Query. Синтаксис таблиц

Таблицы – основа Power Query. Вы пишите запросы, чтобы подсоединиться к одному или нескольким источникам, извлечь, объединить и обработать данные, а затем вернуть результаты в виде таблицы в Excel (или Power BI). В связи с таблицами можно изучать много интересного, но мы сосредоточимся на синтаксисе. Предыдущий пост мы завершили мыслью о том, что по поведению таблицы похожи на списки и записи. При этом таблицы больше, чем просто *список записей*. Начнем рассмотрение с того, что общего в списках и таблицах.[1]

Предыдущая заметка Следующая заметка

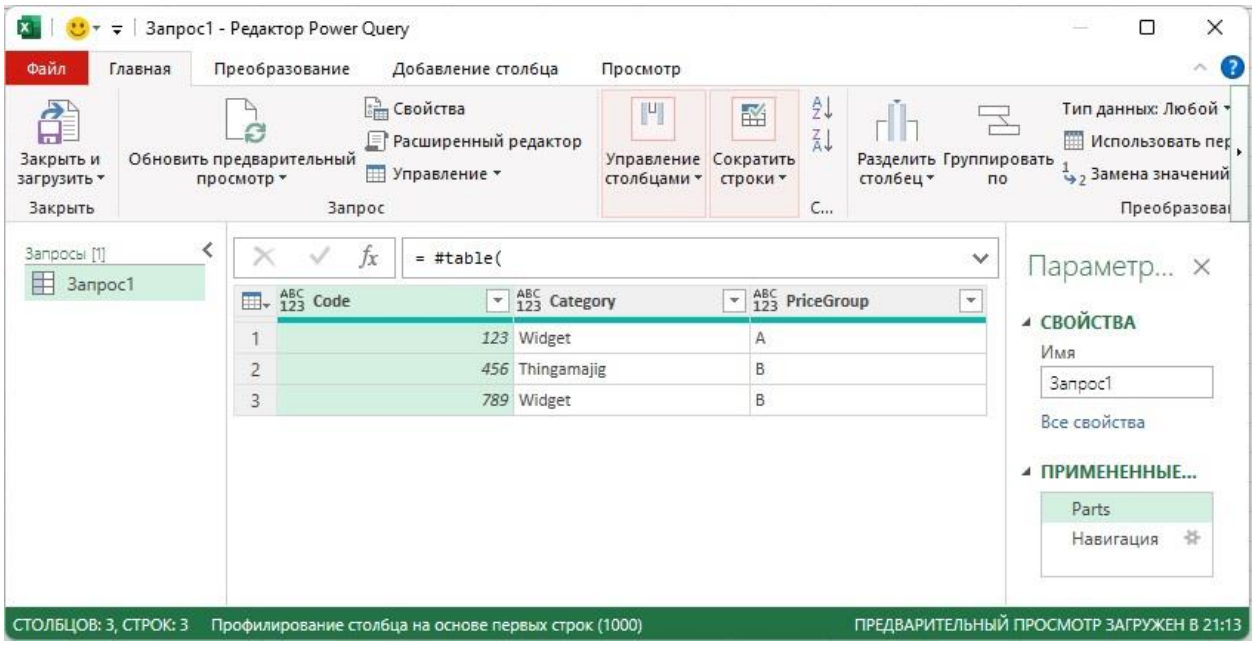

Рис. 1. Таблица в редакторе Power Query

Позиционный выбор

В списках доступ к элементу можно получить с помощью оператора *позиционного индекса.* Например, myList{2} возвращает третий элемент списка *myList* (напомню: индексы в M начинают отсчет с нуля). Таблица также поддерживает оператор индекса, но здесь индекс определяет, какую строку возвращать. Строка возвращается в виде записи, при этом каждый столбец представлен в виде поля в записи.

Листинг 1[2]

```
let
  Parts = #table(
    { "Code", "Category", "PriceGroup" },
    {
      { 123, "Widget", "A" },
      { 456, "Thingamajig", "B" },
      { 789, "Widget", "B" }
    }
  ),
  Result = Parts{1}
in
  Result /* возвращает вторую строку в виде записи:
[Code = 456, Category = "Thingamajig", PriceGroup = "B"] */
```

Библиотечная функция #table создает таблицу (см. рис. 1). Первый аргумент функции #table – список имен столбцов.

---

[1] Заметка написана на основе статьи Ben Gribaudo. Power Query M Primer (Part 11): Tables—Syntax. Если вы впервые сталкиваетесь с Power Query, рекомендую начать с Марк Мур. Power Query.

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

```
  { "Code", "Category", "PriceGroup" },
```

Второй аргумент – список списков, который определяет значения строк: каждый элемент во внешнем списке соответствует строке; каждый элемент во внутреннем списке соответствует значению столбца для этой строки.)

```
  {
   { 123, "Widget", "A" },
   { 456, "Thingamajig", "B" },
   { 789, "Widget", "B" }
  }
```

Переменная (идентификатор) *Result* возвращает вторую строку таблицы в виде *записи* (индекс 1).

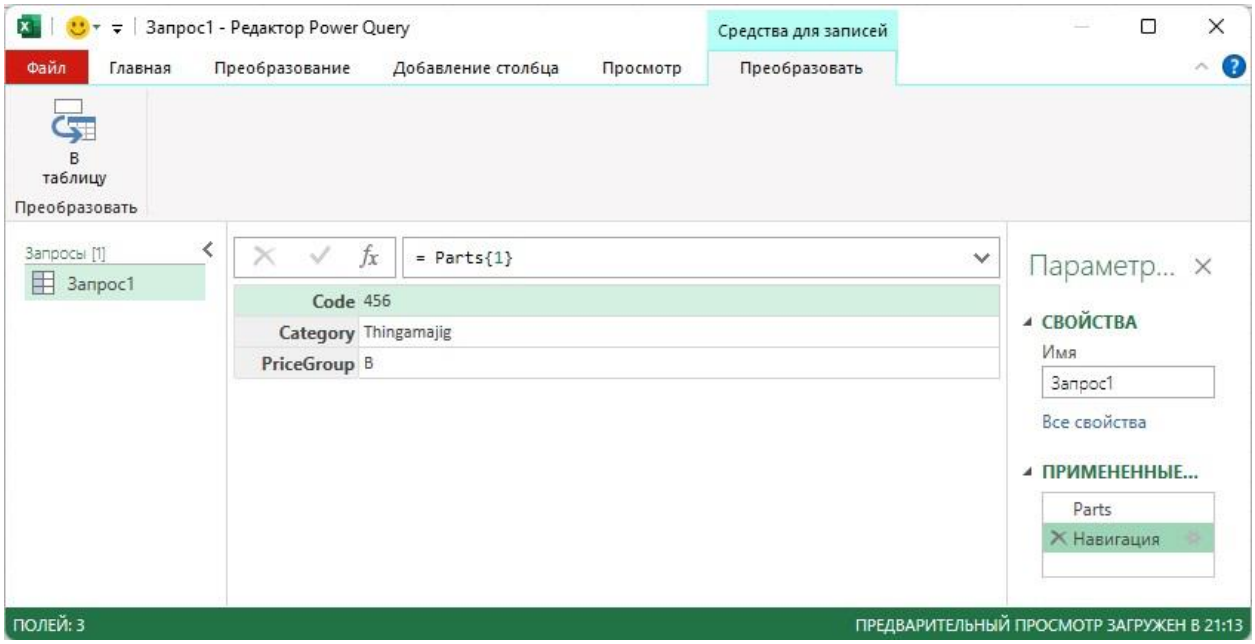

Рис. 2. Результат – вторая строка таблицы в виде записи

Если запрошенный индекс не существует, возвращается ошибка. Если дополнить запрос знаком ?, то вернется *null*.

```
Parts{3} /* Expression.Error: Элементов в перечислении было
недостаточно для выполнения операции. */

Parts{3}? // null
```

Стандартная библиотека содержит множество функций, которые работают со строками таблицы на основе позиции: Table.SingleRow, Table.First, Table.Last, Table.Skip, Table.FirstN, Table.LastN, Table.Range, Table.PositionOf, Table.ReverseRows.

Выбор на основе значений

В таблицах также можно выбрать одну строку на основе значения поля. Для этого внутри фигурных скобок нужно передать запись, определяющую критерии поиска. Каждое имя поля должно соответствовать столбцу таблицы, а каждое значение поля должно соответствовать значению для поиска в этом конкретном столбце. Запись может содержать ссылку не на все столбцы, а только на те, которые используются для поиска.

```
Parts{[Code = 123]} // возвращает [Code = 123, Category = "Widget", PriceGroup = "A"]

Parts{[Category = "Thingamajig"]} // [Code = 456, Category = "Thingamajig", PriceGroup = "B"]

Parts{[Category = "Widget", PriceGroup = "B"]} // [Code = 789, Category = "Widget", PriceGroup = "B"]
```

Оператор выбора возвращает результат в виде записи. Оператор выбора возвращает не более одной строки. Если критерии поиска совпадают более чем с одной строкой, возвращается сообщение об ошибке. Использование необязательного выбора (путем добавления

вопросительного знака) не устраняет ошибку. Необязательный выбор влияет на поведение только тогда, когда совпадение не найдено, а не когда найдено несколько совпадений.

```
Parts{[Category = "Other"]} /* Expression.Error: Ключу
не соответствует ни одна строка в таблице. */

Parts{[Category = "Other" ]}? // null

Parts{[Category = "Widget"]} /* Expression.Error: Ключ
соответствовал более чем одной строке в таблице. */

Parts{[Category = "Widget"]}? /* Expression.Error: Ключ
соответствовал более чем одной строке в таблице. */
```

Поиск выполняется по *значениям* записи критериев. Если вы хотите извлекать строки таблицы динамически, вам потребуется более сложно выражение. Что-то типа:

```
Table.Single Row(Table.SelectRows(Parts, each ...some expression...))
```

Доступ к столбцу (он же Выбор поля)

Как и *record*, тип *table* поддерживает выбор поля. В *записи* имя извлекаемого поля помещается в квадратные скобки. Например: MyRecord[First Name]. В результате возвращается значение этого поля. Когда выбор поля выполняется в таблице, имя интересующего столбца помещается в квадратные скобки (например: myTable[Some Column]), но, поскольку столбец содержит множество значений, возвращается *список*.

Возвращаемый список содержит значение из каждой строки для указанного столбца в порядке, в котором эти значения отображаются в этом столбце (таким образом, первый элемент в списке будет соответствовать значению столбца из первой строки, второй элемент – значению из второй строки и т.д.). Продолжая пример на основе листинга 1, код…

```
  Result = Parts[Category]
```

…вернет список { "Widget", "Thingamajig", "Widget" }

Возвращая список, M позволяет вам работать с набором значений в столбце, используя функции для списков. Вот почему в библиотеке M нет отдельного набора табличных функций для операций с одним столбцом. Например, следующий код…

Листинг 2

```
let
  Parts = #table(
    { "Code", "Category", "PriceGroup" },
    {
      { 123, "Widget", "A" },
      { 456, "Thingamajig", "B" },
      { 789, "Widget", "B" }
    }
  ),
  Result = List.Distinct(Parts[PriceGroup])
in
  Result
```

… вернет уникальные значения столбца [PriceGroup]

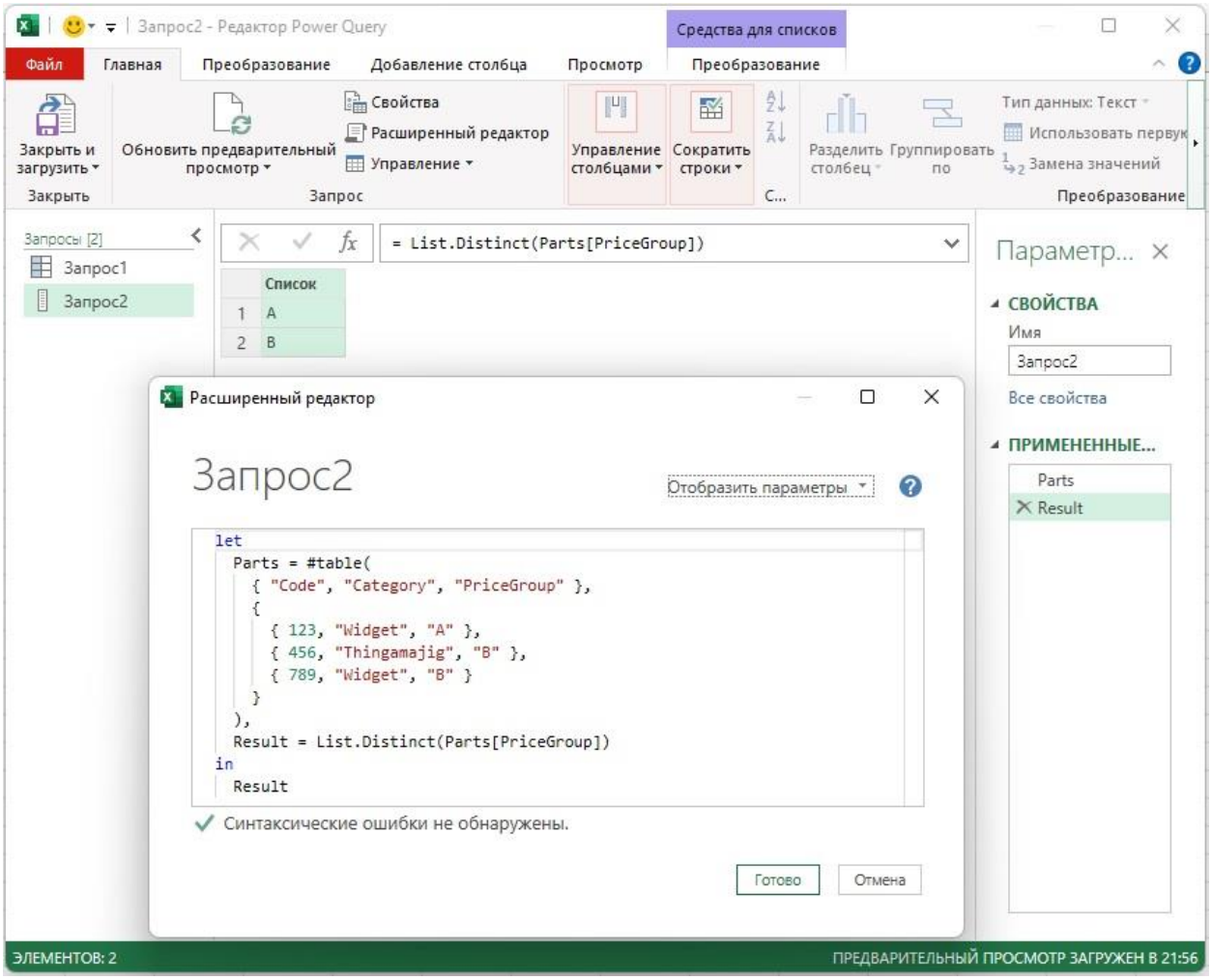

Рис. 3. Список уникальных значений столбца [PriceGroup]

Проекция

Как и запись, таблица поддерживает проекцию. В случае таблиц проекция создает новую таблицу, содержащую уменьшенный набор столбцов. Оператор проекции – имена столбцов в квадратных скобках.

Листинг 3

```
let
  Parts = #table(
    { "Code", "Category", "PriceGroup" },
    {
      { 123, "Widget", "A" },
      { 456, "Thingamajig", "B" },
      { 789, "Widget", "B" }
    }
  ),
  Result = Parts[[Code],[Category]]
in
  Result
```

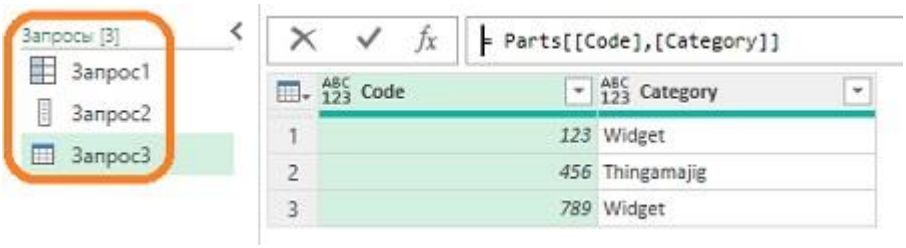

Рис. 4. *Проекция* таблицы

Обратите внимание, как редактор Power Query распознает тип запроса, и подставляет правильную пиктограмму рядом с именами запросов: Запрос1 возвращает *запись*, Запрос2 – *список*, Запрос3 – *таблицу*.

Если столбец, указанный в проекции, не существует, возвращается ошибка.

= Parts[[Code],[Cat]] // Expression.Error: Столбец "Cat" таблицы не найден.

Если дополнить проекцию необязательным выбором (вопросительным знаков за квадратными скобками), несуществующие столбцы будут включены в новую таблицу со значениями, равными *null*.

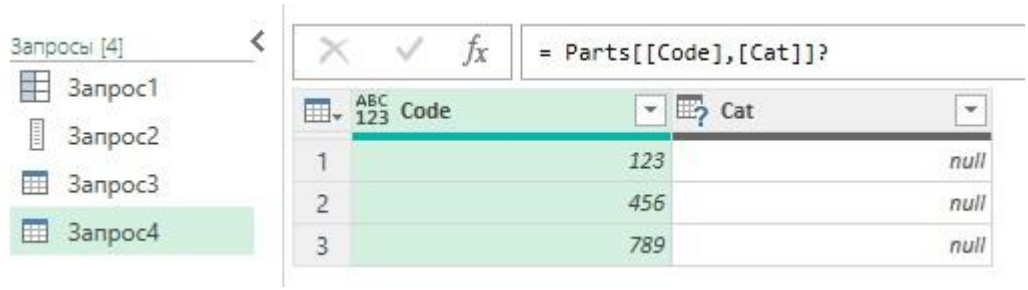

Рис. 5. В проекцию включен несуществующий столбец

В определении таблицы, извлечении столбца или создании проекции имена столбцов должны быть жестко заданы. Если вам требуется динамический доступ к столбцам таблицы, используйте функцию Table.SelectColumns. Её второй аргумент может быть текстовым литералом (именем столбца в кавычках), списком (с именами столбцов), ссылкой (на имя столбца), иным выражением, возвращающим имя столбца (имена столбцов).

Листинг 5

```
let
   MyTable = Table.FromRecords(
      {
         [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
         [CustomerID = 2, Name = "Jim", Phone = "987-6543"],
         [CustomerID = 3, Name = "Paul", Phone = "543-7890"],
         [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
      }
   ),
   a = 1,
   Result = Table.SelectColumns(MyTable, if a = 1 then "Name" else "Phone")
in
   Result // возвращает значения столбца Name в виде списка
```

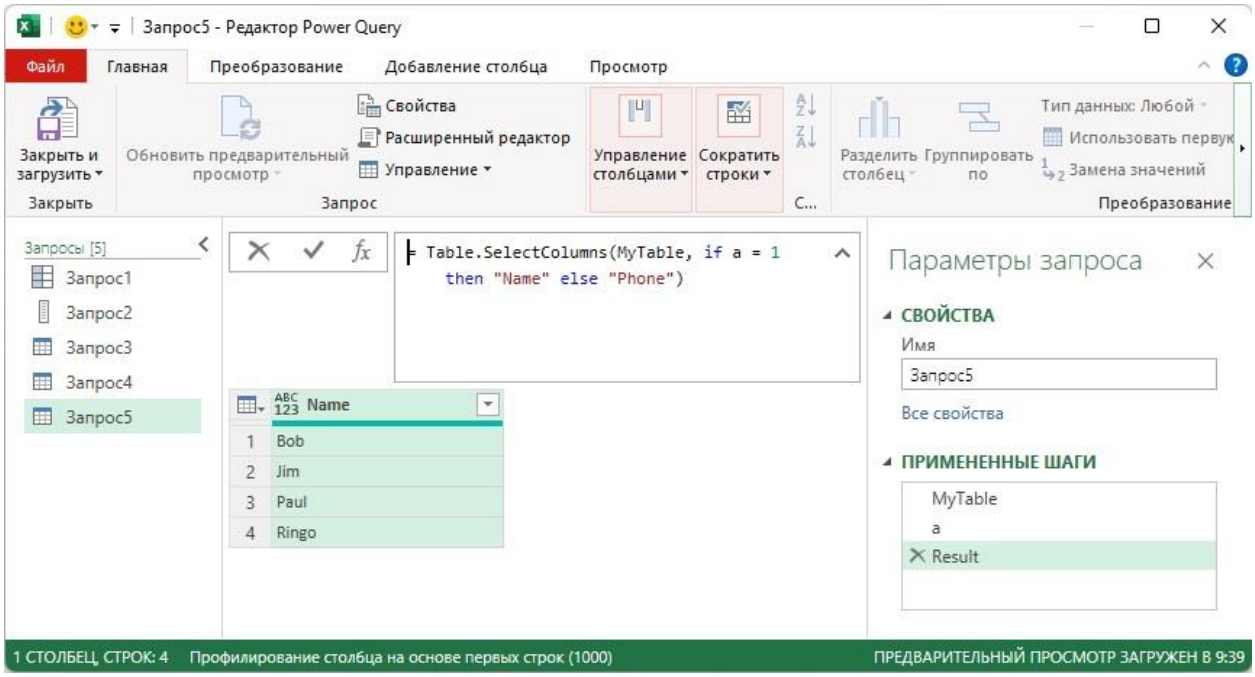

Рис. 6. Работа функции *Table.SelectColumns()*

Сравнение

Если две таблицы содержат одинаковое количество столбцов с одинаковыми именами и одинаковое количество строк, где одноименные столбцы из каждой таблицы имеют одинаковые значения в каждой позиции, оператор равенства (=) считает таблицы равными. В противном случае они не равны (<>).

```
#table({"Col1"}, {{1}, {2}}) = #table({"Col1"}, {{1}, {2}}) // true
```

```
#table({"Col1"}, {{1}, {2}}) <> #table({"Col1"}, {{1}, {2}}) // false
```

```
#table({"Col1", "Col2"}, {{"A", "B"}, {1,2}, {3,4}}) =
   #table({"Col2", "Col1"}, {{"B", "A"}, {2,1}, {4,3}}) /* true,
   несмотря на то, что столбцы находятся в разных позициях,
   они содержат одни и те же значения в одних и тех же
   строках, поэтому таблицы эквивалентны */
```

```
#table({"Col1"}, {{1}, {2}}) = #table({"Col1"}, {{2}, {1}}) /* false,
   несмотря на то, что таблицы содержат одни и те же строки,
   они упорядочены по-разному, поэтому таблицы не равны */
```

Схема столбцов и метаданные не обязаны совпадать, чтобы таблицы считались равными.

Листинг 6

```
let
  Table1 = Table.TransformColumnTypes(#table({"Col1"}, {{1}}),{{"Col1", type any}}),
  Table2 = Table.TransformColumnTypes(#table({"Col1"}, {{1}}),{{"Col1", type number}})
in
   Table1 = Table2 // возвращает true, при том, что типы столбцов различаются
```

Думайте об этом так: M считает две таблицы равными, если они содержат одни и те же данные. Порядок столбцов, сведения о схеме и метаданные, являются вспомогательными элементами, а не данными, поэтому не влияют на сравнение. Например, вы можете получить одни и те же данные из таблицы независимо от того, как расположены ее столбцы.

Объединение таблиц

Две таблицы можно объединить с помощью оператора конкатенации (&).

Листинг 7

```
= #table({"City", "State"}, {{"Chicago", "IL"}}) & #table({"City", "State"}, {{"Washington", "DC"}})
```

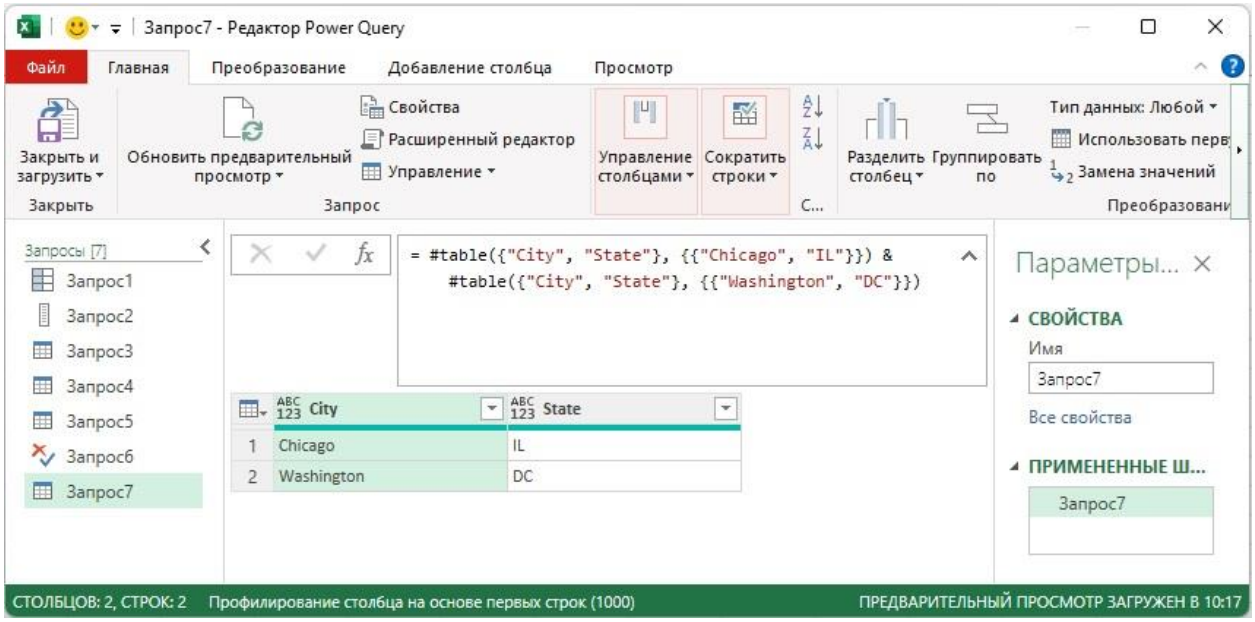

Рис. 7. Объединение таблиц

Столбцы разных таблиц объединяются по имени, а не по позиции. Если столбец существует только в одной из таблиц, в объединенной значения в этом столбце будут *null*.

Листинг 8

```
= #table({"City"}, {{"Chicago"}}) & #table({"State"}, {{"DC"}})
```

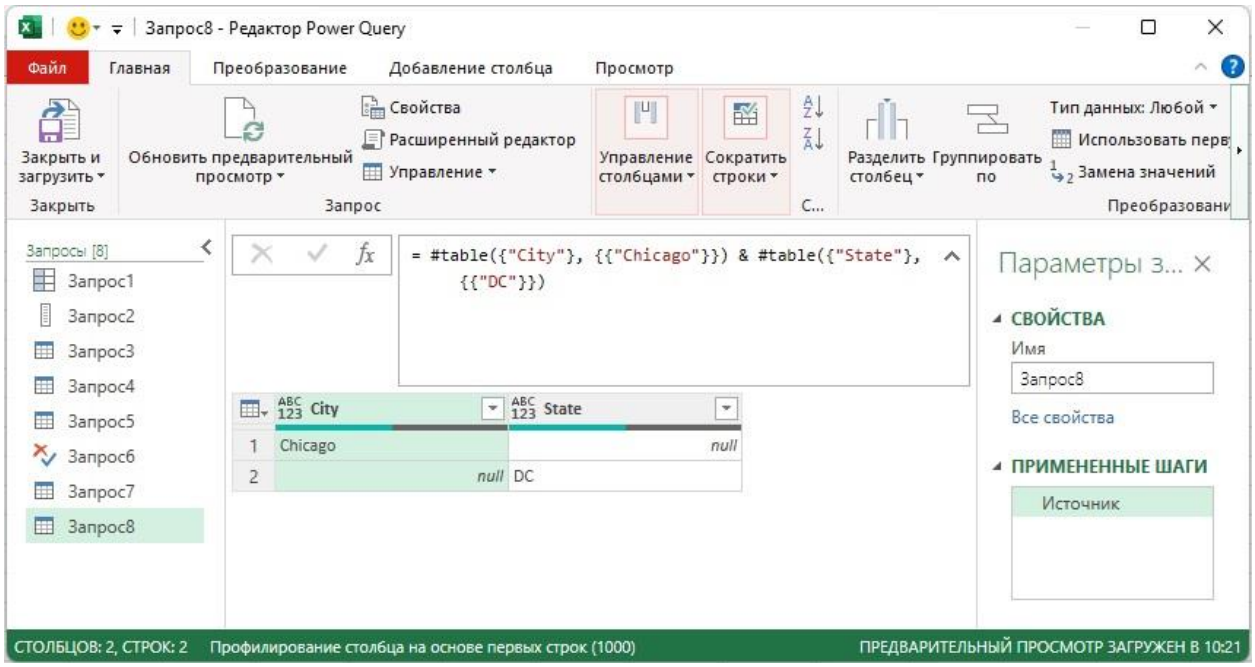

Рис. 8. Значения null при объединении

В возвращаемой таблице все столбцы из первой таблицы выводятся первыми, располагаясь в соответствии с порядком, в котором они отображаются в исходной таблице. Затем выводятся несовпадающие столбцы из второй таблицы. Также с сохранением порядка

М не требует, чтобы типы данных столбцов были совместимы.

Листинг 9

```
= #table({"Age"}, {{18}}) & #table({"Age"}, {{"21"}})
```

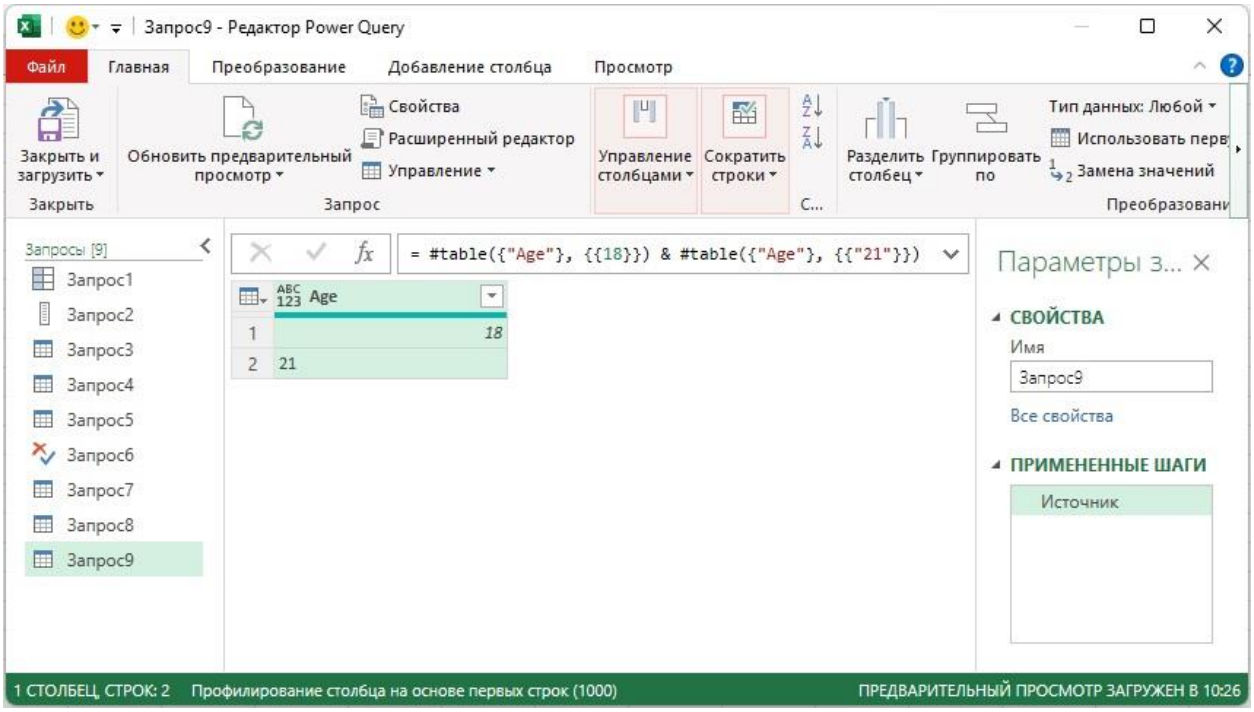

Рис. 9. При объединении типы данных в одном столбце могут не совпадать

Последние три черты поведения отличаются того, как работает SQL UNION ALL operator. В SQL для объединения двух таблиц требуется, чтобы таблицы содержали одинаковое количество столбцов, столбцы объединяются на основе позиции (а не имени), а объединяемые столбцы должны иметь совместимые типы данных. С помощью Power Query таблицы с разным количеством столбцов могут быть объединены вместе, столбцы объединяются по имени (а не по позиции), а типы данных столбцов не имеют значения для целей объединения.

В следующей заметке

Мы рассмотрели синтаксис. В следующей заметке мы изучим среду, в которой этот синтаксис выполняется. Т.е, как Power Query обрабатывает таблицы. Потоковая передача, свертывание запросов, буферизация, ключи, встроенное кэширование, уровень защиты данных, брандмауэр ... все это может оказать серьезное влияние на производительность и способ обработки таблиц в M.