InterSystems MDX の使用法

Version 2024.1
2024-06-03

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

InterSystems MDX の使用法
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

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

目次

1

Business Intelligence および MDX

ここでは、多くのベンダによって実装されているクエリ言語である MDX（多次元式）を InterSystems IRIS® データ・プラットフォーム Business Intelligence がどのようにサポートしているかについて説明します。

Business Intelligence のシステム要件に関する詳細は、"インターシステムズのサポート対象プラットフォーム" を参照してください。

1.1 Business Intelligence の目的

Business Intelligence の目的は、アプリケーションへのビジネス・インテリジェンス（BI）の埋め込みを可能にすることで、ユーザがデータに関して高度な質問と回答を行えるようにすることです。アプリケーションにはダッシュボードが含まれます。ダッシュボードにはピボット・テーブルを含めることができます。

ピボット・テーブルは、対話型で、ドリル可能なデータの表示で、特定のユーザ・ロール、またはユーザ・インタフェースの特定領域向けに設計されています。

各ピボット・テーブルには、実行時に実行される、基礎となる MDX クエリが含まれます。システムは、トランザクション・テーブルに対して直接クエリを実行する代わりに、トランザクション・テーブルと同期化されているキューブに対してクエリを実行します（キューブ定義の詳細は、"InterSystems Business Intelligence のモデルの定義" を参照してください）。

1.2 ピボット・テーブルの概要

ピボット・テーブルは、Business Intelligence の基幹であり、データを選択および集約し、そのデータを対話型形式で表示します。

ピボット・テーブルの例を以下の図に示します。この図は、年齢および男女別に患者数と患者 1 人あたりが抱えるアレルギー数の平均値を表しています。

		Female		Male	
		Patient Count	Avg Allergy Count	Patient Count	Avg Allergy Count
0 to 29	0 to 9	674	0.76	740	0.76
	10 to 19	691	0.78	769	0.79
	20 to 29	680	0.75	704	0.81
30 to 59	30 to 39	772	0.79	741	0.84
	40 to 49	710	0.79	777	0.77
	50 to 59	573	0.85	568	0.82
60+	60 to 69	361	0.83	341	0.76
	70 to 79	336	0.81	236	0.79
	80+	211	0.77	116	0.79

概念は相互に関係しているため、他の概念を参照せずに個々の概念を検討することは難しいので、まず、予備的な定義から始めましょう。

- レベルは、レコードのグループ化を可能にします。レベルにはメンバがあります。同様に、個々のメンバは、ソース・データ内の特定のレコード・グループに対応します。

  例えば、`Age Group` レベルには、メンバ `0 to 29`、`30 to 59`、および `60+` があります。`Age Bucket` レベルには、メンバ `0-9`、`10-19`、`20-29` などがあります。`Gender` レベルには、メンバ `Female` および `Male` があります。

- メジャーはピボット・テーブルの本文に表示される値であり、選択されたレコードのソース・データの値に基づいています。指定されたコンテキストに対して、メジャーは該当するすべてのソース・レコード値を集約し、これらを 1 つの値として表します。

  例えば、メジャー `Patient Count` は患者数、メジャー `Avg Allergy Count` は患者 1 人あたりのアレルギー数の平均値です。

1.3 MDX の概要

MDX は、OLAP (オンライン分析処理) データベースの標準クエリ言語です。MDX 言語は、キューブ要素を参照する構文を提供します。この言語のほとんどの文や関数によって、キューブに対してクエリを実行することができます。返されるデータは結果セットで、ピボット・テーブルとして表示できます。

MDX には、キューブ定義を拡張する機能も用意されています。特に、既存の要素に基づいて新しい要素を定義し、それらの新しい要素を MDX クエリで使用できます。

Business Intelligence では、MDX は以下のようにサポートされます。

- アナライザでピボット・テーブルを作成する際には、システムによって MDX クエリが生成されて使用されます。MDX クエリは直接表示できます。
- アナライザには、MDX クエリを直接実行するためのオプションが用意されています。
- MDX クエリを MDX シェルで実行し、クエリ結果を表示することができます。
- システムには、MDX クエリの実行に使用できる API が用意されています。
- 以下のサブセクションで説明するように、Business Intelligence モデル内で、MDX 式およびクエリを使用して特定の要素を定義します。

一部の MDX クエリは複雑すぎて、現在のユーザ・インタフェース内では作成できません。そのようなクエリは、シェルで実行するか API を使用して実行できますが、アナライザでのドラッグ・アンド・ドロップ操作で作成することはできません。

詳細は、以下のソースを参照してください。

- アナライザの詳細は、"アナライザの使用法" を参照してください。
- MDX シェルの詳細は、"InterSystems Business Intelligence の概要" を参照してください。
- MDX API の詳細は、"InterSystems Business Intelligence の実装" を参照してください。

注釈　MDX の実装は、システムによって提供されます。結果は、他の実装とは異なる場合があります。

1.3.1 Business Intelligence モデルでの MDX

Business Intelligence モデルでは、MDX 式およびクエリを使用できる場所は以下のとおりです。

- キューブ定義内で以下のように使用します。
  - MDX メンバ式を使用して、計算メンバを定義します。
  - MDX セット式を使用して、名前付きセットを定義します。
  - MDX セット式を使用して、キューブをフィルタ処理します。

  これらはすべてオプションです。
- サブジェクト領域の定義内で、MDX セット式を使用して、サブジェクト領域をフィルタ処理します。これはオプションです。サブジェクト領域にフィルタを含める必要はありません。
- KPI (重要業績評価指標) 定義内で、MDX クエリを使用して KPI を定義できます。これはオプションです。代わりに SQL クエリを使用することもできます。

詳細は、"InterSystems Business Intelligence のモデルの定義"および "InterSystems Business Intelligence の実装" を参照してください。

2

基本的な MDX クエリ

ここでは、Business Intelligence の基本的な MDX クエリ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。

2.1 DemoMDX キューブの内容

馴染みのないデータベースで SQL クエリを作成する場合は、テーブルやその列をよく知ることから始めます。同様に、MDX クエリを作成する場合は、使用可能なキューブやその内容をよく知ることから始めます。

1. ターミナルを開始します。
2. サンプルをロードしたネームスペースに切り替えます。
3. MDX シェルにアクセスするには、以下のコマンドを入力します。

   ObjectScript

   ```
   Do ##class(%DeepSee.Utils).%Shell()
   ```

4. 使用可能なキューブを確認するには、以下のコマンドを入力します (大文字と小文字は区別されません)。

   ```
   CUBE
   ```

   すると、ターミナルにキューブのリストが表示されます。
5. キューブの使用可能な内容を確認するには、以下のコマンドを入力します。

   ```
   CUBE cubename
   ```

   例えば、以下のようにします。

   ```
   CUBE demomdx
   ```

   シェルでは、コマンドおよびキューブ名の大文字と小文字は区別されません。

   ターミナルには以下のように表示されます。

   ```
   Measures
     %COUNT
     Age
     Avg Age
     Allergy Count
     Avg Allergy Count
     Test Score
   ```

```
  Avg Test Score
AgeD
    All Patients
  H1
    All Patients
    Age Group
    Age Bucket
AllerD
  H1
    Allergies
BirthD
  H1
    Year
    Quarter Year
BirthQD
  H1
    Quarter
DiagD
  H1
    Diagnoses
GenD
  H1
    Gender
ColorD
  H1
    Favorite Color
HomeD
  H1
    ZIP
    City
DocD
  H1
    Doctor
```

DemoMDX キューブは患者を表しています。このキューブの内容は以下のとおりです。

- Measures セクションには、%COUNT、Age、Ave Age、Allergy Count などの使用可能なメジャーがリストされています。これらのメジャーは、患者と関連付けられており、患者全体で集約されます。
- Dimensions セクションには、ディメンジョンが含まれます。このキューブには、AgeD や AllerD などのディメンジョンが含まれています。

  ここでは、ディメンジョンは 1 つ以上の階層のコンテナだと説明しておきますが、詳細は "ディメンジョンおよび階層を使用した作業" を参照してください。
- ディメンジョン内の最初の要素は階層です。規約により、このサンプルでは、各ディメンジョンに H1 という名前の階層が 1 つずつ含まれています。

  ここでは、階層は 1 つ以上のレベルのコンテナだと説明しておきますが、詳細は "ディメンジョンおよび階層を使用した作業" を参照してください。
- 階層内の要素はレベルです。このキューブには、Age、Age Group、Gender、ZIP、City などのレベルが含まれています。これらのレベルによって、患者のさまざまなグループの選択が可能になります。

多くの MDX アプリケーションでは、ディメンジョン、ディメンジョン内の階層、およびその階層内のレベルに同じ名前が使用されます。この方法は、MDX の学習者に混乱を招く可能性があるため、このサンプル・キューブでは、以下の任意の命名規約を使用します。

- ディメンジョン名は短くし、末尾の文字を D にする。
- 各ディメンジョンに H1 という名前の階層を 1 つ含める。
- レベル名はユーザがわかりやすいものとする (アナライザでは、ディメンジョン名とレベル名の両方が表示されますが、主な作業対象となるのはレベルです)。

後述するように、InterSystems MDX では、識別子の一部を省略できます。このサンプルの命名規約では、どの部分が省略可能かを明確に示します。

注釈　キューブには、計算メンバや名前付きセットも格納できます。これらの要素は、MDX シェルの CUBE コマンドによって表示されることはありませんが、MDX シェルやその他の場所で使用することはできます。

2.2 最も単純なクエリ

MDX シェルで、以下の MDX クエリを入力します (大文字と小文字は区別されません)。

```
SELECT FROM demomdx
```

シェルによって以下のような結果が表示されます。

```
Result:              1,000
```

このクエリは、患者をカウントするだけです。

MDX では、大文字と小文字は区別されませんが、メンバのキーは例外です。これについては、"レベルを使用した作業" で説明します。

2.3 メンバ

MDX クエリの主要な構成要素にメンバがあります。各レベルには、1 つ以上のメンバが含まれます。例えば、City レベルには、複数のメンバ (データ内の各市町村に 1 つずつ) が含まれます。レベルによって、レコードの選択が可能になります。具体的には、レベルの各メンバを使用して、レコードのサブセットにアクセスすることができます。

DemoMDX キューブでは、このキューブ内の各レベルの各メンバで、患者のグループを選択できます。

このセクションでは、以下の手順で単純なクエリを実行して、DemoMDX キューブ内のレベルのメンバを確認します。

1. MDX シェルで、以下の MDX クエリを入力します (大文字と小文字は区別されません)。

   ```
   SELECT homed.h1.city.MEMBERS ON ROWS FROM demomdx
   ```

   シェルによって、City レベルのメンバが以下のように表示されます。

   ```
   1 Cedar Falls                          110
   2 Centerville                           99
   3 Cypress                              112
   4 Elm Heights                          118
   5 Juniper                              122
   6 Magnolia                             114
   7 Pine                                 121
   8 Redwood                              111
   9 Spruce                                93
   ```

   ここでは、2 番目の列に表示されているメンバ名についてのみ説明します。

City レベルには、Cedar Falls、Centerville、Cypress などのメンバが含まれています。このレベルの各メンバは、その出身地の患者セットを表します。例えば、Centerville メンバは、出身地が Centerville の患者すべてを表します。

2.4 メジャー

MDX クエリの別の主要な構成要素にメジャーがあります。すべての InterSystems IRIS Business Intelligence クエリでは、少なくとも 1 つのメジャーが使用されます。メジャーを指定しない場合は、キューブに定義されている既定のメジャーが使用されます。ほとんどのキューブでは、既定のメジャーはレコードの数を示す %COUNT です。サンプル・キューブ内のメジャーをいくつか見てみましょう。

1. MDX シェルで、以下のサンプル・クエリを入力します。

   ```
   SELECT MEASURES.[%COUNT] ON COLUMNS FROM demomdx
   ```

   このクエリによって、1 列のデータが含まれた結果セットが返されます。この結果セットは、キューブが表すデータ・セット全体にわたる %COUNT メジャーの集約値です。サンプルのデータによって異なりますが、シェルには以下のように表示されます。

   ```
   %COUNT
     1,000
   ```

   この例では、1,000 人の患者が存在します。

2. MDX シェルで、以下のクエリを入力します。

   ```
   SELECT MEASURES.[avg test score] ON COLUMNS FROM demomdx
   ```

   このクエリによって、データ・セット全体にわたる Avg Test Score メジャーの集約値を示す結果セットが返されます。

   サンプルのデータによって異なりますが、シェルには以下のように表示されます。

   ```
   Avg Test Score
            74.75
   ```

   この数字は、患者全員の平均テスト・スコアです。

2.5 メンバとメジャーの参照

これまでのセクションで、DemoMDX キューブの要素（特にメジャーとレベル）について見てきましたので、キューブに含まれるデータについての概要がつかめてきたことでしょう。また、単純な MDX クエリも記述しました。次の手順では、メンバやメジャーの参照に使用する構文を学習します。

- メンバを参照するための構文は以下のとおりです。

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

- あるレベルのメンバをすべて参照するための構文は以下のとおりです。

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

  MEMBERS は、レベルのメンバを返す MDX 関数です。"InterSystems MDX リファレンス" は、Business Intelligence でサポートされているすべての MDX 関数のリファレンス情報を提供しています。

- メジャーを参照するための構文は以下のとおりです。

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

以下のバリエーションに注意してください。

- これらの名前のいずれでも、名前を構成するのが英数字のみの場合は角括弧 ([]) を省略できます。

  識別子の詳細は、"InterSystems MDX リファレンス" の "識別子" のセクションを参照してください。

- レベルまたはメンバを参照する場合は、階層名を省略できます。省略すると、MDX 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 (このバリエーションは、インターシステムズによる MDX への拡張機能の 1 つです)。

- メンバを参照する場合は、レベル名を省略できます。省略すると、MDX 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 (このバリエーションは、インターシステムズによる MDX への拡張機能の 1 つです)。

ディメンジョン名は省略できません。

InterSystems MDX では、以下の例はすべて同等です。

```
[GenD].[H1].[GENDER].Female
[GenD].Female
GenD.H1.GENDER.Female
GenD.H1.Female
GenD.Female
```

2.6 %COUNT を使用した単純な MDX クエリ

ここでは、メジャーを参照せず、キューブに定義されている既定のメジャー (通常、%COUNT) を使用する単純な形式の MDX クエリを示します。

- 特定のレベルのメンバを列として使用するには、以下の形式のクエリを使用します。

  ```
  SELECT [dim_name].[hier_name].[lev_name].MEMBERS ON COLUMNS FROM cubename
  ```

- あるレベルのメンバを列として使用し、別のレベルのメンバを行として使用するには、以下の形式のクエリを使用します。

  ```
  SELECT [dim_name].[hier_name].[lev_name].MEMBERS ON COLUMNS,
  [dim_name].[hier_name].[lev_name].MEMBERS ON ROWS FROM cubename
  ```

  注釈　ここに示されるような改行を含めないでください。このドキュメント (特に紙版のドキュメント) では、上記の改行は見やすくするためにのみ使用しています。MDX シェルでは、この改行は許可されていません。

- 単一メンバを列として使用するには、以下の形式のクエリを使用します。

  ```
  SELECT [dim_name].[hier_name].[lev_name].[member_name] ON COLUMNS FROM cubename
  ```

COLUMNS の代わりに 0 を使用し、ROWS の代わりに 1 を使用することもできます (スペースの都合で、このドキュメントでは、COLUMNS と ROWS ではなく、0 と 1 を使用します)。

いずれの場合も、SELECT 文によって結果セットが返されます。その結果セットは、MDX シェルによってテーブル形式で表示されます。

これらのバリエーションを使用するクエリを試してみましょう。

1. 以下の MDX クエリを入力します。

   ```
   SELECT gend.h1.gender.MEMBERS ON 0 FROM demomdx
   ```

シェルによってクエリが実行され、以下のような結果が表示されます (実際の結果は若干異なります)。

```
                                Female                  Male
                                   488                   512
```

これから以下の点がわかります。

- このクエリではメジャーを指定しなかったため、表示されている数は %COUNT の値、つまり患者数です。
- 2 つのメンバが列として (または、結果セットの列軸に) 表示されています。Female メンバは女性患者、Male メンバは男性患者を指しています。

2. 同じクエリの短いバージョンを試してみましょう。

```
SELECT gend.gender.MEMBERS ON 0 FROM demomdx
```

このクエリでは、先ほどのクエリと同じデータが返されます。

3. ここで、以下のバリエーションを入力します。

```
SELECT gend.gender.female ON 0 FROM demomdx
```

この結果は以下のようになります。

```
                                  Female
                                     488
```

この例では、クエリによって、このディメンジョンの両方のメンバではなく特定のメンバが選択されました。

4. 以下のバージョンを試してみましょう (メンバ名の大文字と小文字を変更)。

```
SELECT gend.gender.FEMALE ON 0 FROM demomdx
```

これによって、先ほどのクエリと同じ結果が返されます。

5. もう少し複雑なクエリを入力してみます。

```
SELECT gend.h1.gender.MEMBERS ON 0,homed.h1.zip.MEMBERS ON 1 FROM demomdx
```

シェルによってクエリが実行され、以下のような結果が表示されます。

```
                                Female                  Male
1 32006                            105                   110
2 32007                             58                    53
3 34577                            173                   174
4 36711                             41                    58
5 38928                            111                   117
```

この場合は、結果に複数の行 (患者の郵便番号ごとに 1 行ずつ) があります。郵便番号ごとに、男女別の数が表示されています。

結果に行が複数ある場合、MDX シェルによって、結果の行番号を示す列が表示されます。

2.6.1 軸の省略

MDX の他の実装では、より数字の大きな軸を使用する場合、軸を省略することはできません。つまり、COLUMNS も使用しない限り、ROWS は使用できません。

ただし、InterSystems MDX では、COLUMNS を省略した場合、以下のように %COUNT が使用されます。

```
SELECT gend.h1.gender.MEMBERS ON ROWS FROM demomdx

1 Female                                  488
2 Male                                    512
```

2.7 セット

MDX では、列と行はクエリと結果セットの軸です。例えば、以下の結果セットでは、列軸に性別、行軸に自宅の郵便番号があります。

```
                                Female                Male
1 32006                             105                 110
2 32007                              58                  53
3 34577                             173                 174
4 36711                              41                  58
5 38928                             111                 117
```

軸はセットを使用します。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。また、セットは空でもかまいませんが、その場合はクエリ軸では使用できません。

セット内では、それぞれの式は以下のいずれかにできます。

- member expression。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。以下はその例です。

    ```
    [PatDim].[GENDERH1].[GENDER].[F]
    ```

  - 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。

    ```
    [PatDim].[GENDERH1].[GENDER].[F].NEXTMEMBER
    ```

    (NEXTMEMBER は、レベルの次のメンバを返す MDX 関数です。"レベルを使用した作業" では、この関数をはじめとした関数を紹介します。)

- セットを返す MDX 関数 (MEMBERS など) を使用する式。以下はその例です。

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

これ以外にも、他の形式の式や他の種類のセット要素があります。"セットを使用した作業" および "InterSystems MDX リファレンス" を参照してください。

SELECT 文では、NULL 以外の任意のセット式を使用することができます。一般に SELECT では、1 つの軸を使用するクエリの基本構文は以下のようになります。

```
SELECT set_expression ON COLUMNS FROM cubename
```

または

```
SELECT set_expression ON 0 FROM cubename
```

以下の形式は、2 つの軸を使用するクエリです。

```
SELECT set_expression ON COLUMNS,set_expression ON ROWS FROM cubename
```

または

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

SELECT 文で追加の軸を使用できますが、シェルでは、結果は読み取れる形式で表示されません。

2.7.1 例

ここで、前のセクションで示したように、異なる種類のセットを使用するクエリのバリエーションをいくつか試してみましょう。

1. 以下の例では、コンマ区切りのリストによって作成されたセットを使用しています。

```
SELECT {gend.h1.gender.MEMBERS,homed.h1.city.MEMBERS} ON 0 FROM demomdx
```

```
                  Female      Male   Cedar F   Centerv   Cypress   Elm Hei ...
                     488       512       110        99       112       118...
```

   ご覧のように、結果には列が多すぎて全体は表示されていません。

2. では、同じセットを列ではなく行として使用するバリエーションを試してみましょう。

```
SELECT {gend.h1.gender.MEMBERS,homed.h1.city.MEMBERS} ON 1 FROM demomdx
```

```
 1 Female                              488
 2 Male                                512
 3 Cedar Falls                         110
 4 Centerville                          99
 5 Cypress                             112
 6 Elm Heights                         118
 7 Juniper                             122
 8 Magnolia                            114
 9 Pine                                121
10 Redwood                             111
11 Spruce                               93
```

3. 次に、上記の例を拡張します。性別を列に移動し、自宅の郵便番号を別の行セットとして追加してみましょう。

```
SELECT gend.h1.gender.MEMBERS ON 0,{homed.h1.city.MEMBERS,homed.h1.zip.MEMBERS} ON 1 FROM demomdx
```

```
                               Female                Male
 1 Cedar Falls                      58                  52
 2 Centerville                      41                  58
 3 Cypress                          51                  61
 4 Elm Heights                      53                  65
 5 Juniper                          58                  64
 6 Magnolia                         58                  56
 7 Pine                             64                  57
 8 Redwood                          58                  53
 9 Spruce                           47                  46
10 32006                           105                 110
11 32007                            58                  53
12 34577                           173                 174
13 36711                            41                  58
14 38928                           111                 117
```

4. 1 つのメンバをセット内で複数回使用してみます。

```
SELECT gend.h1.gender.MEMBERS ON 0,{homed.h1.[36711],homed.h1.[36711]} ON 1 FROM demomdx
```

```
                               Female                Male
1 36711                            41                  58
2 36711                            41                  58
```

2.8 メジャーの表示

どのような MDX クエリでも、1 つ以上のメジャーを使用します。使用するメジャーを指定しない場合は、キューブに定義されている既定のメジャー (通常、レコードの数を示す %COUNT) が使用されます。他のメジャーを表示する方法は複数あります。ここでは、その一部を紹介します。

クエリでメジャーを使用するには、以下の操作を実行します。

- メジャーを列として表示できます。また、必要に応じてセットを行として表示できます。以下はその例です。

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

	Avg Allergy Count
1 None	1.08
2 Blue	1
3 Green	1.05
4 Orange	1.16
5 Purple	1.22
6 Red	1.06
7 Yellow	0.94

- メジャーを行として表示できます。また、必要に応じてセットを列として表示できます。これは、前の例とは逆です。以下はその例です。

```
SELECT gend.h1.gender.MEMBERS ON 0, MEASURES.[avg test score] ON 1 FROM demomdx
```

	Female	Male
Avg Test Score	73.49	74.42

- 複数のメジャーから成るセットを作成し、そのセットを行または列として使用できます。以下はその例です。

```
SELECT {MEASURES.[%COUNT],MEASURES.[avg test score]} ON 0,colord.MEMBERS ON 1 FROM demomdx
```

	%COUNT	Avg Test Score
1 None	239	72.68
2 Blue	124	76.94
3 Green	106	72
4 Orange	148	72.89
5 Purple	135	74.87
6 Red	121	74.92
7 Yellow	127	74.41

- 以下のように、CROSSJOIN 関数を使用できます。

```
SELECT CROSSJOIN(MEASURES.[%COUNT],gend.h1.gender.MEMBERS) ON 0, diagd.h1.diagnoses.MEMBERS ON 1
FROM demomdx
```

	Female	Male
1 None	399	429
2 asthma	46	44
3 CHD	14	23
4 diabetes	23	22
5 osteoporosis	21	1

(この関数のより一般的な情報は、"セットの組み合わせ" を参照してください。)

- 以下のように、MEMBERS 関数を使用してメジャーをすべて表示できます (%COUNT を除く)。

```
SELECT gend.MEMBERS ON 0, MEASURES.MEMBERS ON 1 FROM demomdx
```

	Female	Male
1 Age	18,413	17,491
2 Avg Age	37.73	34.16
3 Allergy Count	326	332
4 Avg Allergy Count	1.08	1.07
5 Test Score	29,542	31,108
6 Avg Test Score	73.49	74.42

2.9 クエリへの単純なフィルタの組み込み

MDX クエリには、フィルタを含めることもできます。フィルタを使用すると、クエリが使用する可能性のあるファクト・テーブルの行の数を減らせます。クエリにフィルタを追加するには、SELECT 文の末尾に以下のような節を追加します。

```
WHERE filter_details
```

filter_details の最も簡単な形式は以下のとおりです。

```
[dim_name].[hier_name].[level_name].[member_name]
```

ここでは、このページで前述した ″メンバとメジャーの参照″ と同じバリエーションを使用できます。

この式ではクエリがフィルタ処理されるため、システムはこのメンバに関連付けられたレコードにのみアクセスします。例えば、以下のクエリでは、骨粗しょう症の患者のみを使用しています。

```
SELECT MEASURES.[%COUNT] ON 0,aged.[age bucket].MEMBERS ON 1 FROM demomdx WHERE diagd.osteoporosis
```

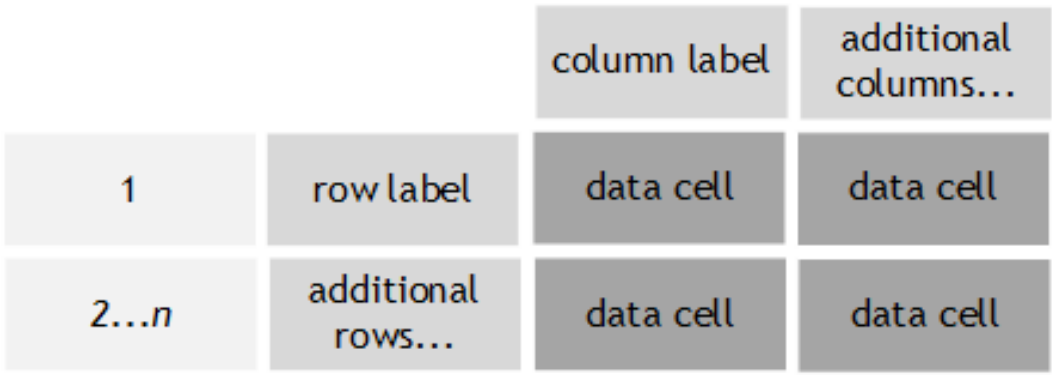
```
                                    %COUNT
1 0 to 9                                 *
2 10 to 19                               *
3 20 to 29                               *
4 30 to 39                               *
5 40 to 49                               *
6 50 to 59                               *
7 60 to 69                               7
8 70 to 79                               7
9 80+                                    8
```

MDX シェルでは、値が NULL であることを示すためにアスタリスク (*) が使用されます。

″クエリのフィルタ処理″ では、WHERE について詳しく説明します。

2.10 MDX 結果の内容の理解

ここまで、さまざまな MDX クエリとその結果を確認してきました。ここでは、それらの結果をより形式的に検討してみましょう。MDX シェルでは、MDX クエリの結果が、以下の一般的な形式で示されます。



結果は以下の規則によって決まります。

- 出力に結果の行が複数含まれる場合は、参照しやすいように、最初の列には行番号 (nnn (最初の行が 1)) が表示されます。結果の行が 1 つのみの場合は、この列は含まれません。

  この番号列は、MDX シェルにのみ表示され、結果セットには含まれません。

- 出力には、列軸で使用するセットのメンバごとに、データ・セル列が 1 つ含まれます。
- 一般に、それぞれの列ラベルは対応するメンバの名前と一致します。このメンバはメジャーである場合も、通常のメンバである場合もあります。
- 行にセットを指定しない場合、出力にはラベルのない行が 1 つ含まれます。
- 行にセットを指定した場合、出力にはそのセットのメンバごとに行が 1 つ含まれます。指定された行のラベルは、対応するメンバの名前です。このメンバはメジャーである場合も、通常のメンバ (つまり、メジャーでないメンバ) である場合もあります。
- 指定されたデータ・セルには、出力に値またはアスタリスク (*) のいずれかが表示されます。アスタリスクは、その値が NULL であることを示します。

  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。
  - 両方のメンバがメジャーである場合は、列軸のメジャーが使用されます。

    (両方のメンバが計算メジャーである場合は、SOLVE_ORDER も考慮されることに注意してください。詳細は、"InterSystems MDX リファレンス" の "SOLVE_ORDER 節" を参照してください。)

2.10.1 クエリの軸の独立性に関するメモ

システムは、各クエリ軸を他から独立して考慮します。場合によっては、結果が直観的でないこともあります。ここでは、2 つの例を挙げます。

2.10.1.1 クエリ内の他のセットから影響を受けないセット順序

どのような場合でも、返されるセットの順序は、クエリ内で使用される他のあらゆるセットから独立しており、結果が直観的でない場合があることを覚えておくことが重要です。例えば、以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0,
TOPCOUNT(homed.city.MEMBERS,100,MEASURES.[%COUNT]) ON 1 FROM demomdx
```

```
                                    %COUNT
1 Juniper                              122
2 Pine                                 121
3 Elm Heights                          118
4 Magnolia                             114
5 Cypress                              112
6 Redwood                              111
7 Cedar Falls                          110
8 Centerville                           99
9 Spruce                                93
```

このクエリは、市町村を患者数に基づいて並べ替えた場合に取得する並べ替え順序を示しています (この例では、選択するメンバの数が 100 に指定されています。これは、実際のメンバの数よりも多いため、すべてのメンバが表示されています)。

上記のクエリを、上位 3 つのメンバが返されるように変更すると、以下のようになります。

```
SELECT MEASURES.[%COUNT] ON 0, TOPCOUNT(homed.city.MEMBERS,3,MEASURES.[%COUNT]) ON 1 FROM demomdx
```

```
                                  %COUNT
1 Juniper                             122
2 Pine                                121
3 Elm Heights                         118
```

ここで、患者数を性別に基づいて分類した場合の結果について考えてみます。

```
SELECT CROSSJOIN(MEASURES.[%COUNT],gend.gender.MEMBERS) ON 0,
TOPCOUNT(homed.city.MEMBERS,3,Measures.[%COUNT]) ON 1 FROM demomdx
```

```
                             Female                Male
1 Juniper                         58                   64
2 Pine                            64                   57
3 Elm Heights                     53                   65
```

このクエリでリストされている市町村の順序は、列の分類を指定しなかった前のクエリの順序と同じです。この例では、Juniper が患者数の合計に基づく最上位の市町村であるため、最初に表示されています。つまり、並べ替えは、表示される値のいずれでもなく、市町村の患者数の合計で制御されています。

2.10.1.2 クエリ内の他のセットから影響を受けないセット・メンバシップ

返されるセットのメンバは、クエリ内で使用されるあらゆるセットから独立しており、結果が直観的でない場合があることを覚えておくことも重要です。例えば、以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, TAIL(birthd.year.MEMBERS,10) ON 1 FROM demomdx
```

```
                                  %COUNT
 1 1912                                3
 2 1918                                1
 3 1919                                1
 4 1920                                4
 5 1921                                2
 6 1922                                1
 7 1923                                2
 8 1924                                1
 9 1925                                2
10 1927                                5
```

ここで、1 つの性別のみを表示した場合の結果について考えてみます。

```
SELECT CROSSJOIN(gend.male,MEASURES.[%COUNT]) ON 0, HEAD(birthd.year.MEMBERS,10) ON 1 FROM demomdx
```

```
                                  %COUNT
 1 1912                               *
 2 1918                               *
 3 1919                               *
 4 1920                               1
 5 1921                               *
 6 1922                               *
 7 1923                               2
 8 1924                               *
 9 1925                               1
10 1927                               1
```

誕生年は、性別全体で集約されたデータを示す前のクエリの場合と同じです。

2.11 Business Intelligence の名前解決

場合によっては、同じタイプの複数のエンティティが同じ名前を持つことも可能です。例えば、MDX キューブで、階層が異なれば (あるいはディメンジョンが異なれば)、同じ名前で 2 つのレベルを設定できます。キューブ・コマンドで、キューブの内容が以下のように示されたとします。

```
...
Dimensions
  Geography
    ShipToHierarchy
      State
      City
    OrderByHierarchy
      State
      City
```

Business Intelligence では、レベルを参照する場合は、階層名を省略できます。ディメンジョンに同名のレベルが複数含まれている場合は、指定した名前を持つ最初のレベルが使用されます。レベルを明確に参照するには、階層名も指定するようにします。

別の例として、レベルでは、同じ名前のメンバを複数設定することができます。異なる都道府県に同じ名前を持つ市町村がある場合がありますが、それらの市町村は別々のメンバです。また、医師名などきめ細かなレベルを含むキューブの場合は、そのレベルに、名前が同じメンバが複数含まれる場合もあります。Business Intelligence では、名前を指定してメンバを参照すると、レベル内でその名前を持つ最初のメンバにアクセスすることになります。明確にメンバを参照するには、メンバのキーを使用します。"メンバのキー" を参照してください。

2.11.1 存在しないメンバ

ほとんどの場合、存在しないメンバに対しては NULL が返されます。これは、シェルではアスタリスク (*) として表されます。例えば、以下のクエリを考えてみます。

```
SELECT colord.h1.color.pink ON 0 FROM demomdx

                                  No Member
                                          *
```

ただし、メジャーについては例外です。メジャーは MEASURES ディメンジョンのメンバです。存在しないメジャーに対しては、エラーが返されます。以下はその例です。

```
SELECT MEASURES.[pat count] ON 0 FROM demomdx

ERROR #5001: Measure not found: pat count
```

2.11.2 入力エラー

ほとんどの状況で、入力エラーは、存在しないメンバの場合と同じように扱われます。以下はその例です。

```
SELECT colord.h1.color.MEMBERSSS ON 0 FROM demomdx

                                  No Member
                                          *
```

別の例を示します。

```
SELECT colord.MEMBERSSS ON 0 FROM demomdx

                                  No Member
                                          *
```

ただし、ディメンジョンまたはディメンジョン内の要素を参照する場合は、ディメンジョン名が必要です。ディメンジョン名を誤入力すると、エラーとして処理されます。

```
SELECT colorddd.h1.color.MEMBERS ON 0 FROM demomdx
ERROR #5001: Dimension not found: colorddd
```

キューブまたはサブジェクト領域の名前を誤入力すると、エラーとして処理されます。

```
SELECT colord.h1.color.MEMBERS ON 0 FROM demo
ERROR #5001: Cannot find Subject Area: 'DEMO'
```

2.12 このドキュメントで使用される規約

スペースの都合で、このドキュメントの残りの部分では以下の規約を使用します。

- COLUMNS と ROWS ではなく、0 と 1 を使用します。
- ディメンジョン、階層、レベル、およびメンバの名前については、可能な限り角括弧を省略します (″InterSystems MDX リファレンス″ の ″識別子″ を参照してください)。
- 必要な場合を除き、階層名を省略します (これは、InterSystems MDX で許可されています)。

また、クエリ例を素早く確認できるように、以下の規約を使用します。

- MDX の文、キーワード、および関数は、大文字で示します。
- キューブ要素などのユーザ指定の詳細は、本文を除き小文字で示します。

3

レベルを使用した作業

ここでは、Business Intelligence のレベルの詳細を説明すると共に、レベルを使用した作業を行うための主要な MDX 関数の概要を示します。

"BI サンプルのアクセス方法" も参照してください。

3.1 レベルの概要

レベルを使用するとデータをグループ化できます。また、レベルにはメンバがあります。

メンバは、キューブからレコードのセットを選択します。`City` レベルの場合、メンバ `Juniper` では、出身地が Juniper である患者が選択されます。逆に、キューブ内の 1 つのレコードは、1 つ以上のメンバに属します。

3.1.1 メンバで可能な重複

レベルのメンバは、互いに重複してもかまいません。つまり、指定されたレコードが複数のメンバに属することができます。このような状況は、レベルがリストに基づく場合に発生します。例えば、`Allergies` というレベルがあり、そのレベルにはアレルギーごとに 1 つのメンバが含まれているとします。1 人の患者は、複数のアレルギーを持っていることがあるため、そのレベルの複数のメンバに属している可能性があります。

3.1.2 NULL 値および NULL メンバ

レベルには NULL メンバを設定できます。このメンバは、そのレベルで使用されるデータに対する値を持たないレコードを選択します。通常、このメンバの名前は `None` です。

3.1.3 階層

レベルは階層に属します。詳細は、"ディメンジョンおよび階層を使用した作業" を参照してください。

3.2 レベルの単一メンバへのアクセス

単一メンバを選択するには、そのメンバを直接参照します。一般的な構文は以下のとおりです。

```
[dimension_name].[hierarchy_name].[level_name].[member name]
```

前述したように、InterSystems MDX では階層名を省略できます。同様に、レベル名も省略できます。

以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, allerd.[ant bites] ON 1 FROM demomdx
```

```
                                    %COUNT
ant bites                                47
```

3.2.1 メンバ名

メンバ名は、指定されたレベルで一意である必要はありません。つまり、キューブの構築時に、指定されたレベルでメンバ名が一意であることの確認は行われません。例えば、Doctor ディメンジョンには、同じ名前のメンバを複数含めることができます。

3.2.2 メンバのキー

適切に定義されたキューブの場合、各メンバには大文字と小文字が区別される一意のキーがあります。そのキーを使用してメンバを参照するには、以下の構文を使用します。

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

多くの場合、member_key はメンバ名と同じです。生成された NULL メンバの場合、キーは <null> になります。

システムによるメンバ・キーの生成方法の詳細は、"InterSystems MDX リファレンス" の参照セクション "キー値" を参照してください。

MDX には、メンバのキー (またはその他のプロパティ) へのアクセスに使用できる関数 (PROPERTIES) が用意されています。この関数については、このページで後述します。

3.3 レベルの複数メンバへのアクセス

レベルの複数メンバにアクセスするには、いくつかの方法があります。

まず、MEMBERS 関数を使用できます。この場合、構文は以下のようになります。

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

例えば、以下のように表示されます。

```
SELECT MEASURES.[%COUNT] ON 0, allerd.allergies.MEMBERS ON 1 FROM demomdx
```

```
                                    %COUNT
 1 No Data Available                   390
 2 additive/coloring agen               46
 3 animal dander                        34
 4 ant bites                            47
 5 bee stings                           36
 6 dairy products                       30
 7 dust mites                           35
 8 eggs                                 32
 9 fish                                 45
10 mold                                 51
11 nil known allergies                 140
12 peanuts                              58
13 pollen                               57
14 shellfish                            54
15 soy                                  36
16 tree nuts                            45
17 wheat                                52
```

また、以下のように、特定のレベルで選択する隣接メンバの範囲を指定することもできます。

```
member1:membern
```

例えば、以下のように表示されます。

```
SELECT MEASURES.[%COUNT] ON 0, {birthd.1942:birthd.1947} ON 1 FROM demomdx
```

```
                                    %COUNT
1 1942                                   6
2 1943                                   7
3 1944                                   6
4 1945                                  11
5 1946                                  12
6 1947                                   9
```

この場合、その範囲の末尾に使用するメンバのディメンジョン、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。

```
SELECT MEASURES.[%COUNT] ON 0, {birthd.1942:1947} ON 1 FROM demomdx
```

隣接しない複数のメンバを選択できます。そのためには、それらのメンバを直接参照し、以下のように中括弧で囲んだコンマ区切りのリストにそれらのメンバを配置します。

```
SELECT MEASURES.[%COUNT] ON 0, {allerd.eggs,allerd.soy,allerd.mold} ON 1 FROM demomdx
```

```
                                    %COUNT
1 eggs                                  32
2 soy                                   36
3 mold                                  51
```

3.4 レベル内のメンバの順序

キューブ定義内では、レベル定義によって、レベル内のメンバとそれらのメンバの既定の順序が、以下のように決定されます。

- 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。
- 日付レベルの場合、メンバは発生順に並べ替えられます。昇順か降順かは、キューブ内の定義によって決まります。

MEMBERS 関数では、メンバはレベルでの定義に従って既定の順序で返されます。以下はその例です。

```
SELECT gend.gender.MEMBERS ON 0,homed.city.MEMBERS ON 1 FROM demomdx
```

```
                              Female                Male
1 Cedar Falls                      58                   52
2 Centerville                      41                   58
3 Cypress                          51                   61
4 Elm Heights                      53                   65
5 Juniper                          58                   64
6 Magnolia                         58                   56
7 Pine                             64                   57
8 Redwood                          58                   53
9 Spruce                           47                   46
```

レベルのメンバのサブセットがあり、それらのメンバが既定の順序で返されるようにする場合は、以下の例のように、HIERARCHIZE 関数を使用します。

```
SELECT MEASURES.[%COUNT] ON 0, HIERARCHIZE({allerd.eggs,allerd.soy,allerd.mold}) ON 1 FROM demomdx
```

```
                                    %COUNT
1 eggs                                  32
2 soy                                   36
3 mold                                  51
```

この関数の詳細は、"セットを使用した作業" を参照してください。

3.5 相対的な位置に基づくレベル・メンバの選択

以下の MDX 関数を使用すると、指定したメンバに基づいて、レベルの特定のメンバを選択することができます。これらの関数はすべて、レベル内のメンバの既定の順序を使用します。なお、これらの詳細は、キューブ定義で定義した時間ディメンジョンとデータ・ディメンジョンに応じて異なります。

- NEXTMEMBER は、指定したレベルの次のメンバを返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[Q1 1920].NEXTMEMBER ON 1 FROM demomdx

                                    %COUNT
Q2 1920                                  *
```

- PREVMEMBER は、前のメンバを返します。
- LEAD 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。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[Q1 1920].LEAD(3) ON 1 FROM demomdx

                                    %COUNT
Q4 1920                                  1
```

- LAG 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。

時間ディメンジョンでは、これらのどの関数でもすべての親レベルが無視されます。例えば、PREVMEMBER 関数では、異なる親を持つメンバが返されることがあります。一方、データ・ディメンジョンでは、これらの関数それぞれで親レベルが考慮されます。例えば、PREVMEMBER 関数では、特定の同じ親メンバを持つ前のメンバのみが考慮されます (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 の定義" を参照してください)。これらの相違を示した例は、"InterSystems MDX リファレンス" を参照してください。

3.6 時間レベルの概要

時間レベルでは、レコードを時間でグループ化します。つまり、どのメンバも特定の日時に関連付けられたレコードで構成されています。例えば、トランザクション日というレベルで、トランザクションをその発生日でグループ化します。時間レベルには、一般的な種類として以下の 2 つがあります。これらの相違を理解しておくことが重要です。

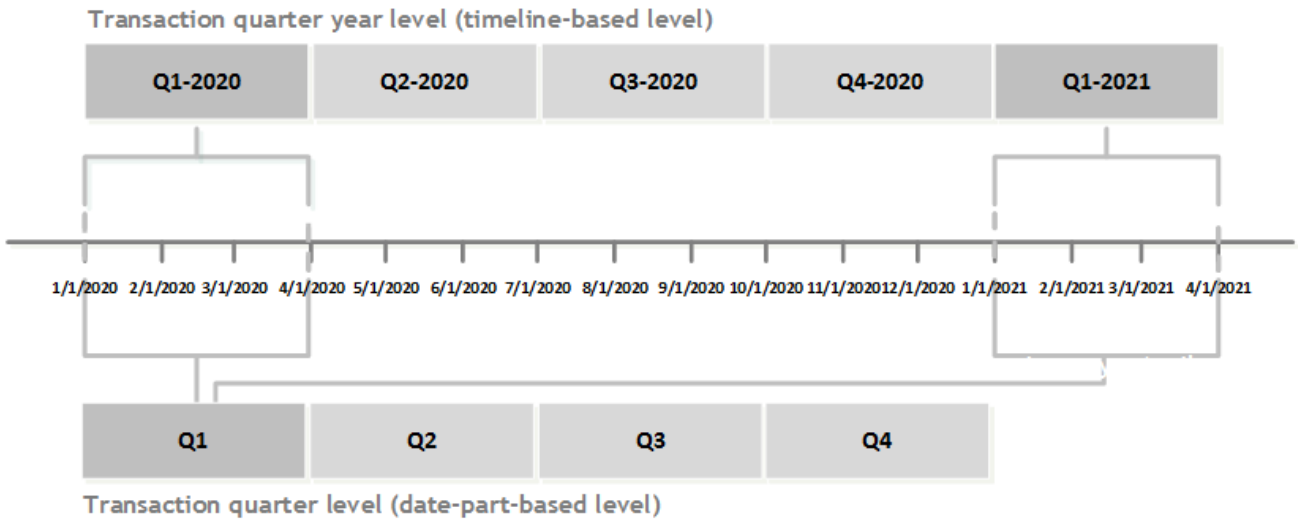
- 時間軸に基づく時間レベル : この種類の時間レベルでは、時間軸を互いに隣接する時間ブロックに分割します。このレベルのどのメンバも、単一の時間ブロックで構成されています。より正確に表現すると、その時間ブロックに関連付けられたレコードで各メンバが構成されています。　　というレベルでは、2011 年の第 1 四半期に属する日付に発生したすべてのトランザクションが `Q1-2011` メンバにグループ化されます。

  この種類のレベルでは、ソース・データに応じて任意の数のメンバを扱うことができます。

- 日付部分に基づく時間レベル : この種類の時間レベルでは、日付部分の値のみが考慮され、時間軸は無視されます。どのメンバも、以下に示すように、時間軸上の異なる位置にある複数の時間ブロックで構成されています。より正確に表現すると、これらの時間ブロックに関連付けられたレコードで各メンバが構成されています。
  　というレベルでは、あらゆる年の第 1 四半期に属する日付に発生したすべてのトランザクションが `Q1` メンバにグループ化されます。

  この種類のレベルでは、固定した数のメンバを扱います。

以下の図は、これらの種類の時間レベルを比較したものです。



これらの種類の時間レベルを同時に使用しても問題ありません。メンバをどのように組み合わせても、MDX では正しいレコード群が必ず返されます。

なお、MDX 関数の中には、時間軸に基づくレベルでは有用であっても、日付部分に基づくレベルでは役に立たないものがある点には注意が必要です。このような関数として、PREVMEMBER、NEXTMEMBER などがあります。

例えば、以下のクエリを考えてみます。これは日付部分に基づくレベルを表します。Q2 で PREVMEMBER を使用すると、予想どおりにエンジンによって Q1 のデータが返されます。

```
SELECT [BirthQD].[Q2].PREVMEMBER ON 1 FROM patients

Q1                                          219
```

しかし、セットの先頭にある Q1 で PREVMEMBER を使用すると、エンジンによって何も返されません。

```
SELECT [BirthQD].[Q1].PREVMEMBER ON 1 FROM patients

                                            *
```

この結果は、正しいものです。Q1 メンバは、すべての年の第一四半期に関連するレコードを表し、それより "前" のレコードにアクセスすることには意味がありません。

これに対して、以下のクエリを考えてみます。これは時間軸に基づくレベルを表します。

```
SELECT [BirthD].[Q1 2011].PREVMEMBER ON 1 FROM patients

Q4 2010                                     4
```

この場合、メンバは時間軸の特定の部分のレコードを表し、それより前のレコードを表すことに意味があります。

3.7 時間レベルで使用する専用の機能

InterSystems MDX には、時間レベルで使用する拡張機能が用意されています。このような機能として、NOW メンバ、%TIMERANGE 関数などがあります。

3.7.1 本日を基準にしたメンバの選択 (時間レベル)

日付/時間のレベルでは、NOW と呼ばれる特殊なメンバがサポートされています。このメンバは、現在の日付 (実行時) を使用して、レベルの適切なメンバにアクセスします。

例えば、以下のクエリは Year ディメンジョンの現在の年にアクセスします。

```
SELECT birthd.year.NOW ON 1 FROM demomdx
```

```
2011                                    9
```

別の例を示します。

```
SELECT birthd.[quarter year].NOW ON 1 FROM demomdx
```

```
Q2 2011                                 5
```

また、Business Intelligence では、NOW からのオフセットであるメンバを示すバリエーションも使用できます。例えば、[NOW-1] では、NOW より 1 つ前に位置するメンバが検索されます。

```
SELECT birthd.[quarter year].[NOW-1] ON 1 FROM demomdx
```

```
Q1 2011                                 1
```

これらのバリエーションは、以下のようなメンバの範囲内で使用できます。

```
SELECT birthd.[quarter year].[now-1]:birthd.[quarter year].now ON 1 FROM demomdx
```

```
1 Q1 2011                               1
2 Q2 2011                               5
```

詳細は、"InterSystems MDX リファレンス" の "日付/時間レベルの NOW メンバ" を参照してください。

3.7.2 時間レベルのメンバの範囲の選択

システムには MDX に対する拡張機能が用意されており、これによって時間レベルについてメンバの範囲を定義できます。この拡張機能は %TIMERANGE 関数です。引数は 3 つあり、それらは、開始メンバ、終了メンバ、およびキーワード (既定の INCLUSIVE または EXCLUSIVE) です。範囲のどちらか片方を省略できますが、範囲の両端は省略できません。

以下の例では、範囲の両端を使用しています。

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

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

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
```

3.8 プロパティへのアクセス

MDX では、レベルはそのレベルに固有のプロパティを持つことができます。レベルの各メンバは、プロパティに対して異なる値を持つことができます。これらのプロパティにアクセスして、クエリ結果に表示することができます。プロパティには、以下の 2 種類があります。

- ユーザ定義のプロパティ。Business Intelligence では、これらはキューブ定義内で定義されます。例えば、`DemoMDX` キューブでは、`City` レベルに `Population`（市町村の人口）と `Principal Export`（市町村の主要輸出品）という 2 つのプロパティがあります。
- 内部プロパティ。これらのプロパティには、メンバ名やメンバのキーなどの情報が含まれます。リストは、"InterSystems MDX リファレンス" の参照セクション "内部プロパティ" を参照してください。

プロパティの名前は、大文字と小文字が区別されません。

メンバのプロパティにアクセスするには、PROPERTIES 関数を使用します。以下はその例です。

```
SELECT homed.city.magnolia.PROPERTIES("Principal Export") ON 0 FROM demomdx
```

```
                                  bundt cake
```

別の例を示します。

```
SELECT homed.cypress.LEAD(1).PROPERTIES("name") ON 0 FROM demomdx
```

```
                                      name
                                    Magnolia
```

3.8.1 文字列式としてのプロパティ

MDX では、プロパティ値は文字列として扱われます。また、文字列リテラル（`"my label"` など）や連結演算子（`+`）もサポートされます。そのため、以下のような式を作成できます。

```
"Next after Cypress: " + homed.cypress.LEAD(1).PROPERTIES("name")
```

また、このような式を MDX クエリで使用できます。以下はその例です。

```
SELECT "Next after Cypress: " + homed.cypress.LEAD(1).PROPERTIES("name") ON 0 FROM demomdx
```

```
                              Expression
                        Next after Cypress: Magnolia
```

3.8.2 プロパティと属性

プロパティは属性とは異なります。属性は、MDX に関する説明でよく言及されます。

MDX の実装の中には、属性を使用してキューブを定義するものがあります。属性を直接使用する MDX 関数はありません。

システムが属性を使用することはありません。

4

ディメンジョンおよび階層を使用した作業

ここでは、Business Intelligence の階層とディメンジョンについて説明します。これらの要素はレベルのコンテナですが、それぞれ独自の目的も持っています。

"BI サンプルのアクセス方法" も参照してください。

4.1 ディメンジョンと階層の概要

ほとんどの MDX 関数は、レベルまたはレベルのメンバを直接参照します。ただし MDX では、レベルは階層に属し、階層はディメンジョンに属します。階層とディメンジョンを使用することで、レベルで提供される機能を超えた機能が実現されます。

階層は、データを特に空間および時間を軸に組織化する、現実的で便利な方法です。例えば、市区町村を国でグループ化したり、国を地域でグループ化できます。このような場合、特定の国の子都市に対するクエリや、特定の国の親郵便地域に対するクエリを実行できると便利です。そこで、そのようなクエリの記述を可能にするために、キューブではレベル間に階層を定義でき、MDX には、階層を使用した作業を可能にする関数が用意されています。

MDX では、ディメンジョンに、レコードを同様に分類する方法を指定する 1 つ以上の階層が含まれます。2 つの異なる階層間、またはある階層のレベルと別の階層のレベルとの間に、形式化されたリレーションシップはありません。ディメンジョンの用途は、その階層やレベルの既定の動作を定義することです。

4.1.1 Measures ディメンジョン

メジャーはすべて、Measures と呼ばれる特殊なディメンジョンに属します。このディメンジョンには、名前のない単一の階層が暗黙的に含まれます。この階層には、レベルは含まれません。この階層のメンバはメジャーです。

4.1.2 All レベル

Measures ディメンジョン以外の各ディメンジョンでは、そのレベルの全階層に表示される All レベルという特殊な任意のレベルを定義できます。このレベルを定義した場合、このレベルには、All メンバという、キューブ内のすべてのレコードに対応するメンバ 1 つが格納されます。

指定したディメンジョンの All メンバの実際の名前は、キューブの定義によって決まります。例えば、サンプルでの `AgeD` ディメンジョンの All メンバは、`All Patients` になります。

4.1.3 例

MDX シェルで cube コマンドを使用した場合、demomdx キューブ内に以下の要素が表示されます。

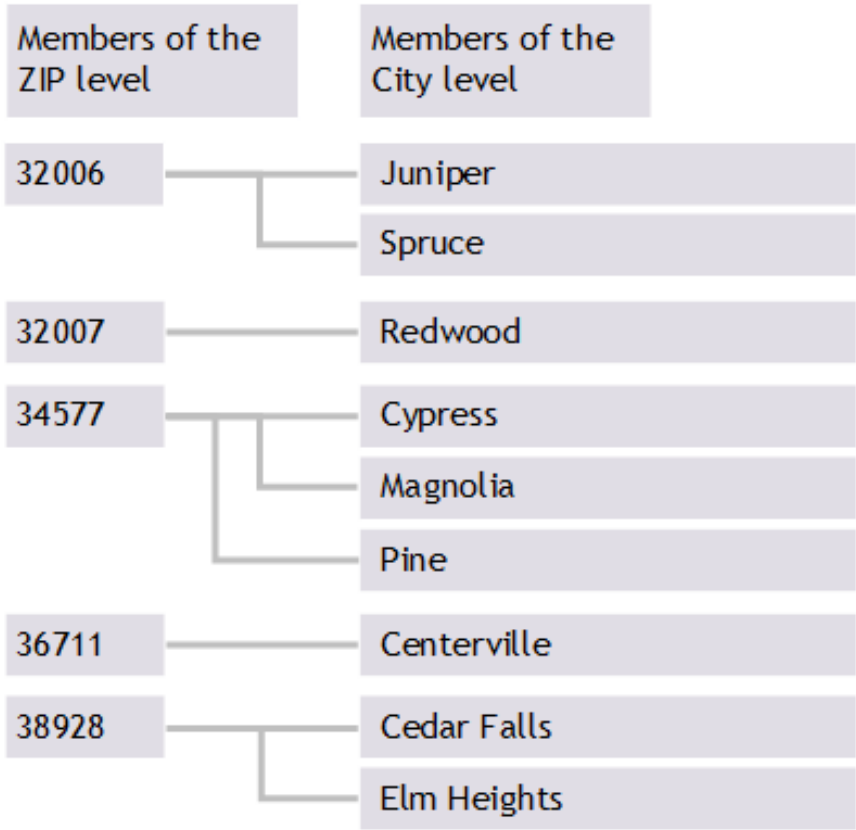
```
Elements of cube 'demomdx':
----------------------------------------
...
Dimensions
...
 HomeD
    H1
      ZIP
      City
...
```

HomeD ディメンジョンには 1 つの階層 (H1) が含まれており、この階層には以下の 2 つのレベルが含まれています。

- ZIP レベル
- City レベル

指定された階層で、レベルはその後にリストされているレベルの親になります。例えば、ZIP は City の親という意味です。より具体的には、ZIP の各メンバは、City の 1 つ以上のメンバの親です。つまり、簡潔に言うと、1 つのレベルは別のレベルの親ですが、実際のリレーションシップは、レベル間ではなくメンバ間にあります。この簡潔表現は必ずしも正確ではありませんが、便利なので広く使用されています。

以下の図は、HomeD.H1 階層のメンバ間のリレーションシップを示しています。



この階層の区別機能により、どの子要素もその親に対して一意になります。この例は、現実には郵便番号と市区町村には多対多のリレーションシップが存在するため形式的なものです。

4.2 階層のメンバへのアクセス

階層のメンバ (つまり、階層のすべてのレベルのすべてのメンバ) にアクセスするには、MEMBERS 関数を使用します。この場合、構文は以下のようになります。

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

InterSystems MDX では、階層名を省略すると、指定されたディメンジョン内で最初に表示される階層を参照するものと見なされます。

例えば、DemoMDX キューブ内では、homed ディメンジョンに階層が 1 つだけ含まれています。以下のクエリは、その階層のメンバを示しています。

```
SELECT MEASURES.[%COUNT] ON 0, homed.MEMBERS ON 1 FROM demomdx
```

```
                                    %COUNT
 1 32006                               215
 2 Juniper                             122
 3 Spruce                               93
 4 32007                               111
 5 Redwood                             111
 6 34577                               347
 7 Cypress                             112
 8 Magnolia                            114
 9 Pine                                121
10 36711                                99
11 Centerville                          99
12 38928                               228
13 Cedar Falls                         110
14 Elm Heights                         118
```

階層に対して MEMBERS 関数を使用すると、メンバのセットが階層順で返されます。最初のメンバは All メンバです (存在する場合)。その後のメンバは、それぞれ以下のいずれかになります。

- 前のメンバの最初の子。
- 前のメンバの次の兄弟。

別の例として、以下のクエリはすべてのメジャーを示しています (%COUNT を除く)。

```
SELECT gend.gender.MEMBERS ON 0, MEASURES.MEMBERS ON 1 FROM demomdx
```

```
                             Female                Male
1 Age                         18,413               17,491
2 Avg Age                      37.73                34.16
3 Allergy Count                  326                  332
4 Avg Allergy Count             1.08                 1.07
5 Test Score                  29,542               31,108
6 Avg Test Score               73.49                74.42
```

4.3 親子リレーションシップの使用法

システムには、親子リレーションシップを直接使用する以下の MDX 関数が用意されています。

- CHILDREN は、指定したメンバに子があれば返します。返される値はメンバのセットで、その順序は、レベルに指定された既定の順序に基づきます。以下はその例です。

  ```
  SELECT MEASURES.[%COUNT] ON 0, homed.zip.[34577].CHILDREN ON 1 FROM demomdx
  ```

  ```
                                      %COUNT
  1 Cypress                             112
  2 Magnolia                            114
  3 Pine                                121
  ```

別の例を示します。

```
SELECT MEASURES.[%COUNT] ON 0, homed.pine.CHILDREN ON 1 FROM demomdx

                                    %COUNT
                                  No Result
```

- PARENT は、指定したメンバに親があれば返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.[Elm Heights].PARENT ON 1 FROM demomdx

                                    %COUNT
38928                                   228
```

- FIRSTCHILD は、指定したメンバに子があれば最初の子を返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.zip.[34577].FIRSTCHILD ON 1 FROM demomdx

                                    %COUNT
Cypress                                 112
```

- LASTCHILD は、最後の子を返します (存在する場合)。

4.4 兄弟へのアクセス

システムには、メンバの兄弟にアクセスする以下の MDX 関数が用意されています。

- FIRSTSIBLING は、指定したメンバに兄弟があれば最初の兄弟を返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, birthd.[Q1 1920].FIRSTSIBLING ON 1 FROM demomdx

                                    %COUNT
Q1 1920                                 *
```

- LASTSIBLING は、最後の兄弟を返します (存在する場合)。
- SIBLINGS は、指定されたメンバとその兄弟をすべて返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, homed.cypress.SIBLINGS ON 1 FROM demomdx

                                    %COUNT
1 Cypress                               112
2 Magnolia                              114
3 Pine                                  121
```

4.5 従兄弟へのアクセス

COUSIN 関数を使用すると、上位レベルのメンバを指定して、従兄弟にアクセスすることができます。

例えば、以下のクエリは、1990 年内の Q1 1943 の従兄弟を検索します。

```
SELECT MEASURES.[%COUNT] ON 0, COUSIN(birthd.[Q1 1943],birthd.1990) ON1 FROM demomdx

                                  %COUNT
Q1 1990                                5
```

システムは相対的な位置を決定するために、キューブ定義で指定したように、レベル内のメンバの既定の順序を使用します。

4.6 子孫メンバへのアクセス

DESCENDANTS 関数を使用して、1 つ以上下位のレベル内で、指定されたメンバの子孫を取得できます。例えば、次のクエリは [BirthD].[H1].[Period] レベル内の 1990 年のすべての子孫を取得します。

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

```
 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                              *
```

DESCENDANTS 関数では、階層のさまざまな領域内の子孫にアクセスするための多くのオプションが用意されていますが、上記の使用法が最も一般的なシナリオです。

4.7 繰り返し内での現在のメンバへのアクセス

典型的なクエリでは、メンバのセットを繰り返し処理して、それぞれを行として表示することがよくあります。また、各メンバに何らかの具体的な操作を順番に実行する場合もあります。そのためには、現在のコンテキストで使用されているメンバにアクセスする CURRENTMEMBER 関数を使用します。

例えば、以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM demomdx
```

```
                                    %COUNT
1 Cedar Falls                          110
2 Centerville                           99
3 Cypress                              112
4 Elm Heights                          118
5 Juniper                              122
6 Magnolia                             114
7 Pine                                 121
8 Redwood                              111
9 Spruce                                93
```

このクエリでは、各市町村が 1 つの行になっています。表示されているデータは、%COUNT メジャーです。代わりに、市町村の人口を表示するとします。その場合のアクセスには、PROPERTIES 関数を使用します。この関数はその行で使用されるメンバへの参照が必要です。そのためには、CURRENTMEMBER 関数を使用します。この関数は以下のように呼び出すことができます。

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

この関数を使用すると、以下のようなクエリのバリエーションが作成できます。

```
SELECT homed.h1.CURRENTMEMBER.PROPERTIES("Population") ON 0, homed.city.MEMBERS ON 1 FROM demomdx

                                      H1
1 Cedar Falls                       90,000
2 Centerville                       49,000
3 Cypress                            3,000
4 Elm Heights                       33,194
5 Juniper                           10,333
6 Magnolia                           4,503
7 Pine                              15,060
8 Redwood                           29,192
9 Spruce                             5,900
```

別の例として、以下のクエリは Doctor のメンバの内部キーを示しています。

```
SELECT docd.h1.CURRENTMEMBER.PROPERTIES("KEY") ON 0, docd.[doctor].MEMBERS ON 1 FROM demomdx

                                     KEY
 1 None                             <null>
 2 Ahmed, Thelma                       34
 3 Alton, Chad                         35
 4 Black, Ashley                        4
..
```

5

セットを使用した作業

ここでは、Business Intelligence でセットを作成して使用する方法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。

5.1 セットの概要

セットにはゼロ個以上の要素が含まれ、一般的な種類の要素として、メンバ、スカラ値、タプルの 3 つがあります (タプルの詳細は、"タプルとキューブ" を参照してください)。

セットは、MDX クエリの軸に対して使用するだけでなく、他のセットの構築にも使用します。MDX セットは空でもかまいませんが、そのような空のセットを軸として使用することはできません。

5.2 セット式の作成

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、要素を任意の数だけ含めることができます。InterSystems MDX では、リスト内の要素が 1 つだけの場合、中括弧を省略できます。

各セット要素は以下のいずれかになります。

- member expression。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。
  - 単一メンバを返す MDX 関数を使用する式。
- セットを返す MDX 関数 (MEMBERS など) を使用する式。
- 同じレベル内のメンバの範囲 ("レベルを使用した作業" で説明)。

  ```
  member1:membern
  ```

- スカラ値。以下のいずれかです。

- メジャーへの参照。式 MEASURES.[Avg Test Score] はスカラ値です。この式では、すべてのコンテキストにおいて、数値または NULL のいずれかが返されます。
- 37 や "label" などの数値定数または文字列定数。
- (37+3)/2 などの数値式。
- 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。
- スカラ値を返す MDX 関数を使用する式。

例えば、PROPERTIES 関数はスカラ値を返します。この関数の概要は、"レベルを使用した作業" を参照してください。

別の例として、AVG 関数などの集計関数はスカラ値を返します。"集計の追加" を参照してください。

完全な詳細は、"InterSystems MDX リファレンス" を参照してください。

5.3 名前付きセットの作成

セットを作成して名前を割り当て、そのセットをさまざまな方法で再利用できるようにしておくと便利です。また多くの場合、クエリの構文は、名前付きセットを使用すると読みやすくなります。

クエリ内で 1 つ以上の名前付きセットを作成するには、以下のコマンドを使用します。

```
WITH with_clause1 with_clause2 ... SELECT query_details
```

以下は、この指定の説明です。

- 各式 with_clause1、with_clause2 (以降同様) の構文は以下のとおりです。

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

- query_details は MDX クエリです。

このようにすると、他のセット式を使用できるあらゆる場所で、名前を指定してその名前付きセットを参照することができます。

以下はその例です。

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

                                    %COUNT
1 Cedar Falls                         184
2 Centerville                         194
3 Cypress                             134
4 Elm Heights                         146
5 Juniper                             176
6 Magnolia                            169
7 Pine                                118
8 Redwood                             182
9 Spruce                              197
```

注釈　この名前付きセットは、クエリを範囲とする名前付きセットであり、その範囲はクエリです。セッションを範囲とする名前付きセットの詳細は、"InterSystems MDX リファレンス" の "CREATE SET 文" を参照してください。

キューブには、あらゆるクエリで使用できる追加の名前付きセットが含まれることがあります。"InterSystems Business Intelligence のモデルの定義" を参照してください。

5.4 セット内のメンバの順序

すべてのセットにはそれぞれ固有の順序があります (最初のメンバ、2 番目のメンバなど)。

セットを返す MDX 関数を使用すると、その関数によってそのセット内のメンバの順序が決定されます。MDX 関数では、可能な限り、キューブ定義の指定に従って、メンバの本来の順序が使用されます。

例えば、MEMBERS 関数は、レベルのメンバを、キューブ定義に指定されている順序で返します (通常は、レベルに応じてアルファベット順または日付順です)。

前のセクションの説明どおりにセットを作成する場合、要素をリストする順序によって、セット内のメンバの順序が制御されます。例えば、以下のようにセットを指定するとします。

```
{gend.gender.MEMBERS,allerd.allergies.MEMBERS}
```

このセットは、Gender ディメンジョンのメンバと、それに続く Allergies ディメンジョンのメンバで構成されます。

5.5 サブセットの選択

SUBSET は、位置に基づいて、指定されたセットからメンバのセットを返します。セット、開始位置、および返すメンバの数を指定します。開始位置は 0 です。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0,SUBSET(homed.city.MEMBERS,0,3) ON 1 FROM demomdx
```

```
                                  %COUNT
1 Cedar Falls                        110
2 Centerville                         99
3 Cypress                            112
```

EXCEPT 関数には、サブセットを取得する別の方法が用意されています。次のセクションを参照してください。

"クエリのフィルタ処理" も参照してください。

5.6 セットの並べ替え

ここでは、セットの並べ替え方法を説明します。以下のトピックについて説明します。

- メジャー値によって項目を並べ替える方法
- 上位または下位のサブセットを選択する方法
- 階層順を適用する方法

5.6.1 メジャー値によるセットの並べ替え

メンバを特定のメジャー値によって並べ替えると役立つことがよくあります。例えば、応答の順に部門を並べ替えることによって、最も応答の遅い部門を確認できるようにする必要がある場合が考えられます。または、総売上高の順に製品を並べ替えることによって、上位ランクの製品を確認できるようにする場合もあります。

指定した順序でセットを返すには、ORDER 関数を使用します。この関数には、セット・メンバの順序を決定する際に使用される値を指定する引数（通常はメジャーへの参照）を使用します。以下はその例です。

```
SELECT MEASURES.[avg test score] ON 0,
ORDER(homed.city.MEMBERS,MEASURES.[avg test score],BDESC) ON 1 FROM demomdx
```

```
                            Avg Test Score
1 Juniper                             75.08
2 Redwood                             75.07
3 Cedar Falls                         75.03
4 Elm Heights                         74.96
5 Pine                                74.76
6 Spruce                              74.47
7 Magnolia                            74.13
8 Cypress                             73.96
9 Centerville                         73.79
```

3 番目のオプションの引数は、以下のいずれかになります。

- ASC（既定）ー この引数を使用すると、該当する場合は、階層を保持したまま昇順で並べ替えが行われます。
- DESC ー この引数を使用すると、該当する場合は、階層を保持したまま降順で並べ替えが行われます。
- BASC ー この引数を使用すると、階層は保持されず、すべてのメンバが昇順で並べ替えられます。
- BDESC ー この引数を使用すると、階層は保持されず、すべてのメンバが降順で並べ替えられます。

例えば、以下のクエリでは階層は保持されません。

```
SELECT MEASURES.[avg test score] ON 0,
ORDER(homed.MEMBERS,MEASURES.[avg testscore],BDESC) ON 1 FROM demomdx
```

```
                            Avg Test Score
 1 Juniper                            75.08
 2 Redwood                            75.07
 3 32007                              75.07
 4 Cedar Falls                        75.03
 5 38928                              75.00
 6 Elm Heights                        74.96
 7 32006                              74.78
 8 Pine                               74.76
 9 Spruce                             74.47
10 34577                              74.28
11 Magnolia                           74.13
12 Cypress                            73.96
13 Centerville                        73.79
14 36711                              73.79
```

これに対して、以下のクエリでは階層が保持されます。

```
SELECT MEASURES.[avg test score] ON 0,
ORDER(homed.MEMBERS,MEASURES.[avg testscore],DESC) ON 1 FROM demomdx
```

```
                            Avg Test Score
 1 32007                              75.07
 2 Redwood                            75.07
 3 38928                              75.00
 4 Cedar Falls                        75.03
 5 Elm Heights                        74.96
 6 32006                              74.78
 7 Juniper                            75.08
 8 Spruce                             74.47
 9 34577                              74.28
10 Pine                               74.76
11 Magnolia                           74.13
12 Cypress                            73.96
13 36711                              73.79
14 Centerville                        73.79
```

5.6.2 上位または下位のサブセットの選択

何らかの方法で項目を並べ替えた後、上位または下位のサブセットを選択できると便利です（上位 5 つなど）。そのためには、以下の MDX 関数を使用します。

- メンバ数を指定することで、HEAD はセットの最初の部分、TAIL はセットの最後の部分を返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0,HEAD(homed.city.MEMBERS,3) ON 1 FROM demomdx
```

```
                                    %COUNT
1 Cedar Falls                          110
2 Centerville                           99
3 Cypress                              112
```

  返されたセットのメンバの順序は、元のセットの順序と同じです。

- TOPCOUNT は HEAD、BOTTOMCOUNT は TAIL に似ていますが、サブセットを抽出する前のセットの並べ替え方法を指定するオプションの引数も含みます。

  例えば、以下のクエリでは、患者数に基づいて上位の市町村が返されます。

```
SELECT MEASURES.[%COUNT] ON 0,TOPCOUNT(homed.city.MEMBERS,4,MEASURES.[%COUNT]) ON 1 FROM demomdx
```

```
                                    %COUNT
1 Juniper                              122
2 Pine                                 121
3 Elm Heights                          118
4 Magnolia                             114
```

  別の例を示します。

```
SELECT MEASURES.[avg test score] ON 0,TOPCOUNT(homed.city.MEMBERS,5,MEASURES.[avg test score]) ON
 1 FROM demomdx
```

```
                              Avg Test Score
1 Juniper                              75.08
2 Redwood                              75.07
3 Cedar Falls                          75.03
4 Elm Heights                          74.96
5 Pine                                 74.76
```

- TOPSUM は TOPCOUNT、BOTTOMSUM は BOTTOMCOUNT に似ています。ただし、返すメンバの数を指定する代わりに、メンバ全体の合計に適用される切り捨て値を指定します。例えば、売上で上位 500 万ドルを占める製品を取得することができます。
- TOPPERCENT は TOPCOUNT、BOTTOMPERCENT は BOTTOMCOUNT に似ています。ただし、返すメンバの数を指定する代わりに、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します。例えば、売上の上位 10 パーセントを占める製品を取得することができます。

5.6.3 階層順の適用

HIERARCHIZE 関数は、同じディメンジョンからメンバのセットを受け入れ、それらのメンバを含むセットを階層順に (階層で指定された順序で) 返します。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0,
HIERARCHIZE({homed.36711,homed.38928,homed.[elm heights],homed.Spruce}) ON 1 FROM demomdx
```

```
                                   %COUNT
1 36711                                99
2 Spruce                               93
3 38928                               228
4 Elm Heights                         118
```

これらのメンバがディメンジョン内の異なる階層に属する場合、それぞれの階層は任意の順序で返されます。

5.7 セットの組み合わせ

セットは、MDX クエリのビルディング・ブロックです。MDX クエリを記述する際には、各軸に対して使用するセットを指定する必要があります。セットを組み合わせるために使用できる以下の MDX 関数がサポートされています。

- UNION は、2 つのセットを組み合わせ (重複するメンバがあれば、必要に応じて破棄)、それらのセットのすべてのメンバを含むセットを返します。以下はその例です。

```
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 demomdx
```

```
                                    %COUNT
1 eggs                                   32
2 soy                                    36
3 wheat                                  52
```

  このクエリでは、UNION を ALL キーワードと組み合わせて使用していないため、重複は削除されます。

- INTERSECT は、2つのセットを検証し、両方のセット内にあるすべてのメンバを含むセットを返します。必要に応じて、重複を保持することも可能です。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 demomdx
```

```
                                    %COUNT
1 Magnolia                              114
2 Redwood                               111
3 Cypress                               112
4 Cedar Falls                           110
```

- EXCEPT は、2 つのセットを検証し、1 番目のセット内にあるメンバのうち、2 番目のセット内にも存在するものを削除します。必要に応じて、重複を保持することも可能です。以下はその例です。

```
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 demomdx
```

```
                                    %COUNT
1 eggs                                   32
2 soy                                    36
```

  ″%NOT 最適化″ も参照してください。

- CROSSJOIN は、2 つのセットのクロス積で構成されるセットを返します。両方のセットがメンバで構成されることが可能です。または、一方のセットがメンバで構成され、他方のセットがメジャーで構成されることも可能です。メジャーが両方のセットに含まれる場合は、分析エンジンによって `[Two measures cannot be crossjoined]` というエラーが出力されます。以下はその例です。

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

```
                                    %COUNT
 1 None->0 to 29                        389
 2 None->30 to 59                       333
 3 None->60+                            106
 4 asthma->0 to 29                       40
 5 asthma->30 to 59                      39
 6 asthma->60+                           11
 7 CHD->0 to 29                           *
 8 CHD->30 to 59                         12
 9 CHD->60+                              25
10 diabetes->0 to 29                      1
11 diabetes->30 to 59                    20
12 diabetes->60+                         24
13 osteoporosis->0 to 29                  *
14 osteoporosis->30 to 59                 *
15 osteoporosis->60+                     22
```

MDX シェルでは、NULL 値はアスタリスク (*) として表示されます。NULL 値の抑制の詳細は、"セットからの NULL 要素の削除" を参照してください。

また、NONEMPTYCROSSJOIN 関数も参照してください。

メンバのセットを返す前述の関数とは異なり、CROSSJOIN は一連のタプルを返す (NONEMPTYCROSSJOIN と同様) ことにも注意してください。タプルの詳細は、"タプルとキューブ" を参照してください。

5.8 メジャー値またはプロパティ値によるセットのフィルタ処理

セット内のメンバのメジャー値を検証し、それらの値を使用してセットをフィルタ処理することもできます。このためには、FILTER 関数を使用します。

FILTER 関数は、セットと論理式を使用します。セットが検証され、指定された論理式がメンバごとに True となるサブセットが返されます。この論理式は通常、メジャー値を定数と比較したり、別のメジャー値と比較したりします。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, FILTER(homed.city.MEMBERS,MEASURES.[%COUNT]>115) ON 1 FROM demomdx
```

```
                                %COUNT
1 Elm Heights                      118
2 Juniper                          122
3 Pine                             121
```

このフィルタ処理は集約レベルで生じると理解しておいてください。つまり、メジャー値はクエリ内の可能なメンバごとに計算されます。FILTER 関数では、それらの集約値が検討され、必要に応じてメンバが削除されます。

以下のように、同じ関数をメンバのプロパティと組み合わせて使用することができます。

```
SELECT homed.h1.CURRENTMEMBER.PROPERTIES("Population") ON 0,
FILTER(homed.city.MEMBERS,homed.h1.CURRENTMEMBER.PROPERTIES("Population")>20000) ON 1 FROM demomdx
```

```
                               ZIP
1 Cedar Falls                90,000
2 Centerville                49,000
3 Elm Heights                33,194
4 Redwood                    29,192
```

5.9 セットからの NULL 要素の削除

場合によっては、セットに NULL 要素が含まれることがあります。例えば、CROSSJOIN 関数は、(前のセクションで示したように) NULL 要素を返す可能性があります。

NON EMPTY キーワードをセット式の前に置くと、NULL 要素が抑制されます。以下はその例です。

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

```
                                %COUNT
 1 None->0 to 29                   389
 2 None->30 to 59                  333
 3 None->60+                       106
 4 asthma->0 to 29                  40
 5 asthma->30 to 59                 39
 6 asthma->60+                      11
 7 CHD->30 to 59                    12
 8 CHD->60+                         25
 9 diabetes->0 to 29                 1
10 diabetes->30 to 59               20
11 diabetes->60+                    24
12 osteoporosis->60+                22
```

5.10 重複の削除

セットを組み合わせるときには、重複を削除することができます。これは特に、複数の手順でセットの作成および組み合わせを行った場合に有効です。結果のセットに重複が含まれないようにするには、DISTINCT 関数を使用します。

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]>115)} ON 1 FROM demomdx
```

```
                                    %COUNT
1 Juniper                               122
2 Elm Heights                           118
3 Juniper                               122
4 Pine                                  121
```

以下のクエリと比較してみてください。そちらでは、重複する基準の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]>115)}) ON 1 FROM demomdx
```

```
                                    %COUNT
1 Juniper                               122
2 Elm Heights                           118
3 Pine                                  121
```

5.11 セットの要素のカウント

セットの要素をカウントするには、COUNT 関数を使用します。以下はその例です。

```
SELECT COUNT(docd.doctor.MEMBERS) ON 0 FROM demomdx
```

```
                                     COUNT
                                       41
```

既定では、COUNT はあらゆる空の要素を考慮して、空以外の要素と合せてカウントします。2 番目の引数として 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。

このことを確認するために、まず以下のクエリを考えてみます。

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

```
                           0 to 29          30 to 59               60+
1 None                          389               333               106
2 asthma                         40                39                11
3 CHD                             *                12                25
4 diabetes                        1                20                24
5 osteoporosis                    *                 *                22
```

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.diagnoses.MEMBERS' SELECT COUNT(myset) ON 0 FROM demomdx WHERE aged.[0 to 29]
```

```
                                     COUNT
                                         5
```

このクエリでは、COUNT が 5 を返します。これは、空のメンバも考慮しているためです。一方、以下では空のメンバは考慮されていません。

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

                                    COUNT
                                         3
```

6
タプルとキューブ

ここでは、Business Intelligence MDX におけるその他 2 つの主要な概念であるタプルとキューブを紹介します。

"BI サンプルのアクセス方法" も参照してください。

6.1 タプルの概要

タプルは、さまざまなディメンジョンのメンバを組み合わせたものです。各タプルには値が 1 つあります (NULL の場合もあります)。

結果セット内の各データ・セルがタプルです。例えば、以下のクエリを考えてみます。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.members ON 1 FROM demomdx
```

```
                                    %COUNT
1 Cedar Falls                          110
2 Centerville                           99
3 Cypress                              112
4 Elm Heights                          118
5 Juniper                              122
6 Magnolia                             114
7 Pine                                 121
8 Redwood                              111
9 Spruce                                93
```

このクエリでは、9 個のタプルから成るセットが返されます。例えば、最初のタプルは (`City` ディメンジョンの) `Cedar Falls` と (`Measures` ディメンジョンの) `%COUNT` を組み合わせたものです。

6.1.1 タプルの作成

タプルは、以下の構文を使用して直接作成することができます。

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

member_expr1、member_expr2、member_expr3 (以降同様) はメンバ式です。

MDX の他の実装では、これらのメンバ式それぞれが、別々のディメンジョンに関連付けられている必要があります。つまり、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、結果は NULL になります。これは、ほとんどの場合にレコードが 1 つのメンバにのみ属するためです。ただし、InterSystems IRIS 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ます。

6.1.2 完全修飾されたタプルと部分修飾されたタプル

タプルは、完全修飾されていても部分修飾されていてもかまいません。

- タプル式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。完全修飾されたタプルは、非常に少数のレコードを参照し、細かすぎて一般的に使用することができません。
- タプル式がキューブ内の各ディメンジョンを参照しない場合、そのタプルは部分修飾されています。部分修飾されたタプルは、特にクエリで使用されるデータをフィルタ処理するために使用する場合に非常に役立ちます。

  タプルが参照するメンバが 1 つだけの場合、タプルはそのメンバと同等になります。例えば、以下の式はどちらも同じデータにアクセスします。

```
(colord.red)
colord.red
```

  式 (colord.red) は、ColorD ディメンジョンの Red メンバを使用するタプル式です。

  式 colord.red は、ColorD ディメンジョンの Red メンバを参照するメンバ式です。

  それぞれの式は、好きな色が赤の患者にのみアクセスします。

6.1.3 タプルのセット

タプルのセットを作成するには、タプル式のコンマ区切りのリストを中括弧で囲みます。

```
{tuple_expression1, tuple_expression2, ...}
```

(MDX の他の実装では、セット内のあらゆるタプルにおいて、各タプルを同じように構築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。)

タプルのセットを作成するために、CROSSJOIN 関数または NONEMPTYCROSSJOIN 関数を使用する方法もあります。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, CROSSJOIN(gend.gender.MEMBERS,homed.city.members) ON 1 FROM demomdx
```

```
                                  %COUNT
 1 Female->Cedar Falls                 58
 2 Female->Centerville                 41
 3 Female->Cypress                     51
 4 Female->Elm Heights                 53
 5 Female->Juniper                     58
 6 Female->Magnolia                    58
 7 Female->Pine                        64
 8 Female->Redwood                     58
 9 Female->Spruce                      47
10 Male->Cedar Falls                   52
11 Male->Centerville                   58
12 Male->Cypress                       61
13 Male->Elm Heights                   65
14 Male->Juniper                       64
15 Male->Magnolia                      56
16 Male->Pine                          57
17 Male->Redwood                       53
18 Male->Spruce                        46
```

これらのセット式は、以下のように、セット式が許可されている場所であればどこでも使用することができます。

- クエリの軸として
- WITH 節の中で

- セットを使用する MDX 関数への引数として

6.2 タプル値

あらゆるタプルには値があります (NULL の場合もあります)。

タプルの値は、以下のように決定されます。

1. システムは、タプル式に使用されているメジャー以外のすべてのメンバに対応する行を、ファクト・テーブル内で検索します。
2. 次に、それらの行の値を以下のように検索します。
   - タプル式に特定のメジャーが含まれている場合、システムはファクト・テーブルの該当する行ごとに、そのメジャーの値を検索します。
   - タプル式に特定のメジャーが含まれていない場合は、既定のメジャー (通常、`%COUNT`) が使用されます。
3. システムは、そのメジャーに対して指定されている集約関数を使用して、それらの値をまとめて集約します。

例えば、以下のタプルを考えてみます。

```
(homed.32006,colord.red,allerd.[dairy products],MEASURES.[avg test score])
```

このタプルの値を決定するために、システムは、郵便番号 32006 に属し、好きな色が赤で、乳製品にアレルギーがあるすべての患者を、ファクト・テーブル内で検索します。次に、それらの患者の `Test Score` メジャーの値にアクセスして、それらの値の平均を求めます。

別の例として、(InterSystems MDX で許可されている) 以下のタプルを考えてみます。

```
(allerd.soy,allerd.wheat)
```

このタプルの値を決定するために、システムは、大豆と小麦の両方にアレルギーがある患者をカウントします。

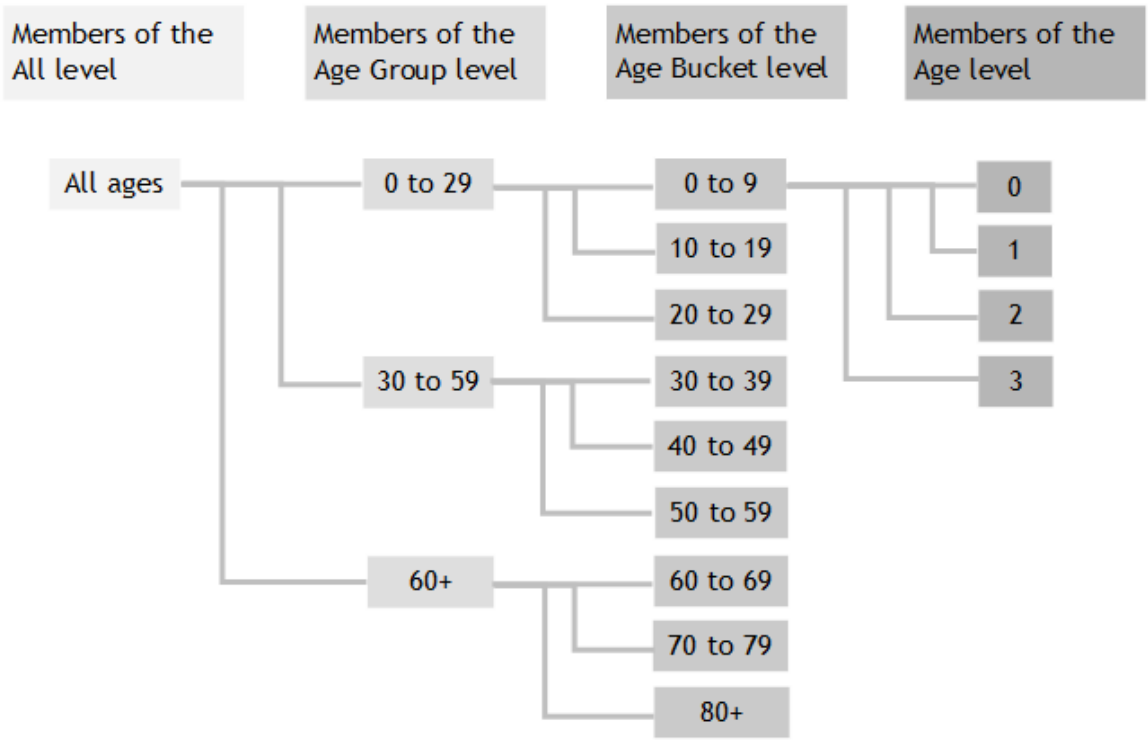
最後に、以下のタプルを考えてみます。

```
(homed.juniper,homed.centerville)
```

このタプルの値を決定するために、システムは、出身地が Juniper かつ Centerville の患者をカウントします。各患者の出身地は 1 つであるため、このタプルの値は NULL になります。

6.3 タプル式の例

タプル式は、1 つのディメンジョンのあらゆる階層内のあらゆるレベルで、メンバを参照できます。以下の (`Patients` キューブからの) ディメンジョンを考えてみます。このディメンジョンには階層が 1 つあり、そこには 4 つのレベルがあります。



これらのいずれかのレベルのメンバを使用するタプルを作成することができます。例えば、以下のタプル式のいずれかを使用できます。

```
(aged.[all patients])
(aged.[0 to 29])
(aged.5)
```

別の例として、上記の式のバリエーションを作成してみましょう。この場合、タプル式に他のディメンジョンのメンバを含めてみます。

```
(aged.[all patients],gend.male)
(aged.[0 to 29],diagd.asthma)
(aged.5,allerd.soy,colord.red)
```

6.4 タプルのセットをクエリの軸として使用する方法

タプルのセットをクエリの軸として使用できます。以下の例は、最も単純な、1 つのタプルで構成されているセットを示しています。

```
SELECT MEASURES.[%COUNT] ON 0, (homed.juniper,allerd.wheat,aged.[20 to 29]) ON 1 FROM demomdx

                                %COUNT
Juniper->wheat->20 to 29              1
```

以下の例は、クエリの軸として使用されている一連のタプルを示しています。

```
WITH SET myset AS
'{(homed.[cedar falls],allerd.soy,colord.red),(homed.magnolia,allerd.soy,colord.green),
(homed.34577,allerd.eggs,colord.green)}'
SELECT MEASURES.[%COUNT] ON 0, myset ON 1 FROM demomdx

                                %COUNT
1 Cedar Falls->soy->Red               *
2 Magnolia->soy->Green                1
3 34577->eggs->Green                  *
```

別の例として、以下は InterSystems MDX で有効なクエリです。

```
WITH SET myset AS
'{(homed.[cedar falls],allerd.soy,colord.green),(colord.red,allerd.soy,homed.pine,gend.male)}'
SELECT MEASURES.[%COUNT] ON 0, myset ON 1 FROM demomdx
```

```
                                  %COUNT
1 Cedar Falls->soy->Green               *
2 Red->soy->Pine->Male                  *
```

最後に、以下の例では、単一のディメンジョンを複数回参照するタプルを使用しています。

```
SELECT MEASURES.[%COUNT] ON 0,
{(allerd.soy,allerd.wheat),(homed.juniper,homed.centerville)} ON 1 FROM demomdx
```

```
                                  %COUNT
1 soy->wheat                            4
2 Juniper->Centerville                  *
```

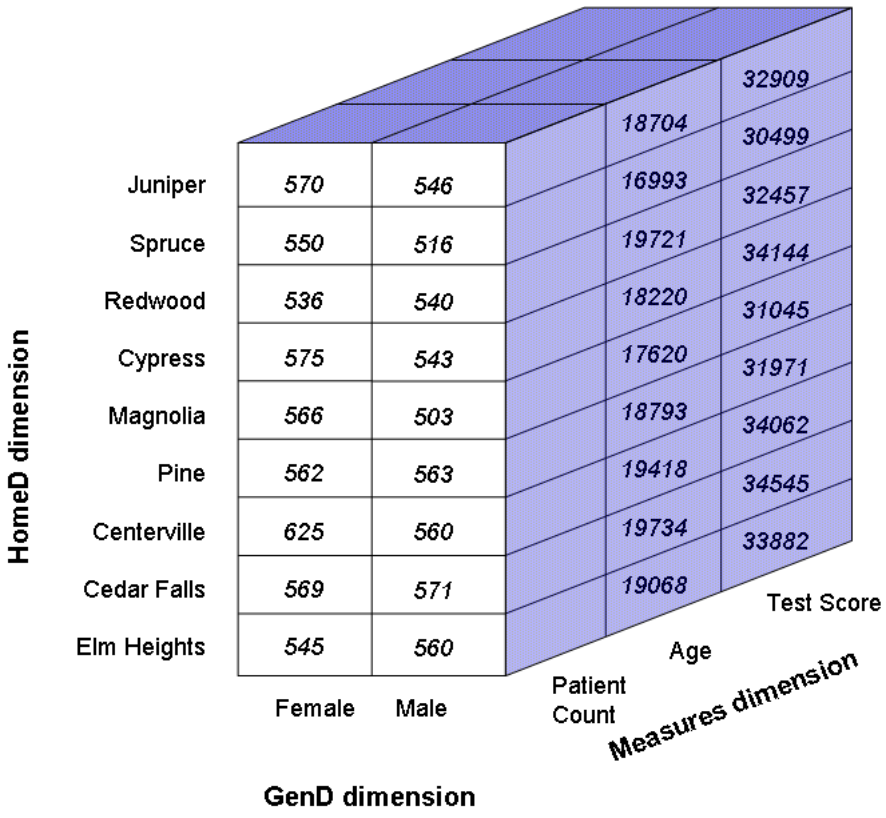
大豆と小麦の両方にアレルギーがある患者が 4 人います。

出身地が 2 つある患者はいません。

6.5 キューブの概要

キューブは、ディメンジョンごとに 1 つの軸 (または、エッジ) を含む多次元構造です。このキューブのセルはタプルです。MDX クエリは、キューブから特定のタプルを取得します。

少なくとも単純な事例では、このキューブを視覚化すると便利です。DemoMDX キューブには、10 個のディメンジョン (Measures ディメンジョンなど) があります。簡略化するために、以下の図では、それらのディメンジョンのうちの 3 つ (HomeD、GenD、および Measures) が示されています。実際に表示されているメジャーは 3 つだけであることに注意してください。



キューブのそれぞれの軸はセグメントに分割されています。セグメントは、対応するディメンジョンの最下位レベルのメンバごとに 1 つです。HomeD 軸の場合、これらのセグメントは City レベルのメンバです。

キューブ内の各セルは、完全修飾されたタプルです。各タプルには、図のように 1 つの値があります。

MDX クエリは、それぞれ値を持つ一連のタプルに対する要求です。以下のクエリを考えてみます。

```
SELECT CROSSJOIN(MEASURES.[%COUNT],gend.gender.MEMBERS) ON 0, homed.city.MEMBERS ON 1 FROM demomdx
```

```
                              Female                Male
1 Cedar Falls                    569                 571
2 Centerville                    625                 560
3 Cypress                        575                 543
4 Elm Heights                    545                 560
5 Juniper                        570                 546
6 Magnolia                       566                 503
7 Pine                           562                 563
8 Redwood                        536                 540
9 Spruce                         550                 516
```

このクエリでは、システムはキューブ内の該当するタプルを検索し、それらの値を取得します。例えば、最初のタプルは (homed.[cedar falls],gend.female,measures.[%COUNT]) です。このタプルの値は 569 です。

加算によって集約される各メジャー (Age など) は、キューブに直接含まれています。他のメジャーの場合、MDX はキューブの値を使用し、それらをメジャー定義に指定されているとおりに集約します。

例えば、Avg Age メジャーはキューブに直接含まれていませんが、Age メジャーは直接含まれています。Age メジャーに含まれているのは、タプル内に表されているすべての患者の累積年齢です。Avg Age メジャーを計算するために、MDX は Age を %COUNT で除算します。以下のクエリを考えてみます。

```
SELECT CROSSJOIN(MEASURES.[avg age],gend.gender.MEMBERS) ON 0, homed.city.members ON 1 FROM demomdx
```

```
                              Female                Male
1 Cedar Falls                 36.90               34.56
2 Centerville                 35.98               34.68
3 Cypress                     37.02               33.55
4 Elm Heights                 36.87               34.05
5 Juniper                     38.09               34.26
6 Magnolia                    35.64               35.03
7 Pine                        36.64               33.38
8 Redwood                     36.70               36.52
9 Spruce                      37.90               32.93
```

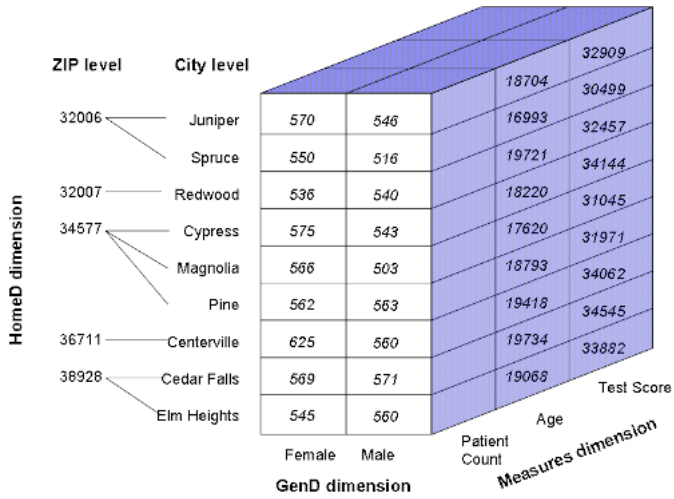
この例では、2 番目のタプルは (homed.[cedar falls],gend.male,measures.[avg age]) です。この値を取得するために、MDX は (homed.[cedar falls],gend.male,measures.[age]) の値を (homed.[cedar falls],gend.male,measures.[%COUNT]) の値で除算します。上記の結果に示されているように、19734 を 571 で除算すると 34.56 になります。

6.6 上位レベルとキューブ・ディメンジョン

ここでは、単一の階層を含むディメンジョンのみを考えてみます。

あらゆるディメンジョンにおいて、対応するキューブの軸に直接表されるのは、最下位のレベルのみです。

例えば、以下の図は、HomeD ディメンジョンのすべてのレベルを示しています。



HomeD 軸に含まれているのは、このディメンジョンのリーフ・メンバのみ、つまり、このディメンジョンの最下位レベルのメンバのみであることに注意してください。上位レベルは、下位メンバの組み合わせで構成されます。例えば、ZIP レベルの各メンバは、City ディメンジョンの 1 つ以上のメンバで構成されています。

ここで、以下のクエリを考えてみます。

```
SELECT CROSSJOIN(MEASURES.[%COUNT],gend.gender.MEMBERS) ON 0, homed.zip.members ON 1 FROM demomdx
```

```
                              Female                Male
1 32006                        1,120               1,062
2 32007                          536                 540
3 34577                        1,703               1,609
4 36711                          625                 560
5 38928                        1,114               1,131
```

このクエリでは、システムはキューブの該当するタプルを検索し、それらの値を取得します。

例えば、最初のタプルは (homed.[32006],gend.female,measures.[%COUNT]) です。メンバ 32006 は、市町村 Juniper および Spruce で構成されています。つまり、タプル (homed.[32006],gend.female,measures.[%COUNT]) は、以下のタプルの組み合わせで構成されています。

- (homed.[juniper],gend.female,measures.[%COUNT])
- (homed.[spruce],gend.female,measures.[%COUNT])

これらのタプルの値は、それぞれ 570 と 550 です。%COUNT メジャーは加算によって集約されるため、(homed.[32006],gend.female,measures.[%COUNT]) の値は 1120 です。

6.7 キューブ・ディメンジョン内の複数の階層

1 つのディメンジョンに複数の階層が存在する場合があります。複数の階層を持つディメンジョンの場合、キューブの対応する軸には、各階層内の最下位レベルのメンバごとにセグメントが 1 つ含まれます。

以下の理論上のキューブについて考えてみます。

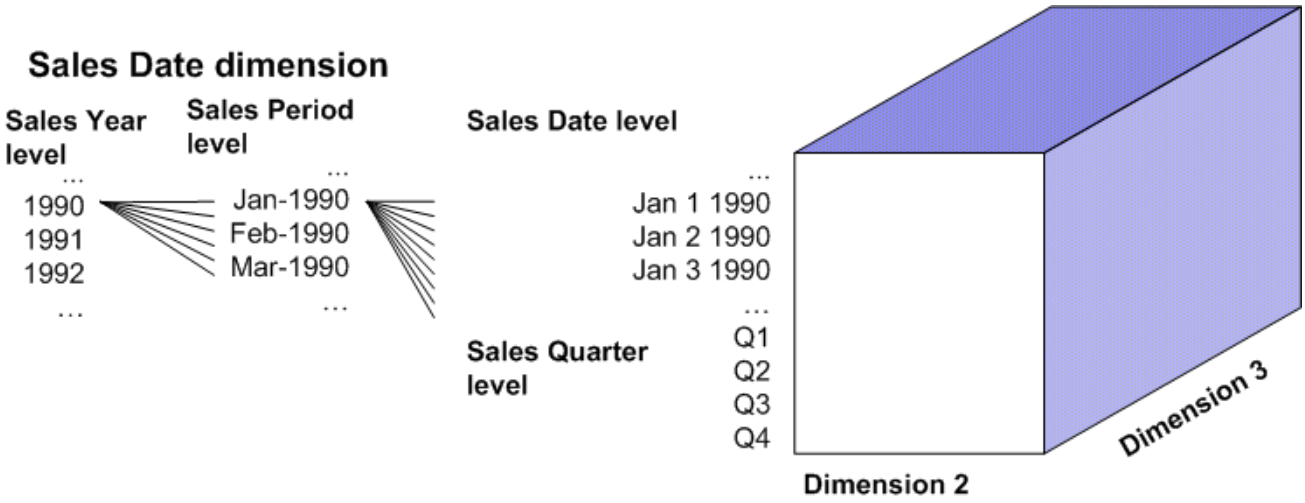
```
Members of cube 'theoretical':
---------------------------------------
...
Dimensions
...
  Sales Date
    H1
      Sales Year
      Sales Period
      Sales Date
    H2
      Sales Quarter
...
```

Sales Date ディメンジョンには、2 つの階層が含まれています。階層 H1 には、以下の 3 つのレベルがあります。

- Sales Year レベル。例えば、このレベルのメンバの 1 つは 1990 です。
- Sales Period レベル。例えば、このレベルのメンバの 1 つは Jan-1990 です。
- Sales Date レベル。例えば、このレベルのメンバの 1 つは Jan 3 1990 です。

もう 1 つの階層には、レベルが 1 つだけ含まれています。

この場合、Sales Date 軸には、Sales Date のメンバごとに 1 つのセグメントと、Sales Quarter のメンバごとに 1 つのセグメントが含まれています。以下はその例です。



(スペースの都合で、このキューブの図はタプルに分割されていません。)

例えば、クエリで Sales Quarter レベルが使用されている場合、システムは、その軸の適切な部分を使用して、要求されたタプルにアクセスします。

7

クエリのフィルタ処理

ここでは、Business Intelligence MDX クエリ内のデータをフィルタ処理する方法を説明します。

"BI サンプルのアクセス方法" も参照してください。

7.1 WHERE 節の概要

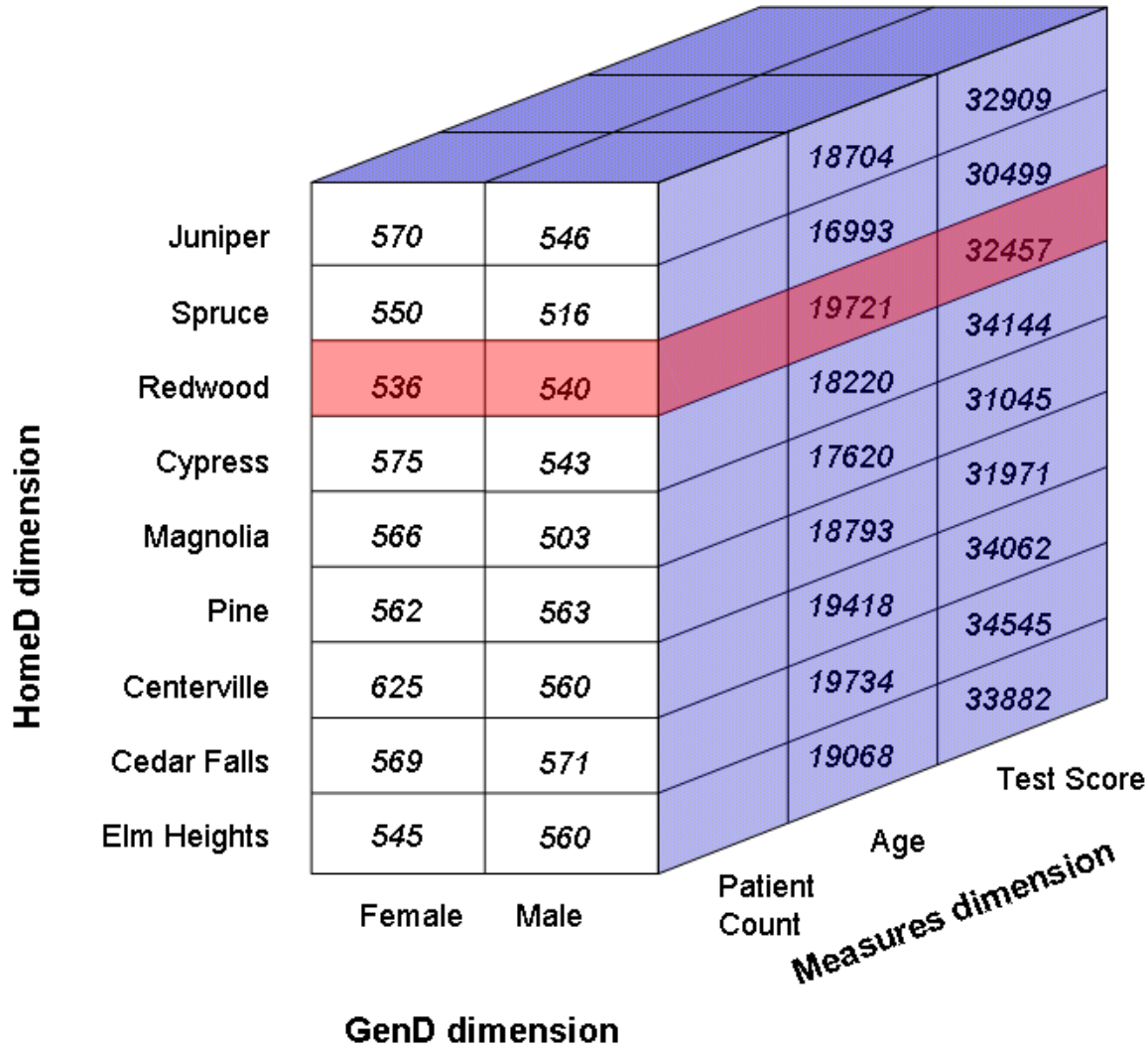
MDX クエリ自体にフィルタ (WHERE 節) を含めることもできます。MDX クエリの WHERE 節は、一般にスライサ軸と呼ばれます。WHERE 節に含まれているメンバが 1 つのみの場合、アクセスされるのはキューブの 1 つのスライスのみになります。

例えば、以下のクエリを考えてみます。

```
SELECT {MEASURES.[%COUNT],MEASURES.[avg age]} ON 0, gend.gender.MEMBERS ON 1 FROM demomdx WHERE
homed.redwood

                        Patient Count              Avg Age
1 Female                          536                36.70
2 Male                            540                36.52
```

このクエリは、キューブの 1 つのスライス (出身地が Redwood の患者のスライス) にのみアクセスします。以下はその例です。



この場合、Redwood スライスは、クエリが考慮するキューブの唯一の部分です。

ただし、WHERE 節で 1 つのセットまたは 1 つのタプルが使用される場合には、スライサ軸という語句はそれほど役に立ちません。このような場合には、キューブが忠実にスライスされないためです。

7.1.1 WHERE 節でのセットの使用

より一般的には、WHERE 節には 1 つのメンバ式の代わりにセット式を含めることができます。この場合、MDX では論理 AND を使用してレコードを組み合わせます。例えば、以下のクエリでは、好きな色が赤である患者と男性患者のみを使用しています。

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

                                %COUNT
1 Cedar Falls                       66
2 Centerville                       72
3 Cypress                           76
4 Elm Heights                       81
5 Juniper                           74
6 Magnolia                          63
7 Pine                              71
8 Redwood                           72
9 Spruce                            58
```

この場合、クエリでは、2 つのメンバで構成されたセット {colord.red,gend.male} を使用しています。システムはファクト・テーブルにアクセスすると、colord.red に関連付けられたレコードと gend.male に関連付けられたレコードを検索し、それらのレコードをすべて使用します。

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まとめて集約されます。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、"InterSystems Business Intelligence の実装" の "Business Intelligence クエリ・エンジンの仕組み" の項目にある "軸のたたみ込み" を参照してください。

次のセクションでは、さらに別の方法でクエリをフィルタ処理する方法について説明します。

7.1.2 WHERE 節でのタプルの使用

WHERE 節では、代わりに 1 つのタプルまたは一連のタプルを使用することもできます。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, NON EMPTY homed.city.MEMBERS ON 1 FROM demomdx
WHERE (aged.[age group].[60 +],gend.male)

                                    %COUNT
1 Cedar Falls                            7
2 Centerville                            9
3 Cypress
     12
4 Elm Heights                           14
5 Juniper                                8
6 Magnolia                               9
7 Pine                                   7
8 Redwood                                6
9 Spruce                                 2
```

別の例を示します。

```
WITH SET myset as '{(aged.[age group].[60 +],diagd.chd),(aged.[age group].[60+],diagd.asthma)}'
SELECT MEASURES.[%COUNT] ON 0, NON EMPTY homed.city.MEMBERS ON 1 FROM demomdx WHERE myset

                                    %COUNT
1 Cedar Falls                            5
2 Centerville                            5
3 Cypress                                8
4 Elm Heights                            3
5 Juniper                                3
6 Magnolia                               5
7 Pine                                   2
8 Redwood                                5
```

クエリ自体をフィルタ処理する場合は、NON EMPTY キーワードを使用して、NULL 以外の値だけをクエリが返すようにすると便利です。このキーワードは、NULL 値を返す可能性のあるセット式の先頭に含めます。以下はその例です。

```
SELECT MEASURES.[%COUNT] ON 0, NON EMPTY homed.city.MEMBERS ON 1 FROM demomdx
WHERE (aged.[age bucket].[30 to 39],diagd.chd)

                                    %COUNT
1 Elm Heights                            1
2 Magnolia                               1
```

一方、NON EMPTY を使用しなかった場合、結果は以下のようになります。

```
SELECT MEASURES.[%COUNT] ON 0, homed.city.MEMBERS ON 1 FROM demomdx
WHERE (aged.[age bucket].[30 to 39],diagd.chd)

                                    %COUNT
1 Cedar Falls                            *
2 Centerville                            *
3 Cypress                                *
4 Elm Heights                            1
5 Juniper                                *
6 Magnolia                               1
7 Pine                                   *
8 Redwood                                *
9 Spruce                                 *
```

7.2 %NOT 最適化

レベルの単一のメンバを除外することが必要な場合も多くあります。これを簡単に行うための方法として、InterSystems の拡張機能である %NOT 関数を使用できます。

```
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
```

%NOT 関数を使用するクエリは、EXCEPT を使用する同等のクエリよりも実行速度が速くなります。

7.3 %OR 最適化

多くの場合は、WHERE 節によって複数のメンバを表す必要があります。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。また、このクエリは前のクエリよりも高速です。

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。以下はその例です。

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

```
                              Patient Count
                                         71
```

8

集計の追加

ここでは、Business Intelligence で MDX クエリに集計 (平均や合計など) を追加する方法を説明します。

"BI サンプルのアクセス方法" も参照してください。

8.1 集計関数の概要

MDX には、指定されたセット全体で、指定された値を集計する関数が用意されています。各関数では、引数はセットとオプションの数値式 (メジャーへの参照など) です。システムにより、式がセットのメンバごとに評価され、単一の値が返されます。数値式が指定されていない場合は、代わりに、クエリ内で使用されているメジャー (場合によっては %COUNT) が評価されます。

関数は以下のとおりです。

- SUM。値の合計を返します。
- AVG。平均値を返します。この関数では、式が NULL のメンバは無視されます。
- MAX。最大値を返します。
- MIN。最小値を返します。
- MEDIAN。セットから中央値に最も近い値を返します。
- STDDEV。値の標準偏差を返します。
- STDDEVP。値の母標準偏差を返します。
- VAR。値の分散を返します。
- VARP。値の母分散を返します。

以下はその例です。

```
SELECT MAX(diagd.diagnoses.MEMBERS,MEASURES.[%COUNT]) ON 0 FROM demomdx
```

```
                                    MAX
                                     828
```

このクエリは、Diagnoses レベルのメンバの %COUNT メジャーの最大値を示しています。

別の例として、2 番目の引数を指定しないで同じ関数を使用してみます。この場合、クエリでは %COUNT メジャーが列として表示されます。

```
SELECT MEASURES.[%COUNT] ON 0, MAX(diagd.diagnoses.MEMBERS) ON 1 FROM demomdx
```

```
                                    %COUNT
MAX                                     828
```

別の例として、クエリ内でメジャーを一切指定しないで同じ関数を使用してみます。

```
SELECT MAX(diagd.diagnoses.MEMBERS) ON 0 FROM demomdx
```

```
                                       MAX
                                       828
```

この場合は、%COUNT が使用されます。

8.2 集計行の追加

集計値だけで表示するよりも、セットのすべての値を表示するクエリにその集計値を含める方が、より一般的です。このプロセスは、スプレッドシートで集計行を (行または列として) 追加することと似ています。

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、このセット全体にわたるそのメジャーの最大値が示されています。

```
SELECT MEASURES.[%COUNT] ON 0,
{diagd.diagnoses.MEMBERS, MAX(diagd.diagnoses.MEMBERS,MEASURES.[%COUNT])} ON 1 FROM demomdx
```

```
                                    %COUNT
1 None                                  828
2 asthma                                 90
3 CHD                                    37
4 diabetes                               45
5 osteoporosis                           22
6 MAX                                   828
```

システムでは、まずメンバごとの %COUNT メジャーが、そのメジャーに対して定義された集約メソッドを使用して計算されていることに注意してください。この場合は、患者がカウントされます。例えば、asthma メンバの合計 %COUNT 値は 90 です。その後で、MAX 関数によって、このメジャー (診断のセット全体) の最大値が取得されます。

別の例を示します。

```
SELECT {gend.gender.MEMBERS, AVG(gend.gender.MEMBERS,MEASURES.[%COUNT])} ON 0,
MEASURES.[%COUNT] ON 1 FROM demomdx
```

```
                              Female              Male               AVG
%COUNT                           488               512               500
```

集計関数を使用する場合、"セットを使用した作業" で説明したように、名前付きセットを使用すると便利な場合があります。例えば、以下のクエリは前のクエリと同等です。

```
WITH SET genders AS 'gend.gender.MEMBERS'
SELECT {genders, AVG(genders,MEASURES.[%COUNT])} ON 0, MEASURES.[%COUNT] ON 1 FROM demomdx
```

集計行を追加するには、表示されるメンバを組み合わせた集計メンバを定義する方法もあります。"計算メジャーおよび計算メンバの作成と使用" の "集計メンバの追加" を参照してください。

9

計算メジャーおよび計算メンバの作成

ここでは、Business Intelligence で計算メジャーと計算メンバを作成して使用する方法を説明します。

注釈　キューブには、あらゆるクエリで使用できる追加の計算メンバが含まれることがあります。"InterSystems Business Intelligence のモデルの定義" を参照してください。

"BI サンプルのアクセス方法" も参照してください。

9.1 計算メジャーおよび計算メンバの概要

MDX では、計算メンバを作成できます。この計算メンバは、他のメンバに基づくメンバです。メジャーとメジャー以外の 2 種類の計算メンバを定義できます (メジャーが、MEASURES ディメンジョンのメンバと見なされることを忘れないでください)。

- 計算メジャーは、他のメジャーに基づきます。例えば、あるメジャーは第 3 のメジャーで除算した第 2 のメジャーとして定義できます。

  計算メジャーという語句は、MDX の標準的な語句ではありません。このドキュメントでは、わかりやすくするためにこの語句を使用します。

- 一般に、非メジャーの計算メンバは、他の非メジャーのメンバと結合します。他の非メジャーのメンバと同様、この計算メンバはファクト・テーブル内のレコードのグループです。

  例えば、メンバ A がファクト・テーブル内の 150 個のレコードを参照し、メンバ B がファクト・テーブル内の 300 個のレコードを参照するとします。また、A と B を結合させるメンバ C を作成するとします。そうすると、メンバ C は、ファクト・テーブル内の該当する 450 個のレコードを参照します。

9.2 計算メンバの作成

クエリ内で 1 つ以上の計算メンバを作成するには、以下のような構文を使用します。

```
WITH with_clause1 with_clause2 ... SELECT query_details
```

Tipヒン 節と節の間にコンマを入れないことに注意してください。
ト

以下はその説明です。

- 各式 with_clause1、with_clause2 (以降同様) の構文は以下のとおりです。

  ```
  MEMBER MEASURES.[new_measure_name] AS 'value_expression'
  ```

  このページの後続のセクションでは、value_expression について説明します。
- query_details は MDX クエリです。

このようにすると、他のメンバを使用できるあらゆる場所で、名前を指定してその計算メンバを参照することができます。以下はその例です。

```
WITH MEMBER MEASURES.avgage AS 'MEASURES.[age]/MEASURES.[%COUNT]'
SELECT MEASURES.avgage ON 0, diagd.diagnoses.members ON 1 FROM demomdx

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

注釈　この計算メンバは、クエリを範囲とする計算メンバであり、その範囲はクエリです。セッションを範囲とする計算メンバの詳細は、"InterSystems MDX リファレンス" の "CREATE MEMBER 文" を参照してください。

9.3 計算メジャーの MDX レシピ

このセクションでは、広く必要とされるいくつかの計算メジャーの MDX 式を作成する方法について説明します。

- 一般的な他のメジャーの組み合わせ
- 集約値のパーセンテージ
- 個別のメンバのカウント
- 準加法メジャー
- フィルタ処理メジャー
- 他の期間を参照するメジャー
- ピボット・テーブルの他のセルを参照するメジャー

9.3.1 他のメジャーの組み合わせ

計算メジャーでは、値式の形式は、メジャー式を組み合わせた数式であることがよくあります。以下はその例です。

```
(MEASURES.[measure A] + MEASURES.[measure B]) * 100
```

または、以下のようになります。

```
(MEASURES.[measure A] + MEASURES.[measure B])/MEASURES.[measure C]
```

さらに正式には、この式では以下の要素を使用できます。

- メジャーへの参照。
- 数値リテラル。例えば、37 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。

- 算術演算子。InterSystems IRIS Business Intelligence 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、+ (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も使用できます。

  例えば、MEASURES.[%COUNT] / 100 です。

- AVG、MAX、COUNT など、数値を返す MDX 関数。

  既に説明した関数に加えて、Business Intelligence では、いくつかのスカラ関数 (SQRT、LOG、および POWER) もサポートされます。

Tipヒント MDX 関数 IIF は、このような式でよく役に立ちます。これは、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。この関数を使用すると、例えば、0 で除算することがなくなります。

9.3.2 集約値のパーセンテージ

合計レコード数のパーセンテージや他の集約値のパーセンテージの計算が必要な場合も多くあります。このような場合は、インターシステムズによる拡張機能の %MDX 関数を使用できます。この関数は、単一の値を返す MDX クエリを実行し、関数を実行したコンテキストの影響を受けない値を返します。このため、以下のような値式で定義されたメジャーを使用してパーセンテージを計算できます。

```
100 * MEASURES.[measure A] / %MDX("SELECT FROM mycube")
```

例えば、以下のように表示されます。

```
WITH MEMBER MEASURES.PercentOfAll AS '100 * MEASURES.[%COUNT]/%MDX("SELECT FROM demomdx")'
SELECT MEASURES.PercentOfAll ON 0, diagd.MEMBERS ON 1 FROM demomdx

                                PercentOfAll
1 None                                 84.56
2 asthma                                6.85
3 CHD                                   3.18
4 diabetes                              4.89
5 osteoporosis                          2.21
```

9.3.3 個別のメンバ数

場合によっては、特定のセルに対して、特定のレベルの個別のメンバ数のカウントが必要になります。例えば、DocD ディメンジョンにレベル Doctor が含まれているとします。この場合、指定した一連の患者の一次診療医である一意の医者の数をカウントすることが可能です。そのためには、以下の value_expression を使用する計算メジャーを定義します。

```
COUNT([docd].[h1].[doctor].MEMBERS,EXCLUDEEMPTY)
```

このメジャーは、クエリで以下のように使用できます。

```
WITH MEMBER MEASURES.[distinct doctor count] AS 'COUNT(docd.doctor.MEMBERS,EXCLUDEEMPTY)'
SELECT MEASURES.[distinct doctor count] ON 0, aged.[age bucket].MEMBERS ON 1 FROM demomdx

                          distinct doctor co
1 0 to 9                                  38
2 10 to 19                                38
3 20 to 29                                38
4 30 to 39                                40
5 40 to 49                                41
6 50 to 59                                40
7 60 to 69                                33
8 70 to 79                                31
9 80+                                     28
```

9.3.4 準加法メジャー

準加法メジャーは、すべてではなくほとんどのディメンジョンを集約するメジャーです。例えば、銀行残高は特定の時点のスナップショットであるため、顧客の銀行残高を経時的に加算することはできません。このようなメジャーを作成するには、インターシステムズによる MDX への拡張機能である %LAST 関数を使用できます。

以下のメジャーについて考えてみます。

- `Balance` はソース・プロパティ `CurrentBalance` に基づき、合計によって集約されます。

  このメジャーの経時的な集計は不適切な結果が生じるため、回避する必要があります。このためこのメジャーの使用は行または列に時間レベルが含まれるピボット・テーブルのみに限定する必要があります。

- `Transactions` は、ソース・プロパティ `TxCount` に基づき、合計によって集約されます。

`LastBalance` と呼ばれる計算メジャーを定義して、以下の value_expression を使用します。

```
%LAST(Date.Day.Members,Measures.Balance)
```

%LAST 関数は、指定セットの各メンバに対して評価されたメジャーの最終的な欠落のない値を返します。この場合は、値が存在する最後の日が検索され、その値が返されます。

9.3.5 フィルタ処理メジャー (タプル・メジャー)

通常のメジャーでは、ファクト・テーブル内でソース値が NULL でないすべてのレコードが考慮されます。状況によっては、以下のように動作するフィルタ処理メジャーの定義が必要になります。

- 特定のレコードでメジャーが NULL である。
- その他のレコードについては、そのメジャーに値が存在する。

フィルタ処理メジャー (非公式にはタプル・メジャーとも呼ばれます) には、以下のような value_expression を使用します。

```
([MEASURES].[my measure],[DIMD].[HIER].[LEVEL].[member name])
```

この場合、value_expression は、以下のとおりのタプル式です。

- `[MEASURES].[my measure]` は、基礎として使用されるメジャーです。
- `[DIMD].[HIER].[LEVEL].[member name]` は、メジャー値が非 NULL のメンバです。

例えば、`Avg Test Score` メジャーがあるとします。これは、テストの値が NULL でないすべての患者の平均テスト・スコアです。`Avg Test Score` メジャーに加え、冠動脈性心疾患 (CHD 診断) を持つ患者の平均テスト・スコアのみを示す列も表示する必要があるとします。このため、メジャー `Avg Test Score - CHD` が必要になります。この場合、以下の value_expression を指定した計算メジャーを作成できます。

```
(MEASURES.[avg test score],diagd.h1.diagnoses.chd)
```

例えば、以下のように表示されます。

```
WITH MEMBER MEASURES.[avg test score - chd] AS
'(MEASURES.[avg test score],diagd.h1.diagnoses.chd)'
SELECT MEASURES.[avg test score - chd] ON 0, aged.[age bucket].MEMBERS ON 1 FROM demomdx

                        avg test score - c
1 0 to 9                                 *
2 10 to 19                               *
3 20 to 29                               *
4 30 to 39                               *
5 40 to 49                           78.00
6 50 to 59                           75.75
7 60 to 69                           80.71
8 70 to 79                           83.33
9 80+                                55.25
```

9.3.6 別の期間のメジャー

多くの場合、以前の期間の指定のメジャーの値を表示しながら、以降の期間を表示すると便利です。例えば、UnitsSoldPreviousPeriod という名前の計算メジャーを定義して、以下の value_expression を使用できます。

```
([DateOfsale].[Actual].CurrentMember.PrevMember ,MEASURES.[units sold])
```

このメジャーは、その定義方法により、クエリの他の軸に DateOfSale ディメンジョンを使用する場合にのみ意味があります。以下はその例です。

```
WITH MEMBER [MEASURES].[UnitsSoldPreviousPeriod] AS
'([DateOfsale].[Actual].CurrentMember.PrevMember ,MEASURES.[units sold])'
SELECT {[Measures].[Units Sold],[MEASURES].[UNITSSOLDPREVIOUSPERIOD]} ON 0,
[DateOfSale].[Actual].[MonthSold].Members ON 1 FROM [HoleFoods]

                          Units Sold           DateOfSale
 1 Jan-2009                       15                    *
 2 Feb-2009                       10                   15
 3 Mar-2009                       13                   10
 4 Apr-2009                       15                   13
 5 May-2009                       22                   15
...
```

2 番目の列のキャプションが、定義した計算メンバの名前ではなく値式内で使用されるディメンジョンに基づいていることに注意してください。%LABEL 関数を使用すると、より適したキャプションを指定できます。以下はその例です。

```
WITH MEMBER [MEASURES].[UnitsSoldPreviousPeriod] AS
'([DateOfsale].[Actual].CurrentMember.PrevMember ,MEASURES.[units sold])'
SELECT {[Measures].[Units Sold],%LABEL([MEASURES].[UNITSSOLDPREVIOUSPERIOD],"Units (Prv Pd)","")} ON
0,
[DateOfSale].[Actual].[MonthSold].Members ON 1 FROM [HoleFoods]

                          Units Sold       Units (Prv Pd)
 1 Jan-2009                       15                    *
 2 Feb-2009                       10                   15
 3 Mar-2009                       13                   10
 4 Apr-2009                       15                   13
 5 May-2009                       22                   15
 6 Jun-2009                       17                   22
 7 Jul-2009                       24                   17
 8 Aug-2009                       30                   24
...
```

これらの例は、時間ベースのレベルを使用します。これは、この種の分析は時間レベルに対して一般的であるためです。ただし、同じ手法をデータ・レベルに対しても使用できます。

9.3.7 他のセルを参照するメジャー

多くの場合、ピボット・テーブルの別のセルの値を参照すると便利です。このためには、%CELL 関数および %CELLZERO 関数を使用できます。これらの関数はそれぞれ、ピボット・テーブルの別のセルの値を位置別に返します。指定された呼び出しに値がない場合、%CELL は NULL を返します。これに対して、%CELLZERO はゼロを返します。

これらの関数には多くの用途があります。1 つの例として、ある時点までの合計を計算するために %CELL を使用できます (この例では、インチ単位の積算降雨量です)。

```
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

9.4 メジャー以外の計算メンバの MDX レシピ

ここでは、一般的なシナリオのいくつかにおけるメジャー以外の計算メンバのレシピについて説明します。

- 年齢メンバ
- ハードコードされたメンバのセットを組み合わせたメンバ
- 条件リストに指定されたメンバを組み合わせたメンバ
- 日付の範囲を組み合わせたメンバ
- 他のメンバを横断するメンバ

9.4.1 年齢メンバの定義

年齢別にレコードをグループ化したメンバを用意すると役に立つことが普通です。このようなメンバを定義するには、既存の時間レベルと専用の NOW メンバを使用します。例えば、HoleFoods サンプルの、MonthSold レベルを考えてみましょう。以下の value_expression を使用して、3 Months Ago という名前の計算メンバを定義することができます。

```
[dateofsale].[actual].[monthsold].[now-3]
```

例えば、以下のように表示されます。

```
WITH MEMBER CalcD.[3 months ago] as '[dateofsale].[actual].[monthsold].[now-3]'
SELECT calcd.[3 months ago] ON 0, {MEASURES.[units sold], MEASURES.target} ON1 FROM holefoods
```

	3 months ago
1 Units Sold	37
2 Target	254.00

9.4.2 ハードコードされたメンバの組み合わせの定義

多くの場合、同じレベルの複数のメンバを結合する広い範囲のグループ化を定義すると便利です。そのためには、以下の形式の value_expression を持つ、メジャー以外の計算メンバを作成します。

```
%OR({member_expression, member_expression,...})
```

例えば、以下のように作成できます。

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

メジャー以外のメンバのそれぞれが一連のレコードを参照します。%OR 関数を使用するメンバを作成する際は、そのコンポーネント・メンバが使用するすべてのレコードを参照するメンバを新規作成します。

以下はその例です。

```
WITH MEMBER CalcD.[primary colors] as '%OR({colord.red,colord.blue,colord.yellow})'
SELECT calcd.[primary colors] ON 0,
{MEASURES.[%COUNT], MEASURES.[avg test score]} ON 1 FROM demomdx
```

9.4.3 条件リストにより定義されるメンバの組み合わせの定義

用語リストにより、プログラミングせずに Business Intelligence モデルをカスタマイズする方法が提供されます。用語リストは、キーと値のペアの単純なリスト (ただし、拡張可能) です。用語リストは複数の方法で使用できます。1 つはメンバ・セットを作成する方法で、一般的にフィルタで使用します。

この場合、%TERMLIST 関数と %OR 関数を使用します。以下の形式の value_expression を持つ、メジャー以外の計算メンバを作成します。

```
%OR(%TERMLIST(term_list_name))
```

term_list_name は、用語リストの名前に評価される文字列です。

以下はその例です。

```
%OR(%TERMLIST("My Term List"))
```

この式は、用語リストで示されたメンバに属するすべてのレコードを参照します (%OR は複数のメンバを単一のメンバに結合することを思い出してください)。

%TERMLIST 関数には、オプションの 2 番目の引数があります。"EXCLUDE" をこの引数に指定すると、関数はそのレベルで用語リストにないすべてのメンバのセットを返します。

9.4.4 日付範囲の集約

%OR によって集約されるメンバの範囲を使用する形式も便利です。

```
%OR(member_expression_1:member_expression_n)
```

式 member_expression_1:member_expression_n は、member_expression_1 から member_expression_n までのすべてのメンバを返します (範囲の端点も含みます)。この形式を使用すると、日付範囲を簡潔な形式で表現できるため、特に時間レベルで役立ちます。

時間レベルでは、特殊な NOW メンバも使用できます。以下の式は、90 日前から本日までの売上レコードを集計します。

```
%OR(DateOfSale.DaySold.[NOW-90]:DateOfSale.DaySold.[NOW])
```

または、これに相当する以下の形式を使用します。

```
%OR(DateOfSale.DaySold.[NOW-90]:[NOW])
```

日付範囲の取得には、PERIODSTODATE 関数も使用できます。例えば、以下の式は、現在の年の初めから本日までの日付範囲を取得し、これらの日数を集約します。

```
%OR(PERIODSTODATE(DateOfSale.YearSold,DateOfSale.DaySold.[NOW]))
```

9.4.5 他のメンバの共通部分としてのメンバの定義

場合によっては(特にフィルタを定義する場合)、メンバの共通部分であるメンバを定義すると便利です。以下のような場合 (リテラル構文を表示しない) にフィルタが必要だとします。

```
Status = "discharged" and ERvisit = "yes" and PatientClass="infant"
```

また、このフィルタを多くの場所で使用する必要があるとします。

フィルタ式を繰り返し定義するのではなく、計算メンバを定義して使用できます。この計算メンバに、以下のような**式**を指定します。

```
%OR({member_expression,member_expression,...}
```

以下はその例です。

```
%OR({birthd.year.NOW,allersevd.[003 LIFE-THREATENING]}
```

式 (`birthd.year.NOW,allersevd.[003 LIFE-THREATENING]`) はタプル式で、メンバ `birthd.year.NOW` とメンバ `allersevd.[003 LIFE-THREATENING]` の共通部分、つまり今年生まれた、生死にかかわるアレルギーを持つすべての患者を示します。