InterSystems IRIS Business Intelligence のモデルの定義

Version 2024.1
2024-06-03

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

InterSystems IRIS Business Intelligence のモデルの定義
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

1

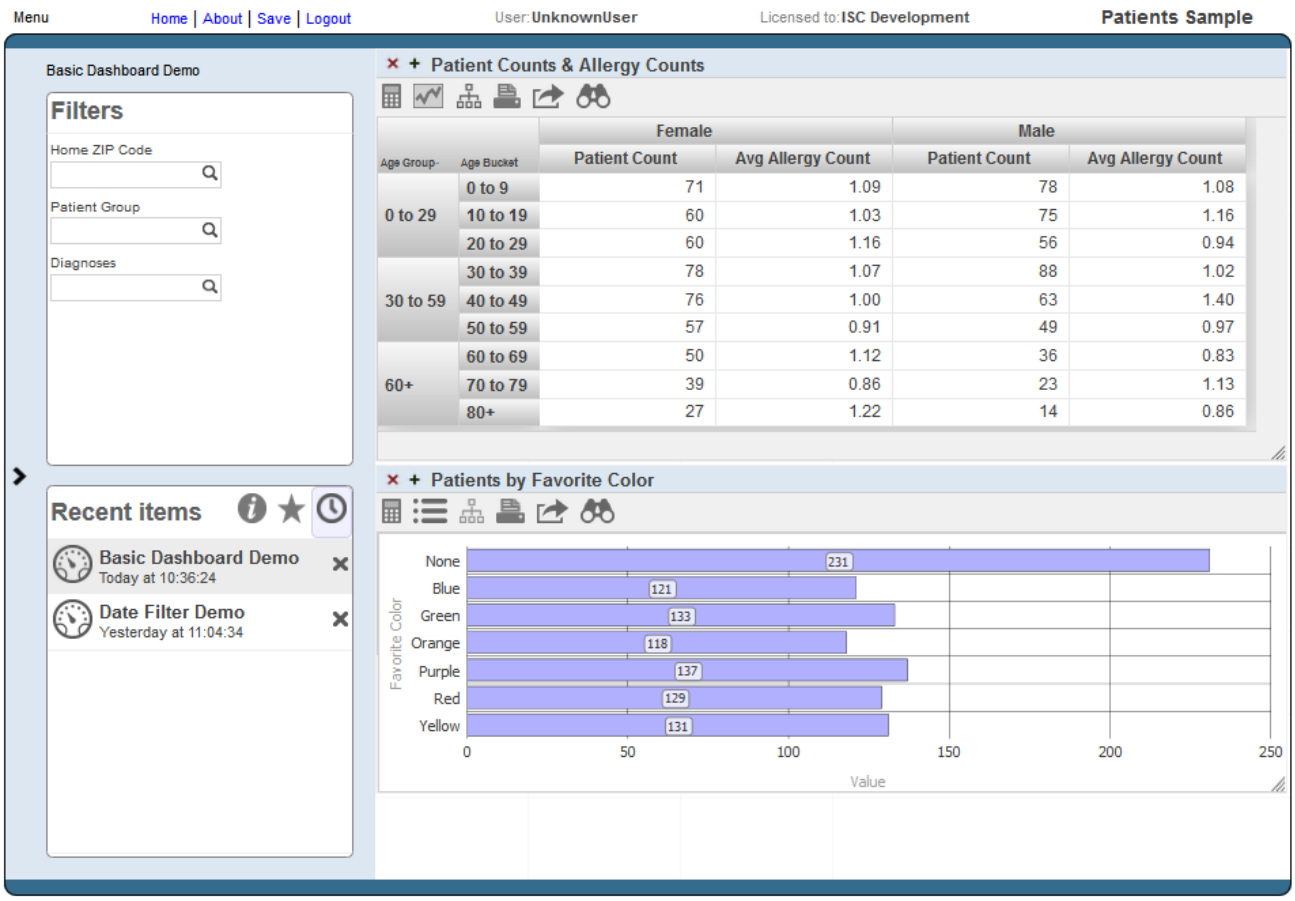
InterSystems IRIS Business Intelligence モデルの概要

ここでは、InterSystems IRIS® データ・プラットフォームの Business Intelligence モデルを紹介します。

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

InterSystems Business Intelligence は、ビジネス・インテリジェンス (BI) をアプリケーションに埋め込むことを可能にすることで、ユーザがデータに関する高度な質問をし、回答できるようにします。次の例のようなダッシュボードをアプリケーションに組み込むことができます。



ダッシュボードのウィジェットは、ピボット・テーブルおよび KPI (重要業績評価指標) に従って動作します。ピボット・テーブルでは、ソース値を表示するリストを表示できます。

ピボット・テーブル、KPI およびリストはクエリで、実行時に実行されます。

- ピボット・テーブルは、ユーザが行うフィルタ選択など、実行時の入力に応答できます。内部的には、キューブと通信を行う MDX (多次元式) クエリが使用されます。

  キューブは、ファクト・テーブルとインデックスで構成されます。ファクト・テーブルは、一連のファクト (行) で構成され、各ファクトは、ベース・レコードに対応します。例えば、ファクトを患者や診療科に対応させることができます。

  構成および実装に応じて、システムはトランザクション・テーブルの変更を検出し、必要に応じてファクト・テーブルに反映させます。

  ユーザがアナライザでピボット・テーブルを作成すると、システムは、MDX クエリを自動生成します。

- KPI も実行時のユーザ入力に反応させることができます。内部的には、MDX クエリ (キューブの場合)、または SQL クエリ (任意のテーブルの場合) を使用します。

  いずれの場合も、クエリは手動で作成することも、別の場所からコピーすることもできます。

- リストには、ユーザが選択したピボット・テーブルの行に使用されたソース・レコードから選択された値が表示されます。内部的には、リストは SQL クエリです。

  使用するフィールドを指定して、システムに実際のクエリを生成させることができます。また、クエリ全体を指定することもできます。

1.2 Business Intelligence モデルの概要

モデルには、以下の要素のいくつか、またはすべてを含めることができます。

- 1 つ以上のキューブ定義。キューブは、特定のベース要素のセット（患者、トランザクションなど）のクエリを実行する方法を記述します。キューブには、ベース・セットのレコードのグループ化を可能にするレベルと、これらのレコードの集約値を表示するメジャーが組み込まれています。また、リストおよびその他の項目も定義されます。

  ピボット・テーブルの作成時は、レベルとメジャーを使用します。次のようなピボット・テーブルがあるとします。

Patient Group	Avg Test Score
Group A	75.08
Group B	74.22
None	

  このピボット・テーブルで、行は `Patient Group` レベルのメンバに対応します。メンバのそれぞれが 1 行に表示されます。データ列には、メンバそれぞれの `Avg Test Score` メジャーの集約値が表示されます。このメジャーに対する平均値はシステムで計算されています。`None` 患者グループに対する `Avg Test Score` が NULL であることがわかります。

- 任意の数のサブジェクト領域。サブジェクト領域は、複数のキューブを必要とせずに、より小さなデータセットに焦点を当てることのできるサブキューブです。また、サブジェクト領域を使用すると、キューブのキャプションおよび既定値をカスタマイズできます。

- 任意の数の KPI。

  Business Intelligence の KPI は、ダッシュボードに表示できるインタラクティブなデータ・セットです。KPI では、ユーザが起動でき、ユーザのカスタム・コードを実行するアクションを定義できます。（アクションについては、"カスタム・アクションの定義" を参照してください。）

これらの要素を使用して、以下のようにダッシュボードを作成できます。

- アナライザ内でピボット・テーブルを作成します。

  各ピボット・テーブルは、ドラッグ・アンド・ドロップ操作で作成可能なクエリです。このクエリは、キューブまたはサブジェクト領域に対して実行されます。

- ダッシュボード・デザイナー内では、ダッシュボードにピボット・テーブルおよび KPI を、フィルタ・コントロールおよびその他の必要なコントロールと共に追加します。

"基本概念" では、キューブおよびサブジェクト領域の概要について説明します。モデルには、多数の追加要素を含めることができます。オプションの完全な比較は、"モデル・オプションの概要" を参照してください。

1.3 モデル開発プロセスの概要

一般的なモデル開発プロセスは、以下のとおりです。

1. 数項目のみの基本キューブを作成、コンパイルおよび構築します。
2. アナライザまたは Business Intelligence シェルを使用して、結果を検証し、必要な変更を特定します。

   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。

   シェルの詳細は、"InterSystems Business Intelligence の概要" を参照してください。

3. 必要に応じて前述の手順を繰り返します。
4. キューブが完成したら、または完成に近づいたら、キューブに基づいてサブジェクト領域を定義します。

複数のキューブが必要な場合もあります。

1.4 アーキテクトの概要

アーキテクトは、キューブおよびサブジェクト領域の作成に使用します。

アーキテクトにアクセスする手順は以下のとおりです。

1. [InterSystems ランチャー] をクリックし、**[管理ポータル]** をクリックします。

   セキュリティの設定によっては、InterSystems IRIS® データ・プラットフォームのユーザ名とパスワードを使用してログインするように求められます。

2. 以下のように、適切なネームスペースに切り替えます。

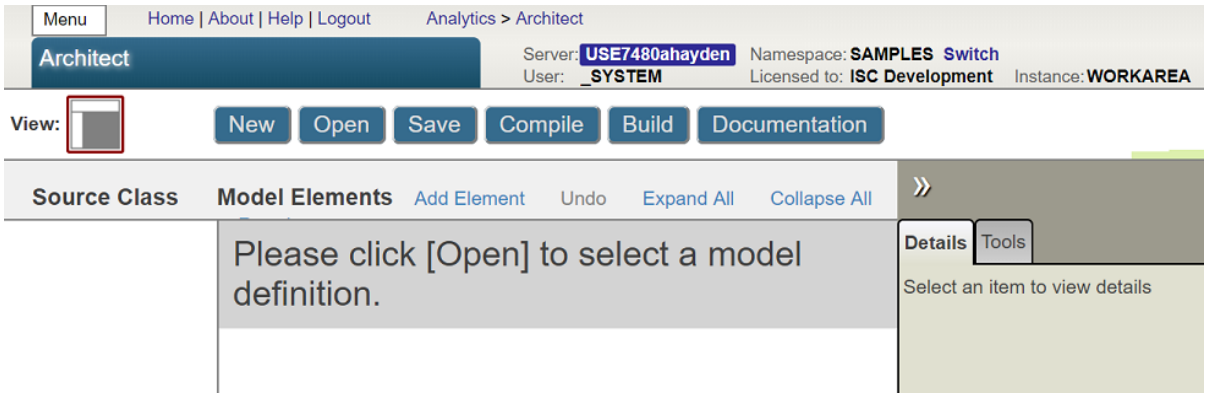
   a. 現在のネームスペースの名前をクリックして、使用可能なネームスペースのリストを表示します。

   b. リストから、該当するネームスペースをクリックします。

   注釈　特定のインターシステムズ製品では、HealthShare 製品と共に提供されている HSCUSTOM ネームスペースなど、カスタマイズされたクラス向けのすぐに使用可能なネームスペースが用意されています。このようなネームスペースにはキューブを導入しないことを強くお勧めします。

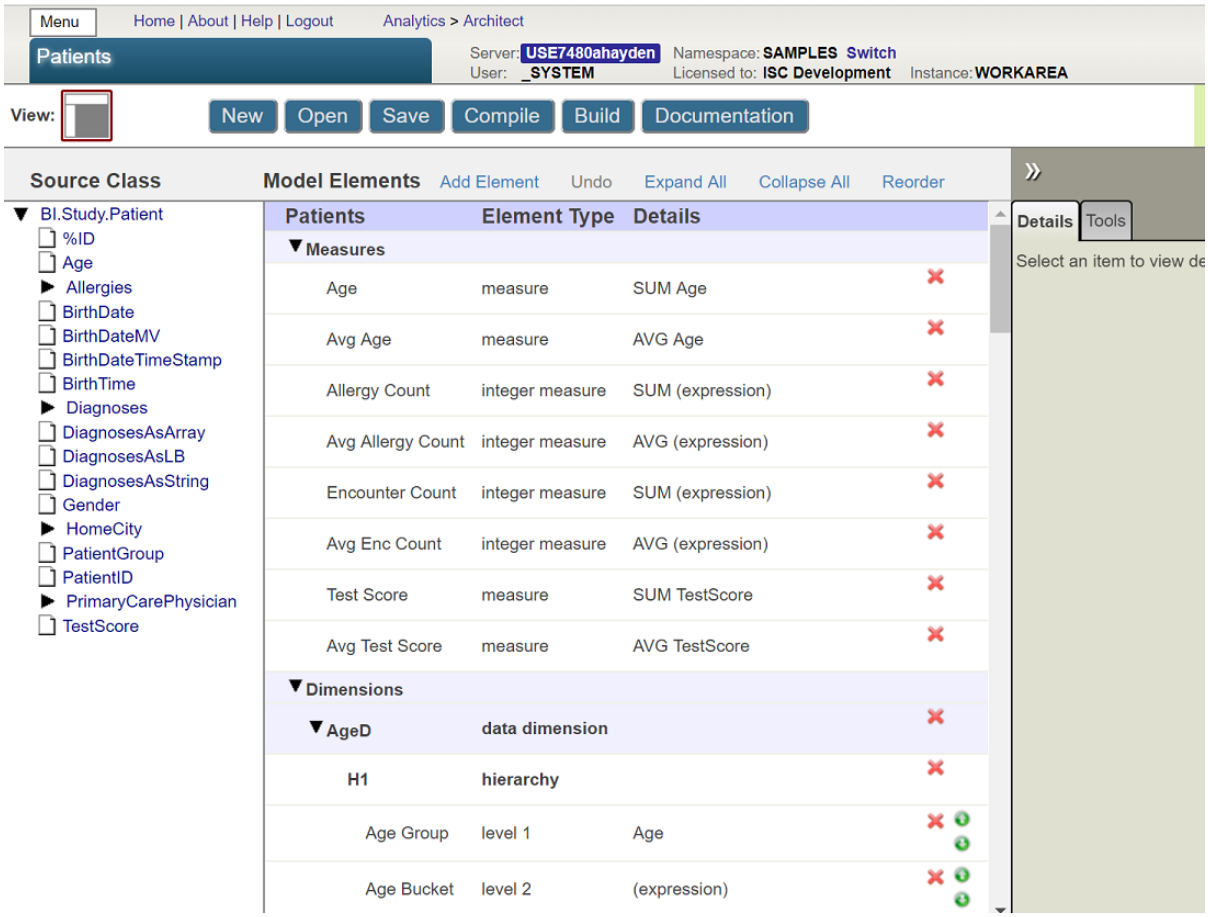
3. [Analytics] をクリックしてから **[アーキテクト]** をクリックします。

   アーキテクトの最初の表示は、以下のようになります。



4. **[開く]→[キューブ]** をクリックして、キューブの名前をクリックしから [OK] をクリックします。

   これで以下のように表示されます。

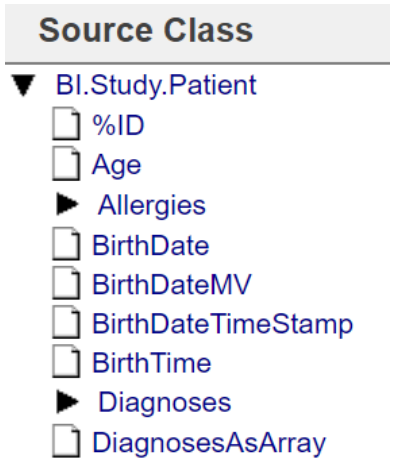


上部領域には、ナビゲーション・リンクと各種のタスクを実行するボタンがあります。

その下の部分のページは、アーキテクトでキューブを表示している場合、以下の領域で構成されます。

1.4.1 クラス・ビューワ

左の領域はクラス・ビューワです。ここにはキューブで使用されるベース・クラスのプロパティが表示されます。サブジェクト領域にはこの領域が表示されません。以下はその例です。



この領域はサイズ変更できます。そのためには、この領域の右端にある縦方向の区切りをドラッグします。

アーキテクトのクラスの表示は、以下の規則によって制御されます。

- リレーションシップ・プロパティ、プライベート・プロパティ、および一時プロパティを除くすべてのプロパティが表示されます。
- この表示は再帰的に使用されます。すなわち、プロパティのプロパティが表示されます。
- プロパティがコレクション (リストまたは配列) またはリレーションシップの場合は、コレクションまたはリレーションシップで使用されるクラスのプロパティを示すフォルダとして表示されます。
- プロパティが **%List** タイプ ($LISTBUILD と同等のオブジェクト) の場合、そのプロパティはフォルダとしては表示されません。

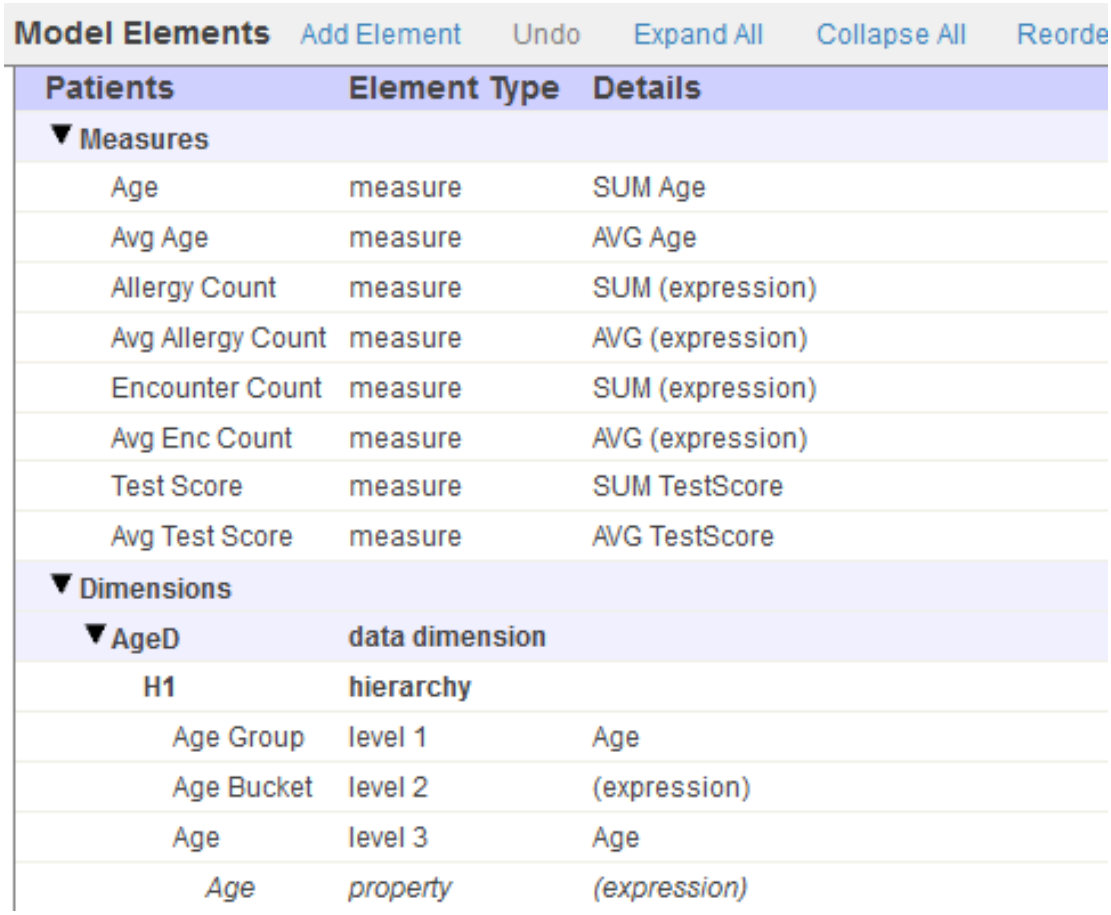
  これを示す例については、Patients サンプルに組み込まれている **DiagnosesAsLB** プロパティを参照してください。
- カスケード・ドット構文を使用してベース・クラスからクラスにアクセスできない場合、そのクラスは表示されません。
- アーキテクトでは、スーパークラスから継承されたプロパティが表示されます (これは、このサンプルには含まれていません)。

主要なキューブ要素はすべて、任意のクラスのプロパティを使用できるソース・プロパティまたはソース式 (ObjectScript 式) のいずれかに基づきます。

重要　アーキテクトは、クラス・プロパティの役に立つ表示を提供します。このことにより、Business Intelligence 要素をこれらのプロパティに基づいて作成するのが非常に簡単になります。ただし、この表示によって一部のプロパティにアクセスする便利な方法が提供されるとしても、ソース式を使用して任意のデータにアクセスすることも可能であるということを認識しておくことは重要です。これらのソース式は、キューブの構築時に評価されるため、実行時のパフォーマンスに影響しません。

1.4.2 モデル・ビューワ

中央の領域はモデル・ビューワで、キューブの現在のコンテンツが表示されます。以下はその例です。



Model Elements Add Element Undo Expand All Collapse All Reorde

Patients	Element Type	Details
▼ Measures		
Age	measure	SUM Age
Avg Age	measure	AVG Age
Allergy Count	measure	SUM (expression)
Avg Allergy Count	measure	AVG (expression)
Encounter Count	measure	SUM (expression)
Avg Enc Count	measure	AVG (expression)
Test Score	measure	SUM TestScore
Avg Test Score	measure	AVG TestScore
▼ Dimensions		
▼ AgeD	data dimension	
H1	hierarchy	
Age Group	level 1	Age
Age Bucket	level 2	(expression)
Age	level 3	Age
Age	*property*	*(expression)*

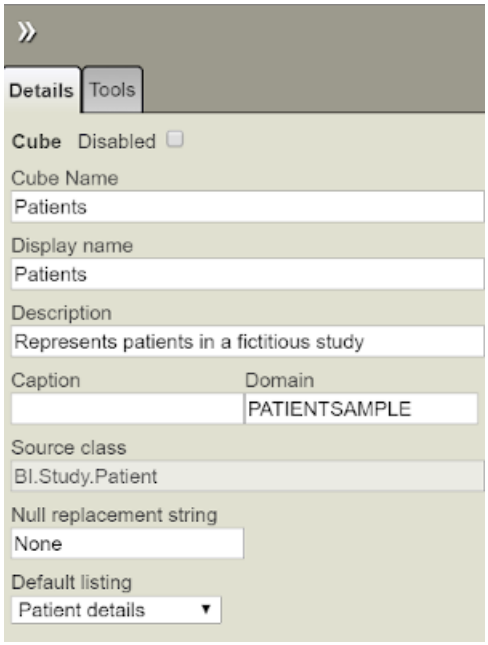
この領域はサイズ変更できます。そのためには、この領域の左端にある縦方向の区切りをドラッグします。

上部にある [**要素を追加**] リンクを使用すると、メジャーやディメンジョンなどの項目をキューブに追加できます。リンクの下の領域では編集対象の項目を選択できます。1 行目のキューブ自体も選択できます。また、項目の行内の [X] ボタンをクリックして、項目を削除することもできます。

1.4.3 詳細領域

右の領域は、詳細領域で、モデル・ビューワで現在選択されている要素の詳細 (選択されている場合)、またはキューブの詳細 (選択されていない場合) が表示されます。

以下はその例です。



この領域を非表示にするには、[詳細を隠す] ボタン をクリックします。詳細を非表示にすると、モデル・ビューワの領域が拡大します。

この領域を再表示するには、[詳細を表示] ボタン をクリックします。

この領域では、主に **[詳細]** タブを使用します。

1.5 サンプルのアクセス方法

このドキュメントのほとんどの例は、Samples-BI サンプル (https://github.com/intersystems/Samples-BI) または Samples-Aviation サンプル (https://github.com/intersystems/Samples-Aviation) の一部です。

サンプルは、専用のネームスペース (SAMPLES など) を作成して、そのネームスペースにロードすることをお勧めします。一般的な手順は、"InterSystems IRIS で使用するサンプルのダウンロード" を参照してください。

2

基本概念

ここでは、InterSystems IRIS® データ・プラットフォームの Business Intelligence モデルで最も重要な概念であるキューブ、サブジェクト領域、およびそのコンテンツについて説明します。

モデルには、"InterSystems Business Intelligence の上級モデリング" で説明している多数の追加要素を含めることができます。オプションの完全な比較は、"モデル・オプションの概要" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

2.1 キューブの概要

キューブは、MDX の概念であり、アナライザで使用される MDX 要素を定義します。これらの要素によって、データ、具体的には特定のレコード (患者レコード、トランザクション・レコードなど) に対するクエリの実行方法が決定されます。レコードのセットは、キューブのソース・クラスによって決まります。

キューブには、以下のすべての定義を組み込むことができます。

- レベル：レコードのグループ化を可能にする。
- 階層：レベルを格納。
- ディメンジョン：階層を格納。
- レベル・プロパティ：レベルのメンバに固有の値。
- メジャー：レコードの集約値を表示。
- リスト：ソース・データへのアクセスを可能にするクエリ。
- 計算メンバ：他のメンバに基づくメンバ。
- 名前付きセット：メンバおよびその他の MDX 要素の再使用可能なセット。

これらの項目の多くについては、後続のセクションで説明します。名前付きセットの詳細は、"InterSystems MDX の使用法" を参照してください。

2.1.1 キューブのソース・クラス

ほとんどの場合、キューブのソース・クラスは永続クラスです。

データ・コネクタもソース・クラスになる可能性があります。データ・コネクタは、%DeepSee.DataConnector を拡張するクラスです。データ・コネクタは、任意の SQL クエリの結果を、キューブのソースとして使用可能なオブジェクトにマップします。通常、データ・コネクタは InterSystems データベース内に存在しない外部データにアクセスしますが、InterSystems

データベースに対する SQL クエリ (ビューに対する SQL クエリを含む) を指定するために使用することもできます。"データ・コネクタの定義と使用" を参照してください。

子コレクション・クラスもソース・クラスとして使用できます。

2.2 ディメンジョン、階層およびレベル

ここでは、ディメンジョン、階層およびレベルについて説明します。

2.2.1 レベルとメンバ

レベルはメンバで構成され、メンバはレコードのセットです。`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、キューブ内の各レコードは、1 つ以上のメンバに属します。

ピボット・テーブルの多くは、単純に 1 つ以上のレベルのメンバのデータを表示します。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。次のピボット・テーブルは、これらのメンバのデータを示しています。

Age Group	Patient Count
0 to 29	4,255
30 to 59	4,119
60+	1,626

別の例として、以下のピボット・テーブルは、`Age Group` と `Gender` レベルのデータを示しています (それぞれ 1 列目と 2 列目に示されています)。

Age Group		Patient Count
0 to 29	Female	2,073
	Male	2,182
30 to 59	Female	2,030
	Male	2,089
60+	Female	939
	Male	687

個別のメンバをドラッグ・アンド・ドロップして行または列に使用できます。以下はその例です。

Gender	Patient Count
Female	5,115
Male	4,885
1910s	78
Elm Heights	1,138

オプションの詳細は、"アナライザの使用法" を参照してください。

2.2.2 メンバ名とキー

各メンバには、名前と内部キーの両方があります。コンパイラでは、同じレベル内にあっても、これらのいずれかが一意である必要はありません。状況によってはメンバ名に重複があることが正当な場合もありますが、メンバ・キーは重複しないように注意してください。

メンバ名の重複が正当である例として、同名異人を単一のメンバに結合しない場合を考えてみます。ユーザがメンバをドラッグ・アンド・ドロップすると、システムで生成されるクエリでは、名前でなくメンバ・キーが使用されるため、常に目的のメンバにアクセスできます。

ただし、メンバ・キーの重複があると、個別メンバの参照が不可能になります。メンバ・キーを指定すると、システムはそのキーを持つ最初のメンバのみを返します。モデルのメンバ・キーは重複しないように確認することができ、またそうする必要があります。

重複メンバ名および重複メンバ・キーの発生するシナリオの詳細は、"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。

2.2.3 ソース値

各レベルは、ソース値に基づいています。この値は、クラス・プロパティまたは ObjectScript 式のいずれかになります。例えば、`Gender` レベルは、患者の `Gender` プロパティに基づきます。別の例として、`Age Group` レベルは、年齢に応じて患者の `Age` プロパティを文字列 (`0 to 29`、`30 to 59`、または `60+`) に変換する式に基づいています。

2.2.4 階層およびディメンジョン

Business Intelligence 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、市区町村を郵便番号でグループ化したり、郵便番号を地方でグループ化できます。

Business Intelligence の階層の定義には、次に示す 3 つの実用的な理由があります。

- システムには、階層を利用した最適化機能があります。例えば、年を期間の親として定義しているモデルで、期間 (年および月) を行または列として表示し、フィルタ処理して特定の年に絞り込むと、クエリの速度が向上します。
- ピボット・テーブル内では、レベルのメンバをダブルクリックすると、システムでドリルダウンが実行され、そのメンバに子メンバが存在する場合は、その子メンバが表示される、というように階層を使用できます。例えば、年をダブルクリックすると、システムはその年に含まれる期間にドリルダウンします。詳細は、"アナライザの使用法" を参照してください。
- MDX には、階層で使用できる関数があります。例えば、特定の国の子郵便番号に対するクエリや、同じ国内の別の郵便番号に対するクエリを実行できます。

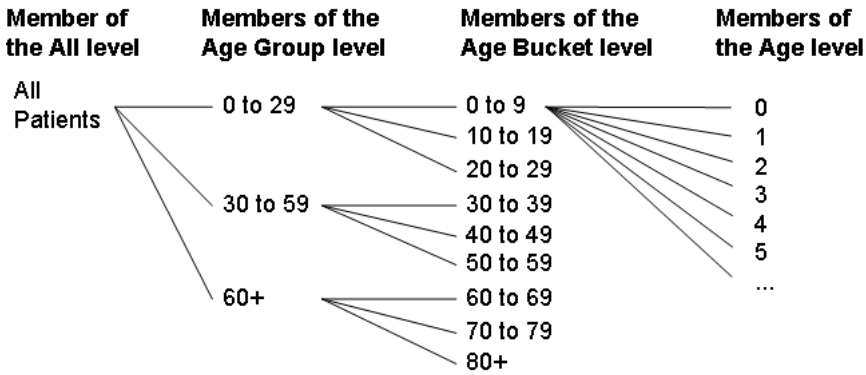
  これらの関数は、手書きのクエリで使用できます。アナライザでは、ドラッグ・アンド・ドロップでこのようなクエリを作成できません。

ディメンジョンには、レコードを類似の方法で組織化する 1 つ以上の親子階層があります。例えば、アレルギーに関連する複数の階層を 1 つのディメンジョンに包含できます。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。ディメンジョンの実際的な用途は、ディメンジョンに格納されるレベル、特に All レベルの既定の動作を定義することです。これについては次のサブセクションで説明します。

2.2.5 All レベルと All メンバ

各ディメンジョンでは、そのディメンジョン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。

例えば、`AgeD` ディメンジョンに、下図に示すようなレベルで構成される 1 つの階層が組み込まれます。



特定のディメンジョンについて、All メンバがその論理名と表示名と共に存在しているかどうかを指定します。このディメンジョン内の All メンバは `All Patients` と命名されています。

2.2.6 リスト・ベースのレベル

Business Intelligence では、他の多くの BI ツールと異なり、レベルのベースにリスト値を使用できます。例えば、1 人の患者に複数の診断が存在する場合があります。`Diagnoses` レベルで患者を診断別にグループ化します。以下はその例です。

Diagnoses	Patient Count
None	8,480
asthma	660
CHD	319
diabetes	508
osteoporosis	217
Total	10,184

リスト・ベースのレベルでは、特定のソース・レコードが複数のメンバに属する場合があります。ここに示すピボット・テーブルでは、一部の患者が複数回含まれています。

2.2.7 関連項目

"ディメンジョン、階層およびレベルの定義" を参照してください。

2.3 プロパティ

各レベルで、任意の数のプロパティを定義できます。レベルにプロパティが設定されていると、そのレベルのメンバそれぞれにそのプロパティ値が存在します。その他のレベルにはそのプロパティの値が存在しません。

Patients サンプルで、`City` レベルにプロパティ `Population` と `Principal Export` が含まれているとします。

各プロパティは、ソース値に基づきます。この値は、クラス・プロパティまたは ObjectScript 式のいずれかになります。`City` レベルでは、プロパティ `Population` と `Principal Export` は、直接クラス・プロパティに基づいています。

クエリでは、メジャーを使用する場合とほとんど同様に、プロパティを使用できます。例えば、アナライザではプロパティを列として使用できます (この例では 2 つのプロパティを示しています)。

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

ただし、メジャーとは異なり、プロパティは集約できません。プロパティは、属するレベル以外のすべてのレベルで NULL になります。

"プロパティの定義" を参照してください。

2.4 メジャー

キューブは、ピボット・テーブルのデータ・セルに集約値を示すメジャーも定義します。

各メジャーは、ソース値に基づきます。これは、クラス・プロパティまたは ObjectScript 式のいずれかです。例えば、`Avg Test Score` メジャーは、患者の `TestScore` プロパティに基づきます。

メジャーの定義には、集約関数も組み込まれ、これによってそのメジャーの値の集約方法が指定されます。関数には、SUM および AVG が含まれます。

例えば、次のピボット・テーブルは、`Patient Count` メジャーと `Avg Test Score` メジャーを示しています。`Patient Count` メジャーは、任意のコンテキストで使用される患者をカウントし、`Avg Test Score` メジャーは、任意のコンテキストで使用される患者のテスト・スコアの平均値を示します。このピボット・テーブルは、`Age Group` レベルのメンバに関する、これらのメジャーの値を示します。

Age Group	Patient Count	Avg Test Score
0 to 29	4,255	59.79
30 to 59	4,119	59.83
60+	1,626	60.09

"メジャーの定義" を参照してください。

2.5 リスト

キューブには、リストも含めることができます。各リストは名前を持ち、ユーザがそのリストを要求したときに表示されるフィールドを指定します。以下に例を示します。

#	PatientID	Age	Gender	Home City	Test Score
1	SUBJ_100524	30	F	Elm Heights	73
2	SUBJ_101001	30	F	Juniper	91
3	SUBJ_101014	30	F	Elm Heights	87
4	SUBJ_101072	30	F	Redwood	70
5	SUBJ_101331	30	F	Cedar Falls	

表示されるレコードは、ユーザがリストを要求する際のコンテキストによって異なります。

キューブでリストを定義しない場合にユーザがリストを要求すると、アナライザでは以下のメッセージが表示されます。

```
Error #5001: %ExecuteListing: this cube does not support drill through
```

"リストの定義" を参照してください。

以下のことも可能です。

- キューブでは個別のリスト・フィールドを定義できます。ユーザはアナライザでこれらのリスト・フィールドを使用してカスタム・リストを作成できます。"リスト・フィールドの定義" を参照してください。
- キューブ定義の外部で (アーキテクトにアクセスすることなく) リストを定義できます。"リスト・グループの定義" を参照してください。

2.6 計算メンバ

計算メンバは、他のメンバに基づきます。以下の 2 種類の計算メンバを定義できます。

- 計算メジャーは、他のメジャーに基づくメジャーです(MDX では、各メジャーは Measures ディメンジョンのメンバです)。

  例えば、あるメジャーを、第 2 のメジャーをさらに第 3 のメジャーで除算した結果として定義することができます。

  計算メジャーという語句は、MDX で標準的な語句ではありませんが、このドキュメントでは簡潔にするためにこれを使用します。
- 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。

計算メンバは、基にするメンバの後で評価されます。

キューブ定義では、いずれの種類の計算メンバも作成でき、アナライザでは、いずれの種類の計算メンバも追加作成できます。

2.6.1 計算メジャー

他のメジャーに基づいて新規メジャーを定義することは大変便利です。例えば、Patients サンプルの場合、`Allergy Count` メジャーを `Count` メジャーで除算した結果として `Avg Allergy Count` メジャーを定義することができます。次のようなピボット・テーブルがあるとします。

Allergy Severities	Count	Allergy Count	Avg Allergy Count
Minor	1,167	1,909	1.64
Moderate	1,114	1,847	1.66
Life-threatening	1,201	1,939	1.61
Inactive	1,143	1,805	1.58
Unable to determine	1,101	1,780	1.62
No Data Available	4,000		
Nil known allergies	1,482	0	0

このピボット・テーブルを実行すると、Allergy Severities レベルのメンバごとに Count メジャーと Allergy Count メジャーの値が決定されます。このページの後続のセクションでは、この動作の内容について説明します。その後、各メンバの Avg Allergy Count 値を求めるため、Allergy Count 値が Count 値で除算されます。

2.6.2 非メジャーの計算メンバ

非メジャーの計算メンバは、MDX 集約関数を使用して他の非メジャーのメンバと結合します。最も便利な関数は %OR です。

非メジャー・メンバのそれぞれが一連のレコードを参照することを忘れないでください。複数のメンバを新規メンバに結合する際は、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作成します。

簡単な例として、ColorD 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、メンバ Red、Yellow、および Blue が組み込まれています。これらのメンバは、赤、黄、青をそれぞれ好む患者にアクセスします。%OR を使用して、3 つの患者グループすべてにアクセスする新規メンバを 1 つ作成することができます。

以下はその例です。

Primary Colors	Patient Count
Primary Colors	3,765

2.6.3 関連項目

"計算メンバの定義" を参照してください。

2.7 サブジェクト領域

サブジェクト領域は、オプションによる項目名のオーバーライドを持つサブキューブです。サブジェクト領域を定義すると、複数のキューブを構築することなく、より小さなデータ・セットに焦点を当てることができます。サブジェクト領域では、以下の操作を実行できます。

- サブジェクト領域で使用可能なデータを制限するフィルタを指定します。フィルタの詳細は、次のセクションを参照してください。

  このフィルタはハードコードすることも、プログラムによって指定することもできます。これは、例えばユーザの $roles に基づいてこれを指定できることを意味します。
- キューブに定義されている要素を非表示にし、これらのサブセットがアナライザで表示されるようにします。
- 表示する要素に新しい名前、キャプション、および説明を定義します。

- サブジェクト領域の既定のリストを指定します。
- キューブで定義されたリストを再定義または非表示にします。
- 新規リストを定義します。

これで、キューブの使用が可能な場所と同じすべての場所でこのサブジェクト領域を使用できます。例えば、このサブジェクト領域をアナライザで使用して、この領域を対象とする MDX クエリをシェルで、または API を使用して実行できます。

"サブジェクト領域の定義" を参照してください。

2.8 フィルタ

BI アプリケーションでは、ピボット・テーブルおよびその他の場所でデータのフィルタ処理が可能であることが非常に重要です。ここでは、Business Intelligence のフィルタ・メカニズムとアプリケーションでの使用方法について説明します。

2.8.1 フィルタ・メカニズム

データのフィルタ処理のために、メンバ・ベース・フィルタとメジャー・ベース・フィルタという 2 つの簡単な方法が用意されています。これらを組み合わせてより複雑なフィルタを作成することもできます。これは特に MDX クエリを直接作成する際に使用できます。

2.8.1.1 メンバ・ベース・フィルタ

メンバはレコードのセットです。最も単純なメンバ・ベース・フィルタでは、メンバを使用して、ピボット・テーブルをフィルタ処理します (例えば、このセクションで後述するように他のコンテキストを使用できます)。つまり、ピボット・テーブルはそのメンバに属するレコードのみにアクセスすることになります。

例えば、次のようなピボット・テーブルがアナライザで表示されるとします。

Age Group	Patient Count
0 to 29	4,364
30 to 59	4,036
60+	1,600
Total	10,000

`Age Group` レベルの `0 to 29` メンバを使用するフィルタを適用するとします。フィルタ適用後のピボット・テーブルは次のようになります。

Age Group: 0 to 29

Age Group	Patient Count
0 to 29	4,364
Total	4,364

アナライザには、NULL の行と列を表示するオプションがあります。NULL 行を表示すると、ピボット・テーブルは次のようになります。

Age Group	0 to 29
Age Group	Patient Count
0 to 29	4,364
30 to 59	
60+	
Total	4,364

どのピボット・テーブルでも同じフィルタを使用できます。例えば、フィルタ処理されていない次のようなピボット・テーブルがあるとします。

Favorite Color	Female	Male	Total
None	1,187	1,220	2,407
Blue	669	566	1,235
Green	675	642	1,317
Orange	658	598	1,256
Purple	639	616	1,255
Red	665	606	1,271
Yellow	622	637	1,259
Total	5,115	4,885	10,000

見出しには示されていませんが、このピボット・テーブルには、`Patient Count` メジャーが表示されます。このピボット・テーブルに前と同様にフィルタを適用すると、次のように表示されます。

Age Group	0 to 29		
Favorite Color	Female	Male	Total
None	494	579	1,073
Blue	263	257	520
Green	261	294	555
Orange	260	256	516
Purple	270	273	543
Red	310	280	590
Yellow	268	299	567
Total	2,126	2,238	4,364

いずれの場合も、合計レコード数は同じであることに注意してください。どちらの場合も、`0 to 29` メンバに属する患者にのみアクセスしています。

また、1 つのフィルタで複数のメンバをまとめて使用することも、異なるレベルのメンバを参照する複数のフィルタを結合することもできます。また、含めるメンバを選択するのではなく、除外するメンバを選択することもできます。

Tipヒント　メンバ・ベース・フィルタは作成が容易でかつ強力であるため、フィルタでの使用専用のレベルを作成するだけの価値があります。

2.8.1.2 メジャー・ベース・フィルタ

システムは、検索可能メジャーをサポートしています。このメジャーを使用すると、ソース・レコード自体のレベルで値を評価するフィルタを適用できます。

Patients サンプルに対して、受診回数が 10 回以上の患者のみにアクセスするフィルタを設定できます。このフィルタをピボット・テーブルで使用すると、以下のようになります。

Age Group	Patient Count	Total
0 to 29	3,246	3,246
30 to 59	3,305	3,305
60+	1,330	1,330
Total	7,881	7,881

同じフィルタを別のピボット・テーブルで使用すると、以下のようになります。

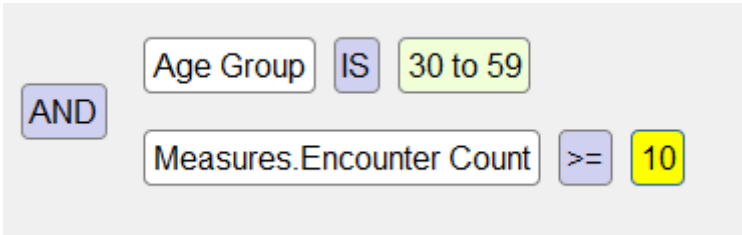
Favorite Color	Female	Male	Total
None	926	973	1,899
Blue	537	451	988
Green	530	523	1,053
Orange	515	454	969
Purple	495	486	981
Red	533	485	1,018
Yellow	469	504	973
Total	4,005	3,876	7,881

いずれの場合も、ピボット・テーブルでは受診回数が 10 回以上の患者のみが使用されるため、患者の合計数は同じです。

検索可能メジャーにはテキスト値が含まれることもあります。このようなメジャーでは、=、< >、および LIKE 演算子を使用してレコードをフィルタ処理できます。

2.8.1.3 より複雑なフィルタ

メンバ・ベース・フィルタとメジャー・ベース・フィルタを結合して、より複雑なフィルタも作成できます。以下は、アナライザで作成したフィルタの例です。



内部的には、クエリでは AND や OR が使用されるのではなく、MDX 構文が使用されます。すべての Business Intelligence フィルタは MDX 構文を使用します。

MDX 関数を使用するフィルタも作成できます。以下はその例です。

- FILTER 関数は、メジャー・ベースのフィルタが使用する最下位レベルの値ではなく、メジャーの集約値を使用します。例えば、これを使用して、患者数が 1000 未満の市区町村に属する患者をフィルタで除外することができます。

  アナライザでは、行または列に対する Levels オプションでこの関数が内部的に使用されています。

- EXCEPT 関数は、特定のメンバの削除に使用できます。選択したメンバを除外するメンバ・ベース・フィルタを作成する際に、この関数がシステムによって使用されます。

  InterSystems MDX には、このほかにも、SET 演算を実行する関数が多数用意されています。

- TOPCOUNT およびその他の関数は、ランキングに基づいてメンバにアクセスします。

MDX の概要およびオプションの概要は、"InterSystems MDX の使用法" を参照してください。"InterSystems MDX リファレンス" も参照してください。

2.8.2 フィルタの使用

ピボット・テーブルの定義時には、フィルタ方法を指定できます。ただし実際は、ピボット・テーブルの管理が困難になるため、フィルタの異なる類似した複数のピボット・テーブルを作成することは推奨されません。この代わりに、以下のツールのいずれかまたはすべてを使用できます。

- アナライザで、名前付きフィルタを定義し、それを複数のピボット・テーブルで使用できます。名前付きフィルタは、アナライザでキューブまたはサブジェクト領域のコンテンツと共に使用できます (次の項目を参照してください)。
- アーキテクトでは、ベース・キューブのフィルタ適用済みビューであるサブジェクト領域を定義できます。この定義後にピボット・テーブルを作成し、キューブ自体ではなくサブジェクト領域から開始します。これらのピボット・テーブルは、ピボット・テーブル自体に固有のフィルタに加え、サブジェクト領域フィルタによって常にフィルタ処理されます。

  サブジェクト領域では、ハードコードされたフィルタを指定したり、コールバック・メソッドをカスタマイズして実行時にフィルタを (例えば、$roles などの値に基づくように) 指定できます。
- ユーザ・ポータルでは、ダッシュボードの作成時にフィルタ・コントロールを組み込むことができます (これは単純なメンバ・ベース・フィルタの場合にのみ適用されます)。その後、ユーザは包含または除外する 1 つ以上のメンバを選択できます。

フィルタは常に累積的です。

2.9 システムによるファクト・テーブルの構築方法と使用方法

キューブのコンパイル時に、システムは、ファクト・テーブルと関連テーブルを生成します。キューブの構築時に、システムは、これらのテーブルに値を入力して、インデックスを生成します。実行時に、システムは、これらのテーブルを使用します。ここでは、このプロセスについて説明しています。ここでは、以下のトピックについて説明します。

- ファクト・テーブルの構造
- システムによるファクト・テーブルへのデータ入力方法
- システムによるファクト・テーブルの使用方法

システムは、サブジェクト領域に対するテーブルを生成しません。サブジェクト領域は、サブジェクト領域の基となるキューブに対して生成されたテーブルを使用します。

2.9.1 ファクト・テーブルの構造

一般にファクト・テーブルには、ベース・テーブルのレコードごとに 1 つのレコードが存在します。この行がファクトです。ファクトには、レベルおよびメジャーごとに、それぞれ 1 つのフィールドが含まれます。以下は、部分的な例です。

Age	Allergies	Home City	Home ZIP	...	Test Score	meas B	...
42	dairy products,eggs	Juniper	32006	*nnnnn*	78	*nnnnn*	*nnnnn*
7		Cypress	34577	*nnnnn*	53	*nnnnn*	*nnnnn*
52		Elm Heights	38928	*nnnnn*	89	*nnnnn*	*nnnnn*
13	ant bites,bee stings,soy	Pine	34577	*nnnnn*	61	*nnnnn*	*nnnnn*
8		Juniper	32006	*nnnnn*	90	*nnnnn*	*nnnnn*
12	dairy	Elm Heights	38928	*nnnnn*	67	*nnnnn*	*nnnnn*
34	ant bites,dairy products	Centerville	36711	*nnnnn*	74	*nnnnn*	*nnnnn*
26	ant bites,bee stings,eggs	Magnolia	34577	*nnnnn*	89	*nnnnn*	*nnnnn*
89	ant bites	Spruce	32006	*nnnnn*	84	*nnnnn*	*nnnnn*
22	soy	Spruce	32006	*nnnnn*	59	*nnnnn*	*nnnnn*
...							

指定されたレベルのフィールドには、値が含まれていない場合もあれば、値が 1 つ、または複数含まれている場合もあります。個別値はそれぞれ、このレベルのメンバに対応します。任意のメジャーに対するフィールドには NULL 値、または単一値が含まれます。

システムはこのファクト・テーブルを構築するときに、対応するインデックスも生成します。

ファクト・テーブルには、階層およびディメンジョンに関する情報は含まれません。ファクト・テーブルでは、レベル間のリレーションシップに関係なく、各レベルが同様に処理されます。ファクト・テーブルには各レベルに列が 1 つ含まれ、その列には各ソース・レコードに適用される 1 つ以上の値が含まれます。

Tipヒント 既定で、ファクト・テーブルには、ベース・テーブルと同じ数の行が存在します。スタジオでキューブ・クラスを編集する際は、OnProcessFact() コールバックをオーバーライドできます。これにより、ベース・テーブルの選択行を無視できます。このようにした場合、ファクト・テーブルの行がベース・テーブルより少なくなります。

2.9.2 ファクト・テーブルへのデータ入力

キューブの完全構築の実行時に、システムはベース・テーブルのレコードを繰り返し処理します。各レコードに対して以下の処理が実行されます。

- 各レベルの定義を調査し、0 個、1 個または複数の値を取得します。

  この手順では、システムによって、レコードの分類方法が決定されます。

- 各メジャーの定義を調査し、0 個、または 1 個の値を取得します。

その後、このデータがファクト・テーブルの対応する行に書き込まれ、インデックスが適切に更新されます。

選択的構築の実行時に、システムは既存のファクト・テーブルを繰り返し処理し、選択された列を更新するか、この列に値を入力します。

2.9.2.1 レベルに対する値の決定

各レベルは、ソース・プロパティ、またはソース式として指定されます。ソース式の大部分は指定のレコードに対して単一の値を返しますが、タイプがリストのレベルの場合、その値は複数の値のリストです。

ベース・テーブル内の特定のレコードについて、システムは構築時にそのプロパティまたは式を評価し、対応する値をファクト・テーブルに格納し、インデックスを適宜更新します。

例えば、`Age Bucket` レベルを、`0-9`、`10-19`、`20-29` などの文字列のいずれか 1 つを返す式として定義します。返される値は、患者の年齢によって異なります。システムは返された値をファクト・テーブルの `Age Bucket` レベルに対応するフィールドに書き込みます。

別の例の場合、`Allergy` レベルは患者の複数のアレルギーのリストです。

2.9.2.2 メジャーに対する値の決定

システムはファクト・テーブルを構築するときに、メジャーの値も決定して格納します。各メジャーは、ソース・プロパティ、または ObjectScript ソース式として指定されます。

ベース・テーブルの指定の行について、システムはメジャーの定義を確認および評価し、値があれば、その値を適切なメジャー・フィールドに格納します。

例えば、Test Score メジャーは、患者の TestScore プロパティに基づいています。

2.9.2.3 プロパティに対する値の決定

システムはファクト・テーブルを構築するときに、プロパティに対する値も決定しますが、この値をファクト・テーブルには格納しません。ファクト・テーブル以外に、システムは各レベルのテーブルも生成します (一部の例外があります。"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください)。システムは、ファクト・テーブルを構築するとき、適切なディメンジョンのテーブルにプロパティの値を格納します。

2.9.3 ファクト・テーブルの使用法

次のようなピボット・テーブルがあるとします。

Age Bucket	Count	Test Score	Avg Test Score
0 to 9	1,410	82,860	58.77
10 to 19	1,471	88,229	59.98
20 to 29	1,374	83,297	60.62
30 to 39	1,529	90,769	59.36
40 to 49	1,487	89,755	60.36
50 to 59	1,103	65,909	59.75
60 to 69	745	44,056	59.14
70 to 79	569	34,698	60.98
80+	312	18,960	60.77

最初の列には Age Bucket レベルのメンバの名前が表示されます。最初のデータ列には Patient Count メジャー、2 番目の列には Test Score メジャー、最後の列には Avg Test Score メジャーが表示されます。Avg Test Score メジャーは計算メンバです。

これらの値は、次のようにして決定されます。

1. 最初の行は、Age Bucket レベルの 0-9 メンバを参照しています。システムでは、インデックスを使用して、ファクト・テーブル内の関連する患者 (ここでは赤でハイライト表示) がすべて検出されます。

Age Bucket	Allergies	level C	...	Test Score	meas B	meas C	...
40 to 49	eggs	nnnnn	nnnnn	84	nnnnn	nnnnn	nnnnn
0 to 9		nnnnn	nnnnn	53	nnnnn	nnnnn	nnnnn
50 to 59		nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
10 to 19	ant bites,bee stings,soy	nnnnn	nnnnn	61	nnnnn	nnnnn	nnnnn
0 to 9		nnnnn	nnnnn	90	nnnnn	nnnnn	nnnnn
10 to 19	dairy	nnnnn	nnnnn	67	nnnnn	nnnnn	nnnnn
30 to 39	ant bites,dairy products	nnnnn	nnnnn	74	nnnnn	nnnnn	nnnnn
20 to 29	ant bites,bee stings,eggs	nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
80 +	ant bites	nnnnn	nnnnn	84	nnnnn	nnnnn	nnnnn
20 to 29	soy	nnnnn	nnnnn	59	nnnnn	nnnnn	nnnnn
...							

2. ピボット・テーブルの Patient Count 列には、特定のコンテキストで使用される患者数が表示されます。

この列の最初のセルでは、0-9 メンバに対して検出されたファクト・テーブル内のレコードの数がカウントされます。

3. ピボット・テーブルの Test Score 列には、特定のコンテキストにおける患者の累積テスト・スコアが表示されます。

   この列の最初のセルに対して、システムは、まず、0-9 メンバに対して検出されたファクト・テーブルで Test Score の値を検索します。

Age Bucket	Allergies	level C	...	Test Score	meas B	meas C	...
40 to 49	eggs	nnnnn	nnnnn	84	nnnnn	nnnnn	nnnnn
0 to 9		nnnnn	nnnnn	53	nnnnn	nnnnn	nnnnn
50 to 59		nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
10 to 19	ant bites,bee stings,soy	nnnnn	nnnnn	61	nnnnn	nnnnn	nnnnn
0 to 9		nnnnn	nnnnn	90	nnnnn	nnnnn	nnnnn
10 to 19	dairy	nnnnn	nnnnn	67	nnnnn	nnnnn	nnnnn
30 to 39	ant bites,dairy products	nnnnn	nnnnn	74	nnnnn	nnnnn	nnnnn
20 to 29	ant bites,bee stings,eggs	nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
80 +	ant bites	nnnnn	nnnnn	84	nnnnn	nnnnn	nnnnn
20 to 29	soy	nnnnn	nnnnn	59	nnnnn	nnnnn	nnnnn
...							

次に、これらの数値を集計します。この例では加算します。

4. ピボット・テーブルの Average Test Score 列には、特定のコンテキストにおける患者のテスト・スコアの平均値が表示されます。

   Avg Test Score メジャーは計算メンバで、Test Score を Patient Count で除算して算出されます。

結果セットのすべてのセルについて、この手順を繰り返します。

2.10 システムによるリストの生成方法

ここでは、キューブで定義されたリストをシステムで使用する方法について説明します。

ピボット・テーブル内で、ユーザは 1 つ以上のセルを選択します。

	Female		Male	
Age Group	Patient Count	Avg Test Score	Patient Count	Avg Test Score
0 to 29	2,064	60.06	2,115	60.62
30 to 59	2,071	60.52	2,090	60.33
60+	961	61.50	699	61.03

次に、ユーザが [詳細リスト] ボタン  をクリックすると、リストが表示されます。このリストには、選択したセルに関連付けられた最下位レベルのレコードの値が示されます (このセルに作用するすべてのフィルタも反映されます)。

#	PatientID	Age	Gender	Home City	Test Score
1	SUBJ_100524	30	F	Elm Heights	73
2	SUBJ_101001	30	F	Juniper	91
3	SUBJ_101014	30	F	Elm Heights	87
4	SUBJ_101072	30	F	Redwood	70
5	SUBJ_101331	30	F	Cedar Falls	

この表示を生成するために、システムでは以下の処理が行われます。

1. 選択したセルで使用されたファクトに対応するソース ID 値のセットを含む一時リスト・テーブルが作成されます。

2. リストの定義と共にこのリスト・テーブルを使用する SQL クエリが生成されます。

3. この SQL クエリが実行され、結果が表示されます。

キューブには複数のリストを含めることができます（用途の異なる複数のフィールドを表示するため）。アナライザでピボット・テーブルを作成する際には、そのピボット・テーブルで使用するリストを指定できます。

リスト・クエリは実行時に実行され、ファクト・テーブルではなくソース・データを使用します。このため、ファクト・テーブルが完全に最新でない場合でも、リストではファクト・テーブルの表示とは異なるレコード・セットを表示することができます。

3

モデル・オプションの概要

Business Intelligence モデルには、キューブやサブジェクト領域以外にも、多数の追加要素を含めることができます。追加要素については、"InterSystems Business Intelligence の上級モデリング" を参照してください。参考および計画用に、このページではすべてのモデリング要素についてまとめています。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

3.1 ピボット・テーブルで使用できる項目

このセクションでは、ピボット・テーブルを定義するために、アナライザで直接使用できる項目を比較します。これらの項目は、以下のように分類されます。

- レコードをグループ化またはフィルタ処理するレベルおよびその他の項目
- メジャーとして動作する項目
- プロパティ
- 直接アクセスできない項目

3.1.1 レコードをグループ化またはフィルタ処理する項目

レコードのグループ化またはフィルタ処理には以下の項目を使用できます。一部の検索可能メジャーを除き、これらの項目をメジャーとして使用することはできません。

アイテム	[行] オプションおよび [列] オプションで使用	[フィルタ] オプションで使用
ディメンジョンまたはレベル	はい	はい
ディメンジョンまたは関連するキューブのレベル	はい	はい
共有ディメンジョンまたは同じ複合キューブのレベル	はい	はい
計算メンバ (メジャー以外)	はい	はい
関連するキューブの計算メンバ (メジャー以外)	はい	はい
同じ複合キューブの別のキューブの計算メンバ (メジャー以外)	いいえ	いいえ
計算ディメンジョン	はい	はい

アイテム	[行] オプションおよび [列] オプションで使用	[フィルタ] オプションで使用
関連するキューブの計算ディメンジョン	はい	はい
同じ複合キューブの別のキューブの計算ディメンジョン	いいえ	いいえ
名前付きセット	はい	はい
別のキューブの名前付きセット	いいえ	いいえ
名前付きフィルタ	いいえ	はい
別のキューブの名前付きフィルタ	いいえ	いいえ
検索可能メジャー	はい (テキスト、文字列、および NLP メジャーは除く)	はい

関連キューブ、複合キューブ、計算ディメンジョン、および NLP メジャーの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

名前付きフィルタの詳細は、"アナライザの使用法" を参照してください。

既存のキューブ・ディメンジョンの一部である計算メンバは、[フィルタに計算メンバを表示] オプションを選択しない限り、フィルタに表示されないことに注意してください。

3.1.2 メジャーのように動作する項目

以下の項目はメジャーのように動作し、ピボット・テーブルの本文に表示されます。例外として (以下を参照)、これらの項目をフィルタ処理に使用することはできません。

アイテム	[行] オプションおよび [列] オプションで使用	[メジャー] オプションで使用	[フィルタ] オプションで使用
メジャー	はい (テキスト、文字列、および NLP メジャーは除く)	はい (テキスト、文字列、および NLP メジャーは除く)	検索可能な場合には使用する
関連するキューブのメジャー	いいえ*	いいえ*	いいえ
同じ複合キューブの別のキューブのメジャー	はい	はい	いいえ
計算メジャー	はい	はい	いいえ
別のキューブからの計算メジャー	いいえ	いいえ**	いいえ
品質メジャー	はい	はい	いいえ
別のキューブからの品質メジャー	該当せず**	該当せず**	該当せず**
ピボットタイプのプラグインのプロパティ	はい	はい	いいえ
別のキューブからのピボットタイプのプラグイン	該当せず**	該当せず**	該当せず**

関連キューブ、複合キューブ、NLP メジャー、品質メジャー、およびプラグインの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

* これらのメジャーは、この方法で使用した場合、正しく集約されません。

** 品質メジャーおよびプラグインは、使用されるキューブに直接関連付けられるように設計されています。

3.1.3 プロパティ

以下のテーブルは、プロパティの使用方法をまとめたものです。

アイテム	[行] オプションおよび [列] オプションで使用	[メジャー] オプションで使用	[フィルタ] オプションで使用
プロパティ	はい	いいえ	いいえ
別のキューブのプロパティ (関連または複合)	はい	いいえ	いいえ

関連キューブおよび複合キューブの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

3.1.4 アナライザで直接アクセスできない項目

参考までに、アナライザ内では以下の項目に直接アクセスできないことに注意してください。

- KPI
- 条件リスト
- 集約タイプのプラグイン
- プロダクション・ビジネス・メトリック

ただし、ビジネス・メトリック以外、これらの項目を使用する計算メジャーを定義できます。次のセクションを参照してください。その後、これらの計算メンバをアナライザで使用できます。

これらの項目の詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

3.2 計算メンバおよび計算メジャーで使用できる項目

計算メンバおよび計算メジャーを使用すると、キューブを再構築することなくモデルを大幅に拡張できます。以下のテーブルは、計算メンバまたは計算メジャーの定義に使用できるモデル要素のすべての種類をまとめたものです。

アイテム	計算メンバまたは計算メジャーからこの項目にアクセスする方法
MDX 標準キューブ項目 (ディメンション、階層、レベル、メジャー、プロパティ、計算メンバ、名前付きセット)	多数のオプション。"InterSystems MDX の使用法" を参照してください。
計算ディメンション*	ディメンションおよびメンバ名を使用して、目的のメンバを参照するメンバ式を作成します。その際、標準メンバに対して行う場合と同じ方法で行います。
品質メジャー*	%QualityMeasure ディメンションを使用して、メジャーを参照する品質メジャー式を作成します。
KPI またはプラグイン*	%KPI 関数を使用して、KPI またはプラグインのプロパティの値を参照します。

アイテム	計算メンバまたは計算メジャーからこの項目にアクセスする方法
条件リスト*	・ %LOOKUP 関数を使用して、条件リスト項目の値を返します。 ・ LOOKUP 関数を使用して、条件リスト項目の指定されたフィールド (値フィールドまたはその他のフィールド) を返します。 ・ %TERMLIST 関数を使用して、一連のメンバを作成し、条件リストのパターンを付与します。

*これらの項目の詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

システムには、計算メンバ内からプロダクション・ビジネス・メトリックにアクセスする方法は用意されていません。

3.3 利用可能なウィジェット・データ・ソースの比較

ピボット・テーブルは、ダッシュボードのウィジェットで最も一般的に使用される種類のデータ・ソースです。システムには、他にも多くの種類のデータ・ソースが用意されています。データ・ソースとして、以下の項目のいずれかを直接使用できます。

- ピボット・テーブル
- KPI ("InterSystems Business Intelligence の上級モデリング" を参照)
- ピボットタイプのプラグイン ("InterSystems Business Intelligence の上級モデリング" を参照)
- プロダクション・ビジネス・メトリック ("プロダクションの開発" を参照)

次のテーブルでは、これらの項目を比較します。

機能	ピボット・テーブル	KPI	プラグイン	プロダクション・ビジネス・メトリック
定義の格納方法	フォルダ項目	クラス定義	クラス定義	クラス定義
定義される場所	アナライザ	スタジオ	スタジオ	スタジオ
データ・ソース・オプション	MDX クエリ	· MDX クエリ · SQL query (SQL クエリ) · カスタム・コードによって返される値	MDX クエリ	カスタム・コードによって返される値
最下位レベルのレコードを使用したカスタム計算をサポートしますか	いいえ	いいえ	はい	いいえ
フィルタ処理をサポートしますか	はい	実装によってはサポートします	実装によってはサポートします	いいえ
カスタムのフィルタ処理をサポートしますか (例えば、レベルのメンバのサブセットのみをユーザに提供)	いいえ	実装によってはサポートします	実装によってはサポートします	いいえ
リストをサポートしますか	はい	実装によってはサポートします	はい	いいえ
ミニ・アナライザをサポートしますか	はい	いいえ	いいえ	いいえ

3.4 オプションの大まかなまとめ

参考のために、各要素の作成に使用するツールの情報を含め、InterSystems IRIS Business Intelligence モデルの利用可能なコンテンツについて以下のテーブルにまとめます。

アイテム	目的	定義の場所	作成する主なツール	説明されている場所
ディメンジョン、階層およびレベル	レコードのグループを定義します	キューブ・クラス	アーキテクト	ディメンジョン、階層およびレベルの定義
メジャー (数値、整数、日付、年齢、またはブーリアン値)	複数のレコード間で値を集約します	キューブ・クラス	アーキテクト	メジャーの定義
メジャー (テキストまたは文字列)	文字列データをファクト・テーブルに格納します。通常、これらのメジャーは検索可能でもあります。	キューブ・クラス	アーキテクト	メジャーの定義

アイテム	目的	定義の場所	作成する主なツール	説明されている場所
検索できるメジャー	複数のレコード間で値を集約します。最下位レコードのフィルタ処理もサポートします	キューブ・クラス	アーキテクト	メジャーの定義
NLP メジャー	NLP ディメンジョンで使用され、アナライザには表示されません	キューブ・クラス	アーキテクト	キューブでの Text Analytics の使用法
プロパティ	レベル固有のデータを組み込みます	キューブ・クラス	アーキテクト	プロパティの定義
リスト	最下位レベルの詳細へのアクセスを提供します	キューブ・クラス	アーキテクト	"リストの定義"、"リスト・フィールドの定義"、"リスト・グループの定義"
計算メジャー	他のモデル要素に基づいてメジャーを定義します	キューブ・クラスまたはピボット・テーブル	アーキテクトまたはアナライザ	計算メンバの定義
計算メンバ（メジャー以外）	他のモデル要素に基づいてメンバを定義します			
名前付きセット	メンバの再使用可能なセットを定義します	キューブ・クラス	アーキテクト	名前付きセットの定義
サブジェクト領域	キューブをフィルタ処理します。それ以外の場合には、その定義を改良します	サブジェクト領域クラス	アーキテクト	サブジェクト領域の定義
計算ディメンジョン	レコードのグループを定義し、実行時に取得します（通常、SQL または MDX を使用）	キューブ・クラス	アーキテクト	計算ディメンジョンの定義
品質メジャー	キューブの範囲外で MDX を使用してメジャーを定義します	品質メジャー・クラス	品質メジャー・マネージャ	品質メジャーの定義
関連キューブ	異なるキューブのレベルを使用します	キューブ・クラス	アーキテクト	キューブ間のリレーションシップの定義
複合キューブ	異なるキューブのメジャーを結合するか、またはこれらのメジャーを並べて表示します	サブジェクト領域クラス	アーキテクト	共有ディメンジョンおよび複合キューブの定義
KPI	代替データ・ソースとして使用するために、より柔軟な方法でデータのクエリを実行します	KPI クラス	スタジオ	基本的な KPI の定義
ピボットタイプのプラグイン	最下位データに基づいて値を計算します。ドラッグ・アンド・ドロップを介して使用します	プラグイン・クラス	スタジオ	プラグインの定義
集約タイプのプラグイン	最下位データに基づいて値を計算します。計算メンバで使用します			
条件リスト	さまざまな目的でキューブの範囲外で値を定義します	Business Intelligence フォルダ項目	条件リスト・マネージャ	条件リストの定義

アイテム	目的	定義の場所	作成する主なツール	説明されている場所
ビジネス・メトリック	プロダクション固有のデータを計算します	ビジネス・メトリック・クラス	スタジオ	″プロダクションの開発″
名前付きフィルタ	特定のキューブに使用するフィルタを定義します	グローバル	アナライザ	″アナライザの使用法″

4

原則と推奨事項

ここでは、InterSystems IRIS® データ・プラットフォームの Business Intelligence モデルに関する主要な原則およびその他の推奨事項について説明します。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

4.1 ベース・テーブルの選択

キューブを定義する際、最初の手順は、そのキューブのベース・クラスとして使用するクラスの選択です。覚えておかなくてはならない主要ポイントは、キューブ内のレコード数はすべてこのクラスのレコード数を指すことです。これはキューブ内で参照される他のクラスと異なる点です。同様に、ベース・クラスの選択で、そのキューブ内のすべてのメジャーの意味が決まります。

以下はその例です。

- ベース・クラスが `Transactions` の場合はトランザクションがカウントされます。このキューブでは、以下のようなメジャーを設定できます。
  - トランザクションあたりの平均ブローカ手数料 (またはトランザクションのグループの平均)
  - トランザクションあたりの平均トランザクション価額 (またはトランザクションのグループの平均)
- ベース・クラスが `Customer` の場合は顧客がカウントされます。このキューブでは、以下のようなメジャーを設定できます。
  - 顧客あたりの平均ブローカ手数料 (または顧客のグループの平均)
  - 顧客あたりの平均トランザクション価額 (または顧客のグループの平均)

それぞれが使用するベース・クラスの異なる複数のキューブを設定し、ダッシュボードでそれらをまとめて使用することができます。

4.2 ベース・テーブルの形式の選択

また、ベース・テーブルの形式を考慮することも重要です。以下のテーブルに主な考慮事項を示します。

ベース・テーブル	DSTIME をサポートしているか	リストをサポートしているか
ローカルの永続クラス (リンク・テーブル以外)	はい	はい
リンク・テーブル	いいえ	はい
ローカル・テーブルを使用するデータ・コネクタ	いいえ	はい
リンク・テーブルを使用するデータ・コネクタ	いいえ	いいえ

DSTIME 機能は、対応するキューブを最新状態に保つ最も簡単な方法です。この機能を使用できない場合は、他の手法を使用できます。詳細は、"キューブの最新状態の維持" を参照してください。

4.3 多数のレベルおよびメジャーの回避

クラスのインデックス数には制限があるため、InterSystems IRIS® データ・プラットフォームは、キューブのレベルおよびメジャーの数に上限を設定します。この制限 (経時的に増加する場合あり) の詳細は、"一般的なシステム制限" を参照してください。レベルおよびメジャー用にシステムが作成するインデックス数の詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。

過度に複雑なモデルを理解して使用するのは困難であるため、レベルおよびメジャーの数をこの制限によって要求される数よりもはるかに少なく維持することが適切です。

4.4 メジャーの適切な定義

メジャーは、ベース・テーブルのレコードと一対一のリレーションシップを持つ値に基づく必要があります。そうでない場合は、そのメジャーが適切に集約されなくなります。このセクションでは、この原則を例を使用して示します。

4.4.1 親テーブルに基づくメジャー

親テーブルのフィールドに基づくメジャーを定義しないでください。例えば、次のような 2 つのテーブルがあるとします。

- `Order` ― 各行は顧客から送信された注文を表しています。フィールド `SaleTotal` は、注文の合計金額を表しています。
- `OrderItem` ― 各行はその注文を構成する項目を表しています。このテーブルでは、注文の中でこの項目が占める合計金額をフィールド `OrderItemSubtotal` で表しています。

`OrderItem` をベース・テーブルとして使用するとします。また、親の `SaleTotal` フィールドに基づいて、メジャー `SaleTotal` を定義するとします。このメジャーの目的は、選択した注文項目のすべての販売における売上総額を表示することです。

次にファクト・テーブルの内容を検討します。以下に例を示します。

dim A	dim B	...	Sale Total	meas B	meas C	...
nnnnn	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	52.14	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	52.14	nnnnn	nnnnn	nnnnn

最初の 4 つの行は、同じ注文にある項目を表しています。次の 2 つの行は、別の注文の項目を表しています。

このモデルには `Item Type` という名前のディメンジョンがあるとします。システムによって、Item Type が type R であるすべての項目のレコードが取得されるときの動作について調べてみましょう。

Item Type	dim B	...	Sale Total	meas B	meas C	...
type Q	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
type R	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
type R	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
type S	nnnnn	nnnnn	279.07	nnnnn	nnnnn	nnnnn
type R	nnnnn	nnnnn	52.14	nnnnn	nnnnn	nnnnn
type T	nnnnn	nnnnn	52.14	nnnnn	nnnnn	nnnnn

type R の `Sale Total` メジャーの値を計算するために、3 つの値 279.07、279.07、および 52.14 が加算されます。ただし、このアクションでは、1 件の注文が 2 回カウントされています。

`Sale Total` メジャーが正しく集約されることもあります。つまり、選択した注文項目の正確な総売上の数字が得られることもあります。ただし、この例で見たように、カウントの重複を回避できないので、このメジャーがいつでも正しく集約されるとは限りません。

4.4.2 子テーブルに基づくメジャー

子テーブルの値をメジャーの基礎として使用することは可能ですが、そのためにはその値を子テーブルの関連行に集約する必要があります。

次の 2 つのテーブルについて考えてみましょう。

- `Customer` ― 各行は 1 人の顧客を表しています。
- `Order` ― 各行は顧客からの 1 件の注文を表しています。フィールド `SaleTotal` は、注文の合計金額を表しています。

`Customer` をベース・テーブルとして使用し、`SaleTotal` フィールドに基づいてメジャーを作成するとします。

1 人の顧客に複数の注文が存在する可能性があるので、指定された顧客の `SaleTotal` には複数の値が存在します。このフィールドをメジャーとして使用するには、これらの値をまとめて集約する必要があります。考えられる選択肢は、このメジャーの目的に応じて、これらの値の合計値または平均値を求めることです。

4.5 時間レベルの理解

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。

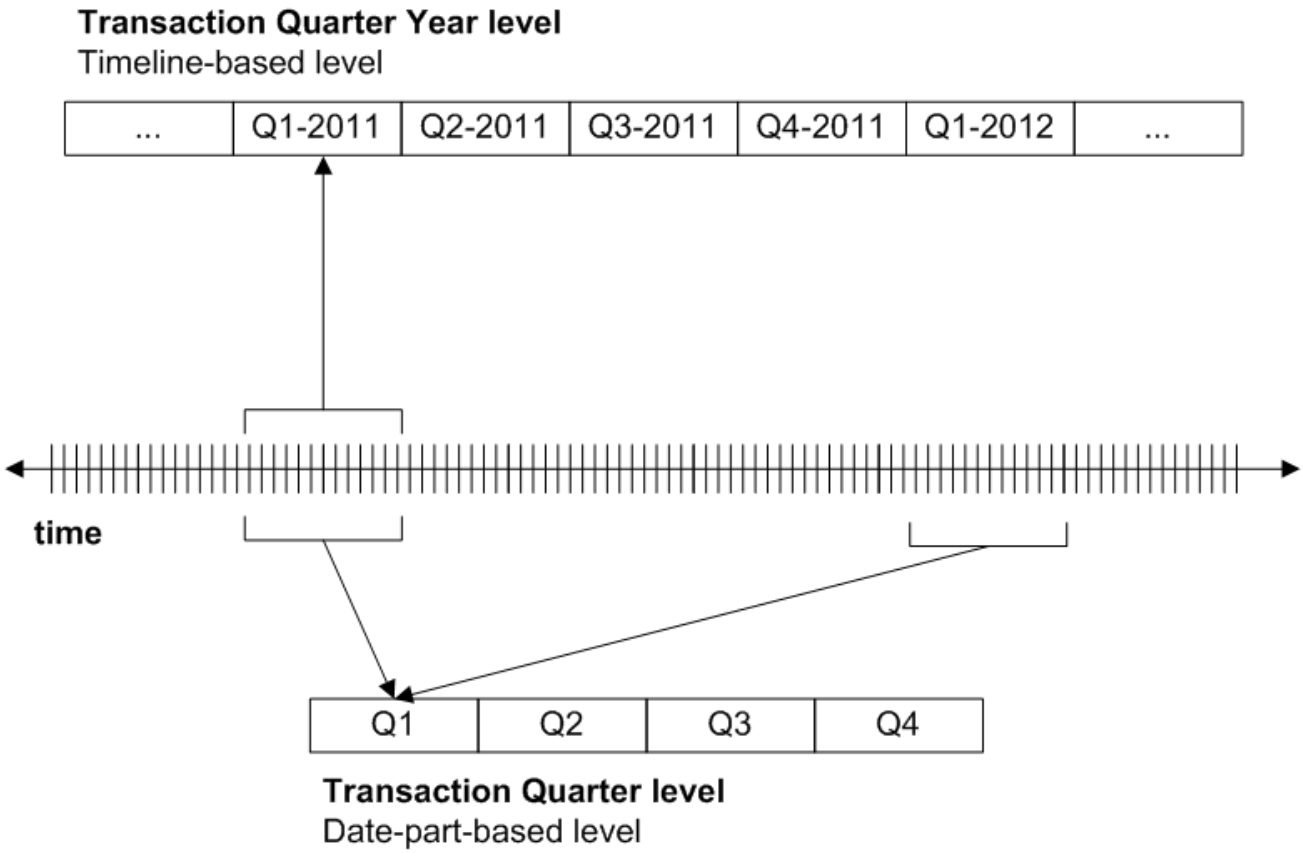
- 時間軸に基づくレベル :この種類の時間レベルでは、時間軸を互いに隣接する時間ブロックに分割します。このレベルのどのメンバも、単一の時間ブロックで構成されています。より正確に表現すると、その時間ブロックに関連付けられたレコードで各メンバが構成されています。 というレベルでは、2011 年の第 1 四半期に属する日付に発生したすべてのトランザクションが `Q1-2011` メンバにグループ化されます。

  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、エンジンでは正しいレコード群が必ず返されます。ただし、混乱を招く典型的な原因が 2 つあります。

- 時間レベルの階層（つまり、複数のレベルを含む階層）を定義する場合、組み込むものを注意深く考慮する必要があります。次のセクションで記述するように、MDX 階層は親子階層です。親レベルのどのメンバにも、その子メンバのすべてのレコードを含める必要があります。

  つまり、例えば、 ・レベルを ・レベルの親として使用できますが、 ・レベルの親としては使用できません。これを確認するには、前の図を参照してください。Q1 メンバはすべての年の最初の四半期を表すため、Q1 の親になることができる単一の年はありません。

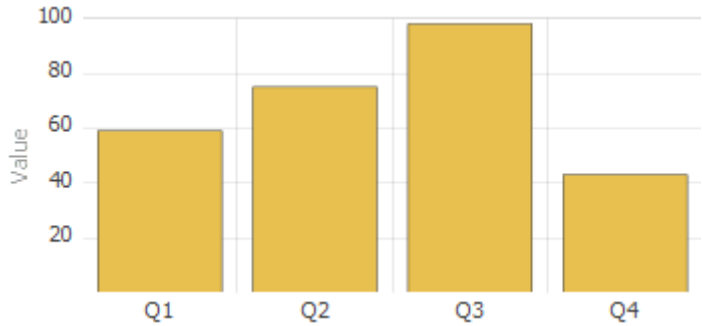
  適切な時間階層のガイドラインについては、"時間レベルと階層" を参照してください。

- 一部の MDX 関数は、時間軸を表すレベルでは有用ですが、日付部分のみを表すレベルでは役に立ちません。このような関数として、PREVMEMBER、NEXTMEMBER などがあります。例えば、該当する日付を収めたデータに対して Q1-2011 で PREVMEMBER を使用すると、Q4-2010 が返されます。Q1 で PREVMEMBER を使用すると、有意なデータは返されません。

以下の柔軟性のある方法の定義を試行します。時間軸を表す 1 つのレベルを含むように Year レベルを定義します。他のすべての時間レベルを日付部分レベルとして定義します。以下の例のように、これらのレベルを同時に使用できます。

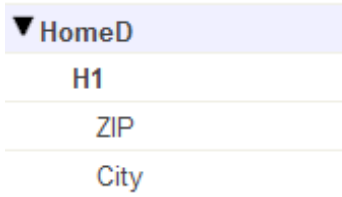
Year	Quarter	Patient Count
2000	Q1	2
	Q2	2
	Q3	1
	Q4	3
2001	Q1	5
	Q2	4
	Q3	2
	Q4	5
2002	Q1	1
	Q2	6

このピボット・テーブルは、行に対して Year（時間軸レベル）と Quarter（日付部分レベル）の組み合わせを使用します。Quarter レベルは、四半期ごとのパターンで情報を提供できるため、単独でも有用です。SAMPLES のサンプル・データはこのことを説明していませんが、この方法を確認できる多種類の季節ごとのアクティビティがあります。以下に例を示します。



4.6 階層の適切な定義

どの階層でも、アーキテクトに示すようにレベルの順序には注意する必要があります。レベルは自動的に、その階層内で前にリストされたレベルの子になります。例えば、Patients サンプルの HomeD ディメンジョンを考えてみましょう。



ZIP レベルは City レベルの親です。より正確には、ZIP レベルの各メンバは City レベルの 1 つ以上のメンバの親です(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しますが、Patients サンプルは単純化されており、このサンプルでは、郵便番号は市区町村より広い地域を表しています)。

階層内では、最初のレベルの粒度が最も低く、最後のレベルは粒度が最も高くなります。

MDX 階層は親子階層です。このルールを実施するために、システムは、キューブを構築するときに以下のロジックを使用します。

- 階層で定義された最初のレベルに対して、システムは一意のソース値ごとに個別のメンバを作成します。
- その後のレベルでは、親メンバと組み合わせて、レベルのソース値が考慮されます。

  例えば、最初のレベルが State で、2 番目のレベルが City だとします。City レベルのメンバを作成する際、システムは市区町村名とその市区町村が含まれる都道府県名の両方を考慮します。

時間階層では内部ロジックが若干異なりますが、目的は同じです。

4.6.1 階層反転の結果について

上記とは逆に、キューブ内で City レベルを ZIP レベルの前に移動して、リコンパイルおよび再構築するとします。ZIP レベルをピボット・テーブルの行に使用すると、以下のようになります。

ZIP	Patient Count
32006	1,126
32006	1,104
32007	1,095
34577	1,136
34577	1,159
34577	1,108
36711	1,076
38928	1,113
38928	1,083

この場合、システムは、(例えば) 異なる市区町村に属する 2 つの ZIP コード 32006 が存在すると想定しているため、同名で複数のメンバを作成しています。

この状況では、同名で複数のメンバが存在することは明らかに不適切です。ただし、シナリオによっては、同名で複数のメンバが存在することが正当である場合もあります。例えば、異なる国々に同名の市区町村が存在する場合があります。つまり、メンバ名に重複がある場合、反転階層を示す可能性があります。ただし、すべての状況で当てはまるわけではありません。

4.6.2 時間階層

MDX 階層は親子階層です。特定の子メンバのレコードのすべてが、親メンバにも含まれている必要があります。時間レベルを階層に配置する方法を注意深く考慮することが重要です。前のセクションを参照してください。また、適切な時間階層のガイドラインについては、"時間レベルと階層" を参照してください。

4.7 メンバ・キーおよび名前の適切な定義

各メンバには、名前と内部キーの両方があります。既定では、これらは時間レベルを除いて同じです。これらの ID は両方とも文字列です。モデルを定義する際は、以下の項目について考慮する必要があります。

- メンバ・キーが非常に長い場合、SUBSCRIPT エラーがトリガされることがあります。非常に長いメンバ・キーが生成されるソース・データを操作している場合、選択した方法でデータが保持されるように、お使いのレベルの sourceExpression にソース処理を実装することができます。
- メンバ名の重複が適切な場合があります。例えば、別人が同じ名前を持つ場合があります。ユーザがメンバをドラッグ・アンド・ドロップすると、システムで生成されるクエリでは、名前でなくメンバ・キーが使用されます。
- メンバ・キーが各レベルで一意であることを確認することをお勧めします。メンバ・キーの重複があると、すべての個別メンバを参照することが困難になります。

  特に、メンバ・キーの重複がある場合、ユーザはダブルクリックで確実にドリルダウンすることができません。
- 1 つのレベルに同名の複数のメンバが存在する場合は、メンバ間の区別を可能にするため、値がキーと同じであるレベルにプロパティを追加することもお勧めします。これは、レベルで使用されるものと同じソース・プロパティまたはソース式をそのプロパティの基にするだけです。

  その後、ユーザはメンバの区別に役立つプロパティを表示できます。

  または、このレベルにツールのヒントを追加して、メンバ・キーを表示するようにします。

以降のセクションでは、メンバ・キーまたは名前に重複が生じる可能性があるシナリオについて説明します。

4.7.1 重複メンバ・キーが生成される状況

メンバ・キーの重複は適切ではありません。メンバ・キーの重複がある場合、ユーザはダブルクリックで確実にドリルダウンすることができません。

時間レベルを除き、レベルに対する一意の各ソース値がメンバ・キーになります(時間レベルは、すべてのメンバに対する一意のキーの生成に使用されるロジックが少し異なります)。

上位レベルが同じ階層に存在する場合、システムは同じキーで複数のメンバを生成することがあります。"メンバ・キーの一意性の保証" を参照してください。

4.7.2 重複メンバ名が生成される状況

重複メンバ名は、業務のニーズによって適切な場合と不適切な場合があります。

時間レベルを除き、レベルに対する一意の各ソース値が既定でメンバ名になります(時間レベルの場合は、システムによりすべてのメンバに対して一意の名前が生成されます)。

システムによって複数のメンバが同じ名前で生成される可能性がある状況は以下の 2 つのみです。

- 高いレベルが同じ階層に存在する場合。この高いレベルは、適切な場合も、不適切な場合もあります。"階層の適切な定義" を参照してください。

- 以下のようにレベルを定義している場合。
  - メンバ名が、レベル定義自体とは別に定義されている。
  - メンバ名が一意でないものに基づいている。

  "プロパティ名をメンバ名として使用する方法" を参照してください。

InterSystems IRIS は、ファクト・テーブルの構築時に文字列値から先頭のスペースを自動的には削除しません。ソース・データに先頭のスペースが含まれる場合は、これらを削除するソース式を使用します。以下はその例です。

```
$ZSTRIP(%source.myproperty,"<W")
```

このようにしないと、同じように見える名前を持つ複数のメンバが作成されることになります (一部の名前の先頭に余分なスペースが存在するため)。

4.8 過度な粒度レベルの回避

Business Intelligence の新規ユーザは、通常、ベース・テーブルとの 1 対 1 のリレーションシップを持つレベルを定義します。このようなレベルは有効ですが、そのレベルをさらにフィルタと組み合わせて、表示されるメンバの数を制限しないときには、特に役立つことはありません。

過度な粒度レベル (詳細レベル) には大量 (おそらく数十万から数百万) のメンバが含まれ、Business Intelligence はこのシナリオに向けて設計されていません。

例えば、Patient サンプルに変更を加えて Patient レベルを持つようにしたとします。このようにすると、以下のようなピボット・テーブルが得られます。

PatientID	Patient Count	Age	Allergy Count
SUBJ_100301	1	2	
SUBJ_100302	1	79	
SUBJ_100303	1	17	2
SUBJ_100304	1	72	
SUBJ_100305	1	66	1
SUBJ_100306	1	44	
SUBJ_100307	1	66	1

これは、有効ですが、これと同じ結果は SQL クエリでより効率的に生成できます。"Business Intelligence によるファクト・テーブルの構築方法と使用方法" で説明した処理について検討してください (その説明では、実際の処理ではなく概念的な流れを示していますが、全体的な考え方は同じです)。その処理は、できる限り短時間で値を集約することを目的としています。上記のピボット・テーブルには、集約がありません。

ここに示したようなピボット・テーブルが必要な場合は、SQL ベースの KPI として作成してください。"InterSystems Business Intelligence の上級モデリング" を参照してください。

4.9 リスト・ベースのレベルの使用に関する注意

Business Intelligence では、他の多くの BI ツールと異なり、レベルのベースにリスト値を使用できます。このようなレベルは便利ですが、その動作を理解しておくことが重要です。

例えば、1 人の患者に複数のアレルギーを指定できます。各アレルギーにはアレルゲンと重症度があります。ベース・テーブルは Patients で、モデルには Allergies と Allergy Severity の各レベルが存在するとします。次のようなピボット・テーブルを作成できます。

Allergies	Nil known allergies	Minor	Moderate
additive/coloring agent		137	121
animal dander		132	136
ant bites		126	131
bee stings		124	141
dairy products		128	114
dust mites		126	103
eggs		132	125
fish		137	135
mold		116	136
nil known allergies	1,578		
peanuts		129	114
pollen		131	135

このピボット・テーブルを初めて見たユーザは、このピボット・テーブルでは、患者のさまざまなアレルギーどうしの相関関係が示されていると考えるかもしれません。しかし、それは正しくありません。

このキューブの他のすべてのピボット・テーブルと同様、このピボット・テーブルも患者のセットを示します。例えば、ant bites の行は、アリによる咬傷に対してアレルギーがある患者を表しています。Minor 列は、軽度と識別されたアレルギーを 1 つ以上持っている患者を表しています。アリによる咬傷に対する 1 つのアレルギーと、軽度とマークされたアレルギーを 1 つ以上持つ患者が 126 人います。これは、アリによる咬傷に対して軽度のアレルギーを持つ患者が 126 人いるということではありません。

患者のさまざまなアレルギーどうしの相関関係を示すピボット・テーブルを作成することもできます。ただし、そのためには、患者ではなく患者のアレルギーに基づいたモデルを定義する必要があります。

フィルタ内でリスト・ベースのレベルを使用する場合は、結果に対する慎重な配慮が必要です。次のようなピボット・テーブルがあるとします。

Allergies	Patient Count
No Data Available	3,943
additive/coloring agent	413
animal dander	418
ant bites	436
bee stings	446
dairy products	445
dust mites	416
eggs	389
fish	419
mold	397
nil known allergies	1,427
peanuts	444
pollen	426
shellfish	442
soy	449
tree nuts	453
wheat	435

このピボット・テーブルは、患者をアレルギー別に分類して表示します。ここで、このピボット・テーブルにフィルタを適用し、フィルタで魚のメンバを選択するとします。

Allergies
fish

Allergies	Patient Count
additive/coloring agent	13
animal dander	19
ant bites	16
bee stings	23
dairy products	24
dust mites	22
eggs	12
fish	419
mold	13
peanuts	22
pollen	22
shellfish	23
soy	20
tree nuts	24
wheat	19

これで、魚に対するアレルギーを持つ患者のみが表示されます。以下のことに注意してください。

- このピボット・テーブルは、魚のメンバを表示します。このメンバの患者数は、前のピボット・テーブルと同じです。
- また、Allergies レベルのその他のメンバも表示されます。これらのメンバについては、患者数が前のピボット・テーブルより少なくなります。

- このピボット・テーブルには、Diagnoses レベルの No Data Available メンバは含まれません。Nil Known Allergies メンバも含まれません。

これらの結果を理解するには、魚に対するアレルギーのある患者だけを表示していること、およびこれが魚アレルギーのみの表示とは同じでないことを忘れないでください。魚に対してアレルギーがある患者は、他の物に対するアレルギーもあります。例えば、魚に対してアレルギーがある患者のうち 13 人は、添加物/着色料に対するアレルギーもあります。

フィルタを変更して No Data Available メンバのみを選択すると、以下のように表示されます。

Allergies No Data Available	
Allergies	Patient Count
No Data Available	3,943

この場合は、アレルギーの記録がない患者のみが表示されます。定義により(このレベルの定義方法により)、これらの患者と特定のアレルギーのある患者の重複はありません。

注釈　　リスト・ベース・レベルは、子レベルまたは親レベルを持つことができません。

4.10 NULL 値の適切な処理

どのメジャーやレベルでも、状況によってはソース値が NULL になる可能性があります。ユーザの業務上の必要性および使いやすさの要件に応じて、システムが NULL を処理する方法を理解して、目的のモデルを調整することは重要です。

4.10.1 メジャーにおける NULL 値

メジャーの場合、特定のレコードのソース値に欠落があると、システムはファクト・テーブルのメジャー列に値を何も書き込みません。また、そのレコードは、メジャー値の集約時にシステムによって無視されます。ほとんどのシナリオでは、これが適切な動作です。これが適切ではない場合は、NULL 値を検出し、これを 0 などの適切な値に置換するソース式を使用する必要があります。

4.10.2 レベルにおける NULL 値

レベルで、ベース・クラスの特定のレコードのソース値が欠落している場合、システムは NULL 値を含むメンバを自動的に作成します (1 つの例外を除く)。メンバ名として使用するヌル置換文字列を指定します。この指定がないと、そのメンバには `Null` という名前が付けられます。

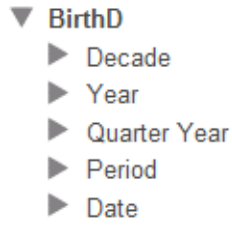
例外は計算ディメンジョンです。詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。置換文字列はこの状況において何の影響も及ぼしません。

4.11 使いやすさに関する考慮事項

モデル要素がユーザにどのように表示され、ユーザがこれらをどのように使用するかを考慮することも有益です。ここではアナライザとピボット・テーブルでモデル要素をどのように表示するかについて説明し、最後にモデルに対する推奨事項を紹介します。

4.11.1 ディメンジョン、階層およびレベルの表示に関する考慮事項

アナライザの [モデル・コンテンツ] 領域には、各ディメンジョンとその中のレベルが表示されますが、階層は表示されません (スペースの関係上)。以下はその例です。



このため、アナライザで作業するユーザは、階層を介してどのレベルが関連しているかを認識しているとは限りません。

レベルが行に使用されている場合、レベル名が列タイトルとして表示されます。以下はその例です。

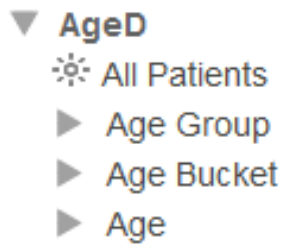
Decade	Patient Count
1910s	79
1920s	203
1930s	532
1940s	721

ディメンジョンが行に使用されている場合は、ディメンジョン名が列タイトルとして表示されます。また、ディメンジョンの All メンバ、およびそのディメンジョンで定義された最初のレベルのすべてのメンバを取得する MDX 関数も使用されます。

AgeD	Patient Count
All Patients	10,000
0 to 29	4,255
30 to 59	4,119
60+	1,626

4.11.2 All メンバの使用に関する考慮事項

[モデル・コンテンツ] 領域にはすべての All メンバが表示されます。以下はその例です。



アナライザで、ユーザは他のメンバのドラッグと同様の方法で All メンバをドラッグ・アンド・ドロップできます。

あるディメンジョンの All メンバは、他のすべてのディメンジョンの All メンバと等しくなります (名前は除きます)。適切な汎用名がある場合は All メンバは最下行として役立ちます。次に例を示します。

Diagnoses	Patient Count	Avg Age	Avg Enc Count
None	8,425	33.24	31.26
asthma	703	34.79	31.42
CHD	323	67.49	52.85
diabetes	504	57.24	47.38
osteoporosis	200	79.46	60.56
All Patients	10,000	36.01	33.03

4.12 複数キューブの考慮事項

業務のある領域を分析するために、複数のキューブを定義する必要があることに気付いたときには、以下の点について検討してください。

- キューブが多数あるときには、多くの場合、リレーションシップが役立ちます。個別の (定義が異なる) キューブに同じディメンジョンを何度も定義する代わりに、そのディメンジョンを 1 か所で定義できます。こうすることで、開発がより容易になり、必要なディスク容量が少なくなります。

  リレーションシップの短所をあえて挙げると、"InterSystems Business Intelligence の上級モデリング" で説明するように、個別のキューブを再構築したときには、それに依存するすべてのキューブも適切な順序で再構築する必要があることです。

- リレーションシップまたは共有ディメンジョンを定義する場合は、キューブ・マネージャを使用してキューブを正しい順序で構築してください。"キューブの最新状態の維持" を参照してください。

  または、キューブを適切な順序で構築するユーティリティ・メソッドまたはルーチンを定義します。キューブを追加するときにそのようなメソッドを維持するほうが、それらのキューブを正しい順序で手動で再構築するよりも簡単です。

  関連キューブを間違った順序で構築すると、トラブルシューティングが困難な問題が発生することがあります。

- 複数のキューブからのメジャーを表示するピボット・テーブルが必要な場合は、共有ディメンジョンと複合キューブを定義する必要があります。複合キューブは、それぞれ異なるキューブに属するメジャーを同時に使用するための唯一の方法です。

4.13 推奨事項

業務のニーズによっては、次の推奨事項も役立ちます。

- ピボット・テーブルでディメンジョンを直接使用するかどうかを決定します。直接使用する場合、それらのディメンジョン (少なくとも表示名) にはわかりやすい名前を割り当てます。

  直接使用しない場合は、MDX クエリおよび式の作成を容易にするため、名前を簡潔なものにして、スペースを省略します。

  また、MDX クエリまたは式の表示や作成が必要な場合は、構文をわかりやすくするため、ディメンジョンにはディメンジョン内のどのレベルとも異なる名前を使用します。

- 階層名はアナライザやピボット・テーブルに表示されません。短い名前 (H1 など) を使用すると、MDX クエリおよび式が簡潔かつ読みやすくなります。

- どのディメンジョンでも階層を 1 つのみ定義します。この規則を使用すると、ユーザはディメンジョン内のレベルが相互に関連付けられているかどうかを容易に認識できます。

- All メンバは、1 つのディメンジョンでのみ定義します。この All メンバには `All Patients` などの適切な汎用名を指定します。All メンバの使用目的によっては、`Total` や `Aggregate Value` などの名前が適している場合もあります。
- レベルにはわかりやすい名前を使用します。この名前はピボット・テーブルに表示されます。
- 階層が異なれば、同じ名前で複数のレベルを設定できます。ただし、ピボット・テーブルおよびフィルタはレベル名しか表示されないため、レベル名は一意にすることをお勧めします。
- 適用可能なレベルおよびメジャーごとに **[ファクト・テーブルのフィールド名]** オプションの値を指定します。このオプションは、時間レベル、NLP レベル、または NLP メジャーには適用されません。一意の名前を使用するように注意してください。

  ファクト・テーブル内で指定されたレベルまたはメジャーが格納されているフィールドを特定できる場合は、トラブルシューティングが非常に簡単になります。

  詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。

5

InterSystems Business Intelligence のモデルの定義

ここでは、Business Intelligence キューブの定義に関する基本的な事項について説明します。

重要　キューブのビルド時や詳細リストの実行時に、システムは SQL を使用してデータにアクセスします。モデルが SQL 予約語のクラス・プロパティを参照する場合は、区切り識別子のサポートを有効にして、InterSystems IRIS Business Intelligence がプロパティ名をエスケープできるようにする必要があります。予約語のリストは、"予約語" を参照してください。区切り識別子のサポートの有効化に関する詳細は、"識別子" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

5.1 キューブの定義

キューブを定義する手順は以下のとおりです。

1. アーキテクトで、**[新規作成]** をクリックします。

   新しいキューブの詳細を入力するダイアログ・ボックスが表示されます。

2. **[キューブ]** をクリックします。
3. 少なくとも、以下の情報は入力してください。
   - **[キューブ名]** – クエリで使用するキューブの論理名。
   - **[キューブのクラス名]** – キューブ・クラスの完全なパッケージおよびクラス名。
   - **[ソースクラス]** – このキューブのベース・クラスの完全なパッケージおよびクラス名。サブセクション "キューブに使用可能なソース・クラス" を参照してください。

     クラス名は入力することも、**[参照]** をクリックしてクラスを選択することもできます。

   その他のオプションは、このページの後続部分で説明します。

   キューブ・クラスのクラス名のほか、キューブの作成後はすべてのキューブ・オプションを編集できます。
4. **[OK]** をクリックします。
5. 必要に応じて、キューブを保存します。そのためには、以下の操作を実行します。

a. [保存] をクリックします。

b. [OK] をクリックします。

これで、クラスが作成されます。

ユーティリティを使用してキューブ・クラスを生成することもできます。これについては、この後のサブセクションで説明します。

または、"キューブ・クラスのリファレンス情報" の説明に従ってクラスを手動で作成します。

5.1.1 キューブ・クラスの生成

クラス %DeepSee.WizardUtils は、キューブ・クラスの生成に使用できる %GenerateCubeDefinition() メソッドを提供します。メソッドは、以下のとおりです。

```
classmethod %GenerateCubeDefinition(pSourceClass As %Library.String(MAXLEN="")="",
                                    pCubeName As %Library.String(MAXLEN=""),
                                    pCubeClass As %Library.String(MAXLEN="")="",
                                    pAutoDelete As %Library.Integer = 0)
```

以下は、この指定の説明です。

- pSourceClass は、キューブのソース・クラスのフルネームです。
- pCubeName は、キューブの論理名です。
- pCubeClass は、キューブ・クラスのフルネームです。
- pAutoDelete は、キューブ・クラスが既存の場合に削除するかどうかを制御します。この引数が非ゼロの場合、クラスは削除されます。ゼロの場合、削除されません。

このメソッドは、以下のようにキューブ定義を生成します。

- ソース・クラスの数値プロパティごとに 1 つのメジャーがあります。
- ソース・クラスの日付プロパティごとに 1 つの日付ディメンジョンがあります。このディメンジョンには、3 つのレベルを持つ階層が 1 つあります。3 つのレベルとは、年、年月、および日付です。
- ソース・クラスの他のプロパティごとに 1 つのデータ・ディメンジョンがあります。このディメンジョンには、1 つのレベルを持つ階層が 1 つあります。
- ソース・クラスのすべてのプロパティを使用する 1 つのリストがあります。

このメソッドでは、一時プロパティおよび多次元プロパティは無視されます。

5.1.2 キューブのベース・クラスの変更

まれに、キューブのベース・クラスを変更する必要性が生じる場合があります。そのためには、アーキテクトで以下のいずれかの操作を実行します。

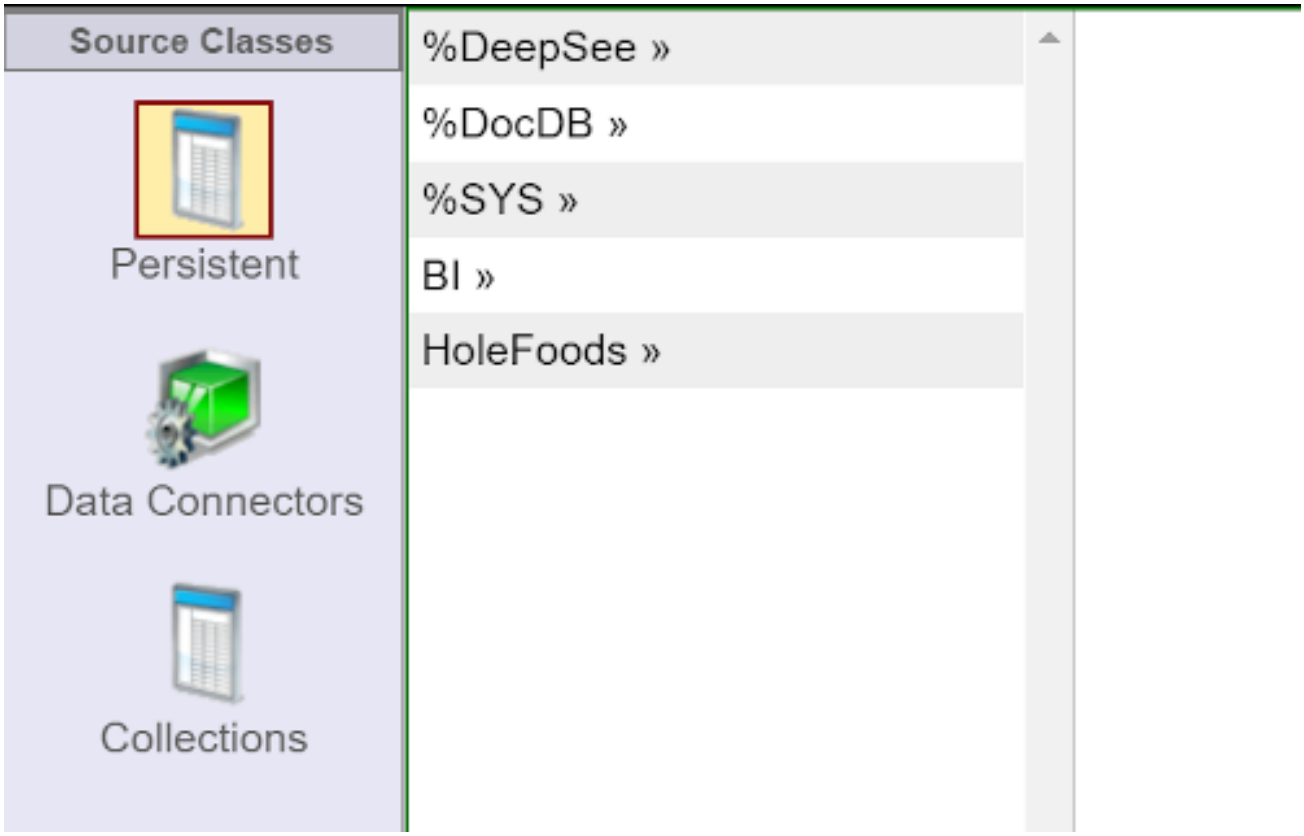
- 詳細領域で、キューブの [ソースクラス] オプションを編集します。
- [クラス・ビューワ] の最上部の [ソース値] の横にある [変更] リンクをクリックします。

  この処理でダイアログ・ボックスが表示され、ソース・クラスを選択できます。これは、"キューブに使用可能なソース・クラス" で表示されているものと同じダイアログ・ボックスです。

この後、モデルのすべての部分でソース・プロパティまたはソース式を適切に変更することを忘れないでください。

5.2 キューブに使用可能なソース・クラス

アーキテクトで、[**ソースクラス**] の横にある [**新規**] をクリックしてから [**参照**] をクリックすると、以下のようなダイアログ・ボックスが表示されます。



ここでは、このネームスペースのキューブのソースとして使用可能な任意のクラスを選択できます。この方法で使用可能なクラスには以下の 3 タイプがあります。

- 永続クラス ― %Library.Persistent を拡張するクラス。
- データ・コネクタ・クラス ― %DeepSee.DataConnector を拡張するクラス。データ・コネクタは、任意の SQL クエリの結果を、キューブのソースとして使用可能なオブジェクトにマップします。通常、データ・コネクタは、InterSystems 以外のデータベースにアクセスしますが、データ・コネクタを使用して InterSystems データベースに対する SQL クエリ (ビューに対する SQL クエリを含む) を指定することもできます。

  データ・コネクタに基づくキューブと、そのキューブ内に同じくデータ・コネクタに基づくリストがある場合、これらのすべてのデータ・コネクタは、`idkey="true"` とマークされた同じプロパティを持つ必要があります。これは、基礎となるメカニズムがいずれの場合も同じ ID 値を使用するためです。

  "データ・コネクタの定義と使用" を参照してください。
- コレクション・クラス。

5.3 他のキューブ・オプション

アーキテクトでは、以下のキューブ・オプションを指定できます。

- **[キューブ名]** ― クエリで使用するキューブの論理名。
- **[表示名]** ― キューブのローカライズ可能な名前。この名前を指定しない場合、代わりに論理名が表示されます。
- **[説明]** ― (オプション) キューブ・クラス定義に追加するコメント。クラス定義の開始時に、各行が個別のコメント行として保存されます。
- **[キャプション]** ― (オプション) このキューブでの作業時にアナライザおよびその他のユーティリティに表示されるキャプションを指定します。
- **[ドメイン]** ― (オプション) このキューブのローカライズ文字列を含むドメインの名前を指定します。すべてのキューブに単一のドメインを使用すると便利な場合があります。また、キューブごとに別のドメインを設定することが適切な場合もあります。"ローカライズの実行" を参照してください。
- **[ソースクラス]** ― このキューブのベース・クラスの完全なパッケージおよびクラス名。
- **[ヌル置換文字列]** ― (オプション) レベルのソース・データが NULL の場合にメンバ名として使用する文字列 (例 : `None`) を指定します。

  レベルでは、同名のレベル・オプションを使用してこのオプションをオーバーライドできます。
- **[デフォルトの詳細リスト]** ― (オプション) このキューブの既定のリストの論理名。このリストはキューブで定義する必要があります。
- **[リソース]** ― (オプション) キューブを保護するリソースを指定します。

  この使用法については、"セキュリティの設定" を参照してください。
- **[所有者]** ― (オプション) キューブの所有者を指定します。InterSystems IRIS® データ・プラットフォーム・ユーザ名を指定します。
- **[カウント・メジャーの名前]** ― (オプション) Count メジャーの代替名を指定します。既定は `%COUNT` です。複合キューブを作成する場合は、Count メジャーの名前を変更すると便利です。"InterSystems Business Intelligence の上級モデリング" を参照してください。
- **[カウント・メジャーのキャプション]** ― (オプション) Count メジャーの代替キャプションを指定します。既定は `COUNT` です。
- **[初期ビルド順]** ― (オプション) キューブ全体を構築する際に使用するオプションの ORDER BY 節を指定します。キューブの同期化およびインクリメンタル更新には影響しません。ソース・テーブルのフィールドのコンマ区切りリストを指定します。SQL キーワード ASC および DESC を使用できます。例 : `Age DESC,Gender`

  このオプションの意味については、"キャッシュのバケットとファクトの順序" を参照してください。
- **[ビルド制限]** ― (オプション) キューブの構築および更新時に使用するオプションの WHERE 節を指定します。これにより、キューブがレコードのサブセットを使用するようになります。ソース・テーブルのフィールドを使用する SQL 比較式を指定します。例 : `Gender='F'`

  キューブがデータ・コネクタに基づいている場合、このオプションは何の効果も持ちません。

  代替オプションについては、"キューブで使用するレコードの制限" を参照してください。
- **[依存]** ― (オプション) このクラスをコンパイルする前に実行可能にする必要がある 1 つ以上のクラスを指定します。このオプションは、アーキテクトが DependsOn コンパイラ・キーワードをどのように設定するかを制御します。

  既定では、キューブを作成すると、システムは自動的に DependsOn キーワードを、キューブのソース・クラスの名前と等しくなるように設定します。場合によっては (例えば、キューブ間のリレーションシップによって)、追加のクラスを指定する必要があります。

  このオプションを指定する必要がある場合は、コンマで区切られたクラスのリストを指定して、そのリスト内の各クラスの完全なパッケージ名とクラス名を指定します。このリストには、キューブのソース・クラスが含まれている必要があります。

キューブ間のリレーションシップの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

- **[SQL Restrict を許可]** ー(オプション)このチェック・ボックスにチェックを付けると、キューブに対して %SQLRESTRICT ディメンジョンを使用できます。このオプションにより、SQL SELECT 文または WHERE 節を追加することで、MDX クエリのスライサに SQL 制限を定義できます。このオプションを選択すると、アナライザの **[ピボット・オプション]** メニューの **[SQL 制限]** フィールドも有効になります。%SQLRESTRICT ディメンジョンの使用法の詳細は、"%FILTER 節" を参照してください。

5.4 キューブへの項目の追加

アーキテクトでは、以下の 2 つの一般的な方法でキューブに項目を追加できます。

- **[要素を追加]** リンクの使用。以下の操作を行います。
  1. モデル・ビューワ上部の **[要素を追加]** をクリックします。

     追加する項目の種類を選択できるダイアログ・ボックスが表示されます。
  2. 項目名を入力します。
  3. 項目の種類をクリックします。
  4. [OK] をクリックします。
- ドラッグ・アンド・ドロップ操作の使用。以下の操作を行います。
  1. クラス・ビューワからプロパティ名をドラッグします。
  2. この名前をモデル・ビューワの見出しにドロップします。名前をどこにドロップするかによって結果が異なります。例えば、プロパティ名を **[メジャー]** の見出しにドラッグ・アンド・ドロップするとメジャーが作成されます。

いずれの場合も、項目が追加されてモデル・ビューワに表示されます。次に、詳細領域で変更を加えることができます。

以下のテーブルは、さまざまな種類のキューブ項目について説明している項目を示しています。これらの項目にも、プロパティ名をドラッグ・アンド・ドロップする場所に関して固有の情報が記載されています。

項目タイプ	参照先
データ・ディメンジョン、時間ディメンジョン、年齢ディメンジョン、階層、またはレベル	ディメンジョン、階層およびレベルの定義
プロパティ	プロパティの定義
メジャー	メジャーの定義
リスト	リストの定義
リスト・フィールド	リスト・フィールドの定義
計算メンバ	計算メンバの定義
名前付きセット	名前付きセットの定義

5.5 モデル要素の名前

モデル要素を定義する際は、その要素の論理名を指定します (アーキテクトの **[名前]** フィールド)。この名前は MDX クエリで使用され、またその要素の既定の表示名になります。ここでは、この名前の要件と推奨事項について説明します。非準拠の名前を定義しようとすると、エラーが表示され、別の名前を入力するよう求められます。

論理名は、以下の規則に従う必要があります。

- 先頭の文字は、文字 (Latin-1 文字セット)、数字、アンダースコア文字 (_) のいずれかにする必要があります。
- 先頭以外の文字は、文字、数字、スペース、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 IRIS バージョン 2020.1.1 以降では、要素名にピリオドとコロンも使用できます。

  名前にスペースを使用した場合、MDX クエリの作成時には名前を角括弧で囲む必要があります。
- 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。
- 要素名は、大文字と小文字の違いだけでは一意とは見なされません。例えば、`zipcode` と `zipCode` は、一意の名前とは見なされません。

論理名は、以下の追加規則にも従う必要があります。

- 特定の InterSystems IRIS ネームスペース内では、各キューブ名が一意であることが必要です。
- 特定のキューブ内では、各ディメンジョン名が一意であることが必要です。
- 特定のディメンジョン内では、各階層名が一意であることが必要です。
- 特定の階層内では、各レベル名が一意であることが必要です。

  注釈　キューブ内では、レベル名が一意である必要はありません。ただし、キューブ内に同じ名前のレベルが複数ある場合、**[ファクト・テーブルのフィールド名]** オプションを指定し、ファクト・テーブル内の各レベルを一意の名前にする必要があります。"ファクト・テーブル内のフィールド名の指定" を参照してください。
- 特定のレベル内では、各プロパティ名が一意であることが必要です。キューブ・クラス定義で、レベルに `<member>` 要素を手動で定義する場合、プロパティにそのレベル内のいずれかの `<member>` 要素と同じ名前を指定することはできません。
- 内部プロパティの名前は大文字と小文字が区別されない予約キーワードであるため、ユーザ指定のプロパティの名前として使用できませんが、1 つだけ例外があります。その例外とは、`Name` (大文字と小文字は区別されない) というプロパティで **[メンバ名として使用]** オプションも有効になっている場合は、このプロパティを作成できるというものです。

  内部プロパティのリストについては、"内部プロパティ" を参照してください。
- 特定のキューブ内では、各メジャー名が一意であることが必要です。
- **[式]** 要素が単一値ではなくオブジェクトを返すように定義されている場合、その論理名にピリオド (".") を含めることはできません。このような場合は、代わりにアンダースコア ("_") 文字を使用することをお勧めします。

5.6 その他の共通オプション

アーキテクトでは、モデル要素を定義する際、その要素に対して以下のオプションも指定できます。

- **[表示名]** – (オプション) ユーザ・インタフェースで使用されるこの要素のローカライズ名。この名前を指定しない場合、代わりに論理名が表示されます。
- **[説明]** – (オプション) この要素の説明。
- **[無効]** – (オプション) このチェック・ボックスにチェックを付けると、その要素は無効 (コンパイラで認識されない) になります。キューブのリコンパイル時は、この要素が無視されます。
- **[追加の説明]** – (オプション) この要素に関する追加の注意事項。アーキテクトとスタジオでのみ表示されます。

5.7 キューブのコンパイルとビルド

アーキテクトでは、キューブを開発する際、多くの場合はそれらのキューブを複数回にわたってリコンパイルしたり、再構築したりします。概要は以下のようになります。

- キューブをコンパイルするには、**[コンパイル]** をクリックします。

  クラスのコンパイルが開始され、進捗状況を示すダイアログ・ボックスが表示されます。

  保存していない変更がある場合は、システムによって保存されます。

  次に **[完了]** をクリックします。
- キューブをビルドするには、**[ビルド]** をクリックします。ダイアログ・ボックスが表示されます。**[ビルド]** をクリックします。

  キューブの構築が開始され、その進捗状況が表示されます。そして、[OK] をクリックします。

  これで、アナライザでキューブの使用が可能になります。

詳細は、"キューブのコンパイルとビルド" を参照してください。

5.8 アナライザでキューブを開く

キューブを開発するときには、定期的にアナライザを使用して、結果を検証する必要があります。アナライザでキューブを開く手順は以下のとおりです。

1. [Analytics]→**[アナライザ]**→**[進む]** をクリックします。

   Tipヒン 既にアナライザが開いている場合は、ページ上部の **[アナライザ]** リンクをクリックするだけです。
   ト

2. 左領域に検証対象のキューブが表示されていない場合は、**[開く]** をクリックして、キューブを選択します。

レベルの検証に固有のヒントについては、"レベルの検証" を参照してください。アナライザの使用に関する一般情報は、"アナライザの使用法" を参照してください。

5.9 キューブの削除

キューブを削除するには、以下の手順を実行します。

1. ターミナルで、以下のコマンドを実行します。

ObjectScript

```
do ##class(%DeepSee.Utils).%KillCube(cubeName)
```

cubeName は、削除するキューブの論理名です。このコマンドによって、キューブのキャッシュとインデックスが削除されます。

2. また、ターミナルで、以下のようにキューブのメタデータを削除します。

ObjectScript

```
kill ^DeepSee.Cubes("cubes",cubeName)
```

cubeName は、削除するキューブの論理名です。

3. 以下のいずれかの方法を使用して、キューブ・クラス（および生成されたクラスとデータ）を削除します。

- ターミナルで、以下のコマンドを実行します。

  ObjectScript

  ```
  do $system.OBJ.Delete(classname)
  ```

  classname は、キューブ・クラスの完全なパッケージおよびクラス名です。以下はその例です。

  ObjectScript

  ```
  do $system.OBJ.Delete("Mypackage.Myclass")
  ```

- スタジオでキューブ・クラスを右クリックして、[**削除**] を選択します。

キューブを削除しない場合は、これをリコンパイルおよび再構築します。

6

キューブのコンパイルとビルド

ここでは、Business Intelligence キューブをコンパイルおよびビルドする方法を説明します。

注釈　構築プロセス中、ユーザはクエリを実行できません。(ただし、クエリを現在実行中の場合は、キューブをビルドできます)。

6.1 リコンパイルおよび再構築のタイミング

注釈　以前のバージョンから InterSystems IRIS をアップグレードする際に、あらゆる新しい最適化を活用できるように、すべてのキューブ・クラスおよびサブジェクト領域クラスをリコンパイルすることをお勧めします。

キューブ・クラスまたはサブジェクト領域クラスに変更を加える場合、その変更を有効にするには、そのクラスをリコンパイルする必要があります。キューブに多数の変更がある場合、その変更を有効にするには、キューブも再構築する必要があります。

以下のテーブルは、変更の後に必要なアクションを示しています。

要素タイプ	変更のタイプ	必要なアクション
キューブ (ルート要素)	[名前] または [ソースクラス] の編集	リコンパイルおよび再構築
フィルタ値	キューブに適用するがキューブ内の特定の要素には適用しないその他の変更	リコンパイル

要素タイプ	変更のタイプ	必要なアクション
メジャー	既存のメジャーの以下のオプションの編集（その他多くの要素にこれらの共通オプションの一部または全部があります） · 無効 · [隠し] · 表示名 · 説明 · 書式文字列	リコンパイル
	メジャーの削除	リコンパイル
	メジャーの追加など、その他のすべての変更	変更されたメジャーのリコンパイルおよび選択的構築の実行、またはリコンパイルおよびキューブの完全再構築の実行
ディメンジョン（計算ディメンジョン以外）	既存のディメンジョンの以下のオプションの編集 · メジャーの場合と同じ共通オプション · このディメンジョンの All レベルの有効化 · All メンバのキャプション · All メンバの表示名	リコンパイル
	ディメンジョンの削除†	リコンパイル
	ディメンジョンの追加など、その他のすべての変更	変更されたディメンジョンを構成するレベルのリコンパイルおよび選択的構築の実行、またはリコンパイルおよびキューブの完全再構築の実行
計算ディメンジョン	すべての変更	リコンパイル
NLP ディメンジョン	すべての変更	リコンパイル
階層	既存の階層の共通オプションの編集（メジャーの場合と同様）	リコンパイル
階層	階層の追加や削除など、その他のすべての変更	リコンパイル

要素タイプ	変更のタイプ	必要なアクション
レベル	既存のレベルの以下のオプションの編集 · メジャーの場合と同じ共通オプション · ヌル置換文字列 · 並べ替えオプション	リコンパイル
	レベルの削除†	リコンパイル
	レベルの追加など、その他のすべての変更*	変更されたレベルのリコンパイルおよび選択的構築の実行、またはリコンパイルおよびキューブの完全再構築の実行
プロパティ	既存のプロパティの以下のオプションの編集 · メジャーの場合と同じ共通オプション · プロパティ値を基準にしたメンバの並べ替え	リコンパイル
プロパティ	プロパティの追加や削除など、その他のすべての変更	リコンパイルおよび再構築
リスト	すべての変更	リコンパイル
計算メンバ	すべての変更	リコンパイル
名前付きセット	すべての変更	リコンパイル
サブジェクト領域	すべての変更	リコンパイル
複合キューブ（サブジェクト領域の一種）	すべての変更	リコンパイル（複合キューブ内で使用されるすべてのキューブのリコンパイル後）
品質メジャー	すべての変更	品質メジャー・クラスのリコンパイル
KPI またはプラグイン	すべての変更	KPI クラスまたはプラグイン・クラスのリコンパイル

*現在のサーバ・ロケールによって時間ディメンジョンのメンバの名前が決まります。(”ロケールを使用した時間メンバ名の制御” を参照してください。)ロケールを変更した場合は、キューブをリコンパイルおよび再構築する必要があります。

†ディメンジョンまたはレベルを削除してリコンパイルしても、関連付けられているレベル・テーブルおよびインデックスは削除されません。また、キューブの再構築によって、不要になったレベル・テーブルおよびインデックスが削除されることもありません。

6.2 キューブのコンパイル

アーキテクトでキューブ・クラスをコンパイルする手順は以下のとおりです。

1. [コンパイル] をクリックします。

   クラスのコンパイルが開始され、進捗状況を示すダイアログ・ボックスが表示されます。

   保存していない変更がある場合は、キューブをコンパイルする前に、システムによって変更が保存されます。

2. [OK] をクリックします。

または、スタジオでキューブ・クラスを開き、他のクラスのコンパイルと同じ方法でこれをコンパイルします。

キューブ・クラスをコンパイルする際には、必要に応じてファクト・テーブルとすべての関連クラスが自動的に生成されます。ファクト・テーブルが既に存在する場合、構造を変更する必要がある場合にのみ、システムによりファクト・テーブルが再生成されます。

このキューブにキャッシュされた結果が存在する場合、これらはシステムにより削除されます。

6.3 キューブの構築

キューブの構築という用語は、ファクト・テーブルおよびその他のテーブルへのデータの追加と、このデータへのアクセスに使用されるインデックスの構築の 2 つのタスクを指します。

アーキテクトでキューブの完全構築を実行する手順は以下のとおりです。

1. [ビルド] をクリックします。

   ダイアログ・ボックスが表示されます。

   [ビルド] オプションがグレー表示されていることがあります。その場合は、構築を実行する前にキューブをコンパイルする必要があります。

2. 必要に応じて、[構築する最大レコード数] の値を指定します。

   既定では、システムは、ソース・テーブル内のすべてのレコードの繰り返し処理を行い、同じ数のレコードをファクト・テーブルに構築します。キューブ作成時にこの動作をオーバーライドできます。[構築する最大レコード数] オプションを指定すると、システムはその数のレコードだけを繰り返し処理します。その結果、ファクト・テーブルは小さくなり、システムによる構築時間が短くなります。

   [構築する最大レコード数] フィールドがある数値で初期化されると、キューブ・クラスは既定の動作をオーバーライドします (詳細は、"キューブ・クラスのリファレンス情報" の "<cube>" で maxFacts 属性を参照してください)。この場合、キューブ・クラスで指定された値を使用するか、またはより小さい値を入力できます。

3. ダイアログ・ボックスの [構築オプション] セクションで [すべて構築] を選択します。

4. [ビルド] をクリックします。

   キューブの構築が開始され、その進捗状況が表示されます。構築中、[コンパイル] ボタンは無効になっています (グレー表示)。

   注釈　構築プロセス中に [閉じる] ボタンをクリックしても、構築プロセスは中断されません。いつでも再度開いて、現在進行中のビルドの現在の状態を確認できます。ダイアログが閉じている間に構築が完了した場合、構築の完了を知らせるためにダイアログが再び表示されます。

5. [完了] をクリックします。

これで、"アナライザの使用法" で説明されているようにキューブの使用が可能になります。

6.4 選択的構築の使用

選択的構築機能を使用すると、キューブ全体を再構築することなく、また再構築に伴うダウンタイムが発生することなく、キューブ内の特定の要素を構築できます。例えば、最近、特定のディメンジョンに変更を加えた場合、または 1 つのディメンジョンにのみ影響するソース・データの変更がある場合、選択的ビルドを使用して、そのディメンジョン内の関連するレベルを構築できます。選択的構築を使用して、最近追加されたレベル、メジャー、またはリレーションシップを構築することもできます。

選択的構築を使用して、キューブ内の特定のレベル、メジャー、またはリレーションシップを構築できます。すなわち、選択的構築を使用すると、ファクト・テーブル内の列（それぞれがレベル、メジャー、またはリレーションシップに対応します）が構築されます。キューブで選択的構築を使用しても、そのキューブの従属キューブを更新する必要はありません。

キューブの定義の要素を他のキューブから継承することがあります。他のキューブから継承したキューブに対して選択的構築が有効になっている場合、継承するキューブはスーパーキューブ定義内の指定された factNumbers を読み取り、factNumbers をサブキューブ定義に適切に割り当てることができます。サブキューブは、スーパーキューブの factNumbers が同じままであるとは想定していないため、自身のすべての factNumbers を再生成します。スーパーキューブには、現在のキューブ内のコンパイル済みの factNumber と競合する factNumber が割り当てられている可能性がありますが、これにより、スーパーキューブでの変更から現在のキューブを保護できます。

6.4.1 選択的構築の影響

選択的構築の実行中は、構築中のキューブ要素のみ、クエリに使用できなくなります。選択的構築とキューブの同期を同時に実行することはできないため、キューブは完全構築のように完全に無効ではないとはいえ、選択的構築の実行中に、キューブの同期は実行できません（逆の場合も同様です）。多大な構築操作を実行すると、計画されている同期がブロックされる可能性があります。さらに、選択的構築は完全構築よりも時間がかかるため、それ相応の時間を割り当てる必要があります。

重要　選択的構築では、主な構築手順の最後に、キューブの同期が試行されます。同期によって、現在のソース・データとファクト・テーブル内の選択的構築で除外した列との不一致を回避できます。

このため、選択的構築を使用するのは、同期が可能なキューブのみにすることをお勧めします。同期が不可能なキューブに選択的構築を実行する場合は、選択的構築で除外した列の精度を確保するために、続いて完全構築を実行する必要があります。

複数の選択的構築を同時に実行することができます。この場合、各選択的構築では、選択したキューブ要素のみが構築されます。選択的構築では複数の列を一度に構築できますが、列を複数回同時に構築することはできません。

選択的構築のエラーは、キューブの完全構築のエラーと同様に処理されます。

特定のレベルまたはメジャーの値に基づいてキューブ内のファクトを条件付きで処理するために %OnProcessFact() を実装する場合、そのレベルまたはメジャーを、そのキューブの選択的構築に含める必要があります。そうしないと、選択的構築でエラーが生じます。

6.4.2 アーキテクトでの選択的構築の使用

選択的構築は、すべてのキューブに対して自動的に有効になります。選択的構築を使用する前に、キューブをコンパイルする必要があります。

以下の手順に、選択的構築機能の使用例を示します。

1. アナライザに移動して、HoleFoods キューブを開きます。
2. [モデル・コンテンツ] ペインで `Outlet` ディメンジョンを拡張してから、`Region` レベルを拡張します。`Region` レベルを [行] 領域にドラッグします。生成されたピボット・テーブルを確認します。

3. 次に、アーキテクトを開きます。モデル・ビューワで Outlet ディメンジョンの Region レベルをクリックします。
4. 右側の**詳細領域**の [**ソース値**] で、[**式**] を選択します。[**式**] テキスト・ボックスに以下を入力します。

```
%source.Outlet.Country.Region.%ID _ "-" _ %source.Outlet.Country.Region.Name
```

5. HoleFoods キューブをコンパイルします。
6. [**ビルド**] をクリックします。[**キューブの構築**] ダイアログが表示されたら、[Outlet].[H1].[Region] レベルが変更されたことが自動的に検出され、[Outlet].[H1].[Region] に選択的構築が事前に選択されることに注意してください。[**ビルド**] をクリックします。
7. アナライザに戻ります。[モデル・コンテンツ] ペインで Outlet ディメンジョンを拡張してから、Region レベルを拡張します。Region レベルを [**行**] 領域にドラッグします。生成されたピボット・テーブルを確認し、Region レベルの違いに注目します。

6.5 プログラムによるキューブのビルド

キューブをプログラムでビルドするには、**%DeepSee.Utils** クラスの %BuildCube() クラス・メソッドを実行します。このメソッドには、以下のシグニチャがあります。

```
classmethod %BuildCube(pCubeName As %String, pAsync As %Boolean = 1,  pVerbose As %Boolean = 1,
                       pIndexOnly As %Boolean = 1, pMaxFacts As %Boolean = 0, pTracking As %Boolean
= 0,
                       ByRef pBuildStatistics As %String = "", pFactList As %String) as %Status
```

以下は、この指定の説明です。

- pCubeName は、XData ブロックで指定されているキューブの論理名です。この名前は大文字と小文字が区別されません。
- pAsync は、システムが複数のバックグラウンド・プロセスでビルドを実行するかどうかを制御します。この引数が true の場合、システムは複数のプロセスを使用し、それらがすべて完了するまで戻りません。この引数が false の場合、システムは単一のプロセスを使用し、それが完了するまで戻りません。

  注釈　キューブ・オプションの [**初期ビルド順**] を指定している場合、システムは pAsync の値を無視し、単一プロセスを使用してキューブをビルドします。これらのオプションについては、"キューブ・オプションの指定" で説明します。

  警告　ビルド・プロセスのカスタム・コードで HALT を呼び出すことはできません。DeepSee エージェントを終了すると、優先度が低いエージェントの終了の連鎖が発生し、新規タスクに使用できるエージェントの不足により、ビルドが停止することがあります。

- pVerbose は、メソッドがステータス情報を書き込むかどうかを制御します。この引数が 1 の場合は、コマンド行に状態の更新が書き込まれます。(この引数は、メソッドが構築エラーやその他のロギング情報を書き込むかどうかに影響しません。)
- pIndexOnly は、メソッドがインデックスのみを更新するかどうかを制御します。この引数が 1 の場合、ファクト・テーブルのインデックスのみが更新されます。
- pMaxFacts は、ファクト・テーブルの最大行数を指定します。これによって、キューブの構築時に使用されるベース・テーブルの行数が決まります。

  pMaxFacts が 0 の場合 (既定)、ベース・テーブルのすべての行が処理されます。
- pTracking は内部用です。

- pBuildStatistics は、キューブの構築に関する情報の配列を返します。この配列には以下のノードがあります。

ノード	値
pBuildStatistics(“elapsedTime”)	構築の経過時間 (秒単位)。
pBuildStatistics(“errors”)	構築エラーの数。
pBuildStatistics(“factCount”)	構築およびインデックス付けされたファクトの数。
pBuildStatistics(“missingReferences”)	欠落している参照の数。
pBuildStatistics(“expressionTime”)	キューブ要素を構築する sourceExpressions の処理にかかった時間。
pBuildStatistics(“iKnowTime”)	NLP インデックスの構築にかかった時間。

- pFactList は、キューブのファクト・クラスの特定のプロパティ名のリストです。pFactList が提供される場合、ビルドではそのファクト・リストに表示されている列のみが更新されます。このリストは、コンマ区切りまたは $LB 形式です。更新中の特定のファクトは、クエリに利用不可と個別にマークされ、これらのファクトに基づいてディメンジョンを参照するクエリを準備すると、エラーが返されます。

このメソッドはステータスを返します。キューブの構築中にエラーが発生した場合は、ステータス・コードにより構築エラーの数が示されます。

以下はその例です。

ObjectScript

```
set status = ##class(%DeepSee.Utils).%BuildCube("patients")
```

このメソッドは、以下の情報を示す出力を記述します。

- 使用されるプロセッサの数。
- 構築プロセスに要する経過時間の合計。
- ソース式を評価するために費やした時間の合計 (すべてのプロセッサの総合計)。

以下はその例です。

```
Building cube [patients]
Existing cube deleted.
Fact table built:        1,000 fact(s) (2 core(s) used)
Fact indexes built:      1,000 fact(s) (2 core(s) used)
Complete
Elapsed time:                  1.791514s
Source expression time:        0.798949s
```

`Source expression time` が大きすぎると考えられる場合は、ソース式ができる限り効率的であることを再検査する必要があります。特に、その式で SQL クエリを使用している場合は、そのクエリが使用するテーブルに適切なインデックスがあることを再度確認してください。

6.5.1 キューブ構築ステータス

進行中の構築がある場合、%BuildStatus() メソッドを使用してその進行状況を監視できます。ターミナルで、次のように呼び出します。

ObjectScript

```
DO ##class(%DeepSee.Utils).%BuildStatus("cubeName")
```

構築の開始にプログラムを使用した場合でも、アーキテクトを使用した場合でも、%BuildStatus() から構築の進行状況がレポートされます。進行中の構築がない場合、%BuildStatus() は最近の構築のタイムスタンプを表示します：

```
06/23/2020 11:31:07
```

プログラムで開始された構築の進行状況は、アーキテクトの構築ダイアログからもレポートされます。

6.6 開発におけるキューブ・サイズの最小化

キューブの開発時は、一般にリコンパイルと再構築を頻繁に行います。大量のデータ・セットを使用している場合は、キューブをより迅速に再構築するため、ファクト・テーブル内のファクト数を制限することがあります。これを行うには、以下のいずれかを実行します。

- アーキテクトでキューブを構築する場合は、[**構築する最大レコード数**] の値を指定します。
- スタジオでキューブ・クラスを編集して、`maxFacts` 属性を `<cube>` 要素に追加します。"キューブ・クラスのリファレンス情報" を参照してください。

  これを実行する場合は、この属性を導入前に確実に削除してください。
- ターミナルでキューブを構築して、pMaxFacts 引数を指定します。"プログラムによるキューブのビルド" を参照してください。

選択的構築時は、これらのオプションはすべて無視されます。

6.7 キューブの構築時の並行処理の使用法

以下のすべての項目が該当する場合、システムはビルドの実行に複数のコアを使用します。

- キューブの構築時に pAsync を 1 に指定している ("プログラムによるキューブのビルド" を参照)。
- キューブのソースが永続クラスである (データ・コネクタではない)。データ・コネクタについては、"InterSystems Business Intelligence の実装" で説明しています。
- 永続クラスがビットマットに適している。
- キューブの [**初期ビルド順**] オプションが設定されていない。これらのオプションについては、"キューブ・オプションの指定" で説明します。

キューブを非同期的に構築するときに、並列処理が使用できる場合、その作業を実行するために、システムによって **%SYSTEM.WorkMgr** エージェントが設定されます。

注釈　これらのエージェントは、クエリの実行にも使用されます。

まれに、これらのエージェントのリセットが必要になることがあります。この操作には、**%DeepSee.Utils** の %Reset() メソッドを使用します。また、このメソッドは、保留中のタスクを消去して、現在のネームスペースの結果キャッシュを消去します。これらは、すべてのユーザに直ちに影響を与えます。このメソッドは、開発時にのみ使用するように意図されています。

6.8 構築エラー

キューブの構築の際は、表示されるエラー・メッセージ、および構築されるファクトとインデックスの数に注意してください。このセクションでは、以下の項目について説明します。

- 構築エラーを確認できる場所
- ファクト数 (全シナリオの構築に関する問題の役立つインジケータ)
- 考えられる構築エラーの原因
- 構築エラーから回復する方法

トラブルシューティング・オプションの詳細は、InterSystems Developer Community を参照してください。

6.8.1 構築エラーの確認

アーキテクトまたはターミナルでキューブを構築する際、構築エラーがあるかどうかが示されますが、それらのすべてが表示されるわけではありません。記録された構築エラーをすべて確認するには、以下のいずれかを実行します。

- ログ・ファイル install-dir/**mgr/DeepSeeUpdate**_cube_name_NAMESPACE.**log** を探します。cube_name はキューブ名で、NAMESPACE はこのキューブが定義されているネームスペースです。

  このファイルのタイム・スタンプでは $NOW を使用してローカルの日時が記録されます。サマータイムは無視されます。

- 以下のように、**%DeepSee.Utils** の %PrintBuildErrors() メソッドを使用します。

  ```
  do ##class(%DeepSee.Utils).%PrintBuildErrors(cubename)
  ```

  cubename はキューブの論理名で、引用符で囲みます。

  このメソッドはすべての構築エラーに関する情報を表示します。以下に例を示します (改行が追加されています)。

  ```
  SAMPLES>do ##class(%DeepSee.Utils).%PrintBuildErrors("holefoods")
      1 Source ID: 100000
        Time: 05/09/2019 14:12:52
        ERROR #5002: ObjectScript error: <DIVIDE>%UpdateFacts+106^HoleFoods.Cube.Fact.1
      2 Source ID: 200000
        Time: 05/09/2019 14:12:41
        ERROR #5002: ObjectScript error: <DIVIDE>%UpdateFacts+106^HoleFoods.Cube.Fact.1
      3 Source ID: 300000
        Time: 05/09/2019 14:13:13
        ERROR #5002: ObjectScript error: <DIVIDE>%UpdateFacts+106^HoleFoods.Cube.Fact.1
      ...
      10 build error(s) for 'holefoods'
  ```

重要　　システムによりエラーが生成されない場合もあるため、ファクト数の確認も重要です (次のセクションを参照)。

6.8.2 ファクト数の確認

キューブの構築の際は、構築されるファクトとインデックスの数が報告されます。

各ファクトは、ファクト・テーブル内のレコードです。以下の場合を除き、ファクト・テーブルには、ベース・テーブルと同じ数のレコードが存在します。

- ファクト数は、このページで前述した説明のとおりに制限できます。

- キューブ・クラスでは、%OnProcessFact() コールバックも定義します。このコールバックは、キューブからレコードを除外するために使用できます。"キューブおよびサブジェクト領域の高度な機能の使用" を参照してください。

また、システムでインデックスが構築されるときには、インデックス数がファクト・テーブルのレコード数と同じになります。例えば、アーキテクトでは [**ファクトをビルド中**] と [**インデックスをビルド中**] に同じ数が示されます。これらの数に不一致がある場合は、ログ・ファイルを確認してください。

6.8.3 考えられる構築エラーの原因

構築エラーやファクト数の原因不明の不一致が存在する場合は、以下の手順を実行します。

1. 範囲式を使用するすべてのレベルを調べ、これらのレベルでレコードが削除されていないことを検証します。"レベルの検証" を参照してください。

   この種のエラーは、インデックス数に影響しますが、ファクト数には影響しません。

2. 選択したディメンジョンまたはメジャーを無効にします。その後、リコンパイルおよび再構築を行い、問題が発生するディメンジョンまたはメジャーを隔離します。

6.8.3.1 <STORE> エラー

構築ログに、以下のようなエラーが記載される場合があります。

```
ERROR #5002: ObjectScript error: <STORE>%ConstructIndices+44^Cube.cube_name.Fact.1
```

このエラーは、レベルに非常に大量のメンバが存在する場合に発生します。既定では、システムはインデックスを構築する際、ローカル・メモリを使用してインデックスをチャンクで格納し、その後それらをディスクに書き込みます。レベルに非常に大量のメンバが含まれる場合は、ローカル・メモリが不足し、<STORE> エラーが発生することがあります。

このようなエラーを回避するには、以下のいずれかの操作を実行します。

- キューブを単一のプロセスで構築する。そのためには、ターミナルで %BuildCube() を使用し、2 番目の引数に 0 を使用します。
- <cube> 要素で、bitmapChunkInMemory="false" (既定) を指定する。バックグラウンド・プロセスを使用してこのキューブを構築すると、ローカル変数ではなくプロセス・プライベート・グローバルが使用されます (ローカル・メモリによる制限がなくなります)。

6.8.3.2 参照の欠落エラー

キューブが他のキューブへのリレーションシップを持つ場合、構築ログに、以下のようなエラーが記載される場合があります。

```
ERROR #5001: Missing relationship reference in RelatedCubes/Patients: source ID 1 missing reference to
 RxHomeCity 4
```

このエラーは、キューブが間違った順序で構築されていることを示す可能性があります。"リレーションシップを持つキューブの構築" を参照してください。キューブ・マネージャを使用する場合、キューブ・マネージャが適切な構築順序を決定することに注意してください。

`missing relationship reference` エラーは、キューブの構築プロセス中に新しいソース・データが使用可能になる場合、つまり既にキューブのいくつかが構築された後にも発生する可能性があります。例えば、サンプル・キューブ RelatedCubes/Cities と RelatedCubes/Patients について考えてみましょう (これらは SAMPLES ネームスペースで使用可能です)。キューブ RelatedCubes/Cities を構築した後、RelatedCubes/Patients のソース・テーブルが、新しい都市を使用するレコードを受け取るとします。この場合、キューブ RelatedCubes/Patients を構築すると、`missing relationship reference` エラーが発生します。

キューブを正しい順序で構築したことが確実である場合、エラーからの回復の詳細について、次のセクションを参照してください。

6.8.4 構築エラーからの回復

キューブ全体を再構築するのではなく、以前に構築エラーを生成したレコードのみ再構築する方法が用意されています。以下はその方法です。

1. これらのエラーの原因となる問題を修正します。
2. 以下のように、**%DeepSee.Utils** の %FixBuildErrors() メソッドを使用します。

```
set sc=##class(%DeepSee.Utils).%FixBuildErrors(cubename)
```

cubename はキューブの論理名で、引用符で囲みます。このメソッドは、進捗状況メッセージを表示するかどうかを指定する 2 番目の引数を受け入れます。この引数の既定値は True です。

例えば以下のように出力されます。

```
Fact '100' corrected
Fact '500' corrected
Fact '700' corrected


3 fact(s) corrected for 'patients'
0 error(s) remaining for 'patients'
```

または、キューブ全体を再構築します。

6.9 Business Intelligence タスク・ログ

システムは、(前述した構築ログ以外に) 他のログ・ファイルも生成します。キューブの構築後またはキューブの構築を試行後、**DeepSeeTasks**_NAMESPACE.**log** ファイルもディレクトリ install-dir/**mgr** に書き込まれます。**%DeepSee.WorkMgr** クラスの %SetLoggingOptions メソッドを使用して、構築プロセス中にシステムが使用したバックグラウンド・エージェントのログを有効にすることができます。それには、以下のような呼び出しを行います。

```
do ##class(%DeepSee.WorkMgr).%SetLoggingOptions(,,1)
```

このファイルを管理ポータルから参照するには、[Analytics] > [**管理**] > [**ログ**] の順に選択します。

Tipヒン このファイルには、リストのエラーや KPI エラーなど、さまざまな種類の実行時エラーに関する情報も含まれます。ト

このファイルのタイム・スタンプではローカルの日時を使用します (サマータイムは考慮されます)。

6.10 関連トピック

以下の関連トピックにも注意してください。

- キューブの同期。このプロセスでは、キューブがインクリメンタルに更新されます (また、クエリの同時実行が許可されます)。"キューブの最新状態の維持" を参照してください。

- InterSystems Business Intelligence キューブ・マネージャ・ユーティリティ。これは通常、プロダクション・システムで使用します。キューブ・マネージャは、指定に従ってキューブを構築または同期する自動化タスクを作成します。"キューブの最新状態の維持" を参照してください。
- キューブ・バージョン機能。この機能を使用すると、実行中クエリのわずかな中断しか伴わずに、キューブ定義の変更、構築、ユーザへの提供を行うことができます。"キューブ・バージョンの使用" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

7

ディメンジョン、階層およびレベルの定義

ここでは、Business Intelligence キューブでディメンジョンと階層を定義する方法、およびレベルの定義に関する基本的な事項について説明します。

Business Intelligence には、レベルに影響を及ぼす多くのオプションが用意されています。"レベルの定義の詳細" も参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

7.1 概要

アーキテクトでは、以下の 2 つの方法のいずれかでモデル要素を作成できます。

- ドラッグ・アンド・ドロップ。クラス・ビューワからモデル・ビューワ内の適切なターゲットにプロパティをドラッグできる場合は、このリストの後で説明されているように、アーキテクトでモデル要素が作成されます。その後、必要に応じて詳細領域で定義を編集できます。

  この方法によって、ソース・プロパティに直接基づいて要素を容易に定義できます。これは、ソース式に基づいて要素を作成する必要がある場合の開始ポイントとしても役立ちます。

- **[要素を追加]** リンクの使用。

参照用として、次のテーブルに各種ドラッグ・アンド・ドロップ操作の結果を示します。

プロパティ XYZ のドロップ先	アーキテクトでの作成内容
[メジャー] の見出しまたは既存のメジャー	一意性の必要に応じて、XYZ、XYZ1、などの名前が付けられたメジャー。このメジャーは、ソース・プロパティ XYZ に基づき、SUM によって集約されます。"メジャーの定義" を参照してください。
[ディメンジョン] の見出し	一意性の必要に応じて、XYZ、XYZ1、などの名前が付けられた新しいデータ・ディメンジョン。このディメンジョンには階層 H1 が組み込まれ、この階層にはレベル XYZ が組み込まれます。このレベルは、ソース・プロパティ XYZ に基づきます。
既存のディメンジョン	新規の階層 (例 : H2)。この階層にはレベル XYZ が組み込まれます。このレベルは、ソース・プロパティ XYZ に基づきます。
既存の階層	階層内の一意性の必要に応じて、XYZ、XYZ1、などの名前が付けられた新規レベル。このレベルは、ソース・プロパティ XYZ に基づきます。
既存のレベル	レベル内の一意性の必要に応じて、XYZ、XYZ1、などの名前が付けられた新規レベル・プロパティ。このレベル・プロパティは、ソース・プロパティ XYZ に基づきます。"プロパティの定義" を参照してください。

7.2 新規ディメンジョン、階層およびレベルの作成

使用可能なレベルを定義するには、少なくとも以下の定義が必要です。

- ディメンジョン
- そのディメンジョン内の階層
- その階層内のレベル

アーキテクトでは、"概要" で説明されているように、ドラッグ・アンド・ドロップ操作を使用できます。または、以下の操作を実行できます。

1. ディメンジョンを追加します。
   a. **[要素を追加]** をクリックします。

      ダイアログ・ボックスが表示されます。

   b. **[新規項目名の入力]** に、ディメンジョン名を入力します。

      "モデル要素の名前" を参照してください。

   c. 作成するディメンジョンのタイプに応じて、次の選択肢のいずれか 1 つをクリックします。
      - **[データ・ディメンジョン]** – ほとんどのディメンジョンではこれをクリックします。
      - **[時刻ディメンジョン]** – 日付または時刻の値別にデータをグループ化するディメンジョンを作成する場合に、これをクリックします。詳細は、このページで後述する "時間レベルの定義" を参照してください。
      - **[年齢ディメンジョン]** – 日付の値に基づいて年齢別にデータをグループ化するディメンジョンを作成する場合に、これをクリックします。通常、年齢ディメンジョンはお勧めしません。詳細は、このページで後述する "年齢レベルの定義" を参照してください。

      [iKnow ディメンジョン] オプションの詳細は、"キューブでの Text Analytics の使用法" を参照してください。

   d. [OK] をクリックします。

      ディメンジョンが作成され、モデル・ビューワにこれが表示されます。以下はその例です。

システムによってこのディメンジョン内に階層が作成され、さらにその階層内にはレベルも作成されていることに注意してください。

2. より適切な名前を使用するように、自動作成されたレベルを変更します。
   a. モデル・ビューワで、レベルをクリックします。

      詳細領域に詳細が表示されます。

   b. 以下のように変更します。
      - **[名前]** – 必要なレベル名に変更します。

        "モデル要素の名前" を参照してください。
      - **[表示名]** – このレベルのローカライズされた表示名を指定します。これをクリアする (代わりに **[名前]** に指定された値を使用) か、表示名を指定します。

      "その他の共通オプション" も参照してください。
3. "ディメンジョンまたはレベルのソース値の定義" の説明に従って、このレベルのソース値を指定します。
4. **[保存]** をクリックします。
5. プロンプトが表示されたら、[OK] をクリックします。

注釈　既定では、ディメンジョンに複数の階層が含まれていない限り、アナライザは階層名を表示しません。代わりに、階層名を必ず表示する、または完全に非表示にするようにディメンジョンを定義できます。"キューブ・クラスのリファレンス情報" の "<dimension>" で、`showHierarchies` のリファレンスを参照してください。

7.3 階層およびレベルの追加

アーキテクトでは、既存のディメンジョンに階層およびレベルを追加するには、"概要" で説明しているとおりドラッグ・アンド・ドロップ操作を使用できます。または、以下の操作を実行できます。

1. モデル・ビューワで、ディメンジョンをクリックします。
2. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。
3. **[新規項目名の入力]** に、階層名を入力します。

   "モデル要素の名前" を参照してください。
4. **[階層]** をクリックします。
5. [OK] をクリックします。

   階層が作成され、モデル・ビューワにこれが表示されます。また、その階層内にレベルが 1 つ作成されます。
6. 必要に応じて、モデル・ビューワで階層を選択し、詳細領域で詳細を編集します。
7. モデル・ビューワでレベルを選択し、詳細領域で詳細を編集します。

7.4 レベルの追加

アーキテクトでは、既存の階層にレベルを追加するには、"概要" で説明しているとおりドラッグ・アンド・ドロップ操作を使用できます。または、以下の操作を実行できます。

1. モデル・ビューワで、階層または階層内の既存レベルをクリックします。

   この操作によって、レベルを追加する場所を示します。

   - 階層をクリックすると、その階層の他のすべてのレベルの後に新しいレベルが追加されます。
   - レベルをクリックすると、そのレベルの直前に新しいレベルが追加されます。

2. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

3. **[新規項目名の入力]** に、レベル名を入力します。
4. **[レベル]** をクリックします。
5. [OK] をクリックします。

   レベルが作成され、モデル・ビューワにこれが表示されます。

6. 必要に応じて、モデル・ビューワでレベルを選択し、詳細領域に示された詳細を編集します。

または、クラス・ビューワからクラス・プロパティをドラッグして、これを階層にドロップします。このプロパティに基づいてレベルが追加されます。このレベルはその階層内の他のすべてのレベルの後に追加されます。

重要　"階層の適切な定義" で述べているように、階層内のレベルの順序によって階層の構造が決まります。定義後にレベルの順序を変更する場合は、"階層内のレベルの順序の変更" を参照してください。

7.5 ディメンジョンまたはレベルのソース値の定義

各レベルにソース値が指定されている必要があります。ディメンジョン内またはレベル内のソース値を指定できます。通常は、以下のとおりです。

- データ・ディメンジョンにはレベル内のソース値を指定します。
- 時間および年齢ディメンジョンには、ディメンジョン内のソース値を指定します。
- NLP ディメンジョンについては、"キューブでの Text Analytics の使用法" を参照してください。NLP ディメンジョンは、ここで説明したメカニズムを使用しません。

アーキテクトでソース値を指定する手順は以下のとおりです。

1. モデル・ビューワで、ディメンジョンまたはレベルを選択します。
2. 詳細領域で、以下のいずれかの値を指定します。
   - **[プロパティ]** – キューブで使用されるベース・クラスに関連するプロパティ名を指定します。ドット構文を使用してプロパティのプロパティを参照できます。以下はその例です。

     ```
     Age
     ```

別の例を示します。

```
HomeCity.PostalCode
```

このプロパティは、SQL へのプロジェクションが必要です。

また、オブジェクト値プロパティを参照することもできます。その場合はオブジェクトの数値 ID がソース値として使用されます。

ストリーム・プロパティを直接使用することはできません (この種のプロパティを使用するには、ストリームのコンテンツまたはコンテンツの選択部分を返す式を作成します)。

- [式] – レベルの基となる ObjectScript 式を指定します。以下はその例です。

```
##class(Cubes.StudyPatients).GetFavoriteColor(%source.PatientID)
```

この式は、キューブが構築されるときに評価されます。詳細は、次のセクションを参照してください。

[プロパティ] および [式] フィールドには、以下のいずれかの方法で値を入力できます。

- クラス・ビューワからクラス・プロパティをドラッグして、これをフィールドにドロップする。

  ドラッグ・アンド・ドロップしたプロパティによって、フィールドの既存コンテンツが置換されます。

- 検索ボタン  をクリックする。
  - [プロパティ] の場合は、そのクラスのプロパティを示すダイアログ・ボックスが表示されます。プロパティをクリックしてから、[OK] をクリックします。
  - [式] の場合は、入力が可能な大きいフィールドのあるダイアログ・ボックスが表示されます。式を入力してから、[OK] をクリックします。

- フィールド内で値を直接編集する。

このセクションおよび次のセクションの内容は、やはりある形式のソース値を必要とするレベル・プロパティおよびメジャーにも適用されます。

7.5.1 データ・コネクタを使用する場合のソース値の指定

キューブがデータ・コネクタに基づく場合、以下の制約に注意してください。

- [式] オプションではなく、[プロパティ] オプションを指定します。
- ドット構文は使用できません。これは、データ・コネクタのいずれのプロパティもオブジェクト参照ではないためです。

データ・コネクタの作成の詳細は、"データ・コネクタの定義" を参照してください。

7.6 ソース式の詳細

前のセクションで述べているように、ディメンジョンまたはレベル (またはプロパティまたはメジャー) の基礎として使用するソース値またはソース式を指定できます。ソース式は以下のように作成できます。

- ソース・クラスでプロパティを参照できます。そのためには、構文 `%source.`PropertyName を使用します。PropertyName はプロパティの名前です。この式は、キューブの構築時に解析され、指定されたプロパティの SqlFieldName が検索されます。

- ドット構文を使用すると、プロパティのプロパティを参照できます。
- オブジェクト値プロパティを参照できます。その場合はオブジェクトの数値 ID がソース値として使用されます。
- 変数 %cube を使用すると、キューブ・クラスを参照できます。これはソース式内で使用するキューブ・クラスでユーティリティ・メソッドを定義している場合に役立ちます。
- <expression> 要素の値を参照できます。そのためには、以下の構文を使用します。

  ```
  %expression.expressionName
  ```

  式の詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。
- 以下の例で示すように、ユーティリティ・メソッド ToUpper()、ToLower()、および Log() を使用できます。

  ```
  ..ToUpper(%source.HomeCity.Name)
  ```

  ファクト・テーブル・クラスは、これらのメソッドを定義する %DeepSee.CubeFunctionSet から継承されるため、システムはこの方法でメソッドを使用できます。これらのメソッドの詳細は、該当するクラスのクラスリファレンスを参照してください。
- %DeepSee.CubeDefinition の %Lookup() メソッドを使用して、用語リストを起動できます。サブセクション "条件リストの使用" を参照してください。
- %DeepSee.CubeDefinition の %Rule() メソッドを使用して、プロダクション・ビジネス・ルールを起動できます。サブセクション "プロダクション・ビジネス・ルールの使用" を参照してください。

注釈　キューブがデータ・コネクタに基づいている場合、ソース式は使用できません。

また、あるレベルがソース式 (ソース・プロパティではなく) に基づいている場合に、メンバの名前が数字のみで構成されている場合は、スタジオでキューブを修正して、そのレベルの定義に castAsNumeric="true" を追加することをお勧めします。このオプションを追加すると、クエリで MDX 範囲式 (特定の種類のセット式) が使用される場合に、システムは、存在しないメンバの代替を検索する際にメンバを数値として扱うようになります。

7.6.1 条件リストの使用

条件リストにより、プログラミングせずに InterSystems IRIS Business Intelligence モデルをカスタマイズする方法が提供されます。条件リストは、キーと値のペアの単純なリスト (ただし、拡張可能) です ("条件リストの定義" を参照してください)。

ソース式内で用語リストを起動できます。この操作には、%Lookup() の %DeepSee.CubeDefinition メソッドを使用します。このメソッドには、以下のシグニチャがあります。

```
%Lookup(term_list_name, lookup_value,default,alternative_field)
```

引数は文字列です。それらの値は以下のとおりです。

- term_list_name は、用語リストの名前として評価されます。
- lookup_value は、用語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 が用語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、指定された用語リストを調査して、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。

すべての用語リストには、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。

注釈　キューブ定義クラスは、%Lookup() メソッドが定義されている %DeepSee.CubeDefinition から継承されるので、ソース式を以下のように使用できます。

```
%cube.%Lookup(term_list_name,lookup_value,default,alternative_field)
```

例えば、LocalTeams という名前の用語リストがあるとします。

Terms

key	value
Atlanta	Braves
Boston	Red Sox
New York	Yankees

以下のように、HoleFoods の City レベルにプロパティを追加できます。

```
<property name="Team" sourceExpression='%cube.%Lookup("LocalTeams",%source.Outlet.City,"No Team")'/>
```

7.6.2 プロダクション・ビジネス・ルールの使用

ビジネス・ルールを使用すると、専門知識を持たないユーザがプロダクションのビジネス・プロセスの動作を変更できるようになります。また、ビジネス・ルールは、キューブのソース式でも使用できます (プロダクション・ビジネス・ルールの詳細は、"ビジネス・ルールの開発" を参照してください)。

ソース式で使用されているプロダクション・ビジネス・ルールにアクセスするには、%DeepSee.CubeDefinition の %Rule() メソッドを使用します。このメソッドには、以下のシグニチャがあります。

```
%Rule(rule_name)
```

rule_name は、プロダクション・ビジネス・ルールの名前です。

この関数をキューブの構築時に (指定されたソース・レコードに対して) 評価する際、システムでは、この関数にコンテキスト・オブジェクトとしてキューブ・ソース・クラスのインスタンスを渡します。このオブジェクトを使用して、ルールが評価され、ルールから返された値にアクセスします。

注釈　キューブ定義クラスは、%Rule() メソッドが定義されている %DeepSee.CubeDefinition から継承されるので、ソース式を以下のように使用できます。

```
%cube.%Rule(rule_name)
```

7.7 キューブ内のディメンジョンの順序の変更

アーキテクトで、キューブ内のディメンジョンの順序を変更する手順は以下のとおりです。

1. **[順序変更]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[ディメンジョン]** をクリックします。

3. 必要に応じて、[**アルファベット順**] をクリックし、アルファベット順に配列します。

   これは、即座にリストに作用します。必要に応じて、リストをさらに再編成できます。また、ディメンジョンを追加する際に、アルファベット順の自動配列は行われません。

4. ディメンジョンの名前をクリックし、必要に応じて上矢印または下矢印をクリックします。
5. 必要に応じて、他のディメンジョンでもこの操作を繰り返します。
6. [OK] をクリックします。

キューブ内のディメンジョンの順序は、アナライザでの表示方法に影響を与えます。その他の影響はありません。ディメンジョンをアルファベット順に配列したほうが便利なユーザもいれば、使用頻度の高いディメンジョンをリストの上部に配置するユーザもいます。

7.8 階層内のレベルの順序の変更

アーキテクトで、階層内のレベルの順序を変更する手順は以下のとおりです。

- 階層内でレベルを上に移動するには、そのレベルの行で上矢印をクリックします。
- 階層内でレベルを下に移動するには、そのレベルの行で下矢印をクリックします。

重要　　"階層の適切な定義" で述べているように、階層内のレベルの順序によって階層の構造が決まります。

7.9 レベルの検証

レベルを定義したら、キューブの完全構築または選択的構築によって、それらのレベルを構築する必要があります。レベルの動作が適切であることを検証します。各レベルについて、アナライザを使用してレベルを行として表示する新しいピボット・テーブルを作成します。

1. [Analytics]→[**アナライザ**]→[**進む**] をクリックします。

   Tipヒン 既にアナライザが開いている場合は、ページ上部の [**アナライザ**] リンクをクリックするだけです。
   ト

2. 左領域に検証対象のキューブが表示されていない場合は、[**開く**] をクリックして、キューブを選択します。
3. 左領域で、レベルを含むディメンジョンを検索し、そのディメンジョンを展開します。
4. レベルを [**行**] ボックスにドラッグ・アンド・ドロップします。

このピボット・テーブルで、以下の項目を検索します。

- レベルに適正な数のメンバが存在することを確認します。

  レベルに同名の重複メンバがある場合は、"重複メンバ名が生成される状況" のセクションを参照してください。

  メンバに欠落がある場合は、管理ポータルを使用して、そのメンバに関連付けられているソース・レコードを検索します。ソース・プロパティがソース式で要求される形式であることを確認します。
- メンバ名が、ビジネス・ユーザの必要に応じて適切な形式であることを確認します。
- NULL メンバが存在する場合、ビジネス・ユーザの必要に応じて適切な名前があることを確認します。
- メンバが必要な順序で並べ替えられることを確認します。

確認できない場合は、前のセクションを参照してください。

- レベルで範囲を使用する場合は、以下の手順を実行して、範囲の端点に関するエラーでレベルのレコードが削除されないことを検証します。

  1. ピボット・オプション・ボタン  をクリックします。
  2. [**行オプション**] 領域で [**合計を表示**] をクリックして [OK] をクリックします。

  合計行にはその上にある値の合計が表示されます。この合計は、キューブ内のレコードの合計数と等しくなります (等しくならないように設定した場合を除く)。等しくない場合は、範囲を慎重に調べ、差異を見つけます。例えば、以下のようなピボット・テーブルがあるとします (10000 の患者を格納するキューブのピボット・テーブル)。

Age Group	
0 to 29	4,179
30 to 59	4,001
60+	1,573
Total	9,753

  このバージョンの `Age Group` レベルは、以下の範囲を使用します。

	From	To		Caption (Required)	
(		29	)	0 to 29	
(	30	59	)	30 to 59	✖
(	60		)	60+	✖

  29 歳、30 歳、59 歳または 60 歳の患者は、適切に定義されていないこのレベルでは、どのメンバにも含まれません。

Tipヒント アナライザでは、ピボット・テーブルの生成された MDX クエリにアクセスできます (アクセスするにはクエリ文字列ボタン  をクリックします)。その後、そのクエリをテキスト・ファイルに保存し、後からプログラムを使用してクエリを再実行できます。この方法によって、データが変更された場合に後からモデルを再検証することができます。新しいデータには、モデルの作成時に考慮されなかった値が含まれる場合があります。

プログラムを使用してクエリを実行する方法の詳細は、"InterSystems Business Intelligence の実装" を参照してください。

別の便利なツールとして、**%DeepSee.Utils** の %AnalyzeMissing() メソッドがあります。このメソッドは、指定されたキューブ内の非計算のレベル、メジャー、およびリレーションシップをすべて分析し、各々に値が存在しないファクトの数を返します。以下はその例です。

```
Do ##class(%DeepSee.Utils).%AnalyzeMissing("patients")

-------------------------------------------------------------------------------
    Level                                                      #Missing %Missing
Lvl [AgeD].[H1].[Age]                                                 0    0.00%
Lvl [AgeD].[H1].[Age Bucket]                                          0    0.00%
Lvl [AgeD].[H1].[Age Group]                                           0    0.00%
Lvl [AllerD].[H1].[Allergies]                                       388   38.80%
Lvl [AllerSevD].[H1].[Allergy Severities]                           388   38.80%
Lvl [BirthD].[H1].[Date]                                              0    0.00%
Lvl [BirthD].[H1].[Decade]                                            0    0.00%
Lvl [BirthD].[H1].[Period]                                            0    0.00%
Lvl [BirthD].[H1].[Quarter Year]                                      0    0.00%
Lvl [BirthD].[H1].[Year]                                              0    0.00%
Lvl [BirthQD].[H1].[Month]                                            0    0.00%
Lvl [BirthQD].[H1].[Quarter]                                          0    0.00%
Lvl [BirthTD].[H1].[Birth Time]                                       0    0.00%
Lvl [ColorD].[H1].[Favorite Color]                                  221   22.10%
Lvl [DiagD].[H1].[Diagnoses]                                        831   83.10%
```

```
Lvl [DocD].[H1].[Doctor]                                  159   15.90%
Lvl [DocD].[H1].[Doctor Group]                            271   27.10%
Lvl [GenD].[H1].[Gender]                                    0    0.00%
Lvl [HomeD].[H1].[City]                                     0    0.00%
Lvl [HomeD].[H1].[ZIP]                                      0    0.00%
Msr [Measures].[Age]                                        0    0.00%
Msr [Measures].[Allergy Count]                            388   38.80%
Msr [Measures].[Avg Age]                                    0    0.00%
Msr [Measures].[Avg Allergy Count]                        388   38.80%
Msr [Measures].[Avg Enc Count]                              0    0.00%
Msr [Measures].[Avg Test Score]                           200   20.00%
Msr [Measures].[Encounter Count]                            0    0.00%
Msr [Measures].[Test Score]                               200   20.00%
Lvl [PatGrpD].[H1].[Patient Group]                          0    0.00%
Lvl [PatGrpD].[H1].[Tested]                                 0    0.00%
Lvl [ProfD].[H1].[Industry]                               564   56.40%
Lvl [ProfD].[H1].[Profession]                             564   56.40%

    TOTAL FACTS                                         1,000
```

8

レベルの定義の詳細

ここでは、Business Intelligence キューブでレベルを定義する方法を詳しく説明します。

ソース値の指定に関する情報を含む基本情報については、"ディメンジョン、階層およびレベルの定義" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

8.1 メンバ・キーの一意性の保証

レベルに親レベルが存在する場合、メンバ・キーが一意でない 可能性があります。詳細は、"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。どのメンバにも容易に直接アクセスできるようにするため、必ずメンバ・キーを各レベルで一意にすることをお勧めします。また、メンバ・キーは最初の 113 文字内で一意である必要があります。

ソース値と階層を考慮し、重複メンバ・キーが存在する可能性がある場合は、以下の操作を実行します。

- ソース値のみを単独で使用する代わりに、ソース値と、親メンバに対して一意の他の値を連結します。例えば、`CityName` と呼ばれるソース・プロパティをレベルの基にし、異なる国に同名の市区町村が存在する可能性があるとします。**[プロパティ]** を指定するのではなく、**[式]** を指定します。以下のような式を使用します。

  ```
  %source.CountryName_%source.CityName
  ```

- このような連結文字列をメンバ名として使用しない場合は、ソース値として一意の値 (ID など) を使用します。次にレベルにプロパティを追加して、そのプロパティ値をメンバ名として使用します。このページで後述の "プロパティ値をメンバ名として使用する方法" を参照してください。

8.2 レベルに対するヌル置換文字列の指定

レベルの **[ヌル置換文字列]** オプションは、NULL 値の代わりに使用する文字列を指定します。この文字列は、キューブで指定したヌル置換文字列より優先されます。

8.3 リスト・ベースのレベルの定義

リスト、配列、または InterSystems IRIS® データ・プラットフォームのクラス・リレーションシップに基づくレベルを定義できます。それぞれの個別リスト項目は、そのレベルの別個のメンバとしてインデックスが作成されます (このようなレベルの詳細は、必ず "リスト・ベースのレベルの使用に関する注意" を参照してください)。

システムでは、データ型 %List のソース値を直接使用できます。これは、$List 関数から返される形式か、文字区切りリストです。その他の形式については、ソース値を変換してから使用する必要があります。

注釈　リスト・ベースのレベルは、独自の階層に含める必要があります。この階層内に他のレベルが存在することはできません。

8.3.1 ソース値が標準形式の場合のレベルの定義

ソース値が、$List 形式、%List 形式、または文字区切りリストで使用できる場合、前述のように、以下の追加手順を使用してレベルを定義します。

1. [ソース値リスト・タイプ] で、以下のいずれかを選択します。
   - [$List 構造] – ソース値が $List 関数で返される形式であるか、データ型 %List である場合に使用します。
   - [コンマ区切り] – ソース値がコンマ区切りリストの場合に使用します。
   - [その他の区切り] – ソース値が別の文字を区切り文字として使用するリストの場合に使用します。
2. [その他の区切り] を選択した場合は、[リスト区切り文字] でリストの区切りに使用する文字を指定します。

8.3.2 ソース値が他の形式の場合のレベルの定義

ソース値が、$List 形式で、または文字区切りリストとして使用できない場合は、以下の方法を使用します。

1. 要求されるいずれかの形式に変換するユーティリティ・メソッドを作成します。以下はその例です。

   Class Member

```
ClassMethod GetAllergies(ID As %Numeric) As %List
{
    Set allergies=##class(BI.Study.Patient).%OpenId(ID,0).Allergies
    If (allergies.Count()=0) {Quit $LISTFROMSTRING("No Data Available")}
    Set list=""
    For i=1:1:allergies.Count() {
        Set $LI(list,i)=allergies.GetAt(i).Allergen.Description
        }
    Quit list
}
```

   以下の点に注意します。

   - %List クラスは、$List 形式と同等です。
   - %OpenId() に対する 2 番目の引数によって、同時処理ロックが指定されます。この引数がゼロの場合、速度の確保のため、オブジェクトの他のユーザをチェックすることなくオブジェクトを開いています。オブジェクトから値を読み取るのみの場合は、複数ユーザの環境であってもこの方法が安全です。
   - アレルギーがない場合、このメソッドは文字列 `No Data Available` を返します。これは、返されるその他の文字列と同様にレベルのメンバになります。

Tipヒント このユーティリティ・メソッドは、キューブ・クラスに配置できます。また、このメソッドはターミナル内で簡単にテストできます。

2. レベルを定義するときは、アーキテクト内で以下のように操作します。
   a. **[式]** で、ユーティリティ・メソッドを起動し、引数として %source.%ID を渡します。以下はその例です。

      ```
      ##class(Cubes.StudyPatients).GetAllergies(%source.%ID)
      ```

   b. **[リスト]** を選択します。
   c. ソース値が文字区切りリストの場合は、**[リスト区切り文字]** でリストの区切りに使用する文字を指定します。

      これが NULL の場合、ソース値は $List 形式であると見なされます。

8.3.3 例

SAMPLES ネームスペースの Patients キューブには、以下のリスト・ベースのレベルの例があります。

- `Allergies` レベルは、ユーティリティ・メソッドを使用してリストを返すソース式に基づいています。
- `Allergy Severities` レベルは、ユーティリティ・メソッドを使用してリストを返すソース式に基づいています。
- `Diagnoses` レベルは、$List 形式のプロパティを使用します。

8.4 時間レベルの定義

時間レベルは、データ内の日付/時間値（生年月日、注文日、回答日など）に従ってデータをグループ化します。ここでは、このタイプのレベルの作成方法について説明します。以下のトピックについて説明します。

- 概要
- 時間レベルを定義する方法
- 時間レベルおよび階層
- 日付オフセットがあるカレンダーの処理
- サンプルにアクセスできる場所

8.4.1 概要

時間レベルは、時刻ディメンジョン内で定義されるレベルです。時間レベルのセットの例については、Patients キューブの `BirthD` ディメンジョンを参照してください。`BirthD` ディメンジョンは、ソース・クラスの BirthDate プロパティに基づきます。このプロパティには、患者の生年月日が含まれています。

このディメンジョンの各レベルは、**[時間関数]** と呼ばれるオプションを使用して、生年月日の特定の部分を抽出します。例えば、`Year` レベルは、生年月日の年の部分のみを抽出します。

結果：

- `Decade` レベルには、`2010s`、`2000s`、`1990s` などのメンバがあります。
- `Year` レベルには、`2009`、`2008` などのメンバがあります。
- `Quarter Year` レベルには、`Q1 2009`、`Q2 2009` などのメンバがあります。

- Period レベルには、2009-01、2009-02 などのメンバがあります。
- Date レベルには、Jan 1 2009、Jan 2 2009 などのメンバがあります。
- 階層 H1 内のレベルの順序によって、Decade レベルが Year レベルの親となるように設定されます。例えば、メンバ 1990s は、メンバ 1990、1991、1992 などの親になります。

  同様に、Year レベルは Quarter Year レベルの親です。例えば、メンバ 2009 は、メンバ Q1 2009、Q2 2009 などの親になります。

時間のレベルでは、NOW と呼ばれる特殊なメンバを使用できます。このメンバは現在の日付 (実行時) を使用して、そのレベルの適切なメンバにアクセスします。

注釈　既定では、グレゴリオ暦が使用されます。代わりに、ヒジュラ暦を使用することもできます。その場合は、メンバ名はここに示したものとは異なります (レコードはヒジュラ暦に適した形でメンバに割り当てられます)。

8.4.1.1 ロケールを使用した時間メンバ名の制御

キューブの構築時、システムは、時間ディメンジョンのメンバの名前を決定するために、現在のサーバ・ロケールを使用できます。以下のようにして、関連するオプションを確認することをお勧めします。

1. 管理ポータルで、**[システム管理]**、**[構成]**、**[各国語の設定]**、**[ロケール定義]** の順に選択します。

   このページには現在のサーバ・ロケールが表示されます。

2. **[ロケールの日付/時刻/数値の形式を使用する]** オプションの設定を確認して、必要に応じて変更します。

   このオプションが **[はい]** に設定されている場合、システムはこのページに表示されるロケールに基づいて、時間ディメンジョンのメンバを構築または同期します。

後でロケールを変更した場合は、キューブをリコンパイルおよび再構築する必要があります。

"ロケール定義" も参照してください。

8.4.2 時間レベルの定義

時間レベルを定義する手順は以下のとおりです。

1. 前述のようにディメンジョンを定義します。ただし、2 つ変更点があります。
   - **[ディメンジョン・タイプ]** で、**[時間]** を選択します。
   - **[暦]** で、**[グレゴリオ]** (既定値)、**[ヒジュラ (表形式)]**、**[ヒジュラ (観測)]**、または **[部分]** を選択します。

     このオプションでは、このディメンジョンのレベルのメンバにソース・レコードを割り当てる際に使用する暦を指定します。**[部分]** オプションの詳細は、以下のサブセクションを参照してください。
   - **[プロパティ]** または **[式]** を指定します。

     [プロパティ] を指定する場合、そのプロパティは %Date、%Time、%TimeStamp のいずれかの型にするか、2 番目のサブセクションに示した要件を満たすカスタムのデータ型にする必要があります。

     [式] を指定した場合、ソース式が $Horolog 形式のデータを返す必要があります。値を NULL にすることはできますが、0 にすることはできません (値を 0 にすると、キューブの構築時に評価エラーが発生します)。

   ディメンジョンを定義すると、ディメンジョンと階層が作成されます。
2. この階層内でレベルを定義します。

   この場合、データ・ディメンジョンとは異なるオプションが表示されます。

3. [関数で値を抽出] で、日付/時間値の必要な部分を抽出する関数を選択します。以下のテーブルのいずれかの値を使用します。ヒジュラ暦の場合は、すべての関数がサポートされるわけではないことに注意してください。

時間関数	格納される値	表示される値	ヒジュラ時間ディメンジョンでの使用可否
MinuteNumber	60958,5083	01:24	はい
HourNumber	60958,5083	1am	はい
DayMonthYear	60958	Nov 24 2007	はい
DayNumber	24	24	はい
WeekYear	2007W47	2007W47	いいえ
WeekNumber	47	47	いいえ
MonthNumber	11	November	はい
MonthYear	200711	November 2007	はい
QuarterNumber	4	Q4	はい
QuarterYear	20074	Q4 2007	はい
Year	2007	2007	はい
Decade	1950	1950s	はい
DayOfWeek	1	Sunday (日曜日)	不可

4. [時刻形式] には、オプションで、日付の表示方法を説明する書式文字列を指定します。詳細は、最初のサブセクションを参照してください。

5. [ヒジュラ (観測)] 暦を使用している場合は、%Calendar.Hijri クラスを使用して太陰観測値を追加します。

   詳細は、%Calendar.Hijri のクラスリファレンスを参照してください。

WeekYear 関数と WeekNumber 関数は、ISO 8601 標準 (1 ～ 53 の週番号を 1 年のすべての日付に割り当てる) に準拠しています。365 または 366 は 7 で割り切れないので、週数が 53 になる年があります。年の第 1 週は、その年の最初の木曜日が含まれる週として定義されています。これは、1 月 1 日が前年最終週に含まれる年があることを意味します。同様に、12 月 31 日が翌年第 1 週に含まれる場合もあります。

8.4.2.1 部分日付

[部分] の [暦] タイプを選択すると、データ内で不確実な日付を使用できます。[部分] の [暦] タイプが設定された時間レベルでは、前のセクションで説明されている、QuarterYear や MonthYearPartial のような特別なバージョンの時刻関数が使用されます。これらの時刻関数は、他の [暦] タイプの時刻関数と同じ名前を共有していますが、`2018-05-02` のような完全な日付が必要な他の [暦] タイプの時刻関数とは異なり、`2017` や `2017-01` のような部分的な日付値を受け付けることができます。その後、部分日付時刻関数は、これらの部分日付を `2017-00-00` や `2017-01-00` のような完全なキーに変換します。欠落している情報がある日付のセクションは、`00` で表されます。例えば、部分日付 `2017-01` の場合、年と月はありますが日がないので、部分日付時刻関数によって `2017-01-00` という形式に変換されます。部分日付時刻関数は、`2017-00-00` や `2017-01-00` のような部分日付形式も受け付けます。

部分日付に基づく暦を時間レベルの基礎として使用する場合、アナライザで検出されたキーでは、`00` の値はすべて `Unknown` というテキストに変換されます。この変換はローカライズされており、フォーマット・ルールに従っています。例えば、キー `2018-07-00` と時刻形式 `ddd mm yyyy` の場合、`Unk 07 2018` というラベルが生成されます。さらに、不明な日付が含まれる各レベルは、これらの不明な日付を格納するための追加メンバを持ちます。例えば、QuarterYear 時刻関数を使用する QuarterYearPartial という名前のレベルは、部分日付が理由で四半期が不明になっている日付を

表すため、Q0 2009 や Q0 2012 のようなメンバを持つ場合があります。QuarterYearPartial レベルには、既知の日付 (Q1 2009 や Q4 2010 など) に対するメンバも含まれる可能性があります。

現在のところ、部分日付の時間レベルでは NOW とオフセットの使用がサポートされています。現在のキーに不明な部分がある場合、NOW オフセットは適用できず、元のキーが返されます。

現在のところ、[部分] の [暦] タイプに基づく時間レベルでは、LAG や LEAD などの MDX クエリ関数の使用はサポートされていません。同様に、キューブ・クラスの <level> 要素の timeOffset プロパティおよび timeFormat プロパティを変更することによる日付オフセットもサポートされていません。

8.4.2.2 時刻形式の指定

時間タイプのディメンジョン内のレベルでは、Time Format でそのレベルのメンバの表示名の形式を指定します。この形式は、クエリ実行時に適用されます。値の格納方法およびインデックスの作成方法には影響しません。

Time Format 属性では、以下の日付部分 (大文字と小文字が区別される) とその他の部分で構成される文字列を指定します。

Piece	変換値
y	年番号
q	四半期番号
mmmm	完全な月名
mmm	月の略称
mm	月の数値表記、必要に応じて先頭の 0 を表記
m	月の数値表記、先頭のゼロなし
dddd	完全な曜日名
ddd	曜日の略称
dd	日付の表記、必要に応じて先頭の 0 を表記
d	日付の表記、先頭の 0 なし
\x	x
ピリオド (.)、スラッシュ (/)、ハイフン (-)、およびスペース文字	変更なし
その他の文字	無視

名前はすべて、現在のサーバ・ロケールに基づきます。"ロケールを使用した時間メンバ名の制御" を参照してください。

重要　時間レベルでは、メンバの表示名は一意である必要があります。また、メンバに設定できる表示名は 1 つに限られます。これらのルールは、前述の日付部分のすべてが、すべての時間レベルに適合するわけではないことを意味します。

以下のテーブルは、各時間レベルに適合する日付部分を示しています。

レベルの種類 ([関数で値を抽出] の設定)	適合する日付部分	既定の timeFormat	その他の例
Year	y	y (2004)	\F\Y y (FY 2004) \F\Y (FY2004)

レベルの種類 ([関数で値を抽出] の設定)	適合する日付部分	既定の timeFormat	その他の例
QuarterYear	y、q	\Qq y (Q3 2004)	\Qq \F\Y y (Q1 FY 2004) \Qq \F\yy (Q1 FY2004)
QuarterNumber	q	\Qq (Q3)	
MonthYear	y、mmmm、mmm、mm、m	mmmm y (February 2004)	mmm y (Feb 2004) y-mm (2004-02) mm/y (02/2004) m/y (2/2004) \F\Yy-mm (FY2004-02)
MonthNumber	mmmm、mmm、mm、m	mmmm (February)	mmm (Feb) mm (02) m (2)
DayMonthYear	y、mmmm、mmm、mm、m、dddd、ddd、dd、d	mmm dd y (Feb 1 2004)	mmmm dd y (February 03 2010) y-mm-dd (2010-02-03) mm/dd/y (02/03/2010) m/d/y (2/3/2010) dddd, mmmm dd y (Wednesday, February 03 2010) ddd, mmm dd y (Wed, Feb 03 2010)
DayNumber	dd、d	d (1)	dd (01)
DayOfWeek	dddd、ddd、dd、d	dddd (Tuesday)	dd (03) d (3)

8.4.2.3 カスタムのデータ型の使用

以下のいずれかのシナリオでは、タイム・レベルのベースに、カスタムのデータ型を使用するプロパティを直接使用できます。

- 最初のシナリオでは、このデータ型はデータを ODBC 日付形式で格納します。データ型クラスの ClientDataType は TIMESTAMP である必要があります。値は、完全な $Horolog 日付に変換されます。
- 2 番目のシナリオでは、このデータ型はデータを $Horolog 時刻 ($Horolog 文字列の 2 番目の部分) として格納します。データ型クラスの ClientDataType は TIME である必要があります。値は、完全な $Horolog 日付に変換されます。(日付の選択は任意であるため、このデータは時間レベルおよび分レベルに対してのみ使用する必要があります。)

%DeepSee.Utils の %ConvertDate() の実装を参照してください。

8.4.3 時間レベルおよび階層

既に述べているように、階層内のレベルの順序はレベルのメンバの作成方法に影響します。同一階層内の隣接する 2 つのレベルでは、最初のレベル (レベル A) が 2 番目のレベル (レベル B) の親になります。階層は親子階層であるため、以下のようになります。

- レベル A のメンバはすべて、レベル B の 1 つ以上のメンバの親です。
- レベル B のメンバはすべて、レベル A の 1 つのメンバの子です。
- レベル B の特定のメンバに属するレコードはすべて、必ずレベル A の同一メンバに属します。

このため、例えば [**年**] に基づくレベルが [**四半期**] に基づくレベルの親になることはできません。2 人の患者について考えてみましょう。1 人は 2007 年第 3 四半期に生まれ、もう 1 人は 1982 年第 3 四半期に生まれています。この 2 人の患者はいずれも Quarter レベルの同じメンバ (Q3) に属していますが、Year レベルでは異なるメンバ (それぞれ 2007 と 1882) に属しています。

また、例えば [**月**] に基づくレベルが [WeekNumber] に基づくレベルの親になることはできません。週は 2 つの月に渡ることがあるため、週メンバに属するレコードが必ずしも同じ月メンバに属しているとは限りません。

以下のテーブルは時間レベルとその一般的な子レベルを示しています。参考のため、各レベルのメンバ例も示しています。

時間関数に基づくレベル	含まれるメンバの例	時間関数に基づく標準的な子レベル	メモ
Decade	1950s	Year、QuarterYear、MonthYear、または DayMonthYear	このレベルは、日付のすべての部分を使用します。
Year	2007	QuarterYear、MonthYear、または DayMonthYear	このレベルは、日付のすべての部分を使用します。
QuarterYear	Q4 2007	MonthYear または DayMonthYear	このレベルは、日付のすべての部分を使用します。
MonthYear	November 2007	DayMonthYear	このレベルは、日付のすべての部分を使用します。
WeekYear	2007W47	WeekNumber	このレベルは、日付のすべての部分を使用します。
DayMonthYear	Nov 24 2007	標準的な子レベルはなし	このレベルは、日付のすべての部分を使用します。
QuarterNumber	Q4	MonthNumber	このレベルは、年に依存しません。
MonthNumber	November	標準的な子レベルはなし	このレベルは、年に依存しません。
WeekNumber	47	標準的な子レベルはなし	このレベルは、年に依存しません。
DayNumber	24	標準的な子レベルはなし	このレベルは、年および年の部分に依存しません。
DayOfWeek	Wednesday (水曜日)	標準的な子レベルはなし	このレベルは、年および年の部分に依存しません。

時間関数に基づくレベル	含まれるメンバの例	時間関数に基づく標準的な子レベル	メモ
HourNumber	1am	MinuteNumber	このレベルは、日に依存しません。
MinuteNumber	01:24	標準的な子レベルはなし	このレベルは、日に依存しません。

8.4.4 日付オフセットがあるカレンダーの処理

場合によっては、日付オフセットを含む財務カレンダーに時間レベルを一致させることが必要になります。例えば、会計年度は多くの企業で 10 月 1 日に開始します。以下のピボット・テーブルで考えてみましょう。

FY 2002	FY 2002-Q1	Oct-2001	14
		Nov-2001	11
		Dec-2001	7
	FY 2002-Q2	Jan-2002	12
		Feb-2002	9
		Mar-2002	10
	FY 2002-Q3	Apr-2002	9
		May-2002	12
		Jun-2002	12
	FY 2002-Q4	Jul-2002	13
		Aug-2002	9
		Sep-2002	15

最も内側のグループ化では、このピボット・テーブルは実際の期間 (年と月の結合) によってレコードをグループ化します。例えば、実際の期間 2001 年 10 月に 14 レコードが関連付けられています。この期間は会計四半期にグループ化され、会計四半期は会計年度にグループ化されます。四半期 FY 2002-Q1 には、2001 年 10 月、11 月、12 月が含まれることに注意してください。

このような時間レベルの場合、スタジオでキューブ・クラスを編集し、`<level>` 要素の `timeOffset` プロパティと `timeFormat` プロパティを指定します。ここで示した例の場合は以下のようになります。

- 最初の 2 つのレベル (FY 2002、FY 2002-Q1 などのメンバのあるレベル) では、`timeOffset` が `"-3m"` になります (これらのレベルで使用される日付値から 3 か月減算します)。
- 3 番目のレベル (Oct-2001 などのメンバのあるレベル) では、`timeOffset` が指定されず、このレベルでは実際の日付値が使用されます。

詳細は、"キューブ・クラスのリファレンス情報" の "日付オフセットの指定" を参照してください。

8.4.5 例

SAMPLES ネームスペースには、以下の時間レベルの例があります。

- Patients キューブで、`BirthD`、`BirthQD`、および `BirthTD` の各ディメンジョンを参照してください。
- HoleFoods Sales キューブで、`DateOfSale` レベルを参照してください。

8.5 カスタム時間レベルの定義

時間を他の方法で使用するカスタム・レベルを定義できます。そのためには、以下の操作を実行します。

- [**タイプ**] が ([**時間**] ではなく) [**データ**] であるディメンジョンを作成します。
- レベルごとに、目的の値を返すように [**式**] に値を指定します。
- このページで後述する ″メンバの並べ替え順序の指定″ を参照してください。

  既定では、メンバはアルファベット順に並べ替えられます。

8.5.1 例

SAMPLES ネームスペースの Patients キューブでは、`BirthWeekdayD` ディメンジョンを参照してください。このディメンジョンの場合、**式**は以下のようになります。

```
$system.SQL.DAYNAME(%source.BirthDate)
```

このメソッドは、%SYSTEM.SQL クラスの DAYNAME() メソッドを実行します。このクラスには、データ値を処理するメソッドが大量に用意されています。

このディメンジョンを使用すると、以下のようにピボット・テーブルを作成できます。

Weekday	Patient Count
Sunday	147
Monday	140
Tuesday	141
Wednesday	141
Thursday	140
Friday	150
Saturday	141

このサンプル・ディメンジョンは、1 つのレベルのみ定義しています。適切なロジックを使用すると、例えば 4-4-5 カレンダーの週、月、および四半期を使用する階層を定義できます。

8.6 年齢レベルの定義

年齢レベルは、データ内のキューブの構築日付を基準にした年齢値に従ってレコードをグループ化します。例えば、(キューブの構築日付を基準に計算した) 年齢で患者をグループ化することができます。

注釈　年齢レベルを正確に維持するにはキューブを再構築する必要があります。InterSystems IRIS Business Intelligence のキューブ同期機能は、それらに対して何も作用しません。そのため、年齢レベルは通常お勧めしません。年齢でレコードをグループ化する必要がある場合、代わりに計算メンバを定義することをお勧めします。″年齢メンバの定義″ を参照してください。

年齢レベルを定義する手順は以下のとおりです。

1. 前述のようにディメンジョンを定義します。ただし、2 つ変更点があります。
   - [**タイプ**] で [**年齢**] を選択します。

- [プロパティ] または [式] を指定します。

重要　[プロパティ] を指定した場合、そのプロパティは $Horolog 形式であるか、データ型 %TimeStamp (またはサブクラス) であることが必要です。

[式] を指定した場合、ソース式が $Horolog 形式のデータを返す必要があります。

ディメンジョンを定義すると、ディメンジョンと階層が作成されます。

2. この階層内でレベルを定義します。

この場合、データ・ディメンジョンとは異なるオプションが表示されます。

3. [関数で値を抽出] で、時間値の必要な部分を抽出する関数を選択します。以下の値のいずれかを使用します。

- `"Days"` – 日単位で年齢を決定します。
- `"Months"` – 月単位で年齢を決定します。
- `"Years"` – 年単位で年齢を決定します。

Patients サンプルは、このタイプのレベルの説明には対応しないことに注意してください。`Age`、`Age Bucket`、および `Age Group` レベルは、架空の研究の患者を、研究の日付での年齢を基準にしてグループ化しています。これは遡及的な研究では一般的なことです。

8.7 範囲式の指定

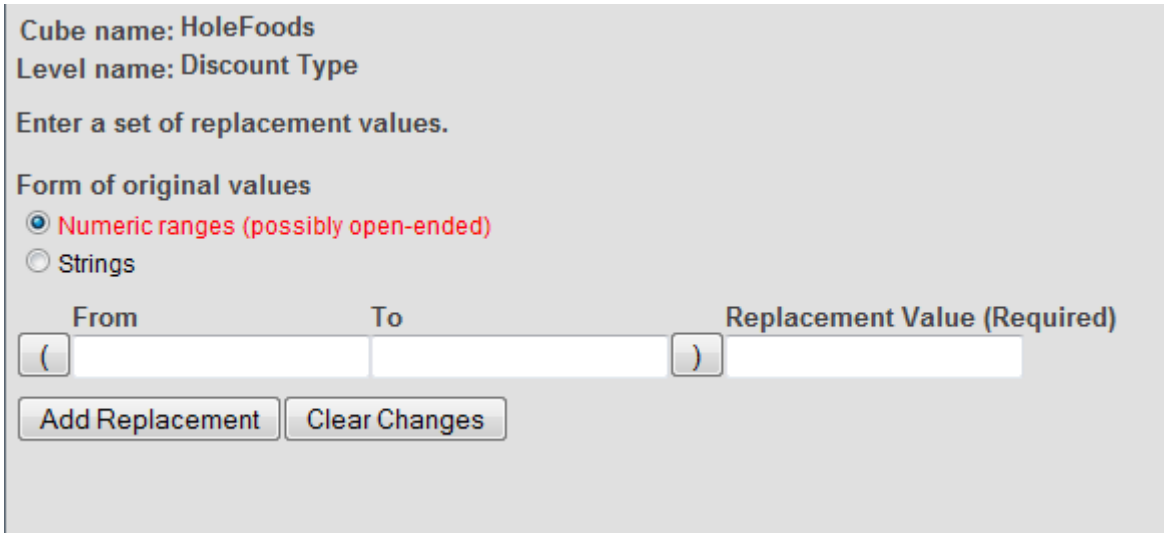
データ・ディメンジョンのどのレベルでも、[範囲式] オプションを使用できます。このオプションを使用すると、実際のソース値の場所に新しい値を使用できます。数値データの場合、この変換によって、ソース値を個別なビンで置換できます。任意のデータに対して、この変換によって、置換文字列を指定できます。

どちらの場合も、新しい値がそのレベルのメンバの名前になります。この値は、メンバのキーにもなります。

範囲式を指定する手順は以下のとおりです。

1. [範囲式] フィールドの横にある検索ボタン をクリックします。

アーキテクトに以下のようなページが表示されます。



2. [**元の値の形式**] では、[**数値範囲**] または [**文字列**] のどちらかを選択します。

   ここで選択した値は、このページの形式に影響します。

3. 以下のように、一連の置換を追加します。

   - テーブルの最後に行を追加するには、[**置換の追加**] をクリックします。

     詳細は、この後のサブセクションの指定に従って入力します。

   - 行を削除するには、その行の [X] ボタンをクリックします。

   行の順序によって、このレベルのメンバの既定の並べ替え順序が決まります。

4. [OK] をクリックして、このページを閉じます。

   [**範囲式**] フィールドが以下のように表示されます (数値ビンの場合)。

   ```
   (,0]:None;(0,0.2):1-19%;[0.2,0.5):20-49%;[0.5,1]:50%+;
   ```

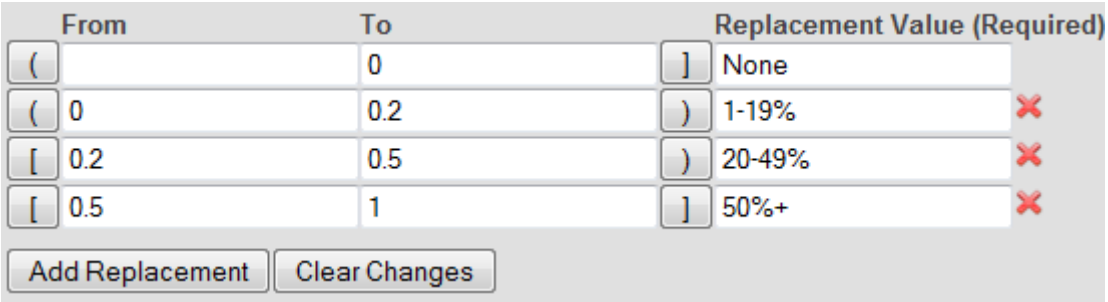
   または (文字列データの場合)

   ```
   NONE:None;MINR:Minor;SERS:Serious;FATL:Fatal;UNK:Unknown;
   ```

メンバ名をローカライズできるようにするには、"レベルのメンバの手動指定" を参照してください。

8.7.1 数値ビンの定義

数値ビンを定義する場合、範囲式エディタは以下のように表示されます。



このテーブルの各行が、以下のように数値ビンを定義します。

- 行の左端のボタンは、下限の値を含む [か、含まない (かを示します。

  下限が存在しない場合は、このオプションは何の効果も持ちません。

  下限の値を含むとは、[**下限**] に示された値が範囲に含まれることを意味します。下限の値を含まないとは、[**下限**] の値が範囲に含まれないことを意味します。

- [**下限**] は、指定されている場合は、ビンの下限を示します。

- [**上限**] は、指定されている場合は、ビンの上限を示します。

- 行の右端のボタンは、上限の値を含む] か、含まない) かを示します。

  上限が存在しない場合は、このオプションは何の効果も持ちません。

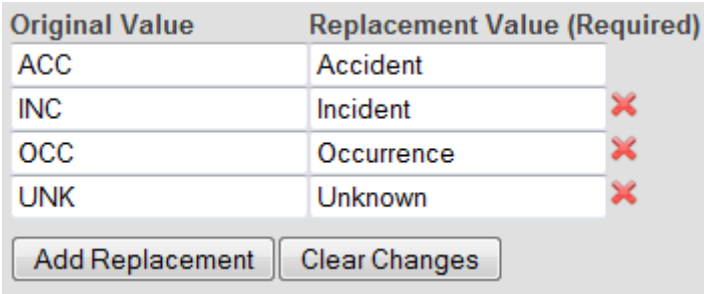
- [**置換値**] は、このレベルのメンバ名として使用する文字列を指定します。ソース値が定義されている範囲内にあるレコードはすべて、このメンバに割り当てられます。

この値は、メンバのキーにもなります。

このレベルでは、ソース値がどの指定ビンにも該当しないレコードは無視されます。

8.7.2 文字列置換の定義

文字列置換を定義する場合、範囲式エディタは以下のように表示されます。



このテーブルの各行が、以下のように文字列置換を定義します。

- **[元の値]** は、このレベルのソース・プロパティまたはソース式の可能な値を指定します。
- **[置換値]** は、このレベルのメンバ名として使用する文字列を指定します。ソース値が **[元の値]** に指定されている値と一致するレコードはすべて、このメンバに割り当てられます。

  この値は、メンバのキーにもなります。

このレベルでは、ソース値がどの元の指定値にも一致しないレコードは無視されます。

8.7.3 例

SAMPLES ネームスペースには、以下の範囲式の例があります。

- Patients キューブで、`Age Group` レベルを参照してください。
- HoleFoods Sales キューブで、`Discount Type` レベルを参照してください。

8.8 表示値を使用するためのレベルの構成

InterSystems IRIS クラス・プロパティは、プロパティ・パラメータの VALUELIST と DISPLAYLIST によって、格納される値と表示される値の両方を保持できます。以下はその例です。

Class Member

```
Property Gender As %String(VALUELIST = ",F,M", DISPLAYLIST = ",Female,Male");
```

このようなプロパティをレベルとして使用すると、既定では、VALUELIST パラメータで指定された値が使用されます。

レベル定義で **[DISPLAYLIST の値を使用]** オプションを選択すると、DISPLAYLIST パラメータで指定された値が使用されます。以下はその例です。

Gender	Patient Count
Female	5,042
Male	4,958

8.8.1 例

SAMPLES ネームスペースには、以下のような [DISPLAYLIST の値を使用] オプションを使用するレベルの例があります。

- Patients キューブの Gender レベル。
- HoleFoods Sales キューブの Channel Name レベル。

8.9 プロパティ値をメンバ名として使用する方法

既定では、レベルのソース値がそのレベルのメンバの名前になります。

代わりに、このレベルにプロパティを定義し、そのプロパティの値をメンバ名に使用することで、メンバ名を指定することができます。これは、以下のシナリオで役立ちます。

- 実行時にメンバ名へのアクセスを可能にする – 実行時にプロパティの値へのアクセスを可能にします (レベルの場合は異なります)。
- メンバにわかりやすい名前を指定する – 場合によっては、わかりにくい一意の値をレベルの基にする必要があります。例えば、患者の一次診療医に基づくレベルがあるとします。人名は必ずしも一意であるとは限りません。このため、代わりに一意の医師 ID をレベルの基にする必要があるとします。この ID はユーザには無意味です。このような状況で、医師の名前にアクセスするプロパティも定義して、そのプロパティをメンバの名前として使用します。

レベルのメンバの名前にプロパティ値を使用する手順は以下のとおりです。

1. このレベルにプロパティを定義します。

   レベルの各メンバにはこのプロパティの値があります。

2. このプロパティに対して [**メンバ名として使用**] オプションを選択します。
3. 必要に応じて、実行時にプロパティ値を取得するには、[**実行時に値を取得**] オプションを選択します。

   このオプションを使用する場合、以下の要件に注意してください。親レベル (取得するプロパティが含まれるレベル) では、そのレベルのソース・プロパティまたはソース式は ID として評価される必要があります。システムはそのことを想定して (少なくともこのレベルで)、ソース・データを正規化します。すなわち、そのレベルでは、データは別のテーブルに存在し、ソース・テーブルにはそのテーブルへのリンクが格納されます。

8.9.1 例

SAMPLES ネームスペースには、以下のような、[**メンバ名として使用**] または [**実行時に値を取得**]、あるいはその両方で構成されたプロパティを持つレベルの例があります。

- Patients キューブの DxDoc レベル。このレベルには、Name プロパティがあります。このプロパティは、[**メンバ名として使用**] で構成されています。DxDoc レベルは、医師の一意の ID に基づいています。生成した患者の数によっては、同名の複数の医師が表示される場合があります。このレベルの定義方法により、これらの医師どうしは結合されません。

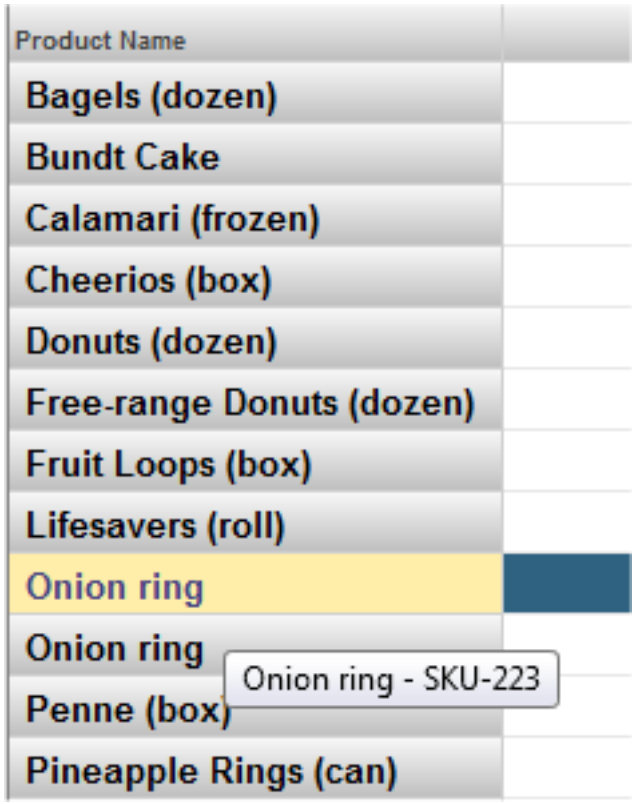
- Patients キューブの `City` レベル。このレベルには、`Name` プロパティがあります。このプロパティは、**[メンバ名として使用]** と **[実行時に値を取得]** で構成されています。
- HoleFoods Sales キューブの `City` レベルと `Channel Name` レベル。これらのレベルには、**[メンバ名として使用]** で構成されるプロパティがあります。
- HoleFoods Sales キューブの `Product Name` レベル。このレベルには、**[メンバ名として使用]** と **[実行時に値を取得]** で構成されるプロパティがあります。

8.10 メンバのツールのヒントの指定

アナライザでメンバ名の近くにカーソルを置くと、ツールのヒントが表示されます。既定では、ツールのヒントにはメンバ名が表示されます。任意のレベルでツールのヒントに表示する情報を追加できます。そのためには、以下の操作を実行します。

1. 目的のレベルにプロパティを追加します。ソース値には、目的のツールのヒントを含む値を使用します。
2. スタジオでキューブ定義を編集し、新しく追加したプロパティを変更します。このプロパティでは、`isDescription` 属性を `"true"` に指定します。"キューブ・クラスのリファレンス情報" の "<property>" を参照してください。

   これでツールのヒントには、メンバ名、その後にハイフン、その後に指定したプロパティの値が表示されるようになります。例えば以下のように出力されます。



これは、スペースの関係上メンバ名には短い名前を使用する必要があり、同時に各メンバについて十分な情報を提供する必要がある場合に便利です。

8.11 メンバの並べ替え順序の指定

ここでは、レベルのメンバの並べ替え順序を制御する方法を説明します。

8.11.1 データ・レベルの場合

データ・レベルの場合、既定では、メンバは以下のように並べられます。

- レベルで範囲式を使用しない場合、メンバは名前の昇順に並べ替えられます（名前の形式にかかわらず、アルファベット順の並べ替えを使用）。
- そのレベルで範囲式を使用する場合、メンバは範囲式で決定される順序で並べ替えられます。すなわち、最初のメンバは範囲式の最初の置換によって決定され、2 番目のメンバは 2 番目の置換によって決定され、以下同様に決定されます。

データ・レベルのメンバの並べ替え順序を変更するには、後続のサブセクションで説明する方法のいずれかを使用します。

8.11.1.1 レベルのソート・オプションの指定

あるレベルのメンバに対するソート順序を指定する最も簡単な方法は、そのレベルの **[ソート]** オプションを設定することです。**[昇順]**、**[昇順数値]**、**[降順]**、または **[降順数値]** を選択します。

[昇順] と **[降順]** は、メンバをアルファベット順（昇順または降順）に並べ替えます。**[昇順数値]** と **[降順数値]** は、数値順（昇順または降順）に並べ替えます。

8.11.1.2 プロパティ値を基準にしたメンバの並べ替え

プロパティ値を基準にレベルのメンバを並べ替える手順は以下のとおりです。

1. このレベルにプロパティを定義します。

   レベルの各メンバにはこのプロパティの値があります。

2. このプロパティには、**[プロパティ値を基準にしたメンバの並べ替え]** で **[昇順]**、**[昇順数値]**、**[降順]**、または **[降順数値]** のいずれかを選択します。

   [昇順] と **[降順]** は、メンバをアルファベット順（昇順または降順）に並べ替えます。**[昇順数値]** と **[降順数値]** は、数値順（昇順または降順）に並べ替えます。

   （既定では、いずれの値も選択されていません。また、プロパティはメンバの並べ替え順序に影響しません）。

レベルの複数のプロパティ内で **[プロパティ値を基準にしたメンバの並べ替え]** を指定すると、これらのプロパティのうち最初のプロパティでメンバが並べ替えられます。その後、これらのプロパティの 2 番目を基準に並べ替えられ、以下同様に実行されます。

例は、SAMPLES ネームスペースの Patients キューブを参照してください。このキューブで、`Allergy Severities` レベルには、**[プロパティ値を基準にしたメンバの並べ替え]** で構成されるプロパティがあります。このプロパティは、数値を返すソース式に基づいています。

8.11.1.3 プロパティ名をメンバ名として使用する方法

前述と同様のもう 1 つの方法として、以下の方法も使用できます。

1. メンバ名が望ましくなくても、必要な順序でメンバを並べ替える、レベルのソース式を使用します。
2. このレベルにプロパティを定義します。

3. このプロパティで、それぞれにより適した名前を返すソース値またはソース式を指定します。このプロパティに対して [**メンバ名として使用**] オプションを選択します。

8.11.1.4 必要な順序でのメンバのリスト

メンバを必要な順序でリストするには、キューブ・クラスを編集して、レベルに <member> 要素を追加します。詳細は、"レベルのメンバの手動指定" を参照してください。

8.11.2 時間レベルの場合

既定では、時間レベルのメンバは、時間によって昇順に並べ替えられています。

メンバに対して別の並べ替え順序を指定するには、[**ソート**] オプションを指定します。[**昇順**] (時間の昇順) または [**降順**] (時間の降順) のいずれかを選択します。

8.12 メンバ名をローカライズ可能にする

ディメンジョン、階層、レベルなどのキューブ要素と同様、キューブにも [**表示名**] の値を指定できます。この値には、後から別の言語での翻訳を定義できます。レベルのすべてのメンバが既知であり、スタジオでキューブ・クラスを編集する場合は、レベルのメンバにも同じ方法を使用できます。"レベルのメンバの手動指定" を参照してください。

時間ディメンジョンでは、現在のロケールによって時間ディメンジョンのメンバの名前が決まります。("ロケールを使用した時間メンバ名の制御" を参照してください。)ロケールを変更した場合は、キューブをリコンパイルおよび再構築する必要があります。

8.13 異なる階層に含まれるレベル間での依存関係の定義

場合によっては、相互に異なる階層に含まれるレベル間に、仮想依存関係が存在します。例えば、Country レベルと Product レベルを持つキューブがあるとします。これらのレベルは、論理的に相互に独立しています。理論上は、任意の製品を任意の国で販売できます。これらのレベルを同じディメンジョンの 1 つの階層に含めることは意味がありません。

しかし、特定の製品の販売先が特定の国々に限定される場合は、これらのレベル間に仮想依存関係が存在します。ユーザが国を選択したとき、望ましいのは、その国で販売可能な製品のみが表示されることです。そのような場合、以下のように、レベル間に依存関係を追加できます。

1. アーキテクトでキューブ・クラスを開きます。
2. 依存レベルの定義を検索します。
3. [**依存**] オプションの値を編集します。以下のような値を使用します。

   ```
   [dim].[hier].[level]
   ```

   dim は他のディメンジョンの名前、hier は階層の名前、level はレベルの名前です。`[dim].[hier].[level]` は、そのレベルの MDX 識別子です。

   以下はその例です。

   ```
   [Region].[H1].[Country]
   ```

レベルが複数のレベルに依存する場合、コンマ区切りリストを指定します。以下はその例です。

```
[dim1].[hier1].[level1],[dim2].[hier2].[level2]
```

レベルの代わりにリレーションシップを指定することもできます。

このオプション属性はファクト・テーブルにインデックスを 1 つ追加するので、結果的に使用されるディスク容量が増えます。この属性は、控えめに使用してください。また、このオプションは、DependsOn コンパイラ・キーワードとはまったく関係ありません。

8.13.1 例

この機能の動作を確認するために、以下のデモを試してみます。

1. アーキテクトで Patients キューブを開きます。
2. `PatGrpD` ディメンジョンの `Patient Group` レベルを編集します。[**依存**] を以下のように指定します。

   ```
   [HomeD].[H1].[ZIP]
   ```

3. キューブを保存して、コンパイルします。
4. ターミナルで、**BI.Populate** の ReassignPatients() メソッドを実行します。

   ```
   d ##class(BI.Populate).ReassignPatients()
   ```

   このメソッドは、Patient Group レベルと ZIP code レベルの間に仮想依存関係が存在するように、このサンプルのデータを変更します。特定の郵便番号の患者は、患者グループ A に属するか、どのグループにも属さないかのどちらかであり、別の郵便番号の患者は、患者グループ B に属するか、どのグループにも属さないかのどちらかです。

   このメソッドはキューブを同期するので、その実行後にこのキューブを再構築する必要はありません。

Basic Dashboard Demo を開き、Home ZIP Code フィルタと Patient Group フィルタを試します。郵便番号を選択すると、Patient Group フィルタのリストに影響します。

8.14 ファクト・テーブル内のフィールド名の指定

キューブ・クラスのコンパイル時に、システムは 1 つのファクト・テーブル・クラスと、いくつかの関連クラスを生成します。キューブの構築時に、これらのテーブルに値が入力されます。詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。

既定では、システムによってファクト・テーブルのフィールド名が生成されますが、その代わりに使用するフィールド名を指定することもできます。これを実行するには、各アプリケーション・レベルの [**ファクト・テーブルのフィールド名**] オプションに値を指定します。一意の名前を使用するように注意してください。このオプションは、時間レベルおよび NLP レベルには使用できません。

重要　[**ファクト・テーブルのフィールド名**] には、SQL の予約語を使用しないでください。SQL 予約語のリストは、"予約語" を参照してください。名前は、英字かパーセント記号 (%) で始まる必要があります。最初の文字が % である場合、2 番目の文字は Z または z である必要があります。制限事項の詳細は、"クラス・メンバ" を参照してください。また、`fact` または `listing` という語は、どのような大文字と小文字の組み合わせであっても使用しないでください。

9
プロパティの定義

ここでは、Business Intelligence キューブでプロパティを定義する方法を説明します。

注釈　　時間ディメンジョンおよび年齢ディメンジョンにプロパティを含めることはできません。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

9.1 プロパティの追加

既存のレベルにプロパティを追加する手順は以下のとおりです。

1. モデル・ビューワで、レベルまたはそのレベル内のプロパティをクリックします。

   この操作によって、プロパティを追加する場所を示します。

   - レベルをクリックすると、そのレベルの他のすべてのプロパティの後に新しいプロパティが追加されます。
   - プロパティをクリックすると、そのプロパティの直前に新しいプロパティが追加されます。

   注釈　　プロパティがメンバの並べ替えに使用される場合は、レベル内のプロパティの順序が重要になります。"メンバの並べ替え順序の指定 (データ・レベルの場合)" を参照してください。

2. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

3. **[新規項目名の入力]** に、プロパティ名を入力します。

   他の種類のキューブ要素とは異なり、予約済みのプロパティ名があります。詳細は、"モデル要素の名前" を参照してください。

4. **[プロパティ]** をクリックします。

5. [OK] をクリックします。

6. モデル・ビューワで、プロパティを選択します。

7. 詳細領域で、**[プロパティ]** または **[式]** の値を指定します。"ディメンジョンまたはレベルのソース値の定義" および "ソース式の詳細" を参照してください。

または、クラス・ビューワからクラス・プロパティをドラッグして、これをモデル・ビューワのレベルにドロップします。次に、必要に応じて詳細領域で変更を加えます。

"その他の共通オプション" も参照してください。

9.1.1 リスト・ベースのレベルのプロパティの定義

関連付けられているレベルがリスト・ベースである場合、以下のようにプロパティを定義する必要があります。

- [式] の値を指定します。
- 式で、%value 変数を参照します。この変数には、関連付けられているリスト要素の値が含まれます。

例えば、Patients キューブの Allergy Severities レベルを考えてみましょう。このレベルには、このレベルのメンバの並べ替え順序を制御するプロパティがあります。このプロパティ (SeveritySort) は、以下の式で定義されます。

```
%cube.GetSeveritySort(%value)
```

これにより、キューブ・クラスの GetSeveritySort() メソッドをアレルギー重症度 (文字列) を引数として実行します。メソッドは、整数を返します。

9.1.2 書式文字列の指定

スタジオでは、書式文字列を指定できます。これにより、プロパティの表示形式を指定できます。この形式設定は、アナライザで (または、手動で MDX クエリを作成することで) オーバーライドできます。詳細は、"<property>" を参照してください。

9.2 実行時のプロパティの値の取得

場合によっては、プロパティの値が適切なソース・テーブルから実行時に取得されるように、プロパティを定義できます。要件は以下のとおりです。

- このプロパティに対して [実行時に値を取得] オプションを選択します。
- 親レベル (取得するプロパティが含まれるレベル) では、そのレベルのソース・プロパティまたはソース式は ID として評価される必要があります。

  システムはそのことを想定して (少なくともこのレベルで)、ソース・データを正規化します。すなわち、そのレベルでは、データは別のテーブルに存在し、ソース・テーブルにはそのテーブルへのリンクが格納されます。

  この要件は、既定では、このレベルのメンバ名が ID であり、通常はユーザによる使用には適していないことを意味します。この場合は、メンバの表示名として使用するため、このレベルのプロパティも構成します。これは、同じプロパティでも別のプロパティでもかまいません。

  "プロパティ名をメンバ名として使用する方法" を参照してください。

9.3 レベル内のプロパティの順序の変更

レベル内のプロパティの順序を変更する手順は以下のとおりです。

- レベル内でプロパティを上に移動するには、そのレベルの行で上矢印をクリックします。
- レベル内でプロパティを下に移動するには、そのレベルの行で下矢印をクリックします。

レベル内のプロパティの順序はレベルのメンバの並べ替え順序に影響する場合があります。"メンバの並べ替え順序の指定 (データ・レベルの場合)" を参照してください。

9.4 ディメンジョン・テーブル内の列名の指定

キューブ・クラスのコンパイル時に、システムは 1 つのファクト・テーブル・クラスと、ディメンジョンに関連するクラスを生成します。キューブの構築時に、これらのテーブルに値が入力されます。詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。レベルのプロパティは、それに対応するディメンジョン・テーブルに格納されます。

既定では、システムによってファクト・テーブルの列名が生成されますが、その代わりに使用する列名を指定することもできます。これを実行するには、各プロパティの **[レベル・テーブルのフィールド名]** オプションに値を指定します。一意の名前を使用するように注意してください。

重要　**[ファクト・テーブルのフィールド名]** には、SQL の予約語を使用しないでください。SQL 予約語のリストは、"予約語" を参照してください。名前は、英字かパーセント記号 (%) で始まる必要があります。最初の文字が % である場合、2 番目の文字は Z または z である必要があります。制限事項の詳細は、"クラス・メンバ" を参照してください。また、`fact` または `listing` という語は、どのような大文字と小文字の組み合わせであっても使用しないでください。

9.5 プロパティの特殊な用途

既定では、プロパティは属するレベルのメンバに何も影響しません。ただし、プロパティを使用して、以下の方法でメンバを変更することができます (方法は組み合わせることもできます)。

- プロパティ値をメンバ名として使用し、レベル定義で定義された既定のメンバ名をオーバーライドできます。

  そのためには、**[メンバの名前として使用]** オプションおよび、(必要に応じて) **[実行時に値を取得]** オプションを使用します。"プロパティ値をメンバ名として使用する方法" を参照してください。

  [メンバ名として使用] を選択し、ソース値が NULL の場合は、適切なヌル置換文字列が使用されることに注意してください。これはレベルまたはキューブのヌル置換文字列 (それぞれ定義がある場合) です。"レベルに対するヌル置換文字列の指定" および "キューブ・オプションの指定" を参照してください。

- プロパティ値を使用して、レベル内のメンバの既定の並べ替え順序を指定できます。

  そのためには、**[プロパティ値でメンバを並べ替え]** オプションを使用します。"メンバの並べ替え順序の指定 (データ・レベルの場合)" を参照してください。

  範囲式を使用するレベルでは、**[プロパティ値を基準にしたメンバの並べ替え]** オプションは何の効果もありません。

- プロパティ値はメンバのツールのヒントとして使用できます。これは、スペースの関係上メンバ名には短い名前を使用する必要があり、同時に各メンバについて十分な情報を提供する必要がある場合に便利です。このようにプロパティを使用するには、`isDescription` 属性を `"true"` に指定します。"キューブ・クラスのリファレンス情報" の "<property>" を参照してください。

Tipヒント ″メンバ・キーおよび名前の適切な定義″ に記載されているように、レベル内のメンバ名は一意である必要はありません。ただし、適切に定義されたキューブでは、指定されたレベルの各メンバには一意のキーがあります。ユーザがアナライザでクエリを作成すると、システムは自動的に名前でなくメンバ・キーを使用します。ユーザは、メンバのリストを展開して、同じ名前を持つ異なるメンバを個別にドラッグ・アンド・ドロップすることができます。便宜を考慮して、値がキーのレベル・プロパティを追加することもお勧めします。このようなプロパティでは、レベルで使用されるものと同じソース・プロパティまたはソース式をそのプロパティの基にするだけです。これで、このプロパティを表示してシステムがメンバに使用する一意の ID を決定できます。

10
メジャーの定義

ここでは、Business Intelligence キューブでメジャーを定義する方法を説明します。

システムは、既定の名前が Count のメジャーを自動的に作成します。この既定の名前をオーバーライドするには、キューブの **[カウント・メジャーのキャプション]** オプションを指定します。"キューブ・オプションの指定" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

10.1 メジャーの追加

メジャーを追加するには、クラス・ビューワからクラス・プロパティをドラッグして、これをモデル・ビューワの **[メジャー]** ラベルにドロップします。次に、必要に応じて詳細領域で変更を加えます。

また、次のようにすることもできます。

1. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[新規項目名の入力]** に、メジャー名を入力します。

   "モデル要素の名前" を参照してください。

3. **[メジャー]** をクリックします。

4. [OK] をクリックします。

5. モデル・ビューワで、メジャーを選択します。

6. 少なくとも、以下のオプションは指定してください。

   - **[タイプ]** – "メジャーのデータ型の指定" の説明に従って、メジャーのデータ型を指定します。
   - **[プロパティ]** または **[式]** – ソース値を指定します。"ディメンジョンまたはレベルのソース値の定義" および "ソース式の詳細" を参照してください。
   - **[集計]** – "メジャー集約方法の指定" の説明に従って、メジャー値の集約方法を指定します。

10.2 メジャーのデータ型の指定

[タイプ] オプションは、以下のように、メジャーがソース・データで予期するデータの種類および生成されるファクト・テーブルで使用されるタイプを指定します。

メジャーのデータ型	ソース・データで予期されるデータ型	メジャーで使用されるデータ型	メジャーの詳細
number (既定)	数値データ	%Double	数値。既定の集約は SUM です。
integer	数値データ (小数値は切捨て)	%Integer	整数値。既定の集約は SUM です。
age	$Horolog 形式の日付/時間データ	%Integer	日数単位の経過時間。共に使用できるのは、AVG (既定)、MIN、および MAX の各集約のみです。夜間にキューブの完全再構築または年齢メジャーの選択的構築を実行する場合を除き、年齢メジャーは一般に推奨されません。
date	$Horolog 形式の日付/時間データ	%DeepSee.Datatype.dateTime	日付値 ($Horolog 形式、ただし秒は省略)。共に使用できるのは、AVG、MIN、および MAX (既定) の各集約のみです。
boolean	0 または 1	%Boolean	集約可能なブーリアン値。既定の集約は COUNT です。
string	任意	%String	ファクト・テーブルに格納される文字列値。インデックスは作成されません。共に使用できるのは COUNT のみです。このメジャーをアナライザにドラッグしてドロップすることはできません。
iKnow*	テキスト値	選択されているソースに応じて %GlobalCharacterStream または %String	NLP スマート・インデックス作成 API を使用して処理してインデックス作成されるテキスト値。このメジャーをアナライザにドラッグしてドロップすることはできません。

*iKnow タイプの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。

10.3 メジャー集約方法の指定

[集計] オプションは、複数のレコードを結合する際、常に、このメジャーの値の集約方法を指定します。このオプションを指定する場合は、以下の値のいずれかを使用します。

- SUM (既定) ― セット内の値を加算します。
- COUNT ― ソース・データが 非 NULL (および非ゼロ) 値のレコード数をカウントします。

- MAX ― セット内の最大値を使用します。
- MIN ― セット内の最小値を使用します。
- AVG ― セットの平均値を計算します。

ブーリアンまたは文字列のメジャーでは COUNT を選択します。

10.4 検索可能なメジャーの指定

メジャーを検索可能 として指定できます。その場合はピボット・テーブルで使用されるレコードをそのメジャーの値でフィルタ処理できます。

メジャーを検索可能として指定するには、詳細領域で [**検索可能**] チェック・ボックスにチェックを付けます。

注釈　　検索可能なメジャーの名前には、角括弧やコンマ ([],) を含めることはできません。

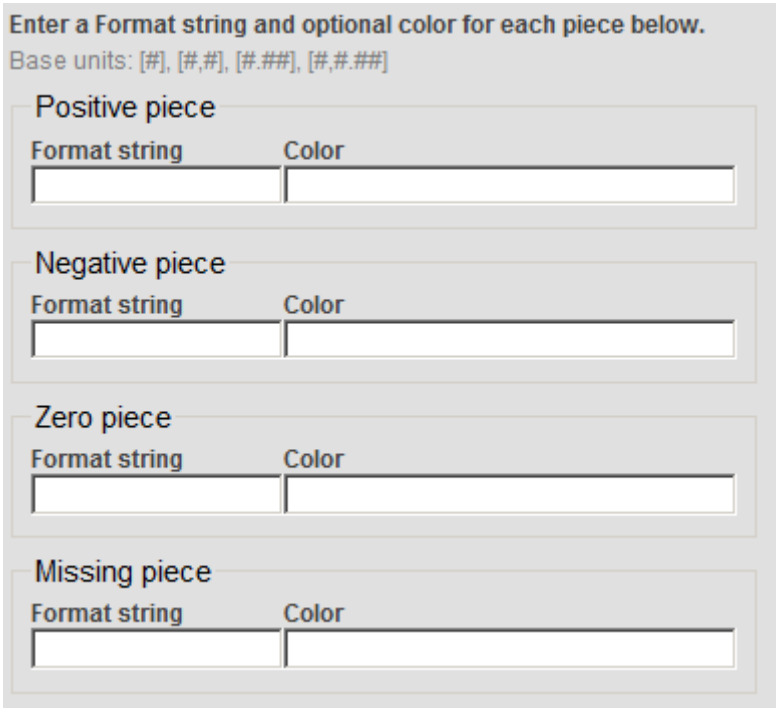
10.5 書式文字列の指定

注釈　　日付メジャーの詳細は、次のセクションを参照してください。

[**書式文字列**] オプションを使用すると、データの表示形式を指定できます。この形式設定は、アナライザで (または、手動で MDX クエリを作成することで) オーバーライドできます。アーキテクトでメジャーの形式設定を指定するには、そのメジャーが表示されているときに以下を実行します。

1. 検索ボタン をクリックします。

   以下のフィールドが含まれるダイアログ・ボックスが表示されます。



以下はその説明です。

- [正の部分] は、正の値に使用する形式を指定します。
- [負の部分] は、負の値に使用する形式を指定します。
- [ゼロの部分] は、ゼロに使用する形式を指定します。
- [欠落部分] は、欠落している値に使用する形式を指定します。これは現在使用されていません。

これらのそれぞれにおいて、[書式文字列] によって数値形式が、[色] によって色が指定されます。

詳細は、日付タイプのメジャーでは異なります。次のセクションを参照してください。

2. 必要に応じて値を指定します (この手順の後の詳細を参照してください)。
3. [OK] をクリックします。

10.5.1 [書式文字列] フィールド

[書式文字列] フィールドは、以下のベース・ユニットのいずれかを含む文字列です。

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記号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記号は必要なだけ指定できます。	12,345.67
%time%	値を hh:mm:ss の形式で表示します。この値は秒数を示しているものと見なされます。これは、継続時間を表示するメジャーで役立ちます。	0:05:32

InterSystems IRIS は、サーバ・ロケールによって決まる 1000 単位の区切り文字と小数点記号を表示します (″ロケールを使用した時間メンバ名の制御″ を参照してください)。ただし、ロケールは、前述のテーブルの最初の列に示した構文には影響しません。

ベース・ユニットの前後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。

以下のテーブルに例を示します。

formatString の例	論理値	表示値
formatString="#,#;(#,#);" これは、既定の数値表示方法に対応します。	6608.9431	6,609
	-1,234	(1,234)
formatString="#,#.###;"	6608.9431	6,608.943
formatString="#%;"	6	600%
formatString="$#,#;($#,#);"	2195765	$2,195,765
	-3407228	($3,407,228)

10.5.2 色の部分

Color フィールドでは、以下のいずれかを指定します。

- MediumBlue、SeaGreen などの CSS カラー名。これらについては、https://www.w3.org/TR/css3-color/ およびインターネットの他の場所を参照してください。
- #FF0000 (赤) などの 16 進カラー・コード。
- rgb(255,0,0) (赤) などの RGB 値。

10.6 日付メジャーの書式文字列の指定

日付メジャーの書式文字列を指定するには、アーキテクトの [詳細] ペインで **[書式文字列]** フィールドに以下のいずれかを入力します。

書式文字列	メジャーの文字列に与える影響	形式の例
%date%	日付は、現行プロセスの既定の日付形式を使用します。	
%date%^color ここで、color は、前のセクションの "色の部分" で説明された色です。	日付は、現行プロセスの既定の日付形式を使用し、指定された色で表示されます。	
^color	日付は指定された色で表示されます。	

10.7 キューブ内のメジャーの順序の変更

キューブ内のメジャーの順序を変更する手順は以下のとおりです。

1. **[順序変更]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[メジャー]** をクリックします。
3. 必要に応じて、**[アルファベット順]** をクリックし、アルファベット順に配列します。

これは、即座にリストに作用します。必要に応じて、リストをさらに再編成できます。また、メジャーを追加する際に、アルファベット順の自動配列は行われません。

4. メジャーの名前をクリックし、必要に応じて上矢印または下矢印をクリックします。
5. 必要に応じて、他のメジャーでもこの操作を繰り返します。
6. [OK] をクリックします。

キューブ内のメジャーの順序は、アナライザでの表示方法に影響を与えます。その他の影響はありません。メジャーをアルファベット順に配列したほうが便利なユーザもいれば、使用頻度の高いメジャーをリストの上部に配置するユーザもいます。

10.8 ファクト・テーブル内のフィールド名の指定

キューブ・クラスのコンパイル時に、システムは 1 つのファクト・テーブル・クラスと、いくつかの関連クラスを生成します。キューブの構築時に、これらのテーブルに値が入力されます。詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。

既定では、システムによってファクト・テーブルの列名が生成されますが、その代わりに使用する列名を指定することもできます。これを実行するには、各メジャーの [ファクト・テーブルのフィールド名] オプションに値を指定します。このオプションは、NLP メジャーには使用できません。一意の名前を使用するように注意してください。

重要　[ファクト・テーブルのフィールド名] には、SQL の予約語を使用しないでください。SQL 予約語のリストは、"予約語" を参照してください。名前は、英字かパーセント記号 (%) で始まる必要があります。最初の文字が % である場合、2 番目の文字は Z または z である必要があります。制限事項の詳細は、"クラス・メンバ" を参照してください。また、fact または listing という語は、どのような大文字と小文字の組み合わせであっても使用しないでください。

10.9 リストの追加フィルタの指定

既定では、ユーザが詳細リストを表示する際、システムは、現在のコンテキスト (リストが要求されたコンテキスト) で使用されているソース・レコードごとに行を 1 つ表示します。指定されたメジャーについて、詳細リストを表示する際にシステムが使用する追加フィルタを指定できます。例えば、Patients サンプルの Avg Test Score メジャーを考えてみます。このメジャーは TestScore プロパティに基づいています。これは一部の患者については NULL になっています。ユーザが Avg Test Score メジャーを開始してリストを表示する際、このメジャーを再定義してそれらの患者をフィルタで除外できます。

このようなフィルタが必要な場合、メジャーの定義の一部に含めます。大半の場合、フィルタは以下の形式になります。

```
measure_value operator comparison_value
```

このフィルタは詳細リスト・クエリに追加され、フィルタ条件を満たさないレコードをすべて除外します。

リスト・フィルタの他の形式に MAX/MIN があります。このようなリスト・フィルタを使用した場合、詳細リストには、メジャーの最大 (または最小) 値を持つレコードのみが表示されます。メジャーは、リスト・フィルタと同じ種類の集約を使用する必要があります (リスト・フィルタが含まれている場合)。

特定のメジャーについて、リストの追加フィルタを指定するには、以下の手順を実行します。

1. モデル・ビューワで、メジャーを選択します。

リスト・フィルタで [**最大**] を使用する場合、[**最大**] として [**集約**] で定義されているメジャーを選択します。

同様に、[**最小**] を使用する場合、[**最小**] として [**集約**] で定義されているメジャーを選択します。

2. ([詳細] 領域の) [**メジャー固有のリスト・フィルタ**] セクションで、以下の値を指定します。
   - [**演算子**] – [<]、[<=]、[>]、[>=]、[<>]、[=]、[**最大**]、[**最小**] のいずれかを選択します。
   - [**値**] – 比較値を指定します。[**演算子**] が [**最大**] または [**最小**] の場合はこのオプションを省略します。

このオプションを使用した場合、このメジャーは検索可能となる必要があるため、アーキテクトで [**検索可能**] チェック・ボックスが自動的に有効になります。

10.9.1 例

Patients キューブの Avg Test Score メジャーを修正し、[**演算子**] に [<>]、[**値**] に "" を指定するとします。つまり、NULL のテスト・スコアを持つ患者をフィルタで除外します。次に、以下のピボット・テーブルを考えてみます。

Diagnoses	Patient Count	Avg Age	Avg Test Score
None	833	32.84	73.95
asthma	74	32.72	74.12
CHD	28	70.57	76.52
diabetes	50	60.62	74.36
osteoporosis	30	81.50	71.74

CHD 行の Patient Count セル (または Avg Age セル) をクリックして詳細リストを表示すると、以下のように表示されます。

#	PatientID	Age	Gender	Home City	Test Score
1	SUBJ_100446	44	M	Elm Heights	94
2	SUBJ_101284	56	F	Spruce	
3	SUBJ_100357	56	M	Centerville	93
4	SUBJ_100497	56	M	Pine	89
5	SUBJ_100597	58	F	Cedar Falls	
6	SUBJ_101281	58	F	Spruce	84
7	SUBJ_100733	58	M	Redwood	88
8	SUBJ_101265	59	F	Elm Heights	88

ただし、ピボット・テーブルの Avg Test Score セルをクリックして詳細リストを表示すると、以下のように表示レコードは少なくなります。

#	PatientID	Age	Gender	Home City	Test Score
1	SUBJ_100446	44	M	Elm Heights	94
2	SUBJ_100357	56	M	Centerville	93
3	SUBJ_100497	56	M	Pine	89
4	SUBJ_101281	58	F	Spruce	84
5	SUBJ_100733	58	M	Redwood	88
6	SUBJ_101265	59	F	Elm Heights	88

11
リストの定義

ここでは、Business Intelligence キューブでリストを定義する方法を説明します。

スタジオでは、リストの形式設定を定義できます。"キューブ・クラスのリファレンス情報" の "<listing>" を参照してください。

また、個別のリスト・フィールドを定義することもできます。ユーザはアナライザでこのフィールドを使用してカスタム・リストを作成できます。

Tipヒント キューブ定義の外部で (アーキテクトにアクセスすることなく) リストを定義することもできます。"リスト・グループの定義" を参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

11.1 リストの追加

リストを追加する手順は以下のとおりです。

1. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[新規項目名の入力]** に、リスト名を入力します。

   "モデル要素の名前" を参照してください。

3. **[リスト]** をクリックします。
4. [OK] をクリックします。
5. モデル・ビューワの **[リスト]** セクションでリスト名を選択します。
6. 必要に応じて、以下の詳細も指定します。
   - **[表示名]** ー リストのローカライズ可能な名前。この名前を指定しない場合、代わりに論理名が表示されます。
   - **[説明]** ー リストの説明。
   - **[リスト・タイプ]** ー リストのタイプ。**[テーブル]** (既定値) または **[マップ]** を選択します。

     [マップ] を選択する場合、"マップ・リストの定義" を参照してください。
   - **[SQL Select モード]** ー リストの取得に使用される SQL クエリの %SelectMode。これによって、アナライザまたはダッシュボード・ウィジェットでの特定のデータ型の表示方法が決まります (特に、日付と時刻)。既定の **[表示**

モード] のほか、[**論理モード**] (内部保存の場合に使用する形式) または [**ODBCモード**] でリストを表示することも選択できます。

これらのオプションの詳細は、"InterSystems SQL の基礎" ページの対応するセクションを参照してください。

- [**リソース**] – リストを保護するリソースを指定します。

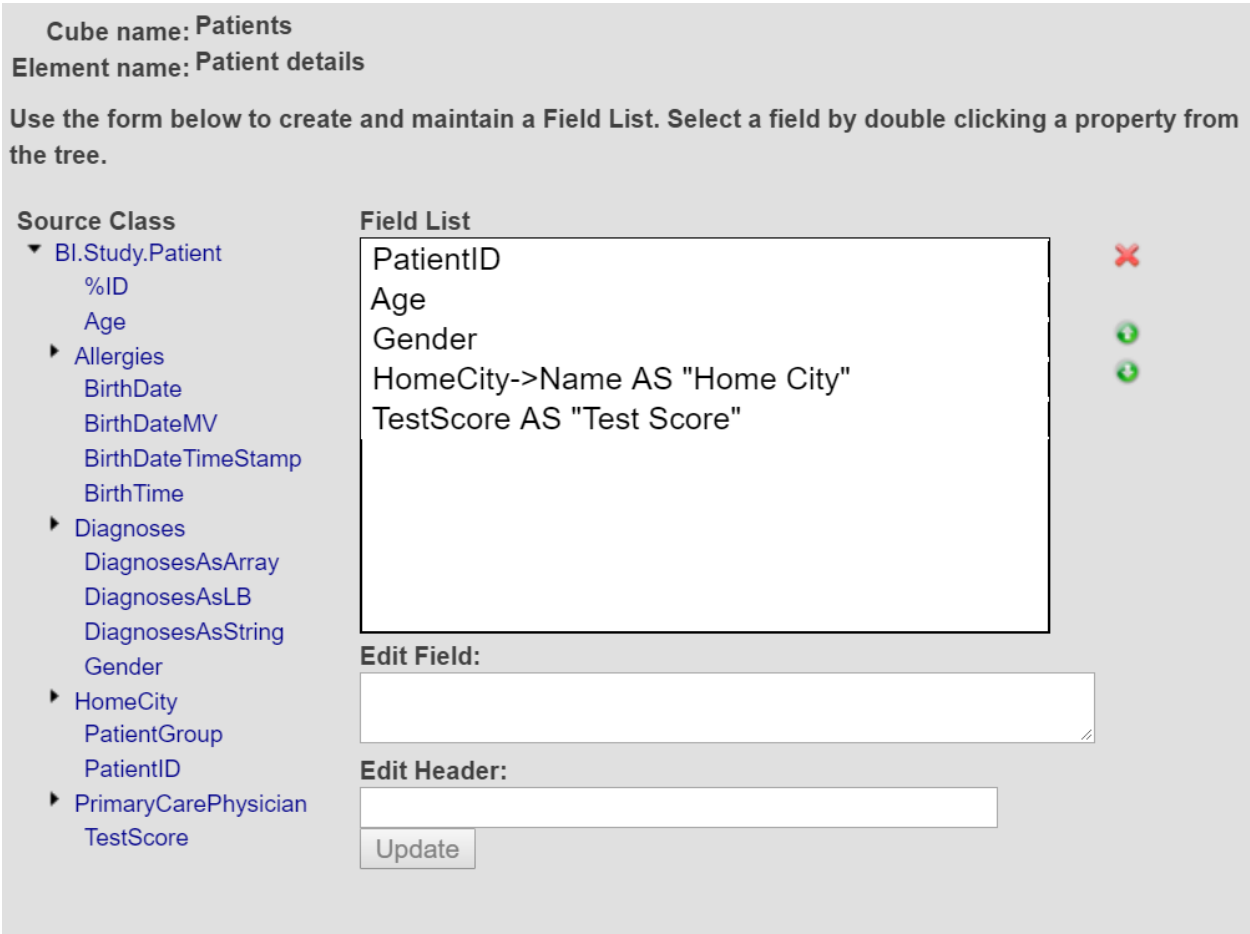
  この使用法については、"セキュリティの設定" を参照してください。

7. リストを定義します。手順は後続のセクションを参照してください。

11.2 単純なリストの定義

単純なリストは、キューブで使用されているソース・テーブルのフィールドを使用します。単純なリストを定義するには、このページの前半で示した説明に従ってリストを追加し、以下のオプションを指定します。

- [**フィールド・リスト**] – 表示するソース・テーブル内のフィールドのコンマ区切りリストを指定します。このオプションを指定するには、[**フィールド・リスト**] の横の検索ボタン をクリックして、フィールドを選択するダイアログ・ボックスを表示します。このようにするとアーキテクトでは以下のように表示されます。



このダイアログ・ボックスを使用する手順は以下のとおりです。

– 必要に応じて、[**ソースクラス**] ツリーの項目を展開します。

- プロパティを追加するには、[**ソースクラス**] ツリーでプロパティ名をダブルクリックします。このプロパティは [**フィールド・リスト**] に表示されるリストの最後に追加されます。既定のキャプションが付いたプロパティは、追加時にエイリアスが自動的に割り当てられます。CAPTION プロパティの詳細は、"主要なプロパティ・パラメータ" を参照してください。プロパティに表示される値は SqlFieldName です。これはプロパティ名と異なる場合があります。
- 項目を上または下に移動するには、[**フィールド・リスト**] でその項目をクリックし、必要に応じて上矢印または下矢印をクリックします。
- 項目を編集するには、[**フィールド・リスト**] でその項目をクリックし、[**フィールド編集** :] で変更して [**更新**] をクリックします。例えば、SQL エイリアスを追加できます。サブセクション "追加のオプション" を参照してください。
- 項目を削除するには、[**フィールド・リスト**] でその項目をクリックし、[X] ボタンをクリックします。

完了したら、[OK] をクリックします。

または、[**フィールド・リスト**] に直接値を入力します。以下はその例です。

```
PatientID,Age,Gender,HomeCity->Name AS "Home City",TestScore AS "Test Score"
```

[**リスト ― フィールド・リスト**] ダイアログ・ボックスを使用すると、ヘッダが自動的に $$$TEXT トークンでラップされることに注意してください。以下の例のように、$$$TEXT を使用してキューブのドメインが追加されます。

```
Product->Name AS "$$$TEXT["Product","HOLEFOODS"]"
```

別のドメインを [**フィールド・リスト**] に直接入力して、キューブの既定のドメインを上書きできます。

- [Order by] ― リストの並べ替え基準として使用するソース・テーブル内のフィールドのコンマ区切りリストを指定します (これらは [**フィールド・リスト**] に含める必要はありません)。全体的な並べ替えは、リストの最初のフィールドによって制御され、二次的な並べ替えは 2 番目のフィールドによって制御される、というように続きます。

  このオプションを指定するには、[**フィールド・リスト**] の横の検索ボタン  をクリックして、フィールドを選択するダイアログ・ボックスを表示します。このダイアログ・ボックスは、[**フィールド・リスト**] のダイアログ・ボックスの簡略バージョンです。

  または、[Order By] に直接値を入力します。以下はその例です。

  ```
  Age,Gender
  ```

  フィールド名の後に ASC または DESC キーワードを組み込んで、それぞれ昇順または降順で並べ替えることができます。

- [**データ・コネクタ**] フィールドと [**カスタム SQL クエリ**] フィールドは無視します。

注釈　リストの生成される SQL クエリは、確実に決まった順序 (%ID 順など) でレコードを返すとは限りません。指定のテーブルに対して InterSystems SQL クエリ・オプティマイザが選択するクエリ・プランは、さまざまな要因に基づいて変更される可能性があります (詳細は、"クエリ・パフォーマンスの調査" を参照してください)。このため、一貫した順序でレコードを提示したい場合は、並べ替え基準を明示的に指定する必要があります。これを行うには、アーキテクトで [Order by] オプションを設定するか、キューブのクラス定義で <listing> 要素に orderBy 属性の値を設定します。

11.2.1 追加のオプション

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。

  ```
  PatientID,HomeCity,PrimaryCarePhysician->DoctorGroup
  ```

- エイリアスを組み込むことができます。

  ```
  PatientID,Age,Gender,HomeCity->Name AS "Home City",TestScore AS "Test Score"
  ```

  または、以下のようにします。

  ```
  PatientID,Age,Gender,HomeCity->Name "Home City",TestScore "Test Score"
  ```

- 特殊なトークン **$$$TEXT[]** を使用して指定することで、エイリアスをローカライズできます。以下はその例です。

  ```
  %ID,DateOfSale As "$$$TEXT["Date Of Sale"]"
  ```

- 関数名を括弧で囲み、フィールド名として解釈されないようにすると、標準 SQL 関数と InterSystems SQL 関数を使用できます。

  ```
  (UCASE(PatientID)),%EXTERNAL(Gender)
  ```

- 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。

  ```
  %ID,'$'||source.Sales AS Sales
  ```

  この例では、**Sales** 列に **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。

Tipヒント　既定では、SQL 述語節で比較が評価される場合、InterSystems IRIS® では、論理 (内部保存) 値が使用されます。代わりに値をその表示形式または ODBC 形式で使用するには、SQL 関数 %EXTERNAL または %ODBCOUT をそれぞれ使用します。

11.3 データ・コネクタ・リストの定義

データ・コネクタ・リストは、データ・コネクタのフィールドを使用します。このようなリストを定義するには、このページの前半で示した説明に従ってリストを追加し、以下のオプションを指定します。

- **[データ・コネクタ]** ― 使用するデータ・コネクタ・クラスを選択します。
- **[フィールドリスト]** ― 組み込むフィールドを指定します。このフィールドは、データ・コネクタ内のフィールドとする必要があります。エイリアス、SQL 関数、または矢印構文を含めることはできません。

  既定では、リストは **%ID** という名前のプロパティのみを表示します (存在する場合)。
- [Order by] ― 使用されません。
- **[カスタム SQL クエリ]** フィールドは無視します。

11.4 SQL カスタム・リストの定義

SQL カスタム・リストは、キューブで使用されるソース・テーブル以外のテーブルからのフィールドの使用が可能です。(データ・コネクタからのフィールドの使用は不可能なので注意してください。)

注釈　システムでは、もう 1 つの種類のカスタム・リスト (アナライザでユーザによって定義されたリスト) がサポートされています。"キューブ・クラスのリファレンス情報" の "<listingField>" を参照してください。

SQL カスタム・リストを定義するには、このページの前半で示した説明に従ってリストを追加し、以下のオプションを指定します。

- **[カスタム SQL クエリ]** ― これを選択します。
- **[カスタム SQL]** ― このセクションで示した説明に従って SQL SELECT クエリを指定します。

[カスタム SQL] クエリの詳細を確認する前に、システムがリストを作成する方法を理解する必要があります。どのリストについても、現在のコンテキスト (ユーザがリストを要求するコンテキスト) で使用されるファクトに対応するソース ID 値のセットを含む一時リスト・テーブルが生成されます。カスタム SQL クエリは、一時リスト・テーブルでレコードとソース ID が一致するフィールドを選択します。内部的には、クエリ全体は以下のようになります。

```
SELECT source.Field1,source.Field2
FROM BI_Study.Patient source,internal-listing-table-name list
WHERE source.%ID=list.sourceID AND list.queryKey='2144874459'
```

SQL カスタム・クエリを指定する際は、これらの詳細の一部を置換するトークンを指定します。具体的に、**[カスタム SQL]** では、以下の基本的な形式で値を指定します。

```
SELECT list of field names  FROM $$$SOURCE, othertable AS alias  WHERE $$$RESTRICT AND otherrestriction
```

以下は、この指定の説明です。

- FROM 節は、クエリする複数のテーブルを指定します。この節の `$$$SOURCE` トークンは、キューブのソース・テーブルのエイリアスとして `source` を確立し、一時リスト・テーブルのエイリアスとして `list` を確立します。

  `$$$SOURCE` は内部で `BI_Study.Patient source,internal-listing-table-name list` などに置換されます。

  また、othertable AS alias は、クエリする他のテーブル、およびこのテーブルのエイリアス (必要な場合) を指定します。さらに他のテーブルも同様に指定できます。

- WHERE 節は、ソース・テーブルをリスト・テーブルおよび他のテーブルに結合する条件を指定します。

  `$$$RESTRICT` トークンは、ソース・テーブルを一時リスト・テーブルに結合する条件によって置換されます。内部でこれは `source.%ID=list.sourceID AND list.queryKey='2144874459'` など、コンテキストに依存するものに置換されます。

  また、otherrestriction は、他の SQL 制限を指定します。上記の構文では、制限が AND で結合されています。必要に応じて、OR を代わりに使用して、より複雑な WHERE 節を作成できます。

- list of field names では、フィールド名のコンマ区切りリストを使用します。これは、FROM 節にリストされている任意のテーブルのフィールドにすることができます。キューブのソース・テーブルのフィールドでは、`source.`fieldname の形式の参照を使用します。矢印構文も同様に組み込むことができます。

- エイリアスを含めることもできます。また、特殊なトークン `$$$TEXT[]` を使用して指定することで、エイリアスをローカライズできます。以下はその例です。

  ```
  SELECT ID,UnitsSold As "$$$TEXT["Units Solds"]" FROM $$$SOURCE WHERE $$$RESTRICT
  ```

- クエリの最後に ORDER BY 節を含めることもできます。

簡単な例 (キューブのソース・テーブルのみを使用) :

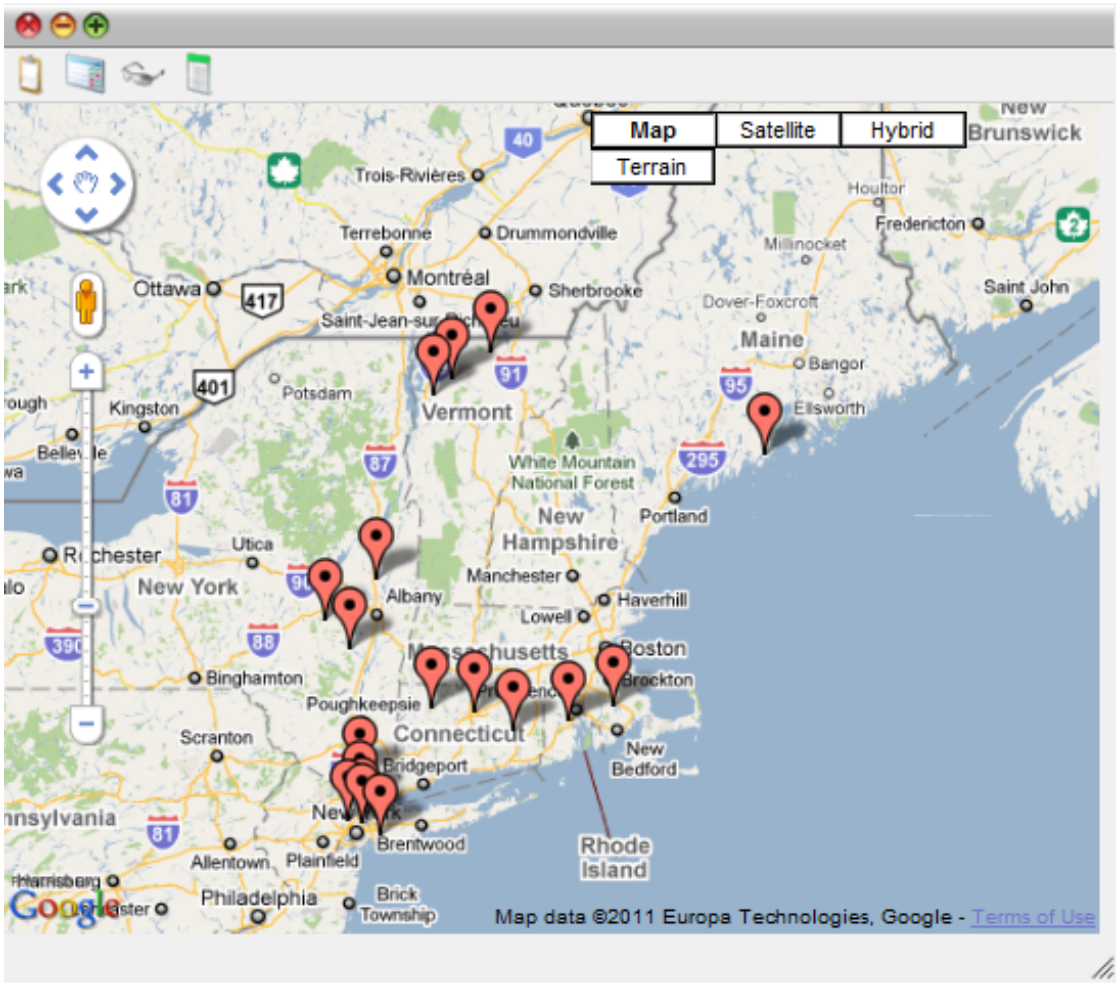
```
SELECT source.PatientID,source.Age,source.HomeCity->Name
FROM $$$SOURCE
WHERE $$$RESTRICT
```

他の例（他のテーブルのデータを使用）：

```
SELECT source.PatientID,FavoriteColor
FROM $$$SOURCE, BI_Study.PatientDetails AS details
WHERE $$$RESTRICT AND source.PatientID=details.PatientID
```

11.5 マップ・リスト（地理リスト）の定義

既定では、[**リスト・タイプ**] は [**テーブル**] で、リストに情報のテーブルが表示されます。適切なデータがある場合、代わりに [**リスト・タイプ**] を [**マップ**] に指定できます。この場合、リストは、リスト・データに含まれている地理的な場所を示すマーカが付いたマップです。例えば、マップで営業場所や顧客の位置をハイライト表示することができます。以下はその例です。



重要　　マップ・リストでは Google Maps API が使用されます。この API が利用条件（Terms of Use）に従って使用されていることを確認してください。利用条件には、このリストに表示されているリンクからアクセスできます（上図を参照）。

Google Maps API を使用するには、API キーを取得する必要があることに注意してください。詳細は、"基本設定の指定" を参照してください。

このようなリストを定義するには、このページで前述した一般的な手順を使用して、以下の操作を実行します。

- [**リスト・タイプ**] を [**マップ**] に指定します。

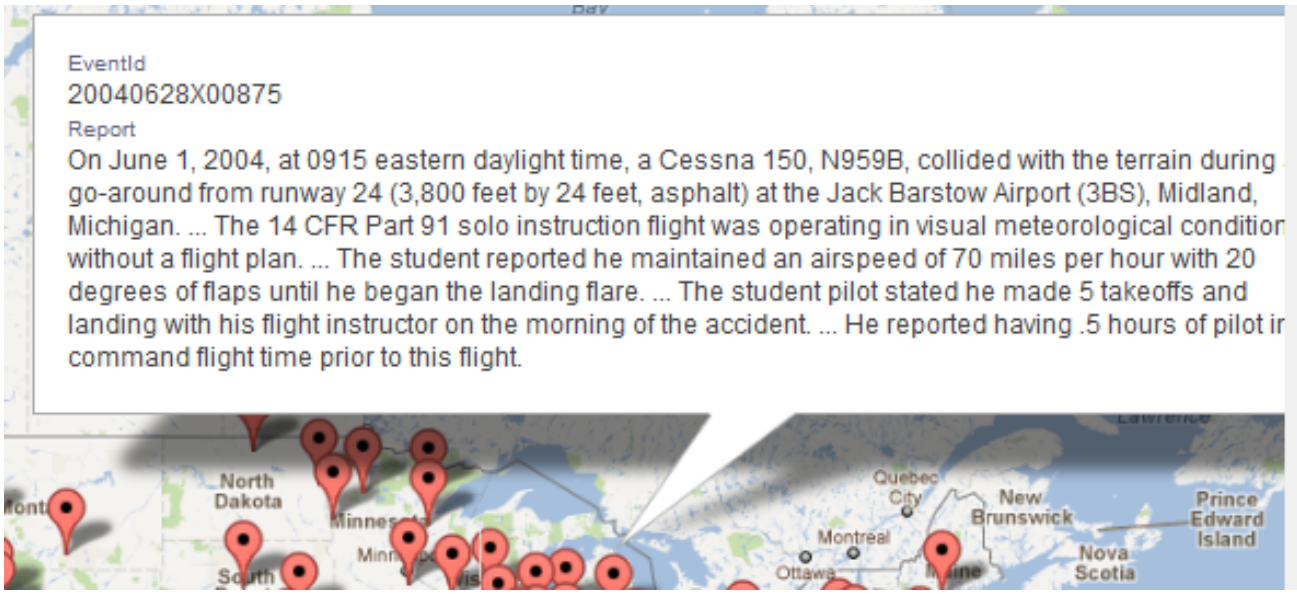
- フィールド `Latitude` および `Longitude` が含まれるように、リスト・クエリを定義します（大文字と小文字は区別されます）。これらのフィールドには、それぞれに適切な緯度と経度が 10 進形式で格納されている必要があります（度/分/秒の形式ではありません）。

  例えば、[SQL クエリ] を以下のように設定できます。

  ```
  EventId, $$$IKSUMMARY as Summary, LocationCoordsLatitude As Latitude, LocationCoordsLongitude As
  Longitude
  ```

  この例は、Aviation デモからの引用です。"キューブでの Text Analytics の使用法" を参照してください。

このリスト・クエリには、他のフィールドも含めることができます。追加したフィールドは、ユーザがマップの位置をクリックしたときに、バルーンに表示されます。以下はその例です。



この例は、Aviation デモからの引用です。"キューブでの Text Analytics の使用法" を参照してください。

12
リスト・フィールドの定義

ここでは、個別のリスト・フィールドを Business Intelligence キューブに追加する方法を説明します。個別のリスト・フィールドを使用すると、アナライザでカスタム・リストを作成できます。

"リストの定義" も参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

12.1 ユーザ定義のカスタム・リストについて

個別のリスト・フィールドの目的は、ユーザがアナライザでカスタム・リストを定義できるようにすることです。このプロセスは、"カスタム・リストの作成" で説明されていますが、以下に大まかな手順を示します。

1. 最初に、既定のリストを表示します。
2. アナライザの左側の領域で **[詳細リスト]** を選択します。
3. **[カスタム・リスト]** フォルダを展開します。以下はその例です。

4. ユーザは、この領域から [**カスタム・リスト・フィールド**] ボックスにリスト・フィールドをドラッグできます。これにより、リスト・クエリが実行され、その結果が表示されます。

12.2 リスト・フィールドの作成

リスト・フィールドを作成する手順は以下のとおりです。

1. [**要素を追加**] をクリックします。

   ダイアログ・ボックスが表示されます。

2. [**新規項目名の入力**] に、リスト名を入力します。

   "モデル要素の名前" を参照してください。

3. [ListingField] をクリックします。

4. [OK] をクリックします。

5. モデル・ビューワの [**リスト・フィールド**] セクションでリスト名を選択します。

6. 必要に応じて、以下の詳細も指定します。

   - [**表示名**] ー このリスト・フィールドのローカライズ可能な名前。この名前を指定しない場合、代わりに論理名が表示されます。
   - [**説明**] ー このリスト・フィールドの説明。
   - [**フィールド式**] ー ソース・テーブルのフィールドまたは (矢印構文などを使用して) 関連テーブルのフィールドを参照する SQL 式。次の例では (Hole Foods Sales キューブから抜粋)、矢印構文を使用しています。

     ```
     (Product->Category)
     ```

     エイリアスを指定しない場合、ここに示すように、式を括弧で囲む必要があります。

     エイリアスを指定する場合、括弧は不要です。以下はその例です。

     ```
     Product->Category Category
     ```

   - [**リソース**] ー このリスト・フィールドを保護するリソースを指定します。

     この使用法については、"セキュリティの設定" を参照してください。

   [**フィールド式**] には、SQL 関数を使用できます (%EXTERNAL などの InterSystems SQL 関数を含む)。ユーザがアナライザでカスタム・リストを作成すると、システムは選択したリスト・フィールドの [**フィールド式**] のコンマ区切りリストを含む SQL SELECT 文を作成します。

   [**フィールド式**] では、SQL エイリアスを指定できます。ただし、これを行うと、[**表示名**] は無視され、フィールド名をローカライズできません。

ドラッグ・アンド・ドロップ操作を使用してリスト・フィールドを作成することもできます。そのためには、左側の領域から [**リスト・フィールド**] 見出しにフィールドをドラッグ・アンド・ドロップします。次に、この新しいリスト・フィールドを選択して、右側の領域で必要に応じて編集します。

異なるソース・フィールドを [**フィールド式**] にドラッグ・アンド・ドロップすることもできます。この操作を行うと、その値が新しいフィールドに置換されます。

13
計算メンバの定義

ここでは、計算メンバ (計算メジャーを含む) を Business Intelligence キューブに追加する方法を説明します。

注釈　ユーザは、アナライザ内で計算メジャー、計算メンバ、および名前付きセットを追加作成できます。

MDX の用語では、計算メジャーは、単に計算メンバの別の表現です。このドキュメントでは、わかりやすくするために非標準的な語句である計算メジャーを使用します。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

13.1 計算メジャーの定義

計算メジャーを追加する手順は以下のとおりです。

1. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。
2. **[新規項目名の入力]** に、メジャーの名前を入力します。

   "モデル要素の名前" を参照してください。
3. **[計算メンバ (メジャー)]** をクリックします。
4. [OK] をクリックします。
5. モデル・ビューワで計算メジャーを選択します (**[計算メンバ]** セクション内)。
6. 新しいメンバを定義する MDX 式を指定します。そのためには、以下のいずれかの操作を行います。
   - **[値式]** に式を直接入力します。
   - Expression Builder を使用します。このツールは、キューブの任意の部分に対して正しい MDX 識別子を容易に取得できるようにすることを目的としています。このツールにアクセスするには、**[式]** の横にある虫眼鏡をクリックします。左側の領域には、すべてのメジャーおよびレベルを含むキューブのコンテンツがリストされます。右側の領域には、作成している式が表示されます。項目を式に追加するには、項目を左側の領域から式にドラッグ・アンド・ドロップします。項目は、式の最後に追加されますが、式の別の部分に移動することも可能です。

   以下のセクションでは、いくつかの式の例を示しています。その他の例については、HoleFoods キューブおよび Patients キューブを参照してください。

アーキテクトの **[メジャー]** グループに、この新しいメジャーが他のメジャーと共に表示されます。

このページの後半にある "計算メジャーのリストの追加フィルタの指定" も参照してください。

13.2 計算メジャーの MDX レシピ

計算メジャーでは、定義として使用する MDX 式が数値式であることが要求されます。MDX で数値式を作成するすべての方法については、"数値式" のセクションを参照してください。

InterSystems MDX の概要は、"InterSystems MDX の使用法" を参照してください。

このセクションでは、以下のシナリオのレシピについて説明します。

- 他のメジャーに基づくメジャー
- ピボット変数を乗数として使用するメジャー
- 集約値のパーセンテージを表示するメジャー
- 個別のメンバ数をカウントするメジャー
- 準加法メジャー
- フィルタ処理メジャー (場合によっては NULL)
- KPI またはプラグインを使用するメジャー ("InterSystems Business Intelligence の上級モデリング" を参照)

サンプルは、HoleFoods キューブおよび Patients キューブを参照してください。

注釈　あるプラグインに基づく別の計算メジャーに基づく計算メジャーを定義しないでください (プラグインの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください)。

13.2.1 他のメジャーに基づくメジャー

以下のような**式**を介して、あるメジャーが別のメジャーに基づくようにする方法が一般的です。

```
([MEASURES].[my measure 1] + [MEASURES].[my measure 2]) / [MEASURES].[my measure 3]
```

より正式には、**式**は数値で評価される MDX 式で、以下の要素を含めることができます。

- メジャーへの参照。構文は、以下のとおりです。

  ```
  [MEASURES].[measure name]
  ```

  または、以下のようにします。

  ```
  MEASURES.[measure name]
  ```

  メジャー名が英数字のみで構成され、先頭が数字でなく、MDX の予約語でない場合は、メジャー名を囲む角括弧を省略できます。

  式では、大文字と小文字は区別されません。
- 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ることはできません。
- ピボット変数。"ピボット変数の定義と使用" を参照してください。

ピボット変数を参照するには、構文 `$variable.`variablename を使用します。variablename はロジカル変数名です。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。

- 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、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も使用できます。

  例えば、`MEASURES.[%COUNT] / 100` です。

- 数値を返す MDX 関数。AVG、MAX、COUNT など、MDX 関数の多くが数値を返します。詳細は、"InterSystems MDX リファレンス" を参照してください。関数名では、大文字と小文字は区別されません。

Tipヒント MDX 関数 IIF は、ゼロによる除算を回避する場合など、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。

13.2.2 ピボット変数を乗数として使用するメジャー

ピボット変数を乗数として使用するメジャーを定義するには、以下のような **[式]** を使用します。

```
measures.[measure A]*$variable.myQueryVariable
```

myQueryVariable は、ピボット変数のロジカル名です。このシナリオでは、数値を提供するリテラル・ピボット変数を使用します。"ピボット変数の定義と使用" を参照してください。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。

13.2.3 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 サブクエリを実行し、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**式**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 MEASURES.[measure A] ON 0 FROM mycube")
```

このサブクエリ `SELECT MEASURES.[measure A] ON 0 FROM mycube` は、指定されたメジャーをキューブから選択して、そのメジャーをすべてのレコード対して集約します。

以下はその例です。

```
100 * MEASURES.[%COUNT]/%MDX("SELECT MEASURES.[%COUNT] ON 0 FROM patients")
```

Count メジャーの場合は、より単純な以下のサブクエリを使用できます。

```
100 * MEASURES.[%COUNT]/%MDX("SELECT FROM patients")
```

以下は、`Percent of All Patients` 計算メジャーを使用する例です。このメジャーは前述の**式**で定義されています。

Diagnoses	Patient Count	Percent of All Patients
None	843	84.30
asthma	55	5.50
CHD	42	4.20
diabetes	46	4.60
osteoporosis	37	3.70

13.2.4 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 Group と Doctor が含まれています。計算メジャー Unique Doctor Count では、以下の**式**を使用します。この式では、Doctor レベルを使用しています。

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

このメジャーは、ピボット・テーブルで以下のように使用できます。

Decade	Patient Count	Unique Doctor Count
None	206	143
1910s	78	63
1920s	215	156
1930s	515	269
1940s	709	321
1950s	1,034	348
1960s	1,414	381
1970s	1,530	387
1980s	1,347	378
1990s	1,421	381
2000s	1,399	382
2010s	132	90

13.2.5 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、**式**には次の式を使用します。

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

%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、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。

13.2.6 フィルタ処理メジャー

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には、次のような**式**を使用します。

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

この場合、AGGREGATE 関数は、指定されたメンバに属するすべてのレコードを通じて、指定された値を集約します。

例えば、Patients サンプルに `Avg Test Score` メジャーがあります。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**式**を指定した計算メジャーを作成できます。

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

13.2.7 KPI またはプラグインを使用するメジャー

任意の KPI またはプラグイン (すべて説明済み) に対して、そこから値を取得する計算メジャーを作成できます。このメジャーは、ドラッグして、アナライザにドロップできます。このような計算メジャーを作成するには、**[式]** で以下の形式の MDX 式を使用します。

```
%KPI(kpiname,propertyname,seriesname,"%CONTEXT")
```

kpiname は KPI またはプラグインの名前、propertyname はプロパティまたは列の名前、seriesname は系列の名前です。seriesname は省略できます。省略した場合、この関数は KPI またはプラグインの最初の系列にアクセスします。

MDX ベースの KPI またはプラグインの場合、コンテキスト情報を格納するパラメータを指定できます。`"%CONTEXT"` は、KPI またはプラグインに行、列、およびフィルタのコンテキストを提供する特殊なパラメータです。この情報は、KPI またはプラグインで使用されるベース MDX クエリに渡されます。このパラメータの既定値は `"all"` で、行、列、およびフィルタのコンテキストを組み合わせて使用します。他のオプションの詳細は、"InterSystems MDX リファレンス" の "%KPI" 関数を参照してください。

例 (系列が 1 つだけの KPI またはプラグインの場合) :

```
%KPI("PluginDemo2","Count",,"%CONTEXT")
```

別の例として、サンプル中央値プラグイン (%DeepSee.PlugIn.Median) を使用する計算メジャーを定義できます。そのためには、以下の**式**を使用します。

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

13.3 計算メンバ (メジャー以外) の定義

メジャーでない計算メンバを追加する手順は以下のとおりです。

1. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[新規項目名の入力]** に、メンバの名前を入力します。

   "モデル要素の名前" を参照してください。

3. **[計算メンバ (ディメンジョン)]** をクリックします。

4. [OK] をクリックします。

5. モデル・ビューワで計算メンバを選択します (**[計算メンバ]** セクション内)。

6. **[ディメンジョン]** に、このメンバが属するディメンジョンの名前を入力します。

   非計算メンバが含まれている既存のディメンジョンや新規のディメンジョンなど、任意のディメンジョンを指定できます。

7. 新しいメンバを定義する MDX 式を指定します。そのためには、以下のいずれかの操作を行います。

   - **[値式]** に式を直接入力します。
   - Expression Builder を使用します。このツールは、キューブの任意の部分に対して正しい MDX 識別子を容易に取得できるようにすることを目的としています。このツールにアクセスするには、**[式]** の横にある虫眼鏡をクリックします。左側の領域には、すべてのメジャーおよびレベルを含むキューブのコンテンツがリストされます。右側の領域には、作成している式が表示されます。項目を式に追加するには、項目を左側の領域から式にドラッグ・アンド・ドロップします。項目は、式の最後に追加されますが、式の別の部分に移動することも可能です。

次のセクションには、いくつかのレシピが用意されています。

詳細と例は、"InterSystems MDX リファレンス" の "WITH 節" を参照してください。

Patients キューブはいくつかのサンプルを定義します。標準メンバの他に 2 つの計算メンバを含む `ColorD` ディメンジョンを参照してください。

13.4 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。

一般的な構文は、"InterSystems MDX リファレンス" の "WITH 節" を参照してください。

InterSystems MDX の概要は、"InterSystems MDX の使用法" を参照してください。

このセクションでは、以下のシナリオのレシピについて説明します。

- 年齢メンバの定義
- 複数のメンバの集約
- 日付範囲の集約
- 用語リストにより定義されるメンバの集約
- 複数のレベルのメンバの結合 (特にフィルタリング)

サンプルは、Patients キューブを参照してください。標準メンバの他に 2 つの計算メンバを含む `ColorD` ディメンジョン。

13.4.1 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**式**を使用して、3 Months Ago という名前の計算メジャーを定義することができます。

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

一連の年齢メンバを定義して、年齢別にグループを作成できます。例えば、以下のメンバを定義できます。

- [ディメンジョン]:AgeGroups

  [メンバ名]:1　　2

  [式]:%OR(DateOfSale.DaySold.[NOW-2y-1d]:[NOW-1y])

- [ディメンジョン]:AgeGroups

  [メンバ名]:2　　3

  [式]:%OR(DateOfSale.DaySold.[NOW-3y-1d]:[NOW-2y])

詳細およびオプションは、"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。

13.4.2 メンバの集約

多くの場合、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**式**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,...})
```

以下はその例です。

```
%OR({[colord].[h1].[favorite color].[red],
[colord].[h1].[favorite color].[blue],
[colord].[h1].[favorite color].[yellow]})
```

どんな場合でも、非メジャー・メンバは、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。

13.4.3 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])
```

%TIMERANGE 関数も使用できます。これによって、片側だけ指定されている範囲のすべてのメンバで構成されるメンバを定義できます。例えば、以下の式により 2009 メンバの後に開始する範囲を定義します。

```
%TIMERANGE(DateOfSale.YearSold.&[2009],,EXCLUSIVE)
```

%TIMERANGE 関数は時間レベルでのみサポートされ、リレーションシップではサポートされていません。

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]))
```

13.4.4 条件リストにより定義されるメンバの集約の定義

条件リストにより、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。条件リストは、キーと値のペアの単純なリスト (ただし、拡張可能) です (″条件リストの定義″ を参照してください)。

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**式**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。

13.4.5 複数のディメンジョンに対してフィルタ処理するメンバの定義

メンバベース・フィルタは非常に役立つため、フィルタでの使用専用のメンバを作成するだけの価値があります。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].[H1].[YEAR].[NOW],[ALLERSEVD].[H1].[ALLERGY SEVERITIES].[003 LIFE-THREATENING]}
```

式 ([BIRTHD].[H1].[YEAR].[NOW],[ALLERSEVD].[H1].[ALLERGY SEVERITIES].[003 LIFE-THREATENING]) はタプル式で、メンバ [BIRTHD].[H1].[YEAR].[NOW] とメンバ [ALLERSEVD].[H1].[ALLERGY SEVERITIES].[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。

あるいは、より一般的に、以下の形式の式を使用します。

```
%OR({set_expression})
```

13.5 計算メジャーのリストの追加フィルタの指定

既定では、ユーザが詳細リストを表示する際、システムは、現在のコンテキスト (リストが要求されたコンテキスト) で使用されているソース・レコードごとに行を 1 つ表示します。指定された計算メジャーについて、詳細リストを表示する際にシステムが使用する追加フィルタを指定できます。

特定の計算メジャーについて、リストの追加フィルタを指定するには、以下の手順を実行します。

1. モデル・ビューワで、計算メジャーを選択します。
2. [詳細] 領域 で、**[リスト・フィルタ]** を MDX フィルタ式として指定します。式を作成するには、このフィールドに直接入力するか、虫眼鏡をクリックして、エディタで式を作成します。

   MDX フィルタ式の詳細および例は、"サブジェクト領域のフィルタ処理" を参照してください。

また、通常メジャーには、リストを表示する際に使用する追加フィルタを含めることができます。構文は異なりますが、実質的な効果は同じになります。詳細および例は、"リストの追加フィルタの指定" を参照してください。

14

名前付きセットの定義

ここでは、名前付きセットを Business Intelligence キューブに追加する方法を説明します。

"このドキュメントに示したサンプルのアクセス方法" を参照してください。

名前付きセットを追加する手順は以下のとおりです。

1. **[要素を追加]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[新規項目名の入力]** に、セットの名前を入力します。

   "モデル要素の名前" を参照してください。

3. **[名前付きセット]** をクリックします。

4. **[OK]** をクリックします。

5. モデル・ビューワで名前付きセットを選択します (**[名前付きセット]** セクション内)。

6. **[セット式]** で、MDX セット式を指定します。

   詳細は、"InterSystems MDX リファレンス" の "セット式" を参照してください。

   サンプルは、HoleFoods キューブおよび Patients キューブを参照してください。

15
サブジェクト領域の定義

ここでは、Business Intelligence サブジェクト領域を定義する方法を説明します。

注釈　現在のリリースでは、サブジェクト領域のリレーションシップを非表示にすることはできません。名前付きセットおよび計算メンバ (計算メジャーを含む) の非表示、追加、およびオーバーライドを行うには、スタジオを使用します。"サブジェクト領域クラスのリファレンス情報" を参照してください。

"複合キューブの定義" も参照してください。

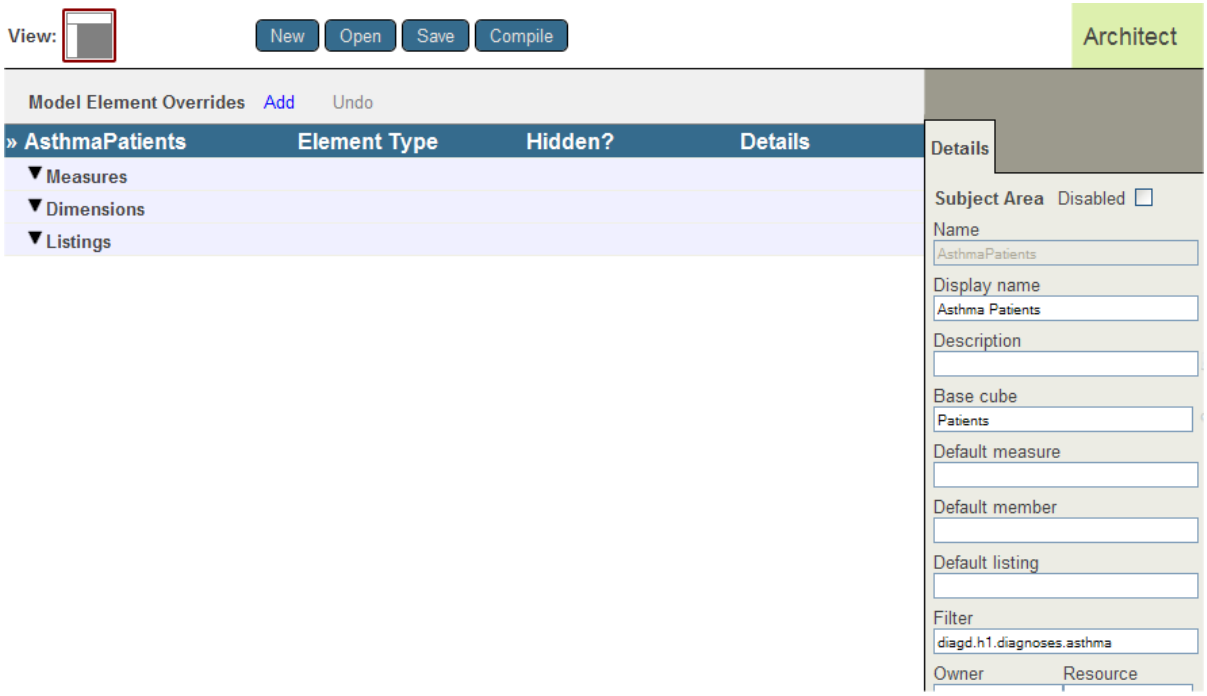
"このドキュメントに示したサンプルのアクセス方法" も参照してください。

15.1 アーキテクトのサブジェクト領域の概要

ここでは、キューブではなくサブジェクト領域を表示している場合のアーキテクトの表示内容について説明します。

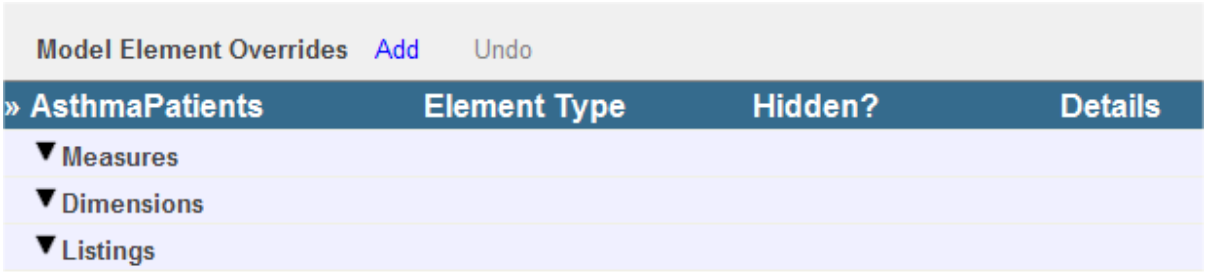
1. [開く]→[サブジェクト領域]→[AsthmaPatients]→[OK] をクリックします。

   これで、キューブの表示と同様のページが表示されます。ただし、キューブとは異なり左側にクラス・ビューワは表示されません。



15.1.1 モデル・ビューワ

左の領域はモデル・ビューワで、サブジェクト領域に定義されたオーバーライドが表示されます。以下はその例です。



このサブジェクト領域ではオーバーライドやリストを定義しません（多くのサブジェクト領域には、フィルタのみが組み込まれています）。

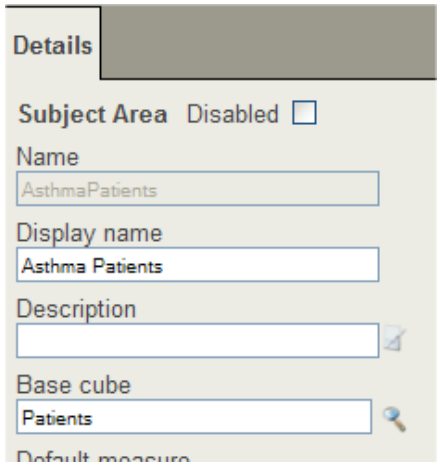
ここでは、オーバーライドの追加、編集のための既存のオーバーライドの選択、およびオーバーライドの削除を実行できます。

新規リストを定義することもできます。

15.1.2 詳細領域

右の領域は、詳細領域で、モデル・ビューワで現在選択されている要素の詳細（選択されている場合）、またはサブジェクト領域の詳細（選択されていない場合）が表示されます。

以下はその例です。



15.2 サブジェクト領域の定義

サブジェクト領域を定義する手順は以下のとおりです。

1. アーキテクトで、[**新規作成**] をクリックします。
2. [**サブジェクト領域**] をクリックします。
3. 少なくとも、以下の情報は入力してください。
   - [**サブジェクト領域名**] – 既定のキャプションとして使用したり、クエリで使用するサブジェクト領域の名前。
   - [**サブジェクト領域のクラス名**] – サブジェクト領域クラスの完全なパッケージおよびクラス名。
   - [**ベース・キューブ**] – このサブジェクト領域の基となるキューブの論理名。

     キューブ名は入力することも、[**参照**] をクリックしてキューブを選択することもできます。

     また、キューブのコンマ区切りリストも使用できます。"複合キューブの定義" を参照してください。

   その他のオプションは、このページの後続部分で説明します。

   クラス名のほか、サブジェクト領域の作成後はすべてのオプションを編集できます。
4. [OK] をクリックします。
5. アーキテクトでサブジェクト定義をクリックして選択します。その後で、詳細領域の [**依存**] オプションを編集します。値として、このサブジェクト領域の基となるキューブ・クラスの完全なパッケージとクラス名を指定します。

   サブジェクト領域クラスは常に、キューブ・クラスの後にコンパイルする必要があります。これを制御するには、[**依存**] 設定が役に立ちます。
6. 必要に応じて、サブジェクト領域を保存します。そのためには、以下の操作を実行します。
   a. [**保存**] をクリックします。
   b. [OK] をクリックします。

   これで、クラスが作成されます。

または、"サブジェクト領域クラスのリファレンス情報" の説明に従ってクラスを手動で作成します。

15.3 サブジェクト領域のフィルタ処理

ほとんどの場合、データへのアクセスの制御にはサブジェクト領域を使用します。このような場合は、サブジェクト領域にフィルタを指定します。このためには、以下のいずれかまたは両方を実行します。

- ハードコードされたフィルタを指定することができます。そのためには、アーキテクトの [**フィルタ**] オプションで値を指定します。このオプションで、このサブジェクト領域のフィルタとして使用する MDX フィルタ式を指定します。

  ここでは、ハードコードされたフィルタを作成する方法について説明します。

- 値が実行時に決定されるフィルタを指定できます。そのためには、サブジェクト領域クラスの %OnGetFilterSpec() メソッドをカスタマイズします。このメソッドは、ハードコードされたフィルタが存在する場合は、そのフィルタにアクセスして変更することができます。

  詳細は、"キューブまたはサブジェクト領域の動的フィルタ処理" を参照してください。

注釈　以下のサブセクションは、サブジェクト領域のフィルタだけではなく、MDX フィルター般に適用されます。

15.3.1 アナライザでのフィルタの構築とモデルへのコピー

MDX に習熟していない場合、最も簡単にフィルタ式を指定する方法は以下のとおりです。

1. "アナライザの使用法" の説明に従って、ドラッグ・アンド・ドロップ操作でフィルタを作成します。
2. クエリ文字列ボタン  をクリックします。

   アナライザに以下のようなテキストが示されたダイアログ・ボックスが表示されます。

   ```
   SELECT FROM [PATIENTS] WHERE [AGED].[H1].[AGE GROUP].&[0 TO 29]
   ```

3. アナライザでのそれまでの実行内容によっては非常に複雑になる場合があるため、**WHERE** の前のテキストはすべて無視します。**WHERE** 以降のすべてのテキストを、システムのクリップボードまたはその他の一時的な場所にコピーします。
4. コピーしたテキストをサブジェクト領域の [**フィルタ**] オプションに貼り付けます。

   以下はその例です。

   ```
   [AGED].[H1].[AGE GROUP].&[0 TO 29]
   ```

15.3.2 フィルタ式の作成

ここでは、フィルタ式の一般的な形式について説明します。

"InterSystems MDX リファレンス" も参照してください。

15.3.2.1 単純なフィルタ

単純なフィルタは、単一のメンバを参照します。

```
[AGED].[H1].[AGE GROUP].&[0 TO 29]
```

このフィルタは年齢範囲が 0 ～ 29 歳の患者のみにアクセスします。

上記の式は MDX メンバ式です。以下の詳細に注意してください。

- [AGED] はこのサンプルのディメンジョンです。
- [H1] は、このディメンジョンの階層です。
- [AGE GROUP] はこの階層のレベルです。
- [0 TO 29] は、このレベルのメンバのキーです。

  上記のメンバ式は、[AGED] ディメンジョン、[H1] 階層、[AGE GROUP] レベルの [0 TO 29] メンバを参照します。

また、以下のさまざまな時間節約の可能性も考慮してください。

- これらの ID は、いずれも適宜大文字、小文字、大文字と小文字の混合で指定できます。MDX では大文字と小文字は区別されません。
- スペース文字を含まない ID はすべて、角括弧を省略できます。
- 生成される式が曖昧にならなければ、階層およびレベル名は省略できます。

  ディメンジョン名は必ず含める必要があります。
- メンバ・キーではなく、メンバ名を使用することもできます。ほとんどの場合、これは、メンバ ID からアンド記号 (&) を単純に省略できることを意味します。

15.3.2.2 メンバ・セットによるフィルタ

複数のメンバが必要な場合もあります。以下はその例です。

```
{[aged].[h1].[age group].[0 to 29],[aged].[h1].[age group].[30 to 59]}
```

このサブジェクト領域は、セットの要素のいずれかに属するファクト・テーブルのすべてのレコードが含まれるようにフィルタ処理されます。このサブジェクト領域は、年齢が 0 ～ 29 歳のグループの患者と年齢が 30 ～ 59 歳のグループの患者、つまり 60 歳未満のすべての患者を対象とします。

重要　　このリストは中括弧で囲まれることに注意してください。これは、リストが MDX のセットであることを示します。セットとは、要素の結合です。

15.3.2.3 タプルによるフィルタ

複数の同時基準を満たすレコードのみを検出することが必要な場合もあります。この場合はタプル式を使用します。以下はその例です。

```
([aged].[h1].[age group].[60+],[diagd].[h1].[diagnoses].[diabetes])
```

このサブジェクト領域は、60 歳以上の糖尿病の患者を対象とします。

重要　　このリストは括弧で囲まれることに注意してください。これは、リストが MDX のタプルであることを示します。タプルは、要素の共通部分です。

15.3.2.4 複数のタプルによるフィルタ

1 つのセット内に複数のタプル式をリストすることができます。以下はその例です。

```
{([aged].[h1].[age group].[60+],[diagd].[h1].[diagnoses].[diabetes]),
([colord.].[h1].[color].[red],[allergd].[h1].[allergy].[soy])}
```

このサブジェクト領域は、以下の基準の 1 つ以上を満たすすべての患者を対象とします。

- 60 歳以上で糖尿病のある患者。
- 大豆アレルギーがあり、好きな色が赤である患者。

15.4 その他のサブジェクト領域オプションの指定

前述のオプション以外に、サブジェクト領域では以下の追加オプションを指定できます。

- **[無効]** ー (オプション) このチェック・ボックスにチェックを付けると、オーバーライドは無効になります。サブジェクト領域のリコンパイル時には、このオーバーライドは無視され、システムはキューブで指定された定義を使用します。
- **[表示名]** ー サブジェクト領域のローカライズ可能な名前。
- **[説明]** ー (オプション) サブジェクト領域のクラス定義に追加するコメント。クラス定義の開始時に、各行が個別のコメント行として保存されます。
- **[既定のメンバ]** ー (オプション) クエリが軸をスキップする際に使用する既定のメンバ。このサブジェクト領域でアクセス可能なメンバを返す MDX 式を指定します。メンバ式の詳細は、"InterSystems MDX リファレンス" を参照してください。これ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ンバが使用されます。
- **[デフォルトの詳細リスト]** ー (オプション) サブジェクト領域の既定のリストの論理名。このリストはキューブ、またはサブジェクト領域で定義する必要があります。これを指定しない場合は、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。
- **[所有者]** ー (オプション) キューブの所有者を指定します。InterSystems IRIS® データ・プラットフォーム・ユーザ名を指定します。
- **[カウント・メジャーのキャプション]** ー (オプション) Count メジャーの代替名を指定します。
- **[リソース]** ー (オプション) サブジェクト領域を保護するリソースを指定します。

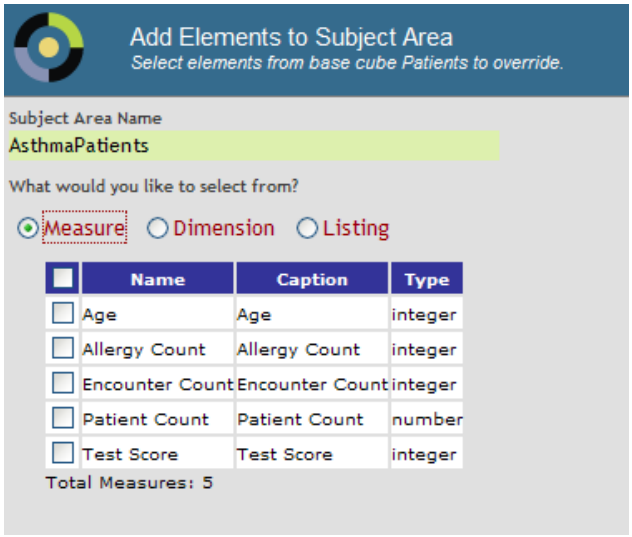
  この使用法については、"セキュリティの設定" を参照してください。
- **[キャプション]** ー (オプション) このキューブでの作業時にアナライザおよびその他のユーティリティに表示されるキャプションを指定します。
- **[ドメイン]** ー (オプション) サブジェクト領域のローカライズ文字列を含むドメインの名前を指定します。すべてのキューブおよびサブジェクト領域に単一のドメインを使用すると便利な場合があります。また、キューブおよびサブジェクト領域ごとに別のドメインを設定することが適切な場合もあります。"ローカライズの実行" を参照してください。

15.5 サブジェクト領域への項目の追加

サブジェクト領域にオーバーライドまたはリストを追加するには、以下の一般的な手順を使用します。

1. **[追加]** をクリックします。

   キューブの項目を選択できるダイアログ・ボックスが表示されます。



サブジェクト領域で項目に対するオーバーライドが既に定義されている場合、このダイアログ・ボックスではその項目にチェック・マークが表示されています。

2. **[メジャー]**、**[ディメンジョン]**、または **[リスト]** をクリックします。

   [メジャー] では、キューブで定義されている任意のメジャーをオーバーライドできます。

   [ディメンジョン] では、キューブで定義されている任意のディメンジョン、階層またはレベルをオーバーライドできます。

   [リスト] では、キューブで定義されている任意のリストをオーバーライドできます。リストを追加することもできます。

3. オーバーライドする項目をクリックしてから、[OK] をクリックします。

   項目が追加され、モデル・ビューワにこれらが表示されて、その詳細が詳細領域に表示されます。

4. 詳細領域で詳細を編集します。

15.6 メジャーのオーバーライドの定義

1 つ以上のメジャーにオーバーライドを定義する方法は以下のとおりです。

1. 必要に応じて、モデル・ビューワでメジャー名をクリックして、新しいオーバーライドを追加する場所を示します。

   この操作を行うと、クリックしたメジャーの前に新しいオーバーライドが追加されます。

2. **[追加]** をクリックします。
3. **[メジャー]** をクリックします。
4. オーバーライドするメジャーをクリックしてから、[OK] をクリックします。

   メジャーが追加され、モデル・ビューワにそれらが表示されます。

5. メジャーのオーバーライドを定義するには、モデル・ビューワでメジャー名をクリックし、詳細領域で以下のオプションを編集します。これらはすべてオプションです。

   - **[隠し]** – チェックを付けると、このサブジェクト領域内でこのメジャーが非表示になります。チェックを外すと、このサブジェクト領域でこのメジャーを使用できます。
   - **[表示名]** – キューブで定義された表示名を置換する、新しい表示名を指定します。

- **[説明]** – キューブで定義された説明を置換する、新しい説明を指定します。
- **[書式文字列]** – キューブで定義された書式文字列を置換する、新しい書式文字列を指定します。"書式文字列の指定" を参照してください。

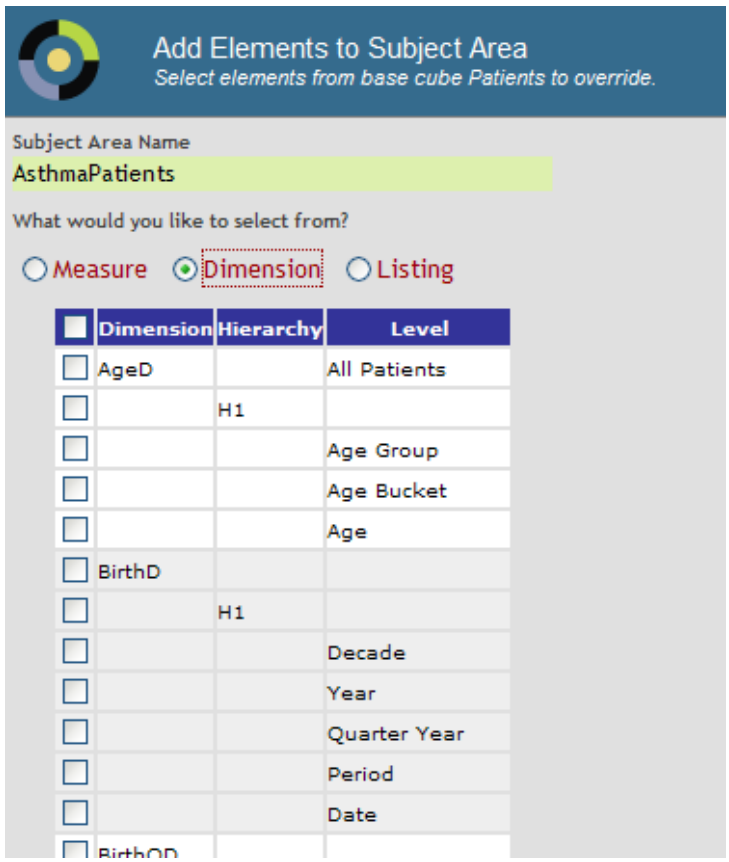
注釈　この方法では、計算メジャーにオーバーライドを定義することはできません。計算メジャーは、実際には計算メンバなので、オーバーライドを定義するにはスタジオを使用する必要があります。

15.7 ディメンジョン、階層またはレベルのオーバーライドの定義

1 つ以上のディメンジョン、階層またはレベルにオーバーライドを定義する方法は以下のとおりです。

1. **[追加]** をクリックします。
2. **[ディメンジョン]** をクリックします。

   キューブのディメンジョン、階層およびレベルが以下のように表示されます。



3. オーバーライドする項目をクリックしてから、[OK] をクリックします。

   項目が追加され、モデル・ビューワにそれらが表示されます。

   ディメンジョン、階層およびレベルの任意の組み合わせをクリックできます。ディメンジョンではなく、そこに含まれる階層をクリックした場合は、常にディメンジョンも追加されます (ディメンジョンは階層定義の一部であるため)。同様に、階層やディメンジョンではなく、そこに含まれるレベルをクリックした場合は、常に階層およびディメンジョンも追加されます。

4. 項目のオーバーライドを定義するには、モデル・ビューワで項目名をクリックし、詳細領域で以下のオプションを編集します。これらはすべてオプションです。
   - [隠し] ー チェックを付けると、このサブジェクト領域内でこの項目が非表示になります。チェックを外すと、このサブジェクト領域でこの項目を使用できます。
   - [表示名] ー キューブで定義された表示名を置換する、新しい表示名を指定します。
   - [説明] ー キューブで定義された説明を置換する、新しい説明を指定します。
   - [ソート] (レベルのみ) ー このレベルのメンバに対して別の並べ替え順序を指定します。

     注釈　特定のサブジェクト領域でレベルの並べ替え属性を変更した場合、その変更はキューブを再構築するまで有効になりません。

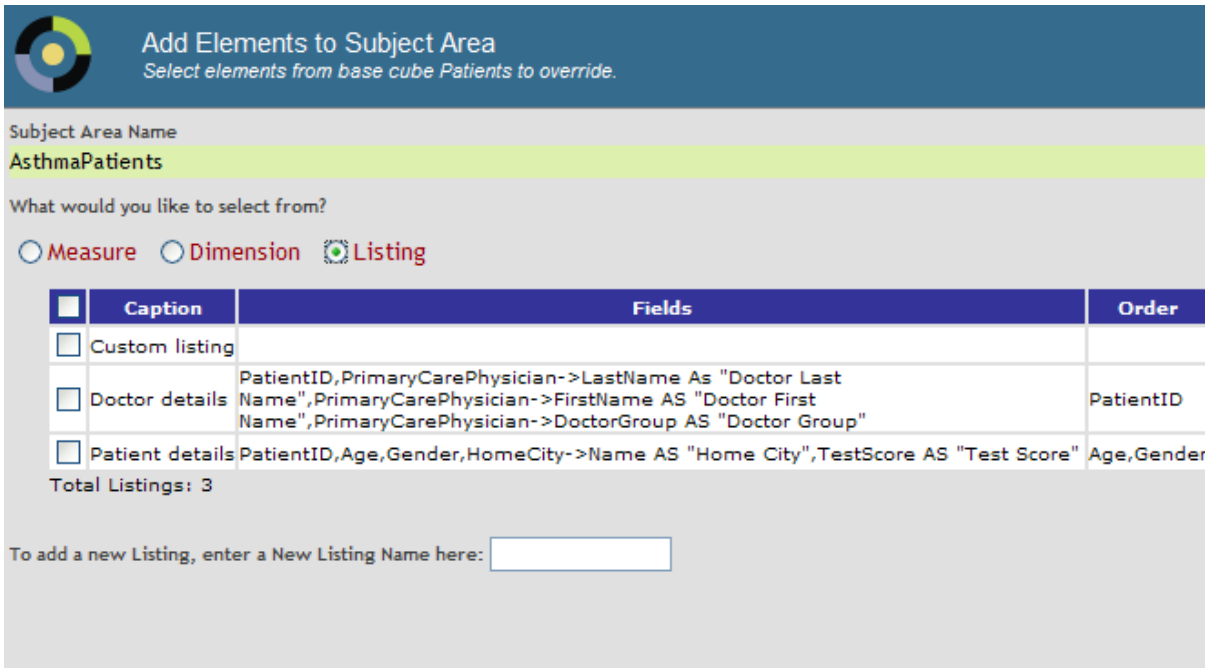
15.8 リストの再定義または新規リストの追加

サブジェクト領域で、リストの再定義、または新規リストの定義を行う手順は以下のとおりです。

1. 必要に応じて、モデル・ビューワでリスト名をクリックして、新しいオーバーライドを追加する場所を示します。

   この操作を行うと、クリックしたリストの前に新しいオーバーライドが追加されます。
2. [追加] をクリックします。
3. [リスト] をクリックします。

   キューブおよびサブジェクト領域のリストが以下のように表示されます。



4. 必要に応じて、再定義するリストをクリックします。
5. 必要に応じて、下部のボックスに新しいリストの名前を入力します。
6. [OK] をクリックします。

リスト（定義なし）が追加され、モデル・ビューワにそれらが表示されます。

7. リストを定義するには、モデル・ビューワでリスト名をクリックし、詳細領域でオプションを編集します。

   詳細は、"リストの定義" を参照してください。

15.9 サブジェクト領域のコンパイル

アーキテクトでサブジェクト領域クラスをコンパイルする手順は以下のとおりです。

1. **[コンパイル]** をクリックします。

   ダイアログ・ボックスが表示されます。

2. **[コンパイル]** をクリックします。

   クラスのコンパイルが開始され、その進捗状況が表示されます。

3. [OK] をクリックします。

または、スタジオでサブジェクト領域クラスを開き、他のクラスのコンパイルと同じ方法でこれをコンパイルします。

Tipヒント　サブジェクト領域クラスをコンパイルする前に、依存しているキューブ・クラスをコンパイルする必要があります。スタジオを使用してサブジェクト領域クラスを編集し、DependsOn キーワードを追加して、それらのクラスが正しい順序でコンパイルされるように強制すると便利です。ただし、必ずそうする必要があるわけではありません。"サブジェクト領域クラスのリファレンス情報" の "サブジェクト領域クラスの要件" を参照してください。

16
リスト・グループの定義

ここでは、リスト・グループ・マネージャの使用法を説明します。リスト・グループ・マネージャを使用すると、どの Business Intelligence キューブ定義にも含まれていないリストを定義できます。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

16.1 リスト・グループの概要

リスト・グループ とは、複数のリストを定義する特別な種類のクラスです。これらのリストは、キューブ定義の外部で定義されるという意味では補足的なものです。これらのリストは、InterSystems IRIS Business Intelligence でサポートされているどの種類のリストでもかまいません。ユーザにリスト・グループ・マネージャへのアクセスを許可することで、それらのユーザはキューブ定義を変更することなく追加のリストを定義できるようになります。

リスト・グループは便利なコンテナとして用意されています。リスト・グループでは、そのグループのすべてのリストに該当する以下の情報も指定します。

- それらのリストを使用できるキューブ
- 各リストを保護するリソース (既定で)

リスト・グループは、別の場所で定義されているリストをオーバーライドすることはできません。

リスト・グループは有効化または無効化されます。リスト・グループが無効化されている場合は、そのグループのリストはどのキューブでも使用できません。リスト・グループが有効化されていて、コンパイル済みである場合は、そのグループのリストは指定されたキューブで使用できます。

重要　　リスト・グループ内のリストは、そのリスト・グループがコンパイルされるまで使用できません。

リスト・グループにリソースを関連付けることで、そのリソースにアクセスできるユーザのみがそのリスト・グループを編集可能になります。

アナライザとダッシュボード・デザイナーでは、リスト・グループ自体が表示される代わりに、単にすべての使用可能なリストが表示されます。エンド・ユーザには、それぞれのリストがどこでどのように定義されているのかはわかりません。

16.2 リスト・グループ・マネージャへのアクセス

リスト・グループ・マネージャを使用すると、リスト・グループおよびグループ内のリストを定義できます。リスト・グループ・マネージャにアクセスする手順は以下のとおりです。

1. [InterSystems ランチャー] をクリックし、**[管理ポータル]** をクリックします。

   セキュリティの設定によっては、InterSystems IRIS® データ・プラットフォームのユーザ名とパスワードを使用してログインするように求められます。

2. 以下のように、適切なネームスペースに切り替えます。

   a. 現在のネームスペースの名前をクリックして、使用可能なネームスペースのリストを表示します。

   b. リストから、該当するネームスペースをクリックします。

3. **[Analytics]→[ツール]→[リスト・グループ・マネージャ]** をクリックします。

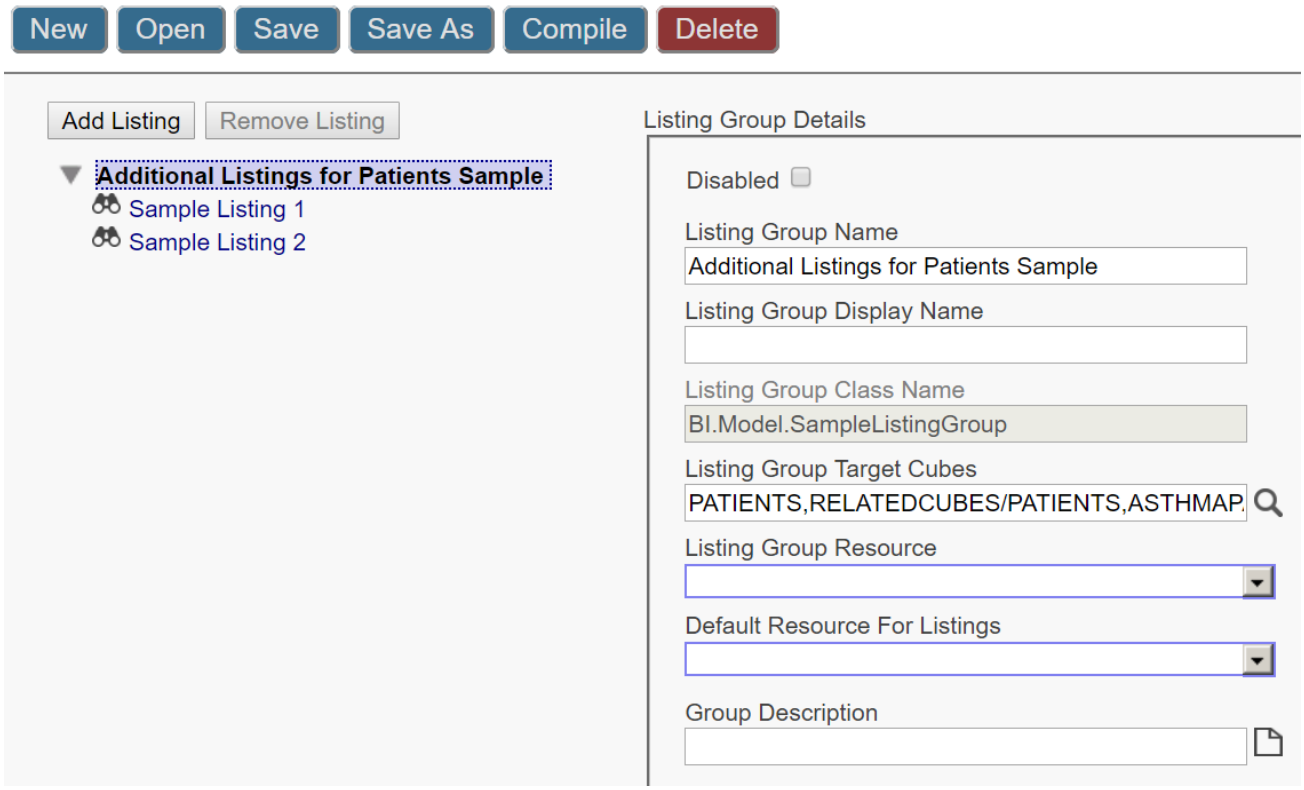
   このネームスペース内にリスト・グループがある場合は、リスト・グループ・マネージャにはユーザが最後に表示したリスト・グループが表示されます。

   このネームスペース内にリスト・グループがない場合は、リスト・グループ・マネージャにはリスト・グループの作成を求めるプロンプトが表示されます。以下の詳細情報を入力してから、**[OK]** をクリックします。

   - **[リスト・グループ名]** – このリスト・グループの論理名。これは後で編集できます。
   - **[リスト・グループのクラス名]** – このリスト・グループのクラス名 (パッケージ名を含む)。
   - **[リスト・グループの説明]** – このリスト・グループの説明 (オプション)。これは後で編集できます。

   または、ダイアログ・ボックスの右上にある [X] をクリックします。

リスト・グループ・マネージャには、下図のようなリスト・グループが表示されます。



左側の領域には、リスト・グループの論理名 (この例では `Sample Listing Group`) とそのリスト・グループ内のリストの名前が表示されます。この表示領域は、折りたたんだり展開したりできます。

右側の領域には、左側で選択した項目 (リスト・グループまたはそのグループで定義されているいずれかのリスト) の詳細情報が表示されます。

16.3 リスト・グループの追加

リスト・グループ・マネージャでリスト・グループを追加する手順は以下のとおりです。

1. [新規作成] をクリックします。

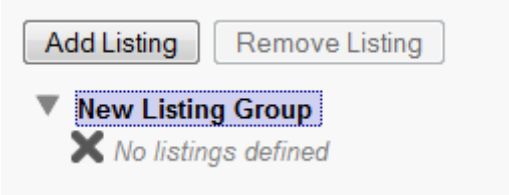
   ダイアログ・ボックスが表示されます。

2. 以下の詳細情報を入力します。

   - [リスト・グループ名] – このリスト・グループの論理名。これは後で編集できます。
   - [リスト・グループのクラス名] – このリスト・グループのクラス名 (パッケージ名を含む)。
   - [リスト・グループの説明] – このリスト・グループの説明 (オプション)。これは後で編集できます。

3. [OK] を押します。

リスト・グループ・マネージャによってクラスが作成および保存されて、新しいリスト・グループが表示されます。左側の領域には、以下のように表示されます。



これで、次のサブセクションの説明に従ってこのリスト・グループの詳細情報を変更できます (この変更はこのグループ内のすべてのリストに適用されます)。または、このページで後述する説明に従ってリストを追加することもできます。

16.3.1 リスト・グループの詳細情報の変更

リスト・グループ・マネージャの左側の領域でリスト・グループを選択すると (グループ内のリストを選択するのではなく)、右側の領域に以下の情報が表示され、これらの情報を変更できます。

- [無効] – 必要に応じて、これをクリックしてリスト・グループを無効にします。これにより、このリスト・グループ内のリストをどのキューブでも使用できなくなります。
- [リスト・グループ名] – このリスト・グループの論理名を指定します。
- [リスト・グループの表示名] – 必要に応じて、このリスト・グループの表示名を指定します。
- [リスト・グループのターゲット・キューブ] – このリスト・グループで定義されたリストを使用可能にするキューブとサブジェクト領域を選択します。メモ :
  - キューブを選択すると、これらのリストはそのキューブで使用可能になると共に、そのキューブに基づいたすべてのサブジェクト領域でも使用可能になります。
  - サブジェクト領域を選択して、そのサブジェクト領域が基づいているキューブを選択しない場合は、これらのリストはそのサブジェクト領域のみで使用可能になります。

  これらのリストを使用できるのは、このリスト・グループが有効化およびコンパイルされている場合のみです。
- [リスト・グループのリソース] – 必要に応じて、このリスト・グループを保護するリソースを選択します。このリソースにアクセスできるユーザのみがこのリスト・グループを編集できます。

  リソースの使用法については、"セキュリティの設定" を参照してください。

- **[リストの既定のリソース]** – 必要に応じて、このリスト・グループ内の各リストを保護する既定のリソースを選択します。
- **[グループの説明]** – 必要に応じて、このリスト・グループの説明を入力します。

変更を加えた後に、**[保存]** または **[名前を付けて保存]** をクリックします。

[名前を付けて保存] をクリックした場合は、新しい論理名、新しいクラス名、およびオプションの新しい説明を入力するためのプロンプトが表示されます。

これらの変更内容がユーザに表示されるようにするには、このリスト・グループをコンパイルする必要もあります。

16.4 リスト・グループの表示

リスト・グループ・マネージャでリスト・グループを表示する手順は以下のとおりです。

1. **[開く]** をクリックします。
2. リスト・グループの名前をクリックします。
3. [OK] をクリックします。

これで、前のセクションの説明に従ってこのリスト・グループの詳細情報を変更できます (この変更はこのグループ内のすべてのリストに適用されます)。または、このページで後述する説明に従ってリストを追加、変更、または削除することもできます。

16.5 リスト・グループへのリストの追加

リスト・グループ・マネージャでリスト・グループにリストを追加する手順は以下のとおりです。

1. 目的のリスト・グループを表示します。
2. **[リストの追加]** をクリックします。

   これにより、新しいリスト (詳細情報なし) が追加されて、そのリスト・グループがこの時点で自動的に保存されます。左側の領域には、新しいリストの名前が表示されます。このリストには `New Listing` や `New Listing1` などの名前が付けられます。
3. 新たに追加したリストの名前をクリックします。
4. 右側の領域に表示される名前などの詳細情報を変更します。これらの詳細情報については、"リストの定義" を参照してください。

   ターゲット・キューブ内に既に存在するリスト名を指定することはできません。リスト・グループによって、別の場所で定義されているリストをオーバーライドすることはできないからです。
5. **[保存]** をクリックしてリスト・グループを保存します。または **[名前を付けて保存]** をクリックして新しいリスト・グループを作成します。
6. これらの変更内容がユーザに表示されるようにするには、このリスト・グループをコンパイルする必要もあります。

16.6 リストの変更

リスト・グループ・マネージャでリストを変更する手順は以下のとおりです。

1. 目的のリストが含まれているリスト・グループを表示します。
2. 左側の領域で、そのリストの名前をクリックします。
3. 右側の領域に表示される名前などの詳細情報を変更します。これらの詳細情報については、"リストの定義" を参照してください。
4. **[保存]** をクリックしてリスト・グループを保存します。または **[名前を付けて保存]** をクリックしてリスト・グループを新しい名前で保存します。
5. これらの変更内容がユーザに表示されるようにするには、このリスト・グループをコンパイルする必要もあります。

16.7 リスト・グループからのリストの削除

リスト・グループ・マネージャでリストを削除する手順は以下のとおりです。

1. 目的のリストが含まれているリスト・グループを表示します。
2. 左側の領域で、そのリストの名前をクリックします。
3. **[リストの削除]** をクリックして、[OK] をクリックします。
4. これらの変更内容がユーザに表示されるようにするには、このリスト・グループをコンパイルする必要もあります。

16.8 リスト・グループのコンパイル

リスト・グループ・マネージャでリスト・グループをコンパイルする手順は以下のとおりです。

1. 目的のリスト・グループを表示します。
2. **[無効]** のチェックが外れている場合は、**[リスト・グループのターゲット・キューブ]** で 1 つ以上のキューブが指定されていることを確認します。
3. **[コンパイル]** をクリックします。

該当するクラスが保存（必要に応じて）およびコンパイルされて、そのクラスがコンパイルされたことを示すメッセージが表示されます。

リスト・グループがコンパイルされると、そのリスト・グループ内のリストがターゲットのキューブおよびサブジェクト領域で使用可能になります。詳細は、"リスト・グループの詳細情報の変更" を参照してください。

16.8.1 考えられるコンパイル・エラー

リスト・グループ・マネージャでは、以下の場合にコンパイル・エラーが発行されます。

- ターゲット・キューブが指定されていない場合。この場合は、**[リスト・グループのターゲット・キューブ]** で 1 つ以上のキューブが指定されていることを確認してから、リコンパイルします。

- このリスト・グループで定義されているリストの名前が、ターゲットのキューブまたはサブジェクト領域内の既存のリストと同じ名前である場合。この場合は、リスト・グループ内の該当するリストの名前を変更してから、リコンパイルします。(または、リスト・グループ内の該当するリストを無効にしてから、リコンパイルします。)
- ターゲットのキューブまたはサブジェクト領域で `disableListingGroups="true"` が指定されている場合。コンパイル・エラーには、`Target cube does not accept listing groups:` というテキストが含まれます。この場合は、そのターゲット・キューブを削除して、他のすべてのキューブを残して、リコンパイルします。または、リスト・グループを受け付けることができるようにようにそのキューブまたはサブジェクト領域を変更してから、そのキューブまたはサブジェクト領域をリコンパイルします。

  `disableListingGroups` の詳細は、"キューブ・クラスのリファレンス情報" および "サブジェクト領域クラスのリファレンス情報" を参照してください。このオプションをアーキテクトで指定することはできません。

16.9 リスト・グループの削除

リスト・グループ・マネージャでリスト・グループを削除する手順は以下のとおりです。

1. 目的のリスト・グループを表示します。
2. **[削除]** をクリックします。
3. [OK] をクリックし、この操作を確定します。

該当するクラスが削除されて、そのことを示すメッセージが表示されます。

このリスト・グループ内のリストは使用できなくなります。

キューブ・クラスのリファレンス情報

アーキテクトは、Business Intelligence キューブ・クラスの作成と変更を行います。キューブ・クラスはスタジオで直接、作成および編集することもできます。このリファレンスでは、以下のクラスの詳細を提供します。

"リコンパイルおよび再構築のタイミング" および "キューブおよびサブジェクト領域の高度な機能の使用" も参照してください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

キューブ・クラスの要件

Business Intelligence キューブ・クラスの基本情報を提供します。

詳細

キューブを定義するには、以下の要件に合致したクラスを作成します。

- %DeepSee.CubeDefinition を拡張する必要があります。
- Cube という名前の XData ブロックを格納している必要があります。
- この XData ブロックに対して、以下のように XMLNamespace が指定されている必要があります。

  ```
  XMLNamespace = "http://www.intersystems.com/deepsee"
  ```

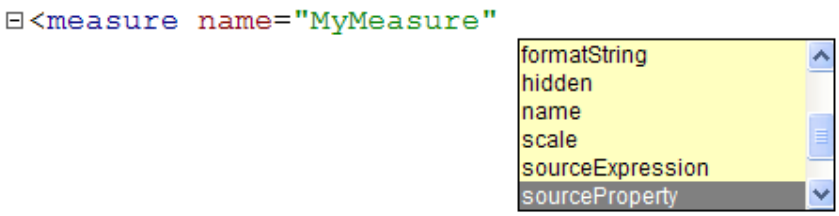
- XData ブロック内のルート要素が <cube> であり、この要素がこのページの他の部分に記載された要件に従っている必要があります。
- クラスで DependsOn コンパイラ・キーワードを指定することによって、キューブのソース・クラスがコンパイルされて使用できるようになった後にのみ、このクラスがコンパイルされるようにすると便利です。ただし、必ずしもそうする必要はありません。
- クラスでは DOMAIN パラメータを定義できます。これによってローカライズされた文字列が属するドメインが指定されます。以下はその例です。

  Class Member

  ```
  Parameter DOMAIN = "PATIENTSAMPLE";
  ```

  詳細は、"ローカライズの実行" を参照してください。

スタジオでは、入力に応じたサポートが提供されます。以下はその例です。



注釈　最後に、特定のインターシステムズ製品では、HealthShare 製品と共に提供されている HSCUSTOM ネームスペースなど、カスタマイズされたクラス向けのすぐに使用可能なネームスペースが用意されています。このようなネームスペースにはキューブを導入しないことを強くお勧めします。

例

```
Class BI.Model.PatientsCube Extends %DeepSee.CubeDefinition [DependsOn=BI.Study.Patient]
{

XData Cube [ XMLNamespace = "http://www.intersystems.com/deepsee" ]
{
<cube
name="Patients"
 owner="_SYSTEM"
 caption="Patients"
 sourceClass="BI.Study.Patient"
 other_cube_options...
>

<measure measure_definition/>
```

```
...
<dimension dimension_definition/>
...
</cube>
}
```

キューブの共通属性

Business Intelligence キューブ定義全体で使用される属性をリストします。

詳細

キューブ内の要素の大部分には以下の属性があります。わかりやすくするため、ここにこれらをリストします。

属性	目的
name	このキューブに対する MDX クエリで使用される要素の論理名。"モデル要素の名前" を参照してください。
displayName	(オプション) ユーザ・インタフェースで使用されるこの要素のローカライズ名。この属性を指定しない場合、代わりに name 属性で指定された値が表示されます。詳細は、"ローカライズの実行" を参照してください。
description	(オプション) この要素の説明。
disabled	(オプション) コンパイラがこの要素を使用するかどうかを制御します。この属性が "true" の場合、コンパイラは要素を無視します。これは要素をコメントアウトすることと同等です。既定では、この属性は "false" です。
additionalDescription	(オプション) この要素に関する追加の注意事項。アーキテクトとスタジオでのみ表示されます。

<cube>

Business Intelligence キューブを定義します。

<cube>

<cube> 要素には以下のコンテンツがあります。

属性または要素	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
sourceClass	このキューブによって使用されるベース・クラスの完全なパッケージおよびクラス名。"キューブに使用可能なソース・クラス" を参照してください。
owner	(オプション) キューブの所有者の名前。
resource	(オプション) アーキテクトを介してアクセスする際に、このキューブへのアクセスの制御に使用されるリソースの名前。"InterSystems Business Intelligence の実装" を参照してください。
caption	(オプション) このキューブでの作業時にアナライザおよびその他のユーティリティに表示するキャプション。
countMeasureCaption	(オプション) 既定のメジャーに使用するキャプション。レコード数をカウントします。既定のキャプションは Count です。内部的には、このメジャーの名前は %Count です。
countMeasureName	(オプション) 既定のメジャーに使用する名前。レコード数をカウントします。このメジャーの既定の名前は %Count です。
defaultListing	(オプション) このキューブの既定として使用する <listing> の論理名を指定します。このページで後述の "<listing>" を参照してください。
nullReplacement	(オプション) レベルのソース・データが NULL の場合にメンバ名として使用する文字列を指定します。例えば、nullReplacement="None" と指定します。 指定の <level> に nullReplacement 属性が指定されている場合は、この属性によってこのオプションがオーバーライドされます。
bucketSize	(オプション) このキューブに使用されるキャッシュ・バケットのサイズを指定します。連続する 64,000 行が 1 つのグループであり、バケットはグループの整数倍です。既定のバケット・サイズは 8 です。つまり、既定では、最初の 512,000 行のバケット番号は 1、次の 512,000 行はバケット番号 2、というようになります。 状況に応じて、多数のファクトがあり、キューブ内の古い値の更新が多くないと予測される場合は、この値を増やすことができます。この値を変更する場合は、必ずそのキューブに対してキャッシュされたすべての結果を削除してください。そのためには、キューブ定義クラスの %KillCache() メソッドを呼び出します。
actionClass	(オプション) このキューブに基づくピボット・テーブルで使用可能なアクションを定義する、関連 KPI クラスを指定します。%DeepSee.KPI のサブクラスの完全なパッケージとクラス名を指定します。

属性または要素	目的
maxFacts	(オプション) ファクト・テーブルの最大行数を指定します。これによって、キューブの構築時に使用されるベース・テーブルの行数が決まります。この属性は、ベース・テーブル全体の処理が不要な場合にデバッグを支援するために使用されます。既定では、ベース・テーブルのすべての行が処理されます。
bitmapChunkInMemory	(オプション) システムによるインデックスの構築方法として、プロセス・プライベート・グローバル (低速、ただし <STORE> エラーが発生しない) を使用するのか、またはすべてメモリ内 (高速) とするのかを制御します。既定は "false" で、システムはインデックス構築時の一時的なストレージにプロセス・プライベート・グローバルを使用します。 キューブに大量のインデックス (ディメンジョンおよびメジャー) がなく、非常に大量のメンバが含まれるディメンジョンがない場合は、"true" を使用してパフォーマンスを高速化できます。この設定は試してみる価値があります。キューブの構築時に <STORE> エラーが発生した場合には、値を既定に戻します。
initialBuildOrder	(オプション) キューブ全体を構築する際に使用するオプションの ORDER BY 節の構築に使用されます。インクリメンタル更新には影響しません。ソース・テーブルのフィールドのコンマ区切りリストを指定します。SQL キーワード ASC および DESC を使用できます。例 : "Age DESC,Gender"
buildRestriction	(オプション) キューブの構築および更新時に使用するオプションの WHERE 節を指定します。これにより、キューブがレコードのサブセットを使用するようになります。ソース・テーブルのフィールドを使用する SQL 比較式を指定します。例: Gender='F'。キューブがデータ・コネクタに基づいている場合、このオプションには何の効果もありません。
abstract inheritsFrom	(オプション) "キューブ継承による再使用可能要素の定義" を参照してください。
disableListingGroups	(オプション) この属性に "true" を指定すると、すべてのユーザはこのキューブをターゲットとして使用するリスト・グループを定義できなくなります。既定は "false" です。"リスト・グループのコンパイル" を参照してください。
version	(オプション) "キューブ・バージョンの使用" を参照してください。
namedFactNums	(オプション) この属性が "true" の場合、キューブの各メジャー、レベル、リレーションシップにはファクト番号を定義する必要があります。この属性を "true" に設定することは、選択的構築を有効にすることと同じです。
defaultMeasure、defaultMember、precompute	使用しません。
<measure>	(オプション) ゼロ個以上の <measure> 要素を含むことができます。このそれぞれがメジャーを定義します。
<dimension>	(オプション) ゼロ個以上の <dimension> 要素を含むことができます。このそれぞれがディメンジョンを定義します。
<listing>	(オプション) ゼロ個以上の <listing> 要素を含むことができます。このそれぞれがリストを定義し、アナライザで使用できます。
<listingField>	(オプション) ゼロ個以上の <listingField> 要素を含むことができます。このそれぞれがリストを定義し、アナライザで使用できます。

属性または要素	目的
<namedSet>	(オプション) ゼロ個以上の <namedSet> 要素を含むことができます。このそれぞれが名前付きセットを定義します。名前付きセットは、メンバまたはメンバのセットのエイリアスです。
<calculatedMember>	(オプション) ゼロ個以上の <calculatedMember> 要素を含むことができます。このそれぞれが、他のメンバに関連して計算されるメンバを定義します。通常、これは他のメジャーに基づいて計算されるメジャーの定義に使用されます。
<relationship>	(オプション) ゼロ個以上の <relationship> 要素を含むことができます。<relationship> 要素は、キューブに別のキューブを接続し、独自のキューブのクエリを実行する際に、その別のキューブのレベルを使用できるようにします。
<index>	(オプション) ゼロ個以上の <index> 要素を含むことができます。このそれぞれがファクト・テーブル上でオプションのカスタム・インデックスを定義します。これは、そのファクト・テーブルに対する SQL クエリを実行する計画がある場合に役立ちます。ファクト・テーブルの詳細は、"ファクト・テーブルおよびディメンジョン・テーブルの詳細" を参照してください。
<expression>	(オプション) ゼロ個以上の <expression> 要素を含むことができます。このそれぞれが、1 つ以上のメジャーまたはディメンジョンの定義で使用される式 (通常は中間値) を定義します。式の値は格納されませんが、キューブの構築時に使用できます。

例

```
<cube
name="Patients"
 owner="_SYSTEM"
 caption="Patients"
 sourceClass="BI.Study.Patient"
 >
...
```

〈measure〉

Business Intelligence キューブでメジャーを定義します。

詳細

<measure> 要素には以下の属性があります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
sourceProperty、sourceExpression	アーキテクトで [プロパティ] または [式] を指定する際とほとんど同じ方法で、これらの属性のいずれかを指定します。"ディメンジョンまたはレベルのソース値の定義" と "ソース式の詳細" を参照してください。メモ： ・ sourceProperty には、[プロパティ] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceProperty='Age' ・ sourceExpression には、[式] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceExpression='%source.property/100' 値自体に二重引用符が含まれない場合は、代わりに二重引用符で値を囲むこともできます。例：sourceExpression="%source.property/100"
type	(オプション) メジャーのデータ型を指定します。ほとんどのメジャーは数値です。これが既定です。"メジャーのデータ型の指定" を参照してください。
iKnowSource、iKnowDomain、iKnowDictionaries、iKnowParameters	(オプション) "キューブでの Text Analytics の使用法" を参照してください。
aggregate	(オプション) 複数のレコードを結合する際、常に、このメジャーの値の集約方法を指定します。このオプションを指定する場合は、以下の値のいずれかを使用します。 ・ "SUM" ・ "COUNT" — ソース・データが 非 NULL (および非ゼロ) 値のレコード数をカウントします。 ・ "MAX" — セット内の最大値を使用します。 ・ "MIN" — セット内の最小値を使用します。 ・ "AVG" — セットの平均値を計算します。 ブーリアンまたは文字列のメジャーでは、この属性は "COUNT" である必要があります。
hidden	(オプション) hidden="true" の場合、このメジャーは定義され、クエリで使用することができますが、アナライザで使用可能なメジャーとしてリストには表示されません。これにより、中間計算の機能を果たすメジャーを定義できます。

属性	目的
scale	(オプション) 保持する小数点以下桁数を指定します。既定では、scale は 0 で、メジャー値のそれぞれがファクト・テーブルに書き込まれる前に整数に丸められます。
linkClass、linkProperty	(オプション) "他のテーブルへのリンク" を参照してください。
searchable	(オプション) searchable="true" の場合、このメジャーは、メジャー値を参照するフィルタ式を作成 (通常はメジャー値と定数と比較するため) できる高度なフィルタのオプションとして表示されます。 これらの式をサポートするため、検索可能メジャーには必要に応じてさらにインデックスが追加されます。"メジャー検索式" を参照してください。検索可能なメジャーの名前には、角括弧やコンマ (],) を含めることはできません。 searchable="true" とマークされていない場合でも、手動 MDX クエリでメジャーを検索可能メジャーとして使用できる場合もあることに注意してください。 いずれの場合も、このメジャーは他のメジャーと同様の方法で使用できます。
factName	(オプション) 生成されたファクト・テーブルにおける、このメジャーに対応する列の名前。この属性が NULL の場合はシステムによって名前が生成されます。このオプションは、NLP メジャーには適用されません。 SQL の予約語は使用しないでください。SQL 予約語のリストは、"予約語" を参照してください。名前は、英字かパーセント記号 (%) で始まる必要があります。最初の文字が % である場合、2 番目の文字は Z または z である必要があります。制限事項の詳細は、"クラス・メンバ" を参照してください。また、fact または listing という語は、どのような大文字と小文字の組み合わせであっても使用しないでください。
factNumber	このメジャーに割り当てられている内部 ID。キューブの namedFactNums が "true" の場合は必須です。
factSelectivity	(オプション、システムでは不使用) ファクト・テーブル・クラス内のプロパティの生成 SELECTIVITY をオーバーライドする値。Business Intelligence のクエリでは、このパラメータは使用しません。このオプションは、生成されたファクト・テーブルに対して直接 SQL を使用するときに、生成された SELECTIVITY をオーバーライドする必要がある場合に使用します。 1 以下の正の値を指定します。詳細は、%SYSTEM.SQL の SetFieldSelectivity() を参照してください。
formatString	(オプション) 値の表示方法を制御します。"formatString の詳細" を参照してください。
units	(オプション) メジャー値が表示される単位を示します。現在、この属性は、一般情報としてのみ提供されています。
listingFilterOperator および listingFilterValue	(オプション) これらの属性は、ユーザがピボット・テーブルでこのメジャーを選択し、リストを表示する際に適用されるオプションの追加フィルタを指定します。詳細は、"メジャーの定義" の "リストの追加フィルタの指定" を参照してください。

例

以下はその例です。

```
<measure name="Test Score" sourceProperty="TestScore" aggregate="SUM"/>
<measure name="Avg Test Score" sourceProperty="TestScore" aggregate="AVG"/>
<measure name="Allergy Count"
   sourceExpression="##class(Cubes.StudyPatients).GetAllergyCount(%source.%ID)"/>
```

Allergy Count メジャーが、キューブ・クラス内で定義されたユーティリティ・メソッドを使用していることに注意してください。

%COUNT メジャー

システムには、クエリ内のファクト数を返す %COUNT という名前の事前定義メジャーがあります。

Measures ディメンジョン

Measures ディメンジョンは、システムによって自動的に生成され、すべてのメジャーがこの中に配置されます。

formatString の詳細

formatString 属性は、以下のような分割化された文字列です。

```
Format^Color
```

- Format 部分には、以下のような 1 から 4 の部分文字列で構成される文字列を指定します。

  ```
  positive_format;negative_format;zero_format;missing_format;
  ```

  positive_format は、正の値の表示方法を制御し、negative_format は、負の値の表示方法を制御します。zero_format は、ゼロの表示方法を制御し、missing_format は、欠落値の表示方法を制御します。

  詳細は、"書式文字列の指定" を参照してください。

- Color 部分には、以下のような 1 から 4 の部分文字列で構成される文字列を指定します。

  ```
  positive_color;negative_color;zero_color;missing_color;
  ```

  positive_color は、正の値の色を制御し、negative_color は、負の値の色を制御します。zero_color は、ゼロの色を制御し、missing_color は、欠落値の色を制御します。

  それぞれに対して、CSS カラー名または 16 進カラー・コードを指定します。

日付メジャーの場合、以下のバリエーションを使用します。

```
%date%^Color
```

Color は前述の説明と同じです。%date% は、現在のプロセスの既定の日付形式に従って、日付をフォーマットします。

〈dimension〉

Business Intelligence キューブでディメンジョンを定義します。

詳細

<dimension> 要素には以下のコンテンツがあります。

属性または要素	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
sourceProperty、sourceExpression	アーキテクトで [プロパティ] または [式] を指定する際とほとんど同じ方法で、これらの属性のいずれかを指定します。"ディメンジョンまたはレベルのソース値の定義" を参照してください。メモ： ・ sourceProperty には、[プロパティ] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceProperty='Age' ・ sourceExpression には、[式] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceExpression='%source.property_"ABC"' 値自体に二重引用符が含まれない場合は、代わりに二重引用符で値を囲むこともできます。例：sourceExpression="%source.property"
type	以下のいずれかを指定します。 ・ "data" (既定) ― 通常のディメンジョンを作成します。 ・ "time" ― 時刻ディメンジョンを作成します。"時間レベルの定義" を参照してください。 ・ "age" ― 年齢ディメンジョンを作成します。"年齢レベルの定義" を参照してください。 ・ "computed" ― 計算ディメンジョンを作成します。"計算ディメンジョンの定義" を参照してください。 ・ "iKnow" ― NLP により処理されるテキストを使用するディメンジョンを作成します。"キューブでの Text Analytics の使用法" を参照してください。
calendar	(オプション) これは type が "time" である場合にのみ指定します。calendar 属性では、時間レベルのメンバにレコードを割り当てる際に使用する暦を指定します。グレゴリオ暦を使用する場合は、"gregorian" (既定値) を指定します。ヒジュラ暦を使用する場合は、"hijriTabular" または "hijriObserved" を指定します。
iKnowMeasure および iKnowType	(オプション) "キューブでの Text Analytics の使用法" を参照してください。
hasAll	(オプション) このディメンジョンに All レベルがあるかどうかを示します。既定は "true" です。

属性または要素	目的
allCaption	(オプション) このディメンジョンの All レベルまたは All メンバに使用される名前。既定の名前は、All dimension です。dimension はディメンジョン名です。
allDisplayName	(オプション) All メンバに使用されるローカライズ名を指定します。この属性を指定しない場合、代わりに allCaption 属性で指定された値が表示されます。
hidden	(オプション) hidden="true" の場合、ディメンジョンは定義され、クエリで使用することができますが、アナライザはディメンジョンを使用可能なものとして表示しません。既定は "false" です。
showHierarchies	(オプション) アナライザにこのディメンジョン内の階層名が表示されるかどうかを制御します。以下のいずれかを指定します。 ・ "default" ― 複数の階層が存在する場合にのみ、階層名を表示します。 ・ "true" ― 階層名を常に表示します。 ・ "false" ― 階層名を表示しません。 この属性はクエリ自体には影響しません。
sharesFrom	(オプション) "形式的に共有されるディメンジョンの定義" を参照してください。このオプションを日付ディメンジョンには指定しないでください。日付ディメンジョンは自動的に共有されます。
dimensionClass	(オプション) "計算ディメンジョンの定義" を参照してください。
<hierarchy>	このディメンジョン内の階層を指定します。1 つ以上の <hierarchy> 要素を組み込む必要があります。また、複数の <hierarchy> 要素を組み込むことができます。

<hierarchy>

Business Intelligence キューブで、指定されたディメンジョン内に階層を指定します。

詳細

<hierarchy> 要素には以下のコンテンツがあります。

属性または要素	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
hidden	(オプション) hidden="true" の場合、階層は定義され、クエリで使用することができますが、アナライザは階層およびそのレベルを使用可能なものとして表示しません。既定は "false" です。
<level>	この階層内のレベルを指定します。1 つ以上の <level> 要素を組み込む必要があります。また、複数の <level> 要素を組み込むことができます。

<level>

Business Intelligence キューブで、指定された階層内にレベルを指定します。

詳細

<level> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
sourceProperty、sourceExpression	これらの属性のどちらかを指定します。構文は、アーキテクトで [プロパティ] または [式] に使用する構文とほとんど同じです。"ディメンジョンまたはレベルのソース値の定義" を参照してください。メモ： ・ sourceProperty には、[プロパティ] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceProperty='Age' ・ sourceExpression には、[式] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceExpression='%source.property_"ABC"' 値自体に二重引用符が含まれない場合は、代わりに二重引用符で値を囲むこともできます。例：sourceExpression="%source.property" 時刻ディメンジョンまたは年齢ディメンジョン内のレベルの要件については、"時間レベルの定義" および "年齢レベルの定義" を参照してください。 NLP ディメンジョンのレベルの場合、ソース値の指定に使用するメカニズムが異なるため、sourceProperty と sourceExpression は無視されます。"キューブでの Text Analytics の使用法" を参照してください。
timeFunction	時刻ディメンジョンまたは年齢ディメンジョン内のレベルでのみ使用されます。"時間レベルの定義" および "年齢レベルの定義" を参照してください。
timeOffset	(オプション) 時刻ディメンジョンのレベルでのみ使用されます。"日付オフセットの指定" を参照してください。
timeFormat	(オプション) 時刻ディメンジョンのレベルでのみ使用されます。"レベルの定義の詳細" の "時刻形式の指定" を参照してください。
list	(オプション) この属性が "true" の場合、ソース値はリストであることが期待され、リスト内の各項目がこのレベルのメンバになります。 既定は "false" です。 既定の場合、リストは $LIST 形式であることが期待されます。文字区切りリストで構成される文字列を使用するには、listDelimiter を指定します。 リスト・ベース・レベルは、親レベルおよび子レベルを持つことはできません。
listDelimiter	(オプション) レベルのソース・データとして使用されるリストの項目の区切りに使用される区切り文字を指定します。これは、リストが文字区切りリストである場合に使用します。

属性	目的
nullReplacement	(オプション) このレベルのソース・データが NULL の場合に、メンバ名として使用する文字列を指定します。例えば、nullReplacement="No City" と指定します。
rangeExpression	(オプション) 数値データの場合は、ビンに数値を割り当てる方法を指定します (各ビンはこのレベルのメンバです)。その他のデータの場合は、この属性によって置換値が指定されます。この属性は、このレベルのメンバの既定の順序も制御します。サブセクション "基本的な範囲式の定義" と "複数範囲式の圧縮の定義" を参照してください。
useDisplayValue	(オプション) DISPLAYLIST および VALUELIST パラメータの値があるプロパティの場合、この属性によってインデックスに組み込む値が指定されます。この属性が "false" (既定) の場合は、VALUELIST で指定された値が使用されます。この属性が "true" の場合は、DISPLAYLIST で指定された値が使用されます。 linkClass および linkProperty を指定した場合、このオプションは無視されます。
sort	(オプション) このレベルのメンバの既定の並べ替え方法を指定します。時刻ディメンジョンのレベルの場合、"asc" または "desc" を指定します。 データ・ディメンジョンのレベルの場合、"asc"、"asc numeric"、"desc"、または "desc numeric" を指定します。
linkClass、linkProperty	(オプション) "他のテーブルへのリンク" を参照してください。
factName	(オプション) 生成されたファクト・テーブルにおける、このレベルに対応する列の名前。この属性が NULL の場合はシステムによって名前が生成されます。このオプションは、時間レベルまたは NLP レベルには適用されません。 <measure> の factName の説明を参照してください。
factNumber	このレベルに割り当てられている内部 ID。キューブの namedFactNums が "true" の場合は必須です。
dependsOn	(オプション) このレベルが依存関係を持つレベル (またはリレーションシップ) を指定します。レベル (またはリレーションシップ) の完全な MDX 識別子を指定します。または、MDX レベル (またはリレーションシップ) 識別子のコンマ区切りリストを指定します。"異なる階層に含まれるレベル間での依存関係の定義" を参照してください。この属性は、DependsOn コンパイラ・キーワードとはまったく関係ありません。
useAsFilter	(オプション) ダッシュボードのフィルタ・コントロールとしてレベルを使用できるかどうかを指定します。この属性が "true" (既定) の場合、ユーザは、フィルタ・コントロールを追加するときにこのレベルを選択できます。この属性が "false" の場合、このレベルはオプションとしてリストされません。このオプションの目的は、ダッシュボード・エディタに表示される選択肢の数を減らすことのみです。アナライザやエンジンには影響しません。
factSelectivity	(オプション、システムでは不使用) 生成されたレベル・テーブル内のプロパティの生成 SELECTIVITY をオーバーライドするための値。Business Intelligence のクエリでは、このパラメータは使用しません。このオプションは、レベル・テーブルに対して直接 SQL を使用するときに、生成された SELECTIVITY をオーバーライドする必要がある場合に使用します。 1 以下の正の値を指定します。詳細は、%SYSTEM.SQL の SetFieldSelectivity() を参照してください。

属性	目的
hidden	(オプション) hidden="true" の場合、レベルは定義され、クエリで使用することができますが、アナライザはレベルを使用可能なものとして表示しません。既定は "false" です。
<member>	(オプション) 任意の数の <member> 要素を含むことができます。このそれぞれがレベルのメンバを定義します。
<property>	(オプション) 任意の数の <property> 要素を含むことができます。このそれぞれがレベルのプロパティを定義します。

例

以下はその例です。

```
<level name="ZIP" sourceProperty="HomeCity.PostalCode" />
```

別の例として、以下のレベル定義は、そのレベルのプロパティも定義します。

```
<level name="City" sourceProperty="HomeCity.Name">
  <property name="Population" sourceProperty="HomeCity.Population" />
```

他のテーブルへのリンク

<level>、<measure>、または <property> 要素内では、linkClass 属性と linkProperty 属性により、ドット構文ではアクセスできない別のクラスのプロパティを使用できます。このリンク機能を使用するには、レコードの ID を使用して別のクラスのレコードを検索できる必要があります。

この機能は、以下のように実行されます。

- 他のクラスの必要なレコードの ID を指定する値として、sourceProperty または sourceExpression を指定します。
- 他のクラスの完全なパッケージおよびクラス名として linkClass を指定します。
- 定義しているレベルの基とする他のクラスのプロパティとして linkProperty を指定します。

例えば、Hole Foods のサンプルに、以下を追加することができます。

XML

```
<level name="Product" sourceProperty ="Product" linkClass="HoleFoods.Product" linkProperty="Name" />
```

外部クラス linkClass とプロパティ linkProperty が定義されていると、システムによって外部クラスに対してクエリが実行され、指定された sourceProperty または sourceExpression に等しい ID のレコードについて、そのクラスから指定プロパティの値がフェッチされます。

日付オフセットの指定

場合によっては、開始日が 1 月 1 日ではない企業の財務カレンダーに時間レベルを一致させることが必要になります。例えば、会計年度は多くの企業で 10 月 1 日に開始します。以下のピボット・テーブルで考えてみましょう。

Year	Amount Sold
FY 2005	7,131.04
FY 2006	12,892.19
FY 2007	14,653.86
FY 2008	16,540.95
FY 2009	18,987.04
FY 2010	16,666.45

この場合、メンバ FY 2005 は、2004 年 10 月 1 日から 2005 年 9 月 30 日までの (端点を含む) 売上レコードで構成されています。

このようなレベルを作成するには、レベル (時間タイプのディメンジョン内にある必要があります) の timeOffset 属性と timeFormat 属性を指定します。ここでは、timeOffset 属性について説明します。timeFormat については次のセクションで説明します。

timeOffset 属性は、このレベルで使用されるソース値に加えられる時間量を指定します。この時間量は、負でも正でもかまいません。これは、キューブの構築時に使用されます。

timeOffset では、以下の形式の文字列を指定することによって、時間量を指定します。

```
#y#m#d
```

は数です。#y は年単位の時間量、#m は月単位の時間量、#d は日単位の時間量を表します。要素を省略した場合、その位置にはゼロが使用されます。例えば、文字列 3m15d は、3 か月と 15 日を表します。

timeOffset で最も使用頻度の高い値は -3m です。これは会計年度が前年 10 月 1 日から開始される場合に使用されます。timeOffset="-3m" の場合、このレベルで使用される各時間値から 3 か月が差し引かれます。例えば、このレベルの日付 2010 年 1 月 1 日は、2009 年 10 月 1 日に変換されます。

同一ディメンジョン内であっても、他のレベルに影響はありません。このため、実際の日付を表示する、より詳細なレベルも定義できます。この例については、"日付オフセットがあるカレンダーの処理" を参照してください。

timeFunction 属性は同じであるが、timeOffset 属性が異なる 2 つのレベルがある場合、アーキテクトはこれらを同じファクトの 2 つの異なる値として認識します。これらは、ファクト・テーブルの 2 つの異なる列に自動的にマップされ、それぞれ別個の factNumber が割り当てられます。

基本的な範囲式の定義

rangeExpression の基本的な構文は以下のとおりです。

```
value_or_range:new_value;value_or_range:new_value; ... ;
```

ここで、new_value は単一の値で、システムはこれを文字列として処理し、レベル・メンバの名前として使用します。また、value_or_range は、以下のいずれかになります。

- 5 や Louisiana などの単一の値
- (value1,value2) 形式の範囲
- [value1,value2] 形式の範囲
- (value1,value2] 形式の範囲
- [value1,value2) 形式の範囲

これらの式で、value1 と value2 は、数値または NULL (式では省略される) です。左丸括弧と右丸括弧は、指定された端点がその範囲に含まれないことを示します。左角括弧と右角括弧は、指定された端点がその範囲に含まれることを示します。例えば、(45,49] は、45 より大きく 49 以下のすべての値を表します。

複数範囲式の圧縮の定義

場合によっては、指定されたレベルの複数の範囲を定義する必要があります。これは単調なプロセスになる可能性があります。また、結果の範囲式は、スタジオで表示したときに読みづらくなります。このような場合には、以下に示す rangeExpression の値の代替構文を使用できます。

```
rangeExpression="[start:increment:end]:replacement;"
```

または、両方の角括弧の代わりに適切な括弧を使用できます (下記の詳細を参照)。

この構文は一連の範囲を生成します。

この start は最初の範囲の開始値 (数値)、end は最後の範囲の終了値 (数値)、increment は範囲を定義する数値です。最初の範囲は start と start + increment の間などになります。

また、replacement は、置換値として使用される式です。replacement では、以下の要素を使用できます。

- %1 ー 範囲の開始値で置換します。
- %2 ー 範囲の終了値で置換します。
- $$$eval(expression) ー これにより、システムは含まれた式を評価します。例えば、$$$eval(%2-1) は範囲の終了値から 1 を引いた数を返します。

角括弧や括弧は、生成された start および end の範囲内で start 値と end 値がどのように処理されるかに影響を与えます。

- 角括弧は値を範囲に含めます。
- 括弧は値を範囲から除外します。

中間範囲の境界では、境界値は下位の範囲ではなく、常に上位の範囲に割り当てられます。つまり中間範囲では、開く括弧は常に [、閉じる括弧は常に) になります。

以下の範囲式を考えてみます。

```
rangeExpression="[0:30:90]:%1 to $$$eval(%2-1);"
```

これは、以下に示す長い式と同じメンバを生成します。

```
rangeExpression="[0,30):0 to 29;[30,60):30 to 59;[60,90]:60 to 90;"
```

値 end − start が increment の整数倍でない場合、最後の範囲は end を超えます。例えば、以下の範囲式を考えてみます。

```
rangeExpression="[0:30:100]:%1 to $$$eval(%2-1);"
```

これは、以下に示す長い式と同じメンバを生成します。

```
rangeExpression="[0,30):0 to 29;[30,60):30 to 59;[60,90):60 to 90;[90,119]:90 to 119;"
```

〈member〉

(特殊なケース) Business Intelligence キューブで、レベルのメンバを手動で指定する際に使用します。

詳細

<member> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
spec	レベルが、type="computed" が指定されているディメンジョンに含まれる場合は、このメンバが使用するファクト・テーブル行 (またはソース・テーブル行) の ID を返す SQL クエリを指定します。 それ以外の場合は、必要に応じて、メンバ・キーとして使用する値を指定します。これを省略すると、name がメンバ・キーになります。

〈property〉

Business Intelligence キューブで、レベルにオプションのプロパティを定義します。

詳細

レベルは、ゼロ個以上のカスタム・レベル・プロパティを含むことができます。これらのプロパティは、値がソース・データから導出され、レベルの特定のメンバに関連付けられます。例えば、市区町村のレベルには、人口や郵便番号などのプロパティを組み込むことができます。各市区町村に、これらの各プロパティの値が 1 つ存在します。

<property> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
sourceProperty、sourceExpression	アーキテクトで [プロパティ] または [式] を指定する際とほとんど同じ方法で、これらの属性のいずれかを指定します。"ディメンジョンまたはレベルのソース値の定義" と "ソース式の詳細" を参照してください。メモ： ・ sourceProperty には、[プロパティ] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceProperty='MyProp' ・ sourceExpression には、[式] に入力するのと同じ値を一重引用符で囲んで使用します。 例：sourceExpression='%source.MyProp_"ABC"' 値自体に二重引用符が含まれない場合は、代わりに二重引用符で値を囲むこともできます。例：sourceExpression="%source.MyProp"
sort	(オプション) このプロパティが属するレベルのメンバを並べ替える際にこのプロパティをどのように使用するかを指定します。レベルは複数のプロパティを基準にして並べ替えることができます。その場合、<property> 要素を定義した順序で並べ替えが適用されます。最初のプロパティが一次の並べ替えを制御し、2 番目のプロパティが二次の並べ替えを制御するというように続きます。 "asc"、"asc numeric"、"desc"、または "desc numeric" を指定します。 既定では、プロパティはメンバの並べ替え順序に影響しません。
isName	(オプション) "true" の場合、指定されたレベルのメンバについて、システムがこのプロパティの値を使用してメンバの名前を指定することが、この属性によって指定されます。"true" または "false" (既定) を指定します。
isDescription	(オプション) "true" の場合、指定されたメンバについて、システムがこのプロパティの値をメンバのツールのヒントに使用することが、この属性によって指定されます。"true" または "false" (既定) を指定します。

属性	目的
isReference	(オプション) "true" の場合、システムがこのプロパティの値を格納するのではなく、プロパティを元のソース・テーブルを参照する SQL 計算フィールドとして定義することが、この属性によって指定されます。isReference="true" と指定した場合は、メンバ・キーが、レベル (およびそのプロパティ) の基礎となっているレコードの ID になるように、レベルを定義する必要があります。
useDisplayValue	(オプション) DISPLAYLIST および VALUELIST パラメータの値があるクラス・プロパティの場合、この属性によってプロパティに使用する値が指定されます。この属性が "true" (既定) の場合は、DISPLAYLIST で指定された値が使用されます。この属性が "false" の場合は、VALUELIST で指定された値が使用されます。
linkClass、linkProperty	(オプション) ″他のテーブルへのリンク″ を参照してください。
factName	(オプション) (生成されたディメンジョン・テーブルで) プロパティに対応する列に使用される名前。この属性が NULL の場合はシステムによって名前が生成されます。生成されたディメンジョン・テーブルに対して SQL クエリを直接発行する予定がない限り、この属性は重要ではありません。 この属性の名前が存在しても、プロパティはファクト・テーブルにはありません。プロパティは、それらが属するレベルに対応するテーブルにあります。 〈measure〉 の factName の説明を参照してください。
formatString	(オプション) 値の表示方法を制御します。″formatString の詳細″ を参照してください。書式文字列では、特殊文字 0 も使用できます。これは先頭のゼロのプレースホルダとして使用します。例えば、00000 は、先頭のゼロが埋め込まれた 5 桁の数値を示します。
hidden	(オプション) hidden="true" の場合、このプロパティは定義され、クエリで使用することができますが、アナライザで使用可能なプロパティとしてリストには表示されません。

例

以下に例を示します。

```
<property name="Population" sourceProperty="City.Population"/>
```

内部プロパティ

各レベルに対して、システムは自動的に一連の内部プロパティも定義します。これらのプロパティは、″InterSystems MDX リファレンス″ を参照してください。

〈listing〉

Business Intelligence キューブでオプションの名前付きリストを定義します。

詳細

キューブは、ゼロ個以上の名前付きリストを含むことができます。これらは、アナライザで使用できます。

キューブの既定のリストは、<cube> 要素の defaultListing 属性（指定されている場合）か、<cube> に含まれる最初の <listing> 要素で指定されたリストです。

<listing> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
fieldList	表示するフィールドのコンマ区切りのリスト。オプションの詳細は、"単純なリストの定義" を参照してください。データ・コネクタ・リストのオプションの詳細は、"データ・コネクタ・リストの定義" を参照してください。
formatList	リスト用の CSS 形式指示のリスト（キャレット区切り）。リストの各部分は、リスト内の対応する列に適用されます。このリストは、列にある要素と同数の要素を持っている必要があり、各リストの要素は、以下の形式のいずれかを持っている必要があります。 ・ {CSS formatting instruction}、例：{text-align:center;} ― 対応する列の形式を設定します。 {text-align:left;} または {text-align:center;} を使用できます。その他の指示は無視されます。 ・ display:none; ― 対応する列を非表示にします。 ・ Null ― 対応する列を既定のスタイルで表示します。
orderBy	単純なリストにのみ適用されます。この属性は、リストの並べ替えの基準となるフィールドのコンマ区切りのリストです。全体的な並べ替えは、リストの最初のフィールドによって制御され、二次的な並べ替えは 2 番目のフィールドによって制御される、というように続きます。フィールド名の後に ASC または DESC キーワードを組み込んで、それぞれ昇順または降順で並べ替えることができます。その他の詳細は、"単純なリストの定義" を参照してください。 ソースがデータ・コネクタである場合、この属性は無視されます。
sourceClass	データ・コネクタが存在する場合に、このリストの基になるデータ・コネクタを指定します。この属性を指定する場合は、%DeepSee.DataConnector を拡張するクラスの名前を指定します。"InterSystems Business Intelligence の実装" を参照してください。
sql	このリストに対するカスタム SQL クエリ（存在する場合）を指定します。"SQL カスタム・リストの定義" を参照してください。

属性	目的
listingType	(オプション) リストの形式を指定します。既定は、"table" です。この代わりに "map" を指定すると、マップタイプ・リストが表示されます。マップタイプ・リストではポイントが緯度と経度で示されるマップが表示されます。この場合は、リスト・クエリにフィールド Latitude および Longitude を含める必要があります (大文字と小文字は区別されます)。"マップ・リスト (地理リスト) の定義" を参照してください。
selectMode	(オプション) このリストの SQL クエリに使用される %SelectMode を指定します。この属性によって、特定タイプのデータの表示方法が決まります。既定値は、"display" です。他の指定可能な値は、"logical" および "odbc" です。"リストの追加" を参照してください。
resource	(オプション) このリストへのアクセスの制御に使用されるリソースの名前。"InterSystems Business Intelligence の実装" を参照してください。

リストには、ソースに応じて、以下の 3 種類があります。

- 基本リスト。キューブで使用されるソース・テーブルを直接使用します。
- データ・コネクタ・リスト。データ・コネクタを使用します。
- カスタム・リスト。カスタム・クエリを使用します。

いずれの場合も、システムでは SQL クエリが作成され、使用されます。

以下のテーブルは、各種のリストに指定される <listing> の属性を示しています。

リストの種類	fieldList	orderBy	sourceClass	sql
基本	必須	オプション	指定しません。この属性は優先されます。	指定しません。この属性は優先されます。
データ・コネクタ	オプション	無視	必須	無視
カスタム	無視	無視	指定しません。この属性は優先されます。	必須

<listingField>

エンド・ユーザが Business Intelligence キューブでカスタム・リストを作成する際に使用できるオプションのリスト・フィールドを定義します。

詳細

注釈　システムでは、もう 1 つの種類のカスタム・リスト (キューブの必ずしもソース・テーブルを使用するとは限らないリスト) がサポートされています。"リストの定義" の "カスタム・リストの定義" を参照してください。

<listingField> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
fieldExpression	ソース・テーブルのフィールドまたは (矢印構文などを使用して) 関連テーブルのフィールドを参照する SQL 式。displayName を指定する場合、SQL 式内でエイリアスを指定しないでください。これに反して、SQL 式内でエイリアスを指定した場合、displayName は無視され、フィールド名をローカライズできません。詳細は、"リスト・フィールドの作成" を参照してください。
resource	(オプション) このリスト・フィールドへのアクセスの制御に使用されるリソースの名前。"InterSystems Business Intelligence の実装" を参照してください。

例

以下に例を示します。

```
<listingField name="PatientID" displayName="PatientID" fieldExpression="PatientID" />
<listingField name="Age" displayName="Age" fieldExpression="Age" />
<listingField name="Gender" displayName="Gender" fieldExpression="Gender" />
<listingField name="Test Score" displayName="Test Score" fieldExpression="TestScore" />
<listingField name="City" displayName="City" fieldExpression="HomeCity->Name" />
<listingField name="Doctor" displayName="Doctor" fieldExpression="PrimaryCarePhysician->LastName" />
```

<calculatedMember>

(Business Intelligence キューブ内のレベルの) 他のメンバに関して定義されるメンバである、オプションの計算メンバを定義します。

詳細

以下の 2 種類の計算メンバを追加できます。

- 他のメジャーに基づく新しいメジャーを定義できます。例えば、以下のような数式でメジャーを定義できます。

  ```
  Measure 3 = (Measure 1 + Measure 2) / Measure 2)
  ```

  これは正確な構文ではありません。

- 他のメジャー以外のメンバに基づく新しいメンバを定義できます。例えば、Favorite Color ディメンジョンの red、yellow、および blue のメンバを結合する Primary Colors メンバを作成できます。

  この新しい Primary Colors メンバは、red、yellow、および blue のメンバに対応するファクト・テーブルのすべてのレコードを参照します。

MDX では、メジャーはメンバと見なされます。これが、両方の種類の計算要素が計算メンバであると見なされる理由です。

注釈　現在、この要素を使用してメジャーを定義する際に MDX シェルの cube コマンドではこのメジャーがリストされません。ただし、MDX シェルまたはクエリ API のセットは使用できます。

<calculatedMember> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。name 属性について、システムでは名前が既に使用されているかどうかのチェックが行われません。同じディメンジョンの別メンバが既に使用している名前を使用すると、そのメンバがオーバーライドされます。
dimension	このメンバが属するディメンジョン。
valueExpression	他のメンバへの参照に関連してこのメンバの値を定義する MDX 式。単純で一般的なシナリオについては、"計算要素の定義" を参照してください。詳細と例は、"InterSystems MDX リファレンス" を参照してください。
formatString	(オプション) 値の表示方法を制御します。"書式文字列の指定" を参照してください。
units	(オプション) メジャー値が表示される単位を示します。現在、この属性は、一般情報としてのみ提供されています。
listingFilter	(オプション、dimension が "measures" である場合のみ) ユーザがピボット・テーブルでこの計算メジャーを表示し詳細リストを要求する際に使用される追加フィルタを指定します。"計算メジャーのリストの追加フィルタの指定" を参照してください。

注釈　上記以外に、計算メンバは以下の 2 つの方法でも定義できます。

- MDX クエリでの WITH 節の使用。
- CREATE MEMBER 文の中。

  これは MDX シェル内でのみ有効です。

"InterSystems MDX リファレンス" を参照してください。

例

以下はその例です。

```
<calculatedMember name="Avg Age" dimension="MEASURES"
valueExpression="[MEASURES].[Age]/[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 Age および Patient Count メジャーが評価され、次に除算が実行されます。

<namedSet>

(Business Intelligence キューブ内のレベルの) メンバまたはメンバ・セットのエイリアスである、オプションの名前付きセットを定義します。名前付きセットは、クエリの行や列で使用でき、クエリの実行時にシステムによってメンバまたはセットに置換されます。

詳細

注釈　名前付きセットを定義する際、MDX シェルの cube コマンドではこのセットがリストされません。ただし、MDX シェルまたはクエリ API のセットは使用できます。

<namedSet> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。name 属性について、システムでは名前が既に使用されているかどうかのチェックが行われません。別の名前付きセットが既に使用している名前を使用すると、その名前付きセットがオーバーライドされます。
setExpression	メンバまたはメンバのセットを返す MDX 式。詳細と例は、"InterSystems MDX リファレンス" を参照してください。

注釈　上記以外に、名前付きセットは以下の 2 つの方法でも定義できます。

- MDX クエリでの WITH 節の使用。
- CREATE SET 文の中。

"InterSystems MDX リファレンス" を参照してください。

例

以下に例を示します。

```
<namedSet name="SampleSet" setExpression="[homed].[h1].[city].MEMBERS" />
```

〈relationship〉

Business Intelligence キューブのリレーションシップを定義します。

詳細

あるキューブから別のキューブへの単方向リレーションシップを定義するには、最初のキューブで <relationship> 要素を定義します。

双方向リレーションシップを定義するには、各キューブに 1 つずつ、計 2 つの補完的な <relationship> 要素を定義します。

重要　キューブ・クラスをコンパイルする際は、独立キューブを最初にコンパイルします。これはリレーションシップのソース・プロパティまたはソース式を定義しないキューブです。コンパイル順序を制御するには、依存キューブのクラス定義で DependsOn キーワードを指定します。

同様に、独立キューブを最初に構築する必要があります。DependsOn キーワードはキューブの構築順に影響しません。

<relationship> 要素には以下の属性があります。

属性	目的
name	リレーションシップの名前。他のキューブのレベルを使用するには、MDX クエリでこの論理名を使用します。通常、これは他のキューブの名前です。
displayName、description、disabled	"キューブの共通属性" を参照してください。
relatedCube	他方のキューブの論理名。
inverse	他方のキューブの <relationship> 要素内の name 属性の値。
cardinality	リレーションシップのカーディナリティ。
sourceProperty、sourceExpression	"キューブ間のリレーションシップの定義" の "単方向リレーションシップの定義" および "双方向リレーションシップの定義" を参照してください。
nullReplacement	(オプション) このリレーションシップのソース・データが NULL の場合に、メンバ名として使用する文字列を指定します。例えば、nullReplacement="No City" と指定します。この属性は、sourceProperty または sourceExpression を指定する <relationship> 内で指定します。
factName	(オプション)(ファクト・テーブルで) リレーションシップに対応する列に使用される名前。この属性が NULL の場合はシステムによって名前が生成されます。 〈measure〉 の factName の説明を参照してください。
factNumber	このリレーションシップに割り当てられている内部 ID。キューブの namedFactNums が "true" の場合は必須です。
linkClass、linkProperty	使用しません。これらは無視されます。

属性	目的
dependsOn	(オプション) このリレーションシップが依存関係を持つリレーションシップを指定します。リレーションシップの論理名を指定します。"異なる階層に含まれるレベル間での依存関係の定義" を参照してください。(または、リレーションシップがレベルに依存する場合は、そのレベルの MDX 識別子を指定します)。 この属性は、DependsOn コンパイラ・キーワードとはまったく関係ありません。

inverse および cardinality の詳細

以下のテーブルは、各シナリオで指定するキーワードをまとめたものです。

属性	単方向リレーションシップ	双方向リレーションシップ
inverse	これを省略します。	双方のキューブでこれを指定します。
cardinality	"one" を使用します。	· 依存キューブでは "one" を使用します。 · 独立キューブでは "many" を使用します。
sourceProperty または sourceExpression	通常どおり指定します。	依存キューブのみに適用できます。
factName	通常どおり指定します。	依存キューブのみに適用できます。他方のキューブでは無視されます。

例

例については、"キューブ間のリレーションシップの定義" を参照してください。

〈expression〉

Business Intelligence キューブで使用するオプションの式を定義します。

詳細

<expression> 要素には以下のコンテンツがあります。

属性	目的
name、description、disabled	"キューブの共通属性" を参照してください。
sourceProperty、sourceExpression	"InterSystems Business Intelligence の上級モデリング" を参照してください。
displayName、factName、linkClass、linkProperty	これらの属性は指定しないでください。

<index>

Business Intelligence キューブに関して生成されるファクト・テーブルに追加するオプションのカスタム・インデックスを定義します。システムはこのインデックスを使用しません（必要なインデックスは自動的に追加されるため）。カスタム・インデックスは、SQL を介してファクト・テーブルにアクセスする計画がある場合に追加できます。

詳細

システムはこのインデックスを使用しません（必要なインデックスは自動的に追加されるため）。カスタム・インデックスは、SQL を介してファクト・テーブルにアクセスする計画がある場合に役立つことがあります。

<index> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"キューブの共通属性" を参照してください。
type	インデックスのタイプを指定します。"bitmap"、"bitslice"、"index"、または "key" を使用します。
properties	インデックスの基とするファクト・テーブルのフィールドを指定します。ファクト・テーブル・クラスのプロパティのコンマ区切りリストを指定します。

Tipヒント ほとんどのレベルおよびメジャーで、factName 属性を指定して、生成されたファクト・テーブル・クラスのプロパティ名を制御できることを覚えておいてください。

例

```
<index name="IndexName" type="bitmap"  properties="MxAge,DxGender"/>
```

サブジェクト領域クラスのリファレンス情報

Business Intelligence サブジェクト領域は、テーブルの SQL ビューに似ています。サブジェクト領域は、複数のキューブを必要とせずに、より小さなデータ・セットに焦点を当てることのできるサブキューブです。また、サブジェクト領域を使用すると、キューブのキャプションおよび既定値をカスタマイズできます。

このリファレンスでは、サブジェクト領域クラスの詳細を提供します。

"キューブおよびサブジェクト領域の高度な機能の使用" も参照してください。

サブジェクト領域クラスの要件

Business Intelligence サブジェクト領域クラスの基本情報を提供します。

詳細

サブジェクト領域を定義するには、以下の要件に合致したクラスを作成します。

- %DeepSee.SubjectArea を拡張する必要があります。
- `SubjectArea` という名前の XData ブロックを格納している必要があります。
- この XData ブロックに対して、以下のように XMLNamespace が指定されている必要があります。

  ```
  XMLNamespace = "http://www.intersystems.com/subjectarea"
  ```

- XData ブロック内のルート要素が `<subjectArea>` であり、この要素がこのページの他の部分に記載された要件に従っている必要があります。
- クラスで DependsOn キーワードを指定することによって、キューブ・クラスがコンパイルされて使用できるようになった後にのみ、このクラスがコンパイルされるようにすると便利です。ただし、必ずしもそうする必要はありません。
- クラスでは DOMAIN パラメータを定義できます。これによってローカライズされた文字列が属するドメインが指定されます。以下はその例です。

  Class Member

  ```
  Parameter DOMAIN = "PATIENTSAMPLE";
  ```

  詳細は、"ローカライズの実行" を参照してください。

スタジオでは、入力に応じたサポートが提供されます。

変更を加えた後は、サブジェクト領域クラスをリコンパイルする必要があります。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

例

以下に例を示します。

```
Class BI.Model.SubjectAreas.AsthmaPatients Extends %DeepSee.SubjectArea [DependsOn=Cubes.StudyPatients]
{

/// This XData definition defines the SubjectArea.
XData SubjectArea [ XMLNamespace = "http://www.intersystems.com/deepsee/subjectarea" ]
{
<subjectArea name="AsthmaPatients"
    displayName="Asthma Patients"
    baseCube="Patients" filterSpec="diagd.h1.diagnoses.asthma" >
    </subjectArea>
}

}
```

サブジェクト領域クラスの共通属性

Business Intelligence サブジェクト領域定義全体で使用される属性をリストします。

詳細

サブジェクト領域内の要素の大部分には以下の属性があります。わかりやすくするため、ここにこれらをリストします。

属性	目的
name	サブジェクト領域の基となるベース・キューブで指定された、要素の論理名。
displayName	(オプション) ユーザ・インタフェースで使用されるこの要素のローカライズ名。この属性を指定しない場合、代わりに論理名が表示されます。詳細は、"ローカライズの実行" を参照してください。
description	(オプション) この要素の説明。この属性を指定しない場合、代わりにキューブ定義で指定された description が表示されます。
disabled	(オプション) コンパイラが、この要素で定義されたオーバーライドを使用するかどうかを制御します。この属性が "true" の場合、コンパイラはこのオーバーライドを無視します。これは、オーバーライドをコメントアウトすることと同等です。既定では、この属性は "false" です。 オーバーライドが無効の場合、システムはキューブで指定される定義を使用します。

〈subjectArea〉

Business Intelligence サブジェクト領域クラスでサブジェクト領域を定義します。

詳細

<subjectArea> 要素には以下の項目が含まれます。

属性または要素	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。 name 属性は、InterSystems IRIS® データ・プラットフォームのネームスペース内で一意である必要があります。
baseCube	このサブジェクト領域の基となるキューブの論理名 (これには、論理キューブ名のコンマ区切りリストでもかまいません。"複合キューブの定義" を参照してください)。
owner	(オプション) サブジェクト領域の所有者の名前。
resource	(オプション) アーキテクトを介してアクセスする際に、このサブジェクト領域へのアクセスの制御に使用されるリソースの名前。"InterSystems Business Intelligence の実装" を参照してください。
filterSpec	(オプション) このサブジェクト領域のフィルタとして使用される MDX セット式。サブセクション "サブジェクト領域のフィルタ処理" を参照してください。既定は、空白文字列で、フィルタリングは行われません。
caption	(オプション) このサブジェクト領域のキャプション。これを指定しない場合、代わりにベース・キューブのキャプションが使用されます。
countMeasureCaption	(オプション) 既定のメジャーに使用するキャプション。レコード数をカウントします。既定のキャプションは Count です。内部的には、このメジャーの名前は %Count です。
defaultListing	(オプション) このサブジェクト領域の既定として使用される <listing> の論理名を指定します。"〈listing〉" を参照してください。これを指定しない場合は、ベース・キューブに指定されている既定のリストが使用されます。
disableListingGroups	(オプション) この属性に "true" を指定すると、すべてのユーザはこのサブジェクト領域をターゲットとして使用するリスト・グループを定義できなくなります。既定は "false" です。"リスト・グループのコンパイル" を参照してください。
defaultMember、defaultMeasure	使用しません。
<measure>	(オプション) ゼロ個以上の <measure> 要素を含むことができます。このそれぞれでメジャーの非表示やカスタマイズを行うことができます。
<dimension>	(オプション) ゼロ個以上の <dimension> 要素を含むことができます。このそれぞれでディメンジョンの非表示やカスタマイズを行うことができます。
<listing>	(オプション) ゼロ個以上の <listing> 要素を含むことができます。このそれぞれでこのサブジェクト領域のリストの非表示、カスタマイズ、または追加を行うことができます。

サブジェクト領域のフィルタ処理

filterSpec 属性を使用すると、サブジェクト領域に適用するフィルタを指定できます。この属性は、有効な MDX セット式と等価である必要があります。以下はその例です。

```
{AgeD.H1.[10 to 19],AgeD.H1.[20 to 29]}
```

"フィルタ式の作成" を参照してください。

filterSpec を指定する代わりに (またはこの指定に加えて)、%OnGetFilterSpec コールバックを実装することもできます。"キューブまたはサブジェクト領域の動的フィルタ処理" を参照してください。

〈measure〉

Business Intelligence サブジェクト領域で、メジャーを非表示にするか、カスタマイズします。

詳細

<measure> 要素には以下の属性があります。

属性	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
hidden	(オプション) hidden="true" の場合、このメジャーは定義され、クエリで使用することができますが、このサブジェクト領域で使用可能なメジャーとしてリストには表示されません。既定は "false" です。
formatString	(オプション) 指定すると、ベース・キューブでこのメジャーに対して指定された formatString 属性がこの属性によってオーバーライドされます。"書式文字列の指定" を参照してください。

注釈　この方法では、計算メジャーにオーバーライドを定義することはできません。計算メジャーは実際には計算メンバです。これにオーバーライドを定義するには、<calculatedMember> を使用する必要があります。

〈dimension〉

Business Intelligence サブジェクト領域で、ディメンジョンを非表示にするか、カスタマイズします。

詳細

<dimension> 要素には以下のコンテンツがあります。

属性または要素	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
hidden	(オプション) hidden="true" の場合、このディメンジョンは定義され、クエリで使用することができますが、使用可能なディメンジョンとしてリストには表示されません。既定は "false" です。
allCaption	(オプション) 指定すると、ベース・キューブでこのディメンジョンに対して指定された allCaption 属性がこの属性によってオーバーライドされます。
allDisplayName	(オプション) 指定すると、ベース・キューブでこのディメンジョンに対して指定された allDisplayName 属性がこの属性によってオーバーライドされます。
〈hierarchy〉	(オプション) ゼロ個以上の <hierarchy> 要素を含むことができます。このそれぞれで階層 (または階層の一部) を非表示にすることができます。

<hierarchy>

Business Intelligence サブジェクト領域で、階層を非表示にするか、カスタマイズします。

詳細

<hierarchy> 要素には以下のコンテンツがあります。

属性または要素	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
<level>	(オプション) ゼロ個以上の <level> 要素を含むことができます。このそれぞれがレベルを非表示にできます。
hidden	(オプション) この属性が "true" の場合、このサブジェクト領域でその階層を使用することはできません。既定は "false" です。 ディメンジョンのすべての階層が非表示である場合、そのディメンジョン自体がサブジェクト領域内で非表示になります。この方法でディメンジョンを非表示にすると、同じディメンジョンに配置できるすべての計算メンバもシステムによって非表示にされます。

<level>

Business Intelligence サブジェクト領域で、レベルを非表示にするか、カスタマイズします。

詳細

<level> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
hidden	(オプション) この属性が "true" の場合、このサブジェクト領域にそのレベルは表示されません。既定は "false" です。
sort	(オプション) 時刻ディメンジョンのレベルの場合、このレベルのメンバの既定の並べ替え方法を指定します。"asc" または "desc" を指定します。 この属性は、データ・ディメンジョンまたは年齢ディメンジョンのレベルには影響しません。

〈listing〉

Business Intelligence サブジェクト領域で、リストを非表示にするか、カスタマイズするか、追加します。

詳細

<listing> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
hidden	(オプション) この属性が "true" の場合、このサブジェクト領域にそのリストは表示されません。既定は "false" です。
他の属性	(オプション) "〈listing〉" を参照してください。これらは、リストの再定義または新規リストの追加を行う場合にのみ指定します。

<calculatedMember>

Business Intelligence サブジェクト領域で、計算メンバを非表示にするか、カスタマイズします。

詳細

<calculatedMember> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
dimension	このメンバが属するディメンジョン。
hidden	(オプション) この属性が "true" の場合、このサブジェクト領域にその計算メンバは表示されません。既定は "false" です。
他の属性	(オプション) "<calculatedMember>" を参照してください。これらは、計算メンバの再定義または追加を行う場合にのみ指定します。

<namedSet>

Business Intelligence サブジェクト領域で、名前付きセットを非表示にするか、カスタマイズします。

詳細

<namedSet> 要素には以下のコンテンツがあります。

属性	目的
name、displayName、description、disabled	"サブジェクト領域の共通属性" を参照してください。
hidden	(オプション) この属性が "true" の場合、このサブジェクト領域にその名前付きセットは表示されません。既定は "false" です。
他の属性	(オプション) 前の項目の "<calculatedMember>" を参照してください。これらは、名前付きセットの再定義または追加を行う場合にのみ指定します。

ファクト・テーブルおよびディメンジョン・テーブルの詳細

Business Intelligence の内部ではスター・スキーマが使用されます。これは、ファクト・テーブルと、ファクト・テーブルが参照するディメンジョン・テーブルで構成されます。ディメンジョン・テーブルはスター・テーブルとも呼ばれることに注意してください。これらのテーブルには、直接クエリを実行することができます。これらは、キューブを構築する際にシステムによって生成され、変更が発生するとシステムによって更新されます。

このリファレンスでは、以下のテーブルの詳細を提供します。

重要　　許可されたインタフェースを使用する場合を除き、これらのテーブルの再定義やこれらへのデータの書き込みはしないでください。"キューブの最新状態の維持" を参照してください。

キューブ定義クラスをコンパイルすると、`Listing` という名前のテーブルもファクト・テーブルと同じパッケージ内に生成されます。この `Listing` テーブルは内部使用専用のため、使用しないでください。

"このドキュメントに示したサンプルのアクセス方法" も参照してください。

ファクト・テーブル

Business Intelligence キューブ定義クラスをコンパイルする際に生成されるファクト・テーブルについて説明します。

基本情報

キューブ定義クラスをコンパイルすると、システムによって Package_CubeClass.Fact という名前の対応するファクト・テーブルが生成されます。Package_CubeClass は、パッケージとクラス名をテーブル名に変換する際の通常のルールに従い、キューブ定義のパッケージとクラスに対応します。スター・テーブル ID へのリンクを宣言する外部キー定義も、ファクト・テーブル内に生成されます。

キューブを構築したり、インクリメンタルに更新すると、常にファクト・テーブルが更新されます。

このテーブルおよびそれを使用するクラス定義を調べると、役に立ちます。

このファクト・テーブルには以下のフィールドがあります。

- ID ― この行の ID。行の作成時に割り当てられます。
- %dspartition ― 将来使用予定。このフィールドは無視してください。
- %sourceID ― このレコードの基となるベース・クラスにおけるレコードの ID。このフィールドは、ソース・テーブルを参照するポインタです。
- メジャーごとにフィールドが 1 つ存在し、このレコードに対するこのメジャーの値が格納されます (ただし、1 つ例外があります。このページで後述の "ファクト・テーブルにおけるソース・プロパティの再使用" を参照してください)。

  このフィールドには、実際のメジャー値が格納されます。ファクト・テーブル・クラスは、これらの値にビットスライス・インデックスを定義します。以下はその例です。

```
/// Index for measure M1.
Index MxAge On MxAge [ Type = bitslice ];


/// Measure: MxAge <br/>
/// Source: Age
Property MxAge As %Integer;
```

- レベルごとにフィールドが 1 つ存在し、このレコードが属するレベル・メンバを示します (ただし、1 つ例外があります。このページで後述の "ファクト・テーブルにおけるソース・プロパティの再使用" を参照してください)。
  - データ・ディメンジョン内のレベルの場合、ファクト・テーブルにはレベル・メンバの ID が格納されます。この ID は、メンバを定義するテーブルへのポインタです。

    ファクト・テーブル・クラスは、これらの値にビットマップ・インデックスを定義します。以下はその例です。

```
/// Index for fact 1. Index DxGender On DxGender [ Type = bitmap ];

/// Dimension: DxGender <br/>
/// Source: Gender
Property DxGender As Test.TestCube.DxGender;
```

    このディメンジョンが別のキューブによって共有される場合、このポインタは他方のキューブの該当するディメンジョン・テーブルのレコードを参照します。共有ディメンジョンの詳細は、"InterSystems Business Intelligence の上級モデリング" を参照してください。
  - 時間タイプおよび年齢タイプのレベルでは、ファクト・テーブルに整数が格納されます。

ファクト・テーブル・クラスは、対応するプロパティを計算値として定義し、そのビットマップ・インデックスを定義します。以下はその例です。

```
/// Index for fact 4. Index DxBirthDateFxYear On DxBirthDateFxYear [ Type = bitmap ];

/// Dimension: DxBirthDateFxMonthYear<br/>
/// Source: BirthDate Property DxBirthDateFxMonthYear As %Integer [ details omitted ];
Property DxBirthDateFxYear As %Integer [ Calculated, SqlComputeCode = ... , SqlComputed ];
```

- キューブ間の <relationship> の場合、このキューブでリレーションシップの sourceProperty または sourceExpression が指定されているときは、ファクト・テーブルに他方のファクト・テーブルの対応する行の ID が格納されます。この ID は、他方のファクト・テーブルへのポインタです。

- 時間タイプ・ディメンジョンまたは年齢タイプ・ディメンジョンごとにフィールドが 1 つ存在し、このレコードの時刻ディメンジョンまたは年齢ディメンジョンの完全な値が格納されます (ただし、1 つ例外があります。このページで後述の"ファクト・テーブルにおけるソース・プロパティの再使用" を参照してください)。

  この値は %DeepSee.Datatype.dateTime という形式で、インデックスは作成されません。

このクラスには、NLP レベルまたは NLP メジャーのプロパティは含まれません。これらは別の方法で処理されます。

レベルに dependsOn 属性を指定した場合、ファクト・テーブルにレベルの組み合わせに対する追加のインデックスが格納されます。

```
Index DxPostalCodeViaHomeCityANDDx2642257510 On (DxPostalCodeViaHomeCity, Dx2642257510) [ Type = bitmap
 ];
```

<cube> に <index> 要素を定義した場合は、ファクト・テーブルに追加のカスタム・インデックスが格納されます。以下に例を示します。

```
Index %UserIndexName On (MxAge, DxGender) [ Type = bitmap ];
```

これらのカスタム・インデックスは、独自で使用するためのものです。システムはこのようなインデックスを使用しません。

フィールド名

以下の表に、システムでメジャー、レベル、およびディメンジョンの各フィールドの名前を決定する方法をまとめます (factName 属性が指定されない場合)。

項目およびシナリオ	ファクト・テーブルのフィールド名 (factName 属性でオーバーライドされていない場合)	例
ソース・プロパティに基づくメジャー	Mxprop_name。prop_name はプロパティの名前です。	MxAge
別のテーブルのソース・プロパティに基づくメジャー、via ドット構文を使用 (まれなケース)	Mxother_prop_nameViaprop_name。other_prop_name は、他のクラスのプロパティの名前です。ただし、生成されるフィールド名が長すぎる場合は、一意の番号が生成されます。	MxAgeViaOtherTable
ソース式に基づくメジャー	MxnnnnnnnnnT。nnnnnnnnn は整数、T はメジャー・タイプを示します (例えば、I は整数メジャーを表します)。	Mx1968652733I
ソース・プロパティに基づくデータ・レベル (範囲式は不使用)	Dxprop_name	DxGender

項目およびシナリオ	ファクト・テーブルのフィールド名 (factName 属性でオーバーライドされていない場合)	例
範囲式を使用するソース・プロパティに基づくデータ・レベル	Dxprop_nameRgnnnnnnnnnn。nnnnnnnnn は整数です。	DxAgeRg855025875
別のテーブルのソース・プロパティに基づくデータ・レベル、via ドット構文を使用	Dxother_prop_nameViaprop_name ただし、生成されるフィールド名が長すぎる場合は、一意の番号が生成されます。	DxPostalCodeViaHomeCity
ソース式に基づくデータ・レベル	Dxnnnnnnnnnn	Dx2163088627
時間タイプ・ディメンジョンまたは年齢タイプ・ディメンジョン	Dxdim_name。dim_name はディメンジョンの名前です(このフィールドはこのディメンジョンのレベルによって使用されます)。	DxBirthDate
時間タイプ・レベルまたは年齢タイプ・レベル	Dxdim_nameFxfunc_name。func_name はこのレベルの timeFunction 属性で指定されている名前です。	DxBirthDateFxYear
リレーションシップ	Rxgenerated_name。generated_name は、ソース・プロパティの名前またはソース式に基づいて生成された名前です。	RxMainCity

ファクト・テーブルにおけるソース・プロパティの再使用

キューブに、(sourceProperty を介して) 同じプロパティを使用する複数のメジャーが存在する場合、ファクト・テーブルにはこれらのメジャーの 1 つ (キューブ定義内の最後のメジャー) のフィールドのみが格納されます。例えば、キューブに以下のメジャー定義が含まれているとします。

```
<measure name="Age" sourceProperty="Age" aggregate="SUM" factName="AgeFact"/>
<measure name="Avg Age" sourceProperty="Age" aggregate="AVG" factName="AvgAgeFact"/>
```

これら 2 つのメジャーは複数のレコード間の集約方法のみが異なります。ファクト・テーブルでは、どのレコードについてもこれらのメジャーに同じ値が格納されることになります。このシナリオでは、エイリアスの競合を警告するために、コンパイル・エラーがスローされます。エイリアスの競合を防ぐために、関連するメジャーのすべての factNames を空白のままにするか、同じ factName 値を指定する必要があります。

キューブに同じプロパティを使用する複数のレベルが存在する場合、またはキューブに同じプロパティを使用する複数の年齢または時間ディメンジョンが存在する場合も同じロジックが適用されます。

指定のメジャー、レベル、またはディメンジョンに対して sourceExpression を使用し、%source.propertyname を介してプロパティにアクセスする場合、システムは、常にその値に対して別個のフィールドを生成します。

ディメンジョン・テーブル

Business Intelligence キューブ定義クラスをコンパイルする際に、レベルごとに生成されるテーブルについて説明します。

基本情報

キューブ定義クラスをコンパイルする際、年齢タイプおよび時間タイプのレベル以外の各レベルにもテーブルが生成されます。これらのテーブルは、ファクト・テーブルと同じパッケージ内に存在します。モデル・スキーマ内で他のディメンジョン・テーブル ID へのリンクを宣言する外部キー定義も、ディメンジョン・テーブル内に生成されます。

キューブを構築したり、インクリメンタルに更新すると、常にディメンジョン・テーブル (スター・テーブルとも呼ばれる) が更新されます。

レベルのディメンジョン・テーブルには、そのレベルのメンバごとに 1 つの行が含まれます。ディメンジョン・テーブルは、システムがベース・テーブルのレコードを処理する際に動的に作成されます。指定のレベルに対して、新たな一意の値が検出されるたびに、適切なディメンジョン・テーブルにその値が新しい行として追加されます。つまり、ディメンジョン・テーブルには必要な時点で行が自動的に追加され、特に操作する必要はありません。

ディメンジョン・テーブルの名前

キューブ定義で対応するレベルに `factName` 属性が指定されている場合、その値が該当するディメンジョン・テーブルの名前として使用されます。

この指定がない場合、ディメンジョン・テーブルの名前の形式は以下のとおりです。

```
Stargenerated_name
```

`generated_name` はファクト・テーブルの対応するフィールド名で、先頭に `Dx` は使用されません。例えば、ファクト・テーブル内で、`Home City` レベルのフィールド名が `DxPostalCodeViaHomeCity` であるとします。この場合、対応するディメンジョン・テーブルは `StarPostalCodeViaHomeCity` と名付けられます。

同じソース・プロパティまたはソース式を使用する 2 つのレベルは、同じ `factName` を持つ必要があります。

ディメンジョン・テーブル内の列

この行の列は以下のようになります。

- `ID` ー この行の ID。行の作成時に割り当てられます。
- 列が 1 つ存在し、このメンバのキーが格納されます。このフィールドの名前は、このレベルに対応するファクト・テーブルの列名のフィールドと同じになります。前のセクションを参照してください。
- このレベルのプロパティごとに 1 つの列が存在し、このメンバの実際のプロパティ値が格納されます。

  このフィールド名は `Dx` で始まり、前述したように、ソース・プロパティ名に基づくか、一意の番号として生成されます。

  `linkClass` と `linkProperty` を使用して、プロパティと、そのプロパティが属するレベルの両方を定義する場合、プロパティとレベルのフィールドの名前は同じものになります。このシナリオでは、システムによりプロパティのフィールド名の末尾に `_Link` が追加されます。
- このレベルの親レベルごとに 1 つの列が存在し、このメンバの親の ID が格納されます。

  このフィールドの名前は、親レベルに対応するファクト・テーブルの列名と同じになります。

レベルの定義方法に応じて、メンバ名は以下のように確認できます。

- 既定では、キーが名前として使用され、その名前は個別には格納されません。

- レベルに `isName="true"` で定義されているプロパティが含まれる場合、(1 つの例外を除いて) メンバ名はそのプロパティが含まれる列に格納されます。例外は、プロパティが `isReference="true"` でも定義されている場合です。この場合、フィールドは実行時に計算されます。