
# InterSystems MDX リファレンス

Version 2024.1
2024-06-03

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

InterSystems MDX リファレンス
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03




インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

# 目次

# テーブル一覧

# 基本的な規則

この参照セクションでは、Business Intelligence 内の MDX の最も基本的な規則、すなわち識別子とコメントの規則について説明します。

"BI サンプルのアクセス方法" も参照してください。

# 識別子 (MDX)

このセクションでは、InterSystems MDX の識別子について説明します。

## MDX 識別子および許可されるバリエーション

あらゆるキューブ、サブジェクト領域、リレーションシップ、ディメンジョン、階層、レベル、メンバ、プロパティ、およびメジャーには、識別子があります。

各 MDX 識別子は、以下のいずれかの形式になります。

- 標準的な識別子：

  ```
  name
  ```

  この形式は、name が以下の規則に従う場合に使用できます。

  - 先頭の文字は、アルファベット文字 (Latin-1) とアンダースコア文字 (_) のいずれかにする必要があります。

    InterSystems MDX では、残りの文字も数値の場合、先頭の文字を数値にできます。このインターシステムズによる MDX への拡張機能により、すべてが数値の名前 (郵便番号など) を持つメンバを簡単に参照できるようになります。

  - 他の文字は、アルファベット文字、アンダースコア文字、数値文字のいずれかにする必要があります。
  - 名前に MDX 予約キーワードを使用することはできません。MDX では予約キーワードの大文字と小文字は区別されません。

- 区切り識別子：

  ```
  [name]
  ```

  この形式は、name が上記の規則に従わない場合に使用する必要があります。name に右角括弧 (]) が含まれる場合、右角括弧を 2 つ連続して (]]) 使用して、この状態をエスケープする必要があります。

- (メンバの場合のみ) メンバ・キー：

  ```
  &[key value]
  ```

  キー値についての詳細は、"キー値" を参照してください。

注釈　name では、大文字と小文字は区別されませんが、key value では区別されます。

重要　name と key value は式ではないことに注意してください。そのため、これらを文字列値の式で置き換えることはできません。

# コメント (MDX)

このセクションでは、InterSystems MDX クエリのコメント、およびモデル定義内で使用される独立した MDX 式のコメントについて説明します。

## コメント

任意の MDX クエリまたは式で、空白を使用できる任意の場所に、以下の形式のコメントを含めることができます。

```
/* comment here */
```

以下の形式のコメントを、MDX クエリまたは式の中で、別の行として含めることができます。

```
-- comment
```

または、以下のようにします。

```
// comment
```

また、上記の 2 つの形式を、MDX を含む行の最後に使用することもできます。

## 例

以下の例は、有効な MDX クエリを示しています。

```
select /* change this later */ birthd.decade.members on 1
from patients


--comment
select birthd.decade.members on 1 --comment
from patients // comment
//another comment
```

# 式のタイプ (MDX)

この参照セクションでは、Business Intelligence でサポートされている MDX 式のタイプについて説明します。

"BI サンプルのアクセス方法" も参照してください。

# 数値式 (MDX)

このセクションでは、InterSystems MDX で数値式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、数値式の形式は以下のいずれかです。

- 数値リテラル。例えば、`37` です。

  このリテラルでは、先頭に小数点を使用できません。そのため、小数値の先頭には 0 を含める必要があります。例えば、`0.1` は有効ですが、`.1` は無効です。

- パーセンテージ・リテラル。例えば、`10%` です。

  数値とパーセント記号の間にスペースを入れてはいけません。

- 数値メジャーを参照する式 (`MEASURES.[%COUNT]` など)。

- 数値を返す MDX 関数を使用する式 (`AVG(aged.age, MEASURES.[test score])` など)。

  AVG、MAX、COUNT など、MDX 関数の多くが数値を返します。また、IIF 関数も数値を返すことができます。この関数は、条件を評価して、条件に応じて 2 つの値のいずれかを返します。

- 算術演算子を使用して数値式を組み合わせる式。例えば、`MEASURES.[%COUNT] / 100` です。

  システムでは、`+` (加算)、`-` (減算)、`/` (除算)、および `*` (乗算) の標準的な算術演算子がサポートされます。また、DeepSee II では、`+` (正)、および `-` (負) の標準単項演算子もサポートされます。

  優先順位を制御する括弧を使用できます。

  式内のいずれかの値が NULL の場合、その式は NULL と評価されます。

  値を 0 で除算すると、システムはその結果を NULL として扱います。

  Tipヒン MDX の関数 IIF は、このような式に利用できます。
  ト

- メンバ式 (`[gend].[h1].[gender].[female]` など)。

  メンバ式の値は、現在使用されているメジャーによって異なります。既定では、この式の評価結果は、このメンバに属するレコードの数です。これに対し、特定のメジャーが使用されている場合、この式の評価結果は、それらのレコード全体にわたるそのメジャーの集約値です。

- ディメンジョンの MDX 識別子 (`[gend]` など)。

  この式の値は、現在使用されているメジャーによって異なります。既定では、この式の評価結果は、キューブ内のレコードの数です。これに対し、特定のメジャーが使用されている場合、この式の評価結果は、キューブ内の全レコードにわたるそのメジャーの集約値です。

- 数値を含むピボット変数への参照。ピボット変数を参照するには、以下の構文を使用します。

  ```
  $VARIABLE.variablename
  ```

  variablename は論理変数名です。この式は角括弧で囲まないでください。この構文では大文字と小文字は区別されません。ピボット変数名も同様です。

  ピボット変数の定義に関する詳細は、"ピボット変数の定義と使用" を参照してください。

## 用途

数値式は、以下の方法で使用できます。

- 多くの MDX 関数の数値引数として。以下はその例です。

```
AVG(diagd.MEMBERS, MEASURES.[%COUNT])
```

- セットの要素として。
- 計算メンバ (この場合はメジャー) のディメンジョンとして。

## 例

このセクションでは、あまり一般的でない種類の数値式の例を示します。最初の例では、メンバ式に数値が含まれています。

```
SELECT gend.h1.gender.female ON 0 FROM patients
```

```
                                    Female
                                        526
```

次の例は、上記のバリエーションです。

```
SELECT gend.h1.gender.female+100 ON 0 FROM patients
```

```
                                Expression
                                       626
```

前述したように、メンバ式の値は、現在使用されているメジャーによって異なります。

```
SELECT gend.h1.gender.female ON 0 FROM patients WHERE MEASURES.[avg age]
```

```
                                    Female
                                     37.23
```

以下も同様です。

```
SELECT gend.h1.gender.female+500 ON 0 FROM patients WHERE MEASURES.[avg age]
```

```
                                Expression
                                     537.23
```

次の例では、ディメンジョンの数値が示されています。

```
SELECT allerd ON 0 FROM patients
```

```

                                     1,000
```

前述したように、このような式の値は、現在使用されているメジャーによって異なります。

```
SELECT allerd ON 0 FROM patients WHERE MEASURES.[avg allergy count]
```

```

                                      0.99
```

# 文字列式 (MDX)

このセクションでは、InterSystems MDX で文字列式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、文字列式の形式は以下のいずれかです。

- 文字列リテラル (二重引用符に任意の文字が続き、さらに別の二重引用符が続く)。例えば、`"label"` や `"my property"` などです。
- 数値リテラル。
- 文字列を返す MDX 関数を使用する式。これらの関数には以下が含まれます。
  - PROPERTIES。これはメンバのプロパティ値を返します。例えば、`homed.city.magnolia.PROPERTIES("Population")` などです。

    ユーザが使用できる内部プロパティについては、参照セクション "内部プロパティ" を参照してください。
  - IIF。これは指定された論理式の値に応じて、2 つの値のいずれかを返します。
  - LOOKUP。これは指定されたキーを条件リストで検索し、代替文字列を返します。
  - %LOOKUP。これは条件リストから値を 1 つ返します。
- 文字列値のメジャーを参照する式。例えば、以下のように計算メンバを定義するとします。

  ```
  CREATE MEMBER patients.measures.stringtest AS 'IIF(MEASURES.[avg test score]<60, "low","high")'
  ```

  式 `MEASURES.stringtest` は、文字列式です。
- MDX 連結演算子 (+) を使用して他の文字列式を組み合わせる式。以下はその例です。

  ```
  "string 1 " + "string 2"
  ```

  別の例を示します。

  ```
  "Pop: " + homed.city.magnolia.PROPERTIES("Population")
  ```
- 文字列値を含むピボット変数への参照。ピボット変数を参照するには、以下の構文を使用します。

  ```
  $VARIABLE.variablename
  ```

  variablename は論理変数名です。この式は角括弧で囲まないでください。この構文では大文字と小文字は区別されません。ピボット変数名も同様です。

  ピボット変数の定義に関する詳細は、"ピボット変数の定義と使用" を参照してください。

## 用途

文字列式は、以下の方法で使用できます。

- ORDER 関数または PROPERTIES 関数の引数として。
- セットの要素として。
- 計算メンバ (この場合はメジャー) のディメンジョンとして。

# 論理式 (MDX)

このセクションでは、InterSystems MDX で論理式を作成し、使用する方法について説明します。

## 詳細および使用法

MDX には、論理データ型は含まれていません。ただし、FILTER 関数と IIF 関数が用意されており、両方とも True または False として扱われる引数を取ります。これらのコンテキストでは、InterSystems MDX は以下のように、数値および文字列値を True または False と解釈します。

- 0 に評価される数値式は False として扱われます。
- 他のすべての数値式は、True として扱われます。
- すべての文字列式は False として扱われます。

同じコンテキストで、以下の標準ツールを使用して True 値と False 値を組み合わせることができます。

- 論理比較演算子 : > (より大きい)、>= (以上)、= (等しい)、< (より小さい)、<= (以下)。例えば、以下の式は、MEASURES.[%COUNT] の値に応じて、True または False を返します。

  ```
  MEASURES.[%COUNT]>1100
  ```

  別の例として、以下の式は、現在のメンバの名前に応じて、True または False を返します。

  ```
  colord.CURRENTMEMBER.PROPERTIES("Name")="Red"
  ```

  数値式と NULL を比較する場合、NULL 値は 0 として扱われます。
- AND 演算子、OR 演算子、および優先順位を制御する括弧。例えば、以下の式は、MEASURES.[%COUNT] の値に応じて、True または False を返します。

  ```
  (MEASURES.[%COUNT]>1100) AND (MEASURES.[%COUNT]<1500)
  ```

タイプの異なるスカラ値を比較する場合、システムは以下のように比較します。

| 条件 | 結果 |
|---|---|
| 両方の式が数値 | 数値比較を実行する |
| 両方の式が文字列 | 文字列比較を実行する |
| 片方の式が数値で、片方が文字列 | 文字列式より数値式が小さい |

メンバ式を論理比較式で使用することはできません。例えば、以下は有効な論理式ではありません。

```
homed.CURRENTMEMBER = homed.h1.city.magnolia
```

代わりに、Name プロパティなどの適切なプロパティ (すべて文字列) を使用する必要があります。例えば、以下は有効な論理式です。

```
homed.CURRENTMEMBER.Properties("Name") = "magnolia"
```

ユーザが使用できる内部プロパティについては、参照セクション ″内部プロパティ″ を参照してください。

″メジャー検索式″ のセクションも参照してください。

# タプル式 (MDX)

このセクションでは、InterSystems MDX でタプル式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、タプル式の形式は以下のいずれかです。

- タプル・リテラルは、メンバ式のコンマ区切りのリストを括弧で囲んだものです。

  ```
  (member_expr1, member_expr2, member_expr3, ...)
  ```

  member_expr1、member_expr2、member_expr3 (以降同様) はメンバ式です。括弧は必須です。

- タプル・リテラルを含むピボット変数への参照。ピボット変数を参照するには、以下の構文を使用します。

  ```
  $VARIABLE.variablename
  ```

  variablename は論理変数名です。この式は角括弧で囲まないでください。この構文では大文字と小文字は区別されません。ピボット変数名も同様です。

  ピボット変数の定義に関する詳細は、"ピボット変数の定義と使用" を参照してください。

## 注意事項と追加用語

MDX の他の実装では、1 つのタプルに、同じディメンジョンの複数のメンバを含めることはできません。InterSystems MDX では、1 つのタプル式に、同じディメンジョンの複数のメンバ式を含めることができます。ほとんどの場合、その結果は NULL になります。ただし、Business Intelligence では、レベルはリスト値に基づくことができます。つまり、ある 1 つのレコードが複数のメンバに属することができます。例えば、タプル `(allerd.soy,allerd.wheat)` は、大豆と小麦の両方にアレルギーがあるすべての患者を表し、このタプルは NULL でない値を持つ可能性があります。

タプルがキューブ内の各ディメンジョンを参照する場合、そのタプルは完全修飾されています。そうでない場合、このタプルは部分修飾されています。`Patients` キューブから部分修飾されたタプルの例を以下に示します。

```
(allerd.[dairy products], colord.red, aged.35)
```

部分修飾されたタプルの別の例を以下に示します。

```
(diagd.asthma, aged.[age group].[30 to 59], MEASURES.[%COUNT])
```

また、クエリによって返された各データ・セルがタプルであることに注意してください。

CROSSJOIN 関数は、NONEMPTYCROSSJOIN 関数と同様に、一連のタプルを返します。

## 用途

タプル式は、以下の方法で使用できます。

- セットの要素として。
- WHERE 節の引数として。

# メンバ式 (MDX)

このセクションでは、InterSystems MDX でメンバ式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、メンバ式の形式は以下のいずれかです。

- 単一メンバへの名前による明示的な参照であるメンバ・リテラル。

  ```
  [dimension_name].[hierarchy_name].[level_name].[member_name]
  ```

  以下は、この指定の説明です。

  - [dimension_name] は、ディメンジョンの名前を付ける MDX 識別子 です。
  - [hierarchy_name] は、そのディメンジョン内の階層の名前を付ける MDX 識別子 です。階層名は省略できます。省略すると、クエリでは、このディメンジョン内の定義に従って、指定された名前の最初のレベルが使用されます。

    計算メンバは階層内にないため、計算メンバを参照する場合は階層名を省略できます。
  - [level_name] は、その階層内のレベルの名前を付ける MDX 識別子 です。レベル名は省略できます。省略すると、クエリでは、このディメンジョン内の定義に従って、指定された名前の最初のメンバが使用されます。

    また、計算メンバはレベル内にないため、計算メンバを参照する場合はレベル名を省略できます。
  - [member_name] は、そのレベルのメンバの名前を示す MDX 識別子 です。

  例えば、以下のように作成できます。

  ```
  [gend].[h1].[gender].[female]
  ```

  メジャーの場合は、任意のメンバが、Measures と呼ばれるディメンジョンのメンバになるため、メンバ・リテラルのバリアント形式は以下のようになります。このディメンジョンには、階層やレベルはありません。

  ```
  [MEASURES].[measure_name]
  ```

  例えば、以下のように作成できます。

  ```
  [MEASURES].[%COUNT]
  ```

  メジャーはメンバですが、上記のような式はスカラ値を返すため、他のメンバ式とまったく同じように使用できるわけではないことに注意してください。

- 以下のような、階層への明示的な参照。

  ```
  [dimension_name].[hierarchy_name]
  ```

  この式は、階層が属するディメンジョンの All メンバを返します。

  以下はその例です。

  ```
  aged.h1
  ```

  上記の式は、AgeD ディメンジョンの All メンバを返します。サンプルの Patients キューブでは、このメンバの名前は All Patients になっています。

- 単一メンバを返す MDX 関数を使用する式。以下はその例です。

  ```
  [gend].[h1].[gender].female.NEXTMEMBER
  ```

LAG、NEXTMEMBER、PARENT など、MDX 関数の多くがメンバを返します。

%TERMLIST はメンバを返せることに注意してください。

- メンバの内部キーを使用して、以下の構文でそのメンバを返す式。

  ```
  [dimension_name].[hierarchy_name].[level_name].&[member_key]
  ```

  ここで、member_key は、メンバの KEY プロパティの値です。例を以下に示します。

  ```
  birthqd.h1.quarter.&[2]
  ```

  member_key は、式ではなく大文字と小文字を区別するリテラル値です。つまり、これを文字列値の式に置き換えることはできません。

  KEY 値の作成方法の詳細は、リファレンスの "内部プロパティ" を参照してください。また、PROPERTIES 関数を使用して、KEY プロパティなど、メンバのプロパティの値を検索できます。

- InterSystems MDX 拡張機能を使用して、以下の構文で別のキューブ内のメンバを参照する式。

  ```
  [relationship_name].member_expression
  ```

  [relationship_name] は、クエリで使用されるキューブ内のリレーションシップの名前を付ける MDX 識別子、member_expression は、そのキューブに適したメンバ式です。

- メンバ式を含むピボット変数への参照。ピボット変数を参照するには、以下の構文を使用します。

  ```
  $VARIABLE.variablename
  ```

  variablename は論理変数名です。この式は角括弧で囲まないでください。この構文では大文字と小文字は区別されません。ピボット変数名も同様です。

  ピボット変数の定義に関する詳細は、"ピボット変数の定義と使用" を参照してください。

例えば、以下のメンバ式はすべて同等です。

```
[gend].[h1].[gender].Female
[gend].female
gend.H1.gender.female
gend.h1.FEMALE
gend.female
```

## 計算メンバ

メンバは、レベルの定義の一部として、キューブ定義内で定義できます。計算メンバを作成することもできます。これは通常、他のメンバに基づきます。計算メンバは、以下の 2 つの方法で定義できます。

- クエリの WITH 節の中。このメンバは、クエリの残りの部分で使用可能ですが、他のクエリでは使用できません。
- CREATE MEMBER 文の中。このメンバは、セッションの残りの部分 (MDX シェルのセッションの残りの部分など) で使用可能です。

## 用途

メンバ式は、以下の方法で使用できます。

- 多くの MDX 関数のメンバ引数として。
- WHERE 節の引数として。
- セットの要素として。

# レベル式 (MDX)

このセクションでは、InterSystems MDX でレベル式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、レベル式の形式は以下のいずれかです。

- 以下のように、レベルへの直接参照であるレベル・リテラル。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  以下は、この指定の説明です。

  - [dimension_name] は、ディメンジョンの名前を付ける MDX 識別子 です。
  - [hierarchy_name] は、そのディメンジョン内の階層の名前を付ける MDX 識別子 です。階層名は省略できます。省略すると、クエリでは、このディメンジョン内の定義に従って、指定された名前の最初のレベルが使用されます。
  - [level_name] は、その階層内のレベルの名前を付ける MDX 識別子 です。

  例えば、以下のように作成できます。

  ```
  [gend].[h1].[gender]
  ```

- InterSystems MDX 拡張機能を使用して、以下の構文で別のキューブ内のレベルを参照する式。

  ```
  [relationship_name].level_expression
  ```

  relationship_name はクエリで使用されるキューブ内のリレーションシップの名前で、level_expression はそのリレーションシップに含まれるレベルを参照します。

## 用途

レベル式は、以下の MDX 関数の引数として使用できます。

- %TOPMEMBERS
- ALLMEMBERS
- MEMBERS
- ANCESTOR
- CLOSINGPERIOD
- OPENINGPERIOD
- PARALLEL PERIOD

# 階層式 (MDX)

このセクションでは、InterSystems MDX で階層式を作成し、使用する方法について説明します。

## 詳細

InterSystems MDX では、階層式の形式は以下のいずれかです。

- 以下のように、階層への直接参照である階層リテラル。

  ```
  [dimension_name].[hierarchy_name]
  ```

  以下は、この指定の説明です。

  - [dimension_name] は、ディメンジョンの名前を付ける MDX 識別子 です。
  - [hierarchy_name] は、そのディメンジョン内の階層の名前を付ける MDX 識別子 です。

  例えば、以下のように作成できます。

  ```
  [gend].[h1]
  ```

- ディメンジョンへの参照。

  ```
  [dimension_name]
  ```

  例えば、以下のように作成できます。

  ```
  [gend]
  ```

  システムはこれを、そのディメンジョンの最初に表示される階層への参照と解釈します。

- InterSystems MDX 拡張機能を使用して、以下の構文で別のキューブ内の階層を参照する式。

  ```
  [relationship_name].hierarchy_expression
  ```

  relationship_name はクエリで使用されるキューブ内のリレーションシップの名前で、hierarchy_expression はそのリレーションシップに含まれる階層を参照します。

## 用途

むき出しの階層式はすべてのレコードを返します。この式は All メンバと等しくなります。(ディメンジョンで形式的に All レベルを定義しなくても、むき出しの階層式を使用できることに注意してください。)以下はその例です。

```
SELECT MEASURES.[%count] ON 0, colord.h1 ON 1 FROM patients
```

| | Patient Count |
|---|---|
| | 1,000 |

また、階層式は、以下の任意の関数の引数として使用できます。

- %TOPMEMBERS
- ALLMEMBERS
- CURRENTMEMBER
- MEMBERS

# セット式 (MDX)

このセクションでは、InterSystems MDX でセット式を作成し、使用する方法について説明します。

## 詳細

set expression の一般的な構文は以下のとおりです。

```
{expression1, expression2, ...}
```

このリストには、項目を任意の数だけ含めることができます。InterSystems MDX では、リスト内の項目が 1 つだけの場合、中括弧を省略できます。

このリストでは、expression1、expression2 (以降同様) の形式は以下のいずれかです。

- メンバ式。
- 数値式または文字列式。
- 以下のように指定された、ある範囲のメンバ。

  ```
  member1:member2
  ```

  この式は、指定された 2 つのメンバとその間のすべてのメンバで構成されるセットを返します (これらのメンバを含むレベル内のメンバの順序が指定されている場合)。これらのメンバは、同じレベルに属している必要があります。

  以下はその例です。

  ```
  birthd.year.1960:birthd.year.1980
  ```

  member2 では、ディメンジョン、階層、およびレベルの識別子を省略できます。以下はその例です。

  ```
  birthd.year.1960:1980
  ```

- セットを返す MDX 関数を使用する式。以下に例を示します。

  ```
  homed.zip.MEMBERS
  ```

  MEMBERS、NONEMPTYCROSSJOIN、ORDER など、MDX 関数の多くがセットを返します。

  %TERMLIST はセットを返せることに注意してください。
- 別のセット式。
- 名前付きセットの名前。次のセクションを参照してください。
- タプル式。

  (MDX の他の実装では、セット内の各タプルにおいて、セット内の他のタプルと同じ順序でディメンジョンを使用する必要があります。例えば、最初のタプルがその最初のリスト項目にディメンジョン A を使用している場合、他のすべてのタプルも同様にする必要があります。InterSystems MDX には、この制約はありません。同様に、MDX の他の実装では、タプルと他のタイプのセット要素の組み合わせをセットに含めることはできません。InterSystems MDX には、この制約もありません)。
- セット式を含むピボット変数への参照。ピボット変数を参照するには、以下の構文を使用します。

  ```
  $VARIABLE.variablename
  ```

  variablename は論理変数名です。この式は角括弧で囲まないでください。この構文では大文字と小文字は区別されません。ピボット変数名も同様です。

  ピボット変数の定義に関する詳細は、"ピボット変数の定義と使用" を参照してください。

キーワード句 NON EMPTY を任意のセット式の先頭に付加できます。以下はその例です。

```
NON EMPTY {birthd.year.1960:1980}
NON EMPTY birthd.year.1960:1980
NON EMPTY {homed.zip.MEMBERS}
NON EMPTY homed.zip.MEMBERS
```

NON EMPTY キーワード句は、セットの空の要素を抑制します。すなわち、セットが評価され、空の要素は削除されます。このキーワードは、CROSSJOIN を使用し、フィルタによって要素が NULL になる可能性があるシナリオで特に役に立ちます。

## 名前付きセット

名前付きセットは、セット名とセット式という 2 つの要素で構成されます。名前付きセットは、以下の 2 つの方法で定義できます。

- クエリの WITH 節の中。このセット名は、クエリの残りの部分で使用可能ですが、他のクエリでは使用できません。
- CREATE SET 文の中。この名前付きセットは、セッションの残りの部分 (MDX シェルのセッションの残りの部分など) で使用可能です。

## 用途

セット式は、以下の方法で使用できます。

- 多くの MDX 関数のセット引数として。一部の関数では、セットはメンバのみで構成されている必要があります。また、セットがメンバまたはタプルのみで構成されている必要のある関数もあります。このドキュメントでは、必要に応じてこれらの要件を記述します。
- SELECT 文の軸として。
- 前のサブセクションの説明のとおりに、名前付きセットの定義として。

# メジャー検索式 (MDX)

このセクションでは、メジャー検索式を作成して使用する方法について説明します。この式を使用すると、ファクト自体 (つまり、集約レベルではなく最下位レベル) のメジャーの値に基づいて、ファクト・テーブルから行にアクセスできます。これらの式は、インターシステムズによる MDX への拡張機能です。

## 詳細

メジャー検索式には、以下の構文があります。これは、%SEARCH という、Business Intelligence の特殊なディメンジョンを参照します。

```
%SEARCH.&[comparison expression]
```

ここで、comparison expression は、以下の例のような論理式です。

```
[MEASURES].[test score]>60
```

注釈　比較式を囲む角括弧と比較式内のメジャー識別子の角括弧の両方のセットの角括弧が必要です。したがって、有効な検索式は常に **%SEARCH.[[** で開始します。

例えば、以下のクエリは、テスト・スコアが 60 より高い患者をすべて選択します。

```
SELECT FROM patients WHERE %SEARCH.&[[MEASURES].[Test Score]>60]
Result:                6,191
```

より一般的には、comparison expression は、論理式の組み合わせにできます。式には、以下の要素を含めることができます。

- 論理比較演算子 : > (より大きい)、>= (以上)、= (等しい)、< (より小さい)、<= (以下)。

  検索可能メジャーに文字列値が含まれる場合は、SQL LIKE 演算子も使用できます。
- AND 演算子、OR 演算子、および優先順位を制御する括弧。
- 数値リテラル。
- 一重引用符で囲んだ文字列リテラル。
- SQL 式の IS NULL および IS NOT NULL。例えば以下のようになります。

  ```
  SELECT FROM HOLEFOODS WHERE [%Search].&[[Measures].[Units Sold] IS NULL]
  ```

## 用途

メジャー検索式は、以下のすべての状況で使用できます。

- %FILTER 節の引数として。
- WHERE 節の引数として。
- FILTER 関数の引数として。

## その他の注意事項

システムでは、メジャー検索式が以下のように解析されます。

1. %Search は、ディメンジョンとして扱われます。
2. 比較式は **&[]** 内に囲まれているため、システムはこれを KEY 値として扱い、任意の構文を含むことを許可します。

3. 比較式は、ファクト・テーブルに対する SQL 文に変換されます。

これは、comparison expression に SQL 構文を含めることができることを意味します。

また、キューブの定義で `searchable="true"` とマークされていない場合でも、メジャー検索式でメジャーを使用できる場合があります。この属性値があると、システムでは以下の 2 つを実行します。

- このメジャーを、高度なフィルタのオプションとして表示する。
- 必要に応じてインデックスを追加し、このメジャーを検索可能にする。

# 品質メジャー式 (MDX)

このセクションでは、品質メジャー式を作成および使用する方法について説明します。品質メジャー式を使用すると、品質メジャーの値にアクセスできます。これらの式は、インターシステムズによる MDX への拡張機能です。

## 詳細

品質メジャー式には、以下の構文があります。これは、%QualityMeasure という、Business Intelligence の特殊なディメンジョンを参照します。

```
[%QualityMeasure].&[catalog/set/qm name]
```

または、以下のようにします。

```
[%QualityMeasure].&[catalog/set/qm name/group name]
```

以下は、この指定の説明です。

- catalog は、その品質メジャーが属するカタログです。
- set は、そのカタログ内のセットです。
- qm name は、その品質メジャーの短い名前です (品質メジャーのフルネームは catalog/set/qm name になります)。
- group name は、指定された品質メジャーで定義されているグループの名前です。

最初の式は、品質メジャーの値を返します。2 番目の式は、指定されたグループの値を返します。

## 用途

品質メジャー式は、他のメジャーと同じ方法で使用できます。

# MDX の文と節

この参照セクションでは、Business Intelligence でサポートされている MDX の文と節について説明します。

"BI サンプルのアクセス方法" も参照してください。

# %FILTER 節 (MDX)

SELECT 文にフィルタを適用します。SELECT 文の結果をスライスする方法について説明します。この節は WHERE に似ていますが、1 つの文に複数の %FILTER 節を含めることができる点が異なります。%FILTER は、インターシステムズによる MDX への拡張機能です。

## 構文および詳細

```
select_statement %FILTER set_expression
```

以下は、この指定の説明です。

- select_statement は、SELECT を使用する文です。
- set_expression は、メンバまたはタプルのセットを返す式です。

  set_expression の代わりに、メジャー検索式を使用できます。この場合の %FILTER の動作については、例を参照してください。

InterSystems MDX では、必要に応じてタイプが自動的に変換されるため、セット式の代わりに、単一のメンバ式またはタプル式を使用することもできます。

%FILTER 節は、必要な数だけ含めることができます。この節は、SELECT 文に付加するだけで、結果をさらにフィルタ処理できるため、プログラムでクエリを実行する場合に特に便利です (これに対して、WHERE 節を使用して別のフィルタ項目を追加する必要がある場合、許可される WHERE 節が 1 つのみであるため、WHERE 節を書き換える必要があります)。

重要

各セット要素は別のスライサ軸として使用され、(すべての %FILTER 節の) すべてのスライサ軸の結果がいっしょに集約されます。これは軸のたたみ込みのプロセスです (フィルタはクエリ軸と見なされます)。軸のたたみ込みを実行すると、どのスライサ軸にも NULL の結果がないソース・レコードは複数回カウントされます。

軸のたたみ込みでは、キューブの定義での指定に従い、対象としているメジャーの集約メソッドに基づいて値が組み合わされます (ここの例では、**%COUNT** が追加されます)。

詳細は、"Business Intelligence クエリ・エンジンの仕組み" の "軸のたたみ込み" を参照してください。

## 例

%FILTER 節をメジャー検索式で使用する場合、この節は、ファクト・テーブルの行のうち、指定された条件を満たした行を使用します (メジャー検索式は、インターシステムズによる MDX への拡張機能で、ファクト・テーブル自体のメジャー値を考慮するものです)。

```
SELECT MEASURES.[%COUNT] ON 0 FROM patients %FILTER %SEARCH.&[[MEASURES].[age]<10]
                            Patient Count
                                    1,370
```

## %SQLRESTRICT

キューブに対して %SQLRESTRICT ディメンジョンを有効にしている場合、そのキューブに対する MDX クエリに SQL **SELECT** 文または **WHERE** 節を含めることができます。これにより、MDX クエリの SQL 文によって実行時の制限を有効にできます。%SQLRESTRICT ディメンジョンを使用するには、以下の構文で %FILTER 節を使用します。

```
%FILTER %SQLRESTRICT.&[sqlStatement]
```

sqlStatement は、SQL **SELECT** 文または SQL **WHERE** 節です。**SELECT** 文は、キューブのソース・テーブルから ID のリストを返すように設計する必要があります。これらのID により、最終的な MDX 結果で考慮されるファクトを制限します。

最終的な結果には、返される ID リスト内に ID があるレコードに対応するファクトのみが含まれます。sqlStatement に SELECT 文を使用する以下の MDX クエリの例について考えてみます。

```
SELECT FROM HOLEFOODS
WHERE [Comments].[H1].[Comments].&[SELECT ID FROM HoleFoods.SalesTransaction WHERE ID IN (1,2,3,4)]
```

外部参照テーブルにクエリを実行することもできます。例えば、以下のような単純なテーブルがあるとします。

テーブル C-1: キューブ・ソース・テーブルの例

| ID | Name |
|---|---|
| 1 | Bill |
| 2 | Sally |
| 3 | Tom |
| 4 | Mike |
| 5 | Teresa |
| 6 | Alice |

テーブル C-2: 外部参照テーブルの例 (Cohort.Table)

| ID | PatientID | Name | Age |
|---|---|---|---|
| 1 | 1 | Bill | 1 |
| 2 | 6 | Sally | 10 |
| 3 | 4 | Tom | 45 |

以下の %SQLRESTRICT サブクエリは PatientID 値 1 および 6 を返し、MDX クエリに対して考えられる結果として、キューブ・ソース・テーブルの例で ID 1 と 6 を持つ Bill と Alice のみが残ります。

```
%SQLRESTRICT.&[SELECT PatientID From Cohort.Table WHERE Age < 35]
```

上のサンプル・テーブルは非常に単純化されたもので、例を示す目的でのみ使用されています。

%SQLRESTRICT サブクエリは、ソース・テーブルの ID と想定される 1 列の数値を返します。

または、WHERE 節をショートカットとして使用して、ソース・レコードの制限を直接記述することもできます。以下の例を検討します。

```
SELECT FROM HOLEFOODS %FILTER %SQLRESTRICT.&[WHERE ID IN (1,2,3,5,10)]

SELECT FROM HOLEFOODS %FILTER %SQLRESTRICT.&[WHERE Outlet = 24]

SELECT NON EMPTY [DateOfSale].[Actual].[YearSold].Members ON 1
FROM [HOLEFOODS] %FILTER %SQLRESTRICT.&[WHERE YEAR(DateOfSale)='2017']
```

内部的には、この WHERE 節構文は、MDX クエリでクエリされるキューブのソース・クラスをターゲットとして、システムにより SELECT 文を %SQLRESTRICT サブクエリに挿入します。例えば、Holefoods.SalesTransaction は Holefoods キューブのソース・クラスであるため、以下の 2 つのクエリは同等です。

```
SELECT FROM HOLEFOODS %FILTER %SQLRESTRICT.&[SELECT ID FROM Holefoods.SalesTransaction WHERE ID IN
(1,2,3,4)]

SELECT FROM HOLEFOODS %FILTER %SQLRESTRICT.&[WHERE ID IN (1,2,3,4)]
```

## 関連項目

″WHERE 節″ を参照してください。

# CREATE MEMBER 文 (MDX)

現在のセッション内で使用できる計算メンバを作成します。

## 構文および詳細

```
CREATE SESSION MEMBER calc_mem_details, FORMAT_STRING='format_details', SOLVE_ORDER=integer
```

この calc_mem_details は、以下のとおりです。

```
cube_name.[dimension_name].[new_member_name] AS 'value_expression'
```

および、

- cube_name は、このメンバの追加先のキューブの名前です。
- dimension_name は、このメンバの追加先のディメンジョンの名前です。
- new_member_name は、メンバの名前です。このメンバは、キューブ内で既に定義されている場合と定義されていない場合があります。定義されている場合、ここで指定される定義が優先されます。
- value_expression は、通常は他のメンバへの参照に関連して、計算メンバを定義する MDX 式です。詳細は、"WITH 節" を参照してください。
- `FORMAT_STRING='format_details'` は、値の表示方法を指定するオプションの節です。この節は、数値にのみ適用できます。"FORMAT_STRING 節" を参照してください。
- `SOLVE_ORDER=integer` は、この計算メンバを評価する相対順序を指定するための、オプションの節です。この節は、両方の軸についての計算メンバがクエリに含まれている場合にのみ関連します。"SOLVE_ORDER 節" を参照してください。

"識別子" も参照してください。

MDX シェルを使用するとセッションが開始され、シェルを終了するとセッションが終了します。このセッションの間に CREATE MEMBER 文を使用すると、セッションが終了するまで、または DROP MEMBER 文を使用するまで、作成したメンバを使用できます。

## 例

まず、MDX シェルで以下のように、`Patients` キューブに新しいメンバを定義します。

```
>>CREATE SESSION MEMBER patients.MEASURES.scoresquared AS 'MEASURES.[test score]*MEASURES.[test score]'


-------------------------------------------------------------------------------
Elapsed time:       .013701s
```

次に、以下のように、その新しいメジャーをクエリで使用します。

```
>>SELECT MEASURES.scoresquared ON 0, aged.[age group].MEMBERS ON 1 FROM patients
                               scoresquared
1 0 to 29                     66,801,054,681
2 30 to 59                    61,070,271,376
3 60+                          9,120,632,004
-------------------------------------------------------------------------------
Elapsed time:       .016856s
```

## 関連項目

- WITH 節
- FORMAT_STRING 節

- SOLVE_ORDER 節
- DROP MEMBER 文

# CREATE SET 文 (MDX)

現在のセッション内で使用できる名前付きセットを作成します。

## 構文および詳細

```
CREATE SESSION SET cube_name.set_name AS 'set_expression'
```

- cube_name は、このセットの追加先のキューブの名前です。
- set_name は、このセットの名前を付ける引用符なしの文字列です。後で、同じセッション内の任意の MDX クエリで、セット式の代わりにこのセット名を使用できます。
- set_expression は、セットを参照する式です。

MDX シェルを使用するとセッションが開始され、シェルを終了するとセッションが終了します。このセッションの間に CREATE SET 文を使用すると、セッションが終了するまで、または DROP SET 文を使用するまで、作成したメンバを使用できます。

## 例

まず、MDX シェルで以下のように、**Patients** キューブに新しい名前付きセットを定義します。

```
>>CREATE SESSION SET patients.testset AS 'birthd.decade.MEMBERS'

-------------------------------------------------------------------------------
Elapsed time:       .014451s
```

次に、以下のように、その名前付きセットをクエリで使用します。

```
>>SELECT MEASURES.[%COUNT] ON 0, testset ON 1 FROM patients
                            Patient Count
 1 1910s                                71
 2 1920s                               223
 3 1930s                               572
 4 1940s                               683
 5 1950s                             1,030
 6 1960s                             1,500
 7 1970s                             1,520
 8 1980s                             1,400
 9 1990s                             1,413
10 2000s                             1,433
11 2010s                               155
-------------------------------------------------------------------------------
Elapsed time:       .018745s
```

## 関連項目

- WITH 節
- DROP SET 文

# DRILLFACTS 文 (MDX)

キューブによって定義されているファクトおよびディメンジョン・テーブルを使用して、指定された SELECT 文の結果の最初のセルに関連付けられている最下位レベルのデータを表示します。

## 構文および詳細

```
DRILLFACTS select_statement
```

または、以下のようにします。

```
DRILLFACTS select_statement RETURN fieldname1, fieldname2, ...
```

または、以下のようにします。

```
DRILLFACTS select_statement RETURN fieldname1, ... %ORDER BY fieldname3, ...
```

以下は、この指定の説明です。

- select_statement は、SELECT を使用する文です。
- fieldname1、fieldname2、fieldname3、fieldname4 などは、キューブによって定義されているファクト・テーブル・クラスのフィールドの名前です。

RETURN 節を指定しないと、クエリは、レコードの ID を返します。

%ORDER BY 節は、MDX に対する InterSystems 拡張機能です。この節は、表示されているレコードの並べ替え方法を指定します。

RETURN および %ORDER BY の詳細は、"DRILLTHROUGH 文" を参照してください。

システムは、内部で SQL クエリを構築して使用します。

重要　SELECT 文が複数のデータ・セルを返した場合、リストに表示されるのは、最初のセルに関連付けられたフィールドのみです。

## 例

最初の例は、RETURN を使用しません。したがって、キューブに定義されている既定のリストが使用されます。

```
DRILLFACTS SELECT diagd.osteoporosis ON 0 FROM patients
```

```
   #  ID
   1: 7
   2: 13
   3: 42
   4: 123
   5: 140
...
```

次の例は、RETURN 節を使用します。

```
drillfacts select diagd.osteoporosis on 0 from patients return dxcolor->dxcolor, dxage,
dxpatgroup->dxpatgroup
```

```
   #  DxColor        DxAge          DxPatGroup
   1: Orange         7              Group A
   2: Red            11             Group B
   3: <null>         30             Group A
   4: Purple         58             Group A
   5: Purple         62             None
...
```

## 関連項目

- DRILLTHROUGH 文

# DRILLTHROUGH 文 (MDX)

指定された SELECT 文の結果の最初のセルに関連付けられた最下位レベルのデータを表示します。

## 構文および詳細

```
DRILLTHROUGH optionalmaxrows select_statement
```

または、以下のようにします。

```
DRILLTHROUGH optionalmaxrows select_statement RETURN fieldname1, fieldname2, ...
```

または、以下のようにします。

```
DRILLTHROUGH optionalmaxrows select_statement RETURN fieldname1, ... %ORDER BY fieldname3, ...
```

または、以下のようにします。

```
DRILLTHROUGH optionalmaxrows select_statement %LISTING [listingname]
```

以下は、この指定の説明です。

- optionalmaxrows (オプション) は、MAXROWS integer の形式です。

  この引数は、返される行の最大数を指定します。既定値は 1,000 です。
- select_statement は、SELECT を使用する文です。
- fieldname1、fieldname2、fieldname3、fieldname4 (以降同様) は、キューブで使用されるベース・クラスにあるフィールドの名前です。
- listingname は詳細リストの名前です。このリストが既に定義されており、ユーザがそれを使用する権限を持っている必要があります。この名前にスペースが含まれていない場合は、名前を囲む角括弧を省略することができます。リストの名前は大文字と小文字が区別されます。

RETURN 節で、表示するフィールドを指定します。%ORDER BY 節は、MDX に対する InterSystems 拡張機能です。この節が含まれている場合、この節で、表示されているレコードの並べ替え方法を指定します。

%LISTING 節は、もう 1 つの、MDX に対する InterSystems 拡張機能です。この節では、使用するリストを指定します。この節を指定すると、そのリストで指定されているフィールドが表示されます。

RETURN 節または %LISTING 節を指定しないと、クエリは、キューブで定義された既定のリストを使用します。

システムは、内部で SQL クエリを構築して使用します。

重要　SELECT 文が複数のデータ・セルを返した場合、リストに表示されるのは、最初のセルに関連付けられたフィールドのみです。

## RETURN および ORDER BY の追加オプション

RETURN 節および %ORDER BY 節では、以下の点に注意してください。

- 矢印構文を使用すると、別のテーブルのプロパティを参照できます。"暗黙結合 (矢印構文)" を参照してください。
- エイリアスを組み込むことができます。
- 標準的な SQL 関数および InterSystems IRIS® データ・プラットフォーム関数を使用できます。標準的な SQL 関数を使用するには、これを括弧で囲み、関数名がフィールド名と解釈されないようにします。InterSystems SQL 関数は、パーセント文字 (%) が先頭にあるため、括弧で囲む必要はありません。

- field_name ではなく、source.field_name を使用すると、より高度な SQL 機能を使用できます。

以下はその例です。

```
... RETURN %ID,%EXTERNAL(Field1) F1,'$'||source.Sales Sales
```

任意のリストの最初の行は、フィールド名またはそのエイリアスを示すヘッダです。リストの各ヘッダの下にデータ列があります。この場合、列は以下のようになります。

- %ID ー この列には %ID フィールドが表示されます。
- F1 ー この列は、InterSystems SQL %EXTERNAL 関数を使用して、Field1 フィールドの値を DISPLAY 形式で返します。
- Sales ー この列には、Sales フィールドが表示され、先頭にドル記号 ($) が付加されています。

## 例

最初の例は、RETURN を使用しません。したがって、キューブに定義されている既定のリストが使用されます。

```
DRILLTHROUGH SELECT homed.Magnolia ON 1 FROM patients

   #  PatientID  Age        Gender     TestScore  HomeCity   DoctorGrou
   1: SUBJ_10161 0          F          76         3          I
   2: SUBJ_10330 0          F                     3          II
   3: SUBJ_10554 0          F          68         3          II
   4: SUBJ_10555 0          F          78         3          II
   5: SUBJ_10686 0          F          91         3          I
...
```

次の例は、MAXROWS 引数を使用します。

```
DRILLTHROUGH MAXROWS 3 SELECT homed.Magnolia ON 1 FROM patients

   #  PatientID  Age        Gender     TestScore  HomeCity   DoctorGrou
   1: SUBJ_10161 0          F          76         3          I
   2: SUBJ_10330 0          F                     3          II
   3: SUBJ_10554 0          F          68         3          II
```

次の例は、RETURN 節を使用します。

```
DRILLTHROUGH SELECT homed.Magnolia ON 1 FROM patients RETURN Gender, HomeCity->PostalCode
   #  Gender         PostalCode
   1: F              34577
   2: F              34577
   3: F              34577
   4: F              34577
   5: F              34577
...
```

次の例は、%ORDER BY 節も使用します。

```
DRILLTHROUGH SELECT homed.Magnolia ON 1 FROM patients RETURN PatientID, Age, Gender %ORDER BY Age

   #  PatientID      Age            Gender
   1: SUBJ_101616    0              F
   2: SUBJ_102705    0              M
   3: SUBJ_103210    0              M
   4: SUBJ_103300    0              F
   5: SUBJ_103972    0              M
...
```

最後の例は、2 つのフィールド名で %ORDER BY 節を使用し、順序を指定します。

```
DRILLTHROUGH SELECT homed.Magnolia ON 1 FROM patients RETURN PatientID, Age, Gender %ORDER BY Gender,
 Age

   #  PatientID      Age            Gender
   1: SUBJ_101616    0              F
   2: SUBJ_103300    0              F
   3: SUBJ_105548    0              F
   4: SUBJ_105556    0              F
   5: SUBJ_106865    0              F
...
```

この場合、レコードはまず性別で並べ替えられます。各性別の中で、さらに年齢で並べ替えられます。

# DROP MEMBER 文 (MDX)

現在のセッションでこれまでに定義された計算メンバを削除します。

## 構文および詳細

```
DROP MEMBER cube_name.calculated_member_expression
```

- cube_name は、このメンバの追加先のキューブの名前です。
- calculated_member_expression は、メンバを参照する式です。通常、calculated_member_expression の形式は、`MEASURES.new_measure_name` です。

MDX シェルを使用するとセッションが開始され、シェルを終了するとセッションが終了します。このセッションの間に CREATE MEMBER 文を使用すると、セッションが終了するまで、または DROP MEMBER 文を使用するまで、作成したメンバを使用できます。

## 例

```
>>DROP MEMBER patients.MEASURES.avgscore

---------------------------------------------------------------------------
Elapsed time:       .011952s
```

## 関連項目

- CREATE MEMBER 文

# DROP SET 文 (MDX)

現在のセッションでこれまでに定義された名前付きセットを削除します。

## 構文および詳細

```
DROP SET cube_name.set_name
```

- cube_name は、このメンバの追加先のキューブの名前です。
- set_name は、このセットの名前を付ける引用符なしの文字列です。

MDX シェルを使用するとセッションが開始され、シェルを終了するとセッションが終了します。このセッションの間に CREATE SET 文を使用すると、セッションが終了するまで、または DROP SET 文を使用するまで、作成したセットを使用できます。

## 例

```
>>DROP SET patients.testset

-------------------------------------------------------------------------------
Elapsed time:       .011825s
```

## 関連項目

- CREATE SET 文

# FORMAT_STRING 節 (MDX)

計算メンバの定義に使用します。この節では、データの表示形式を指定します。

## 構文および詳細

この節は、CREATE MEMBER 文または WITH 節で計算メンバを定義するときに使用できます。

```
FORMAT_STRING = 'positive_piece;negative_piece;zero_piece;missing_piece;'
```

以下は、この指定の説明です。

- positive_piece では、正の値の表示方法を制御します。
- negative_piece では、負の値の表示方法を制御します。
- zero_piece では、ゼロの表示方法を制御します。
- missing_piece では、欠落値の表示方法を制御します。これは、現在使用されていません。

各部分はリテラルであり、以下のベース・ユニットのいずれかを含む 1 つ以上の文字で構成されます。

| ベース・ユニット | 意味 | 例 |
|---|---|---|
| # | 1000 単位の区切り文字なしで値を表示します。小数点以下の桁は含まれません。 | 12345 |
| #,# | 1000 単位の区切り文字を使用して値を表示します。小数点以下の桁は含まれません。これは正の数値に対する既定の表示形式です。 | 12,345 |
| #.## | 1000 単位の区切り文字なしで値を表示します。小数点以下 2 桁 (またはピリオド後のシャープ記号の数に応じた小数点以下桁数) まで含まれます。ピリオド以降のシャープ桁は必要なだけ指定できます。 | 12345.67 |
| #,#.## | 1000 単位の区切り文字を使用して値を表示します。小数点以下 2 桁 (またはピリオド後のシャープ記号の数に応じた小数点以下桁数) まで含まれます。ピリオド以降のシャープ桁は必要なだけ指定できます。 | 12,345.67 |

ベース・ユニットの前後には、以下のように、追加の文字を組み込むことができます。

- パーセント記号 (%) を組み込むと、値がパーセントとして表示されます。つまり、値に 100 が乗じられ、指定した位置にパーセント記号 (%) が表示されます。
- その他の文字も、指定の位置に指定どおり表示されます。

クエリに複数の計算メンバが含まれていて、それらの書式文字列が異なる場合は、SOLVE_ORDER 節で、どの書式文字列を使用するかを制御します。(この節は、両方の軸についての計算メンバがクエリに含まれている場合にのみ関連します。)

## 例

以下のテーブルに例を示します。

| 例 | 論理値 | 表示値 |
|---|---|---|
| FORMAT_STRING='#,#;(#,#);'<br>これは、既定の数値表示方法に対応します。 | 6608.9431 | 6,609 |
| | -1234 | (1234) |
| FORMAT_STRING='#,#.###;' | 6608.9431 | 6,608.943 |
| FORMAT_STRING='#%;' | 6 | 600% |
| FORMAT_STRING='$#,#;($#,#);' | 2195765 | $2,195,765 |
| | -3407228 | ($3,407,228) |

## 関連項目

- CREATE MEMBER 文
- SOLVE_ORDER 節
- WITH 節

# SELECT 文 (MDX)

クエリを実行し、その結果を返します。このセクションでは、基本的な構文について説明します。

## 構文および詳細

```
SELECT set_expression ON 0, set_expression ON 1 FROM cube_name
```

以下は、この指定の説明です。

- ON 節は、オプションの軸指定です。その形式は以下のようになります。

  ```
  set_expression ON axis_name_or_number
  ```

  - set_expression は、セットに対して評価される式です。
  - axis_name_or_number は COLUMNS (または同等の 0) あるいは ROWS (または同等の 1) です。

  ON 節は指定しないことも、1 つまたは 2 つ指定することもできます。ON 節を 2 つ指定する場合は、それらを任意の順序で指定できます。

  MDX の他の実装では、ROWS を指定した場合、COLUMNS も指定する必要があります。InterSystems MDX では、ROWS を指定して COLUMNS を指定しない場合、システムが COLUMNS に対して ON 節を自動的に生成します。この節はカウント・メジャーを使用します。

- cube_name は、使用するキューブの名前です。セット式は、そのキューブ内で意味を持つものにする必要があります。

ON 節を省略すると、MDX はキューブ内のレコード数を返します。複合キューブの場合は、その複合キューブに結合されたすべてのキューブの合計数が返されます。

結果の任意のセルについて、使用する値を決定するために、システムは列に使用されるメンバおよび行のメンバ (使用される場合) の交差部分を検索します。

- 1 つのメンバがメジャーで、もう 1 つのメンバがメジャーでない場合、システムは前者のメンバのそのメジャーの値を検索します。例えば、1 つのメンバが Ave Age メジャーで、もう 1 つのメンバが郵便番号 34577 である場合、対応するデータ・セルには、郵便番号が 34577 の患者の平均年齢が含まれます。
- どちらのメンバもメジャーでない場合は、既定のメジャー (通常、%COUNT) が使用されます。例えば、1 つのメンバが性別 F で、もう 1 つのメンバが郵便番号 34577 である場合、対応するデータ・セルには、郵便番号が 34577 の全女性患者数が含まれます。

## 例

以下の単純な例は、郵便番号別の患者数を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.MEMBERS ON 1 FROM patients
                                Patient Count
1 32006                                 2,272
2 32007                                 1,111
3 34577                                 3,399
4 36711                                 1,069
5 38928                                 2,149
```

以下の例では、patients2 キューブに Home Zip レベルは含まれていません。代わりに、このキューブには、cities という別のキューブを指す Home City というリレーションシップがあります。このクエリは、このリレーションシップを使用します。

```
SELECT MEASURES.[%COUNT] ON 0, city.cityd.city.members ON 1 FROM patients2
```

```
                              Patient Count
1 Cedar Falls                         1,097
2 Centerville                         1,136
3 Cypress                             1,124
4 Elm Heights                         1,089
5 Juniper                             1,133
6 Magnolia                            1,063
7 Pine                                1,124
8 Redwood                             1,083
9 Spruce                              1,151
```

## 関連項目

- DRILLTHROUGH 文
- WHERE 節
- WITH 節

# SET 文 (MDX)

開発用に現在のセッションで使用するピボット変数を作成します。この文は MDX シェル内でのみ使用できます。

## 構文および詳細

```
SET variable_name variable_value
```

以下は、この指定の説明です。

- variable_name はピボット変数の名前です (先頭に `$variable.` を使用しない構文)。名前の大文字と小文字は区別されません。変数参照時に大文字と小文字の違いが無視されます。名前の大文字と小文字のみが異なる変数を複数作成することはできません。

  このピボット変数はどのキューブからも独立しており、現在のシェル・セッション内でのみ使用できることに注意してください。

- variable_value は、ピボット変数の値です。

  ピボット変数の詳細は、"ピボット変数の定義と使用" を参照してください。

## 例

以下の MDX シェル・セッションは例を示しています。

```
>>set mypivotvar colord.red
mypivotvar is: colord.red
>>select $variable.mypivotvar on 0 from patients

                                    Red
                                     129
```

# SOLVE_ORDER 節 (MDX)

計算メンバの定義に使用します。この節では、この計算メンバの定義を適用する順序を、他の計算メンバに対して相対的に指定します。この節は、両方の軸についての計算メンバがクエリに含まれている場合にのみ関連します。

## 構文および詳細

この節は、CREATE MEMBER 文または WITH 節で計算メンバを定義するときに使用できます。

```
SOLVE_ORDER=integer
```

SOLVE_ORDER キーワードは、大文字と小文字が区別されません。integer には、リテラルの整数を指定します。既定値は 0 です。

この節は、計算メンバを列に、競合する計算メンバを行に持つクエリで使用すると便利です。この節は、システムによるセルの値およびセルに適用する形式の決定方法に影響します。既定では、列が FORMAT_STRING を決定します。

行によって形式と値を決定する場合は、その SOLVE_ORDER が、行に使用する計算メンバのものよりも大きい必要があります。

SOLVE_ORDER の大きい計算メンバは最後に評価され、結果を制御しますが、1 つの例外があります。行または列で %CELL 関数が使用されている場合、その暗黙的な既定の SOLVE_ORDER は 10 になります。

行と列の両方で %CELL が使用されており、行によって値と書式文字列を決定する場合は、行の SOLVE_ORDER を 11 に設定します。

行と列のメンバの SOLVE_ORDER が同じ場合は、既定の場合と同様に、列のメンバが結果を制御します。

計算メジャーが他の計算メジャーに依存する場合、システムは依存関係を認識して適切な順序でメジャーを評価するため、それらのメジャーに SOLVE_ORDER を使用する必要はありません。

## 関連項目

- CREATE MEMBER 文
- FORMAT_STRING 節
- WITH 節

# WHERE 節 (MDX)

SELECT 文にフィルタを適用します。SELECT 文の結果をスライスする方法について説明します。

## 構文および詳細

```
select_statement WHERE set_expression
```

以下は、この指定の説明です。

- select_statement は、SELECT を使用する文です。
- set_expression は、メンバまたはタプルのセットを返す式です。

  set_expression の代わりに、メジャー検索式を使用できます。この場合の WHERE の動作については、例を参照してください。

システムでは、必要に応じてタイプが自動的に変換されるため、セット式の代わりに、単一のメンバ式またはタプル式を使用することもできます。

重要

各セット要素は別のスライサ軸として使用され、(すべての %FILTER 節の) すべてのスライサ軸の結果がいっしょに集約されます。これは軸のたたみ込みのプロセスです (フィルタはクエリ軸と見なされます)。軸のたたみ込みを実行すると、どのスライサ軸にも NULL の結果がないソース・レコードは複数回カウントされます。

軸のたたみ込みでは、キューブの定義での指定に従い、対象としているメジャーの集約メソッドに基づいて値が組み合わされます (ここの例では、%COUNT が追加されます)。

詳細は、"Business Intelligence クエリ・エンジンの仕組み" の "軸のたたみ込み" を参照してください。

## 例

以下の 2 つの SELECT 文の例を比較します。1 つは WHERE 節を使用した文、もう 1 つは WHERE 節を使用していない文です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients
                            Patient Count
1 Cedar Falls                       1,039
2 Centerville                       1,107
3 Cypress                           1,096
4 Elm Heights                       1,093
5 Juniper                           1,150
6 Magnolia                          1,092
7 Pine                              1,157
8 Redwood                           1,125
9 Spruce                            1,141
```

前のクエリは、各都市の患者数を表示します。一方、以下のクエリを考えてみます。これは、各都市の男性の患者数を示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients WHERE gend.male

                            Patient Count
1 Cedar Falls                         509
2 Centerville                         569
3 Cypress                             517
4 Elm Heights                         531
5 Juniper                             574
6 Magnolia                            527
7 Pine                                569
8 Redwood                             553
9 Spruce                              557
```

WHERE 節に複数の項目がある場合の効果を例示するために、まず以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients WHERE colord.green
```

```
                              Patient Count
1 Cedar Falls                           137
2 Centerville                           129
3 Cypress                               150
4 Elm Heights                           128
5 Juniper                               126
6 Magnolia                              143
7 Pine                                  155
8 Redwood                               148
9 Spruce                                147
```

ここで以下のクエリを考えてみます。これは、WHERE 節のセット要素として、gend.male と colord.green の両方を使用しています。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients WHERE {gend.male,colord.green}
```

```
                              Patient Count
1 Cedar Falls                           646
2 Centerville                           698
3 Cypress                               667
4 Elm Heights                           659
5 Juniper                               700
6 Magnolia                              670
7 Pine                                  724
8 Redwood                               701
9 Spruce                                704
```

例えば、Cedar Falls の結果を比較すると、このクエリが、男性患者の結果および好きな色が緑である患者の結果を加算していることがわかります。代わりに、好きな色が緑である男性患者の結果を表示するには、以下のように、WHERE 節で、CROSSJOIN とタプル式のいずれかを使用します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients
WHERE CROSSJOIN(gend.male,colord.green)
```

```
                              Patient Count
1 Cedar Falls                            56
2 Centerville                            65
3 Cypress                                80
4 Elm Heights                            59
5 Juniper                                73
6 Magnolia                               74
7 Pine                                   82
8 Redwood                                70
9 Spruce                                 74
```

以下の例は、WHERE 節でタプル式を使用しています。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients WHERE (gend.male,aged.60)
                              Patient Count
1 Cedar Falls                            3
2 Centerville                            9
3 Cypress                                7
4 Elm Heights                            1
5 Juniper                                8
6 Magnolia                               2
7 Pine                                   5
8 Redwood                                6
9 Spruce                                 3
```

WHERE 節は、特定のメジャーを表示する方法として使用することもできます。

```
SELECT gend.gender.MEMBERS ON 0 FROM patients WHERE MEASURES.[avg test score]
                               Female                Male
All Patients                     74.78                74.46
```

ただし、メジャー名は表示されません。

WHERE 節をメジャー検索式で使用する場合、この節は、ファクト・テーブルの行のうち、指定された条件を満たした行のみを使用します (メジャー検索式は、インターシステムズによる MDX への拡張機能で、ファクト・テーブル自体のメジャー値を考慮するものです)。

```
SELECT MEASURES.[%COUNT] ON 0 FROM patients WHERE %SEARCH.&[[MEASURES].[age]<10]
                              Patient Count
                                      1,370
```

## 関連項目

"%FILTER 節" を参照してください。

# WITH 節 (MDX)

SELECT 文で使用する 1 つ以上の計算メンバ、名前付きセット、またはパラメータを定義します。

## 構文および詳細

```
WITH with_details1 with_details2 ... select_statement
```

以下は、この指定の説明です。

- select_statement は、SELECT を使用する文です。
- with_details1、with_details2 (以降同様) の構文は、以下のいずれかです。

  ```
  MEMBER calc_mem_definition
  ```

  または、以下のようになります。

  ```
  SET named_set_definition
  ```

  または、以下のようになります。

  ```
  %PARM named_parameter_definition
  ```

  これらの従属節は、単一の WITH 節に組み合わせることができます。

  Tipヒン WITH 従属節どうしを区切るコンマはありません。WITH 節と SELECT 文の間にもコンマはありません。
  ト

以下のセクションで、MEMBER、SET、%PARM の各従属節の詳細を説明します。

## WITH MEMBER

WITH 節では、MEMBER は、クエリで使用する計算メンバを定義します。MEMBER 従属節の構文は以下のとおりです。

```
MEMBER calc_mem_details, FORMAT_STRING='format_details', SOLVE_ORDER=integer
```

この calc_mem_details は、以下のとおりです。

```
cube_name.[dimension_name].[new_member_name] AS 'value_expression'
```

および、

- cube_name は、キューブの名前です。
- dimension_name は、ディメンジョンの名前です。
- new_member_name は、メンバの名前です。このメンバは、キューブ内で既に定義されている場合と定義されていない場合があります。定義されている場合、ここで指定される定義が優先されます。
- value_expression は、通常は他のメンバへの参照に関連して、計算メンバを定義する MDX 式です。

  以下はその例です。

  ```
  MEASURES.[test score]/MEASURES.[%COUNT]
  ```

  この計算メンバを使用するコンテキストでは、まず、そのコンテキストの `Test Score` および `%COUNT` メジャーが評価され、次に除算が実行されます。

別の例を示します。

```
%OR({colord.red,colord.blue,colord.yellow})
```

この新しいメンバは、ファクト・テーブルのレコードのうち、colord ディメンジョンの red、yellow、blue の各メンバに対応するすべてのレコードを参照します。

その他のバリエーションは、"計算メンバの定義" を参照してください。

- FORMAT_STRING='format_details' は、値の表示方法を指定するオプションの節です。この節は、数値にのみ適用できます。"FORMAT_STRING 節" を参照してください。
- SOLVE_ORDER=integer は、この計算メンバを評価する相対順序を指定するための、オプションの節です。この節は、両方の軸についての計算メンバがクエリに含まれている場合にのみ関連します。"SOLVE_ORDER 節" を参照してください。

最初の例は、WITH 節内で定義された計算メンバを表示します。

```
WITH MEMBER MEASURES.avgage AS 'MEASURES.age/MEASURES.%COUNT'
SELECT MEASURES.avgage ON 0, diagd.MEMBERS ON 1 FROM patients
```

```
1 None                                  33.24
2 asthma                                34.79
3 CHD                                   67.49
4 diabetes                              57.24
5 osteoporosis                          79.46
```

システムは、まず、Age メジャーと %COUNT メジャーを評価してから、avgage メジャーの除算を実行します。

## WITH SET

WITH 節では、SET は、クエリで使用する名前付きセットを定義します。SET 従属節の構文は以下のとおりです。

```
SET set_name AS 'set_expression'
```

- set_name は、このセットの名前を付ける引用符なしの文字列です。
- set_expression は、セットを参照する式です。

以下の例は、WITH 節内で定義された名前付きセットを表示します。

```
WITH SET testset AS '{homed.city.members}' SELECT MEASURES.[%COUNT] ON 0, testset ON 1 FROM patients
```

```
                                Patient Count
1 Cedar Falls                           1,045
2 Centerville                           1,069
3 Cypress                               1,150
4 Elm Heights                           1,104
5 Juniper                               1,155
6 Magnolia                              1,111
7 Pine                                  1,138
8 Redwood                               1,111
9 Spruce                                1,117
```

## WITH %PARM

WITH 節では、%PARM は、クエリで使用する名前付きパラメータを定義します。%PARM 従属節の構文は以下のとおりです。

```
%PARM parameter_name AS 'value:default_value'
```

または、以下のようになります。

```
%PARM parameter_name AS 'value:default_value,caption:label'
```

- parameter_name は、パラメータの名前です。

- default_value は、パラメータの既定値です。
- label は、このパラメータの値の入力を求めるときに使用するキャプションです。

  MDX シェル内でクエリを実行すると、名前付きパラメータがあればその値を入力するように求められます。

次に、クエリ自体の中のパラメータを参照するには、@parameter_name を使用します。

以下はその例です。

```
>>WITH %PARM c as 'value:Pine' select homed.[city].@c ON 0 FROM patients
Please supply parameter value(s) for this query:
C [Pine]:

                                    Pine
                                    1,073
------------------------------------------------------------------------
Elapsed time:       2.136337s
>>WITH %PARM c as 'value:Pine' select homed.[city].@c ON 0 FROM patients

Please supply parameter value(s) for this query:
C [Pine]:Magnolia

                                 Magnolia
                                    1,113
------------------------------------------------------------------------
Elapsed time:       2.627897s
>>WITH %PARM c as 'value:Pine,caption:city' select homed.[city].@c ON 0 FROM patients

Please supply parameter value(s) for this query:
city [Pine]:

                                    Pine
                                    1,073
------------------------------------------------------------------------
Elapsed time:       2.235228s
>>WITH %PARM c AS 'value:5,caption:count' SELECT TOPCOUNT(birthd.decade.MEMBERS, @c) ON 1 FROM patients


Please supply parameter value(s) for this query:
count [5]:3


1 1970s                             1,593
2 1960s                             1,505
3 2000s                             1,442
------------------------------------------------------------------------
Elapsed time:       1.207581s
```

## 関連項目

WITH 節は、要素を定義するクエリの間だけ使用可能な要素を定義します。

セッションの間中ずっと使用できる計算メンバおよび名前付きセットを定義するには、以下の文を使用します。

- CREATE MEMBER 文
- CREATE SET 文

(すべてのセッションで使用可能な) キューブ定義の一部として計算メンバおよび名前付きセットを定義する方法の詳細は、"InterSystems Business Intelligence のモデルの定義" を参照してください。

# MDX 関数

この参照セクションでは、Business Intelligence でサポートされている MDX 関数について説明します。

"BI サンプルのアクセス方法" も参照してください。

# %ALL (MDX)

メンバが属している階層を使用する ROW コンテキストおよび COLUMN コンテキストを無視しながら、このメンバを使用できるようにします。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.%ALL
```

以下は、この指定の説明です。

- member_expression は、メンバに評価される式です。

この関数を使用すると、階層内のあるメンバをその階層内の別のメンバと比較する(例えば、ある製品をすべての製品と比較する) 計算メンバを作成できます。

## 例

以下のような値を計算することが必要になる場合もあります。

- すべての製品と比較したある製品の割合
- 別の製品と比較したある製品の割合

例えば、以下のクエリは、各年齢グループの患者数と等しい計算メンバを、すべての年齢グループの患者の割合として使用します。

```
WITH MEMBER MEASURES.[pct age grps] AS 'aged.CURRENTMEMBER/aged.[all patients].%ALL', FORMAT_STRING='#.##'

SELECT {MEASURES.[%COUNT],MEASURES.[pct age grps]} ON 0,
aged.h1.[age group].MEMBERS ON 1 FROM patients
```

```
                        Patient Count         pct age grps
1 0 to 29                       4,216                 0.42
2 30 to 59                      4,212                 0.42
3 60+                           1,572                 0.16
```

この計算メンバは、AgeD ディメンジョンの現在のメンバとして定義され、そのディメンジョンの All メンバで除算されます。

```
aged.CURRENTMEMBER/aged.[all patients].%ALL
```

一方、以下のクエリを考えてみます。ここでは、計算メンバが %ALL 関数を使用しません。

```
WITH MEMBER MEASURES.[BADpct age grps] AS 'aged.CURRENTMEMBER/aged.[all patients]', FORMAT_STRING='#.##'

SELECT {MEASURES.[%COUNT],MEASURES.[BADpct age grps]} ON 0,
aged.h1.[age group].MEMBERS ON 1 FROM patients
```

```
                        Patient Count      BADpct age grps
1 0 to 29                       4,216                 1.00
2 30 to 59                      4,212                 1.00
3 60+                           1,572                 1.00
```

この場合、各行の aged.[all patients] の値は、aged.CURRENTMEMBER の値と同じです。これは、この行のメンバが aged.[all patients] と同じ階層に属しているためです。

%ALL 関数が、他の階層のメンバによって指定されるコンテキストを考慮しないことに注意してください (この関数は、この関数で使用するメンバに関連付けられた階層のみを無視します)。以下はその例です。

```
WITH MEMBER MEASURES.[pct age grps] AS 'aged.CURRENTMEMBER/aged.[all patients].%ALL', FORMAT_STRING='#.##'

SELECT CROSSJOIN(gend.MEMBERS,{MEASURES.[%COUNT],MEASURES.[pct age grps]}) ON 0,
aged.h1.[age group].MEMBERS ON 1 FROM patients
```

```
                  Patient Coun   pct age grps   Patient Coun   pct age grps
1 0 to 29                1,985           0.39          2,231           0.45
2 30 to 59               2,123           0.42          2,089           0.42
3 60+                      926           0.18            646           0.13
```

ここで、最初の 2 つの `Patient Count` 列および `pct age grps` 列は女性患者に対応し、2 番目の 2 つの列は男性患者に対応します。それぞれの `pct age grps` 列は、その性別の患者数を、その性別のすべての年齢グループに対する割合で示します。

また、%ALL 関数が、その階層のメンバが WHERE 節または FILTER 節で使用されている場合に、これらのメンバを無視しないことに注意してください。つまり、%ALL 関数は、クエリに適用されるすべてのフィルタを全面的に尊重します。以下はその例です。

```
WITH MEMBER MEASURES.[pct of all ages] AS 'aged.CURRENTMEMBER/aged.[all patients].%ALL',
FORMAT_STRING='#.##'
SELECT {MEASURES.[%COUNT],MEASURES.[pct of all ages]} ON 0,
aged.h1.[age group].MEMBERS ON 1 FROM patients
WHERE aged.h1.[age group].[0 to 29]
```

```
                       Patient Count      pct of all ages
1 0 to 29                      4,216                1.00
2 30 to 59                         *                   *
3 60+                              *                   *
```

## 関連項目

- %MDX

# %CELL (MDX)

ピボット・テーブルの別のセルの値を、位置別に返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値または文字列を返します。

## 構文および詳細

```
%CELL(relative_column_position,relative_row_position)
```

以下は、この指定の説明です。

- relative_column_position は整数です。現在の列には 0、前の列 (左の列) には -1、次の列 (右の列) には 1 を使用します (以降同様)。
- relative_row_position は整数です。現在の行には 0、前の行 (上の行) には -1、次の行 (下の行) には 1 を使用します (降同様)。

指定したセルの値が返されます。セルに値がない場合は null が返されます。

システムは、クエリの残りの部分を解決してから %CELL を評価します。そのため、この関数は、別の関数で使用される式の中で使用することはできません。

## 例

以下の例は、指定された期間の降雨量データおよび累積降雨量データを表示します。

```
SELECT {MEASURES.[Rainfall Inches],%CELL(-1,0)+%CELL(0,-1)} ON 0, {dated.year.1960:1970} ON 1 FROM
cityrainfall
```

| | Rainfall Inches | Expression |
|---|---|---|
| 1 1960 | 177.83 | 177.83 |
| 2 1961 | 173.42 | 351.25 |
| 3 1962 | 168.11 | 519.36 |
| 4 1963 | 188.30 | 707.66 |
| 5 1964 | 167.58 | 875.24 |
| 6 1965 | 175.23 | 1,050.47 |
| 7 1966 | 182.50 | 1,232.97 |
| 8 1967 | 154.44 | 1,387.41 |
| 9 1968 | 163.97 | 1,551.38 |
| 10 1969 | 184.84 | 1,736.22 |
| 11 1970 | 178.31 | 1,914.53 |

ここでの既定のラベルは `Expression` です。%LABEL を使用して、より適したラベルを指定できます。以下はその例です。

```
SELECT {MEASURES.[Rainfall Inches],%LABEL((%CELL(-1,0)+%CELL(0,-1)),"Cumulative Inches")} ON 0,
{dated.year.1960:1970} ON 1 FROM cityrainfall
```

| | Rainfall Inches | Cumulative Inches |
|---|---|---|
| 1 1960 | 177.83 | 177.83 |
| 2 1961 | 173.42 | 351.25 |
| 3 1962 | 168.11 | 519.36 |
| 4 1963 | 188.30 | 707.66 |
| 5 1964 | 167.58 | 875.24 |
| 6 1965 | 175.23 | 1,050.47 |
| 7 1966 | 182.50 | 1,232.97 |
| 8 1967 | 154.44 | 1,387.41 |
| 9 1968 | 163.97 | 1,551.38 |
| 10 1969 | 184.84 | 1,736.22 |
| 11 1970 | 178.31 | 1,914.53 |

## 関連項目

- %CELLZERO

# %CELLZERO (MDX)

ピボット・テーブルの別のセルの値が返されるか、セルに値がない場合は 0 が返されます。この関数は、MDX に対する Business Intelligence 拡張機能です。

## 返りタイプ

この関数は、数値または文字列を返します。

## 構文および詳細

```
%CELLZERO(relative_column_position,relative_row_position)
```

以下は、この指定の説明です。

- relative_column_position は整数です。現在の列には 0、前の列 (左の列) には -1、次の列 (右の列) には 1 を使用します (以降同様)。
- relative_row_position は整数です。現在の行には 0、前の行 (上の行) には -1、次の行 (下の行) には 1 を使用します (降同様)。

指定したセルの値が返されます。セルに値がない場合は 0 が返されます。

## 例

以下のクエリは 3 つのメジャーを使用します。最初のメジャーは指定した期間に販売されたユニット数を表示します。2 番目のメジャーは前回の期間に販売されたユニット数を表示します。これは PREVMEMBER で定義される計算メジャーです。3 番目のメジャーは前回の期間以降に販売されたユニットの変化を表示します。これは %CELLZERO で定義される計算メジャーです。

```
WITH  MEMBER [MEASURES].[UnitsSoldPreviousPeriod]
AS '%LABEL(([DateOfsale].[Actual].CurrentMember.PrevMember,MEASURES.[units sold]),"Units (Prev Period)")'

MEMBER [MEASURES].[Delta Since Prev Period] AS '%CELLZERO(-2,0)-%CELLZERO(-1,0)'
SELECT
{[Measures].[Units Sold],[MEASURES].[UNITSSOLDPREVIOUSPERIOD],[MEASURES].[DELTA SINCE PREV PERIOD]} ON
 0,
[DateOfSale].[Actual].[MonthSold].Members ON 1
FROM [HoleFoods]
%FILTER [PRODUCT].[P1].[PRODUCT CATEGORY].&[Dairy]
```

```
                       Units Sold   Units (Prev Peri   Delta Since Prev
 1 Jan-2009                     *                  *                  0
 2 Feb-2009                     *                  *                  0
 3 Mar-2009                     *                  *                  0
 4 Apr-2009                     1                  *                  1
 5 May-2009                     *                  1                 -1
 6 Jun-2009                     8                  *                  8
 7 Jul-2009                     1                  8                 -7
 8 Aug-2009                     *                  1                 -1
 9 Sep-2009                     *                  *                  0
10 Oct-2009                     *                  *                  0
11 Nov-2009                     *                  *                  0
12 Dec-2009                     *                  *                  0
13 Jan-2010                     1                  *                  1
14 Feb-2010                     *                  1                 -1
15 Mar-2010                     2                  *                  2
...
```

## 関連項目

- %CELL

# %FIRST (MDX)

セットのうち、空でない最初のメンバについて評価された、指定されたメジャー (または他の数値式) の値が返されます。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
%FIRST(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

%FIRST 関数は、指定されたセットの各メンバに対して評価された、欠落のない最初の値を返します。

## 例

以下のクエリは、喘息を持つ患者を誕生年の年代別にグループ化して示します。

```
SELECT MEASURES.[%Count] ON 0, birthd.decade.MEMBERS ON 1 FROM patients WHERE diagd.asthma
```

```
                              Patient Count
 1 1910s                                  *
 2 1920s                                  *
 3 1930s                                  1
 4 1940s                                  9
 5 1950s                                  8
 6 1960s                                 11
 7 1970s                                 12
 8 1980s                                 14
 9 1990s                                 11
10 2000s                                 14
11 2010s                                  4
```

以下のクエリは %FIRST を使用して、前のセットから最初の空でない患者セットを取得します。

```
SELECT MEASURES.[%Count] ON 0, %FIRST(birthd.decade.MEMBERS) ON 1 FROM patients WHERE diagd.asthma
```

```
                              Patient Count
FIRST                                    1
```

## 関連項目

- %LAST

# %KPI (MDX)

KPI またはプラグインから値を返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
%KPI(kpiName,propName,series)
```

または、以下のようになります。

```
%KPI(kpiName,propName,series,parmName1,parmValue1,parmName2,parmValue2)
```

以下は、この指定の説明です。

- kpiName は、KPI またはプラグインの名前です。
- propName は、KPI またはプラグインの <property> 要素の引用符付きの名前です。
- series は、KPI またはプラグインの系列 (行) のオプションの番号または引用符付きの名前です。既定値は 1 です。
- parmName1、parmName2 (以降同様) は、KPI またはプラグインのパラメータのオプションの引用符付きの名前です (多くの場合、これらはフィルタです)。このパラメータ名は大文字と小文字を区別しますので、注意してください。

  フィルタをリストする順序は、KPI には影響しません。

  16 個までのパラメータとその値を指定できます。
- parmValue1、parmValue2 (以降同様) は、名前付きフィルタの対応値です。

%KPI は指定されたすべてのパラメータ値を使用して、指定された series の指定された propName の値を返します。KPI およびプラグインの場合、正規化とローカライズが行われたプロパティ名が値のキャプションになります。

MDX ベースの KPI およびプラグインの場合、特殊な %CONTEXT パラメータを使用して、クエリのコンテキストを考慮させることができます。このパラメータを使用しない場合、クエリのコンテキストは無視されます。その値に対して、以下のフラグの組み合わせを指定します。

- "rows" は、現在のピボット行のコンテキストを使用する必要があることを指定します。
- "columns" は、現在のピボット列のコンテキストを使用する必要があることを指定します。
- "filters" は、現在のピボットのフィルタのコンテキストを使用する必要があることを指定します。
- "all" は、上記をすべて使用する必要があることを指定します (既定)。

パイプ文字 (|) は、"rows|columns" のように、フラグを組み合わせるために使用します。値 "all" は "rows|columns|filters" と同じことを指定します。(%MDX 関数もこのパラメータを使用します。使用例は、%MDX を参照してください。)

重要　%CONTEXT パラメータを使用する場合は、引用符で囲むことに注意してください。また、最後のパラメータとして使用しない限り、このパラメータには値を指定する必要があります。例えば、以下は有効です。

```
%KPI("%DeepSee.Median","MEDIAN",1,"%measure","Amount Sold","%CONTEXT")
```

以下も有効です。

```
%KPI("%DeepSee.Median","MEDIAN",1,"%CONTEXT","all","%measure","Amount Sold")
```

しかし、以下は正しくないため、目的どおりには解釈されません。

```
%KPI("%DeepSee.Median","MEDIAN",1,"%CONTEXT","%measure","Amount Sold")
```

KPI (プラグイン以外) は既定で同期的に実行されますが、非同期的に実行するよう定義することができます。

## 例

以下の例では、DemoMDX KPI の先頭行の `PatCount` プロパティの値を取得します。

```
SELECT %KPI("demomdx","PatCount") ON 0 FROM patients
```

```
                            Patient Count
                                      115
```

以下の例では、%DeepSee.Plugin.Median サンプル・プラグインを使用する計算メジャーを定義します。

```
WITH MEMBER [MEASURES].[Median Amount Sold] AS
'%KPI("%DeepSee.Median","MEDIAN",1,"%measure","Amount Sold","%CONTEXT")'
SELECT NON EMPTY {[Measures].[Amount Sold],[MEASURES].[MEDIAN AMOUNT SOLD]} ON 0,
NON EMPTY [Product].[P1].[Product Name].Members ON 1
FROM [HoleFoods]
```

```
                        Amount Sold   Median Amount Sold
 1 Bagels (dozen)             38.96                 2.95
 2 Bundt Cake             1,632.01                19.95
 3 Calamari (frozen         566.90                22.95
 4 Cheerios (box)           600.11                 3.95
 5 Donuts (dozen)           429.36                 2.95
 6 Free-range Donut       1,310.64                12.95
 7 Fruit Loops (box         772.83                 4.95
 8 Lifesavers (roll         248.96                 1.15
 9 Onion ring               377.25                 4.95
10 Onion ring                28.57                 5.95
11 Penne (box)             176.72                  1.95
12 Pineapple Rings         512.00                  8.95
13 Pretzels (bag)           88.12                  3.95
14 Swiss Cheese (sl        445.10                  5.95
15 Tortellini (froz      1,000.89                  6.95
16 Unsalted Pretzel        316.70                  4.25
17 Ziti (box)              979.43                  4.81
```

## 使用可能なプラグイン・クラス

システムには、%KPI 関数と併用できる複数のプラグインがあります。

### %DeepSee.Distinct

指定されたレベルの個別値の数を取得します。

このプラグインは、プロパティ `DISTINCT` を提供します。このプラグインには、以下のパラメータを使用できます。

| パラメータ | 値 |
|---|---|
| %cube | キューブの論理名。 |
| %level | カウントする個別値を持つレベルまたはリレーションシップの MDX 識別子。例：[DocD].[H1].[Doctor]、[RelatedCubes/Doctors].[DocD] |

このプラグインは、クラス %DeepSee.PlugIn.Distinct で定義されています。

%DeepSee.Median

セルで使用される最下位レベルのすべてのレコード間で、指定されたメジャーの中央値を取得します。

このプラグインは、プロパティ MEDIAN を提供します。このプラグインには、以下のパラメータを使用できます。

| パラメータ | 値 |
|---|---|
| %cube | キューブの論理名。 |
| %measure | 使用する値を持つメジャーの MDX 識別子。例：[MEASURES].[Measure Name] |

このプラグインは、クラス %DeepSee.PlugIn.Median で定義されています。

%DeepSee.Percentile

最下位レベルのすべてのレコード間で、指定されたメジャーのパーセンタイル値を取得します。

このプラグインは、プロパティ PERCENTILE を提供します。このプラグインには、以下のパラメータを使用できます。

| パラメータ | 値 |
|---|---|
| %cube | キューブの論理名。 |
| %measure | 使用する値を持つメジャーの MDX 識別子。例：[MEASURES].[Measure Name] |
| %percentile | 評価するパーセンタイル。既定値は 50 です（プラグインは、50 番目のパーセンタイルを算出します）。 |

このプラグインは、クラス %DeepSee.PlugIn.Percentile で定義されています。

これらのプラグイン・クラスは PLUGINTYPE が "Aggregate" と定義されているため、アナライザまたはウィジェットでプラグインを直接使用することはできません。

## 関連項目

- %MDX

# %LABEL (MDX)

MDX 式が指定されると、同じ式を、行ヘッダまたは列ヘッダとして使用する異なるラベルを付けて返します。%LABEL では、行または列のフォーマットを指定することもできます。この関数は、MDX に対する Business Intelligence 拡張機能です。

## 返りタイプ

この関数は、関数で使用するものと同じタイプの式を返します。

## 構文および詳細

```
%LABEL(MDX_expression,label,format_string,solve_order,cell_style,heading_style)
```

以下は、この指定の説明です。

- MDX_expression は、任意のタイプの MDX 式です。
- label は、行ヘッダまたは列ヘッダの新しいラベルとして使用する文字列です。
- format_string は、値の表示方法を指定するオプションのリテラルです (`"#.##"` など)。"FORMAT_STRING 節" を参照してください。
- solve_order は、ラベルの適用順序を指定するオプションの整数です。"SOLVE_ORDER 節" を参照してください。
- cell_style は、データ・セルに適用するカスケード・スタイル・シート (CSS) のスタイル定義を指定するオプションのリテラルです。
- heading_style は、ヘッダに適用するカスケード・スタイル・シート (CSS) のスタイル定義を指定するオプションのリテラルです。

## 例

```
SELECT %LABEL(MEASURES.[avg allergy count],"my label") ON 0, colord.MEMBERS ON 1 FROM patients
```

```
                                  my label
1 None                                1.04
2 Blue                                1.01
3 Green                               1.02
4 Orange                              1.01
5 Purple                              1.06
6 Red                                 1.03
7 Yellow                              1.00
```

一方、以下は別の例です。

```
SELECT MEASURES.[avg allergy count] ON 0, colord.MEMBERS ON 1 FROM patients
```

```
                          Avg Allergy Count
1 None                                1.04
2 Blue                                1.01
3 Green                               1.02
4 Orange                              1.01
5 Purple                              1.06
6 Red                                 1.03
7 Yellow                              1.00
```

さらに別の例は、"%CELL"、"CURRENTMEMBER"、"IIF"、および "PROPERTIES" を参照してください。

## 関連項目

- IIF
- ISNULL

# %LAST (MDX)

セットのうち、空でない最後のメンバについて評価された、指定されたメジャー (または他の数値式) の値が返されます。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
%LAST(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

%LAST 関数は、指定されたセットの各メンバに対して評価された、欠落のない最後の値を返します。

## 例

以下のクエリは、骨粗しょう症を持つ患者を誕生年の年代別にグループ化して示します。

```
SELECT MEASURES.[%Count] ON 0, birthd.decade.MEMBERS ON 1 FROM patients WHERE diagd.osteoporosis
```

```
                              Patient Count
 1 1910s                                  2
 2 1920s                                  5
 3 1930s                                 10
 4 1940s                                  5
 5 1950s                                  *
 6 1960s                                  *
 7 1970s                                  *
 8 1980s                                  *
 9 1990s                                  *
10 2000s                                  *
11 2010s                                  *
```

以下のクエリは %LAST を使用して、前のセットから最後の空でない患者セットを取得します。

```
SELECT MEASURES.[%Count] ON 0, %LAST(birthd.decade.MEMBERS) ON 1 FROM patients WHERE diagd.osteoporosis
```

```
                              Patient Count
LAST                                     5
```

## 関連項目

- %FIRST

# %LIST (MDX)

値のセットが指定されると、コンマで区切られた値のリストを返します。この関数は、インターシステムズによる MDX への拡張機能で、KPI での使用を目的としています。

## 返りタイプ

この関数は、コンマ区切りの値のリストで構成された文字列を返します。

## 構文および詳細

```
%LIST(set_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。

## 例

以下の例では、診断ごとの %COUNT メジャーが示され、それに続いて、これらの値のコンマで区切られたリストを含む文字列が示されています。

```
WITH SET temp AS 'diagd.MEMBERS'
SELECT MEASURES.[%COUNT] ON 0, {temp,%LIST(temp)} ON 1 FROM patients
```

```
                              Patient Count
1 None                                8,603
2 asthma                                676
3 CHD                                   345
4 diabetes                              546
5 osteoporosis                          212
6 LIST                    8603,676,345,546,212
```

以下の例では、各都市の LIST 列に、その都市の出生数を 10 年ごとにコンマで区切ったリストが含まれています。これは、Scorecard with Trend Lines ダッシュボードのスコアカードに使用される基本的なクエリです。

```
SELECT {MEASURES.[%COUNT],%LIST(birthd.decade.MEMBERS)} ON 0, homed.city.MEMBERS ON 1 FROM patients
```

```
                        Patient Count                LIST
1 Cedar Falls                   1,079   11,22,42,92,115,159,176,135,127,179,21
2 Centerville                   1,058   3,24,51,57,114,157,167,152,161,145,27
3 Cypress                       1,095   12,14,57,84,105,146,160,159,153,166,39
4 Elm Heights                   1,086   8,22,54,72,121,154,168,135,176,143,33
5 Juniper                       1,152   9,21,56,79,124,165,200,152,169,155,22
6 Magnolia                      1,152   6,18,55,83,116,181,190,159,172,145,27
7 Pine                          1,120   8,22,47,72,108,160,166,173,161,170,33
8 Redwood                       1,132   6,17,61,90,109,174,189,129,171,154,32
9 Spruce                        1,126   11,20,70,74,107,167,171,161,164,156,25
```

注釈　この関数は、スコアカード・ウィジェットに表示される KPI での使用を目的にしています。具体的には、トレンド・ラインまたはトレンド・バー・グラフィックを表示するスコアカード・プロパティのソース値として使用します。

# %LOOKUP (MDX)

条件リストから値を 1 つ返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値または文字列を返します。

## 構文および詳細

```
%LOOKUP(termlist,key,field,default)
```

以下は、この指定の説明です。

- termlist は、条件リストの名前です。
- key は、その条件リストのキー値です。
- field は、値を取得するために使用するオプションのフィールド (列) 名です。既定では、value 列が返されます。
- default は、条件リスト、キー、またはフィールドが見つからない場合に返す値です。

%LOOKUP は、%TERMLIST と異なり、単一の値を返します。

条件リストのセルで、[Outlet].[h1].[city].[boston] など、有効なメンバ参照が定義されている場合、%LOOKUP は、このメンバ参照をリテラル値として返すのではなく、この参照を解決します。これは主に、%TERMLIST で使用するために作成された条件リストとの互換性を維持するためです。

用語リストの定義に関する詳細は、"条件リストの定義" を参照してください。

## 他の条件リスト関数との比較

条件リストと共に使用できる関数について、以下の表で比較します。

| 関数 | 目的 | 返り値 |
|---|---|---|
| %LOOKUP | 指定された条件リスト項目のキーに基づいて、値を検索します。条件リスト項目の値を返します。返す既定値を指定できます。 | 数値または文字列 (これはメンバ名の場合もあります)。 |
| LOOKUP | 条件リスト項目のフィールドを返します。既定では、このフィールドはキー・フィールドですが、別のフィールドを返すこともできます。返す既定値を指定できます。 | |
| %TERMLIST | 指定された条件リストに基づいてセットを返します。 | セットを返します。 |

## 例

キーと値のペアが 1 つ存在する VALUES という名前の用語リストを考えます。

```
key      value
CutOff   10000000
```

この場合、以下のように %LOOKUP を使用できます。

```
SELECT %LOOKUP("Values","CutOff") ON ROWS FROM HOLEFOODS
```

```
==> 1000000
```

もう 1 つの例として、以下のクエリは、条件リストの切り捨て値よりも人口が多い市町村のリストを返します。

```
SELECT
FILTER(Outlet.City.Members,Outlet.H1.City.CurrentMember.Properties("Population")>%LOOKUP("Values","CutOff"))

ON ROWS,Outlet.H1.City.CurrentMember.Properties("Population") ON COLUMNS  FROM HOLEFOODS
```

## 関連項目

- %TERMLIST
- LOOKUP

# %MDX (MDX)

現在のクエリのコンテキストの外側で MDX クエリを実行し、単一の結果を返します。この関数は、MDX に対する Business Intelligence 拡張機能です。

## 返りタイプ

この関数は、数値または文字列を返します。

## 構文および詳細

```
%MDX(mdx_query,parmName1,parmValue1,parmName2,parmValue2)
```

以下は、この指定の説明です。

- mdx_query は、引用符付きの MDX クエリです。これは単一の値を返し、左上のセルのみが使用されます。

  このクエリには、名前付きパラメータ、計算メンバ、および名前付きセットを含めることができます。

- parmName1、parmName2 (以降同様) は、クエリの名前付きパラメータ (オプション) です。これらは、引用符で囲む必要があります。

  パラメータをリストする順序は、クエリには影響しません。

  16 個までのパラメータとその値を指定できます。

- parmValue1、parmValue2 (以降同様) は、名前付きパラメータの対応値です。

この関数は、指定されたクエリを実行し、単一の値を返します。クエリが複数の行または列を返す場合、この関数は左上のセルのみを返します。これは、別のクエリ内のサブクエリを含めるために使用します。

システムには、%MDX で使用できる特殊な %CONTEXT パラメータが用意されています。詳細は、"%KPI" を参照してください。%KPI にもこのパラメータを使用できます。

重要　　%CONTEXT パラメータを使用する場合は、引用符で囲むことを忘れないでください。

## 例

%MDX を使用して、クエリに含める値を取得します。ただしこの値は、クエリに含めないと、クエリの行および列の定義の影響を受ける値です。例えば、合計レコード数にアクセスする必要がある場合に使用できます。

```
WITH MEMBER A.FRACTION AS 'MEASURES.[%COUNT]/%MDX("SELECT FROM patients")'
SELECT { MEASURES.[%COUNT], A.FRACTION } ON 0, diagd.MEMBERS ON 1 FROM patients
```

| | Patient Count | FRACTION |
|---|---|---|
| 1 None | 8,428 | 0.84 |
| 2 asthma | 712 | 0.07 |
| 3 CHD | 343 | 0.03 |
| 4 diabetes | 485 | 0.05 |
| 5 osteoporosis | 212 | 0.02 |

以下の例では、%MDX を名前付きパラメータ (City) と共に使用しています。

```
SELECT
%MDX("WITH %PARM City AS 'value:[All Cities]' SELECT FROM HOLEFOODS WHERE Outlet.@City",
"City",Outlet.CurrentMember.Properties("NAME")) ON 0,
Outlet.City.Members on 1 FROM HOLEFOODS
```

| | NAME |
|---|---|
| 1 Amsterdam | 1,633 |
| 2 Antwerp | 421 |
| 3 Atlanta | 3,331 |
| 4 Bangalore | 3,786 |
| ... | |

この場合、サブクエリは以下のようになります。

```
WITH %PARM City AS 'value:[All Cities]' SELECT FROM HOLEFOODS WHERE Outlet.@City
```

以下の例で、%CONTEXT の効果を示します。まず、以下のクエリでは、%CONTEXT を使用せずに %MDX を使用します。

```
WITH
MEMBER [MEASURES].[PercentOfAllRevenue] AS '100 * MEASURES.[Amount Sold] /
%MDX("SELECT MEASURES.[Amount Sold] ON 0 FROM holefoods")'

SELECT NON EMPTY [Channel].[H1].[Channel Name].Members ON 0,
NON EMPTY [Product].[P1].[Product Category].Members ON 1
FROM [HoleFoods]
WHERE [MEASURES].[PERCENTOFALLREVENUE]
```

| | No Channel | Online | Retail |
|---|---|---|---|
| 1 Candy | 0.28 | 0.83 | 0.61 |
| 2 Cereal | 0.11 | 0.58 | 0.39 |
| 3 Dairy | 0.23 | 2.43 | 1.01 |
| 4 Fruit | 1.04 | 7.55 | 3.76 |
| 5 Pasta | 0.79 | 7.19 | 4.14 |
| 6 Seafood | 3.60 | 22.23 | 10.41 |
| 7 Snack | 2.58 | 10.84 | 7.28 |
| 8 Vegetable | 1.42 | 6.63 | 4.05 |

計算メンバ [MEASURES].[PercentOfAllRevenue] は現在のピボット・テーブルのセルの Amount Sold メジャーを計算します (キューブ全体にわたるこのメジャーの集約値で除算されます)。その後この値は 100 で除算されるため、結果に表示された値の合計が 100 になります。

これに対し、"rows" として %CONTEXT を使用した場合の結果について考えてみます。

```
WITH
MEMBER [MEASURES].[PercentOfRows] AS '100 * MEASURES.[Amount Sold] /
%MDX("SELECT MEASURES.[Amount Sold] ON 0 FROM holefoods","%CONTEXT","rows")'

SELECT NON EMPTY [Channel].[H1].[Channel Name].Members ON 0,
NON EMPTY [Product].[P1].[Product Category].Members ON 1
FROM [HoleFoods]
WHERE [MEASURES].[PercentOfRows]
```

| | No Channel | Online | Retail |
|---|---|---|---|
| 1 Candy | 16.08 | 48.30 | 35.62 |
| 2 Cereal | 10.29 | 53.69 | 36.02 |
| 3 Dairy | 6.38 | 66.15 | 27.48 |
| 4 Fruit | 8.41 | 61.14 | 30.45 |
| 5 Pasta | 6.49 | 59.33 | 34.18 |
| 6 Seafood | 9.93 | 61.34 | 28.73 |
| 7 Snack | 12.46 | 52.38 | 35.16 |
| 8 Vegetable | 11.72 | 54.77 | 33.51 |

この場合、%MDX サブクエリは、行コンテキストを使用します。その結果、各行の数値の合計が 100 になります。

今度は、"columns" として %CONTEXT を使用した場合の結果について考えてみます。

```
WITH
MEMBER [MEASURES].[PercentOfCols] AS '100 * MEASURES.[Amount Sold] /
%MDX("SELECT MEASURES.[Amount Sold] ON 0 FROM holefoods","%CONTEXT","columns")'

SELECT NON EMPTY [Channel].[H1].[Channel Name].Members ON 0,
NON EMPTY [Product].[P1].[Product Category].Members ON 1
FROM [HoleFoods]
WHERE [MEASURES].[PercentOfCols]
```

| | No Channel | Online | Retail |
|---|---|---|---|
| 1 Candy | 2.76 | 1.43 | 1.94 |
| 2 Cereal | 1.10 | 0.99 | 1.22 |
| 3 Dairy | 2.34 | 4.18 | 3.19 |
| 4 Fruit | 10.35 | 12.96 | 11.88 |
| 5 Pasta | 7.83 | 12.34 | 13.08 |
| 6 Seafood | 35.83 | 38.14 | 32.89 |
| 7 Snack | 25.67 | 18.60 | 22.99 |
| 8 Vegetable | 14.12 | 11.37 | 12.80 |

この場合、各列の数値の合計が 100 になります。

## 関連項目

- %KPI

# %NOT (MDX)

特定のレベルの単一のメンバを除外できます。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.%NOT
```

以下は、この指定の説明です。

- member_expression は、メンバ ID です(一般のメンバ式は使用できません)。

この関数を使用すると、指定したメンバを除外できます。

## 例

多くの場合は、WHERE 節によって単一のメンバを除外する必要があります。例えば、まず以下のクエリを考えてみます。EXCEPT を使用しています。

```
SELECT aged.[age bucket].MEMBERS ON 1 FROM patients WHERE EXCEPT(aged.[age group].MEMBERS,aged.[age
group].[0 to 29])

1 0 to 9                                    *
2 10 to 19                                  *
3 20 to 29                                  *
4 30 to 39                                166
5 40 to 49                                139
6 50 to 59                                106
7 60 to 69                                 86
8 70 to 79                                 62
9 80+                                      41
```

%NOT 関数を使用して、前のクエリを以下のように書き換えることができます。

```
SELECT aged.[age bucket].MEMBERS ON 1 FROM patients WHERE aged.[age group].[0 to 29].%NOT

 1 0 to 9                                   *
2 10 to 19                                  *
3 20 to 29                                  *
4 30 to 39                                166
5 40 to 49                                139
6 50 to 59                                106
7 60 to 69                                 86
8 70 to 79                                 62
9 80+                                      41
```

この関数を列軸または行軸に使用すると、この関数がメンバを返すことを確認できます。

```
SELECT aged.[age group].[0 to 29].%NOT ON 1 FROM patients

Not 0 to 29                               600
```

このように、メンバの名前として、NOT の後に除外するメンバの名前を指定します。

%NOT 関数には、いくつかの利点があります。

- システムは、そのレベルのすべてのメンバを実体化する必要がなくなります。
- 効率を上げるために、処理の初期部分で否定が行われます。
- %NOT は、単純なタプル式を形成するために、他のフィルタと (内部的に) 結合できる単一のメンバを返します。

## 関連項目

- EXCEPT

# %OR (MDX)

効率を上げるため、またカウントの重複を回避するために、複数メンバを単一のメンバに結合できます。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、メンバを返します。このメンバの名前は、メンバの名前の後に +Others が続いたものになります。

## 構文および詳細

```
%OR(set_expression)
```

以下は、この指定の説明です。

- set_expression は、メンバまたはタプルのセットに対して評価される式です。この式は、中括弧で囲む必要があります。

この関数を使用すると、指定したメンバまたはタプルを単一のユニットに結合できます。

## 例

多くの場合は、WHERE 節によって複数メンバのセットを含める必要があります。以下はその例です。

```
SELECT gend.MEMBERS ON 1 FROM patients WHERE {allerd.[ant bites],allerd.soy,allerd.wheat}


1 Female                                  56
2 Male                                    59
```

ただし、このクエリ構文では、システムがクエリの結果を複数回 (WHERE 節に含まれる項目ごとに 1 回) 評価してから、それらを結合することになります。そのため、実行時間が長くなる可能性や重複して項目をカウントする可能性があります (この例では、特定の患者が、WHERE 節のアレルギーごとに 1 回ずつ、合計 3 回カウントされることがあります)。

%OR 関数を使用して、このクエリを以下のように書き換えることができます。

```
SELECT gend.MEMBERS ON 1 FROM patients WHERE %OR({allerd.[ant bites],allerd.soy,allerd.wheat})


1 Female                                  55
2 Male                                    57
```

前よりも数が減っていることに注意してください。これは、このクエリでは患者を重複してカウントしていないからです。

異なるレベルのメンバ (またはタプル) を含むセットで %OR を使用できます。例えば以下のようになります。

```
SELECT NON EMPTY [Measures].[%COUNT] ON 0 FROM [Patients]
WHERE %OR({[AgeD].[H1].[Age Bucket].&[80+],[DiagD].[H1].[Diagnoses].&[CHD]})

                              Patient Count
                                         71
```

この関数を列軸または行軸に使用すると、この関数がメンバを返すことを確認できます。

```
SELECT %OR({allerd.[ant bites],allerd.soy,allerd.wheat}) ON 1 FROM patients

ant bites+Others                         112
```

%OR 関数には、いくつかの利点があります。

- セットのメンバは、1 つのユニットとして扱われます。
- 効率を上げるために、処理の初期部分でメンバの組み合わせが行われます。

- %OR は、単純なタプル式を形成するために、他のフィルタと (内部的に) 結合できる単一のメンバを返します。

"計算メンバの定義" の例も参照してください。

# %SEARCH (MDX)

WHERE 節および %FILTER 節で使用できるメジャー検索式を返します。

## 返りタイプ

"式のタイプ" の "メジャー検索式" のセクションを参照してください。

# %SPACE (MDX)

ラベルのない空白の行または列を挿入します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、空の文字列を返します。

## 構文および詳細

```
%SPACE()
```

## 例

```
SELECT {allerd.MEMBERS,%SPACE(),allersevd.MEMBERS} ON 1 FROM patients
```

```
 1 No Data Available                 3,962
 2 additive/coloring agen              423
 3 animal dander                       428
 4 ant bites                           457
 5 bee stings                          407
 6 dairy products                      460
 7 dust mites                          422
 8 eggs                                419
 9 fish                                429
10 mold                                438
11 nil known allergies              1,382
12 peanuts                            441
13 pollen                             424
14 shellfish                          431
15 soy                                455
16 tree nuts                          452
17 wheat                              419
18
19 Nil known allergies              1,382
20 Minor                            1,229
21 Moderate                         1,205
22 Life-threatening                 1,202
23 Inactive                         1,184
24 Unable to determine              1,141
```

# %TERMLIST (MDX)

条件リストに基づいてメンバのセットを作成できるようにします。%OR 関数で %TERMLIST をフィルタ処理に使用すると、特に有効です。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
%TERMLIST(term_list_name, flag)
```

以下は、この指定の説明です。

- term_list_name は、条件リストの名前として評価される文字列式です。
- flag (オプション) は、"EXCLUDE" または "Include" (既定) のどちらかです。

この関数は、既定では、条件リストのキー値によって指定されるメンバと条件リスト・パターンの組み合わせで構成されるセットを返します。条件リスト・パターンは、メンバが属するレベル、およびメンバのフル ID を作成する方法を示します。

flag として "EXCLUDE" を指定すると、条件リストで指定されたメンバを除く、指定されたレベルのすべてのメンバで構成されるセットを返します。

用語リストの定義に関する詳細は、"条件リストの定義" を参照してください。

## 例

例えば、HoleFoods に対して、MyCities という名前の条件リストが以下のように定義されているとします。

Terms

| key | value |
|---|---|
| Atlanta | Atlanta |
| Boston | Boston |
| New York | New York |

この条件リストに、以下に示す**パターン**式が指定されているとします。

```
[Outlet].[H1].[City].[*]
```

この場合、この条件リストに対して、%TERMLIST 関数は、City レベルのメンバ Atlanta、Boston、および New York で構成されるセットを返します。つまり、以下の 2 つの式は同等です。

```
%TERMLIST("MyCities")
```

および、

```
{[Outlet].[H1].[City].[Atlanta],[Outlet].[H1].[City].[Boston],[Outlet].[H1].[City].[New York]}
```

%OR 関数で %TERMLIST をフィルタ処理に使用すると、特に有効です。以下はその例です。

```
SELECT FROM holefoods %FILTER %OR(%TERMLIST("MyCities"))
```

## 関連項目

- %LOOKUP ("他の条件リスト関数との比較" を含む)
- LOOKUP
- %OR

# %TIMERANGE (MDX)

時間メンバの範囲 (範囲の片側を省略可能) を定義できるようにします。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
%TIMERANGE(start_member,end_member,keyword)
```

以下は、この指定の説明です。

- start_member は、時間レベルのメンバとして評価される式 (オプション) です。省略すると、このレベルの最も古いメンバが使用されます。
- end_member は、時間レベルのメンバとして評価される式 (オプション) です。省略すると、このレベルの最も新しいメンバが使用されます。
- keyword (オプション) は、INCLUSIVE または EXCLUSIVE のどちらかです。

  既定値は INCLUSIVE です。

start_member、end_member、または両方を指定する必要があります。

## 例

以下の例では、start_member と end_member の両方を使用しています。

```
SELECT NON EMPTY DateOfSale.YearSold.MEMBERS ON 1 FROM holefoods
WHERE %TIMERANGE(DateOfSale.YearSold.&[2009],DateOfSale.YearSold.&[2011])


1 2009                                  179
2 2010                                  203
3 2011                                  224
```

次の例では、範囲の片側のみ指定しています。

```
SELECT NON EMPTY DateOfSale.YearSold.MEMBERS ON 1 FROM holefoods
WHERE %TIMERANGE(DateOfSale.YearSold.&[2009])


1 2009                                  179
2 2010                                  203
3 2011                                  224
4 2012                                  114
```

次の例も範囲の片側のみ指定していますが、ここでは EXCLUSIVE キーワードを使用しています。

```
SELECT NON EMPTY DateOfSale.YearSold.MEMBERS ON 1 FROM holefoods
WHERE %TIMERANGE(,DateOfSale.YearSold.&[2009],EXCLUSIVE)


1 2007                                  124
2 2008                                  156
```

# %TIMEWINDOW (MDX)

時刻ディメンジョンのメンバのうち、指定された範囲テンプレートと一致するメンバのセットを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
%TIMEWINDOW(periodSet,rangeTemplateStart)
```

または、以下のようにします。

```
%TIMEWINDOW(periodSet,rangeTemplateStart,rangeTemplateEnd)
```

以下は、この指定の説明です。

- periodSet は、時間レベルのメンバのセットです。
- rangeTemplateStart は、periodSet より下位レベルにある、同じ階層内の時間レベルのメンバです。
- rangeTemplateEnd は、periodSet より下位レベルにある、同じ階層内の時間レベルの別のメンバです。指定された場合、rangeTemplateEnd は、rangeTemplateStart と同じ期間内にする必要があります (例えば、これら 2 つのメンバは、同じ年または同じ月に属する必要があります)。

  rangeTemplateEnd の既定値は rangeTemplateStart です。

システムは、rangeTemplateStart から rangeTemplateEnd までのメンバのセットを生成し、これをテンプレートとして使用して時間ウィンドウを指定します。例えば、rangeTemplateStart が 2000 年 1 月で rangeTemplateEnd が 2000 年 6 月の場合、時間ウィンドウは、指定された任意の年の 1 月 1 日から 6 月 30 日までの日付で構成されます。

次にこの関数は、指定された periodSet の各メンバを検証し、それぞれに対して、指定された時間ウィンドウに入る子メンバを返します。

この関数は、WHERE 節または %FILTER 節の中での使用を目的にしています。この関数は、可能であれば、より高い時間レベルのメンバを返すように最適化されています。したがって、多数のメンバは返されません。

## 例

まず、以下のクエリは、%TIMEWINDOW を行として使用します。このクエリは誕生年を検証し、それぞれの年の 1 月 1 日から 1月 5 日 (両方の日を含む) に生まれた患者のみを選択します。

```
SELECT NON EMPTY %TIMEWINDOW(birthd.year.MEMBERS,birthd.[jan 01 1924],birthd.[jan 05 1924]) ON 1
FROM patients
```

```
 1 Jan 4 1918                              1
 2 Jan 3 1934                              1
 3 Jan 3 1937                              1
 4 Jan 4 1937                              1
 5 Jan 2 1938                              1
 6 Jan 1 1940                              1
 7 Jan 1 1941                              1
 8 Jan 4 1947                              1
 9 Jan 5 1947                              1
10 Jan 2 1949                              1
11 Jan 1 1953                              1
...
```

この例では、範囲テンプレートは、1924 年の日付を任意に参照します。代わりにどの年を使用することもできます。

既述のとおり、この関数は主にフィルタ処理のためのものです。以下のクエリは、指定された任意の年の 1 月 1 日から 1 月 5 日に生まれた患者をすべて選択するだけのものです。

```
SELECT MEASURES.[%COUNT] ON 0 FROM patients
WHERE %TIMEWINDOW(birthd.year.MEMBERS,birthd.[jan 01 1924],birthd.[jan05 1924])
                               Patient Count
                                         806
```

以下のクエリは、同じフィルタを使用しますが、患者を誕生年別にグループ化して表示します。

```
SELECT MEASURES.[%COUNT] ON 0, NON EMPTY birthd.year.MEMBERS on 1
FROM patients
WHERE %TIMEWINDOW(birthd.year.MEMBERS,birthd.[jan 01 1924],birthd.[jan05 1924])

                               Patient Count
 1 1918                                    1
 2 1934                                    1
 3 1937                                    2
 4 1938                                    1
 5 1940                                    1
 6 1941                                    1
 7 1947                                    2
 8 1949                                    1
 9 1953                                    1
...
```

結果を理解しやすくするために、以下のクエリは %LABEL を使用して、わかりやすいキャプションを適用します。

```
SELECT %LABEL(MEASURES.[%COUNT],"Born Jan 1-5") ON 0, NON EMPTY birthd.year.MEMBERS on 1
FROM patients
WHERE %TIMEWINDOW(birthd.year.MEMBERS,birthd.[jan 01 1924],birthd.[jan 05 1924])

                                Born Jan 1-5
 1 1918                                    1
 2 1934                                    1
 3 1937                                    2
 4 1938                                    1
 5 1940                                    1
 6 1941                                    1
 7 1947                                    2
 8 1949                                    1
 9 1953                                    1
...
```

# %TOPMEMBERS (MDX)

指定された階層にある最初のレベルのすべてのメンバのセットを返します。または、レベルを指定すると、そのレベルのすべてのメンバのセットを返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
level_expression.%TOPMEMBERS
```

または、以下のようにします。

```
hierarchy_expression.%TOPMEMBERS
```

または、以下のようにします。

```
dimension_expression.%TOPMEMBERS
```

以下は、この指定の説明です。

- level_expression は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

- hierarchy_expression は、階層を返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name]
  ```

- dimension_expression はディメンジョンの名前で、必要に応じて角括弧に囲まれます (″識別子″ を参照)。以下はその例です。

  ```
  [dimension_name]
  ```

  システムはこれを、指定されたディメンジョン内の最初に表示される階層への参照と解釈します。

レベル名を指定すると、この関数は MEMBERS 関数と同等になります。

階層名を指定すると、この関数は、その階層に定義された最初のレベルのメンバで構成されるセットを返します。

ディメンジョン名を指定すると、この関数は、このディメンジョンの最初に表示される階層に定義された最初のレベルのメンバで構成されるセットを返します。

アナライザは、ディメンジョンを [**行**] または [**列**] にドラッグ・アンド・ドロップするときにこの関数を使用します。具体的には、ディメンジョンをドラッグ・アンド・ドロップするとき、アナライザは式 `[dimension_name].[hierarchy_name].%TOPMMEMBERS` を使用します。ここで hierarchy_name は、ディメンジョンに定義された最初の階層です。

## 例

例えば、以下のキューブ・コンテンツを考えてみます。

```
BirthD
  H1
    Decade
    Year
    Period
    Date
```

以下のクエリは、Decade レベルのメンバをすべて取得するように、H1 階層 (この場合は唯一の階層) を指定して %TOPMEMBERS 関数を使用します。

```
SELECT birthd.%TOPMEMBERS ON 1 FROM patients
                              All Patients
 1 1910s                                71
 2 1920s                               223
 3 1930s                               572
 4 1940s                               683
 5 1950s                             1,030
 6 1960s                             1,500
 7 1970s                             1,520
 8 1980s                             1,400
 9 1990s                             1,413
10 2000s                             1,433
11 2010s                               155
```

## 関連項目

- MEMBERS

# AGGREGATE (MDX)

メジャーの集約ロジックに従い、セットのすべての要素にわたって、指定されたメジャー (または現在のメジャー) の集約値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
AGGREGATE(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の集約値を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
 1 1910s                       80              5,359              75.17
 2 1920s                      227             12,910              74.20
 3 1930s                      567             33,211              74.67
 4 1940s                      724             38,420              73.39
 5 1950s                    1,079             46,883              73.72
 6 1960s                    1,475             57,814              74.16
 7 1970s                    1,549             49,794              74.35
 8 1980s                    1,333             35,919              74.13
 9 1990s                    1,426             29,219              74.79
10 2000s                    1,406             20,072              74.95
11 2010s                      134              1,346              73.55
```

次に、以下のクエリは、AGGREGATE を使用して、メンバのこのセットにわたって、これらのメジャーの集約値を検索します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
AGGREGATE(birthd.decade.MEMBERS) ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
AGGREGATE                  10,000            330,947              74.28
```

以下のクエリは、AGGREGATE 関数の 2 番目の引数を使用します。

```
SELECT AGGREGATE(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                AGGREGATE
                                    10,000
```

追加の類似例は、"AVG" を参照してください。"計算メンバの定義" の例も参照してください。

## 関連項目

- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# ALLMEMBERS (MDX)

指定されたレベルまたは階層のすべてのメンバのセットを返します。または、ディメンジョンの最初の階層のすべてのメンバのセットを返します。いずれの場合も、計算メンバがあればそれも返されます。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
level_expression.ALLMEMBERS
```

または、以下のようにします。

```
hierarchy_expression.ALLMEMBERS
```

または、以下のようにします。

```
dimension_expression.ALLMEMBERS
```

以下は、この指定の説明です。

- level_expression は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

- hierarchy_expression は、階層を返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name]
  ```

- dimension_expression はディメンジョンの名前で、必要に応じて角括弧に囲まれます (″識別子″ を参照)。以下はその例です。

  ```
  [dimension_name]
  ```

  システムはこれを、指定されたディメンジョン内の最初に表示される階層への参照と解釈します。

レベル式を指定すると、この関数は、そのレベルのメンバで構成されるセットを返します。このメンバは、キューブのレベル定義で指定された順序になります。

階層式を指定すると、この関数は、その階層にあるすべてのレベルのメンバ (定義されていれば All メンバを含む) で構成されるセットを返します。このメンバは、階層順に返されます。

ディメンジョン名を指定すると、この関数は、そのディメンジョンの最初に表示される階層にあるすべてのレベルのメンバで構成されるセットを返します。

いずれの場合も、計算メンバがあればそれも返されます (MEMBERS 関数とはこの点が異なります)。

階層順の詳細は、HIERARCHIZE 関数を参照してください。

## 例

以下のクエリは、Home Zip レベルのすべてのメンバを行として表示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.ALLMEMBERS ON 1 FROM patients

                              Patient Count
1 32006                                2,272
2 32007                                1,111
3 34577                                3,399
4 36711                                1,069
5 38928                                2,149
```

以下のクエリは、Home.H1 階層にあるすべてのレベルのすべてのメンバを行として表示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.h1.ALLMEMBERS ON 1 FROM patients
                              Patient Count
 1 32006                               2,272
 2 Juniper                             1,155
 3 Spruce                              1,117
 4 32007                               1,111
 5 Redwood                             1,111
 6 34577                               3,399
 7 Cypress                             1,150
 8 Magnolia                            1,111
 9 Pine                                1,138
10 36711                               1,069
11 Centerville                         1,069
12 38928                               2,149
13 Cedar Falls                         1,045
14 Elm Heights                         1,104
```

以下のクエリは、すべてのメジャーを表示します。それぞれのメジャーは、キューブ全体にわたって集約されています。

```
SELECT MEASURES.ALLMEMBERS ON 1, gend.gender.MEMBERS on 0 FROM patients
                              Female                 Male
1 Patient Count                 5,067                4,933
2 Age                         187,139              170,117
3 Avg Age                       36.93                34.49
4 Allergy Count                 3,067                3,131
5 Avg Allergy Count              1.02                 1.04
6 Encounter Count             169,164              158,183
7 Avg Encounter Cou             33.39                32.07
8 Test Score                  302,267              298,818
9 Avg Test Score                74.78                74.46
```

## 関連項目

- MEMBERS

# ANCESTOR (MDX)

指定されたレベル内で指定されたメンバの祖先を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
ANCESTOR(member_expression,ancestor_level)
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

- ancestor_level は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  このレベルは、member_expression の親レベル、またはそのメンバの祖先です。

この関数は、指定されたレベル内で指定されたメンバの祖先を返します。

## 例

以下のクエリは、1943 年 3 月 24 日の祖先である年を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, ANCESTOR(birthd.[Mar 24 1943],birthd.year) ON 1 FROM patients
```

```
                            Patient Count
1943                                   76
```

これに対して、以下のクエリは、1943 年 3 月 24 日の祖先である期間を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, ANCESTOR(birthd.[Mar 24 1943],birthd.period) ON 1 FROM patients
```

```
                            Patient Count
Mar-1943                               5
```

## 関連項目

- CHILDREN
- CLOSINGPERIOD
- COUSIN
- DESCENDANTS
- OPENINGPERIOD
- PARENT
- PERIODSTODATE

# AVG (MDX)

セットのうち、指定された式に対して値が NULL でないすべての要素にわたって、この式 (または現在のメジャー) の平均値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
AVG(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、この値が NULL である要素があればすべて無視し、残りの要素の平均値を計算します。

平均に NULL 要素を含める場合は、値を NULL から 0 に置き換える optional_numeric_expression の式を使用します。

すべての要素の数値が NULL の場合、この関数は NULL を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
 1 1910s                       80              5,359              75.17
 2 1920s                      227             12,910              74.20
 3 1930s                      567             33,211              74.67
 4 1940s                      724             38,420              73.39
 5 1950s                    1,079             46,883              73.72
 6 1960s                    1,475             57,814              74.16
 7 1970s                    1,549             49,794              74.35
 8 1980s                    1,333             35,919              74.13
 9 1990s                    1,426             29,219              74.79
10 2000s                    1,406             20,072              74.95
11 2010s                      134              1,346              73.55
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの平均値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
AVG(birthd.decade.MEMBERS) ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
AVG                        909.09          30,086.09              74.28
```

ここで、それぞれの値は、上記クエリの列内の値の平均です。例えば、Patient Count 値は、上記クエリの Patient Count 値の平均です。

別の例では、AVG の 2 番目の引数を使用します。

```
SELECT AVG(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                        AVG
                                     909.09
```

以下の例では、メジャーを指定しないクエリで AVG を使用します。

```
SELECT AVG(birthd.decade.MEMBERS) ON 0 FROM patients
                                        AVG
                                     909.09
```

この場合、この関数は、ファクト・テーブル内のレコードをカウントする %COUNT を使用します。

最後に、以下の例では、WHERE 節でメジャーを指定するクエリで AVG を使用します。

```
SELECT AVG(birthd.decade.MEMBERS) ON 0 FROM patients WHERE MEASURES.[encounter count]
                                        AVG
                                  30,086.09
```

この場合、この関数は、WHERE 節で指定されたメジャーを使用します。

## 関連項目

- AGGREGATE
- MAX
- MEDIAN
- MIN
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# BOTTOMCOUNT (MDX)

目的の要素数を指定すると、セットを並べ替え、その値の低い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
BOTTOMCOUNT(set_expression, element_count, optional_ordering_expression)
```

以下は、この指定の説明です。

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- element_count は整数リテラルです。

  この関数は、この引数を使用して、サブセットで返す要素の数を決定します。この引数が要素数よりも大きい場合、すべての要素が返されます。
- optional_ordering_expression は、セット要素の順序を決定する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の昇順で並べ替えます。階層があればすべて無視されます。

  この引数を省略すると、この関数はセット要素の現在の順序を使用します (そして、この関数は TAIL 関数のように動作します)。

## 例

まず、以下のクエリ、およびそれが返す結果について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
BOTTOMCOUNT(birthd.decade.MEMBERS, 100, MEASURES.[%COUNT]) ON 1
FROM patients
```

| | Patient Count |
|---|---|
| 1 1910s | 71 |
| 2 2010s | 155 |
| 3 1920s | 223 |
| 4 1930s | 572 |
| 5 1940s | 683 |
| 6 1950s | 1,030 |
| 7 1980s | 1,400 |
| 8 1990s | 1,413 |
| 9 2000s | 1,433 |
| 10 1960s | 1,500 |
| 11 1970s | 1,520 |

count_expression がメンバ数よりも大きいため、すべてのメンバが返されます。メンバは、%COUNT メジャーの値の昇順で並べ替えられます。

次に、類似のクエリについて考えてみます。こちらでは、使用する count_expression が 3 に等しくなっています。

```
SELECT MEASURES.[%COUNT] ON 0,
BOTTOMCOUNT(birthd.decade.MEMBERS, 3, MEASURES.[%COUNT]) ON 1
FROM patients
```

| | Patient Count |
|---|---|
| 1 1910s | 71 |
| 2 2010s | 155 |
| 3 1920s | 223 |

このクエリは、セットの中で、値の低い方からメンバを 3 つ選択します。

## 関連項目

- TOPCOUNT

# BOTTOMPERCENT (MDX)

メンバ全体の合計に適用される切り捨てパーセンテージを指定すると、セットを並べ替え、その値の低い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
BOTTOMPERCENT(set_expression, percentage, ordering_expression)
```

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- percentage は、100 以下の数値リテラルです。例えば、15 は 15 パーセントを表します。

  この関数は、この引数を使用して、サブセットで返す要素の切り捨てポイントを決定します。

  通常は、切り捨てポイントをまたぐメンバが存在します。このメンバは、下位のセットではなく、上位のセットに割り当てられます。その結果、返されるサブセットでは、ordering_expression の累計が、セット全体のパーセンテージを示す percentage よりも小さくなる可能性があります。
- ordering_expression は、セット・メンバの順序を決定する数値式です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の昇順で並べ替えます。階層があればすべて無視されます。

## 例

まず、以下のクエリ、およびそれが返す結果について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
BOTTOMPERCENT(birthd.decade.MEMBERS, 100, MEASURES.[%COUNT]) ON 1 FROM patients
```

```
                              Patient Count
 1 1910s                                  6
 2 1920s                                 13
 3 2010s                                 44
 4 1940s                                 54
 5 1930s                                 56
 6 1950s                                107
 7 1970s                                128
 8 1960s                                136
 9 1990s                                144
10 1980s                                155
11 2000s                                157
```

percentage が 100 であるため、すべてのメンバが返されます。

ここで、上記の例のバリエーションを考えてみます。percentage が 50 であるため、下位 50 パーセントが表示されます。

```
SELECT MEASURES.[%COUNT] ON 0, BOTTOMPERCENT(birthd.decade.MEMBERS, 50, MEASURES.[%COUNT]) ON 1 FROM
patients
```

```
                              Patient Count
1 1910s                                  6
2 1920s                                 13
3 2010s                                 44
4 1940s                                 54
5 1930s                                 56
6 1950s                                107
7 1970s                                128
```

これらのメンバの %COUNT メジャーの合計は、指定されたしきい値 (合計の 50%) をわずかに下回ります。

## 関連項目

- TOPPERCENT

# BOTTOMSUM (MDX)

要素全体の合計に適用される切り捨て値を指定すると、セットを並べ替え、その値の低い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
BOTTOMSUM(set_expression, cutoff_value, ordering_expression)
```

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- cutoff_value は数値リテラルです。

  この関数は、この引数を使用して、サブセットで返す要素の切り捨て値を決定します。

  返されたサブセットのすべての要素に関して、ordering_expression の値の合計は、cutoff_value 以下になります。
- ordering_expression は、セット要素の順序を決定する数値式です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の昇順で並べ替えます。階層があればすべて無視されます。

## 例

まず、切り捨て値が大きく、すべてのメンバが含まれる例について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
BOTTOMSUM(birthd.decade.MEMBERS, 10000, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
 1 1910s                                71
 2 2010s                               155
 3 1920s                               223
 4 1930s                               572
 5 1940s                               683
 6 1950s                             1,030
 7 1980s                             1,400
 8 1990s                             1,413
 9 2000s                             1,433
10 1960s                             1,500
11 1970s                             1,520
```

次に、切り捨て値を 2500 に設定したバリエーションについて考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
BOTTOMSUM(birthd.decade.MEMBERS, 2500, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
1 1910s                                71
2 2010s                               155
3 1920s                               223
4 1930s                               572
5 1940s                               683
```

## 関連項目

- TOPSUM

# CHILDREN (MDX)

指定されたメンバに子があれば、それを含むセットを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
member_expression.CHILDREN
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

指定されたメンバに子がない場合、この関数は空のセットを返します。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[1960s].CHILDREN ON 1 FROM patients
                                Patient Count
 1 1960                                   105
 2 1961                                   153
 3 1962                                   144
 4 1963                                   153
 5 1964                                   136
 6 1965                                   149
 7 1966                                   187
 8 1967                                   159
 9 1968                                   169
10 1969                                   145
```

## 関連項目

- ANCESTOR
- COUSIN
- DESCENDANTS
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- LASTSIBLING
- PARENT
- SIBLINGS

# CLOSINGPERIOD (MDX)

指定されたメンバと同じレベルで、指定されたレベルの最後の子孫メンバを返します。この関数は主に、時間レベルで使用するためのものです。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
CLOSINGPERIOD(ancestor_level,member_expression)
```

以下は、この指定の説明です。

- ancestor_level は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  このレベルは、member_expression の親レベル、またはそのメンバの祖先です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

レベルとメンバを指定すると、この関数は、指定されたレベルの子孫であり、かつメンバと同じレベルにある最後のメンバを返します。

## 例

以下のクエリは、Q3 2003 を含む年の最終四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, CLOSINGPERIOD (birthd.year,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
Q4 2003                                40
```

これに対して、以下のクエリは、Q3 2003 を含む 10 年間の最終四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, CLOSINGPERIOD (birthd.decade,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
Q4 2010                                36
```

## 関連項目

- ANCESTOR
- COUSIN
- OPENINGPERIOD
- PERIODSTODATE

# COUNT (MDX)

指定されたセット内の要素数を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
COUNT(set_expression)
```

または、以下のようにします。

```
COUNT(set_expression,EXCLUDEEMPTY)
```

- set_expression は、セットに対して評価される式です。

既定では、COUNT によってあらゆる空の要素が考慮され、空以外の要素と共にカウントされます。EXCLUDEEMPTY キーワードを使用すると、この関数は、空以外の要素の数を返します。

## 例

例えば、以下のクエリは、Home City レベルのメンバをカウントします。

```
SELECT COUNT(homed.city.MEMBERS) ON 0 FROM patients
```

```
                                    COUNT
Results                                 9
```

次の例は、EXCLUDEEMPTY キーワードを使用した場合を示しています。まず、以下のクエリについて考えてみます。

```
SELECT aged.[age group].MEMBERS ON 0, diagd.MEMBERS ON 1 FROM patients WHERE MEASURES.[%COUNT]
```

```
                           0 to 29           30 to 59                60+
1 None                        3,839              3,615                971
2 asthma                        308                282                113
3 CHD                             1                 93                229
4 diabetes                       30                246                228
5 osteoporosis                    *                  *                200
```

以下のクエリは、Diagnoses レベルのメンバの数をカウントします。

```
WITH SET myset AS 'diagd.MEMBERS'
SELECT COUNT(myset) ON 0 FROM patients
```

```
                                    COUNT
All Patients                            5
```

以下のクエリでは、Diagnoses レベルのメンバの数がカウントされ、WHERE 節を使用して 0 ～ 29 歳の年齢グループの患者のみが取得されます。

```
WITH SET myset AS 'diagd.MEMBERS'
SELECT COUNT(myset) ON 0 FROM patients WHERE aged.[0 to 29]
```

```
                                    COUNT
                                        5
```

見てわかるように、クエリが WHERE 節を使用していても、COUNT 関数は、以前と同じ値を返します。これは、既定では COUNT が空の要素を考慮するためです。

次のクエリは、上記のバリエーションですが、EXCLUDEEMPTY を使用しています。

```
WITH SET myset AS 'diagd.MEMBERS' SELECT COUNT(myset,EXCLUDEEMPTY) ON 0 FROM patients WHERE aged.[0 to
 29]
                                      COUNT
                                           4
```

また別の例では、より一般的なメンバのセットではなく、スカラ項目のセットと共に COUNT を使用できます。

```
WITH SET test AS '{"item 1","item 2",23}'
SELECT COUNT(test) ON 0 FROM patients                                    COUNT
All Patients                                3
```

# COUSIN (MDX)

参照メンバ、および同じ階層内でのより高いレベルのメンバを指定すると、この関数は、そのより高いレベルで参照メンバの祖先を検索し、その祖先に対する参照メンバの相対位置を判別し、相対位置が同じである、より高いメンバの子孫を返します。この関数は主に、時間レベルで使用するためのものです。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
COUSIN(member_expression, higher_member_expression)
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。
- higher_member_expression は、member_expression を含む階層内のより高いレベルのメンバであるメンバを返す式です。

これらの式は、メジャーを参照できません。

この関数は、より高いレベルで参照メンバの祖先を検索し、その祖先に対する参照メンバの相対位置を判別し、相対位置が同じである、より高いメンバの子孫を返します。そのようなメンバが見つからなかった場合、この関数は空のセットを返します。

## 例

例えば、以下のクエリは、1990 年内で 1943 年 3 月 24 日の従兄弟を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, COUSIN(birthd.[Mar 24 1943],birthd.1990) ON 1 FROM patients
```

```
                            Patient Count
Mar 24 1990                             1
```

これに対して、以下のクエリは、1990 年 1 月内で 1943 年 3 月 24 日の従兄弟を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, COUSIN(birthd.[Mar 24 1943],birthd.[jan-1990]) ON 1 FROM patients
```

```
                            Patient Count
Jan 24 1990                             *
```

## 関連項目

- ANCESTOR
- CHILDREN
- DESCENDANTS
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- LASTSIBLING
- PARALLELPERIOD
- PARENT

- SIBLINGS

# CROSSJOIN (MDX)

指定されたセットのクロス積で形成されたタプルのセットを返します。

## 返りタイプ

この関数は、タプルのセットを返します。

## 構文および詳細

```
CROSSJOIN(set_expression1, set_expression2)
```

以下は、この指定の説明です。

- set_expression1 および set_expression2 は、メンバのセットに対して評価される式です。

  これらのセットのうち、数字に評価されるメジャーまたは式を含めることができるのは 1 つのみであることに注意してください。数字に評価されるメジャーまたは式が両方のセットに含まれる場合は、Analytics エンジンによって `[Two measures cannot be crossjoined]` というエラーが出力されます。

この関数は、各セットのすべてのメンバを識別し、最初のセットの各メンバと 2 番目のセットの各メンバを組み合わせるタプルのセットを生成します。

Tipヒント キーワード句 `NON EMPTY` は、この関数で特に有効です。このキーワード句は、任意のセット式の直前に使用できることに注意してください。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, CROSSJOIN(diagd.MEMBERS, aged.[age group].MEMBERS) ON 1 FROM patients
```

```
                              Patient Count
 1 None->0 to 29                      3,839
 2 None->30 to 59                     3,615
 3 None->60+                            971
 4 asthma->0 to 29                      308
 5 asthma->30 to 59                     282
 6 asthma->60+                          113
 7 CHD->0 to 29                           1
 8 CHD->30 to 59                         93
 9 CHD->60+                             229
10 diabetes->0 to 29                     30
11 diabetes->30 to 59                   246
12 diabetes->60+                        228
13 osteoporosis->0 to 29                  *
14 osteoporosis->30 to 59                 *
15 osteoporosis->60+                    200
```

一方、NON EMPTY キーワード句を追加すると以下のようになります。

```
SELECT MEASURES.[%COUNT] ON 0, NON EMPTY CROSSJOIN(diagd.MEMBERS, aged.[age group].MEMBERS) ON 1 FROM
 patients
```

| | Patient Count |
|---|---|
| 1 None->0 to 29 | 3,839 |
| 2 None->30 to 59 | 3,615 |
| 3 None->60+ | 971 |
| 4 asthma->0 to 29 | 308 |
| 5 asthma->30 to 59 | 282 |
| 6 asthma->60+ | 113 |
| 7 CHD->0 to 29 | 1 |
| 8 CHD->30 to 59 | 93 |
| 9 CHD->60+ | 229 |
| 10 diabetes->0 to 29 | 30 |
| 11 diabetes->30 to 59 | 246 |
| 12 diabetes->60+ | 228 |
| 13 osteoporosis->60+ | 200 |

別の例を示します。

```
SELECT CROSSJOIN([GenD].[H1].[Gender].Members,
{[Measures].[%COUNT],[Measures].[Avg Age]}) ON 0,[DiagD].[H1].[Diagnoses].Members ON 1
FROM [Patients]
```

| | Patient Coun | Avg Age | Patient Coun | Avg Age |
|---|---|---|---|---|
| 1 None | 419 | 35.61 | 412 | 32.36 |
| 2 asthma | 51 | 37.12 | 33 | 27.79 |
| 3 CHD | 17 | 70.47 | 19 | 62.42 |
| 4 diabetes | 25 | 63.96 | 21 | 57.67 |
| 5 osteoporosis | 20 | 80.55 | 2 | 74.50 |

## 関連項目

- NONEMPTYCROSSJOIN

# CURRENTMEMBER (MDX)

階層のメンバの繰り返し内で、プログラムでメンバを参照できるようにします。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
hierarchy_expression.CURRENTMEMBER

dimension_expression.CURRENTMEMBER
```

- hierarchy_expression は、階層に評価される式です。
- dimension_expression はディメンジョンの名前で、必要に応じて角括弧に囲まれます (″識別子″ を参照)。以下はその例です。

  ```
  [dimension_name]
  ```

  システムはこれを、指定されたディメンジョン内の最初に表示される階層への参照と解釈します。

この関数は、階層を繰り返すコンテキストで使用します。CURRENTMEMBER 関数は、そのコンテキストで指定されたメンバを返します。

抽象的には、この関数は、ObjectScript の $this と同じ目的を持っています。

注釈　CURRENTMEMBER 関数は、MEASURES ディメンジョンではサポートされません。つまり、dimension_expression を MEASURES にすることはできません。

## 例

以下の例では、市町村が行として使用されています。列に表示されるデータは、市町村ごとの `Principal Export` プロパティで、PROPERTIES 関数を使用して取得されます。

```
SELECT homed.CURRENTMEMBER.PROPERTIES("Principal Export") ON 0, homed.city.MEMBERS ON 1
FROM patients
```

```
                                Home ZIP
1 Cedar Falls                         iron
2 Centerville                  video games
3 Cypress                           gravel
4 Elm Heights                      lettuce
5 Juniper                            wheat
6 Magnolia                       bundt cake
7 Pine                            spaghetti
8 Redwood                           peaches
9 Spruce                               mud
```

以下のバリエーションは、%LABEL 関数を使用して、わかりやすいキャプションをデータ列に指定しています。

```
SELECT %LABEL(homed.CURRENTMEMBER.PROPERTIES("Principal Export"),"Exports") ON 0,
homed.city.MEMBERS ON 1 FROM patients
```

```
                                   Export
1 Cedar Falls                         iron
2 Centerville                  video games
3 Cypress                           gravel
4 Elm Heights                      lettuce
5 Juniper                            wheat
6 Magnolia                       bundt cake
7 Pine                            spaghetti
8 Redwood                           peaches
9 Spruce                               mud
```

以下のクエリは、City レベルの Principal Export プロパティと Population プロパティの両方を示しています。

```
SELECT {%LABEL(homed.CURRENTMEMBER.PROPERTIES("Principal Export"),"Export"),
%LABEL(homed.CURRENTMEMBER.PROPERTIES("Population"),"Population")} ON 0,
homed.city.MEMBERS ON 1 FROM patients
```

| | Export | Population |
|---|---|---|
| 1 Cedar Falls | iron | 90,000 |
| 2 Centerville | video games | 49,000 |
| 3 Cypress | gravel | 3,000 |
| 4 Elm Heights | lettuce | 33,194 |
| 5 Juniper | wheat | 10,333 |
| 6 Magnolia | bundt cake | 4,503 |
| 7 Pine | spaghetti | 15,060 |
| 8 Redwood | peaches | 29,192 |
| 9 Spruce | mud | 5,900 |

# DESCENDANTS (MDX)

指定したレベル内で指定したメンバの子孫であるメンバを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
DESCENDANTS(member_expression, level_expression, OPTIONAL_FLAG)
```

または、以下のようにします。

```
DESCENDANTS(member_expression, level_offset, OPTIONAL_FLAG)
```

- member_expression は、メンバに評価される式です。以下の説明では、このメンバが 対象メンバです。
- level_expression は、レベル識別子です。これは、対象メンバと同じ階層およびそれ以下の階層にあるレベル (より詳細なレベル) を識別する必要があります。

  上記の例の代わりに、対象メンバを含むレベルを基準とした相対レベルを示す整数である level_offset を指定できます。例えば、次に低いレベルを指定するには 1 を使用します。

  以下の説明では、対象レベル という語句は、level_expression または level_offset で指定されたレベルを表します。
- OPTIONAL_FLAG は、メンバを返す必要があるレベルを指定します。このオプションは、対象レベルとの関係を説明します。OPTIONAL_FLAG を指定する場合、以下のキーワードのいずれかにする必要があります (大文字と小文字の区別はありません)。
  - SELF (既定) ー 対象レベルのメンバである子孫 のみを返します。
  - AFTER ー 対象レベル未満のすべてのレベルのメンバである子孫を返します。
  - BEFORE ー 対象メンバ以下のすべての子孫を返します。つまり、対象レベルより上のすべてのレベルのメンバである子孫を返し、対象メンバも返します。
  - BEFORE_AND_AFTER ー 対象レベルの上と下の両方の子孫を返しますが、対象レベルのメンバは含みません。
  - SELF_AND_AFTER ー 対象レベルのメンバである子孫および対象レベルより下のすべてのレベルの子孫を返します。
  - SELF_AND_BEFORE ー 対象レベルのメンバである子孫、指定したレベルより上のすべてのレベルに属している子孫、および対象メンバを返します。
  - SELF_BEFORE_AFTER ー 対象メンバを含むすべてのレベルの子孫を返します。

    この場合、対象レベルは結果に影響を与えません。

この関数が複数のレベルのメンバを返す場合、その階層が以下のようにメンバの順序に影響します。より高いレベルのメンバの後に次に低いレベルの子が続き、さらにより高いレベルの次のメンバが続き、以下同様になります。SELF_AND_AFTER の例を参照してください。

注釈　DESCENDANTS のこの実装は、オプションの LEAVES フラグをサポートしません。

## 例

参考に Patients キューブでは、BirthD ディメンジョンに以下のレベルが高いものから低いものの順に格納されます。

- [BirthD].[H1].[Decade]
- [BirthD].[H1].[Year]
- [BirthD].[H1].[Quarter Year] (年と四半期を表します)
- [BirthD].[H1].[Period] (年と月を表します)
- [BirthD].[H1].[Date] (年、月、日を表します)

以下の例では、1990 年の [BirthD].[H1].[Period] レベル内のすべての子孫を取得します。

```
SELECT DESCENDANTS(birthd.1990,birthd.period) ON 1 FROM patients
```

```
 1 Jan-1990                              *
 2 Feb-1990                              2
 3 Mar-1990                              1
 4 Apr-1990                              1
 5 May-1990                              1
 6 Jun-1990                              *
 7 Jul-1990                              2
 8 Aug-1990                              2
 9 Sep-1990                              1
10 Oct-1990                              3
11 Nov-1990                              1
12 Dec-1990                              *
```

この例では OPTIONAL_FLAG に既定値 (SELF) を使用するため、この関数は期間レベルのメンバである 1990 の子孫のみを返します。

以下のバリエーションは NON EMPTY を使用するため、患者が生まれなかった期間をフィルタで除外します。

```
SELECT NON EMPTY DESCENDANTS(birthd.1990,birthd.period) ON 1 FROM patients
```

```
1 Feb-1990                               2
2 Mar-1990                               1
3 Apr-1990                               1
4 May-1990                               1
5 Jul-1990                               2
6 Aug-1990                               2
7 Sep-1990                               1
8 Oct-1990                               3
9 Nov-1990                               1
```

この期間レベルは、年レベルの 2 レベル下で、以下のクエリ (level_offset に 2 を使用) は最初のクエリと同等です。

```
SELECT DESCENDANTS(birthd.1990,2) ON 1 FROM patients
```

```
 1 Jan-1990                              *
 2 Feb-1990                              2
 3 Mar-1990                              1
 4 Apr-1990                              1
 5 May-1990                              1
 6 Jun-1990                              *
 7 Jul-1990                              2
 8 Aug-1990                              2
 9 Sep-1990                              1
10 Oct-1990                              3
11 Nov-1990                              1
12 Dec-1990                              *
```

次のバリエーションは AFTER を使用します。

```
SELECT DESCENDANTS(birthd.1990,birthd.period,AFTER) ON 1 FROM patients
```

```
  1 Jan 1 1990                           *
  2 Jan 2 1990                           *
  3 Jan 3 1990                           *
...
363 Dec 29 1990                          *
364 Dec 30 1990                          *
365 Dec 31 1990                          *
```

この例は、期間レベルの下のすべてのレベルの 1990 の子孫を返します。この場合、下のレベルは日付の 1 つしかありません。これは、年、月、日に対応します。

次のバリエーションは SELF_AND_AFTER を使用します。この例は、複数のレベルのメンバを返し、これらのメンバが返される順序を示します。

```
SELECT DESCENDANTS(birthd.1990,birthd.period,SELF_AND_AFTER) ON 1 FROM patients
```

```
  1 Jan-1990                              *
  2 Jan 1 1990                            *
  3 Jan 2 1990                            *
  4 Jan 3 1990                            *
...
 33 Feb-1990                              2
 34 Feb 1 1990                            *
 35 Feb 2 1990                            *
 36 Feb 3 1990                            1
...
346 Dec-1990                              *
347 Dec 1 1990                            *
348 Dec 2 1990                            *
349 Dec 3 1990                            *
...
377 Dec 31 1990                           *
```

次のバリエーションは BEFORE を使用します。

```
SELECT DESCENDANTS(birthd.1990,birthd.period,BEFORE) ON 1 FROM patients
```

```
1 1990                                   14
2 Q1 1990                                 3
3 Q2 1990                                 2
4 Q3 1990                                 5
5 Q4 1990                                 4
```

この場合、クエリは、期間レベルの上のレベルのメンバである 1990 のすべての子孫を取得します (つまり、四半期レベルのメンバを返します)。1990 も返されます。

## 関連項目

- CHILDREN
- COUSIN
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- LASTSIBLING
- PARENT
- SIBLINGS

# DISTINCT (MDX)

セットを検証し、重複している要素を削除し、残りの要素のセットを返します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
DISTINCT(set_expression)
```

- set_expression は、セットに対して評価される式です。

## 例

例えば、他の市町村との比較に必要な基準として、特定の市町村をクエリで返す必要があるとします。基準の市町村を表示し、さらに指定した患者数を持つ市町村のセットを表示するという、以下のクエリを考えてみます。

```
WITH SET refcity AS '{homed.juniper}'
SELECT MEASURES.[%COUNT] ON 0,
{refcity,FILTER(homed.city.MEMBERS,MEASURES.[%COUNT]>1100)} ON 1 FROM patients
```

```
                            Patient Count
 1 Juniper                          1,197
 2 Cedar Falls                      1,188
 3 Centerville                      1,155
 4 Cypress                          1,221
 5 Elm Heights                      1,266
 6 Juniper                          1,197
 7 Magnolia                         1,156
 8 Pine                             1,139
 9 Redwood                          1,144
10 Spruce                           1,135
```

以下のクエリと比較してみてください。そちらでは、重複する市町村が削除されます。

```
WITH SET refcity AS '{homed.juniper}' SELECT MEASURES.[%COUNT] ON 0,
DISTINCT({refcity,FILTER(homed.city.MEMBERS,MEASURES.[%COUNT]>1100)}) ON 1 FROM patients
```

```
                            Patient Count
1 Juniper                           1,197
2 Cedar Falls                       1,188
3 Centerville                       1,155
4 Cypress                           1,221
5 Elm Heights                       1,266
6 Magnolia                          1,156
7 Pine                              1,139
8 Redwood                           1,144
9 Spruce                            1,135
```

# EXCEPT (MDX)

2 つのセットを検証し、最初のセットの要素で構成されるセットを返します。ただし、2 番目のセットにもある要素があれば除外します。この関数は、オプションで、そのセット内にある重複を削除します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
EXCEPT(set_expression1, set_expression2, ALL)

EXCEPT(set_expression1, set_expression2)
```

- set_expression1 および set_expression2 は、セットに対して評価される式です。
- オプションのキーワード ALL が含まれている場合、これは、すべての重複を保持することを指定します。既定では、最初のセットに、重複する要素が含まれている場合、それらのうち最初の要素のみが含まれます。

set_expression1 にない要素が set_expression2 に含まれている場合、それらの要素は無視されます。返されたセットには、set_expression1 の要素のみが含まれます。

## 例

2 つの名前付きセットを定義する以下のクエリを考えてみます。

```
WITH SET set1 AS '{allerd.eggs,allerd.eggs,allerd.soy,allerd.wheat}'
SET set2 AS '{allerd.[diary products],allerd.pollen,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, EXCEPT(set1,set2) ON 1 FROM patients

                                Patient Count
1 eggs                                    451
2 soy                                     462
```

このクエリは、`set1` のメンバのうち、`set2` にはないメンバを示します。メンバ `allerd.eggs` は、`set1` 内に 2 回リストされていますが、結果には 1 回しか表示されていません。

一方、以下のバリエーションでは ALL キーワードを使用しています。

```
WITH SET set1 AS '{allerd.eggs,allerd.eggs,allerd.soy,allerd.wheat}'
SET set2 AS '{allerd.[diary products],allerd.pollen,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, EXCEPT(set1,set2,ALL) ON 1 FROM patients

                                Patient Count
1 eggs                                    451
2 eggs                                    451
3 soy                                     462
```

## 関連項目

- INTERSECT
- UNION

# FILTER (MDX)

セットを検証し、指定された式が要素ごとに True となるサブセットを返します。セットの順序は変わりません。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
FILTER(set_expression, logical_expression)
```

- set_expression は、セットに対して評価される式です。
- logical_expression は論理式で、通常はメジャー値またはプロパティ値を検証します。

  logical_expression の代わりに、メジャー検索式を使用できます。この場合の FILTER の動作については、例を参照してください。

## 例

例えば、以下のクエリを考えてみます。これは、患者数が 1150 人より多い市町村のみを返します。

```
SELECT MEASURES.[%COUNT] ON 0,
FILTER(homed.city.MEMBERS, MEASURES.[%COUNT]>1150) ON 1 FROM patients
                              Patient Count
1 Cedar Falls                         1,188
2 Centerville                         1,155
3 Cypress                             1,221
4 Elm Heights                         1,266
5 Juniper                             1,197
6 Magnolia                            1,156
```

一方、以下のクエリを考えてみます。これは、患者数に関係なく、患者のいる市町村を返します。

```
SELECT MEASURES.[%COUNT] ON 0,
FILTER(homed.city.MEMBERS, MEASURES.[%COUNT]>=0) ON 1 FROM patients

                              Patient Count
1 Cedar Falls                         1,188
2 Centerville                         1,155
3 Cypress                             1,221
4 Elm Heights                         1,266
5 Juniper                             1,197
6 Magnolia                            1,156
7 Pine                                1,139
8 Redwood                             1,144
9 Spruce                              1,135
```

別の例として、以下のクエリは、より複雑なフィルタ式を使用しています。

```
SELECT MEASURES.[%COUNT] ON 0,
FILTER(diagd.members, (MEASURES.[%COUNT]>500 and MEASURES.[%COUNT]<1000)) ON 1
FROM patients

                              Patient Count
1 asthma                                746
2 diabetes                              555
```

次の例は、プロパティを評価するフィルタ式を使用しています。

```
SELECT homed.CURRENTMEMBER.PROPERTIES("Population") ON 0,
FILTER(homed.city.MEMBERS,homed.CURRENTMEMBER.PROPERTIES("Population")>20000) ON 1 FROM patients
```

```
                                    ZIP
1 Cedar Falls                     90,000
2 Centerville                     49,000
3 Elm Heights                     33,194
4 Redwood                         29,192
```

FILTER をメジャー検索式で使用する場合、この関数は、指定された条件を満たした少なくとも 1 つのファクトに基づくメンバのみを返します (メジャー検索式は、インターシステムズによる MDX への拡張機能で、ファクト・テーブル自体のメジャー値を考慮するものです)。

```
SELECT {MEASURES.[%COUNT]} ON 0, FILTER(diagd.members, %SEARCH.&[[MEASURES].[age]<10]) ON 1 FROM patients
```

```
                              Patient Count
1 None                                8,425
2 asthma                                703
```

このクエリは、10 歳未満の少なくとも 1 人の患者がいる診断を示しています。その他、糖尿病や骨粗しょう症などの診断では、このような若い患者はいません。

## 関連項目

- IIF

# FIRSTCHILD (MDX)

指定されたメンバの最初の子を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.FIRSTCHILD
```

以下は、この指定の説明です。

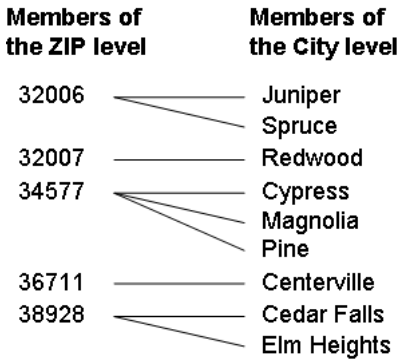
- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、このメンバの最初の子を返します。どの子が最初かを判別するために、この関数は、子のセットの既定の順序を考慮します。

## 例

例えば、以下の階層を考えてみます。




ここで、以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.[34577].FIRSTCHILD ON 1 FROM patients
```

```
                            Patient Count
Cypress                             1,089
```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1960.FIRSTCHILD ON 1 FROM patients
                            Patient Count
Jan-1960                                5
```

## 関連項目

- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTSIBLING

- LASTCHILD
- LASTSIBLING
- PARENT
- SIBLINGS

# FIRSTSIBLING (MDX)

指定されたメンバの最初の兄弟を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.FIRSTSIBLING
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、指定されたメンバの親のすべての子を検証し、(そのセットの既定の順序を考慮して) そのセットの最初のメンバを返します。

この関数は、引数として指定したメンバと同じメンバを返すこともできます (そのメンバが最初の兄弟の場合)。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[Mar 2003].FIRSTSIBLING ON 1 FROM patients

                            Patient Count
Jan-2003                               10
```

## 関連項目

- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTCHILD
- LASTCHILD
- LASTSIBLING
- PARENT
- SIBLINGS

# HEAD (MDX)

セットの現在の順序を使用して、セットの先頭からサブセットを返します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
HEAD(set_expression, optional_integer_expression, optional_sample_flag)
```

- set_expression は、セットに対して評価される式です。
- optional_integer_expression は整数リテラルです。

  この引数の既定値は 1 です。

  この関数は、この引数を使用して、サブセットで返す要素の数を決定します。
- optional_sample_flag は SAMPLE です (含まれている場合)。

  このオプション・フラグは、set_expression に複数の入れ子になった CROSSJOIN 式が含まれている場合にのみ有効です。そのような場合、既定では、set_expression が完全に評価され、すべてのセット要素が認識されるまで、システムはサブセットの決定を試みません。optional_sample_flag を含める場合、システムはそれぞれの CROSSJOIN が評価されると結果を切り捨てます。この方法を使用すると、非常に速く実行できますが、結果として、サブセットに含まれる要素が optional_integer_expression で指定されたものより少なくなる可能性があります。

  アナライザでドラッグ・アンド・ドロップ操作を使用し、複数の入れ子になった CROSSJOIN 式を含むセットに対して HEAD 関数を使用するクエリを作成する場合、アナライザはこのフラグを自動的に追加することに注意してください。

この関数は、指定されたセットの先頭から、(セットの現在の順序を考慮して) 指定された要素数で構成されるセットを返します。integer_expression が 1 未満の場合、この関数は空のセットを返します。integer_expression がセットの要素数より大きい場合、この関数は元のセットを返します。

サブセットの要素は、元のセットで指定された順序と同じ順序で返されます。

## 例

```
SELECT MEASURES.[%COUNT] ON 0, HEAD(birthd.decade.MEMBERS, 3) ON 1
FROM patients
                              Patient Count
1 1910s                                  71
2 1920s                                 223
3 1930s                                 572
```

## 関連項目

- TAIL

# HIERARCHISE (MDX)

HIERARCHIZE の同義語です。

## 関連項目

- HIERARCHIZE

# HIERARCHIZE (MDX)

セットを指定すると、階層順に (階層で指定された順序で) セットを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
HIERARCHIZE(set_expression)
```

または、以下のようにします。

```
HIERARCHIZE(set_expression, POST)
```

以下は、この指定の説明です。

- set_expression は、メンバのセットに対して評価される式です。
- POST を指定すると、子メンバがその親の前になります。これは、自然な順序の逆順と呼ばれます。

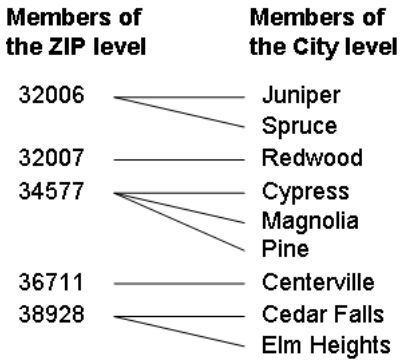
セット・メンバがさまざまな階層内にある場合、階層そのものの順序は不確定になります。つまり、一部のメンバが階層 A にあり、他のメンバが階層 B にある場合、A のメンバは階層順に連続してリストされ、B のメンバも階層順に連続してリストされますが、全体として A のメンバと B のメンバのいずれが先かを定める規則はありません。

## 例

階層内では、階層順は以下のように決定されます。

- ディメンジョンの All メンバがある場合は、これが最初になります。
- 次のメンバは、その階層の最高レベルの最初のメンバです。
- 次のメンバは、そのメンバの最初の子です。

以降、同様に繰り返されます。例えば、以下の階層を考えてみます。



これらのメンバの全体の階層順を表示するには、以下のクエリを使用します。これは、これらのメンバが属するディメンジョンのすべてのメンバで構成されるセットを使用します。

```
SELECT MEASURES.[%COUNT] ON 0, HIERARCHIZE(homed.members) ON 1 FROM patients
                              Patient Count
 1 32006                               2,272
 2 Juniper                             1,155
 3 Spruce                              1,117
 4 32007                               1,111
 5 Redwood                             1,111
 6 34577                               3,399
 7 Cypress                             1,150
 8 Magnolia                            1,111
 9 Pine                                1,138
10 36711                               1,069
11 Centerville                         1,069
12 38928                               2,149
13 Cedar Falls                         1,045
14 Elm Heights                         1,104
```

以下の例は、Home City レベルおよび Home ZIP レベルのいくつかのメンバのセットを作成し、HIERARCHIZE 関数を使用して、これらのメンバを階層順に配置します。

```
SELECT MEASURES.[%COUNT] ON 0,
HIERARCHIZE({homed.36711, homed.38928, homed.[elm heights], homed.Spruce}) ON 1
FROM patients
                              Patient Count
1 36711                                1,069
2 Spruce                               1,117
3 38928                                2,149
4 Elm Heights                          1,104
```

一方、次の例では POST キーワードを使用しています。

```
SELECT MEASURES.[%COUNT] ON 0,
HIERARCHIZE({homed.36711, homed.38928, homed.[elm heights], homed.Spruce}, POST) ON 1
FROM patients
                              Patient Count
1 36711                                1,069
2 Spruce                               1,117
3 Elm Heights                          1,104
4 38928                                2,149
```

## 関連項目

- ORDER

# IIF (MDX)

指定された論理式の値に応じて、2 つの値のいずれかを返します。

## 返りタイプ

この関数は、使用する引数、および論理式の値に応じて、数値または文字列を返します。

## 構文および詳細

```
IIF(check_expression, expression1, expression2)
```

- expression1 および expression2 は、数値式または 文字列式です。これらは、同じタイプである必要はありません。

  InterSystems MDX システムでは、他のタイプの引数はサポートされていません。

  NULL 値と比較するには、代わりに ISNULL 関数を使用します。
- check_expression は論理式で、通常はメジャーまたはプロパティを定数と比較します。

check_expression が True の場合、この関数は、expression1 で指定された値を返します。True でない場合は、expression2 で指定された値を返します。

## 例

以下はその例です。

```
SELECT IIF(MEASURES.[%COUNT]<500, "fewer than 500", "500 or more") ON 0, diagd.MEMBERS ON 1 FROM patients
```

```
                                    IIF
1 None                          500 or more
2 asthma                        500 or more
3 CHD                         fewer than 500
4 diabetes                      500 or more
5 osteoporosis                fewer than 500
```

バリエーションとして、以下のクエリは、%LABEL を使用して、データ列に適切なキャプションを適用します。

```
SELECT %LABEL(IIF(MEASURES.[%COUNT]<500, "fewer than 500", "500 or more"),"Patient Count") ON 0,
diagd.MEMBERS ON 1 FROM patients
```

```
                              Patient Count
1 None                           500 or more
2 asthma                         500 or more
3 CHD                          fewer than 500
4 diabetes                       500 or more
5 osteoporosis                 fewer than 500
```

別の例として、以下のクエリは、プロパティの値を使用しています。

```
SELECT %LABEL(IIF(homed.h1.CURRENTMEMBER.PROPERTIES("Population")>20000,"big","small"),
"Town Size") ON 0,
homed.city.MEMBERS ON 1 FROM patients
```

```
                                  Town Size
1 Cedar Falls                           big
2 Centerville                           big
3 Cypress                             small
4 Elm Heights                           big
5 Juniper                             small
6 Magnolia                            small
7 Pine                                small
8 Redwood                               big
9 Spruce                              small
```

以下の例は、メンバの名前を検証し、メジャーの表示を条件付きで抑制します。

```
WITH MEMBER MEASURES.iif AS 'IIF(homed.CURRENTMEMBER.PROPERTIES("name")="Pine"," ",
MEASURES.[%COUNT])' SELECT MEASURES.iif ON 0, homed.city.MEMBERS ON 1 FROM patients
```

| | iif |
|---|---|
| 1 Cedar Falls | 1,120 |
| 2 Centerville | 1,106 |
| 3 Cypress | 1,139 |
| 4 Elm Heights | 1,078 |
| 5 Juniper | 1,109 |
| 6 Magnolia | 1,122 |
| 7 Pine | |
| 8 Redwood | 1,128 |
| 9 Spruce | 1,108 |

## 関連項目

- ISNULL
- FILTER

# INTERSECT (MDX)

指定された 2 つのセットの両方にある要素で構成されたセットを返します。オプションで、そのセットにある重複を除外します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
INTERSECT(set_expression1, set_expression2, ALL)
```

または、以下のようにします。

```
INTERSECT(set_expression1, set_expression2)
```

- set_expression1 および set_expression2 は、セットに対して評価される式です。
- オプションのキーワード ALL が含まれている場合、これは、2 番目のセットにあるすべての重複を保持することを指定します。既定では、返されたセットに、重複する要素が含まれている場合、それらのうち最初の要素のみが含まれます。

  このキーワードは、最初のセットにある重複には影響しません。

## 例

2 つの名前付きセットを定義する以下のクエリを考えてみます。

```
WITH SET set1 AS '{allerd.eggs,allerd.soy,allerd.wheat,allerd.wheat}'
SET set2 AS '{allerd.[dairy products],allerd.pollen,allerd.soy,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, INTERSECT(set1,set2) ON 1 FROM patients
                               Patient Count
1 soy                                    462
2 wheat                                  479
```

一方、ALL キーワードを使用している以下のバリエーションを考えてみます。

```
WITH SET set1 AS '{allerd.eggs,allerd.soy,allerd.wheat,allerd.wheat}'
SET set2 AS '{allerd.[dairy products],allerd.pollen,allerd.soy,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, INTERSECT(set1,set2,ALL) ON 1 FROM patients
                               Patient Count
1 soy                                    462
2 wheat                                  479
3 wheat                                  479
```

最後になりますが、より関心のあるセットを引数として使用することも、もちろんできます。以下はその例です。

```
WITH SET set1 AS 'TOPCOUNT(homed.city.members,5,MEASURES.[avg allergy count])'
SET set2 AS 'TOPCOUNT(homed.city.members,5,MEASURES.[avg age])'
SELECT MEASURES.[%COUNT] ON 0, INTERSECT(set1,set2) ON 1 FROM patients
                               Patient Count
1 Centerville                          1,155
2 Magnolia                             1,156
```

## 関連項目

- EXCEPT
- UNION

# ISNULL (MDX)

スカラ MDX 式を評価し、その値または代替値 (式の値が NULL の場合) のどちらかを返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、関数で使用するものと同じタイプの式を返します。

## 構文および詳細

```
ISNULL(MDX_expression,value_if_null)
```

以下は、この指定の説明です。

- MDX_expression は、スカラ MDX 値 (数値、文字列、または論理式) です。
- value_if_null は、MDX_expression が NULL と評価された場合に返す値です。

## 例

以下に簡単な例を示します。

```
SELECT ISNULL(MEASURES.[%COUNT],"None") ON 0, birthd.decade.MEMBERS ON 1 FROM patients where diagd.chd
```

```
                                    ISNULL
 1 1910s                               None
 2 1920s                                  1
 3 1930s                                 12
 4 1940s                                  3
 5 1950s                                 10
 6 1960s                                  3
 7 1970s                               None
 8 1980s                                  1
 9 1990s                               None
10 2000s                               None
11 2010s                               None
```

## 関連項目

- IIF
- %LABEL

# LAG (MDX)

レベル・メンバおよび負でない整数を指定すると、この関数は、レベル内で逆方向にカウントして前のメンバを返します。詳細は、時間ディメンジョンとデータ・ディメンジョンで異なります。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.LAG(optional_integer_expression)
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

- optional_integer_expression は、負でない整数リテラルです。

  この引数の既定値は 0 です。この場合、この関数は、member_expression で指定されたメンバを返します。

  integer_expression が 1 の場合、この関数は、PREVMEMBER 関数と同等です。

この関数は、指定されたメンバが属するレベルのメンバを検証し、(integer_expression を使用して) 現在のメンバから逆方向にカウントし、その位置のメンバを返します。時間ディメンジョンでは、この関数はすべての親レベルを無視します。データ・ディメンジョンでは、この関数は親レベルを考慮し、指定された親メンバ内で現在のメンバから逆方向にカウントします。(ここでいう時間ディメンジョンとデータ・ディメンジョンは、キューブで定義されているディメンジョン・タイプのみを参照します。"InterSystems Business Intelligence のモデルの定義" を参照してください。)

どの時間ディメンジョン内でも、この関数は、日付部分に基づく時間レベル (月だけでレコードをグループ化する Month など) よりも、時間軸に基づく時間レベル (年と月でレコードをグループ化する Period など) で役立ちます。レベルが日付部分に基づく場合、この関数では、セットの末尾を超えたレベルを参照すると null が返されます。同様のシナリオの例については、PREVMEMBER を参照してください。詳細は、"時間レベルの概要" を参照してください。

## 例

最初の例は時間ディメンジョンを使用します。参考として示された以下のクエリを検討してください。

```
SELECT MEASURES.[%COUNT] ON 0,
{birthd.1948,birthd.1949,birthd.1950,birthd.1951,birthd.1952} ON 1
FROM patients
```

```
                              Patient Count
1 1948                                   10
2 1949                                    4
3 1950                                   12
4 1951                                    8
5 1952                                    6
```

以下のクエリは LAG を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1951.LAG(1) ON 1 FROM patients
                              Patient Count
1950                                     12
```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1951.LAG(2) ON 1 FROM patients
                              Patient Count
1949                                      4
```

この例では、Year レベルは Decade レベルの子になります。つまり、メンバ 1949 と 1950 は異なる親に属します。示されているとおり、時間ディメンジョンで LAG 関数を使用すると、この関数は親レベルを無視します。

2 番目の例では、データ・ディメンジョン (HomeD ディメンジョン) を使用します。このディメンジョンの階層を表示するには、FIRSTCHILD 関数の例を参照してください。以下のクエリは、このディメンジョンで LAG を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Magnolia.LAG(1) ON 1 FROM patients
```

```
                            Patient Count
Cypress                                104
```

これはデータ・ディメンジョンであるため、このクエリは、親 ZIP Code 内の City レベルの前のメンバを検索します。この ZIP Code 内では、Cypress が最初の City であるため、以下のクエリは結果を返しません。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Cypress.LAG(1) ON 1 FROM patients
```

```
                            Patient Count
                                        *
```

## 関連項目

- LEAD
- NEXTMEMBER
- PREVMEMBER

# LASTCHILD (MDX)

指定されたメンバの最後の子を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.LASTCHILD
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、このメンバの最後の子を返します。どの子が最後かを判別するために、この関数は、子のセットの既定の順序を考慮します。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.[34577].LASTCHILD ON 1 FROM patients

                              Patient Count
Pine                                  1,100
```

この例で使用されている階層を図で参照するには、FIRSTCHILD 関数を参照してください。

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[1950].LASTCHILD ON 1 FROM patients
                              Patient Count
Dec-1950                                 8
```

## 関連項目

- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTSIBLING
- FIRSTCHILD
- LASTSIBLING
- PARENT
- SIBLINGS

# LASTSIBLING (MDX)

指定されたメンバの最後の兄弟を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.LASTSIBLING
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、指定されたメンバの親のすべての子を検証し、(そのセットの既定の順序を考慮して) そのセットの最後のメンバを返します。

この関数は、引数として使用したメンバと同じメンバを返すこともできます (そのメンバが最後の兄弟の場合)。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[Mar 2003].LASTSIBLING ON 1 FROM patients
```

```
                            Patient Count
Dec-2003                               17
```

## 関連項目

- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- PARENT
- SIBLINGS

# LEAD (MDX)

レベル・メンバおよび負でない整数を指定すると、この関数は、レベル内で順方向にカウントして後のメンバを返します。詳細は、時間ディメンジョンとデータ・ディメンジョンで異なります。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.LEAD(optional_integer_expression)
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

- optional_integer_expression は、負でない整数リテラルです。

  この引数の既定値は 0 です。この場合、この関数は、member_expression で指定されたメンバを返します。

  integer_expression が 1 の場合、この関数は、NEXTMEMBER 関数と同等です。

この関数は、指定されたメンバが属するレベルのメンバを検証し、(integer_expression を使用して) 現在のメンバから順方向にカウントし、その位置のメンバを返します。時間ディメンジョンでは、この関数はすべての親レベルを無視します。データ・ディメンジョンでは、この関数は親レベルを考慮し、指定された親メンバ内で現在のメンバからカウントします。(ここでいう時間ディメンジョンとデータ・ディメンジョンは、キューブで定義されているディメンジョン・タイプのみを参照します。"InterSystems Business Intelligence のモデルの定義" を参照してください。)

どの時間ディメンジョン内でも、この関数は、日付部分に基づく時間レベル (月だけでレコードをグループ化する Month など) よりも、時間軸に基づく時間レベル (年と月でレコードをグループ化する Period など) で役立ちます。レベルが日付部分に基づく場合、この関数では、セットの末尾を超えたレベルを参照すると null が返されます。同様のシナリオの例については、NEXTMEMBER を参照してください。詳細は、"時間レベルの概要" を参照してください。

## 例

最初の例は時間ディメンジョンを使用します。参考として示された以下のクエリを検討してください。

```
SELECT MEASURES.[%COUNT] ON 0,
{birthd.1948,birthd.1949,birthd.1950,birthd.1951,birthd.1952} ON 1
FROM patients
```

```
                              Patient Count
1 1948                                   10
2 1949                                    4
3 1950                                   12
4 1951                                    8
5 1952                                    6
```

以下のクエリは LEAD を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1948.LEAD(1) ON 1 FROM patients
```

```
                              Patient Count
1949                                     4
```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1948.LEAD(3) ON 1 FROM patients
```

```
                              Patient Count
1951                                     8
```

この例では、Year レベルは Decade レベルの子になります。つまり、メンバ 1948 と 1951 は異なる親に属します。示されているとおり、時間ディメンジョンで LEAD 関数を使用すると、この関数は親レベルを無視します。

2 番目の例では、データ・ディメンジョン (HomeD ディメンジョン) を使用します。このディメンジョンの階層を表示するには、FIRSTCHILD 関数の例を参照してください。以下のクエリは、このディメンジョンで LEAD を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Magnolia.LEAD(1) ON 1 FROM patients
```

```
                            Patient Count
Pine                                  114
```

これはデータ・ディメンジョンであるため、このクエリは、親 ZIP Code 内の City レベルの次のメンバを検索します。この ZIP Code 内では、Pine が最後の City であるため、以下のクエリは結果を返しません。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Pine.LEAD(1) ON 1 FROM patients
```

```
                            Patient Count
                                        *
```

## 関連項目

- LAG
- NEXTMEMBER
- PREVMEMBER

# LOG (MDX)

指定された数値の常用対数を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
LOG(numeric_expression)
```

以下は、この指定の説明です。

- numeric_expression は、数値に評価される式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

## 例

最初の例は、%COUNT メジャーの常用対数を示しています。

```
SELECT LOG(MEASURES.[%COUNT]) ON 0 FROM patients
```

```
                                        LOG
                                          4
```

次の例は、%LABEL 関数を使用して、より詳細なキャプションを適用します。

```
SELECT %LABEL(LOG(MEASURES.[%COUNT]),"LOG PAT CNT") ON 0 FROM patients
```

```
                                    LOG PAT CNT
                                              4
```

## 関連項目

- POWER
- SQRT

# LOOKUP (MDX)

指定されたキーを条件リストで検索し、代替文字列を返します。この関数によって、クエリ内で文字列置換を実行することが可能になります。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、文字列を返します。

## 構文および詳細

```
LOOKUP(term_list_name, lookup_value,default,alternative_field)
```

引数は以下に示す文字列式です。

- term_list_name は、条件リストの名前として評価されます。
- lookup_value は、条件リストで検索する文字列として評価されます。
- default (オプション) は、lookup_value が条件リストに見つからなかった場合に返す値として評価されます。
- alternative_field (オプション) は、返すフィールドの名前です。既定値は "value" です。

  この引数では大文字と小文字は区別されません。

この関数は、指定された条件リストを調査して、その "key" フィールドが lookup_value で指定された文字列に等しい用語を見つけて、alternative_field で指定されたフィールドに格納されている値を返します。

すべての条件リストには、少なくとも "key" と "value" の 2 つのフィールドが存在します。フィールドは追加できます。詳細は、"条件リストの定義" を参照してください。

システムは、すべての条件リストの最終変更日時を示すタイム・スタンプを記録しており、古くなった条件リストを使用するクエリ・キャッシュは無効にします。

## 例

例えば、HoleFoods に対して、Teams という名前の条件リストが以下のように定義されているとします。

Terms

| key | value |
|---|---|
| Atlanta | Braves |
| Boston | Red Sox |
| New York | Yankees |

この条件リストを使用する単純なクエリを以下に示します。

```
SELECT Lookup("Teams",Outlet.Boston.Properties("NAME")) ON ROWS FROM HOLEFOODS
Lookup                              Red Sox
```

より複雑なクエリを以下に示します。

```
SELECT Lookup("Teams",Outlet.CURRENTMEMBER.Properties("NAME"),"No Team") ON 0,
outlet.city.MEMBERS ON 1 FROM HOLEFOODS
```

```
                                   Lookup
 1 Amsterdam                       No Team
 2 Antwerp                         No Team
 3 Atlanta                          Braves
 4 Bangalore                       No Team
 5 Barcelona                       No Team
 6 Beijing                         No Team
 7 Berlin                          No Team
 8 Boston                          Red Sox
 9 Brasilia                        No Team
...
```

## 関連項目

- %LOOKUP (″他の条件リスト関数との比較″ を含む)
- %TERMLIST

# MAX (MDX)

セットのすべての要素にわたって、指定された式 (または現在のメジャー) の最大値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
MAX(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の最大のものを返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
 1 1910s                        80             5,359              75.17
 2 1920s                       227            12,910              74.20
 3 1930s                       567            33,211              74.67
 4 1940s                       724            38,420              73.39
 5 1950s                     1,079            46,883              73.72
 6 1960s                     1,475            57,814              74.16
 7 1970s                     1,549            49,794              74.35
 8 1980s                     1,333            35,919              74.13
 9 1990s                     1,426            29,219              74.79
10 2000s                     1,406            20,072              74.95
11 2010s                       134             1,346              73.55
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの最大値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
MAX(birthd.decade.MEMBERS) ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
MAX                          1,549            57,814              75.17
```

ここで、それぞれの値は、上記クエリの列内の値の最大です。例えば、Patient Count 値は、上記クエリの Patient Count 値の最大です。

これらの最大値は、同じメンバに属していません。例えば、患者数が最高である 10 年間は、平均テスト・スコアが最高である 10 年間と同じではありません。

別の例では、MAX の 2 番目の引数を使用します。

```
SELECT MAX(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    MAX
                                  1,549
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# MEDIAN (MDX)

セットのうち、指定された式に対して値が NULL でないすべての要素にわたって、この式 (または現在のメジャー) の中央値に最も近い値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
MEDIAN(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、この値が NULL であるセット要素があればすべて無視し、残りの要素の中央値に最も近い値を検索します。

Tipヒント 代わりに、最下位レベルの全レコードの中央値を検索するには、サンプル・プラグイン・クラス %DeepSee.PlugIn.Median で %KPI 関数を使用します。例を参照するには、"BI サンプルのアクセス方法" の説明に従って、サンプルをダウンロードして設定します。その後、サンプルをダウンロードしたネームスペースでユーザ・ポータルを表示し、KPIs & Plug-ins フォルダ内のサンプル・ダッシュボードを確認します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
 1 1910s                       80              5,359              75.17
 2 1920s                      227             12,910              74.20
 3 1930s                      567             33,211              74.67
 4 1940s                      724             38,420              73.39
 5 1950s                    1,079             46,883              73.72
 6 1960s                    1,475             57,814              74.16
 7 1970s                    1,549             49,794              74.35
 8 1980s                    1,333             35,919              74.13
 9 1990s                    1,426             29,219              74.79
10 2000s                    1,406             20,072              74.95
11 2010s                      134              1,346              73.55
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの平均値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
MEDIAN(birthd.decade.MEMBERS) ON 1 FROM patients
                    Patient Count    Encounter Count     Avg Test Score
MEDIAN                      1,079             33,211              74.20
```

ここで、それぞれの値は、上記クエリの列内の値の中央です。例えば、Patient Count 値は、上記クエリの Patient Count 値の中央値です。

別の例では、MEDIAN の 2 番目の引数を使用します。

```
SELECT MEDIAN(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    MEDIAN
                                      1,079
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# MEMBERS (MDX)

指定されたレベルまたは階層のすべてのメンバ (計算メンバは除く) のセットを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
level_expression.MEMBERS
```

または、以下のようにします。

```
hierarchy_expression.MEMBERS
```

または、以下のようにします。

```
dimension_expression.MEMBERS
```

以下は、この指定の説明です。

- level_expression は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

- hierarchy_expression は、階層を返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name]
  ```

- dimension_expression はディメンジョンの名前で、必要に応じて角括弧に囲まれます (″識別子″ を参照)。以下はその例です。

  ```
  [dimension_name]
  ```

  システムはこれを、指定されたディメンジョン内の最初に表示される階層への参照と解釈します。

レベル名を指定すると、この関数は、そのレベルのメンバ (計算メンバは除く) で構成されるセットを返します。このメンバは、キューブのレベル定義で指定された既定の順序になります。この既定の順序は、以下のとおりです。

- 日付以外のレベルの場合、キューブで異なる並べ替え順序が指定されていない限り、メンバは名前のアルファベットの昇順に並べ替えられます。システムには、各レベルの既定の並べ替え順序に対して柔軟なオプションが用意されています。
- 日付レベルの場合、メンバは発生順に並べ替えられます。

階層名式を指定すると、この関数は、その階層にあるすべてのレベルのメンバ (定義されていれば All メンバを含む) で構成されるセットを返します。このメンバは、階層順に返されます。階層順の詳細は、HIERARCHIZE 関数を参照してください。

ディメンジョン名を指定すると、この関数は、そのディメンジョンの最初に表示される階層にあるすべてのレベルのメンバで構成されるセットを返します。

## 例

以下のクエリは、Home Zip ディメンジョンのすべてのメンバを行として表示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.MEMBERS ON 1 FROM patients
                               Patient Count
1 32006                                 2,272
2 32007                                 1,111
3 34577                                 3,399
4 36711                                 1,069
5 38928                                 2,149
```

以下のクエリは、homed.h1 階層にあるすべてのレベルのすべてのメンバを行として表示します。

```
 1 32006                                2,272
 2 Juniper                              1,155
 3 Spruce                               1,117
 4 32007                                1,111
 5 Redwood                              1,111
 6 34577                                3,399
 7 Cypress                              1,150
 8 Magnolia                             1,111
 9 Pine                                 1,138
10 36711                                1,069
11 Centerville                          1,069
12 38928                                2,149
13 Cedar Falls                          1,045
14 Elm Heights                          1,104
```

## 関連項目

- ALLMEMBERS

# MIN (MDX)

セットのすべての要素にわたって、指定された式 (または現在のメジャー) の NULL でない最小値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
MIN(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、NULL があればすべて無視して、これらの値の最小のものを返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
 1 1910s                        80              5,359              75.17
 2 1920s                       227             12,910              74.20
 3 1930s                       567             33,211              74.67
 4 1940s                       724             38,420              73.39
 5 1950s                     1,079             46,883              73.72
 6 1960s                     1,475             57,814              74.16
 7 1970s                     1,549             49,794              74.35
 8 1980s                     1,333             35,919              74.13
 9 1990s                     1,426             29,219              74.79
10 2000s                     1,406             20,072              74.95
11 2010s                       134              1,346              73.55
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの最小値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
MIN(birthd.decade.MEMBERS) ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
MIN                            80              1,346              73.39
```

ここで、それぞれの値は、上記クエリの列内の値の最小です。例えば、Patient Count 値は、上記クエリの Patient Count 値の最小です。

これらの最小値は、同じメンバに属していません。例えば、患者数が最低である 10 年間は、最高テスト・スコアが最低である 10 年間と同じではありません。

別の例では、MIN の 2 番目の引数を使用します。

```
SELECT MIN(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                      MIN
                                       80
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# NEXTMEMBER (MDX)

指定されたメンバが属するレベルの次のメンバを返します。詳細は、時間ディメンジョンとデータ・ディメンジョンで異なります。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.NEXTMEMBER
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、指定されたメンバが属するレベルのメンバを検証し、(そのセットの既定の順序を考慮して) そのセットの次のメンバを返します。データ・ディメンジョンでは、この関数は親レベルを考慮し、指定された親メンバ内で次のメンバを探します。(ここでいう時間ディメンジョンとデータ・ディメンジョンは、キューブで定義されているディメンジョン・タイプのみを参照します。"InterSystems Business Intelligence のモデルの定義" を参照してください。)

NEXTMEMBER 関数は、LEAD(1) と同等です。

どの時間ディメンジョン内でも、この関数は、日付部分に基づく時間レベル (月だけでレコードをグループ化する Month など) よりも、時間軸に基づく時間レベル (年と月でレコードをグループ化する Period など) で役立ちます。例を参照してください。詳細は、"時間レベルの概要" を参照してください。

## 例

最初の例は時間ディメンジョンを使用します。参考として示された以下のクエリを検討してください。

```
SELECT MEASURES.[%COUNT] ON 0,
{birthd.1948,birthd.1949,birthd.1950,birthd.1951,birthd.1952} ON 1
FROM patients
```

```
                              Patient Count
1 1948                                   10
2 1949                                    4
3 1950                                   12
4 1951                                    8
5 1952                                    6
```

以下のクエリは NEXTMEMBER を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1948.NEXTMEMBER ON 1 FROM patients
```

```
                              Patient Count
1949                                      4
```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1949.NEXTMEMBER ON 1 FROM patients
```

```
                              Patient Count
1950                                     12
```

この例では、Year レベルは Decade レベルの子になります。つまり、メンバ 1949 と 1950 は異なる親に属します。示されているとおり、時間ディメンジョンで NEXTMEMBER 関数を使用すると、この関数は親レベルを無視します。

2 番目の例では、データ・ディメンジョン (HomeD ディメンジョン) を使用します。このディメンジョンの階層を表示するには、FIRSTCHILD 関数の例を参照してください。以下のクエリは、このディメンジョンで NEXTMEMBER を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Magnolia.NEXTMEMBER ON 1 FROM patients
```

```
                            Patient Count
Pine                                  114
```

これはデータ・ディメンジョンであるため、このクエリは、親 ZIP Code 内の City レベルの次のメンバを検索します。この ZIP Code 内では、Pine が最後の City であるため、以下のクエリは結果を返しません。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Pine.NEXTMEMBER ON 1 FROM patients
```

```
                            Patient Count
                                        *
```

日付の部分に基づく時間レベルでは、この関数はレベルの最後のメンバについて null を返します。月だけでレコードをグループ化する Month レベルについて考えます。この関数に December を指定すると、エンジンによって null が返されます。

```
SELECT [BirthQD].[H1].[Month].[December].NEXTMEMBER ON 1 FROM patients
```

```

                                        *
```

詳細は、"時間レベルの概要" を参照してください。

## 関連項目

- LAG
- LEAD
- PREVMEMBER

# NONEMPTYCROSSJOIN (MDX)

指定されたセットのクロス積で構成されるセットを返します。NULL のタプルがあれば除外します。

## 返りタイプ

この関数は、タプルのセットを返します。

## 構文および詳細

```
NONEMPTYCROSSJOIN(set_expression1, set_expression2)
```

以下は、この指定の説明です。

- set_expression1 および set_expression2 は、メンバのセットに対して評価される式です。

この関数は、各セットのすべてのメンバを識別し、最初のセットの各メンバと 2 番目のセットの各メンバを組み合わせるタプルのセットを生成します。返されるセットには、空のタプルは含まれません。

注釈　複合キューブを参照するクエリでこの関数を使用すると、同じ結果が CROSSJOIN として返されます。サブクエリに同じ数の行が存在し、結合可能であるようにするため、このようにする必要があります。この場合、空のタプルを除外するには、関数の先頭にキーワード句 NON EMPTY を指定します。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0,
NONEMPTYCROSSJOIN(diagd.MEMBERS, aged.[age group].MEMBERS) ON 1 FROM patients
```

```
                              Patient Count
 1 None->0 to 29                        363
 2 None->30 to 59                       348
 3 None->60+                            120
 4 asthma->0 to 29                       36
 5 asthma->30 to 59                      37
 6 asthma->60+                           11
 7 CHD->30 to 59                         13
 8 CHD->60+                              23
 9 diabetes->0 to 29                      1
10 diabetes->30 to 59                    20
11 diabetes->60+                         25
12 osteoporosis->60+                     22
```

# OPENINGPERIOD (MDX)

指定されたメンバと同じレベルで、指定されたレベルの最初の子孫メンバを返します。この関数は主に、時間レベルで使用するためのものです。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
OPENINGPERIOD(ancestor_level,member_expression)
```

以下は、この指定の説明です。

- ancestor_level は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  このレベルは、member_expression の親レベル、またはそのメンバの祖先です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

レベルとメンバを指定すると、この関数は、指定されたレベルの子孫であり、かつメンバと同じレベルにある最初のメンバを返します。

## 例

以下のクエリは、Q3 2003 を含む年の最初の四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, OPENINGPERIOD(birthd.year,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
Q1 2003                                35
```

これに対して、以下のクエリは、Q3 2003 を含む 10 年間の最初の四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, OPENINGPERIOD(birthd.decade,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
Q1 2000                                33
```

## 関連項目

- ANCESTOR
- CLOSINGPERIOD
- COUSIN
- PERIODSTODATE

# ORDER (MDX)

指定どおりに整列されたセットを返します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
ORDER(set_expression, ordering_expression, optional_keyword)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットとは通常、メンバのセットです。
- ordering_expression は、セット要素の順序を決定する数値式または文字列式です。

  数値の場合、ordering_expression は通常、[MEASURES].[measure_name] です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値に従って並べ替えます。
- optional_keyword は、セット内に階層があった場合に、MDX でどのように処理するかを制御します。以下のキーワードのいずれかを使用します。
  - ASC ー これを使用すると、階層を保持したまま、(ordering_expression で返された値を使用して) 昇順で並べ替えが行われます。階層順の詳細は、"HIERARCHIZE" を参照してください。

    この関数では、キーワードを省略すると、この方法で並べ替えられます。
  - DESC ー これを使用すると、階層を保持したまま、(ordering_expression で返された値を使用して) 降順で並べ替えが行われます。
  - BASC ー これを使用すると、階層は保持されず、(ordering_expression で返された値を使用して) すべてのメンバが昇順で並べ替えられます。
  - BDESC ー これを使用すると、階層は保持されず、(ordering_expression で返された値を使用して) すべてのメンバが降順で並べ替えられます。

## 例

例えば、以下のクエリは、市町村を平均テスト・スコアで降順に並べ替え、各市町村が属する郵便番号を保持します。

```
SELECT MEASURES.[avg test score] ON 0,
ORDER(homed.city.MEMBERS, MEASURES.[avg test score], DESC) ON 1 FROM patients
                            Avg Test Score
1 Pine                                75.67
2 Magnolia                            74.65
3 Cypress                             74.61
4 Centerville                         74.85
5 Cedar Falls                         74.62
6 Elm Heights                         74.36
7 Juniper                             74.52
8 Spruce                              74.14
9 Redwood                             74.16
```

この例で使用されている階層を図で参照するには、FIRSTCHILD 関数を参照してください。

一方、以下の例では、BDESC キーワードを使用し、階層は無視しています。

```
SELECT MEASURES.[avg test score] ON 0,
ORDER(homed.city.MEMBERS, MEASURES.[avg test score], BDESC) ON 1 FROM patients
                            Avg Test Score
1 Pine                                75.67
2 Centerville                         74.85
3 Magnolia                            74.65
4 Cedar Falls                         74.62
5 Cypress                             74.61
6 Juniper                             74.52
7 Elm Heights                         74.36
8 Redwood                             74.16
9 Spruce                              74.14
```

別の例として、以下のクエリは、文字列メジャーを (計算メンバとして) 定義し、これを ORDER 関数で使用します。

```
WITH MEMBER measures.stringtest AS 'IIF(MEASURES.[avg test score]<75, "low","high")'
SELECT {MEASURES.[avg test score],MEASURES.stringtest} on 0,
ORDER(homed.city.MEMBERS,measures.stringtest,BASC) ON 1 FROM patients
                      Avg Test Score           stringtest
1 Pine                         75.67                 high
2 Cedar Falls                  74.62                  low
3 Centerville                  74.85                  low
4 Cypress                      74.61                  low
5 Elm Heights                  74.36                  low
6 Juniper                      74.52                  low
7 Magnolia                     74.65                  low
8 Redwood                      74.16                  low
9 Spruce                       74.14                  low
```

## 関連項目

- HIERARCHIZE

# PARALLELPERIOD (MDX)

参照メンバ、そのメンバの親レベル、および整数を指定すると、この関数は、親レベルで逆方向にカウントし、そのレベル内で前のメンバを見つけ、参照メンバと同じ位置にあるその子を返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
PARALLELPERIOD(level_expression,offset,member_expression)
```

以下は、この指定の説明です。

- level_expression は、レベルを返す式です。以下はその例です。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  このレベルは、階層内で、参照メンバを含む、より高いレベルである必要があります。
- offset は整数リテラルです。

  負の整数を使用できます。
- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

  これは、参照メンバとして使用されます。

指定されたメンバの場合、この関数は、指定されたレベル内の先祖を検証し、(offset を使用して) そのメンバから逆方向にカウントし、そのレベル内で別のメンバを見つけ、参照メンバと同じ位置にある子メンバを返します。

この関数は階層を無視します。つまり、2 つのメンバは、親が別々であったとしても、隣接していると見なすことができます。

## 例

例えば、以下のクエリは、1 年さかのぼって検索して、Q1 1943 と並列の四半期を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, PARALLELPERIOD(birthd.year,1,birthd.[Q1 1943]) ON 1 FROM patients
```

```
                              Patient Count
Q1 1942                                  22
```

これに対して、以下のクエリは、10 年間さかのぼって検索して、Q1 1943 と並列の四半期を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, PARALLELPERIOD(birthd.decade,1,birthd.[Q1 1943]) ON 1 FROM patients
```

```
                              Patient Count
Q1 1939                                  17
```

以前に記述したように、offset には負の整数を指定できます。以下のクエリは、3 年先まで検索して、Q1 1943 と並列の四半期を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, PARALLELPERIOD(birthd.year,-3,birthd.[Q1 1943]) ON 1 FROM patients
```

```
                              Patient Count
Q1 1946                                  18
```

## 関連項目

- COUSIN

# PARENT (MDX)

指定されたメンバの親であるメンバを返します。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.PARENT
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。返されるのは、ディメンジョンの All メンバとなる場合があります。

  この式は、メジャーを参照できません。

この関数は、このメンバの親を返します。

メジャーは別として、各メンバはレベルに属し、各レベルは階層に属し、階層はメンバ間の親子リレーションシップを定義します。システムは、各階層の最高部で (All という 1 つのメンバを持つ) All レベルを作成します (キューブがディメンジョンの All レベルを無効にしている場合は除く)。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.[Elm Heights].PARENT ON 1 FROM patients

                              Patient Count
38928                                 2,276
```

この例で使用されている階層を図で参照するには、FIRSTCHILD 関数を参照してください。

## 関連項目

- ANCESTOR
- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- LASTSIBLING
- SIBLINGS

# PERCENTILE (MDX)

セットのすべての要素にわたって、指定された式（または現在のメジャー）を評価し、指定された百分位数レベルの値を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
PERCENTILE(set_expression, optional_numeric_expression, optional_percentile_value)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです（これらがある場合）。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。
- optional_percentile_value は、検索する百分位数を表す数値リテラルです。例えば、30 を使用すると、30 番目の百分位数が検索されます。これは、他の値の 30 パーセントより大きい値です。

  この引数を省略すると、50 番目の百分位数が計算されます。

この関数は、セットの各要素に対して数値を評価し、指定された百分位数となる値を返します。

Tipヒント 代わりに、最下位レベルのレコード全体の百分位の値を検索するには、サンプル・プラグイン・クラス %DeepSee.PlugIn.Percentile で %KPI 関数を使用します。例を参照するには、"BI サンプルのアクセス方法" の説明に従って、サンプルをダウンロードして設定します。その後、サンプルをダウンロードしたネームスペースでユーザ・ポータルを表示し、KPIs & Plug-ins フォルダ内のサンプル・ダッシュボードを確認します。

## 例

参考として、以下のクエリは、aged.year レベルのメンバに関する、Patient Count メジャーを示します。ORDER 関数は、このクエリの今後の結果と比較しやすくするために、これらのメンバを Patient Count の値の順に並べ替えます。

```
SELECT MEASURES.[%COUNT] ON 0, ORDER(birthd.year.MEMBERS,MEASURES.[%COUNT],BASC) ON 1 FROM patients
```

```
                              Patient Count
 1 1916                                   1
 2 1921                                   1
 3 1922                                   1
 4 1925                                   1
 5 1941                                   1
 6 1914                                   2
...
82 1967                                  18
83 1969                                  18
84 1973                                  18
85 1978                                  18
86 1979                                  18
87 1981                                  18
88 2002                                  18
89 2009                                  18
90 1968                                  19
91 1998                                  21
```

```
92 1991                                  23
93 2003                                  23
94 1977                                  25
```

次に、以下のクエリは、これらのメンバの 5 番目の百分位数の値を示します。

```
SELECT MEASURES.[%COUNT] ON 0, PERCENTILE(birthd.year.MEMBERS,,5) ON 1 FROM patients
```

```
                            Patient Count
5 Percentile                             1
```

つまり、5 番目の百分位数は、患者数が 1 人以下の誕生年から構成されます。

以下のクエリは、代わりに 95 番目の百分位数を示します。

```
SELECT MEASURES.[%COUNT] ON 0, PERCENTILE(birthd.year.MEMBERS,,95) ON 1 FROM patients
```

```
                            Patient Count
95 Percentile                           18
```

つまり、95 番目の百分位数は、患者数が 18 人以下の誕生年から構成されます。

別の例では、PERCENTILE の 2 番目の引数を使用します。

```
SELECT PERCENTILE(birthd.year.MEMBERS,MEASURES.[%COUNT],50) ON 1 FROM patients
```

```
50 Percentile                           10
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# PERCENTILERANK (MDX)

数値を指定した場合、この関数は、セットのすべての要素にわたって、指定された式（または現在のメジャー）を評価し、その式の百分位数ランク（指定された値以下のパーセント）を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
PERCENTILERANK(set_expression, numeric_expression, comparison_value)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。
- comparison_value は、すべてのセット・メンバの数値式をシステムが比較する際に対象とする値を表す数値リテラルです。

この関数は、セットの各要素に対して numeric_expression を評価し、その値を comparison_value と比較し、比較値の百分位数ランクを返します。

## 例

参考として、以下のクエリは、aged.year レベルのメンバに関する、Patient Count メジャーを示します。ORDER 関数は、このクエリの今後の結果と比較しやすくするために、これらのメンバを Patient Count の値の順に並べ替えます。

```
SELECT MEASURES.[%COUNT] ON 0, ORDER(birthd.year.MEMBERS,MEASURES.[%COUNT],BASC) ON 1 FROM patients
```

| | | Patient Count |
|---|---|---|
| 1 | 1916 | 1 |
| 2 | 1921 | 1 |
| 3 | 1922 | 1 |
| 4 | 1925 | 1 |
| 5 | 1941 | 1 |
| 6 | 1914 | 2 |
| 7 | 1918 | 2 |
| 8 | 1920 | 3 |
| 9 | 1927 | 3 |
| 10 | 1931 | 3 |
| 11 | 1934 | 3 |
| 12 | 1926 | 4 |
| 13 | 1932 | 4 |
| 14 | 1949 | 4 |
| 15 | 1955 | 4 |
| 16 | 1956 | 4 |
| 17 | 1928 | 5 |
| 18 | 1930 | 5 |
| 19 | 1937 | 5 |
| 20 | 1938 | 5 |
| 21 | 1966 | 5 |
| 22 | 1929 | 6 |
| 23 | 1940 | 6 |
| 24 | 1952 | 6 |
| 25 | 1939 | 7 |
| ... | | |
| 93 | 2003 | 23 |
| 94 | 1977 | 25 |

次に、以下のクエリは、5 人の患者がいるメンバの百分位数ランクを示します。

```
SELECT PERCENTILERANK(birthd.year.MEMBERS,MEASURES.[%Count],5) ON 1 FROM patients
```

```
Rank of 5                              19.68
```

つまり、患者数が 5 人以下の誕生年は、年セットの 19.68% となります。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- STDDEV
- STDDEVP
- SUM
- VAR
- VARP

# PERIODSTODATE (MDX)

指定されたレベルの子メンバまたは子孫メンバのセットを返します (指定されたメンバまで)。この関数は主に、時間レベルで使用するためのものです。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
PERIODSTODATE(ancestor_level,member_expression)
```

以下は、この指定の説明です。

- ancestor_level は、レベルを返す式です。例を以下に示します。

  ```
  [dimension_name].[hierarchy_name].[level_name]
  ```

  このレベルは、member_expression で指定されたメンバの先祖である必要があります。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

レベルとメンバを指定すると、この関数は、指定されたレベルの子孫である最初のメンバから指定されたメンバまでの、一定範囲のメンバで構成されるセットを返します。このメンバは、キューブのレベル定義で指定された既定の順序になります。

## 例

以下のクエリは、2003 年内の Q3 2003 までのすべての四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, PERIODSTODATE(birthd.year,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
1 Q1 2003                              35
2 Q2 2003                              44
3 Q3 2003                              43
```

これに対して、以下のクエリは、この 10 年間の Q3 2003 までのすべての四半期を表示します。

```
SELECT MEASURES.[%COUNT] ON 0, PERIODSTODATE(birthd.decade,birthd.[Q3 2003]) ON 1 FROM patients
```

```
                            Patient Count
 1 Q1 2000                             33
 2 Q2 2000                             33
 3 Q3 2000                             45
 4 Q4 2000                             39
 5 Q1 2001                             37
 6 Q2 2001                             40
 7 Q3 2001                             36
 8 Q4 2001                             37
 9 Q1 2002                             39
10 Q2 2002                             35
11 Q3 2002                             40
12 Q4 2002                             38
13 Q1 2003                             35
14 Q2 2003                             44
15 Q3 2003                             43
```

この関数は、メンバの範囲の形式でセット式を返します。つまり、例えば、以下の 2 つの式は同等です。

```
PERIODSTODATE(birthd.decade,birthd.[Q3 2003])
birthd.[Q1 2000]:birthd.[Q3 2003]
```

## 関連項目

- ANCESTOR
- COUSIN
- CLOSINGPERIOD
- OPENINGPERIOD

# POWER (MDX)

指定された数値を、2 番目の引数のべき乗に累乗して返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
POWER(numeric_expression,power)
```

以下は、この指定の説明です。

- numeric_expression は、数値に評価される式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

- power は、数値式です。

  通常、この式は整数です。

  1 未満の 10 進数値を使用する場合は、先頭は必ず 0. にしてください (例えば 0.5)。

## 例

最初の例は、3 乗に累乗される %COUNT メジャーを示しています。

```
SELECT POWER(MEASURES.[%COUNT],3) ON 0 FROM patients
```

```
                                POWER
                    1,000,000,000,000
```

次の例は、%LABEL 関数を使用して、より詳細なキャプションを適用します。

```
SELECT %LABEL(POWER(MEASURES.[%COUNT],3),"PAT CNT^3") ON 0 FROM patients
```

```
                                PAT CNT^3
                    1,000,000,000,000
```

以下の例は、小数のべき乗を示しています。

```
SELECT POWER(MEASURES.[%COUNT],0.5) ON 0 FROM patients
```

```
                                    POWER
                                       100
```

## 関連項目

- LOG
- SQRT

# PREVMEMBER (MDX)

指定されたメンバが属するレベルの前のメンバを返します。詳細は、時間ディメンジョンとデータ・ディメンジョンで異なります。

## 返りタイプ

この関数は、メンバを返します。

## 構文および詳細

```
member_expression.PREVMEMBER
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

この関数は、指定されたメンバが属するレベルのメンバを検証し、(そのセットの既定の順序を考慮して) そのセットの前のメンバを返します。時間ディメンジョンでは、この関数はすべての親レベルを無視します。データ・ディメンジョンでは、この関数は親レベルを考慮し、指定された親メンバ内で現在のメンバから逆方向にカウントします。(ここでいう時間ディメンジョンとデータ・ディメンジョンは、キューブで定義されているディメンジョン・タイプのみを参照します。"InterSystems Business Intelligence のモデルの定義" を参照してください。)

PREVMEMBER 関数は、LAG(1) と同等です。

どの時間ディメンジョン内でも、この関数は、日付部分に基づく時間レベル (月だけでレコードをグループ化する Month など) よりも、時間軸に基づく時間レベル (年と月 でレコードをグループ化する Period など) で役立ちます。例を参照してください。詳細は、"時間レベルの概要" を参照してください。

## 例

最初の例は時間ディメンジョンを使用します。参考として示された以下のクエリを検討してください。

```
SELECT MEASURES.[%COUNT] ON 0,
{birthd.1948,birthd.1949,birthd.1950,birthd.1951,birthd.1952} ON 1
FROM patients
```

```
                              Patient Count
1 1948                                   10
2 1949                                    4
3 1950                                   12
4 1951                                    8
5 1952                                    6
```

以下のクエリは PREVMEMBER を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1951.PREVMEMBER ON 1 FROM patients
```

```
                              Patient Count
1950                                     12
```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.1950.PREVMEMBER ON 1 FROM patients
```

```
                              Patient Count
1949                                      4
```

この例では、Year レベルは Decade レベルの子になります。つまり、メンバ 1949 と 1950 は異なる親に属します。示されているとおり、時間ディメンジョンで PREVMEMBER 関数を使用すると、この関数は親レベルを無視します。

2 番目の例では、データ・ディメンジョン (HomeD ディメンジョン) を使用します。このディメンジョンの階層を表示するには、FIRSTCHILD 関数の例を参照してください。以下のクエリは、このディメンジョンで PREVMEMBER を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Magnolia.PREVMEMBER ON 1 FROM patients
```

```
                            Patient Count
Cypress                                104
```

これはデータ・ディメンジョンであるため、このクエリは、親 ZIP Code 内の City レベルの前のメンバを検索します。この ZIP Code 内では、Cypress が最初の City であるため、以下のクエリは結果を返しません。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.Cypress.PREVMEMBER ON 1 FROM patients
```

```
                            Patient Count
                                        *
```

日付の 部分に基づく時間レベルでは、この関数はレベルの最初のメンバについて null を返します。月だけでレコードをグループ化する Month レベルについて考えます。この関数に January を指定すると、エンジンによって null が返されます。

```
SELECT [BirthQD].[H1].[Month].[January].PREVMEMBER ON 1 FROM patients
```

```
                                        *
```

PREVMEMBER を使用して前回の期間に販売されたユニット数を取得する例については、%CELLZERO を参照してください。

## 関連項目

- LAG
- LEAD
- NEXTMEMBER

# PROPERTIES (MDX)

指定されたメンバに対して、指定されたプロパティの値を返します。

## 返りタイプ

この関数は、文字列を返します。

## 構文および詳細

```
member_expression.PROPERTIES(property_name,default_value))
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

- property_name は、プロパティの名前と等しい文字列です。

  すべてのメンバには、特定の内部プロパティがあります。これは、"内部プロパティ" にリストされています。キューブ定義には、追加のプロパティの定義を含めることができます。

- default_value は、メンバにこのプロパティの値がない場合に返されるオプションの値です。この引数を省略するか、指定されたメンバにプロパティがない場合、この関数は @NOPROPERTY を返します。

  この引数は、インターシステムズによる MDX への拡張機能です。

プロパティの名前は、大文字と小文字が区別されません。

## 例

以下の例は、内部プロパティである KEY プロパティの値を取得します。

```
SELECT docd.h1.CURRENTMEMBER.PROPERTIES("KEY") ON 0, docd.[doctor].MEMBERS ON 1 FROM patients
```

```
                                    Doctor
  1 None                            <null>
  2 Adam, Dan                           41
  3 Adam, Danielle                     391
  ...
```

以下のバリエーションは、%LABEL を使用して、わかりやすいキャプションを指定しています。

```
SELECT %LABEL(docd.h1.CURRENTMEMBER.PROPERTIES("key"),"key") ON 0,
docd.doctor.MEMBERS ON 1 FROM patients
```

```
                                     key
  1 None                            <null>
  2 Adam, Dan                           41
  3 Adam, Danielle                     391
  ...
```

以下の例は、CURRENTMEMBER を使用して、郵便番号を繰り返し、ID と LEVEL_NUMBER という 2 つの内部プロパティの値を取得します。

```
WITH SET test AS '{homed.h1.CURRENTMEMBER.PROPERTIES("id"),
homed.h1.CURRENTMEMBER.PROPERTIES("level_number")}'
SELECT test ON 0, homed.zip.MEMBERS ON 1 FROM patients
```

```
                            Home ZIP            Home ZIP
1 32006                            2                   1
2 32007                            4                   1
3 34577                            1                   1
4 36711                            5                   1
5 38928                            3                   1
```

バリエーションとして、以下のクエリは、%LABEL を使用して、わかりやすいキャプションを指定しています。

```
WITH SET test AS '{%LABEL(homed.h1.CURRENTMEMBER.PROPERTIES("id"),"id"),
%LABEL(homed.h1.CURRENTMEMBER.PROPERTIES("level_number"),"level_number")}'
SELECT test ON 0, homed.zip.MEMBERS ON 1 FROM patients
```

```
                                  id         level_number
1 32006                             2                    1
2 32007                             4                    1
3 34577                             1                    1
4 36711                             5                    1
5 38928                             3                    1
```

その他の例は、"CURRENTMEMBER" を参照してください。

# RANK (MDX)

指定されたセット内で、指定されたメンバのランクを示す整数を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
RANK(member_expression, set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。
- set_expression は、セットを返す式です。
- optional_numeric_expression は、セット内のメンバごとに関数が評価する数値式です。

  通常、この式は、[MEASURES].[measure_name] です。

  - この引数を指定すると、システムは特定のメンバと、セット内のその他すべてのメンバについて、この式を評価します。次に、システムは、他のメンバに比べて、指定されたメンバがどのランクかを示す整数を返します。値が最低のメンバは、位置 1 になります。
  - この引数を指定しないと、システムは指定されたセット内のこのメンバの順序位置を返します。最初の位置は 1 です。

## 例

例えば、以下のクエリは、メンバが患者数に基づいてランキングされる場合に、colord ディメンジョンのメンバのセット内でのメンバ colord.green のランクを示します。

```
SELECT RANK(colord.green, colord.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
```

```
                                    Green
Results                                 2
```

これが正しいことを確認するには、以下のクエリを考えてみます。これは、このディメンジョンのメンバを患者数で並べ替えます。

```
SELECT MEASURES.[%COUNT] ON 0,
ORDER(colord.MEMBERS, MEASURES.[%COUNT]) ON 1 FROM patients
```

```
                              Patient Count
1 None                                1,243
2 Green                               1,304
3 Blue                                2,381
4 Orange                              1,302
5 Purple                              1,276
6 Red                                 1,244
7 Yellow                              1,250
```

# ROUND (MDX)

数値 MDX 式を評価して、丸めた値を返します。この関数は、インターシステムズによる MDX への拡張機能です。

## 返りタイプ

この関数は、数値式を返します。

## 構文および詳細

```
ROUND(numeric_expression,decimal_places)
```

以下は、この指定の説明です。

- numeric_expression は、数値式です。
- decimal_places は、返り値に使用する小数点以下桁数を指定する整数リテラルです。既定値は 0 です。

## 例

以下に簡単な例を示します。

```
SELECT ROUND(MEASURES.[avg allergy count],2) ON 0,
patgrpd.[patient group].MEMBERS ON 1 FROM patients
```

```
                                    ROUND
1 Group A                             0.97
2 Group B                             1.06
3 None                                0.97
```

# SIBLINGS (MDX)

指定されたメンバおよびそのすべての兄弟を含むセットを返します。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
member_expression.SIBLINGS
```

以下は、この指定の説明です。

- member_expression は、メンバを返す式です。

  この式は、メジャーを参照できません。

## 例

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.cypress.SIBLINGS ON 1 FROM patients
```

```
                              Patient Count
1 Cypress                             1,089
2 Magnolia                            1,073
3 Pine                                1,039
```

この例で使用されている階層を図で参照するには、FIRSTCHILD 関数を参照してください。

## 関連項目

- CHILDREN
- COUSIN
- DESCENDANTS
- FIRSTCHILD
- FIRSTSIBLING
- LASTCHILD
- LASTSIBLING
- PARENT

# SQRT (MDX)

指定された数値の平方根を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
SQRT(numeric_expression)
```

以下は、この指定の説明です。

- numeric_expression は、数値に評価される式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

## 例

最初の例は、Patient Count メジャーの平方根を示しています。

```
SELECT SQRT(MEASURES.[%COUNT]) ON 0 FROM patients
```

```
                                    SQRT
                                      100
```

次の例は、%LABEL 関数を使用して、より詳細なキャプションを適用します。

```
SELECT %LABEL(SQRT(MEASURES.[%COUNT]),"SQRT PAT CNT") ON 0 FROM patients
```

```
                             SQRT PAT CNT
                                      100
```

## 関連項目

- LOG
- POWER

# STDDEV (MDX)

セットのすべての要素にわたって、指定された式 (または現在のメジャー) の標準偏差を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
STDDEV(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の標準偏差を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、2 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                      Patient Count      Encounter Count
 1 1910s                         80                5,359
 2 1920s                        227               12,910
 3 1930s                        567               33,211
 4 1940s                        724               38,420
 5 1950s                      1,079               46,883
 6 1960s                      1,475               57,814
 7 1970s                      1,549               49,794
 8 1980s                      1,333               35,919
 9 1990s                      1,426               29,219
10 2000s                      1,406               20,072
11 2010s                        134                1,346
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの標準偏差を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
STDDEV(birthd.decade.MEMBERS) ON 1 FROM patients
                      Patient Count      Encounter Count
STDDEV                       579.41           18,401.33
```

ここで、それぞれの値は、上記クエリの列内の値の標準偏差です。例えば、Patient Count 値は、上記クエリの Patient Count 値の標準偏差です。

別の例では、STDDEV の 2 番目の引数を使用します。

```
SELECT STDDEV(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    STDDEV
                                     579.41
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEVP
- SUM
- VAR
- VARP

# STDDEVP (MDX)

セットのすべての要素にわたって、指定された式の母標準偏差を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
STDDEVP(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。

この関数は、セットの各要素に対して数値を評価し、それらの値の母標準偏差を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、2 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                      Patient Count      Encounter Count
 1 1910s                         80                5,359
 2 1920s                        227               12,910
 3 1930s                        567               33,211
 4 1940s                        724               38,420
 5 1950s                      1,079               46,883
 6 1960s                      1,475               57,814
 7 1970s                      1,549               49,794
 8 1980s                      1,333               35,919
 9 1990s                      1,426               29,219
10 2000s                      1,406               20,072
11 2010s                        134                1,346
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの母標準偏差を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
STDDEVP(birthd.decade.MEMBERS) ON 1 FROM patients
                      Patient Count      Encounter Count
STDDEVP                      552.44            17,544.98
```

ここで、それぞれの値は、上記クエリの列内の値の母標準偏差です。例えば、Patient Count 値は、上記クエリの Patient Count 値の母標準偏差です。

別の例では、STDDEVP の 2 番目の引数を使用します。

```
SELECT STDDEVP(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    STDDEVP
                                      552.44
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE

- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- SUM
- VAR
- VARP

# STDEV (MDX)

STDDEV の同義語です。

## 関連項目

- STDDEV

# STDEVP (MDX)

STDDEVP の同義語です。

## 関連項目

- STDDEVP

# SUBSET (MDX)

位置に基づいて、指定されたセットから要素のセットを返します。最初のメンバは、位置 0 にあります。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
SUBSET(set_expression, first_element_position, optional_element_count)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。
- first_element_position は、返す最初の要素の位置を指定する整数リテラルです。

  最初の要素の位置は 0 です。
- optional_element_count は、返す要素の数の位置を指定する整数リテラルです。

  この引数を省略すると、この関数は、first_element_position にある要素、およびそれに続くすべての要素を返します。

## 例

```
SELECT MEASURES.[%COUNT] ON 0, SUBSET(homed.city.MEMBERS, 0, 3) ON 1 FROM patients
```

```
                              Patient Count
1 Cedar Falls                         1,188
2 Centerville                         1,155
3 Cypress                             1,221
```

一方、完全なセットを表示する以下の例を考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM patients
```

```
                              Patient Count
1 Cedar Falls                         1,188
2 Centerville                         1,155
3 Cypress                             1,221
4 Elm Heights                         1,266
5 Juniper                             1,197
6 Magnolia                            1,156
7 Pine                                1,139
8 Redwood                             1,144
9 Spruce                              1,135
```

# SUM (MDX)

セットのすべての要素にわたって、指定された式 (または現在のメジャー) の合計を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
SUM(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の合計を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、3 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
 1 1910s                        80              5,359              75.17
 2 1920s                       227             12,910              74.20
 3 1930s                       567             33,211              74.67
 4 1940s                       724             38,420              73.39
 5 1950s                     1,079             46,883              73.72
 6 1960s                     1,475             57,814              74.16
 7 1970s                     1,549             49,794              74.35
 8 1980s                     1,333             35,919              74.13
 9 1990s                     1,426             29,219              74.79
10 2000s                     1,406             20,072              74.95
11 2010s                       134              1,346              73.55
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの合計を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count],MEASURES.[avg test score]} ON 0,
SUM(birthd.decade.MEMBERS) ON 1 FROM patients
                     Patient Count    Encounter Count     Avg Test Score
SUM                         10,000            330,947             817.08
```

ここで、それぞれの値は、上記クエリの列内の値の合計です。例えば、Patient Count 値は、上記クエリの Patient Count 値の合計です。Avg Test Score 値は、平均テスト・スコアの合計で、有用な値になることはあまりないと思われます。

別の例では、SUM の 2 番目の引数を使用します。

```
SELECT SUM(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                      SUM
                                   10,000
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- VAR
- VARP

# TAIL (MDX)

セットの現在の順序を使用して、セットの末尾からサブセットを返します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
TAIL(set_expression, optional_integer_expression)
```

- set_expression は、セットに対して評価される式です。
- optional_integer_expression は整数リテラルです。

  この引数の既定値は 1 です。

  この関数は、この引数を使用して、サブセットで返す要素の数を決定します。

この関数は、指定されたセットの末尾から、(セットの現在の順序を考慮して) 指定された要素数で構成されるセットを返します。integer_expression が 1 未満の場合、この関数は空のセットを返します。integer_expression がセットの要素数より大きい場合、この関数は元のセットを返します。

サブセットの要素は、元のセットで指定された順序と同じ順序で返されます。

## 例

```
SELECT MEASURES.[%COUNT] ON 0, TAIL(birthd.decade.MEMBERS, 3) ON 1
FROM patients
                              Patient Count
1 1990s                               1,413
2 2000s                               1,433
3 2010s                                 155
```

## 関連項目

- HEAD

# TOPCOUNT (MDX)

目的の要素数を指定すると、セットを並べ替え、その値の高い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
TOPCOUNT(set_expression, element_count, optional_ordering_expression)
```

以下は、この指定の説明です。

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- element_count は整数リテラルです。

  この関数は、この引数を使用して、サブセットで返す要素の数を決定します。この引数が要素数よりも大きい場合、すべての要素が返されます。
- optional_ordering_expression は、セット要素の順序を決定する数値式です。

  通常、ordering_expression は [MEASURES].[measure_name] です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の降順で並べ替えます。階層があればすべて無視されます。

  この引数を省略すると、この関数はセット要素の現在の順序を使用します (そして、この関数は HEAD 関数のように動作します)。

## 例

まず、以下のクエリ、およびそれが返す結果について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPCOUNT(birthd.decade.MEMBERS, 100, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
 1 1970s                              1,520
 2 1960s                              1,500
 3 2000s                              1,433
 4 1990s                              1,413
 5 1980s                              1,400
 6 1950s                              1,030
 7 1940s                                683
 8 1930s                                572
 9 1920s                                223
10 2010s                                155
11 1910s                                 71
```

count_expression がメンバ数よりも大きいため、すべてのメンバが返されます。メンバは、%COUNT メジャーの値の昇順で並べ替えられます。

次に、類似のクエリについて考えてみます。こちらでは、使用する count_expression が 3 に等しくなっています。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPCOUNT(birthd.decade.MEMBERS, 3, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
1 1970s                              1,520
2 1960s                              1,500
3 2000s                              1,433
```

このクエリは、セットの中で、値の高い方からメンバを 3 つ選択します。

## 関連項目

- BOTTOMCOUNT

# TOPPERCENT (MDX)

全体の合計に適用される切り捨て値を指定すると、セットを並べ替え、その値の高い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
TOPPERCENT(set_expression, percentage, ordering_expression)
```

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- percentage は、100 以下の数値リテラルです。例えば、15 は 15 パーセントを表します。

  この関数は、この引数を使用して、サブセットで返す要素の切り捨てポイントを決定します。

  通常は、切り捨てポイントをまたぐメンバが存在します。このメンバは、下位のセットではなく、上位のセットに割り当てられます。その結果、返されるサブセットでは、ordering_expression の累計が、セット全体のパーセンテージを示す percentage よりも大きくなる可能性があります。
- ordering_expression は、セット要素の順序を決定する数値式です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の降順で並べ替えます。階層があればすべて無視されます。

## 例

まず、以下のクエリ、およびそれが返す結果について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPPERCENT(birthd.decade.MEMBERS, 100, MEASURES.[%COUNT]) ON 1 FROM patients
```

```
                              Patient Count
 1 2000s                                157
 2 1980s                                155
 3 1990s                                144
 4 1960s                                136
 5 1970s                                128
 6 1950s                                107
 7 1930s                                 56
 8 1940s                                 54
 9 2010s                                 44
10 1920s                                 13
11 1910s                                  6
```

percentage が 100 であるため、すべてのメンバが返されます。

ここで、上記の例のバリエーションを考えてみます。percentage が 50 であるため、上位 50 パーセントが表示されます。

```
SELECT MEASURES.[%COUNT] ON 0, TOPPERCENT(birthd.decade.MEMBERS, 50, MEASURES.[%COUNT]) ON 1 FROM
patients
```

```
                              Patient Count
1 2000s                                157
2 1980s                                155
3 1990s                                144
4 1960s                                136
```

これらのメンバの %COUNT メジャーの合計は、合計の 50% をわずかに上回ります (1960s が除外された場合は合計は 50% を下回ります)。

## 関連項目

- BOTTOMPERCENT

# TOPSUM (MDX)

要素全体の合計に適用される切り捨て値を指定すると、セットを並べ替え、その値の高い方からサブセットを返します。

## 返りタイプ

この関数は、使用されるセットに応じて、メンバまたはタプルのセットを返します。

## 構文および詳細

```
TOPSUM(set_expression, cutoff_value, ordering_expression)
```

- set_expression は、メンバまたはタプルのセットに対して評価される式です。
- cutoff_value は数値リテラルです。

  この関数は、この引数を使用して、サブセットで返す要素の切り捨て値を決定します。

  返されたサブセットのすべての要素に関して、ordering_expression の値の合計は、cutoff_value 以下になります。
- ordering_expression は、セット・メンバの順序を決定する数値式です。

  この関数は、セットの各要素に対してこの式を評価し、セットの要素をこの値の降順で並べ替えます。階層があればすべて無視されます。

## 例

まず、切り捨て値が大きく、すべてのメンバが含まれる例について考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPSUM(birthd.decade.MEMBERS, 10000, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
 1 1970s                              1,520
 2 1960s                              1,500
 3 2000s                              1,433
 4 1990s                              1,413
 5 1980s                              1,400
 6 1950s                              1,030
 7 1940s                                683
 8 1930s                                572
 9 1920s                                223
10 2010s                                155
11 1910s                                 71
```

次に、切り捨て値を 2500 に設定したバリエーションについて考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPSUM(birthd.decade.MEMBERS, 2500, MEASURES.[%COUNT]) ON 1 FROM patients
                              Patient Count
1970s                                1,520
```

## 関連項目

- BOTTOMSUM

# UNION (MDX)

指定された 2 つのセットの要素で構成されたセットを返します。オプションで重複を除外します。

## 返りタイプ

この関数は、セットを返します。

## 構文および詳細

```
UNION(set_expression1, set_expression2, ALL)
```

または、以下のようにします。

```
UNION(set_expression1, set_expression2)
```

- set_expression1 および set_expression2 は、セットに対して評価される式です。
- オプションのキーワード ALL が含まれている場合、これは、すべての重複を保持することを指定します。既定では、返されたセットに、重複する要素が含まれている場合、それらのうち最初の要素のみが含まれます。

## 例

2 つの名前付きセットを定義する以下のクエリを考えてみます。

```
WITH SET set1 AS '{allerd.eggs,allerd.soy,allerd.wheat}'
SET set2 AS '{allerd.[dairy products],allerd.pollen,allerd.soy,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, UNION(set1,set2) ON 1 FROM patients
                              Patient Count
1 eggs                                  451
2 soy                                   462
3 wheat                                 479
4 dairy products                        463
5 pollen                                447
```

このクエリは、`set1` および `set2` にあるすべてのメンバを示します。

一方、ALL キーワードを使用して重複を保持している以下のバリエーションを考えてみます。

```
WITH SET set1 AS '{allerd.eggs,allerd.soy,allerd.wheat}'
SET set2 AS '{allerd.[dairy products],allerd.pollen,allerd.soy,allerd.wheat}'
SELECT MEASURES.[%COUNT] ON 0, UNION(set1,set2,ALL) ON 1 FROM patients
                              Patient Count
1 eggs                                  451
2 soy                                   462
3 wheat                                 479
4 dairy products                        463
5 pollen                                447
6 soy                                   462
7 wheat                                 479
```

最後になりますが、より関心のあるセットを引数として使用することも、もちろんできます。以下はその例です。

```
WITH SET set1 AS 'TOPCOUNT(homed.city.members,5,MEASURES.[avg allergy count])'
SET set2 AS 'TOPCOUNT(homed.city.members,5,MEASURES.[avg age])'
SELECT MEASURES.[%COUNT] ON 0, UNION(set1,set2) ON 1 FROM patients
          Patient Count
1 Juniper                              1,197
2 Spruce                               1,135
3 Centerville                          1,155
4 Redwood                              1,144
5 Magnolia                             1,156
6 Cedar Falls                          1,188
7 Elm Heights                          1,266
8 Pine                                 1,139
```

## 関連項目

- EXCEPT
- INTERSECT

# VAR (MDX)

セットのすべての要素にわたって、指定された式 (または現在のメジャー) の分散を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
VAR(set_expression, optional_numeric_expression)
```

または、以下のようにします。

```
VAR(set_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の分散を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、2 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                      Patient Count      Encounter Count
 1 1910s                         80                5,359
 2 1920s                        227               12,910
 3 1930s                        567               33,211
 4 1940s                        724               38,420
 5 1950s                      1,079               46,883
 6 1960s                      1,475               57,814
 7 1970s                      1,549               49,794
 8 1980s                      1,333               35,919
 9 1990s                      1,426               29,219
10 2000s                      1,406               20,072
11 2010s                        134                1,346
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの分散を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
VAR(birthd.decade.MEMBERS) ON 1 FROM patients
                      Patient Count      Encounter Count
VAR                       335,710.89        338,609,051.69
```

ここで、それぞれの値は、上記クエリの列内の値の分散です。例えば、Patient Count 値は、上記クエリの Patient Count 値の分散です。

別の例では、VAR の 2 番目の引数を使用します。

```
SELECT VAR(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    VAR
                                335,710.89
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VARP

# VARIANCE (MDX)

VAR の同義語です。

## 関連項目

- VAR

# VARIANCEP (MDX)

VARP の同義語です。

## 関連項目

- VARP

# VARP (MDX)

セットのすべての要素にわたって、指定された式の母分散を返します。

## 返りタイプ

この関数は、数値を返します。

## 構文および詳細

```
VARP(set_expression, optional_numeric_expression)
```

以下は、この指定の説明です。

- set_expression は、セットに対して評価される式です。このセットは通常、メンバまたはタプルのセットです。
- optional_numeric_expression は、セット要素ごとにこの関数が評価する数値式です。

  通常、この式の形式は、[MEASURES].[measure_name] です。

  数値式を指定しない場合、システムは現在の結果セルで使用されるメジャーを使用します。例えば、0 軸で使用されたメジャーまたは WHERE 節で指定されたメジャーです (これらがある場合)。クエリ自体がメジャーを指定しない場合は、ファクト・テーブル内のレコードをカウントする %COUNT が代わりに使用されます。

この関数は、セットの各要素に対して数値を評価し、それらの値の母分散を返します。

## 例

まず、以下のクエリは、aged.decade レベルのメンバに関する、2 つのメジャーの値を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
birthd.decade.MEMBERS ON 1 FROM patients
                      Patient Count      Encounter Count
 1 1910s                         80                5,359
 2 1920s                        227               12,910
 3 1930s                        567               33,211
 4 1940s                        724               38,420
 5 1950s                      1,079               46,883
 6 1960s                      1,475               57,814
 7 1970s                      1,549               49,794
 8 1980s                      1,333               35,919
 9 1990s                      1,426               29,219
10 2000s                      1,406               20,072
11 2010s                        134                1,346
```

次に、以下のクエリは、このレベルのメンバに関する、これらのメジャーの母分散を示します。

```
SELECT {MEASURES.[%COUNT],MEASURES.[encounter count]} ON 0,
VARP(birthd.decade.MEMBERS) ON 1 FROM patients
                      Patient Count      Encounter Count
VARP                      305,191.72       307,826,410.63
```

ここで、それぞれの値は、上記クエリの列内の値の母分散です。例えば、Patient Count 値は、上記クエリの Patient Count 値の母分散です。

別の例では、VARP の 2 番目の引数を使用します。

```
SELECT VARP(birthd.decade.MEMBERS, MEASURES.[%COUNT]) ON 0 FROM patients
                                    VARP
                                305,191.72
```

追加の類似例は、"AVG" を参照してください。

## 関連項目

- AGGREGATE
- AVG
- MAX
- MEDIAN
- MIN
- PERCENTILE
- PERCENTILERANK
- STDDEV
- STDDEVP
- SUM
- VAR

# VISUALTOTALS (MDX)

階層順にメンバのセットを指定すると、その表示部分の合計と共にセットを返します。表示部分の合計では、任意の上位レベルのメンバの実際の値は、クエリに含まれている子の値の合計値に置き換えられます。

## 返りタイプ

この関数は、メンバのセットを返します。

## 構文および詳細

```
VISUALTOTALS(set_expression, optional_parent_name_pattern)
```

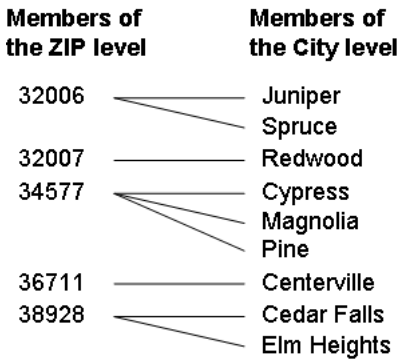
以下は、この指定の説明です。

- set_expression は、メンバのセットに対して評価される式です。このセットには、同じディメンジョン内の異なるレベルのメンバを含めることができますが、それらのメンバは階層順になっている必要があります。
- optional_parent_name_pattern は、親名が使用される場所にアスタリスク (*) がある文字列です。例えば、"SUB *" や "* (SUBTOTAL)" などです。

  これを省略すると、親の名前には余分な文字列が追加されません。

## 例

まず参考のために、この例で使用される階層を以下の図に示します。




以下のクエリについて考えてみます。ここでは VISUALTOTALS を使用していません。

```
WITH SET demo AS 'HIERARCHIZE({homed.32006,homed.34577,homed.CYPRESS,homed.PINE,homed.SPRUCE})'
SELECT MEASURES.[%COUNT] ON 0, demo ON 1 FROM patients
                              Patient Count
1 32006                              2,272
2 Spruce                             1,117
3 34577                              3,399
4 Cypress                            1,150
5 Pine                               1,138
```

このクエリは、リストされた郵便番号および市町村ごとの患者数を示しています。郵便番号ごとの患者数は、その郵便番号の患者数の合計です。

ここで、以下のバリエーションについて考えてみます。ここでは、VISUALTOTALS を使用しています。

```
WITH SET demo AS 'HIERARCHIZE({homed.32006,homed.34577,homed.CYPRESS,homed.PINE,homed.SPRUCE})'
SELECT MEASURES.[%COUNT] ON 0, VISUALTOTALS(demo) ON 1 FROM patients
                               Patient Count
1 32006                                 1,117
2 Spruce                                1,117
3 34577                                 2,288
4 Cypress                               1,150
5 Pine                                  1,138
```

この場合、すべての上位レベルのメンバ (郵便番号) の患者数には、クエリに含まれている子のみが反映されています。例えば、郵便番号 34577 の患者数は、Pine と Cypress の各市町村の患者数の合計です。

別のバリエーションとして、以下のクエリについて考えてみます。これは、上記のバリエーションに似ていますが、VISUALTOTALS の 2 番目の引数も使用している点が異なります。

```
VISUALTOTALS */WITH SET demo AS
'HIERARCHIZE({homed.32006,homed.34577,homed.CYPRESS,homed.PINE,homed.SPRUCE})'
SELECT MEASURES.[%COUNT] ON 0, VISUALTOTALS(demo,"* (included cities)") ON 1 FROM patients
                               Patient Count
1 32006 (included cities)               1,117
2 Spruce                                1,117
3 34577 (included cities)               2,288
4 Cypress                               1,150
5 Pine                                  1,138
```

表示される値は上記のクエリと同じですが、それぞれの郵便番号は、末尾に文字列 `(included cities)` が付いて表示されます。

## 関連項目

- HIERARCHIZE

# 内部プロパティ (MDX)

この参照セクションでは、Business Intelligence のキューブにあるレベルの内部プロパティについて説明します。

"BI サンプルのアクセス方法" も参照してください。

# 内部プロパティ (MDX)

このセクションでは、キューブにあるレベルの内部プロパティをリストします。

## 内部プロパティ

すべてのメンバには、以下の内部プロパティがあります。このテーブルでは、サンプルの市町村 Juniper を使用した例を示します。

| プロパティ | 詳細 | 例 |
|---|---|---|
| KEY | "メンバ式" で説明したように、メンバを参照するときに構文 &[key] で使用するメンバのキー。詳細は、"キー値" を参照してください。 | `Juniper` |
| ID | メンバの内部 ID。 | `6` |
| NAME | メンバの表示名。これは、ソース値または `isName="true"` とマークされたプロパティの値 (ある場合) から取得されます。 | `Juniper` |
| MEMBER_NAME | | |
| CAPTION | | |
| CUBE_NAME | このメンバが属するキューブの論理名の大文字バージョン。これは、キューブの `name` 属性によって制御されます。 | `PATIENTS` |
| LEVEL_NUMBER | このメンバが属するレベルの番号。レベル 0 は All レベルです。 | `3` |
| LEVEL_CAPTION | 現在のレベルの `displayName` のローカライズされたバージョン。`displayName` 属性は、キューブ定義内で指定されます。 | `City` |
| LEVEL | レベルの論理名の大文字バージョン。これは、キューブ定義のレベルの `name` 属性によって制御されます。 | `CITY` |
| HIERARCHY_CAPTION | 現在の階層の `displayName` のローカライズされたバージョン。`displayName` 属性は、キューブ定義内で指定されます。 | `H1` |
| HIERARCHY | 階層の論理名の大文字バージョン。これは、キューブ定義の階層の `name` 属性によって制御されます。 | `H1` |
| DIMENSION_CAPTION | 現在のディメンジョンの `displayName` のローカライズされたバージョン。`displayName` 属性は、キューブ定義内で指定されます。 | `HomeD` |
| DIMENSION | ディメンジョンの論理名の大文字バージョン。これは、キューブ定義のディメンジョンの `name` 属性によって制御されます。 | `HOMED` |

プロパティの名前は、大文字と小文字が区別されません。

# キー値 (MDX)

このセクションでは、レベル・メンバの KEY 値がシステムで生成される方法について説明します。

## キー値

システムは、以下のように KEY 値を生成します。これらの値は、MDX の他のすべての項目と異なり、大文字と小文字が区別されることに注意してください。

| シナリオ | KEY 値の形式 | メンバ名の例* | 対応する KEY 値 |
|---|---|---|---|
| NULL メンバ | <null> | | <null> |
| All メンバ | * | All Patient Addresses | * |
| HourNumber 時間関数を使用するレベルのメンバ | 時間の数字を表す整数 | 2 | 2 |
| DayMonthYear 時間関数を使用するレベルのメンバ | $HOROLOG 形式で日付を表す整数 | Dec 27 2004 | 59896 |
| DayNumber 時間関数を使用するレベルのメンバ | 日付の数字を表す整数 | 21 | 21 |
| WeekNumber 時間関数を使用するレベルのメンバ | 日付の数字を表す整数 | 6 | 6 |
| MonthNumber 時間関数を使用するレベルのメンバ | 月の数字を表す整数 | July | 7 |
| WeekYear 時間関数を使用するレベルのメンバ | YYYYMWW 形式の文字列 | 2012W06 | 2012W06 |
| MonthYear 時間関数を使用するレベルのメンバ | YYYYMM 形式の整数 | July 2010 | 201007 |
| QuarterNumber 時間関数を使用するレベルのメンバ | 四半期の数字を表す整数 | Q3 | 3 |
| QuarterYear 時間関数を使用するレベルのメンバ | YYYYQQ 形式の整数 | Q4 2010 | 201004 |
| Year 時間関数を使用するレベルのメンバ | 年の数字を表す 4 桁の整数 | 2009 | 2009 |
| Decade 時間関数を使用するレベルのメンバ | 年の数字を表す 4 桁の整数に s を付加したもの | 1990s | 1990s |
| 計算ディメンジョンのメンバ | メンバ名、その後にコロン、その後に値を取得する SQL SELECT 文 | member 1 | このテーブルの後に表示*** |
| その他のシナリオ | メンバのソース値 (既定でメンバ名としても使用される)** | Juniper | Juniper |
| | | Sorenson, Violet | 169 |

*時刻ディメンジョンのレベルの場合、この列は既定の表示名を示します。レベルの定義方法に応じて、メンバ名が異なる可能性があります。"InterSystems Business Intelligence のモデルの定義" を参照してください。

**適切に定義されたキューブでは、各メンバ・キーは一意です。"メンバ・キーおよび名前の適切な定義"を参照してください。

***計算ディメンジョンのメンバのキーの例を以下に示します。

```
&[member 1:select ID from BI_Model_PatientsCube.Fact WHERE MxAge<50 AND DxHomeCity->PxName='Elm Heights']
```

PROPERTIES 関数を使用して、KEY プロパティなど、メンバのプロパティの値を検索できます。

# 時間レベルの NOW メンバ

このリファレンス・セクションでは、Business Intelligence の日付/時間レベルの NOW メンバについて説明します。

"BI サンプルのアクセス方法" も参照してください。

# NOW メンバ (MDX)

このセクションでは、日付/時間レベルの NOW メンバについて説明します。この構文は、インターシステムズによる MDX への拡張機能です。

## Basic 構文

```
date_time_level.[NOW]
```

以下は、この指定の説明です。

- date_time_level は、日付/時間ディメンジョンのレベルを参照するレベル式です。

この構文は、現在の日時に対応する指定されたレベルのメンバを返します。以下のテーブルに例を示します (2012 年 5 月 24 日作成)。

| 式 | 返されるメンバ |
|---|---|
| birthd.decade.NOW | birthd.decade.2010s |
| birthd.[quarter year].NOW | birthd.[quarter year].[Q2 2012] |
| birthqd.[quarter].NOW | birthd.[quarter year].Q2 |
| birthtd.NOW | birthtd.4pm |

NOW は大文字と小文字が区別されません。また、角括弧で囲まれている場合は、このトピックで後述するバリエーションを使用する場合を除いて、省略可能です。

## Now を基準にする日付

以下の形式の式も使用できます。

```
date_time_level.[NOW-integer]
```

または、以下のようにします。

```
date_time_level.[NOW+integer]
```

以下のテーブルに例を示します (2012 年 5 月 24 日作成)。

| 式 | 返されるメンバ |
|---|---|
| birthd.decade.[NOW-4] | birthd.decade.1970s |
| birthd.[quarter year].[NOW-4] | birthd.[quarter year].[Q2 2011] |
| birthqd.quarter.[NOW-4] | birthqd.quarter.Q2 |

カレンダー全体と無関係な時間レベルを使用する式を使用する場合は注意してください。その場合、[NOW-integer] は、サイクル内の過去のバケットを参照します。テーブルの 3 番目の例を見てください。birthqd.quarter レベルは、レコードを、年に関係なく、四半期でグループ化します。このレベルでは、NOW は現在の四半期番号のみを参照します。このレベルにはサイクルあたり 4 つのメンバが存在するので、このレベルでは [NOW-4] と [NOW] は同等です。

同じロジックは、[NOW+integer] という形式の式にも適用されます。

## 範囲式で使用する場合の制約

以下のいずれかの時間関数に基づくレベルの範囲式では、NOW を使用できません。

| 時間関数 | 一般的なメンバ | メモ |
|---|---|---|
| QuarterNumber | Q4 | これらのレベルは、年に依存しません。 |
| MonthNumber | November | |
| DayNumber | 24 | このレベルは、年および年の部分に依存しません。 |
| HourNumber | 1am | これらのレベルは、日に依存しません。 |
| MinuteNumber | 01:24 | |

## 日付レベルの他のオプション

DayMonthYear 時間関数に基づくレベルでは、以下のような年数、月数、および日数の組み合わせを使用した追加の式が Business Intelligence によってサポートされています。

```
daymonthyear_level.[NOW-offset_expression]
```

または、以下のようにします。

```
daymonthyear_level.[NOW+offset_expression]
```

単純な場合、offset_expression は、nnynnmnnd となります。それぞれ以下のとおりです。

- nn は、1 桁または 2 桁の整数です。
- nny は、年数を指定する省略可能なユニットです。
- nnm は、月数を指定する省略可能なユニットです。
- nnd は、日数を指定する省略可能なユニットです。

以下のテーブルに例を示します (2012 年 5 月 24 日作成)。

| 式 | 返されるメンバ |
|---|---|
| birthd.date.[NOW-4y3m2d] | birthd.date.[Feb 22 2008] |
| birthd.date.[NOW-1m] | birthd.date.[Apr 24 2012] |

既定では、各ユニットは加算されます。例えば、4y3m2d は、4 年と 3 か月と 2 日を合計した期間を意味します。

ユニット間にマイナス記号を使用して減算させることもできます。1y-1d は 1 年から 1 日を減算した期間を意味します。以下はその例です。

| 式 | 返されるメンバ |
|---|---|
| dateofsale.actual.daysold.[NOW-1y-1d] | dateofsale.actual.daysold.[May 25 2011] |

うるう年は自動的に考慮されます。月ごとの日数の違いも、内部ロジックで考慮されます。

# A

# 関数のクイック・リファレンス (MDX)

以下のテーブルは、Business Intelligence でサポートされている MDX 関数それぞれの構文および返りタイプをまとめたものです。

| 機能 | 構文 | 返りタイプ |
|---|---|---|
| %ALL | member_expression.%ALL | メンバ |
| %CELL | %CELL(relative_column_position, relative_row_position) | 数値または文字列 |
| %CELLZERO | %CELLZERO(relative_column_position, relative_row_position) | 数値または文字列 |
| %FIRST | %FIRST(set_expr, optional_numeric_expr) | 数値 |
| %KPI | %KPI(kpi_name, kpi_prop_name, kpi_series_name, parm, value, parm, value,...) | 数値 |
| %LABEL | %LABEL(MDX_expr, label, format_details, solve_order, cell_style, heading_style) | MDX_expr と同じ |
| %LAST | %LAST(set_expr, optional_numeric_expr) | 数値 |
| %LIST | %LIST(set_expr) | 文字列 (コンマで区切られたリスト) |
| %LOOKUP | %LOOKUP(termlist, key, field, default) | 数値または文字列 |
| %MDX | %MDX("MDX select query", parm, value, parm, value, parm, value,...) | 数値または文字列 |
| %NOT | member_expression.%NOT | メンバ |
| %OR | %OR(set_expr) | メンバ |
| %SEARCH | %SEARCH.&[comparison_expression] | メジャー検索式 |
| %SPACE | %SPACE() | 空のスペース |
| %TERMLIST | %TERMLIST(term_list_name, INCLUDE \| EXCLUDE) | セット |
| %TIMERANGE | %TIMERANGE(start_member, end_member, INCLUSIVE \| EXCLUSIVE) | メンバ |

| 機能 | 構文 | 返りタイプ |
|---|---|---|
| %TIMEWINDOW | %TIMEWINDOW(set_expr, start_member, optional_end_member) | メンバのセット |
| %TOPMEMBERS | level_expr.%TOPMEMBERS<br>hierarchy_expr.%TOPMEMBERS<br>dimension_expr.%TOPMEMBERS | メンバのセット |
| AGGREGATE | AGGREGATE(set_expr, optional_numeric_expr) | 数値 |
| ALLMEMBERS | level_expr.ALLMEMBERS<br>hierarchy_expr.ALLMEMBERS<br>dimension_expr.ALLMEMBERS | メンバのセット |
| ANCESTOR | ANCESTOR(member_expr, ancestor_level) | メンバ |
| AVG | AVG(set_expr, optional_numeric_expr) | 数値 |
| BOTTOMCOUNT | BOTTOMCOUNT(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| BOTTOMPERCENT | BOTTOMPERCENT(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| BOTTOMSUM | BOTTOMSUM(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| CHILDREN | member_expr.CHILDREN | メンバのセット |
| CLOSINGPERIOD | CLOSINGPERIOD(ancestor_level, member_expr) | メンバ |
| COUNT | COUNT(set_expr)<br>COUNT(set_expr, EXCLUDEEMPTY) | 数値 |
| COUSIN | COUSIN(member_expr, higher_member_expr) | メンバ |
| CROSSJOIN | CROSSJOIN(set_expr1, set_expr2)<br>NON EMPTY CROSSJOIN(set_expr1, set_expr2) | タプルのセット |
| CURRENTMEMBER | hierarchy_expr.CURRENTMEMBER<br>dimension_expr.CURRENTMEMBER | メンバ |
| DESCENDANTS | DESCENDANTS(member_expression, level_expression, OPTIONAL_FLAG)<br>DESCENDANTS(member_expression, level_offset, OPTIONAL_FLAG) | メンバのセット |
| DISTINCT | DISTINCT(set_expr) | セット |
| EXCEPT | EXCEPT(set_expr1, set_expr2, ALL)<br>EXCEPT(set_expr1,set_expr2) | セット |
| FILTER | FILTER(set_expr, logical_expr) | セット |
| FIRSTCHILD | member_expr.FIRSTCHILD | メンバ |

| 機能 | 構文 | 返りタイプ |
|---|---|---|
| FIRSTSIBLING | member_expr.FIRSTSIBLING | メンバ |
| HEAD | HEAD(set_expr, optional_integer_expr, optional_sample_flag) | セット |
| HIERARCHIZE、HIERARCHISE | HIERARCHIZE(set_expr)<br>HIERARCHIZE(set_expr, POST) | メンバのセット |
| IIF | IIF(logical_expr, expression1, expression2) | 数値または文字列 |
| INTERSECT | INTERSECT(set_expr1, set_expr2) | セット |
| ISNULL | ISNULL(scalar_expression,scalar_value_if_null) | 数値または文字列 |
| LAG | member_expr.LAG(optional_nonnegative_integer_expr) | メンバ |
| LASTCHILD | member_expr.LASTCHILD | メンバ |
| LASTSIBLING | member_expr.LASTSIBLING | メンバ |
| LEAD | member_expr.LEAD(optional_nonnegative_integer_expr) | メンバ |
| LOG | LOG(numeric_expr) | 数値 |
| LOOKUP | LOOKUP(term_list_name, lookup_value, default, alternative_field) | 文字列 |
| MAX | MAX(set_expr, optional_numeric_expr) | 数値 |
| MEDIAN | MEDIAN(set_expr, optional_numeric_expr) | 数値 |
| MEMBERS | level_expr.MEMBERS<br>hierarchy_expr.MEMBERS<br>dimension_expr.MEMBERS | メンバのセット |
| MIN | MIN(set_expr, optional_numeric_expr) | 数値 |
| NEXTMEMBER | member_expr.NEXTMEMBER | メンバ |
| NONEMPTYCROSSJOIN | NONEMPTYCROSSJOIN(set_expr1, set_expr2) | タプルのセット |
| OPENINGPERIOD | OPENINGPERIOD(ancestor_level, member_expr) | メンバ |
| ORDER | ORDER(set_expr, ordering_expr, ASC \| DESC \| BASC \| BDESC)<br>ORDER(set_expr, ordering_expr) | セット |
| PARALLELPERIOD | PARALLELPERIOD(level_expr, offset, member_expr) | メンバ |
| PARENT | member_expr.PARENT | メンバ |
| PERCENTILE | PERCENTILE(set_expr, numeric_expr, numeric_expr, optional_percentile_value) | 数値 |
| PERCENTILERANK | PERCENTILERANK(set_expr, numeric_expr, comparison_value) | 数値 |
| PERIODSTODATE | PERIODSTODATE(ancestor_level, member_expr) | メンバのセット |
| POWER | POWER(numeric_expr,numeric_expr_for_power) | 数値 |
| PREVMEMBER | member_expr.PREVMEMBER | メンバ |

| 機能 | 構文 | 返りタイプ |
|---|---|---|
| PROPERTIES | member_expr.PROPERTIES(property_name) | 文字列 |
| RANK | RANK(tuple_expr, set_expr, optional_numeric_expr) | 数値 |
| ROUND | ROUND(numeric_expr,decimal_places) | 数値 |
| SIBLINGS | member_expr.SIBLINGS | メンバのセット |
| SQRT | SQRT(numeric_expr) | 数値 |
| STDDEV、STDEV | STDDEV(set_expr, optional_numeric_expr) | 数値 |
| STDDEVP、STDEVP | STDDEVP(set_expr, optional_numeric_expr) | 数値 |
| SUBSET | SUBSET(set_expr, first_element_expr, optional_element_count) | セット |
| SUM | SUM(set_expr, optional_numeric_expr) | 数値 |
| TAIL | TAIL(set_expr, optional_integer_expr) | セット |
| TOPCOUNT | TOPCOUNT(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| TOPPERCENT | TOPPERCENT(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| TOPSUM | TOPSUM(set_expr, element_count, optional_ordering_expr) | メンバまたはタプルのセット |
| UNION | UNION(set_expr1,set_expr2)<br>UNION(set_expr1,set_expr2, ALL) | セット |
| VAR、VARIANCE | VAR(set_expr, optional_numeric_expr) | 数値 |
| VARP、VARIANCEP | VARP(set_expr, optional_numeric_expr) | 数値 |
| VISUALTOTALS | VISUALTOTALS(set_expr, optional_parent_name_pattern) | メンバのセット |