第 5 章

迭 代 函 数

DAX 中，所有带后缀 X 的聚合函数都是迭代器函数，例如 COUNTX()、MAXX()、SUMX()、AVERAGEX()等。大多数带 X 后缀的聚合类迭代器函数的语法相似，大多为(< table >,< expression >)结构，第 1 个参数为表，第 2 个参数为表达式。与 Excel 通用的聚合函数相比，迭代类聚合函数是行上下文函数，它们能够识别当前行并实现逐行运算，最终聚合成一个值，迭代函数多用于度量值中。

5.1 常见迭代聚合

聚合迭代函数的运算一般分以下三步来完成：

(1) 对第 1 个参数的表进行运算(例如引用表、筛选表；选择列、新增列等)。

(2) 依据行上下文条件，对运算后的参数表进行逐行迭代运算。

(3) 对运算结果进行聚合。

5.1.1 COUNTX()

COUNTX()对表表达式的运算值进行条件计数(空值不计算)，语法如下：

```
COUNTX(< table >,< expression >)
```

在 DAX 中，类似的函数有 COUNTAX()。

对比 FILTER()迭代函数及 CALCULATETABLE()筛选器函数在计算列中的差别。创建两个计算列，表达式如下：

```
库.B 吨行数 F: = COUNTX(FILTER(DK,DK[等级] = "B"),DK[吨数])

库.B 吨行数 C: = COUNTX(CALCULATETABLE(DK,DK[等级] = "B"),DK[吨数])
```

为了方便比较，隐藏了表中的其他列，返回的值如图 5-1 所示。

FILTER()是迭代函数，是行上下文函数；聚合函数在计算列中会忽略行上下文，而

	吨数	等级	库.B吨行数F	库.B吨行数C
1	21.27	A	2	
2	20.89	B	2	1
3	22.89	A	2	
4	18.86	A	2	
5		B	2	
6	21.22	C	2	
7	20.89	B	2	1

图 5-1　COUNTX()函数的应用(1)

CALCULATETABLE()函数与 CALCULATE()函数一样,具备上下文转换能力。唯一的区别是 CALCULATE()返回的是标量值而 CALCULATETABLE()返回的是表。

创建计算列库.月行数 F,表达式如下:

```
库.月行数 F: = COUNTX(
FILTER(
                DK,
                RELATED(DT[月]) = 9),
          DK[包装方式]
)
```

为了方便理解,隐藏了表中的其他列,返回的值如图 5-2 所示。

	包装方式	库.月行数F
1	散装	4
2	箱装	4
3	桶装	4
4	捆	4
5	桶装	4
6	箱装	4
7	扎	4

图 5-2　COUNTX()函数的应用(2)

创建计算列库.月行数 C,表达式如下:

```
库.月行数 C: = COUNTX(
CALCULATETABLE(
                DK,
                RELATED(DT[月]) = 9),
          DK[包装方式]
)
```

返回的值如图 5-3 所示。

在 DAX 中,CALCULATETABLE()与 CALCULATE()是仅有的两个能够创建筛选上下文的函数。在筛选上下文中,关系是自动传递的,不需要额外使用 RELATED()或 RELATEDTABLE()手动传递,且在计算列中涉及聚合函数的运算时,CALCULATETABLE()与

图 5-3 COUNTX()函数的应用(3)

CALCULATE()会将聚合函数的筛选上下文转换为计算列中的行上下文。

对上面的表达式进行修改,修改后的表达式如下:

```
库.月行数C:=COUNTX(
CALCULATETABLE(DK,DT[月]=9),
DK[包装方式]
)
```

返回的值如图 5-4 所示。

	包装方式	库.月行数F	库.月行数C
1	散装	4	1
2	箱装	4	1
3	桶装	4	
4	捆	4	1
5	桶装	4	1
6	箱装	4	
7	扎	4	

图 5-4 COUNTX()函数的应用(4)

5.1.2 MAXX()

MAXX()对表表达式的运算值按条件获取最大值(跳过空值),语法如下:

```
MAXX(<table>,<expression>)
```

在 DAX 中,类似的函数为 MINX(),MINX()函数对表表达式的运算值按条件获取最小值(跳过空值)。MAXX()和 MINX()只计算数字、文本、日期,不支持 TRUE/FALSE 值。

创建计算列库.乘积最大和库.各行乘积,表达式如下:

```
库.乘积最大:=MAXX('DK','DK'[入库]*'DK'[出库])

库.各行乘积:=CALCULATE(MAXX('DK','DK'[入库]*'DK'[出库]))
```

为了方便理解，隐藏了表中的其他列并对返回值进行排序，返回的值如图 5-5 所示。

	入库	出库	库.乘积最大	库.各行乘积
1	32	12	3906	384
2	55	23	3906	1265
3	65	32	3906	2080
4	76	35	3906	2660
5	78	36	3906	2808
6	87	33	3906	2871
7	93	42	3906	3906

图 5-5　MAXX()函数的应用

出于 ABC 分类管理的需要，以合同表中的产品及订单表中的数量为依据，设置 A 类产品的起点值(总订单数的前 30%中的最小值，并且数量值大于 8)。在合同表中创建计算列合.A 类值，表达式如下：

```
合.A类值 := 
VAR A =
    MINX (
        TOPN (
            CALCULATE ( COUNTROWS ( '订单' ) ) * 0.3,
            RELATEDTABLE ( '订单' ),
            '订单'[数量], DESC
        ),
        '订单'[数量]
    )
RETURN
    IF ( A > 8, A, BLANK () )
```

为了方便理解，隐藏了表中的其他列并对返回值进行排序，返回的值(仅截取非空值)如图 5-6 所示。

	产品	合.A类值
1	木材	16
2	油漆	11
3	钢化膜	9
4	异型件	9

图 5-6　MINX()函数的应用

5.1.3　AVERAGEX()

平均数也称为均值，用于反映一组数据中心点所在的位置。AVERAGEX()用于对表表达式的运算值求算术平均值(空值不计算)，语法如下：

```
AVERAGEX(<table>,<expression>)
```

在 DAX 中，类似的函数有 MEDIANX()中位数、PERCENTILEX. EXC()分位数(不含 0、1)、PERCENTILE. INC ()分位数(含 0、1)、MAXX()最大值、MINX()最小值。中位数、分位数是指一组数据按升序或降序排列后，所排在中间位置的值(中位数，50%位置)或某百分位位置的值(如 25%位置、75%位置)。DAX 中没有众数对应的函数。

以上聚合方式的区别与联系说明：平均数 AVERAGE()/AVERAGEX()是全部数据的算术平均值，中位数 MEDIAN()/MEDIANX()是处于一组数据中中间位置的值，如果数据是对称的，则平均值与中位数相等。如果数据中存在极小值 MIN()/MINX()，则数据呈左偏态，中位数将大于平均值；如果数据中存在极大值 MAX()/MAXX()，则数据呈右偏态，中位数将小于平均值。中位数不受极值(极大值、极小值)影响，它是一组数据中间位置的值；当数据呈左偏态或右偏态时，中位数的稳健性高于平均数，此种情况宜采用中位数进行分析。

在 DAX 中，可采用多种表达式灵活地实现同一需求。考虑到代码的优雅与易维护性，应尽量采用简洁、实用的表达式。以实现各类包装的平均入库为例，创建度量值 M.均入库1、M.均入库 2、M.均入库 3，各表达式如下：

```
M.均入库 1 : =
ROUND (
    AVERAGEX (
        SUMMARIZE ( 'DK', 'DK'[包装方式], 'DK'[入库] ),
        'DK'[入库]
    ),
    2
)
```

或

```
M.均入库 2 : =
ROUND (
    AVERAGEX (
        ADDCOLUMNS (
            SELECTCOLUMNS (
                DK,
                "日期", [日期], "产品", [产品], "入库", [入库]
            ),
            "月", MONTH ( [日期] )
        ),
        [入库]
    ),
    2
)
```

或

```
M.均入库 3 : =
VAR A =
    ADDCOLUMNS (
        SELECTCOLUMNS (
```

```
            DK,
            "日期", [日期], "产品", [产品], "入库", [入库]
        ),
        "月", MONTH ( [日期] )
    )
RETURN
    ROUND ( AVERAGEX ( A, [入库] ), 2 )
```

创建透视表中，将包装方式拖入行标签，勾选度量值 M. 均入库 1、M. 均入库 2、M. 均入库 3，返回的值如图 5-7 所示。

行标签	M.均入库1	M.均入库2	M.均入库3
捆	87	87	87
散装	76	76	76
桶装	74	74	74
箱装	55	55	55
扎	65	65	65
总计	**69.43**	**69.43**	**69.43**

图 5-7　AVERAGEX()函数的应用

5.1.4　PERCENTILEX.EXC()

PERCENTILEX.EXC()用于返回针对表中的每行计算的表达式的百分数。若要返回列中数字的百分数，应使用 PERCENTILE.EXC()函数，语法如下：

```
PERCENTILEX.EXC(<table>, <expression>, k)
```

在 DAX 中，类似的函数有 PERCENTILEX.INC ()，此函数用于返回针对表中的每行计算的表达式的百分数。函数 PERCENTILE.EXC()与 PERCENTILE.INC()中，EXC 是 exclude(不包含)单词的简写，排除 k 值为 0 和 1 的情况；INC 是 include(包含)单词的简写，包含 k 值为 0 和 1 的情况。

当数据呈正态分布(对称分布)时：68%的数据处于±1Ω(标准差)的范围内，95%的数据处于±2Ω 的范围内，99%的数据处于±3Ω 的范围内。如果准备剔除±3Ω 之外的数据，则 PERCENTILE.INC()/PERCENTILEX.INC()、PERCENTILE.EXC()/PERCENTILEX.EXC()这几个函数就是很好的选择。

当数据呈左偏或右偏不对称分布时，可以采用切比雪夫不等式进行相关数据的筛选或剔除，该不等式的内容如下：

(1) 至少有 75%的数据在平均数±2Ω 范围内。

(2) 至少有 89%的数据在平均数±3Ω 范围内。

(3) 至少有 94%的数据在平均数±4Ω 范围内。

至于采用单侧还是双侧拒绝域则需结合数据分布的实际情况进行判断。

创建一个查询表达式，取 DK 表中位于 15%和 85%分位的入库数据，表达式如下：

```
EVALUATE
VAR A =
    PERCENTILEX.EXC ( DK, DK[入库], 0.15 )
VAR B =
    PERCENTILEX.EXC ( DK, DK[入库], 0.85 )
RETURN
    FILTER ( DK, DK[入库] >= A && DK[入库] <= B )
```

返回的值及相关说明如图 5-8 所示。

日期	产品	包装方式	入库	出库	方数	吨数	等级	说明
2020/9/2	钢化膜	散装	76	35	56	18.86	A	
2020/11/2	油漆	桶装	55	23	72	20.89	B	
2021/9/29		捆	87	33		21.22	C	筛选后返回的值
2021/10/11	苹果醋	箱装	78	36	65		B	
2021/11/14	包装绳	扎	65	32	69	22.89	A	
2020/9/27	蛋糕纸	箱装	32	12	85	21.27	A	筛选后被剔除的值
2021/9/29	净化剂	桶装	93	42	59	20.89	B	

图 5-8 PERCENTILEX.EXC()函数的应用(1)

数据分析时可用 PERCENTILEX.INC()、PERCENTILEX.EXC()或 PERCENTILE.EXC()、PERCENTILE.EXC()函数对极大值、极小值进行剔除,以便确保数据分析的准确性。

5.1.5 VARX.P()

总体(population)是包含所研究的全部个体或数据的集合,有有限总体和无限总体之分。VARX.P()对表表达式的运算值按条件计算总体的方差(跳过空值),语法如下:

```
VARX.P(<table>, <expression>)
```

样本(sample)是从总体中抽取的一部分元素的集合,抽样的目的是根据样本提供的信息来推断总体的特征。VARX.S()函数对表表达式的运算值按条件计算样本的方差(跳过空值)。

VARX.P()函数在不涉及第 1 个参数的额外表运算时,VAR.P()就是它的简写,以下两个表达式的返回值相同:

```
M.入库方差 1:=VAR.P('DK'[入库])

M.入库方差 2:=VARX.P(DK,'DK'[入库])
```

将包装方式拖入透视表的行标签,勾选以上两个度量值,返回的值如图 5-9 所示。

行标签	M.入库 方差1	M.入库 方差2
捆	0	0
散装	0	0
桶装	361	361
箱装	529	529
扎	0	0
总计	372.8163265	372.8163265

图 5-9　VAR.P()函数的应用

5.1.6　STDEVX.P()

在对实际问题进行分析时更多地使用标准差，因为标准差有量纲且它与原始数据的计量单位相同，这是方差所不能比拟的。STDEVX.P()对表表达式的运算值按条件计算总体的标准偏差(跳过空值)，语法如下：

```
STDEVX.P(<table>, <expression>)
```

类似的函数有 STDEVX.S()，此函数用于对表表达式的运算值按条件计算样本的标准偏差。

在数据分析过程中，标准差的参照对象是平均值。创建计算列库.异常值标识，标识入库数据中的异常值，表达式如下：

```
库.异常值标识 :=
VAR A =
    AVERAGE ( DK[入库] )
VAR B =
    STDEVX.P ( DK, DK[入库] )
RETURN
    IF (
        DK[入库] >= A + 1 * B
            || DK[入库] <= A - 1 * B,
        "异常值",
        BLANK ()
    )
```

为了方便理解，隐藏了表中的其他列，如图 5-10 所示。

数据分析时可用 STDEVX.P()、STDEVX.S()或 STDEV.P()、STDEV.S()函数对指定标准差之外的数据进行剔除，确保数据分析的准确性。

	入库	库.异常值标识
1	76	
2	32	异常值
3	55	
4	87	
5	93	异常值
6	78	
7	65	

图 5-10　STDEVX.P()函数的应用

5.1.7 RANKX()

RANKX()用于对某列表中的数字进行排名。第 1 个和第 2 个参数为必选参数,其他为可选参数,语法如下:

```
RANKX(
    <table>,              //要排序的静态物理表或动态虚拟表
    <expression>          //排序依据的表达式
    [,<value>             //修改需要排序的内容,通常情况下无须使用
    [,<order>             //默认为降序(DESC、0 或 FALSE);升序为 ASC、1 或 TRUE
    [,<ties>]]]           //DENSE(稠密排名)或 SKIP(稀疏排名)
)
```

1. 计算列中排序

创建计算列库.入库降序和库.入库升序,表达式如下:

```
库.入库降序 = RANKX(DK,DK[入库])

库.入库升序 = RANKX(DK,DK[入库],,ASC)
```

为了方便理解返回的值,隐藏了表中其他不必要的列,并对入库列进行了升序排列,返回的值如图 5-11 所示。

	入库	等级	库.入库降序	库.入库升序
1	32	A	7	1
2	55	B	6	2
3	65	A	5	3
4	76	A	4	4
5	78	B	3	5
6	87	C	2	6
7	93	B	1	7

图 5-11 RANKX()函数的应用(1)

2. 多重排名分析

创建度量值 M.产品组内排名、M.产品组间排名,表达式如下:

```
M.产品组内排名 =
IF(
    HASONEVALUE(DK[产品]),
    RANKX(ALL(DK[产品]),[M.入库量]),        //ALL()函数用于移除筛选,扩大上下文
    BLANK()
)

M.产品组间排名 = IF(
```

```
    HASONEVALUE(DK[产品]),
    RANKX(ALL(DK[包装方式],DK[产品]),[M.入库量]),
BLANK()
)
```

创建透视表，将 DK 表中的包装方式、产品分别拖入行区域，勾选度量值 M.入库量、M.产品组内排名、M.产品组间排名，返回的值如图 5-12 所示。

行标签	M.入库量	M.产品组内排名	M.产品组间排名
捆			
(空白)	87	1	2
散装			
钢化膜	76	1	4
桶装			
净化剂	93	1	1
油漆	55	2	6
箱装			
蛋糕纸	32	2	7
苹果醋	78	1	3
扎			
包装绳	65	1	5
总计	486		

图 5-12　RANKX()函数的应用(2)

3. 第 2 个参数为聚合函数时

创建度量值 M.排序 1 和 M.排序 2，第 2 个参数为聚合函数的表达式，表达式如下：

```
M.排序 1: = RANKX(DK,SUM(DK[入库]))

M.排序 2: = RANKX(FILTER(DK,DK[等级] = "A"),SUM(DK[入库]))
```

创建透视表，将包装方式拖入行值，勾选 M.排序 1 和 M.排序 2 这两个度量值，返回的值如图 5-13 所示。

行标签	M.排序1	M.排序2
捆	1	1
散装	1	1
桶装	1	1
箱装	1	1
扎	1	1
总计	1	1

图 5-13　RANKX()函数的应用(3)

如图 5-13 所示，返回的值不是期望的值，继续新增度量值，表达式如下：

```
M.排序 3: = RANKX(DK,CALCULATE(SUM(DK[入库])))

M.排序 4: = RANKX(DK,CALCULATE(SUM(DK[入库])),,ASC)

M.排序 5: = RANKX(FILTER(DK,DK[等级] = "A"),CALCULATE(SUM(DK[入库])),,ASC)
```

在以上度量值 M.排序 3～5 的聚合值外面嵌套 CALCULATE()，将行上下文转换为筛选上下文。将新增的度量值继续拖入透视表中，如图 5-14 所示。

在图 5-14 中，M.排序 3 的显示结果仍不是期望值，而度量 M.排序 4～5 中显示的值发

行标签	M.排序1	M.排序2	M.排序3	M.排序4	M.排序5
捆	1	1	1	1	1
散装	1	1	1	1	1
桶装	1	1	1	3	1
箱装	1	1	1	3	2
扎	1	1	1	1	1
总计	**1**	**1**	**1**	**8**	**4**

图 5-14　RANKX()函数的应用(4)

生了变化。这是因为 RANKX()函数的特点：在(默认的)降序排序中，聚合列的排序序号永远是 1，而在升序排序中，聚合列的排序序号则是当前排序列中最大排序号加 1。

5.2　SUMX()迭代函数

SUMX()是使用频率很高的一个函数，用于对表中的每行计算表达式进行求和运算，语法如下：

```
SUMX(<table>, <expression>)
```

SUMX()函数仅对列中的数字进行计数。空白、逻辑值和文本会被忽略。函数中第 1 个参数的表可为静态物理表，也可为动态虚拟表。

5.2.1　第 1 个参数(表)

1. 直接整表引用

在 DK 表中，新建计算列库.数量和，表达式如下：

```
库.数量和:=SUMX('DK','DK'[入库])
```

在 DAX 中，以上表达其实是 SUM('DK'[入库])的简写，返回的值如图 5-15 所示。

	日期	产品	包装方式	入库	出库	方数	吨数	等级	库.数量和
1	2020/9/2	钢化膜	散装	76	35	56	18.86	A	486
2	2020/9/27	蛋糕纸	箱装	32	12	85	21.27	A	486
3	2020/11/2	油漆	桶装	55	23	72	20.89	B	486
4	2021/9/29		捆	87	33		21.22	C	486
5	2021/8/30	净化剂	桶装	93	42	59	20.89	B	486
6	2021/10/11	苹果醋	箱装	78	36	65		B	486
7	2021/11/14	包装绳	扎	65	32	69	22.89	A	486

图 5-15　SUMX()函数的应用(1)

在计算列中，聚合函数 SUMX()不受行上下文影响，每行的值都是整列的聚合值。

2. 通过 FILTER()多条件筛选引用

在 DK 表中，新建计算列库.筛选，首先获取入库> 70 且入库>出库 * 1.5 的动态虚拟表，然后对表中的入库与出库数据进行相乘相加，表达式如下：

```
库.相乘相加 F: =
SUMX (
    FILTER (
        'DK',
        'DK'[入库] > 70
&& 'DK'[入库]> 'DK'[出库] * 1.5
    ),
    'DK'[入库] * 'DK'[出库]
)
```

返回的值如图 5-16 所示。

	日期	产品	包装方式	入库	出库	方数	吨数	等级	库.数量和	库.相乘相加F
1	2020/9/2	钢化膜	散装	76	35	56	18.86	A	486	12245
2	2020/9/27	蛋糕纸	箱装	32	12	85	21.27	A	486	12245
3	2020/11/2	油漆	桶装	55	23	72	20.89	B	486	12245
4	2021/9/29		捆	87	33		21.22	C	486	12245
5	2021/8/30	净化剂	桶装	93	42	59	20.89	B	486	12245
6	2021/10/11	苹果醋	箱装	78	36	65		B	486	12245
7	2021/11/14	包装绳	扎	65	32	69	22.89	A	486	12245

图 5-16　SUMX()函数的应用(2)

通过对比图 5-15 及图 5-16 可看出：在计算列中，聚合类迭代函数同样不受行上下文影响。在 DAX 中，CALCULATE()和 CALCULATETABLE()可改变上下文。在 DK 表中，新建计算列库.乘加 CF，表达式如下：

```
库.乘加 CF : =
CALCULATE (
    SUMX (
        FILTER (
            'DK',
            'DK'[入库] > 70
                && 'DK'[入库] > 'DK'[出库] * 1.5
        ),
        'DK'[入库] * 'DK'[出库]
    )
)
```

返回的值如图 5-17 所示。

	日期	产品	包装方式	入库	出库	方数	吨数	等级	库.数量和	库.相乘相加F	库.乘加CF
1	2020/9/2	钢化膜	散装	76	35	56	18.86	A	486	12245	2660
2	2020/9/27	蛋糕纸	箱装	32	12	85	21.27	A	486	12245	
3	2020/11/2	油漆	桶装	55	23	72	20.89	B	486	12245	
4	2021/9/29		捆	87	33		21.22	C	486	12245	2871
5	2021/8/30	净化剂	桶装	93	42	59	20.89	B	486	12245	3906
6	2021/10/11	苹果醋	箱装	78	36	65		B	486	12245	2808
7	2021/11/14	包装绳	扎	65	32	69	22.89	A	486	12245	

图 5-17　SUMX()函数的应用(3)

图 5-17 中库.乘加 CF 列所返回的值由于发生了上下文转变，所以每行的值均为各行对应的入库 * 出库的值，所有符合条件的值相加（2660 + 2871 + 3906 + 2808）后的总值为 12245。

创建度量值（M.相乘相加 F），表达式如下：

```
M.相乘相加 F: =
SUMX (
    FILTER (
        'DK',
        'DK'[入库] > 70
&& 'DK'[入库]> 'DK'[出库] * 1.5
    ),
    'DK'[入库] * 'DK'[出库]
)
```

行标签	M.相乘相加F
(空白)	2871
钢化膜	2660
净化剂	3906
苹果醋	2808
总计	12245

图 5-18 SUMX()函数的应用(4)

创建透视表，将产品列拖入行标签，勾选 M.相乘相加 F 度量值，如图 5-18 所示。

图 5-18 中的值是图 5-17 中库.乘加 CF 列的值分类汇总，能对应上。度量值的应用原理与说明将在第 6 章详细讲解。本章后续各迭代函数的表达式将主要采用度量值方式进行讲解。

5.2.2 第 2 个参数（条件表达式）

在 SUMX()中，第 1 个参数的表可为动态虚拟表，第 2 个参数可以为复杂的条件表达式。在上面表达式的基础上，对于入库量大于 70 的将按 0.9 的倍数与出库量相乘相加，表达式如下：

```
M.乘加 FI : =
SUMX (
    FILTER (
        'DK',
        'DK'[入库] > 70
            && 'DK'[入库] > 'DK'[出库] * 1.5
    ),
    IF (
        'DK'[入库] >= 70,
        'DK'[入库] * 0.9,
        'DK'[入库]
    ) * 'DK'[出库]
)
```

在图 5-18 的透视表中，勾选新建的 M.乘加 FI 度量值，如图 5-19 所示。

图 5-19 中，M. 乘加 FI＝M. 相乘相加 F＊0.9。通过图 5-18 及图 5-19 不难发现，DAX 很容易实现各类个性化的数据分析。

行标签	M.相乘相加F	M.乘加FI
(空白)	2871	2583.9
钢化膜	2660	2394
净化剂	3906	3515.4
苹果醋	2808	2527.2
总计	12245	11020.5

图 5-19　SUMX()函数的应用(5)

5.2.3　SUMX()综合应用

以下是 SUMX()＋FILTER()的一些条件运算的拓展应用举例，FILTER()的条件可以来自单一的表，也可以来自数据模型中的关联表，甚至可以是两个多对多的表。

1. 同一表内数据的条件求和

创建度量值 M. 入库 F，统计 DK 表中包装方式为箱装和桶装的入库量，表达式如下：

```
M.入库 F :=
SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] IN { "箱装", "桶装" }
    ),
    'DK'[入库]
)
```

在图 5-19 的透视表中，勾选新建的 M. 入库 F，如图 5-20 所示。

行标签	M.相乘相加F	M.乘加FI	M.入库 F
(空白)	2871	2583.9	
蛋糕纸			32
钢化膜	2660	2394	
净化剂	3906	3515.4	93
苹果醋	2808	2527.2	78
油漆			55
总计	12245	11020.5	258

图 5-20　SUMX()函数的应用(6)

在上面表达式的基础上，在 FILTER()原有条件表达式的基础上增加指定产品的要求，表达式如下：

```
M.入库 F&: = SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] IN { "箱装", "桶装" }
          && 'DK'[产品] IN { "蛋糕纸", "钢化膜" , "净化剂" }
    ),
    'DK'[入库]
)
```

在图 5-20 的透视表中，勾选新建的 M.入库 F&，返回的值如图 5-21 所示。

行标签	M.相乘相加F	M.乘加FI	M.入库 F	M.入库 F&
(空白)	2871	2583.9		
蛋糕纸			32	32
钢化膜	2660	2394		
净化剂	3906	3515.4	93	93
苹果醋	2808	2527.2	78	
油漆			55	
总计	**12245**	**11020.5**	**258**	**125**

图 5-21　SUMX()函数的应用(7)

2. 表间关联值条件求和

从运单表中获取指定的包装方式和产品，匹配到装货表中的包装单位，对克与吨为单位的产品进行单位转换(转换为千克)，然后对产品求积求和，表达式如下：

```
M.质量 FV := 
SUMX (
    FILTER (
        '运单',
        '运单'[包装方式]
            IN { "箱装", "桶装", "散装", "扎" }
            && '运单'[产品] IN { "蛋糕纸", "钢化膜", "净化剂", "包装绳" }
    ),
    VAR A =
        RELATED ( '装货'[单位] )
    VAR B =
        SWITCH ( TRUE (), A = "克", 0.001, A = "千克", 1, A = "吨", 1000 )
    RETURN
        B * '运单'[数量] * RELATED ( '装货'[质量] )
)
```

返回的值如图 5-22 所示。

包装方式	产品
箱装	蛋糕纸
散装	钢化膜
扎	包装绳

包装方式	产品	重量	单位
箱装	蛋糕纸	20	千克
散装	钢化膜	150	千克
扎	包装绳	1	吨

(a) 筛选的表

行标签	M.重量FV
散装	4500
箱装	80
扎	25000
总计	**29580**

(b) 返回的值

图 5-22　SUMX()函数的应用(8)

3. 无关系数据的条件求和

现有 DK、DT、DQ 三张表，完整的数据如图 5-23 所示。

日期	产品	包装方式	入库	出库	方数	吨数	等级
2020/9/2	钢化膜	散装	76	35	56	18.86	A
2020/9/27	蛋糕纸	箱装	32	12	85	21.27	A
2020/11/2	油漆	桶装	55	23	72	20.89	B
2021/9/29		捆	87	33		21.22	C
2021/9/29	净化剂	桶装	93	42	59	20.89	B
2021/10/11	苹果醋	箱装	78	36	65		B
2021/11/14	包装绳	扎	65	32	69	22.89	A

表名：DK

日期	年	月
2020/9/2	2020	9
2020/9/3	2020	9
2020/9/27	2020	9
2020/11/2	2020	11
2021/9/29	2021	9
2021/9/30	2021	9
2021/8/30	2021	8
2021/10/1	2021	10
2021/10/11	2021	10
2021/11/14	2021	11

表名：DT

月	季度
8	3
9	3
10	4
11	4

表名：DQ

图 5-23　SUMX()函数的应用(9)

在 DOCK.xlsx 工作簿的 DK 与 DT 表之间创建数据模型及表间关系，其中 DT 表位于一端，DK 表位置多端，如图 5-24 所示。

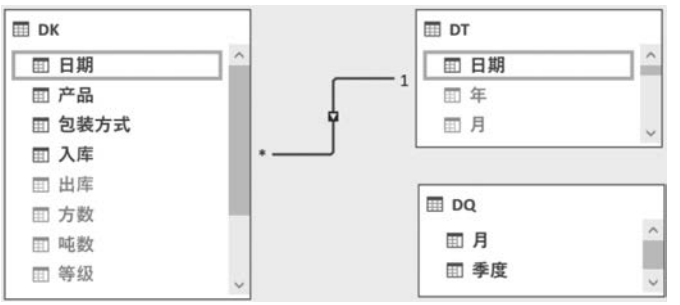

图 5-24　数据模型说明

在 DQ 表中创建计算列季.入库量，引用 DK 表中的入库数据进行求和，表达式如下：

```
季.入库量 := 
SUMX (
    FILTER (
        DK,
        'DQ'[月] = RELATED ( DT[月] )
    ),
    'DK'[入库]
)
```

在上述表达式中，通过 SUMX()＋FILTER()实现无关系数据的获取与求和，返回的值如图 5-25 所示。

	月	季度	季.入库量
1	8	3	
2	9	3	288
3	10	4	78
4	11	4	120

图 5-25　SUMX()函数的应用(10)

5.3 当前行

5.3.1 EARLIER()

EARLIER()可理解为当前行,按照当前行进行逐行扫描运算。在 Excel 2016 及以后版本的 Power BI 中,EARLIER()函数已逐渐被 VAR 变量所取代,语法如下:

```
EARLIER(<column>, <number>)
```

新建计算列,对 DK 表中入库列的数据进行累加,表达式如下:

```
库.入库累加 :=
SUMX (
    FILTER (
        DK,
        DK[入库]<= EARLIER ( DK[入库] )
    ),
    DK[入库]
)
```

为了方便观察,先对入库列数据进行升序排列并隐藏其他不相关的数据列,返回的值如图 5-26 所示。

	入库	库.入库 累加
1	32	32
2	55	87
3	65	152
4	76	228
5	78	306
6	87	393
7	93	486

图 5-26 EARLIER()函数的应用(1)

创建计算列库.入库排名,表达式如下:

```
库.入库排名 :=
COUNTROWS (
    FILTER (
        'DK',
        'DK'[入库] < EARLIER ( 'DK'[入库])
         )
) + 1
```

返回的值如图 5-27 所示。

	包装方式	入库	等级	库.入库 累加	库.入库 排名
1	箱装	32	A	32	1
2	桶装	55	B	87	2
3	扎	65	A	152	3
4	散装	76	A	228	4
5	箱装	78	B	306	5
6	捆	87	C	393	6
7	桶装	93	B	486	7

图 5-27　EARLIER()函数的应用(2)

创建计算列库.包装累加,表达式如下:

```
库.包装累加:=
SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] <= EARLIER ( 'DK'[包装方式])

    ),
    'DK'[入库]
)
```

为了方便观察,先对包装方式列数据进行升序排列,返回的值如图 5-28 所示。

	包装方式	入库	等级	库.入库 累加	库.入库 排名	库.包装 累加
1	捆	87	C	393	6	87
2	散装	76	A	228	4	163
3	桶装	55	B	87	2	311
4	桶装	93	B	486	7	311
5	箱装	32	A	32	1	421
6	箱装	78	B	306	5	421
7	扎	65	A	152	3	486

图 5-28　EARLIER()函数的应用(3)

创建计算列库.包装等级累加,表达式如下:

```
库.包装等级累加 :=
SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] <= EARLIER ( 'DK'[包装方式] )
            && 'DK'[等级] = EARLIER ( 'DK'[等级] )
    ),
    'DK'[入库]
)
```

返回的值如图 5-29 所示。

	包装方式	入库	等级	库.入库 累加	库.入库 排名	库.包装 累加	库.包装等级 累加
1	散装	76	A	228	4	163	76
2	捆	87	C	393	6	87	87
3	箱装	32	A	32	1	421	108
4	桶装	55	B	87	2	311	148
5	桶装	93	B	486	7	311	148
6	扎	65	A	152	3	486	173
7	箱装	78	B	306	5	421	226

图 5-29 EARLIER()函数的应用(4)

5.3.2 VAR 变量

用 VAR 变量来嵌套两个不同的上下文,具有比 EARLIER()更灵活直观的上下文嵌套能力。

在 DK 表中,创建计算列库.入库累加 A,表达式如下:

```
库.入库累加 A:=
VAR A = DK[入库] RETURN
SUMX (
    FILTER (
        DK,
        DK[入库]<= A ),
    DK[入库]
)
```

在 DK 表中,创建计算列库.入库排名 A,表达式如下:

```
库.入库排名 A:=
VAR A = 'DK'[入库] RETURN
COUNTROWS (
    FILTER (
        'DK',
        'DK'[入库] < A)
) + 1
```

在表中继续创建计算列库.包装累加 A,表达式如下:

```
库.包装累加 A:=
VAR A = 'DK'[包装方式] RETURN
SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] <= A ),
    'DK'[入库]
)
```

继续新增计算列库.包装等级累加 A,表达式如下：

```
库.包装等级累加 A := 
VAR A = 'DK'[包装方式]
VAR B = 'DK'[等级]
RETURN
SUMX (
    FILTER (
        'DK',
        'DK'[包装方式] <= A
          && 'DK'[等级] = B
    ),
    'DK'[入库]
)
```

返回的值如图 5-30 所示。

	包装方式	入库	等级	库.入库 累加A	库.入库 排名A	库.包装 累加A	库.包装等级 累加A
1	箱装	32	A	32	1	421	108
2	桶装	55	B	87	2	311	148
3	扎	65	A	152	3	486	173
4	散装	76	A	228	4	163	76
5	箱装	78	B	306	5	421	226
6	捆	87	C	393	6	87	87
7	桶装	93	B	486	7	311	148

图 5-30　VAR 变量的应用

5.4　CONCATENATEX()

CONCATENATEX()函数用于文本迭代,该函数有两个必选参数,语法如下：

```
CONCATENATEX(
    <table>,
    <expression>
    [, <delimiter>
    [, <orderBy_expression>
    [, <order>]]...]
)
```

利用 CONCATENATEX()函数创建度量值,表达式如下：

```
M.文本串接 := 
CONCATENATEX (
    DISTINCT ( 'DK'[包装方式] ),
```

```
    'DK'[包装方式],
    "、"
)
```

创建透视表，将产品拖入行标签，勾选 M.文本串接度量值，返回的值如图 5-31 所示。

行标签	M.文本串接
(空白)	捆
包装绳	扎
蛋糕纸	箱装
钢化膜	散装
净化剂	桶装
苹果醋	箱装
油漆	桶装
总计	散装、箱装、桶装、捆、扎

图 5-31 CONCATENATEX ()函数的应用(1)

结合 HASONEVALUE()函数，创建度量值 M.产品名串接，表达式如下：

```
M.产品名串接:= IF(HASONEVALUE('DK'[包装方式]),
        CONCATENATEX(VALUES('DK'[产品]),
        'DK'[产品],
        "、")
)
```

在 DAX 中，HASONEVALUE()返回的值为 TRUE 或 FALSE，常用于 IF()表达式的第 1 个参数。创建透视表，将包装方式拖入行标签，勾选 M.产品名串接度量值，返回的值如图 5-32 所示。

行标签	M.产品名串接
捆	
散装	钢化膜
桶装	油漆 、 净化剂
箱装	蛋糕纸 、 苹果醋
扎	包装绳

图 5-32 CONCATENATEX ()函数的应用(2)

5.5 本章回顾

本章主要对 SUMX()、RANKX()、AVERAGEX()、MAXX()、CONCATENATX()等迭代聚合函数的用法及其背后的数理统计知识进行了深入介绍。