智达方通多维数据库

Intcube Booster v1.0

MDX 函数语法手册

北京智达方通科技有限公司

2022-09

MDX-Multidimensional Expressions 是多维数据库的查询及计算语言，是多维数据库的标准功能。MDX 可用在包含多维数据查询、修改和计算的各类脚本中。本文档提供智达方通多维数据库 Intcube Booster v1.0 的 MDX 语法及函数的使用方法。

文档中示例用到的业务规则及多维数据表单名称如下，用户也可在 Intcube Booster 的预置模型中找到它们：

业务规则：

A03 管理费用表系列规则

多维数据表单：

A03 管理费用表

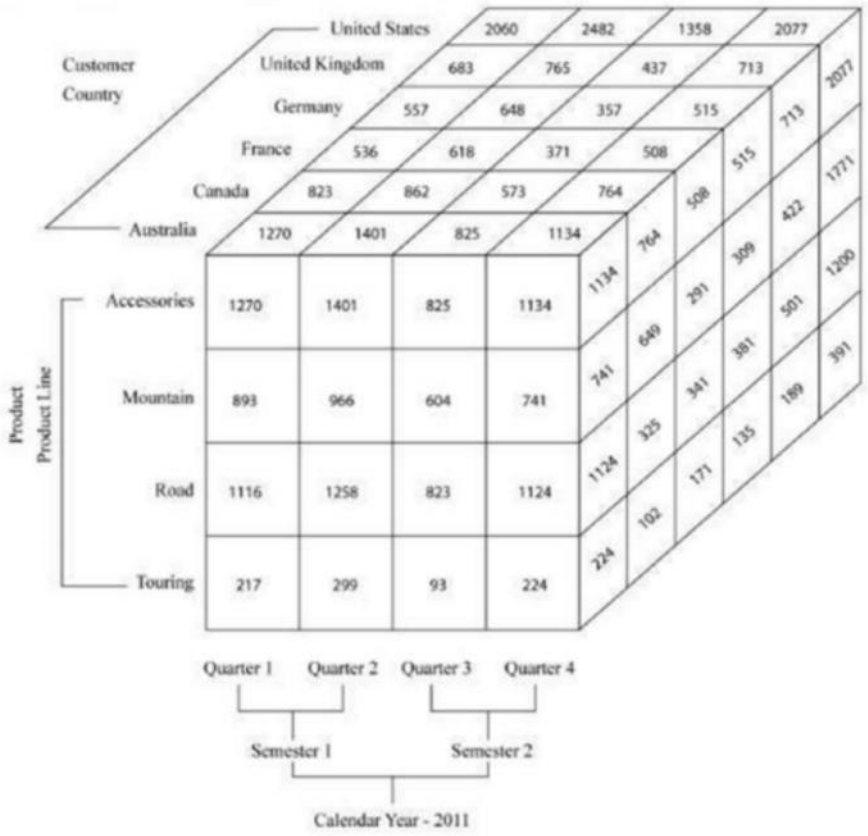
概念介绍

多维数据集 Cube

Cube 是一组多维数据集。其数据为由多个维度中的维度成员交叉形成的单元格数据组成。三个维度可构成空间结构立方体-Cube， 多个维度构成的则是超立方体 hypercube。在多维数据库领域中，Cube 即可代表由多个维度构成的数据集。

Cube 中的数据通常是为了做业务数据的记录和分析（比如通过对比各个月份的某产品的销售额，来确定该产品的销售旺季和淡季）。一个 cube 的主要组成部分是维度和度量。维度定义多维数据集的数据结构，用于对数据进行分类、划分和展示数据细节。度量则是多维数据结构中，用于度量的数值型的、量化的数据。在一些特殊情况，度量值也可以是文本型或其他类型的数据。（本文档使用的实例库中，度量是在度量维度中定义的）

下面是一个 3 维的 cube 示例图：



维度 dimension

维度是用于多维数据集中事实数据分类的结构。常用的维度包括：科目、组织、时间、产品等。

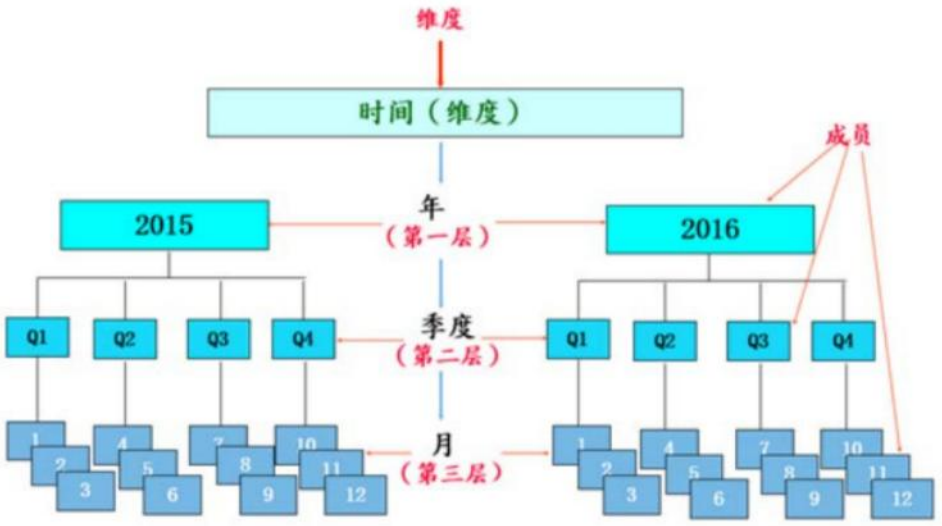
层次结构 hierarchy、级别 level 和成员 member

Hierarchy-层次结构是维度大纲的结构表示。一个维度大纲中，可以有多个层级结构。例如，时间维度中，可以有日历年度层级结构，如全年，1，2，3，4 季度，1-12 月，和财务年度层级结构。层级结构决定其中各层级中维度成员的聚集计算顺序和关系。

如图是时间维度一个层次结构的图例，每一层即为一个级别，每个级别有 1 到多

个成员（如：图中年级别有两个成员 2015 和 2016，季度级别有四个成员 Q1、Q2、Q3、Q4）。级别的编号是从下往上， 所以示例图中月级别为 0 级别，季度为 1 级别，年为 2 级别。

时间维度可以有更多的层次结构，当前是日历类型的层次结构，完全可以再定义一个财务年度类型的层次结构



Member-成员是维度层级结构中的成员。Member 可代表 Cube 中事实数据的一个特征。

元组 tuple

元组用于定义或标识多维数据集 Cube 中的一个切面或者切块数据。它是由一个或多个维度中的一个或多个维度成员组合而成。如果一个元组 Tuple 是由 Cube 中的每一个维度中的一个成员组合而成，那么它描述的就是一个单元格事实数据。在 MDX 查询或表达式中定义元组时，不需要显式地包含每个维度层次结构中的

成员。如果查询或表达式中没有显式包含维度层次结构的成员，则该维度层次结构的默认成员是隐式包含在元组中的成员。除非在多维数据集中明确定义，否则如果存在(All)成员，则每个维度层次结构的默认成员都是(All)成员。如果维度层次结构中不存在(All)成员，则默认成员是该层次结构的顶级成员。

格式为小括号（）中以逗号分隔指定各个维度轴上的成员，cube 示例图中（Australia，Accessories，Quarter1）指定正面左上角值为 1270 的单元格。(Australia)这种其他属性轴层次结构的成员都是隐式包含，元组补齐后有可能为（Australia，Touring，2011），后两个成员是当前上下文（最初的上下文是各个属性轴的默认成员，在表达式中指定了某个属性轴上的成员，就会覆盖上下文中原有的默认成员）中的成员。

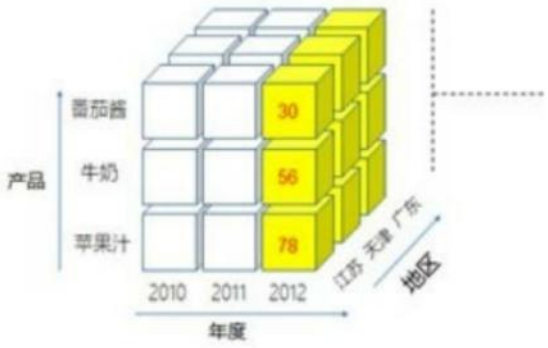
集合 set

集合包含 0 个至多个 tuple，可以是示例中的样式，也可以是某些函数的结果集（children 函数返回的就是一个集合）

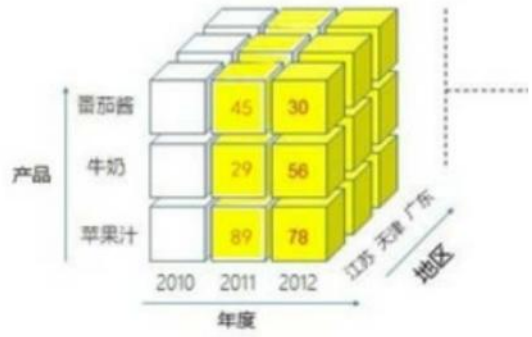
示例：

{
tuple1,
tuple2,
tuple3
}

切片和切块



以上图三个维度的多维数据集为例，切片是选择一个维度的成员查询。切片是会降维的（三维→二维），如图选择 2012 年各地区各产品销售额



切块是不降维的（仍然是三维），如图是 2011、2012 年各地区各产品销售额

第一章 集合函数

Ancestors

返回指定成员在指定级别上的所有祖先的集合。

语法

ANCESTORS(member, level)

示例

select

Ancestors([组织].[XYZ 集团].[北京总部].[研发中心].[开发 1 部],[组织].[LEVEL1]) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([期间].&[50652],[场景].&[50685],[产品].&[50688],[科目].&[50253])

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 10 月

科目: 物业外包

组织	年初编报 01 版本
研发中心	50.00

组织维度成员[开发 1 部]层级为 0，他层级为 1 的祖先为[研发中心]。

BottomCount

返回指定集合中指定数目具有较小值的组合组成的集合

语法

BOTTOMCOUNT(set , index [,numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

index

是返回组合的数目（即单元格的数目）。

numeric_value

（可选）如果指定了 numeric_value，此函数将根据 numeric_value 计算在集合上的值，并按升序对组合进行排序。

示例一

select

BottomCount({[期间].[2023 年].[2023 年 2 季度].Children},2) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
2023 年 4 月	7.00
2023 年 5 月	2.00

期间维度[2023 年 2 季度]的子项有 3 个分别为[2023 年 4 月]、[2023 年 5 月]、[2023 年 6 月]，对应的[办公费]值分别为 7、2、1，由于 index 值为 2 且没有指定 numeric_value，所以顺序返回[2023 年 4 月]和[2023 年 5 月]。

示例二

```
select
BottomCount({[期间].&[50643].Children},2,[科目].&[50220]) on rows,
{[科目].&[50220]} on columns
from [模型一]
where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织: 开发 1 部

期间	办公费
2023 年 6 月	1.00
2023 年 5 月	2.00

期间维度[2023 年 2 季度]的子项有 3 个分别为[2023 年 4 月]、[2023 年 5 月]、[2023 年 6 月]，对应的[办公费]值分别为 7、2、1，由于指定了 numeric_value 参数，所以要先对 7、2、1 进行升序排列，结果为 1、2、7，并且 index 的值为 2，所以会取升序排列后前两个的值，返回对应的[2023 年 6 月]和[2023 年 5 月]。

BottomPercent

将指定维度成员范围内数据升序排列，从最小值开始累加，当累加大于或等于范围总和的指定百分比后，返回集合，再结合数值函数，将集合中的值写到单元格中。

语法

BOTTOMPERCENT(set , percentage ,numeric_value)

参数

Percentage

是返回的单元格的值占 set 指定的所有单元格值的和的最小比例

numeric_value

根据 numeric_value 计算在集合上的值，并按升序对指定集合中的组合进行排序

备注

百分比不需要输入%

示例

```
select
BottomPercent({[科目].&[50220]}*{[期间].[2023 年].[2023 年 2 季度].Children},30,[组织].&[50584])on rows,
{[组织].&[50584]} on columns
```

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目	期间	开发 1 部
办公费	2023 年 6 月	1.00
	2023 年 5 月	2.00

由于期间维度[2023 年 2 季度]的子项[2023 年 4 月]、[2023 年 5 月]、[2023 年 6 月]的[科目].&[50220]值分别为 7、2、1，按升序从 1、2、7 里面取，指定百分比为 30，也就是 1、2、7 先求和再乘以 30%为 3，当取到 2 时，1+2 刚好等于 3，满足大于等于 3 的条件，所以返回对应的[2023 年 6 月]和[2023 年 5 月]。

BottomSum

将指定维度成员范围内数据升序排列，从最小值开始累加，当累加大于或等于给定的值后，返回集合。

语法

BOTTOMSUM(set, value, numeric_value)

参数

set

返回集的有效多维表达式 (MDX)。

Value

指定值 value

Numeric_value

返回集的排序依据，优先级高于 Set_Expression。

备注

根据 numeric_value 计算在集合上的值，按升序排列之后，返回较小值的一组元组，其累加和等于或者刚超过指定值 value。

示例一

select

BottomSum({[科目].&[50236]}*{[期间].[2023 年].[2023 年 1 季度].Children},2,[组织].&[50582])on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目	期间	北京总部本部
研究开发费	2023 年 2 月	0.00
	2023 年 1 月	2.00

由于期间维度[2023 年 1 季度]的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3

月]的[研究开发费]值分别为 2、0、24，在 BottomSum 函数的集合中，期间维度位置靠后，所以会对[研究开发费]的[2023 年 3 季度]的子项的值进行升序排列，顺序为 0、2、24，对应返回的期间顺序为[2023 年 2 月]、[2023 年 1 月]、[2023 年 3 月]；根据 numeric_value 计算在集合上的值，按升序排列之后，返回较小值的一组元组，其累加和等于或大于指定值 value=2，所有返回集合为 0、2，对应返回的期间顺序为[2023 年 2 月]、[2023 年 1 月]。

Children

返回指定成员的子代集合

语法

member.Children

示例

select

[期间].[2023 年].[2023 年 3 季度].Children on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

期间维度[2023 年 3 季度]的子带有[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]三个月份.

CrossJoin

语法

CROSSJOIN(set, set, ……..)
set * set……

参数

set

返回集的有效多维表达式（MDX），至少要有两个。

备注

返回一个或多个集合的外积，例如，当第一个集合包含{x1, x2，…，xn}，第二个集合由{y1, y2，…， yn}，这些集合的外积是

{(x1, y1), (x1, y2),…,(x1, yn), (x2, y1), (x2, y2),…,
(x2, yn),…, (xn, y1), (xn, y2),…, (xn, yn)}

示例

select

CrossJoin({[科目].&[50236],[科目].&[50231]},{[期间].&[50640],[期间].&[50641],[期间].&[50642]})on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目	期间	北京总部本部
研究开发费	2023 年 1 月	2.00
	2023 年 2 月	0.00
	2023 年 3 月	24.00
无形资产摊销	2023 年 1 月	
	2023 年 2 月	
	2023 年 3 月	

由于期间维度[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[研究开发费]值分别为 2、0、24，[无形资产摊销]值为 null；在 CROSSJOIN 中返回这几个集合的外积，分别为 2、0、24.

Descendants

返回成员在指定级别上的后代集合，可以选择包括或不包括其他级别上的后代。

语法

DESCENDANTS(member, level [, desc_flags])

参数

desc_flags 用于指定后代集合的类别

Desc_flags 类别

Flag	Description
SELF	只返回来自指定级别的后代成员。如果指定的级别是指定成员的级别，则返回结果包含指定的成员。
AFTER	返回从属于指定级别的所有级别的后代成员
BEFORE	返回指定成员与指定级别之间的所有级别的后代成员。它包括指定的成员，但不包括来自指定级别的成员。
BEFORE_AND_AFTER	返回从属于指定成员级别的所有级别的后代成员。它包括指定的成员，但不包括来自指定级别的成员。
SELF_AND_AFTER	返回来自指定级别的后代成员，以及从属于指定级别所有级别的成员。
SELF_AND_BEFORE	返回来自指定级别的后代成员，以及来自指定成员与指定级别之间的所有级别(包括指定成员)的后代成员。
SELF_BEFORE_AFTER	返回从属于指定成员级别的所有级别

	的后代成员，并包括指定成员。

示例一

select

Descendants([科目].&[50187],[科目].[LEVEL0],SELF) on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[期间].[2023 年].[2023 年 3 季度].[2023 年 7 月])

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 7 月

版本: 年初编报 01 版本

科目	北京总部本部
工资	-10000.00
福利费	4200.00
社保费	6800.00
住房公积金	3500.00
奖金	240.00

行上成员是【科目】维度 key 为[50187]的成员的后代中级别为 0 的成员

示例二

select

Descendants([科目].&[50187],[科目].[LEVEL0]) on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[期间].[2023 年].[2023 年 3 季度].[2023 年 7 月])

运行结果:

产品: 不分

场景: 累计预算

期间: 2023 年 7 月

版本: 年初编报 01 版本

科目	北京总部本部
工资	-10000.00
福利费	4200.00
社保费	6800.00
住房公积金	3500.00
奖金	240.00

结果与示例 1 相同

Distinct

计算指定集合，从集合中移除重复元组，并返回结果集。

语法

DISTINCT(set)

参数

set

返回集的有效多维表达式（MDX）。

示例

select

Distinct({[期间].&[50640],[期间].&[50641],[期间].&[50642],[期间].&[50642],[期间].&[50642]})on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 1 月	2.00
2023 年 2 月	0.00
2023 年 3 月	24.00

由于期间维度[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[研究开发费]值分别为 2、0、24，期间维度[2023 年 3 月]重复了 3 次，在 DISTINCT 函数计算该集合时，从集合中移除重复元组[2023 年 3 月]，并返回结果集，分别为 2、0、24.

DrilldownLevel

将某个集的成员深化到该集中所表示的最低级别的下一个级别。

指定向下钻取的级别是可选的，但如果设置了级别，则可以使用 级别表达式 或 索引级别。 这两种参数互相排斥。 最后，如果计算成员出现在查询中，你可指定一个聚合以将这些成员包含在行集中。

语法

DRILLDOWNLEVEL(set [[,level] | [,,index]])

参数

set

返回集的有效多维表达式 (MDX)。

level

（可选）显式标识维度上要钻取层级的 MDX 表达式。

Index

（可选）有效的数值表达式，它指定要钻取的维度在集合中从左往右的顺序编号，从 0 开始。

备注

leveL 和 index 不可同时存在。

当 level 和 index 未指定，或者参数无效时，默认对集合中第一个维度层级小的成员进行下钻。

示例一

```
select
DrilldownLevel({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]},[期间].[LEVEL2])
on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年 1 季度	200.00
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 2 季度	400.00
2023 年 3 季度	300.00
2023 年 4 季度	300.00

由于集合中期间维度[期间].[2023 年]的层级为 2，符合第二个参数的层级[期间].[LEVEL2]，所以对[期间].[2023 年]进行下钻，返回[2023 年]和它的子项[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]。

而[期间].[2023 年].[2023 年 1 季度]的层级为 1，不符合第二个参数的层级[期间].[LEVEL2]，所以不进行下钻。

示例二

```
select
DrilldownLevel({[科目].&[50215]}*{[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]},,1) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

科目	期间	年初编报 01 版本
工资-管理费用合计	2023 年 1 季度	200.00
	2023 年 1 月	200.00
	2023 年 2 月	-100.00
	2023 年 3 月	100.00
	2023 年	1200.00

集合中第一个维度为[科目]，第二个维度为[期间]，由于 Index 为 1（从 0 开始），所以对[期间]维度进行下钻，而[期间].[2023 年].[2023 年 1 季度]层级为 1，[期

间].[2023 年]层级为 2，优先对层级低的成员进行下钻，所以对[2023 年].[2023 年 1 季度]进行下钻，返回[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]。

示例三

```
select
DrilldownLevel({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}*{[科目].&[50215]}) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

期间	科目	年初编报 01 版本
2023 年 1 季度	工资-管理费用合计	200.00
2023 年 1 月	工资-管理费用合计	200.00
2023 年 2 月	工资-管理费用合计	-100.00
2023 年 3 月	工资-管理费用合计	100.00
2023 年	工资-管理费用合计	1200.00

集合中第一个维度为[期间]，第二个维度为[科目]，由于未指定参数，所以对第一个维度[期间]维度进行下钻，而[期间].[2023 年].[2023 年 1 季度]层级为 1，[期间].[2023 年]层级为 2，优先对层级小的成员进行下钻，所以对[2023 年].[2023 年 1 季度]进行下钻，返回[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]。

DrilldownLevelBottom

默认对集合中第一个维度下级别最低维度成员进行下钻，返回下钻后的前几个成员组成的集合。如果指定了 level，则对集合中指定维度下指定层级维度成员进行下钻。如果指定了 numeric_value，则优先对下钻出的成员进行升序排列，再返回前几个成员。

语法

DRILLDOWNLEVELBOTTOM(set, count [, [level] , numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

count

限制钻取的 tuple 数目（不能超过 count 值）。

level

（可选) 不指定 level 则下钻 set 中第一个维度的最低级别的成员；level 指定了，筛选出 set 中该级别上的成员做下钻操作。

numeric_value

不指定则直接按照层次结构的顺序取最多 count 个子代成员；

如果指定了，钻取的子代成员先排序（升序），再从小到大取最多 count 个子代

成员;

示例一

```
select
DrilldownLevelBottom({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}*{[科目].&[50215]},2) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果:

产品: 不分

场景: 累计预算

组织: 开发 2 部

期间	科目	年初编报 01 版本
2023 年 1 季度	工资-管理费用合计	200.00
2023 年 1 月	工资-管理费用合计	200.00
2023 年 2 月	工资-管理费用合计	-100.00
2023 年	工资-管理费用合计	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于未指定 level，默认对 set 中第一个维度[期间]进行下钻，[2023 年 1 季度]的级别为 1，[2023 年]的级别为 2，由于[2023 年 1 季度]的级别小，所以会对[2023 年 1 季度]进行下钻。

未指定 numeric_value，默认按原维度成员顺序排列。由于 count 为 2，所以只返回[2023 年 1 月]和[2023 年 2 月]。

示例二

```
select
DrilldownLevelBottom({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}*{[科目].&[50215]},2,,[版本].&[50672]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

期间	科目	年初编报 01 版本
2023 年 1 季度	工资-管理费用合计	200.00
2023 年 2 月	工资-管理费用合计	-100.00
2023 年 3 月	工资-管理费用合计	100.00
2023 年	工资-管理费用合计	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于未指定 level，默认对 set 中第一个维度[期间]进行下钻，[2023 年 1 季度]的级别为 1，[2023 年]的

级别为 2，由于[2023 年 1 季度]的级别小，所以会对[2023 年 1 季度]进行下钻。由于指定了 numeric_value，会将[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的值按照升序排列，-100、100、200。由于 count 为 2，所以只取数值升序排列后前两个对应的维度分别是[2023 年 2 月]和[2023 年 3 月]。

示例三

```
select
DrilldownLevelBottom({[科目].&[50215]}*{[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]},2,[期间].[LEVEL1],[版本].&[50672]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

科目	期间	年初编报 01 版本
工资-管理费用合计	2023 年 1 季度	200.00
	2023 年 2 月	-100.00
	2023 年 3 月	100.00
	2023 年	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3

月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于指定了 level，[期间].[LEVEL1]，而集合中[2023 年 1 季度]的级别为 1，所以会对[2023 年 1 季度]进行下钻。

由于指定了 numeric_value，会将[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的值按照升序排列，-100、100、200。由于 count 为 2，所以只取数值升序排列后前两个对应的维度分别是[2023 年 2 月]和[2023 年 3 月]。

DrillDownLevelTop

默认对集合中第一个维度下级别最低维度成员进行下钻，返回下钻后的前几个成员组成的集合。如果指定了 level，则对集合中指定维度下指定层级维度成员进行下钻。如果指定了 numeric_value，则优先对下钻出的成员进行降序排列，再返回前几个成员。

语法

DRILLDOWNLEVELTOP(set, count　[, [level] , numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

count

限制钻取的 tuple 数目（不能超过 count 值）。

level

不指定 level 则下钻第一个维度的最低级别的成员；

指定了，筛选出 set 中该级别上的成员做下钻操作。

numeric_value

不指定则直接按照层次结构的顺序取最多 count 个子代成员；

如果指定了，钻取的同级成员先排序（降序），再从大到小取最多 count 个子代成员;

示例一

```
select
DrilldownLevelTop({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}*{[科目].&[50215]},2) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果:

产品: 不分

场景: 累计预算

组织: 开发 2 部

期间	科目	年初编报 01 版本
2023 年 1 季度	工资-管理费用合计	200.00
2023 年 1 月	工资-管理费用合计	200.00
2023 年 2 月	工资-管理费用合计	-100.00
2023 年	工资-管理费用合计	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于未指定 level，默认对 set 中第一个维度[期间]进行下钻，[2023 年 1 季度]的级别为 1，[2023 年]的

级别为 2，由于[2023 年 1 季度]的级别小，所以会对[2023 年 1 季度]进行下钻。未指定 numeric_value，默认按原维度成员顺序排列。由于 count 为 2，所以只返回[2023 年 1 月]和[2023 年 2 月]。

示例二

```
select
DrilldownLevelTop({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}*{[科目].&[50215]},2,,[版本].&[50672]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

期间	科目	年初编报 01 版本
2023 年 1 季度	工资-管理费用合计	200.00
2023 年 1 月	工资-管理费用合计	200.00
2023 年 3 月	工资-管理费用合计	100.00
2023 年	工资-管理费用合计	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于未指定 level，默

认对 set 中第一个维度[期间]进行下钻，[2023 年 1 季度]的级别为 1，[2023 年]的级别为 2，由于[2023 年 1 季度]的级别小，所以会对[2023 年 1 季度]进行下钻。由于指定了 numeric_value，会将[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的值按照降序排列，200、100、-100。由于 count 为 2，所以只取数值升序排列后前两个对应的维度分别是[2023 年 1 月]和[2023 年 3 月]。

示例三

```
select
DrilldownLevelTop({[科目].&[50215]}*{[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]},2,[期间].[LEVEL1],[版本].&[50672]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 开发 2 部

科目	期间	年初编报 01 版本
工资-管理费用合计	2023 年 1 季度	200.00
	2023 年 1 月	200.00
	2023 年 3 月	100.00
	2023 年	1200.00

期间维度[2023 年 1 季度]和它的子项[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的[工资-管理费用合计]值分别为 200、200、-100、100，由于指定了 level，[期间].[LEVEL1]，而集合中[2023 年 1 季度]的级别为 1，所以会对[2023 年 1 季度]进行下钻。

由于指定了 numeric_value，会将[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的值按照降序排列，200、100、-100。由于 count 为 2，所以只取数值升序排列后前两个对应的维度分别是[2023 年 1 月]和[2023 年 3 月]。

DrillDownMember

遍历集合 1 中的成员，如果也存在于集合 2 中则做该成员的下钻。结果集包含集合 1 中的成员和下钻得到的成员。

语法

DRILLDOWNMEMBER(set, set [, RECURSIVE])

参数

set

返回集的有效多维表达式（MDX）。

recursive

（可选）指定了 recursive，则递归地将结果集中的成员与第二个集合进行比较，判断是否继续下钻

备注

如果第一个集合包含父成员和一个或者多个子成员，则父成员不会下钻；

第二个集合的维度只能有一个；

示例一

select

DrilldownMember({[期间].[2023 年]},{[期间].[2023 年]}) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 2 季度	400.00
2023 年 3 季度	300.00
2023 年 4 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，结果集中返回[2023 年]和他的子项。

示例二

select

DrilldownMember({[期间].[2023 年]},{[期间].[2023 年],[期间].[2023 年].[2023 年 1

季度]},recursive) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 1 月	200.00
2023 年 2 月	-100.00
2023 年 3 月	100.00
2023 年 2 季度	400.00
2023 年 3 季度	300.00
2023 年 4 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，结果集中返回[2023 年]和他的子项。由于有 recursive 参数，且集合 2 中有[2023 年]的子项[2023 年 1 季度]，所以对[2023 年 1 季度]进

行下钻，结果集中返回[2023 年 1 季度]的子项。

DrillDownMemberBottom

遍历集合 1 中的成员，如果也存在于集合 2 中则做该成员的下钻。结果集包含集合 1 中的成员和下钻得到的成员，默认顺序排列。若指定了 numeric_value 参数，则下钻得到的成员按值进行升序排列，再返回不大于 count 个数成员。若指定了 recursive 参数，并且集合 2 中存在集合 1 第一次钻取的子项，则对该子项进行下钻，返回的子项同时受 recursive 和 count 参数影响，效果同上。

语法

DRILLDOWNMEMBERBOTTOM(set, set, count [, [numeric_value] [, RECURSIVE]])

参数

set

返回集的有效多维表达式（MDX)。

count

限制下钻的 tuple 的数目（不超过 count 值）

numeric_value

（可选）指定了 numeric_value 则下钻的同级成员会据此排序（升序）

recursive

（可选）指定了 recursive，则递归地将结果集中的成员与第二个集合进行比较，判断是否继续下钻

备注

如果第一个集合包含父成员和一个或者多个子成员，则父成员不会下钻；

第二个集合的维度只能有一个；

示例一

select

DrilldownMemberBottom({[期间].[2023 年]},{[期间].[2023 年]},2) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 2 季度	400.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，根据 count 的值，结果集中返回[2023 年]和他的前两个子项[2023 年 1 季度]和[2023 年 2 季度]。

示例二

select

DrilldownMemberBottom({[期间].[2023 年]},{[期间].[2023 年]},2,[版本].&[50672])

on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果:

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 4 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，对应的值分别为 200、400、300、300。由于指定了 numeric_value 参数，所以将[2023 年]的子项的值按照升序排列，结果为 200、300、300、400。根据 count 的值，结果集中返回[2023 年]和他子项中升序排列的前两项分别为[2023 年 1 季度]和[2023 年 4 季度]。

示例三

select

DrilldownMemberBottom({[期间].[2023 年]},{[期间].[2023 年],[期间].[2023 年].[2023 年 1 季度]},2,,recursive) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果:

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 1 月	200.00
2023 年 2 月	-100.00
2023 年 2 季度	400.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，由于指定了 recursive 参数，且集合 2 中有[2023 年 1 季度]，所以对[2023 年 1 季度]下钻，[2023 年 1 季度]的子项有[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]。根据 count 的值，结果集中返回[2023

年]和他子项中前两项分别为[2023 年 1 季度]和[2023 年 2 季度]还有[2023 年 1 季度]子项中的前两项[2023 年 1 月]和[2023 年 2 月]。

示例四

select

DrilldownMemberBottom({[期间].[2023 年]},{[期间].[2023 年],[期间].[2023 年].[2023 年 1 季度]},2,[版本].&[50672],recursive) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果:

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00
2023 年 2 月	-100.00
2023 年 3 月	100.00
2023 年 4 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，

所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，对应的值分别为 200、400、300、300。由于指定了 numeric_value 参数，所以将[2023 年]的子项的值按照升序排列，结果为 200、300、300、400。根据 count 的值，结果集中返回[2023 年]和他子项中升序排列的前两项分别为[2023 年 1 季度]和[2023 年 4 季度]。

由于指定了 recursive 参数，且[2023 年 1 季度]也在集合 2 中，所以对[2023 年 1 季度]进行下钻，[2023 年 1 季度]的子项有[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]，对应的值分别为 200、-100、100，升序排列后为-100、100、200，根据 count 的值，结果集中返回[2023 年 1 季度]和他子项中升序排列的前两项分别为[2023 年 2 月]和[2023 年 3 月]。

DrillDownMemberTop

遍历集合 1 中的成员，如果也存在于集合 2 中则做该成员的下钻。结果集包含集合 1 中的成员和下钻得到的成员，默认顺序排列。若指定了 numeric_value 参数，则下钻得到的成员按值进行降序排列，再返回不大于 count 个数成员。若指定了 recursive 参数，并且集合 2 中存在集合 1 第一次钻取的子项，则对该子项进行下钻，返回的子项同时受 recursive 和 count 参数影响，效果同上。

语法

DRILLDOWNMEMBERTOP(set, set, count [, [numeric_value] [, RECURSIVE]])

参数

set

返回集的有效多维表达式 (MDX)。

count

限制下钻的 tuple 的数目（不超过 count 值）

numeric_value

（可选）指定了 numeric_value 则下钻的同级成员会据此排序（降序）

recursive

（可选）指定了 recursive，则递归地将结果集中的成员与第二个集合进行比较，判断是否继续下钻

备注

如果第一个集合包含父成员和一个或者多个子成员，则父成员不会下钻；

第二个集合的维度只能有一个；

示例一

```
select
DrilldownMemberTop({[期间].[2023 年]},{[期间].[2023 年]},2) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00

2023 年 1 季度	200.00
2023 年 2 季度	400.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，根据 count 的值，结果集中返回[2023 年]和他的前两个子项[2023 年 1 季度]和[2023 年 2 季度]。

示例二

```
select
DrilldownMemberTop({[期间].[2023 年]},{[期间].[2023 年]},2,[版本].&[50672]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 2 季度	400.00
2023 年 3 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，对应的值分别为 200、400、300、300。由于指定了 numeric_value 参数，所以将[2023 年]的子项的值按照降序排列，结果为 400、300、300、200。根据 count 的值，结果集中返回[2023 年]和他子项中降序排列的前两项分别为[2023 年 2 季度]和[2023 年 3 季度]。

示例三

```
select
DrilldownMemberTop({[期间].[2023 年]},{[期间].[2023 年],[期间].[2023 年].[2023 年 1 季度]},2,,recursive) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00

2023 年 1 月	200.00
2023 年 2 月	-100.00
2023 年 2 季度	400.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，由于指定了 recursive 参数，且集合 2 中有[2023 年 1 季度]，所以对[2023 年 1 季度]下钻，[2023 年 1 季度]的子项有[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]。根据 count 的值，结果集中返回[2023 年]和他子项中前两项分别为[2023 年 1 季度]和[2023 年 2 季度]还有[2023 年 1 季度]子项中的前两项[2023 年 1 月]和[2023 年 2 月]。

示例四

```
select
DrilldownMemberTop({[期间].[2023 年]},{[期间].[2023 年],[期间].[2023 年].[2023 年 2 季度]},2,[版本].&[50672],recursive) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 2 季度	400.00
2023 年 4 月	200.00
2023 年 5 月	100.00
2023 年 3 季度	300.00

集合 1 和集合 2 中均有期间维度[2023 年]，并且集合 1 中没有[2023 年]的子项，所以对[2023 年]进行下钻，[2023 年]的子项有[2023 年 1 季度]、[2023 年 2 季度]、[2023 年 3 季度]、[2023 年 4 季度]，对应的值分别为 200、400、300、300。由于指定了 numeric_value 参数，所以将[2023 年]的子项的值按照降序排列，结果为 400、300、300、200。根据 count 的值，结果集中返回[2023 年]和他子项中降序排列的前两项分别为[2023 年 2 季度]和[2023 年 3 季度]。

由于指定了 recursive 参数，且[2023 年 2 季度]也在集合 2 中，所以对[2023 年 2 季度]进行下钻，[2023 年 2 季度]的子项有[2023 年 4 月]、[2023 年 5 月]、[2023 年 6 月]，对应的值分别为 200、100、100，降序排列后为 200、100、200，根据 count 的值，结果集中返回[2023 年 2 季度]和他子项中降序排列的前两项分别为[2023 年 4 月]和[2023 年 5 月]。

DrillupLevel

默认用集合中第一个维度中去除级别最低的维度成员来组成结果集。指定 level

后用不低于指定级别的成员组成结果集。

语法

DRILLUPLEVEL(set [, level])

参数

set

返回集的有效多维表达式（MDX)。

level

（可选）指定了 level，筛选集合中不低于指定级别的成员，如果没有指定 level，则该函数通过检索比指定集合中引用的第一个维度的最低级别高的成员来构造集合。

示例一

select

DrillupLevel({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]}) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本

2023 年	1200.00

集合中只有期间维度，[2023 年 1 季度]的级别为 1，[2023 年]的级别为 2，[2023 年 1 季度]的级别最低，所以去掉[2023 年 1 季度]，返回[2023 年]。

示例二

```
select
DrillupLevel({[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年]},[期间].[LEVEL0]) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年 1 季度	200.00
2023 年	1200.00

集合中只有期间维度，[2023 年 1 季度]的级别为 1，[2023 年]的级别为 2，指定层级为[期间].[LEVEL0]而目前集合中的成员层级均大于 0 都满足条件，所以返回

[2023 年]和[2023 年 1 季度]。

DrillupMember

遍历第一个集合的成员，如果是第二个集合中的成员，则删除集合一中紧跟着该成员的子代。

语法

DRILLUPMEMBER(set, set)

参数

set

返回集的有效多维表达式（MDX)。

备注

第二个 set 只能包含一个维度

示例一

select

DrillupMember({[期间].[2023 年],[期间].[2023 年].[2023 年 1 季度],[期间].[2023 年].[2023 年 1 季度].[2023 年 1 月]},{[期间].[2023 年].[2023 年 1 季度]}) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50215])

运行结果：

产品: 不分

场景: 累计预算

科目: 工资-管理费用合计

组织: 开发 2 部

期间	年初编报 01 版本
2023 年	1200.00
2023 年 1 季度	200.00

集合 2 中维度成员为[2023 年 1 季度]，集合 1 中[2023 年 1 季度]后面为[2023 年 1 月]刚好是前者的子项，所以被去掉，返回[2023 年]和[2023 年 1 季度]。

Except

遍历第一个集合，如果成员或者元组不存在于第二个集合，加入结果集，默认不保留集合中的重复项，可以选择加上 ALL 参数保留重复项。

语法

EXCEPT(set , set [, [ALL]])

参数

set

返回集的有效多维表达式（MDX）。

ALL

（可选）如果写了参数，则保留重复项。

示例一

select

{Except({{[科目].&[50187].Children},{[科目].&[50188]}},{[科目].&[50189]})} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[期间].&[2023 年 7 月])

运行结果:

产品: 不分

场景: 累计预算

期间: 2023 年 7 月

版本: 年初编报 01 版本

科目	北京总部本部
工资	-10000.00
社保费	6800.00
住房公积金	3500.00
奖金	240.00

[科目]维度 key 为[50187]的成员的子项一共有 5 个，key 分别为[50188]，[50189]，[50190]，[50191]，[50192]

结果集中去除了 key 为[50189]的成员，key 为[50188]的重复项只保留了一个。

示例二

select

{Except({{[科目].&[50187].Children},{[科目].&[50188]}},{[科目].&[50189]},all)} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[期间].&[2023 年 7 月])

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 7 月

版本: 年初编报 01 版本

科目	北京总部本部
工资	-10000.00
福利费	4200.00
社保费	6800.00
住房公积金	3500.00
工资	-10000.00

和示例 1 的结果相比，最后一行多了个重复项【工资】

Extract

查询粒度粗化

语法

EXTRACT(set, hier_name...)

参数

set

返回集的有效多维表达式（MDX)。

hier_name

有 1 至多个，用逗号分隔

示例

select

{Extract(crossjoin({[科目].&[50187].Children},{[期间].&[50648]}),[科目])} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

版本: 年初编报 01 版本

科目	北京总部本部
工资	-10000.00
福利费	4200.00
社保费	6800.00
住房公积金	3500.00
奖金	240.00

行上本来是两个维度（【科目】和【期间】）的成员交叉，使用 extract 函数提取

出指定维度（【科目】）的成员

Filter

返回根据条件筛选指定集的结果集。

语法

FILTER(set, search_condition)

参数

set

返回集的有效多维表达式 (MDX)

search_condition

是逻辑表达式 ， 作为筛选条件

示例

select

{Filter({[期间].[2023 年].[2023 年 3 季度].Children},[版本].&[50672]>0 and [版本].&[50672]<5000)} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

科目: 研究开发费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 8 月	4800.00

筛选条件是大于 0 并且小于 5000，所以筛选出的值为 4800.00

Intersect

语法

INTERSECT(set, set [, [ALL]])

参数

set

返回集的有效多维表达式（MDX）

All

有 ALL 参数则保留重复项

备注

返回两个输入集的交集

示例一

select

Intersect({[期间].&[50640],[期间].&[50640],[期间].&[50642]},{[期间].&[50640],[期间].&[50641],[期间].&[50640]}) on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 1 月	2.00

由于期间维度下第一个集合中 2023 年 1 月、2023 年 2 月值分别为 0、2、；第二个集合中 2023 年 1 月、2023 年 3 月值分别为 2、24，故在 INTERSECT 函数作用下，返回两个集合的交集，返回 20203 年 3 月值为 2.

示例二

```
select
Intersect({[期间].&[50640],[期间].&[50640],[期间].&[50642]},{[期间].&[50640],[期间].&[50641],[期间].&[50640]},ALL) on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 1 月	2.00
20203 年 1 月	2.00

由于期间维度下第一个集合中 2023 年 1 月、2023 年 2 月值分别为 0、2、；第二个集合中 2023 年 1 月、2023 年 3 月值分别为 2、24，故在 INTERSECT 函数作用下，又添加了 ALL，返回两个集合的交集并保留重复项，所有返回 20203 年 1 月值为 2、2。（保留重复项）

Members

返回指定集合范围

语法

level.MEMBERS
hier_name.MEMBERS

参数

hier_name

返回指定集合范围

备注

返回指定级别或者层次结构上的所有成员

示例一

select

{[科目].&[50236]} on columns,

{[期间].[2023 年].level.members}on rows

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[组织].&[50583])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织：研究中心

期间	研究开发费
2021 年	100.00
2022 年	100.00
2023 年	1901.21

由于期间维度下 2021 年、2022 年、2023 年分别为同层级成员，对应结果为 100、100、1901.21，在 MEMBERS 函数下返回了期间维下 2023 年成员同层级的成员，分别为 2021 年、2022 年、2023 年，对应结果为 100、100、1901.21.

示例二

select

{[科目].&[50236]} on columns,

{[期间].levels(2).members}on rows

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[组织].&[50583])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织：研究中心

期间	研究开发费
2021 年	100.00
2022 年	100.00
2023 年	1901.21

由于期间维度下 2021 年、2022 年、2023 年分别为同层级成员均为第 2 层级，2021 年 1、2、3、4 季度，2022 年 1、2、3、4 季度，2023 年 1、2、3、4 季度均为第 1 层级成员，2021 年 1-12 月、2022 年 1-12 月、2023 年 1-12 月均为第 0 层级成员，“[期间].levels(2).members”，取得是期间维下层级为 2 的同层级成员，分别为 2021 年、2022 年、2023 年，对应结果为 100、100、1901.21.

Mtd

返回与给定成员相同级别的一组兄弟成员，从第一个兄弟成员开始，到给定成员结束，这受时间维度中的月份级别的限制。

语法

MTD([member])

参数

Member

是时间类型维度中的成员

备注

月份级别的名字必须是 MONTHS（不区分大小写）

示例一

```
select
{[Organization].&[6]} on columns,
{MTD([Time].&[20101205])} on rows
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Account].&[60])
```

运行结果：

Account: Salaries
DepartmentGroup: Research and Development
Scenario: Actual

Time	Southeast Division
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN

指定成员的月份为 2010 年 12 月 5 日，月份级别的祖先是 2010 年 12 月，第一个和指定成员同级的后代为 20101201，

因为从 20101201 到指定成员 20101205 共五个成员，所以结果有 5 行（显示的名字是成员在当前月的编号）。

示例二

```
with member [Account].[x]    as Sum(MTD([Time].currentmember),[Account].&[60])
select
{[Organization].&[6]} on columns,
{[Account].[x]} on rows
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Time].&[20101229])
```

运行结果：

DepartmentGroup: Research and Development
Scenario: Actual
Time: 29

Account	Southeast Division
x	17108.00

维度 Time 的当前成员为 20101229，所以代入 mtd 函数后返回从 20101201 至 20101229 共 29 个成员，所以结果值为对应的 29 个单元格的值的和。

Order

排列指定集合中的成员，可以选择保留或打乱原有的层次结构。

语法

Order(set, numeric_value [, { ASC | DESC | BASC | BDESC }])

参数

set

返回集的有效多维表达式（MDX），按集合中位置靠后的维度成员进行排序。

numeric_value

返回集的排序依据，优先级高于 set。

备注

Order 有两种变化形式：按层次结构排列（ASC 或 DESC）和不按层次结构排列

（BASC 或 BDESC，其中 B 代表打乱原有层次结构）。按层次结构排列的排序首先按成员在层次结构中的位置对其进行排序，然后再对每个级别进行排序。而不按层次结构排列的排序在对集合中的成员排序时不考虑层次结构。如果没有明确说明，则根据默认设置使用 ASC。

示例一

```
select
Order({[科目].&[50236],[科目].&[50240]}*{[期间].[2023 年].[2023 年 3 季度].Children},[版本].&[50672],desc) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50582],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 北京总部本部

科目	期间	年初编报 01 版本
研究开发费	2023 年 9 月	6800.00
	2023 年 8 月	4200.00
	2023 年 7 月	-10000.00
会务费	2023 年 7 月	-510.89
	2023 年 9 月	-5800.00

	2023 年 8 月	-10000.00

由于期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，在 Order 函数的集合中，期间维度位置靠后，所以会对[研究开发费]的[2023 年 3 季度]的子项的值进行降序排列，顺序为 6800、4200、-10000，对应返回的期间顺序为[2023 年 9 月]、[2023 年 8 月]、[2023 年 7 月];

由于期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[会务费]对应的值分别为-510.89、-10000、-5800，在 Order 函数的集合中，期间维度位置靠后，所以会对[会务费]的[2023 年 3 季度]的子项的值进行降序排列，顺序为-510.89、-5800、-10000，对应返回的期间顺序为[2023 年 7 月]、[2023 年 9 月]、[2023 年 8 月]。

示例二

```
select
Order({[期间].[2023 年].[2023 年 3 季度].Children}*{[科目].&[50236],[科目].&[50240]},[版本].&[50672],bdesc) on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50582],[场景].&[50685],[产品].&[50688])
```

运行结果：

产品: 不分

场景: 累计预算

组织: 北京总部本部

期间	科目	年初编报 01 版本
2023 年 9 月	研究开发费	6800.00
2023 年 8 月	研究开发费	4200.00
2023 年 7 月	会务费	-510.89
2023 年 9 月	会务费	-5800.00
2023 年 7 月	研究开发费	-10000.00
2023 年 8 月	会务费	-10000.00

由于期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，[会务费]对应的值分别为-510.89、-10000、-5800，在 Order 函数中使用了 bdesc 参数，来打破集合中维度成员的层次结构，对数据进行降序排列，所以顺序为 6800、4200、-510.89、-5800、-10000、-10000，对应返回的维度组合顺序为[2023 年 9 月]的[研究开发费]、[2023 年 8 月]的[研究开发费]、[2023 年 7 月]的[会务费]、[2023 年 9 月]的[会务费]、[2023 年 7 月]的[研究开发费]、[2023 年 8 月]的[会务费]。

PeriodsToDate

返回与给定成员相同级别的一组同级成员，从第一个同级成员开始，以给定成员结束。

语法

PERIODSTODATE([level [, member]])

参数

level

只有 level 参数，返回指定成员同一级别从开始到指定成员。

Member

（可选）是时间类型维度中的成员

示例一

select

{PeriodsToDate([期间].[LEVEL0],[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月])}

on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50253])

运行结果：

产品: 不分

场景: 累计预算

科目: 物业外包

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 9 月	11000.00

既指定了 Member 又指定了 level，并且他们在同一级别，返回成员本身。

示例二

select

{PeriodsToDate([期间].[LEVEL1],[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月])}

on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50253])

运行结果:

产品: 不分

场景: 累计预算

科目: 物业外包

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 7 月	81773333.33
2023 年 8 月	8.00
2023 年 9 月	11000.00

既指定了 Member[2023 年 9 月]，它的级别是 0，又指定了 level 级别为 1，并且 Member 的级别小于 level 级别，返回[2023 年 3 季度]第一个同级别子项[2023 年 7 月直到成员本身。

示例三

with member [科目].[x] as Sum(PeriodsToDate([期间].[LEVEL1]),[科目].&[50253])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月])

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 9 月

组织: 北京总部本部

科目	年初编报 01 版本
x	81784341.33

只指定了 level 级别为 1，示例二指定的 Member[2023 年 9 月]放在 where 后面，它的级别是 0 并且 Member 的级别小于 level 级别，返回[2023 年 3 季度]第一个同级别子项[2023 年 7 月直到成员本身，的累加。

Qtd

返回与给定成员来自同一级别的兄弟成员集合，从第一个兄弟成员开始，以给定成员结束，这受 Time 维度中的季度级别的限制。

语法

QTD([member])

参数

Member

是时间类型维度中的成员

备注

季度级别的名字必须为 QUARTER（不区分大小写）

示例一

```
select
{[Organization].&[6]} on columns,
{QTD([Time].&[20130106])} on rows
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Account].&[60])
```

运行结果：

Account: Salaries
DepartmentGroup: Research and Development
Scenario: Actual

Time	Southeast Division
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN

指定成员的月份为 2013 年 1 月 6 日，季度级别的祖先是 2013 年 1 季度，第一个和指定成员同级的后代为 20130101，

因为从 20130101 到指定成员 2010106 共六个成员，所以结果有 6 行（显示的名字是成员在当前月的编号）

示例二

```
with member [Account].[x] as Sum(Qtd([Time].currentmember),[Account].&[60])
select
{[Organization].&[6]} on columns,
{[Account].[x]} on rows
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Time].[2013].[1].[February])
```

运行结果：

DepartmentGroup: Research and Development
Scenario: Actual
Time: February

Account	Southeast Division
x	41016.00

维度 Time 的当前成员为 2013 年 1 季度 2 月，所以代入 qtd 函数后返回从 2013 年 1 季度 1 月至 2013 年 1 季度 2 月共 2 个成员，所以结果值为对应的 2 个单元格的值的和。

TopCount

返回指定集合中指定数目具有较大值的元组组成的集合。

语法

TOPCOUNT(set, index [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

index

指定了返回 tuple 的数目（不超过 index 的值）

numeric_value

（可选）指定了 numeric_value 则返回的同级成员会据此排序（降序）

示例一

select

TopCount({[期间].[2023 年].[2023 年 3 季度].Children},2) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

科目: 研究开发费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 7 月	-10000.00
2023 年 8 月	4200.00

期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，由于指定了 index，所以只返回前两个月[2023 年 7 月]和[2023 年 8 月]。

示例二

select

TopCount({[期间].[2023 年].[2023 年 3 季度].Children},2,[版本].&[50672]) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

科目: 研究开发费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 9 月	6800.00
2023 年 8 月	4200.00

期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，由于指定了 numeric_value 参数，所以[2023 年 3 季度]的子项要按值的大小进行降序排列，结果为 6800，4200，-10000 由于 index 的限制，所以只返回排序后的前两个月[2023 年 9 月]和[2023 年 8 月]。

TopPercent

将指定维度成员范围内数据降序排列，从最大值开始累加，当累加大于或等于全部返回值总和的指定百分比后，返回集合。

语法

TOPPERCENT(set, percentage , numeric_value)

参数

set

返回集的有效多维表达式（MDX）。

percentage

指定了返回元组和在总值中的占比。

numeric_value

返回的同级成员会据此排序（降序）。

备注

百分比不需要输入%

示例一

select

TopPercent({[期间].[2023 年].[2023 年 3 季度].Children},10,[版本].&[50672]) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

科目: 研究开发费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 9 月	6800.00

由于期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，按降序排列结果为 6800、4200、-10000，指定百分比为 10，也就是 6800、4200、-10000 先求和再乘以 10%为 1000，当取到 6800 时，刚好大于 1000，满足大于等于 10000 的条件，所以只需要返回[2023 年 9 月]。

TopSum

将指定维度成员范围内数据降序排列，从最大值开始累加，当累加大于或等于给定的值后，返回集合。

语法

TOMSUM(set, value, numeric_value)

参数

set

返回集的有效多维表达式（MDX）。

Value

指定了返回的元组需要满足的条件。

numeric_value

返回的同级成员会据此排序（降序）。

示例

select

TopSum({[期间].[2023 年].[2023 年 3 季度].Children},1000,[版本].&[50672]) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

科目: 研究开发费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 9 月	6800.00

由于期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]值分别为-10000、4200、6800，降序排列结果为 6800、4200、-10000，指定值为 1000，当取到 6800 时，刚好大于 1000，满足大于等于 1000 的条件，所以返回[2023 年 9 月]。

Union

默认返回两个维度组合的并集，并进行去重，若添加参数 ALL，则可保留并集中的重复项。

语法

UNION(set,set... [, [ALL]])

参数

set

返回集的有效多维表达式（MDX）。

ALL

不加 ALL 参数，剔除所有的重复项，有 ALL 参数，保留所有项。

示例一

select

Union({[期间].[2023 年].[2023 年 3 季度].Children},{[期间].[2023 年].[2023 年 3 季度].[2023 年 7 月]}) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50254])

运行结果：

产品: 不分

场景: 累计预算

科目: 保洁费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 7 月	1.00
2023 年 8 月	2.00
2023 年 9 月	10.00

由于[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[保洁费]的值分别为 1、2、10，集合 1 和集合 2 中[2023 年 7 月]重复，没有参数 ALL，默认去重，所以返回[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]。

示例二

select

Union({[期间].[2023 年].[2023 年 3 季度].Children},{[期间].[2023 年].[2023 年 3 季度].[2023 年 7 月]},ALL) on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[科目].&[50254])

运行结果:

产品: 不分

场景: 累计预算

科目: 保洁费

组织: 北京总部本部

期间	年初编报 01 版本
2023 年 7 月	1.00
2023 年 8 月	2.00
2023 年 9 月	10.00
2023 年 7 月	1.00

由于[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[保洁费]的值分别为 1、2、10，集合 1 和集合 2 中[2023 年 7 月]重复，已设置参数 ALL，保留重复项，所以返回[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]、[2023 年 7 月]。

Wtd

返回与给定成员相同级别的兄弟成员集合，从第一个兄弟成员开始，到给定成员结束，这受时间维度中的周级别的限制。

语法

WTD([member])

参数

Member

是时间类型维度中的成员

备注

周级别的名字必须为 WEEKS（不区分大小写）

示例一

```
select
{[Organization].&[6]} on columns,
{WTD([Time].[2013].[9].[Thursday])} on rows
from [Week AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Account].&[60])
```

运行结果：

Account: Salaries
DepartmentGroup: Research and Development
Scenario: Actual

Time	Southeast Division
Sunday	NaN
Monday	NaN
Tuesday	NaN
Wednesday	NaN
Thursday	21039.00

指定成员的月份为 2013 年第 9 周的 Thursday，所以行上为 Sunday 至 Thursday（第 9 周）

示例二

```
with member [Account].[x]    as Sum(WTD([Time].currentmember),[Account].&[60])
select
{[Organization].&[6]} on columns,
{[Account].[x]} on rows
from [Week AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Time].[2013].[9].[Thursday])
```

运行结果：

DepartmentGroup: Research and Development
Scenario: Actual
Time: Thursday

Account	Southeast Division
x	21039.00

维度 Time 的当前成员为 2013 年第 9 周的 Thursday，所以代入 wtd 函数后返回从 2013 年第 9 周的 Sunday 至 2013 年第 9 周的 Thursday 共 5 个成员（如示例 1 所示），所以结果值为对应的 5 个单元格的值的和。

Ytd

返回与给定成员相同级别的一组兄弟成员，从第一个兄弟成员开始，到给定成员结束，这受时间维度中的 Year 级别的限制。

语法

YTD([member])

参数

Member

是时间类型维度中的成员

备注

季度级别的名字必须为 YEAR（不区分大小写）

示例一

```
select
{[Organization].&[6]} on columns,
{YTD([Time].&[20130106])} on rows
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Account].&[60])
```

运行结果：

Account: Salaries
DepartmentGroup: Research and Development
Scenario: Actual

Time	Southeast Division
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN

指定成员的月份为 2013 年 1 月 6 日，年级别的祖先是 2013 年，第一个和指定成员同级的后代为 20130101，

因为从 20130101 到指定成员 2010106 共六个成员，所以结果有 6 行。

示例二

```
with member [Account].[x] as Sum(Ytd([Time].currentmember),[Account].&[60])
select
{[Organization].&[6]} on columns,
{[Account].[x]} on rows
```

```
from [AdventureWorks]
where ([DepartmentGroup].&[6],[Scenario].&[1],[Time].[2013].[3])
```

运行结果：

```
DepartmentGroup: Research and Development
Scenario: Actual
Time: 3
```

Account	Southeast Division
x	183938.00

维度 Time 的当前成员为 2013 年 3 季度，所以代入 qtd 函数后返回从 2013 年 1 季度至 2013 年 3 季度共 3 个成员，所以结果值为对应的 3 个单元格的值的和

第二章 逻辑函数

And

逻辑运算符 And 和 IIF 函数配合使用（不仅仅可用于 IIF 函数），用多个判断条件的结果来决定最终结果为真还是假

语法

IIF(search_condition1 And search_condition2, true_part, false_part)

参数

search_condition

是逻辑表达式，

And

是连接两个表达式的逻辑运算符，

true_part

是条件为真时的取值，

false_part

是条件为假时的取值

备注

逻辑运算符可以连接两个表达式，两个表达式结果共同决定最终的结果是 True 还是 False。下面是 And 在连接两个条件时的所有情况。

口诀：一假为假

条件 1	条件 2	返回值
true	true	true
true	false	false

false	**true**	**false**
false	**false**	**false**

示例一

with member [期间].[MyMember] as

IIF ([期间].&[50648] < -9999 and [期间].&[50648] > -10001 ,[期间].&[50649] ,[期间].&[50650])

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	4200.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的元组值为-10000.00，小于-9999 且大于-10001，同时满足小于-9999 和大于-10001，因此[期间]维度用的是 key 为 50649 的成员对应的元组的值，即 4200.00

示例二

```
with member [期间].[MyMember] as
IIF ([期间].&[50648] > -9999 and [期间].&[50649] > 4199 ,[期间].&[50649] ,[期间].&[50650])
select
{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	6800.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00

2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的元组值为-10000.00，大于-9999，[期间]维度上 key 为 50649 的成员对应的元组值为 4200.00，大于 4199.00，[期间]维度上 key 为 50648 的成员对应的元组值不满足大于-9999 的条件，因此[期间]维度用的是 key 为 50650 的成员对应的元组的值，即 6800.00

Case

返回不同情况中第一个满足逻辑表达式（case_eperand）对应的值（result）。

语法一

CASE case_eperand
WHEN when_operand THEN result
WHEN when_operand THEN result.....
ELSE result END

参数一

case_eperand

整个函数的逻辑表达式

WHEN when_operand THEN result

表达式为 0 至多个，代表不同情况；

when_operand

是当前情况下的值，

Result

是符合当前情况下的取值；

ELSE 情况可以省略

示例一

```
with member  [期间].[x]  as  Case  2*3
WHEN 1 THEN 1
WHEN 2 THEN 2
WHEN 6 THEN 6
ELSE 100
END
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	6.00

由于逻辑表达式的值为 6，第三种情况刚好符合逻辑表达式的值，所以返回第三种情况对应的取值 6。

语法二

```
CASE
WHEN search_condition THEN result
WHEN search_condition THEN result
WHEN search_condition THEN result...
```

ELSE result END

参数二

WHEN search_condition THEN result

表达式为 0 至多个，代表不同情况；

search_condition

是当前情况下的逻辑表达式，

Result

是符合当前情况下逻辑表达式为真的取值；

ELSE

情况可以省略

示例二

```
with member    [期间].[x]    as    Case

WHEN count([期间].[2023 年].Children)>10 THEN 10

WHEN count([期间].[2023 年].Children)>2 THEN 2

WHEN count([期间].[2023 年].Children)>3 THEN 3

ELSE 100
END
select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	2.00

由于[2023 年]的子项有 4 个，所以第二种情况是第一个满足条件的，返回情况二 THEN 后面的值 2。

EQ : '='

执行比较运算，以确定一个多维表达式（MDX）的值是否等于另一个 MDX 表达式的值。

语法

MDX_Expression = MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

```
with member [期间].[MyMember] as

IIF ([期间].&[50648] = -10000 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，等于-10000，条件成立，返回[期间].&[50648]的值。

GE : '>='

执行比较操作，确定一个多维表达式（MDX）表达式的值是否大于或等于另一个 MDX 表达式的值。

语法

MDX_Expression >= MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

with member [期间].[MyMember] as

IIF ([期间].&[50648] >= -10002 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，大于等于-10002，条件成立，返回[期间].&[50648]的值。

GT : '>'

执行比较运算，以确定一个多维表达式 (MDX) 的值是否大于另一个 MDX 表达式的值。

语法

MDX_Expression > MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

with member [期间].[MyMember] as

IIF ([期间].&[50648] > -10001 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，大于-10001，条件成立，返回[期间].&[50648]的值。

IIf

此函数先判断条件是否成立，条件成立返回第一个值，不成立返回第二个值。

语法

IIF(search_condition, true_part, false_part)

参数

search_condition

逻辑表达式

true_part

条件为真时的取值

false_part

是条件为假时的取值

示例

```
with member [期间].[MyMember] as

IIF ([期间].&[50648] > -9999 ,[期间].&[50649] ,[期间].&[50650])

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	6800.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的单元格值为-10000.00，小于-9999，不满足>-9999 的条件，因此[期间]维度用的是 key 为 50650 的成员对应的元组的值，即 6800.00

IsEmpty

语法

ISEMPTY(value)

备注

判断单元格值是否为空，是为 true，否为 false.

示例一

```
with member  [科目].[x]  as
IIF(NOT(IsEmpty([科目].[管理费用合计].[工资-管理费用合计])),[科目].[管理费用合计].[工资-管理费用合计],[期间].[2023 年].[2023 年 1 季度].[2023 年 2 月])
select
{[科目].[管理费用合计].[工资-管理费用合计]} on rows,
{[版本].&[50672]} on columns
```

from [模型一]

where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年 1 季度].[2023 年 2 月])

运行结果:

产品: 不分

场景: 累计预算

期间: 2021 年 1 月

组织：北京总部

科目	年初编报 01 版本
工资-管理费用合计	100.00

科目维度上[科目].[管理费用合计].[工资-管理费用合计]的成员对应的单元格值为 100.00，不为空返回 false，NOT 函数取反后返回 true，则 IIF 函数的条件为真，取第一个表达式的值，所以展示的是[科目].[管理费用合计].[工资-管理费用合计]成员对应的值 100.00.

LE：'<='

执行比较运算，以确定一个多维表达式（MDX）的值是否小于等于另一个 MDX 表达式的值。

语法

MDX_Expression <= MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

```
with member [期间].[MyMember] as

IIF ([期间].&[50648] <= -1000 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，小于等于-1000，条件成立，返回[期间].&[50648]的值。

LT : '<'

执行比较操作，确定一个多维表达式（MDX）表达式的值是否小于另一个 MDX 表达式的值

语法

MDX_Expression < MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

with member [期间].[MyMember] as

IIF ([期间].&[50648] < -100 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部

MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，小于-100，条件成立，返回[期间].&[50648]的值。

NE : '<>'

执行比较运算，以确定一个多维表达式（MDX）的值是否不等于另一个 MDX 表达式的值。

语法

MDX_Expression <> MDX_Expression

参数

MDX_Expression

有效的 MDX 表达式。

示例

with member [期间].[MyMember] as

IIF ([期间].&[50648] < > -100 , [期间].&[50648],null)

select

{[期间].[MyMember],[期间].&[50648]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	-10000.00
2023 年 7 月	-10000.00

[期间].&[50648]的值为-10000.00，不等于-100，条件成立，返回[期间].&[50648]的值。

Not

逻辑运算符非可以将原表达式取反

语法

NOT(value)

备注

条件 返回值

false true
true false

示例

with member [期间].[MyMember] as

IIF (NOT([期间].&[50648] > -9999) ,[期间].&[50649] ,[期间].&[50650])

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
MyMember	4200.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的单元格值为-10000.00，小于-9999，不满足>-9999 的条件，因此[期间]维度用的是 key 为 50650 的成员对应的元组的值，即 6800.00，调用 Not 函数取反，因此[期间]维度用的是 key 为 50650 的成员对应的元组的值，即 4200.

Or

逻辑运算符 Or 和 IIF 函数配合使用（不仅仅可用于 IIF 函数），用多个判断条件的结果来决定最终结果为真还是假

语法

IIF(search_condition1 Or search_condition2, true_part, false_part)

参数

search_condition 是逻辑表达式

or 是连接两个表达式的逻辑运算符

true_part 是条件为真时的取值

false_part 是条件为假时的取值

备注

逻辑运算符可以连接两个表达式，两个表达式结果共同决定最终的结果是 True 还是 False。下面是 Or 在连接两个条件时的所有情况。

口诀：一假为假

条件 1	条件 2	返回值
true	true	true
true	false	true
false	true	true
false	false	false

示例 1

```
with member [期间].[MyMember] as
IIF ([期间].&[50648] < -9999 or [期间].&[50648] > -10001 ,[期间].&[50649] ,[期间].&[50650])
select
{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,
```

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	4200.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的元组值为-10000.00，小于-9999 或大于-10001，同时满足小于-9999 和大于-10001，因此[期间]维度用的是 key 为 50649 的成员对应的元组的值，即 4200.00

示例 2

with member [期间].[MyMember] as

IIF ([期间].&[50648] > -9999 or [期间].&[50649] > 4199 ,[期间].&[50649] ,[期间].&[50650])

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	4200.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的元组值为-10000.00，大于-9999，[期间]维度上 key 为 50649 的成员对应的元组值为 4200.00，大于 4199.00，[期间]维度上 key 为 50648 的成员对应的元组值不满足大于-9999 的条件，因此[期间]维度用的是 key 为 50648 的成员对应的元组的值，即 4200.00

示例 3

with member [期间].[MyMember] as

IIF ([期间].&[50648] > -9999 or [期间].&[50649] < 4199 ,[期间].&[50649] ,[期

间].&[50650])

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	6800.00
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度上 key 为 50648 的成员对应的元组值为-10000.00，大于-9999，[期间]维度上 key 为 50649 的成员对应的元组值为 4200.00，不满足小于 4199.00，[期间]维度上 key 为 50648 的成员对应的元组值不满足大于-9999 的条件，因此[期间]维度用的是 key 为 50650 的成员对应的元组的值，即 6800.00

第三章 数值函数

Abs

返回数据的绝对值

语法

Abs（numeric）或者 Abs（tuple）

参数

数值或者元组

数学表达式

$y = |x|$

例:

①X 为负数时，y=-x。如：x=-5 时，y=5

②X 为非负数时，y=x。如：x=5 时，y=5；x=0 时，y=0

备注

查询语句中只能在 with member 后使用

示例一

```
with member  [科目].[x]  as  Abs(-600)
select
{[科目].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50582],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年
```

1 季度].[2023 年 3 月])

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 3 月

组织: 北京总部本部

科目	年初编报 01 版本
x	600.00

取-600 的绝对值给计算维度成员[x]，返回结果为 600。

示例二

```
with member [科目].[x] as Abs([科目].[管理费用合计].[工资-管理费用合计])
select
{[科目].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年
1 季度].[2023 年 2 月])
```

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 2 月

组织: 开发 2 部

科目	年初编报 01 版本
x	100.00

[工资-管理费用合计]的值为-100，经过 Abs 函数取绝对值后，[x]的结果为 100。

示例三

with member [科目].[x] as Abs([科目].[管理费用合计].[工会经费]-[科目].[管理费用合计].[职工教育经费])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年 1 季度].[2023 年 2 月])

运行结果:

产品: 不分

场景: 累计预算

期间: 2023 年 2 月

组织: 开发 2 部

科目	年初编报 01 版本
x	6.00

由于[2023 年 2 月]的[工会经费]为 2， [职工教育经费]为 8，2 减去 8 等于-6，再

取绝对值，所以[x]的结果为 6。

Aggregate

返回一个数字，该数字是通过对集合表达式返回的单元格进行聚合来计算的。如果没有提供数值表达式，此函数将使用为每个度量值指定的默认聚合操作符在当前查询上下文中聚合每个度量值。如果提供了数值表达式，此函数将首先计算指定集合中每个单元格的数值表达式，然后对其求和。

语法

AGGREGATE(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

（可选）如果指定了 numeric_value，此函数将根据度量维度自带的聚合符做聚合；如果不指定，默认进行累加。

数学表达式（以聚合符为+时为例）

$$y = \sum_{i=1}^{n} x_i$$

例：若干个单元格的值为 7，15，9，12，则

i=1 时，x=7；

i=2 时，x=15；

i=3 时，x=9；

i=4 时，x=12；

y = 7+15+9+12 = 43

备注

当前度量的聚合方式支持+、-、*、/、%

示例一

with member [科目].[x] as Aggregate({[期间].[2023 年].[2023 年 3 季度].Children},[科目].&[50221])
select
{[科目].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	24.00

期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50221]值分别为 7、8、9，numeric_value 的聚合方式为+，所以[x]的值为他们的求和，是 24。

示例二

with member [科目].[x] as Aggregate({[期间].[2023 年].[2023 年 3 季度].Children}*{[科目].&[50221]})

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果:

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	24.00

期间维度[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50221]值分别为 7、8、9，未指定 numeric_value 默认进行累加，所以[x]的值为 24。

Avg

计算集合并返回集合中单元格的非空值的平均值，即对集合中的度量值或对指定度量值（numeric_value）的平均值。

语法

AVG(set [, numeric_value])

参数

set

返回集的有效多维表达式 (MDX)。

numeric_value

（可选）如果指定了 numeric_value，对集合中的度量值或对指定度量值（numeric_value）的平均值，若未指定，只对集合中的非空元组计算平均值。

数学表达式

$y = \bar{x}$

例：若干个单元格的值为 7，15，9，13，则

i=1 时，x=7；

i=2 时，x=15；

i=3 时，x=9；

i=4 时，x=14；

样本数目 n = 4;

y = (7+15+9+12) ÷ 4 = 11

示例一

```
with member [期间].[平均值 x] as avg({[期间].&[50648],[期间].&[50649],[期间].&[50650]})
select
{[期间].[平均值 x]} on rows,
```

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
平均值 x	333.33

【期间】维度的三个成员对应的单元格值-10000.00、4200.00、6800.00，所以求均值后为 333.33

示例二

with member [期间].[平均值 x] as avg({[期间].&[50648],[期间].&[50649],[期间].&[50650]},[版本].&[50672])

select

{[期间].[平均值 x]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
平均值 x	333.33

【期间】维度的三个成员对应的单元格值-10000.00、4200.00、6800.00，所以求均值后为 333.33

CoalesceEmpty

如果指定了一个或多个表达式，CoalesceEmpty 函数将返回第一个表达式(从左到右)的数值，该数值可以解析为非空值。如果指定的表达式都不能解析为非空值，则函数返回空单元格值。通常，第二个表达式的值是替换第一个表达式为空时的值。

语法

CoalesceEmpty (value_expression (, value_expression)+)

参数

Value_expression

至少有两个，以逗号分隔，可以是数值类型的，也可以是字符串类型的

备注

Value_expresssio 对应的数值的类型必须一致，要么都是数值类型的，要么都是字符串类型的。

示例以数值类型的为例

示例一

```
with member [期间].[x] as CoalesceEmpty([期间].[2022 年].[2022 年 2 季度].[2022 年 4 月],0.0)
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
x	0.00

由于[2022 年 4 月]的值为空，从左到右取值，0.0 为非空的值，所以返回 0.0。

示例二

with member [期间].[x] as CoalesceEmpty([期间].[2022 年].[2022 年 2 季度].[2022 年 4 月],[期间].[2022 年].[2022 年 2 季度].[2022 年 5 月],0.0)

select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
x	0.00

由于[2022 年 4 月]、[2022 年 5 月]的值均为空，从左到右取值，0.0 为非空的值，所以返回 0.0。

示例三

with member 期间].[x] as CoalesceEmpty([期间].[2022 年].[2022 年 2 季度].[2022 年 4 月],[期间].[2022 年].[2022 年 2 季度].[2022 年 5 月])

select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
x	NaN

由于从左到右取的值均为空，所以返回 NaN。

示例四

with member [期间].[x] as CoalesceEmpty([期间].[2023 年].[2023 年 2 季度].[2023 年 4 月],[期间].[2022 年].[2022 年 2 季度].[2022 年 5 月])

select

{[期间].[x],[期间].[2023 年].[2023 年 2 季度].[2023 年 4 月]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
x	7.00
2023 年 4 月	7.00

由于从左到右取值，第一个[2023 年 4 月]的值为 7，所以返回 7。

Correlation

返回在集合上计算的 x-y 值的相关系数

语法

CORRELATION(set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式 (MDX)。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，不指定 x 轴的值，使用指定集合中的单元格的当前上下文作为 x 轴的值。

数学表达式

$$Correl(X,Y)=\frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

求和符号省略了上下标

例:

4 个样本点（2，4），（3，7），（4，8），（5，9）

i	x	y
1	2	4
2	3	7
3	4	8
4	5	9
Avg	$\bar{x} = 3.5$	$\bar{y} = 7$

代入公式中，求得结果值为 0.96

示例一

with member [科目].[x] as Correlation({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221],[科目].&[50220])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果:

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本

x	0.96

[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），相关系数计算结果为 0.96，和 excel 中结果相同。

示例二

```
with member [期间].[x] as Correlation({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	0.96

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后

边的切片处，[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），相关系数计算结果为 0.96，和 excel 中结果相同。

示例三

```
with member [期间].[x] as Correlation({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	0.96

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科

目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），相关系数计算结果为 0.96，和 excel 中结果相同。

Count

返回集合中单元格的数量，默认包括空单元格。有 INCLUDEEMPTY 参数则计数时包括空单元格，有 EXCLUDEEMPTY 参数则计数时不包括空单元格，

语法

COUNT(set [, [INCLUDEEMPTY] [, EXCLUDEEMPTY]])

参数

set

返回集的有效多维表达式 (MDX)。

INCLUDEEMPTY

（可选）指定参数后，效果和默认一样，包含空单元格。

EXCLUDEEMPTY

（可选）指定参数后，不包含空单元格。

数学表达式

无

示例一

with member [科目].[x] as Count({[期间].[2023 年].Children}*{[科目].&[50220]})

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	4.00

由于[2023 年]的子项共有 4 个，默认统计空单元格，所以返回结果为 4。

示例二

with member [期间].[x] as Count([期间].[2023 年].Children,INCLUDEEMPTY)

select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	4.00

指定了 INCLUDEEMPTY 参数，效果和示例一一样，[2023 年]的子项共有 4 个（统计空单元格），所以返回结果为 4。

示例三

```
with member [期间].[x] as Count([期间].[2023 年].Children,EXCLUDEEMPTY)
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果:

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	1.00

指定了 EXCLUDEEMPTY 参数，[2023 年]的子项共有 4 个，3 个单元格为空，所以

返回结果为 1。

Covariance

返回在集合上计算的 x-y 值构成的样本的协方差，不指定 x 轴的值，函数将 y 轴的值作为 x 轴的值参与计算。

语法

COVARIANCE(set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，不指定 x 轴的值，函数将当前上下文中的度量维的成员作为 x 轴。

数学表达式

$$Cov(X,Y)=\frac{\sum(x-\bar{x})(y-\bar{y})}{n}$$

求和符号省略了上下标

例:

样本点（1，14.6），（4，2），（2，1），（63.6，7），（7，2），（1，1）

i	x	y
1	1	14.6
2	4	2
3	2	1

4	63.6	7
5	7	2
6	1	1
Avg	$\bar{x} = 13.1$	$\bar{y} = 4.6$

代入公式中，求得结果值为 20.54

示例一

with member [科目].[x] as Covariance({[期间].[2023 年].[2023 年 1 季度].[2023 年 1 月]:[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]},[科目].&[50220],[科目].&[50221])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 1 部

科目	年初编报 01 版本
x	20.54

由于期间维度[2023 年 1 月]到[2023 年 6 月]的[科目].&[50220]值为 14.6、2、1、7、2、1，[科目].&[50221]值为 1、4、2、63.6、7、1，样本点为（1，14.6），（4，

2），（2，1），（63.6，7），（7，2），（1，1）所以临时变量[x]的值为 20.54，和 excel 中结果一样。

示例二

```
with member [期间].[x] as Covariance({[期间].[2023 年].[2023 年 1 季度].[2023 年 1 月]:[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]},[科目].&[50220])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50221])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 差旅费

组织: 开发 1 部

期间	年初编报 01 版本
x	20.54

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，由于期间维度[2023 年 1 月]到[2023 年 6 月]的[科目].&[50220]值为 14.6、2、1、7、2、1，[科目].&[50221]值为 1、4、2、63.6、7、1，样本点为（1，

14.6），（4，2），（2，1），（63.6，7），（7，2），（1，1）所以临时变量[x]的值为 20.54，和 excel 中结果一样。

LinRegIntercept

计算集合的线性回归并返回回归线的 x 截距值，即 b 的值，y = ax + b。

语法

LINREGINTERCEPT(set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，如果没有指定第二个数值表达式，函数将使用指定集合中单元格的当前上下文作为 x 轴的值。

数学表达式

回归线 $\hat{y} = a + bx$

回归线 a 的截距公式为：$a = \bar{y} - b\bar{x}$

公式中斜率 b 计算如下：

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$$

求和符号省略了上下标

例:

样本点（2，4），（3，7），（4，8），（5，9）

i	x	y
1	2	4
2	3	7
3	4	8
4	5	9
Avg	$\bar{x} = 3.5$	$\bar{y} = 7$

代入公式中，求得 a 的值为 1.4

示例一

with member [科目].[x] as LinRegIntercept({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221],[科目].&[50220])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果:

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	1.40

由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9）所以[x]的值为 1.4，和 excel 中结果相同。

示例二

```
with member [期间].[x] as LinRegIntercept({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	1.40

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、

9，样本点为（2，4），（3，7），（4，8），（5，9）所以[x]的值为 1.4，和 excel 中结果相同。

LinRegR2

计算维度组合的线性回归并返回决定系数 R^2

语法

LINREGR2 (set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，如果没有指定第二个数值表达式，函数将使用指定集合中单元格的当前上下文作为 x 轴的值。

数学表达式

皮尔生(Pearson)乘积矩相关系数 r 的计算公式如下:

$$r=\frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

例:

样本点（4，2），（7，3），（8，4），（9，5）

i	x	y
1	4	2
2	7	3

3	8	4
4	9	5
Avg	$\bar{x} = 7$	$\bar{y} = 3.5$

代入公式中，求得 a 的值为 0.91

示例一

with member [科目].[x] as LinRegR2({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50220],[科目].&[50221])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	0.91

由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（4，2），（7，3），（8，4），（9，5）所以[x]的值为 0.91，和 excel 中结果相同。

示例二

with member [期间].[x] as LinRegR2({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50220])

select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50221])

运行结果:

产品: 不分

场景: 累计预算

科目: 差旅费

组织: 开发 2 部

期间	年初编报 01 版本
x	0.91

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（4，2），（7，3），（8，4），（9，5）所以[x]的值为 0.91，和 excel 中结果相同。

LinRegSlope

计算集合的线性回归，并返回回归线上斜率的值，即 a 的值，y = ax + b。

语法

LINREGSLOPE(set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，如果没有指定第二个数值表达式，函数将使用指定集合中单元格的当前上下文作为 x 轴的值。

数学表达式

回归线 $\hat{y} = a + bx$

公式中斜率 b 计算如下：

$$b = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2}$$

求和符号省略了上下标

例：

样本点（2，4），（3，7），（4，8），（5，9）

i	x	y
1	2	4
2	3	7
3	4	8
4	5	9

Avg	$\bar{x} = 3.5$	$\bar{y} = 7$

代入公式中，求得 b 的值为 1.6

示例一

with member [科目].[x] as LinRegSlope({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221],[科目].&[50220])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

期间: 期间不分

组织: 开发 2 部

科目	年初编报 01 版本
x	1.60

由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），所以[x]的值为 1.6，和 excel 中结果相同。

示例二

with member [期间].[x] as LinRegSlope({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 2 部

期间	年初编报 01 版本
x	1.60

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），所以[x]的值为 1.6，和 excel 中结果相同。

LinRegVariance

计算维度组合的线性回归并返回与回归线 y = ax + b 相关的方差 SSR

语法

LINREGVARIANCE (set, numeric_value [, numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

numeric_value

第一个代表 y 轴，

numeric_value

（可选）第二个代表 x 轴，如果没有指定第二个数值表达式，函数将使用指定集合中单元格的当前上下文作为 x 轴的值。

数学表达式

回归线 $\hat{y} = a + bx$

回归线 a 的截距公式为：$a = \bar{y} - b\bar{x}$

公式中斜率 b 计算如下：

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$$

SSR 计算公式如下(预测值和真实值之间差的平方和)：

$SSR = \Sigma(\hat{y} - \bar{y})^2$

注：求和符号都省略了上下标

例：

样本点（3，1），（4，2）（7，3），（8，4），（9，5）

i	x	y

1	3	1
2	4	2
3	7	3
4	8	4
5	9	5
Avg	$\bar{x} = 6.2$	$\bar{y} = 5$

计算出回归方程后计算预测值如下：

观测值	预测 Y
1	1.08955
2	1.68657
3	3.47761
4	4.07463
5	4.67164

代入 SSR 计算公式后，求得 SSR 的值为 9.55224

示例一

with member [期间].[x] as LinRegVariance({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221],[科目].&[50220])

select

{[期间].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

科目: 科目不分

组织: 开发 2 部

期间	年初编报 01 版本
x	12.80

由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），所以[x]的值为 12.8，和 excel 中结果相同。

示例二

```
with member [期间].[x] as LinRegVariance({[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]},[科目].&[50221])
select
{[期间].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50220])
```

运行结果:

产品: 不分

场景: 累计预算

科目: 科目不分

组织: 开发 2 部

期间	年初编报 01 版本
x	12.80

不指定第二个 numeric_value，示例一指定的第二个 numeric_value 放在 where 后边的切片处，由于[2023 年 6 月]、[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[科目].&[50220]的值分别为 2、3、4、5，[科目].&[50221]的值分别为 4、7、8、9，样本点为（2，4），（3，7），（4，8），（5，9），所以[x]的值为 12.8，和 excel 中结果相同。

Max

返回在集合上求值的数值表达式的最大值。

语法

MAX(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式（MDX)。

数学表达式

例：若干个单元格的值为 7，15，9，13，则最大值为 15

示例一

with member [期间].[Max] as Max({[期间].[2023 年].[2023 年 3 季度].Children},[版本].&[50672])

select

{[期间].[Max]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
Max	6800.00

由于[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]对应的[科目].&[50236]的值分别为-10000、4200、6800，其中最大值为 6800。

示例二

with member [期间].[Max] as Max({[期间].[2023年].[2023年3季度].Children})
select
{[期间].[Max]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
Max	6800.00

虽然没指定 numeric_value，但是计算上下文和示例 1 是相同的（numeric_value 移到了 where 后边的切片），所以结果也是相同的，最大值为 6800.0

Median

返回在集合上求值的数值表达式的中位数。

语法

MEDIAN(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式（MDX）。

数学表达式

例：若干个单元格的值为 7，15，9，13，则中位数为（9 + 13）÷ 2 = 11

示例

with member [期间].[Median] as MEDIAN({[期间].[2023 年].[2023 年 3 季

度].Children},[版本].&[50672])

select

{[期间].[Median]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
Median	4200.00

由于[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]对应的

[科目].&[50236]的值分别为-10000、4200、6800，其中中位数可以找到为 4200.0

Min

返回在集合上求值的数值表达式的最小值。

语法

MIN(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式（MDX)。

数学表达式

例：若干个单元格的值为 7，15，9，13，则最小值为 7

示例

with member [期间].[Min] as MIN({[期间].[2023 年].[2023 年 3 季度].Children},[版本].&[50672])

select

{[期间].[Min]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
Min	-10000.00

由于[2023 年 3 季度]的子项[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]对应的

[科目].&[50236]的值分别为-10000、4200、6800，其中最小值可以找到为-10000.0。

Rank

默认返回维度组合中，某一个元组的位置（次序），若在最后添加度量维度（维度成员），则先按照降序排列，再返回元组的位置（等级）

语法

RANK(tuple, set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX）。

numeric_value

（可选）不指定的话，返回 tuple 在 set 中的次序，不存在返回 0；指定 numeric_value，返回 tuple 的秩，即按照 numeric_value 求值后降序排列，秩依次为 1、2、3...；如果有 set 中有重复项，秩表示例：1、2、2、4、4、4、7、8...

数学表达式

无

备注

若集合中多个元组的值相等，则位置顺序一样

示例 1

```
with member [期间].[MyMember] as
rank (([期间].&[50641]),{[期间].[2023 年].[2023 年 1 季度].Children})
select
{[期间].[MyMember],[期间].&[50640],[期间].&[50641],[期间].&[50642]} on rows,
```

{[科目].&[50215]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[组织].&[50585])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织: 开发 2 部

期间	工资-管理费用合计
MyMember	2.00
2023 年 1 月	200.00
2023 年 2 月	-100.00
2023 年 3 月	100.00

期间成员[2023 年 2 月]位于集合[2023 年 1 季度]中的第二个位置，所以结果为 2

示例 2

with member [期间].[MyMember] as

rank (([期间].&[50641]),{[期间].[2023 年].[2023 年 1 季度].Children},[组织].&[50585])

select

{[期间].[MyMember],[期间].&[50640],[期间].&[50641],[期间].&[50642]} on rows,

{[科目].&[50215]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[组织].&[50585])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织: 开发 2 部

期间	工资-管理费用合计
MyMember	3.00
2023 年 1 月	200.00
2023 年 2 月	-100.00
2023 年 3 月	100.00

在期间维度[2023 年 1 季度]集合中，[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的值为 200、-100、100，维度成员[2023 年 2 月]在集合[2023 年 1 季度]中按降序排列为 3，所以所以结果为 3

Stdev

财务函数，返回数值表达式在集合上计算的样本标准差，使用无偏总体公式(除以 n-1 的那个公式)。

语法

STDEV(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式（MDX)。

数学表达式

无偏的样本标准差计算公式如下：

$$\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$$

求和符号省略了上下标

例： 若干个单元格的值为-10000，4200，6800

i=1 时，x=-10000；

i=2 时，x=4200；

i=3 时，x=6800；

$\bar{x} = 333.334$

代入公式，结果值为 9042.861

示例 1

```
with member [期间].[MyMember] as
stdev ({[期间].[2023 年].[2023 年 3 季度].Children})
select
{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	9042.86
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度的[2023 年 3 季度]的成员的子代有 3 个，所以样本值有 3 个，标准差

维 9042.86

示例 2

```
with member [期间].[MyMember] as
stdev ({[期间].[2023 年].[2023 年 3 季度].Children},[科目].&[50236])
select
{[期间].[MyMember]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	9042.86
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

[期间]维度的[2023 年 3 季度]的成员的子代有 3 个，所以样本值有 3 个，标准差维 9042.86

示例 1 中 where 切片中没有指定科目，但在 numeric_value 中指定了[科目].&[50236]，所以示例 2 和示例 1 的计算上下文是相同的。所以结果也是一样的

Sum

此函数返回维度组合区域元组值的和

语法

SUM(set [, numeric_value])

参数

set

返回集的有效多维表达式（MDX)。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式（MDX)。

数学表达式

例：若干个单元格的值为 7，15，9，13，则和为 7+15+9+13=44

示例 1

```
with member [期间].[MyMember] as
stdev ({[期间].[2023 年].[2023 年 3 季度].Children},[科目].&[50236])
select
{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	26.00
2023 年 1 月	2.00
2023 年 2 月	0.00
2023 年 3 月	24.00

由于[2023 年 1 季度]的子项中[2023 年 1 月]、[2023 年 2 月]、[2023 年 3 月]的汇

总值为 26

示例 2

with member [期间].[MyMember] as

sum ({[期间].[2023 年].[2023 年 1 季度].Children})

select

{[期间].[MyMember],[期间].&[50640],[期间].&[50641],[期间].&[50642]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	26.00
2023 年 1 月	2.00
2023 年 2 月	0.00
2023 年 3 月	24.00

示例 1 指定的 numeric_value（[科目].&[50236]）移到了 where 后边的切片中指定，所以两个示例的计算上下文是一致的，结果同样为 26

Var

财务函数，返回数值表达式（numeric_value）在集合上计算的样本方差，使用无偏总体公式(除以 n-1 的那个公式)。

语法

VAR(set [, numeric_value])

参数

set

返回集的有效多维表达式 (MDX)。

numeric_value

返回数字的有效数值表达式，通常为单元坐标的多维表达式 (MDX)。

数学表达式

无偏样本方差计算公式如下：

$$\frac{\sum(x-\bar{x})^2}{(n-1)}$$

求和符号省略了上下标

例： 若干个单元格的值为-10000，4200，6800

i=1 时，x=-10000；

i=2 时，x=4200；

i=3 时，x=6800；

$\bar{x}$=333.334

代入公式，结果值为 81773333.33

示例 1

with member [期间].[MyMember] as

```
var ({[期间].[2023 年].[2023 年 3 季度].Children},[科目].&[50236])
select
{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	81773333.33
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

由于[2023 年 3 季度]的子项中[2023 年 7 月]、[2023 年 8 月]、[2023 年 9 月]的[研究开发费]的值分别为-10000、4200、6800，这三个值计算的方差为 81773333.33，和 excel 中同名函数值相同。

示例 2

```
with member [期间].[MyMember] as
```

var ({[期间].[2023 年].[2023 年 3 季度].Children})

select

{[期间].[MyMember],[期间].&[50648],[期间].&[50649],[期间].&[50650]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
MyMember	81773333.33
2023 年 7 月	-10000.00
2023 年 8 月	4200.00
2023 年 9 月	6800.00

示例 1 指定的 numeric_value 由函数处移到了 where 后边的切片中，因此和示例的计算上下文相同，结果也是一致

第四章 成员函数

ClosingPeriod

返回指定成员下指定级别的最后一个维度成员

语法

CLOSINGPERIOD([level [, member]])

参数

level

只有 level 参数，返回指定级别上最后一个同级的成员。

member

指定了 level 和 member，则返回在指定级别上的指定成员的后代成员的最后一个同级

备注

如果没有任何参数，则使用时间类型的维度的默认 level 和默认成员（gmm 返回的是时间类型的维度上除维度成员所在级别外，最高 level 的最后一个成员）。

示例一

```
select
{ClosingPeriod([期间].[LEVEL0])} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 12 月	-10000.00

【期间】维度上 0 级别成员最后一个为【2023 年 12 月】（如行上所示）

示例二

```
select
{ClosingPeriod([期间].[LEVEL0],[期间].&[50651])} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 12 月	-10000.00

指定了的成员是【期间】维度 key 为 50651 的成员【2023 年 4 季度】，此成员的子项均为 0 级别，最后一个为【2023 年 12 月】

示例三

select

{ClosingPeriod([期间].[LEVEL0],[期间].&[50651])} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品：不分

场景：累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年	-28746.00

时间类型的维度为 Time，年份所在级别是除维度成员 Time 之外的最高级别，年份所在级别的最后一个成员是 2023 年

CurrentMember

在迭代期间沿指定层次结构返回当前成员

语法

hierarchy.CURRENTMEMBER

identifier. CURRENTMEMBER

参数

hier_name

返回指定集合范围

示例

with member [科目].[x] as ([科目].&[50220],[期间].CurrentMember)

select

{[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月]:[期间].[2023 年].[2023 年 3 季度].[2023 年 9 月]} on rows,

{[科目].[x]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

组织: 开发 2 部

期间	x
2023 年 6 月	2.00
2023 年 7 月	3.00
2023 年 8 月	4.00
2023 年 9 月	5.00

由于行上期间范围为[2023 年 6 月]到[2023 年 9 月]，所以按照当前期间范围，返回期间上的成员和对应的值。

FirstChild

返回指定父项的第一个子项

语法

member.FIRSTCHILD
FIRSTCHILD(member)

参数

member

指定成员

示例一

select

{FirstChild([期间].[2023 年].[2023 年 2 季度])} on rows,

{[组织].&[50584]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50253])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 物业外包

期间	开发 1 部
2023 年 4 月	50.00

由于[2023 年 2 季度]的第一个子项为[2023 年 4 月]，所以返回[2023 年 4 月]和对应的值。

示例一

select

{[期间].[2023 年].[2023 年 2 季度].FirstChild} on rows,

{[组织].&[50584]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50253])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 物业外包

期间	开发 1 部
2023 年 4 月	50.00

由于[2023 年 2 季度]的第一个子项为[2023 年 4 月]，所以返回[2023 年 4 月]和对应的值。

FirstSibling

返回指定成员同一父项下的第一个同级

语法

member.FIRSTCHILD
FIRSTCHILD(member)

参数

member

指定成员

示例一

select

{FirstSibling([期间].[2023 年].[2023 年 2 季度].[2023 年 6 月])} on rows,

{[组织].&[50584]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50253])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 物业外包

期间	开发 1 部
2023 年 4 月	50.00

由于[2023 年 6 月]一父项（[2023 年 2 季度]）下的第一个同级为[2023 年 4 月],

且[2023 年 4 月]的[物业外包]的值为 50，所以返回[2023 年 4 月]和对应的值。

示例二

select

{[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月].FirstSibling} on rows,

{[组织].&[50584]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50253])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 物业外包

期间	开发 1 部
2023 年 4 月	50.00

由于[2023 年 6 月]一父项（[2023 年 2 季度]）下的第一个同级为[2023 年 4 月]，

且[2023 年 4 月]的[物业外包]的值为 50，所以返回[2023 年 4 月]和对应的值。

Lag

返回当前维度成员前面或后面的第几个维度成员，正数代表前，负数代表后。

语法

member.LAG(index)

参数

index

指定成员相对于指定成员的偏移量

示例一

select

{[期间].[2023 年].[2023 年 4 季度].[2023 年 11 月].lag(-1)} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果:

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
2023 年 12 月	14.60

由于[2023 年 10 月]、[2023 年 11 月]、[2023 年 12 月]的[科目].&[50220]的值分别为 5、10.4、14.6，index 值为-1.表示[2023 年 11 月]后一个月也就是[2023 年 12 月]的值为 14.6。

示例二

select

{[期间].[2023 年].[2023 年 4 季度].[2023 年 11 月].lag(1)} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果:

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
2023 年 10 月	5.00

由于[2023 年 10 月]、[2023 年 11 月]、[2023 年 12 月]的[科目].&[50220]的值分别为 5、10.4、14.6，index 值为 1.表示[2023 年 11 月]前一个月也就是[2023 年 10 月]的值为 5。

LastChild

返回指定成员的最后一个子代。

语法

member.LASTCHILD
LASTCHILD(member)

参数

member 指定成员

示例一

select

{LastChild([期间].&[50639])} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 3 月	24.00

由于期间维度 2023 年第一季度下子级分别为 2023 年 1 月、2023 年 2 月、2023 年 3 月，对应的值分别为 2、0、24；在 LastChild 函数下，如“LastChild（2023 年 1 月）”，返回的是 2023 年 1 月的最后一个子级成员为 2023 年 3 月，其值为 24.

LastSibling

返回成员的父项的最后一个子级。

语法

member.LASTSIBLING

LASTSIBLING(member)

参数

member

指定成员

示例一

select

{LastSibling([期间].&[50604])} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年	-28746.00

由于期间维度下第二层级父项分别为 2021 年、2022 年、2023 年，第三层级分别为 2021 年第一季度、2021 年第二季度、2021 年第三季度；2022 年第一季度、2022 年第二季度、2022 年第三季度；2023 年第一季度、2023 年第二季度、2023 年第三季度，在 LastSibling 函数下，如“LastSibling（2021 年）”，返回期间第二层级父项最后一个成员为 2023 年，值为-28746.

Lead

返回一个成员，该成员在指定成员之后的指定位置处。

语法

member.LEAD (index)

参数

Index

指定成员相对于指定成员的偏移量。

示例一

select

{[期间].[2023 年].[2023 年 2 季度].lead(1)}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 3 季度	1000.00

由于期间维度[2023 年]下有[2023 年 1 季度]、[2021 年 2 季度]、[2021 年 3 季度]、[2021 年 4 季度]，当 index 为 1 时，代表.[2023 年 2 季度]后 1 个位置的成员，为

[2023 年 3 季度]。

示例二

select

{[期间].[2023 年].[2023 年 2 季度].lead(-1)}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 1 季度	26.00

由于期间维度[2023 年]下有[2023 年 1 季度]、[2021 年 2 季度]、[2021 年 3 季度]、[2021 年 4 季度]，当 index 为-1 时，代表.[2023 年 2 季度]前 1 个位置的成员，为[2023 年 1 季度]。

Name

获取成员的名字

语法

member.NAME

参数

Member

指定成员

示例

with member [期间].[x] as [期间].&[50606].name

select

{[期间].[x]}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	2021 年 1 月

根据 Name 函数，获取的是[期间].[x]维度上 key 为 50606 的成员的名字,2021 年 1 月。

NextMember

返回指定成员所在级别的下一个维度成员。

语法

member.NEXTMEMBER
NEXTMEMBER(member)

示例一

select

{[期间].[2023 年].[2023 年 1 季度].[2023 年 3 月].NextMember} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果：

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
2023 年 4 月	7.00

[2023 年 3 月]的同级的下一个成员是[2023 年 4 月]，由于[2023 年 4 月]的[科目].&[50220]的值为 7，所以返回的值为 7。

示例二

select

{NextMember([期间].[2023 年].[2023 年 1 季度].[2023 年 3 月])} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50584],[场景].&[50685],[产品].&[50688],[科目].&[50220])

运行结果:

产品: 不分

场景: 累计预算

科目: 办公费

组织: 开发 1 部

期间	年初编报 01 版本
2023 年 4 月	7.00

[2023 年 3 月]的同级的下一个成员是[2023 年 4 月]，由于[2023 年 4 月]的[科目].&[50220]的值为 7，所以返回的值为 7。

OpeningPeriod

返回某一指定级别的后代中的第一个维度成员，或者某一指定成员的后代中的第一个维度成员。

语法

OPENINGPERIOD([level [, member]])

参数

level

（可选）只指定了，返回指定级别的成员中的第一个成员

member

（可选）指定了 level 和 member，返回指定成员的指定级别的后代中的第一个

备注

如果未指定任何参数，则找时间类型的维度，除了根节点外的最高基本中的第一个成员。

示例一

```
select
{OpeningPeriod([期间].[LEVEL0],[期间].[2023 年].[2023 年 3 季度])} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2023 年 7 月	-10000.00

由于指定了参数 level 为[LEVEL0] ， member 为[2023 年 3 季度]，所以返回[2023 年 3 季度]零级后代中的第一个成员[2023 年 7 月]。

示例二

```
select
{OpeningPeriod([期间].[LEVEL1])} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
2021 年 1 季度	

由于指定了参数 level 为[LEVEL1] ，所以返回[期间]维度在级别为 1 的后代中的第一个成员[2021 年 1 季度]。

示例三

```
select
select
{OpeningPeriod()} on rows,
{[组织].&[50582]} on columns
```

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果:

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目: 研究开发费

期间	北京总部本部
期间不分	

由于未指定任何参数，所以函数默认返回[期间]维度上除了根节点外最高级别的第一个成员[期间不分]。

ParallelPeriod

返回维度在指定层级中根据维度成员顺序，向前移动指定位数后的维度成员

语法

PARALLELPERIOD([level [,index [,member]]])

参数

level

指定了 level，筛选出 member 在指定层级的祖先

index

（可选）指前推的偏移量，偏移量的单位与 level 有关，index 不指定则为 1

备注

member 的级别不能大于 level 指定的级别

示例一

select

{ParallelPeriod([期间].[LEVEL1],1,[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月])}

on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[科目].&[50221])

运行结果:

产品: 不分

场景: 累计预算

科目: 差旅费

组织: 开发 2 部

期间	年初编报 01 版本
2023 年 3 月	700.00

由于指定了 level 为[LEVEL1]，member 为[2023 年 6 月]，index 为 1，所以会先找[2023 年 6 月]在[LEVEL1]级别的父项为[2023 年 2 季度]，由于 index 为 1，所以找[2023 年 2 季度]前一个维度成员为[2023 年 1 季度]，由于[2023 年 6 月]为[2023 年 2 季度]中第 3 个子项,所以找[2023 年 1 季度]的第 3 个子项为[2023 年 3 月]，所以最后返回[2023 年 3 月]的值 700。

示例二

with member [科目].[x] as (ParallelPeriod([期间].[LEVEL1],1),[科目].&[50221])

select

{[科目].[x]} on rows,

{[版本].&[50672]} on columns

from [模型一]

where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月])

运行结果:

产品: 不分

场景: 累计预算

期间: 2023 年 6 月

组织: 开发 2 部

科目	年初编报 01 版本
x	700.00

指定了 level 为[LEVEL1]，index 为 1，示例一指定的 member 放在 where 后面，为[2023 年 6 月]，所以会先找[2023 年 6 月]在[LEVEL1]级别的父项为[2023 年 2 季度]，由于 index 为 1，所以找[2023 年 2 季度]前一个维度成员为[2023 年 1 季度]，由于[2023 年 6 月]为[2023 年 2 季度]中第 3 个子项，所以找[2023 年 1 季度]的第 3 个子项为[2023 年 3 月]，所以最后返回[2023 年 3 月]的值 700。

示例三

select

```
with member [科目].[x]    as (ParallelPeriod([期间].[LEVEL1]),[科目].&[50221])
select
{[科目].[x]} on rows,
{[版本].&[50672]} on columns
from [模型一]
where ([组织].&[50585],[场景].&[50685],[产品].&[50688],[期间].[2023 年].[2023 年 2 季度].[2023 年 6 月])
```

运行结果：

产品: 不分

场景: 累计预算

期间: 2023 年 6 月

组织: 开发 2 部

科目	年初编报 01 版本
x	700.00

指定了 level 为[LEVEL1]，示例一指定的 member 放在 where 后面，为[2023 年 6 月]，index 未指定默认为 1，所以会先找[2023 年 6 月]在[LEVEL1]级别的父项为[2023 年 2 季度]，由于 index 为 1，所以找[2023 年 2 季度]前一个维度成员为[2023 年 1 季度]，由于[2023 年 6 月]为[2023 年 2 季度]中第 3 个子项，所以找[2023 年 1 季度]的第 3 个子项为[2023 年 3 月]，所以最后返回[2023 年 3 月]的值 700。

Parent

返回成员的父级。

语法

member.PARENT
PARENT(member)

参数

Member

指定成员

示例

select

{Parent([期间].&[50640])} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([场景].&[50685],[产品].&[50688],[版本].&[50672],[科目].&[50236])

运行结果：

产品: 不分

场景: 累计预算

版本: 年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 1 季度	26.00

由于期间维度 2023 年第一季度下子级分别为 2023 年 1 月、2023 年 2 月、2023 年 3 月，对应的值分别为 2、0、24；在 Parent 函数下，如“Parent（2023 年 1 月）”，返回的是 2023 年 1 月的父级成员为 2023 年第一季度，其值为 26.

PrevMember

此函数返回指定成员级别中的前一个成员。

语法

member. PrevMember
PrevMember (member)

参数

此函数返回与 «Member» 中所指定的成员位于同一级别的前一个成员。

备注

PREVMEMBER 有两种写法：按维度成员在前，函数在其后和函数在前，维度成员在其后两种形式。

示例一

select

{[期间].[2023 年].[2023 年 3 季度].PrevMember}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 2 季度	228.00

由于 PrevMember 是指定成员同一级别的前一个成员，即期间维度[2023 年 3 季度]的同级成员的前一个成员是[2023 年 2 季度]，在 PrevMember 函数中，返回结果为[2023 年 2 季度]的北京总部本部、研究开发费等组合的费用为 228。

示例二

select

{[期间].[2023 年].[2023 年 3 季度].[2023 年 8 月].PrevMember}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 7 月	-10000.00

由于 PrevMember 是指定成员同一级别的前一个成员，即期间维度[2023 年 8 月]的同级成员的前一个成员是[2023 年 7 月]，在 PrevMember 函数中，返回结果为[2023 年 7 月]的北京总部本部、研究开发费等组合的费用为-10000

示例三

select

{PrevMember([期间].[2023 年].[2023 年 3 季度])}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 2 季度	228.00

由于 PrevMember 指定成员同一级别的前一个成员，即期间维度[2023 年 3 季度]的同级成员的前一个成员是[2023 年 2 季度]，在 PrevMember 函数中，返回结果为[2023 年 2 季度]的北京总部本部、研究开发费等组合的费用为 228

示例四

select

{PrevMember([期间].[2023 年].[2023 年 3 季度].[2023 年 8 月])}on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
2023 年 7 月	-10000.00

由于 PrevMember 指定成员同一级别的前一个成员，即期间维度[2023 年 8 月]的同级成员的前一个成员是[2023 年 7 月]，在 PrevMember 函数中，返回结果为[2023 年 7 月]的北京总部本部、研究开发费等组合的费用为-10000

Properties

此函数为维度成员属性函数，返回类型由关键字类型来决定

语法

member. PROPERTIES (property_name [, TYPED])

参数

返回包含成员属性值的字符串，或保留了原始类型的数据（使用了 TYPED 参数）

成员的固有属性一定返回的原始类型的数据

备注

目前 property_name 支持的取值（都是固有属性）和返回结果说明

CATALOG_NAME：库名

SCHEMA_NAME：库名

CUBE_NAME：库名

DIMENSION_UNIQUE_NAME：维度名（唯一）

HIERARCHY_UNIQUE_NAME：层次结构名（唯一）

LEVEL_UNIQUE_NAME：层次名（唯一）

LEVEL_NUMBER：层次编号

MEMBER_UNIQUE_NAME：成员名（唯一）

MEMBER_NAME：成员名

MEMBER_TYPE：成员类型

MEMBER_CAPTION：成员名

MEMBER_ORDINAL：成员次序编号

CHILDREN_CARDINALITY：子代的数目

PARENT_LEVEL：父项的层次编号

PARENT_UNIQUE_NAME：父项的名字（唯一）

PARENT_COUNT：祖先的数目

DESCRIPTION：成员描述（当前为成员名）

示例一

```
with member  [期间].[x] as [期间].&[2023 年 1 季度].Properties ('LEVEL_NUMBER')
select
{[期间].[x]} on rows,
{[组织].&[50582]} on columns
```

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	1.00

由于 Properties 是属性函数，返回由关键字类型来决定。即期间维度[x]中，查询[2023 年 1 季度]的层级编码，返回结果为[2023 年 1 季度]的北京总部本部、研究开发费等组合的层级编码为 1，零级成员的层级编码为 0；

示例二

With member [期间].[x] as [期间].&[2023 年 1 季度].Properties ('CHILDREN_CARDINALITY')
select

{[期间].[x]} on rows,

{[组织].&[50582]} on columns

from [模型一]

where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	3.00

由于 Properties 是属性函数，返回由关键字类型来决定。即期间维度[x]中，查询[2023 年 1 季度]的子代的数目，返回结果为[2023 年 1 季度]的北京总部本部、研究开发费等组合的子代的数目为 3（2023 年 1 月/2023 年 2 月/2023 年 3 月）；

示例三

```
with member [期间].[x] as [期间].&[2023 年 1 季度].Properties ('DIMENSION_UNIQUE_NAME')
select
{[期间].[x]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	[期间]

由于 Properties 是属性函数，返回由关键字类型来决定。即期间维度[x]中，查询[2023 年 1 季度]的维度名，返回结果为[2023 年 1 季度]的北京总部本部、研究开发费等组合的维度名为期间（2023 年 1 季度在期间维度下的成员）；

示例四

```
with member [期间].[x] as [期间].&[2023 年 1 季度].Properties ('LEVEL_NUMBER',TYPED)+1
select
{[期间].[x]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	2.00

由于 Properties 是属性函数，返回由关键字类型来决定。即期间维度[x]中，查询[2023 年 1 季度]的层级编码，返回结果为[2023 年 1 季度]的北京总部本部、研究

开发费等组合的层级编码为 1 再加上 1 后变成了 2，零级成员的层级编码为 0；

示例五

```
with member [期间].[x] as [期间].&[2023 年 1 季度].Properties ('PARENT_LEVEL',TYPED)+1
select
{[期间].[x]} on rows,
{[组织].&[50582]} on columns
from [模型一]
where ([科目].&[50236],[场景].&[50685],[产品].&[50688],[版本].&[50672])
```

运行结果：

产品: 不分

场景: 累计预算

版本：年初编报 01 版本

科目：研究开发费

期间	北京总部本部
x	3.00

由于 Properties 是属性函数，返回由关键字类型来决定。即期间维度[x]中，查询[2023 年 1 季度]的父项的层级编码，返回结果为[2023 年 1 季度]的北京总部本部、研究开发费等组合的父项的层级编码为 3（因为 2023 年 1 季度的编码是 1，其父项变成了 2，加上 1 变成了 3）。