InterSystems IRIS Business Intelligence の開発者向けチュートリアル

Version 2024.1
2024-06-03

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

InterSystems IRIS Business Intelligence の開発者向けチュートリアル
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

Business Intelligence チュートリアル：はじめに

この Business Intelligence チュートリアルのほとんどの例は、Samples-BI サンプル (https://github.com/intersystems/Samples-BI) 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、"サンプルのダウンロード" を参照してください。

サンプルの 1 つに、**BI.Study.Patient** クラスと関連クラスが含まれています。このサンプルは、Business Intelligence モデルの基礎として使用するためのものです。最初は、データは格納されていません。**BI.Model** パッケージには、このチュートリアルで参考用に使用するサンプル・キューブ、サブジェクト領域、KPI、ピボット・テーブル、およびダッシュボードが含まれています。

このサンプルは、Business Intelligence の使用の開始点に柔軟性を持たせることを目的としています。このサンプルを使用して、必要に応じた量のデータを生成し、アーキテクトを使用して、生成したデータを調査する Business Intelligence モデルを作成します。その後、このモデルに基づいて Business Intelligence のピボット・テーブル、KPI、およびダッシュボードを作成できます。このサンプルは、Business Intelligence の主な機能の使用だけでなく一般に想定される現実的な多数のシナリオのテストでも可能なほど十分な複合性を備えています。このチュートリアルには、このサンプルを使用する実践練習があります。

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、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、"InterSystems SQL リファレンス" の "予約語" のセクション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。

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 準備

使用するツールの大部分は、管理ポータルにあります。

ログオンの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 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、サンプルをロードした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名をクリックして、使用可能なネームスペースのリストを開きます。
   b. リストで、サンプルをロードしたネームスペースの名前をクリックします。

1.2 データの再生成

チュートリアルで使用する一連のデータは、Samples-BI サンプルで当初提供されるデータよりも大きく多少複雑になります。

このチュートリアル用のデータを生成する手順は以下のとおりです。

1. ターミナルで、サンプルをインストールしたネームスペースに切り替えます。以下に例を示します。

   ObjectScript

   ```
   set $namespace="SAMPLES"
   ```

2. 以下のコマンドを実行します。

   ObjectScript

   ```
   do ##class(BI.Populate).GenerateData(10000,25,"ADETR")
   ```

   このクラス・メソッドによって、10,000 人の患者が生成されます (医者 1 人あたり 25 人の患者)。"ADETR" 文字列は、サンプルにアレルギーのデータ (A)、診断データ (D)、受診データ (E)、詳細 (T)、および都市の降雨データ (R) が含まれることを意味します。

3. データを入力したテーブルに対して実行される SQL クエリを使用するため、これらのテーブルに対してテーブルのチューニング機能を実行することをお勧めします。

   a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

   b. **[システムエクスプローラ]→[SQL]** をクリックします。

   c. **[クエリ実行]** タブをクリックします。

   d. **[アクション]**、**[スキーマのすべてのテーブルをチューニング]** の順にクリックします。

      スキーマを選択してアクションを確認するダイアログ・ボックスが表示されます。

   e. **スキーマ** で、BI_Study スキーマを選択します。

   f. **[完了]** をクリックします。

   g. **[完了]** をクリックします。

   バックグラウンドでテーブルのチューニング機能が実行されます。

2

Business Intelligence チュートリアル：キューブ要素の概要

独自の Business Intelligence キューブを作成する前に、サンプル・キューブを検証し、使用方法を確認することは有益です。

2.1 Patients キューブへのアクセス

1. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
2. **[ホーム]**→[Analytics]→**[アナライザ]** に移動します。
3. `Patients` をクリックします。
4. [OK] をクリックします。

アナライザ・ページには、以下のように 3 つの主要領域があります。

- 左側の [モデル・コンテンツ] 領域には、選択したキューブのコンテンツがリスト表示されます。フォルダを展開して、項目をピボット・ビルダ領域にドラッグ・アンド・ドロップすることができます。
- 右上のピボット・ビルダ領域には、ピボット・テーブルの作成に使用するオプションがあります。この領域は、**[行]**、**[列]**、**[メジャー]**、および **[フィルタ]** のボックスで構成されます。
- 右下のピボット・プレビュー領域には、ダッシュボードでの表示とほぼ同様にピボット・テーブルが表示されます。

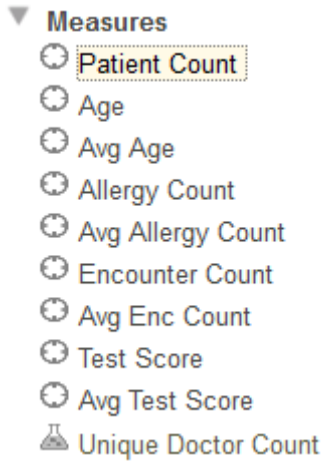
2.2 [モデル・コンテンツ] 領域の案内

[モデル・コンテンツ] 領域には、現在表示しているキューブの内容がリストされます。このチュートリアルで、ドロップダウン・リストから **[ディメンジョン]** を選択します。このオプションでは、指定されたキューブのメジャーとディメンジョンが表示されます。

上部のセクションには名前付きセットが表示されますが、このチュートリアルではこれらは使用されません。領域の下部は以下のセクションで構成されます。

2.2.1 メジャー

[**メジャー**] セクションには、キューブ内のすべてのメジャーが表示されます。以下はその例です。



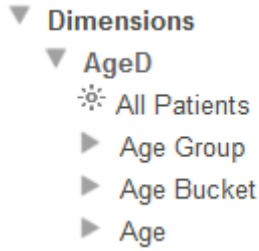
2 種類のメジャーが存在する場合があり、それらは別々のアイコンで示されます。

	**標準メジャー**
	**計算メジャー (他のメジャーに換算して定義します)**

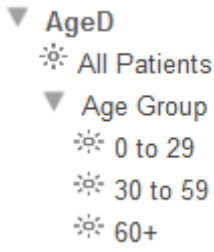
2.2.2 ディメンジョン

[**ディメンジョン**] セクションには、ディメンジョンとそこに含まれているレベル、メンバ、およびプロパティがリストされます (また、メジャー以外の計算メンバおよびセットも含まれますが、このページではこれらの項目については説明しません)。

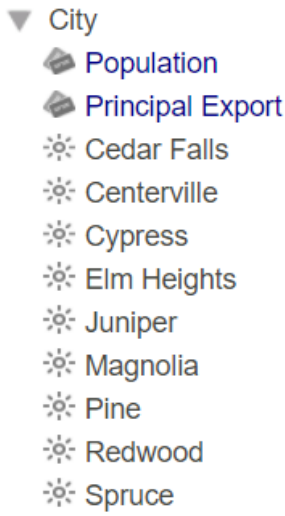
ディメンジョン名の横にある三角形をクリックして、これを展開します。ディメンジョンには、1 つ以上のレベルがあります。また、All メンバとして知られている特殊なメンバが含まれる場合もあります。以下の例では、`AgeD` ディメンジョンに、`All Patients` という名前の All メンバがあります。また、`Age Group`、`Age Bucket`、`Age` というレベルもあります。



レベルを展開すると、そのレベルのメンバが表示されます。以下はその例です。



レベルにプロパティも含まれている場合、それらのプロパティはリストの先頭に異なるアイコンと共に青い文字で表示されます。例えば、City レベルに Population プロパティと Principal Export プロパティが含まれているとします。



2.3 単純なピボット・テーブルの作成

ここでは、一般的な方法でレベルとメジャーを使用する単純なピボット・テーブルを作成します。このセクションの目標は、レベルとメジャーの機能を確認し、メンバがどのようなものであるかを理解することです。

実際に表示される数は、ここでの表示と異なる場合もあります。

1. [モデル・コンテンツ] ペインで、[DiagD] ディメンジョンを展開します。
2. [Diagnoses] を [行] にドラッグ・アンド・ドロップします。

   または、[Diagnoses] をダブルクリックします。

   以下のように表示されます。

Diagnoses	All
None	8,432
asthma	691
CHD	287
diabetes	531
osteoporosis	217

3. [Patient Count] を [**メジャー**] にドラッグ・アンド・ドロップします。

   または、[Patient Count] をダブルクリックします。
4. [Avg Age] を [**メジャー**] にドラッグ・アンド・ドロップします。

   または、[Avg Age] をダブルクリックします。

   以下のように表示されます。

Diagnoses	Patient Count	Avg Age
None	8,432	33.11
asthma	691	34.02
CHD	287	68.32
diabetes	531	57.66
osteoporosis	217	79.58

5. [**保存**] をクリックします。

   ピボット・テーブルの名前を指定できるダイアログ・ボックスが表示されます。

6. [**フォルダ**] に、Test と入力します。

7. [**ピボット名**] に Patients by Diagnosis (Patients Cube) と入力します。

8. [OK] をクリックします。

この操作によって、表示コンテキストと、データそのものではなく、データを取得する基礎となるクエリが保存されます。

表示内容を正しく理解することは重要です。以下の点に注意してください。

- ベース・テーブルは Patients で、これはすべてのメジャーが患者に関するデータを集計していることを意味します。
- このピボット・テーブルのヘッダ行以外の行はそれぞれ、Diagnoses ディメンジョンの 1 メンバのデータを表示します。

  いずれの場合も、個々のメンバは、ファクト・テーブル内のレコード・セットに対応します。(ほとんどの場合、ファクト・テーブルの各レコードが、ベース・テーブルの 1 つのレコードに対応します。)

  したがって、このピボット・テーブルの各行には、特定の診断を受けた一連の患者のデータが表示されます。

  これ以外のレイアウトも可能ですが (後述のとおり)、いずれの場合もピボット・テーブルのデータ・セルはすべてファクト・テーブルのレコード・セットに関連付けられます。
- 一般的なピボット・テーブルでは、各データ・セルにメジャーの集約値が表示されます。これはそのデータ・セルで使用されるすべてのレコードを集約したものです。
- 特定のデータ・セルの内容を理解するには、そのセルに対応するラベルの情報を参照します。例えば、asthma 行の Patient Count 列にあるセルについて考えてみます。このセルには、喘息を持つ患者の総数が表示されます。

  同様に、この行の Avg Age 列について考えてみます。このセルには、喘息を持つ患者の平均年齢が表示されます。
- 集約の実行方法は、メジャーによって異なる場合があります。Patient Count の場合は、数値が合計されます。Avg Age の場合は、数値の平均値が計算されます。その他の集約も可能です。

2.4 メジャーおよびレベル

ここでは、メジャーおよびレベルについてもう少し詳しく見ていきます。

1. [**新規**] をクリックします。

2. [Patient Count] および [Avg Age] を [**メジャー**] 領域にドラッグ・アンド・ドロップします。

   これにより、以下のように表示されます。

	Patient Count	Avg Age
	10,000	35.86

この単純なピボット・テーブルでは、ベース・クラスのすべてのレコードにわたる、これらの各メジャーの集約値が表示されます。10000 人の患者が存在し、その平均年齢は 35.93 歳です (この例の場合)。

3. これらの値を、ソース・テーブルから直接取得された値と比較します。そのためには、以下の操作を実行します。
   a. 別のブラウザ・タブまたはウィンドウで、前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
   b. **[ホーム]**→**[システムエクスプローラ]**→[SQL] に移動します。
   c. **[クエリ実行]** タブをクリックします。
   d. 以下のクエリを実行します。

      SQL

      ```
      select count(*) as "count",avg(age) as avgage from bi_study.patient
      ```

   同じ数値が表示されます。以下はその例です。

count	avgage
10000	35.8574

   Tipヒン このブラウザ・タブまたはウィンドウは、後で使用できるように開いたままにしておきます。
   ト

4. アナライザで、前出のピボット・テーブルを以下のように変更します。
   a. 左側の [GenD] を展開します。
   b. [Gender] を [行] 領域にドラッグ・アンド・ドロップします。これにより、表示は以下のようになります。

Gender	Patient Count	Avg Age
Female	5,072	37.23
Male	4,928	34.44

5. これらの値を、ソース・テーブルから取得された集約値と比較します。そのためには、以下の操作を実行します。
   a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
   b. **[ホーム]**→**[システムエクスプローラ]**→[SQL] に移動します。
   c. **[クエリ実行]** タブをクリックします。
   d. **[履歴を表示]** をクリックします。
   e. 以前実行したクエリをクリックします。
   f. クエリの最後に以下を追加して、クエリを再実行します。

      ```
      group by gender
      ```

      ピボット・テーブルの表示と同じ数値が表示されます。以下はその例です。

count	avgage
5072	37.23107255520504732
4928	34.44358766233766234

6. 最後の例では、アナライザで以下の変更を行います。
   a. [行] ペインの [X] ボタンをクリックします。これによって、行の定義がクリアされます。
   b. [ProfD] および [Profession] を展開します。
   c. [Electrician] を [行] にドラッグ・アンド・ドロップします。

   表示は以下のようになります。

Electrician	Patient Count	Avg Age
Electrician	191	39.22

7. これらの値を、ソース・テーブルの値と比較します。そのためには、以下の操作を実行します。
   a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
   b. [ホーム]→[システムエクスプローラ]→[SQL] に移動します。
   c. [クエリ実行] タブをクリックします。
   d. 以下のクエリを実行します。

```
SQL

select count(*) as "count",avg(age) as avgage from bi_study.patient join bi_study.patientdetails

on bi_study.patient.patientid = bi_study.patientdetails.patientid
where bi_study.patientdetails.profession->profession='Electrician'
```

   同じ数値が表示されます。以下はその例です。

count	avgage
191	39.21989528795811518

2.5 ディメンジョンおよびレベル

多くのシナリオで、ディメンジョンとレベルは互換的に使用できます。ここでは、これらを比較して異なる点を確認します。

1. アナライザで、[新規] をクリックします。
2. [GenD] 定義を [行] 領域にドラッグ・アンド・ドロップします。以下のような表示になります。

Gender	All
Female	5,072
Male	4,928

   表示されるメジャーは Count です。これは患者の数です。

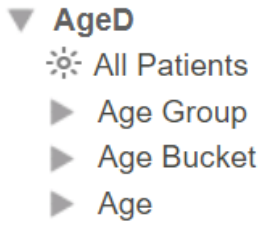
3. [新規] をクリックします。
4. [GenD] ディメンジョンを展開します。[Gender] レベルを [行] 領域にドラッグ・アンド・ドロップします。以下のような表示になります。

Gender	All
Female	5,072
Male	4,928

この場合、同じ結果が表示されます。

Patients サンプルでは、ディメンジョンの名前は省略形で、末尾に D の文字が付きます。レベルの名前は、そのレベルを含むディメンジョンの名前と同じになることはありません。この名前付け規約は必須ではなく、レベルとそのレベルを含むディメンジョンに同じ名前を使用することもできます。

5. [新規] をクリックします。
6. [AgeD] ディメンジョンを展開します。左領域に以下のように表示されます。



このディメンジョンの定義は、以下の 2 つの点で GenD ディメンジョンと異なっています。

- AgeD では All Patients と呼ばれる特別なメンバ、つまり All メンバが定義されています。All メンバは、ベース・クラスの全レコードを参照します。
- AgeD では、Age Group、Age Bucket、および Age という複数のレベルが定義されています。

7. [AgeD] ディメンジョンを [行] 領域にドラッグ・アンド・ドロップします。以下のような表示になります。

Age Group	All
0 to 29	4,193
30 to 59	4,179
60+	1,628

行 (または列) として使用するディメンジョンをドラッグ・アンド・ドロップする際、そのディメンジョンに定義されている最初のレベルのすべてのメンバが表示されます。この場合、最初のレベルは Age Group です。

2.6 All メンバ

All メンバは、ベース・クラスの全レコードを参照します。どのディメンジョンも All メンバを持つことができますが、Patients キューブで All メンバがあるディメンジョンは 1 つのみです。

ここでは、All メンバを使用する方法を示します。

1. [新規] をクリックします。

2. [AgeD] ディメンジョンを展開します。
3. [Age Group] を [行] にドラッグ・アンド・ドロップします。
4. メジャー [Patient Count]、[Avg Age] および [Avg Test Score] を [**メジャー**] 領域にドラッグ・アンド・ドロップします。以下のように表示されます。

Age Group	Patient Count	Avg Age	Avg Test Score
0 to 29	4,193	14.28	74.37
30 to 59	4,179	43.33	74.77
60+	1,628	72.26	74.53

5. ピボット・オプション・ボタン  をクリックします。
6. [**行オプション**] 領域で [**集計**] チェック・ボックスにチェックを付け、ドロップダウン・リストで [**合計**] が選択されたままにして [OK] をクリックします。

   これで、以下のように Total 行が表示されます。

Age Group	Patient Count	Avg Age	Avg Test Score
0 to 29	4,193	14.28	74.37
30 to 59	4,179	43.33	74.77
60+	1,628	72.26	74.53
Total	**10,000**	**129.86**	**223.67**

   Total 値は、Patient Count には適していますが、他のメジャーには適しません。Avg Age および Avg Test Score には、合計ではなく平均値の表示が適しています。

7. ピボット・オプション・ボタン  を再度クリックします。
8. [**行オプション**] 領域で [**集計**] チェック・ボックスのチェックを外して [OK] をクリックします。
9. [Age Group] の下で、[All Patients] を [行] にドラッグ・アンド・ドロップします。これで、All Patients は Age Group レベルのメンバの後に表示されます。

Age Group	Patient Count	Avg Age	Avg Test Score
0 to 29	4,193	14.28	74.37
30 to 59	4,179	43.33	74.77
60+	1,628	72.26	74.53
All Patients	10,000	35.86	74.56

   All Patients 行は、Total 行より有益な集計行です。Patient Count、Avg Age および Avg Test Score のメジャーが、それぞれすべての患者にわたって集約され、表示されています。

   注釈　Avg Age および Avg Test Score には、ピボット・テーブルに表示される値の平均値を示すほうがよい場合もあります。例えば、Avg Age の場合、この集計行ではすべての患者の年齢を加算して 10000 で除算します。ここに示した 3 つのメンバの Avg Age の値を加算して、その値を 3 で除算することが望まれる場合もあります。All メンバはこれには役立ちません。代わりに、計算メンバ (このチュートリアルで後述) を作成します。

10. [行] ペインの [X] ボタンをクリックします。これによって、行の定義がクリアされます。
11. [DiagD] ディメンジョンを展開します。
12. [Diagnoses] を [行] ペインにドラッグ・アンド・ドロップします。
13. [AgeD] の [All Patients] メンバを [行] の [Diagnoses] の下にドラッグ・アンド・ドロップします。これにより、表示は以下のようになります。

Diagnoses	Patient Count	Avg Age	Avg Test Score
None	8,432	33.11	74.55
asthma	691	34.02	74.60
CHD	287	68.32	74.33
diabetes	531	57.66	75.08
osteoporosis	217	79.58	74.84
All Patients	10,000	35.86	74.56

上の図からもわかるように、総称的な名前の All Patient メンバは、Age 以外の、このメンバが偶然に定義されているディメンジョンと共に使用することができます。

2.7 階層

ディメンジョンには 1 つ以上の階層が格納され、各階層には複数のレベルを格納できます。[モデル・コンテンツ] 領域には、階層によって指定された順序に従ってレベルがリストされますが、(スペースを節約するため) このキューブの階層名は表示されません。

ユーザは階層を使用して、下位レベルにドリルダウンできます。ここでは、この機能について説明します。

1. [新規] をクリックします。
2. [モデル・コンテンツ] ペインで、[BirthD] ディメンジョンを展開します。[Decade] を [行] にドラッグ・アンド・ドロップします。

   または、[Decade] をダブルクリックします。

   以下のように表示されます。

Decade	All
1920s	86
1930s	196
1940s	540
1950s	747
1960s	1,061
1970s	1,464
1980s	1,554
1990s	1,384
2000s	1,383

表示されるメジャーは Count です。これは患者の数です。

3. [1950s] 行 (または比較的患者数が多いその他の行) をダブルクリックします。<< 記号の右側の任意の場所をクリックします。

   これで、その年代に生まれた患者が年別 (階層内の最下位の直上レベル) にグループ化されて以下のように表示されます。

	All
◀ 1950	63
◀ 1951	71
◀ 1952	64
◀ 1953	86
◀ 1954	79
◀ 1955	78
◀ 1956	90
◀ 1957	77
◀ 1958	80

   このダブルクリック操作は、アナライザ内だけではなく、ダッシュボードに表示されたピボット・テーブル内で使用できます。

4. 行を再度ダブルクリックします。その年に生まれた患者が、以下のように年および四半期別にグループ化されて表示されます。

	All
◀ Q1 1950	16
◀ Q2 1950	18
◀ Q3 1950	14
◀ Q4 1950	15

5. 行を再度ダブルクリックします。その年および四半期に生まれた患者が、以下のように年および月別にグループ化されて表示されます。

	All
◀ Jan-1950	2
◀ Feb-1950	8
◀ Mar-1950	6

6. 行を再度ダブルクリックします。その年および月に生まれた患者が、以下のように実際の日付別にグループ化されて表示されます。

	All
◀ Feb 7 1950	1
◀ Feb 12 1950	1
◀ Feb 15 1950	1
◀ Feb 17 1950	1
◀ Feb 22 1950	1
◀ Feb 24 1950	1
◀ Feb 26 1950	1
◀ Feb 28 1950	1

7. << 記号を繰り返しクリックすると、ピボット・テーブルの元の状態に戻ります。

2.8 プロパティ

レベルにはプロパティを含めることができ、このプロパティはピボット・テーブルに表示できます。

1. **[新規]** をクリックします。
2. [モデル・コンテンツ] ペインで、`[HomeD]` ディメンジョンを展開します。
3. `[City]` レベルを展開し、次に `[City]` を **[行]** にドラッグ・アンド・ドロップします。

   以下のように表示されます。

City	All
Cedar Falls	1,090
Centerville	1,089
Cypress	1,111
Elm Heights	1,163
Juniper	1,108
Magnolia	1,113
Pine	1,097
Redwood	1,100
Spruce	1,129

   表示されるメジャーは Count です。これは患者の数です。
4. `[Population]` を **[列]** にドラッグ・アンド・ドロップします。
5. `[Principal Export]` を **[列]** にドラッグ・アンド・ドロップします。

   以下のように表示されます。

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

6. [行] ペインの [X] ボタンをクリックします。
7. [ZIP] を [行] にドラッグ・アンド・ドロップします。

   以下のように表示されます。

ZIP	Population	Principal Export
32006		
32007		
34577		
36711		
38928		

   これらのプロパティにこのレベルの値はありません。

ピボット・テーブルでは、プロパティとメジャーはいくつかの点で異なります。

- プロパティでは文字列値を使用できます。
- プロパティには、それが定義されたレベルの値のみが存在します。

キューブの定義方法によって、プロパティは、属するレベルの並べ替えおよびメンバ名に影響を与える場合もあります。これに関する例は、このチュートリアルの後続部分で紹介します。

2.9 リスト

ここでは、選択したセル (複数の場合もあり) の最下位レベルのデータから選択されたレコードが表示されるリストについて説明します。この機能を確認するため、最初にごく少数のレコードのみを使用するピボット・テーブルを作成します。そのリストを表示すると、開始ポイントのセルの集約値と容易に比較できます。

1. [新規] をクリックします。
2. [Patient Count] および [Avg Test Score] を [メジャー] にドラッグ・アンド・ドロップします。
3. [モデル・コンテンツ] ペインで、[AgeD] ディメンジョンを展開します。
4. [Age] レベルを展開します。

5. メンバ [0] を [列] にドラッグ・アンド・ドロップします。このメンバは、1 歳未満のすべての患者を参照します。

   左側にあるアイコンでなくメンバ名をクリックする必要があります。

   以下のように表示されます。

	0	
	Patient Count	Avg Test Score
	130	73.93

6. メンバ [1] を、[列] のメンバ [0] の下にドラッグ・アンド・ドロップします。

   以下のように表示されます。

	0		1	
	Patient Count	Avg Test Score	Patient Count	Avg Test Score
	130	73.93	136	74.18

7. [BirthTD] ディメンジョンを展開します。
8. [Birth Time] レベルを [行] にドラッグ・アンド・ドロップします。

   以下のように表示されます。

	0		1	
Birth Time	Patient Count	Avg Test Score	Patient Count	Avg Test Score
12am	2	91.50	5	80.80
1am	9	75.13	9	80.00
2am	7	77.29	5	84.25
3am	6	69.00	7	86.00
4am	3	89.67	12	79.44
5am	6	77.17	6	56.00
6am	4	69.67	2	67.00
7am	9	72.86	3	61.50

9. セルをクリックします。例えば、[0] の下の [12am] 行の [Patient Count] セルをクリックします。
10. リストを表示ボタン  をクリックします。

    システムによって選択コンテキストが考慮されます。この例では、1 歳未満で深夜 0 時から午前 1 時の間に生まれた患者です。そのソース・データに対する SQL クエリが実行されます。このクエリには、以下のように、これらの患者に対して選択されたフィールドが含まれています。

#	PatientID	Age	Gender	Home City	Test Score
1	SUBJ_101178	0	Female	Centerville	84
2	SUBJ_108656	0	Female	Cypress	99
3	SUBJ_107655	0	Male	Pine	54
4	SUBJ_108786	0	Male	Redwood	68
5	SUBJ_109119	0	Male	Centerville	91
6	SUBJ_109279	0	Male	Pine	67

11. 表示された行をカウントします。このカウントは、開始ポイントの行の `Patient Count` 値と等しくなります。
12. テーブル表示ボタン をクリックして、ピボット・テーブルの表示を元の状態に戻します。

    既定では、Patients キューブは `Patient details` と呼ばれるリストを使用します。このリストには、ご覧のように `PatientID`、`Age`、`Gender` などのフィールドが含まれています。他のリストも表示できます。
13. ピボット・オプション・ボタン をクリックして、ピボット・テーブルのオプションを表示します。

    ダイアログ・ボックスが表示されます。
14. **[詳細リスト]** ドロップダウン・リストで `[Doctor details]` をクリックしてから、[OK] をクリックします。

    `Doctor details` リストには、選択した患者の一次診療医に関する情報が表示されます。
15. 以前クリックしたセルと同じセルをクリックして、リストを表示ボタン をクリックします。

    これで以下のように表示されます。

#	PatientID	Doctor Last Name	Doctor First Name	Doctor Type	Doctor Group
1	SUBJ_100792	Pape	Janice	Pediatrician	III
2	SUBJ_103836	Orlin	Elmo	Cardiologist	III
3	SUBJ_104964	Page	Greta	Pediatrician	II
4	SUBJ_109487	Edison	Violet	Pediatrician	II

2.10 フィルタおよびメンバ

通常のピボット・テーブルでは、このページで前述したように、メンバを行、列、またはその両方として使用します。メンバのもう 1 つの一般的な使用法は、データのフィルタ処理を可能にすることです。

1. アナライザで、**[新規]** をクリックします。
2. `[ColorD]` および `[Favorite Color]` を展開します。
3. `[Favorite Color]` を **[行]** にドラッグ・アンド・ドロップします。

   以下のように表示されます。

Favorite Color	All
None	2,401
Blue	1,220
Green	1,282
Orange	1,279
Purple	1,252
Red	1,262
Yellow	1,304

このピボット・テーブルには、Favorite Color のメンバが行として表示されます。表示されるメジャーは Count です。これは患者の数です。

4. [Red] を [フィルタ] にドラッグ・アンド・ドロップします。

   これでアナライザには Favorite Color レベルのメンバが 1 つだけ表示されます。以下のように表示されます。

Favorite Color	All
Red	1,262

   患者の合計数をメモしておきます。

5. [行] ボックスの [X] ボタンをクリックします。
6. [AgeD] を展開します。
7. [Age Group] を [行] にドラッグ・アンド・ドロップします。

   アナライザの表示は以下のようになります。

Age Group	All
0 to 29	508
30 to 59	552
60+	202

8. ピボット・オプション・ボタン  をクリックします。
9. [行オプション] 領域で [集計] チェック・ボックスにチェックを付け、ドロップダウン・リストで [合計] が選択されたままにして [OK] をクリックします。

   アナライザの表示は以下のようになります。

Age Group	All
0 to 29	508
30 to 59	552
60+	202
Total	1,262

Total 行に、この列の数値の合計が表示されます。この合計数は、前述の数値と同じであることがわかります。

ピボット・テーブルが何を行 (または列) に使用するかに関係なく、任意のメンバをそのピボット・テーブルのフィルタとして使用できます。いずれの場合も、指定のメンバに関連付けられているレコードのみが取得されます。

複数のメンバをフィルタとして使用できます。また、フィルタを組み合わせることもできます。詳細は、"アナライザの使用法" を参照してください。

2.11 フィルタおよび検索可能メジャー

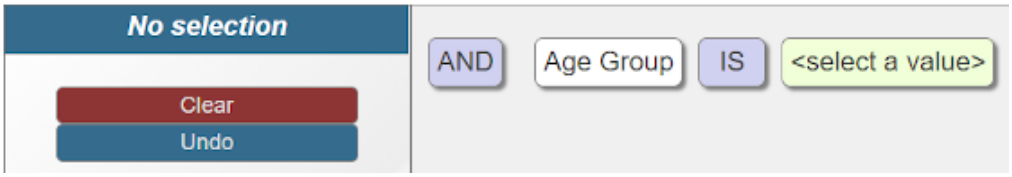
InterSystems IRIS Business Intelligence では、検索可能なメジャーを定義できます。このメジャーを使用すると、ソース・レコード自体のレベルで値を評価するフィルタを適用できます。

1. **[新規]** をクリックします。

   すべての患者の数が表示されます。

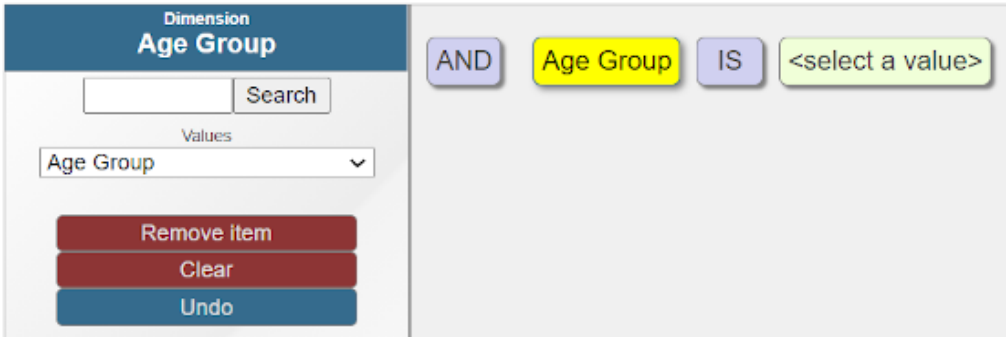
	All
Count	10,000

2. **[フィルタ]** ボックスで詳細オプションのボタン  をクリックします。
3. **[条件を追加]** をクリックします。これで、以下のように表示されます。

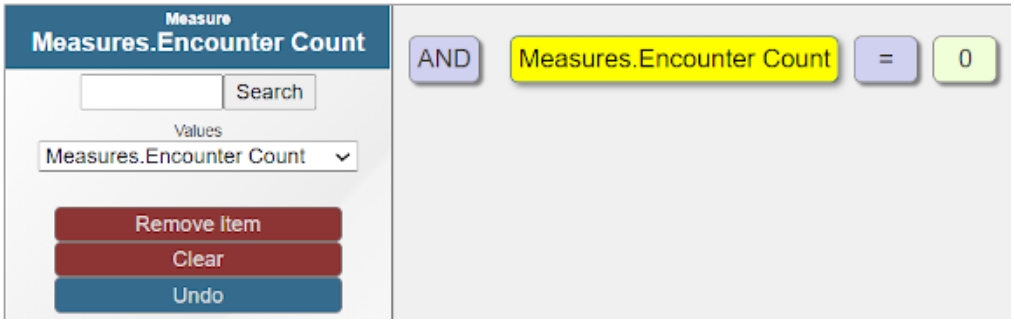


4. `[Age Group]` をクリックします。これによって、式のこの部分を編集できます。

   このダイアログ・ボックスは以下のようになります。

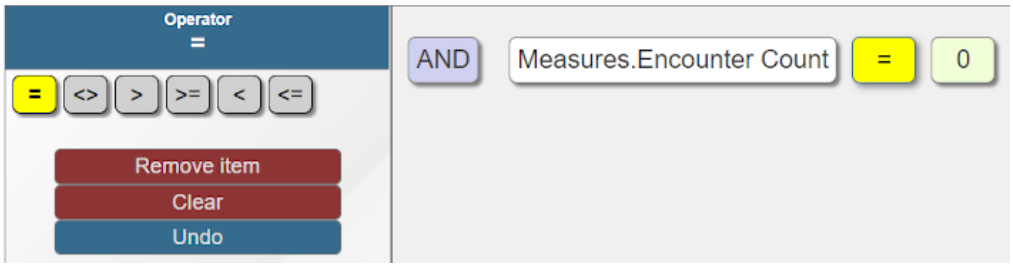


5. 左側のドロップダウン・リストをクリックしてから、下にスクロールして、`[Measures.Encounter Count]` をクリックします。その後すぐに、式が以下のように更新されます。以下はその例です。

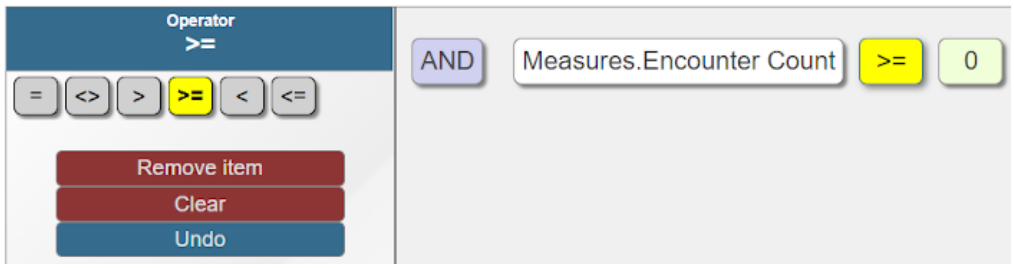


6. [=] 演算子をクリックします。これによって、式のこの部分を編集できます。

   このダイアログ・ボックスは以下のようになります。

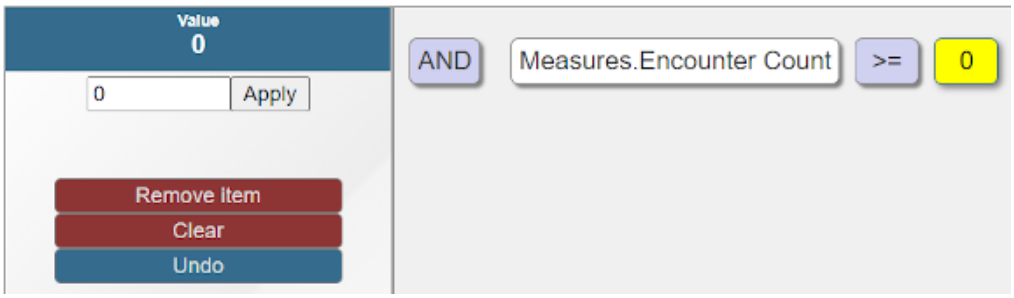


7. [>=] 演算子をクリックします。その後すぐに、式が以下のように更新されます。以下はその例です。



8. [0] をクリックします。これによって、式のこの部分を編集できます。

   このダイアログ・ボックスは以下のようになります。



9. フィールドに 10 を入力して、[適用] をクリックします。
10. [OK] をクリックします。

    これで、10 回以上受診したすべての患者の合計数が表示されます。

	All
Count	7,944

ここで、ピボット・テーブルにレベルを追加した場合の影響を見てみましょう。

11. [モデル・コンテンツ] ペインで、[AgeD] ディメンジョンを展開します。
12. [Age Group] を [行] にドラッグ・アンド・ドロップします。

    以下のように表示されます。

Age Group	
0 to 29	3,217
30 to 59	3,353
60+	1,374

13. ピボット・オプション・ボタン  をクリックします。
14. [**行オプション**] 領域で [**集計**] チェック・ボックスにチェックを付け、ドロップダウン・リストで [**合計**] が選択されたままにして [OK] をクリックします。
15. [OK] をクリックします。

    アナライザの表示は以下のようになります。

Age Group	
0 to 29	3,217
30 to 59	3,353
60+	1,374
Total	7,944

Total 行に、この列の数値の合計が表示されます。この合計数は、前述の数値と同じであることがわかります。

3

Business Intelligence Tutorial：キューブの作成

ここでは、単純な Business Intelligence キューブを作成します。

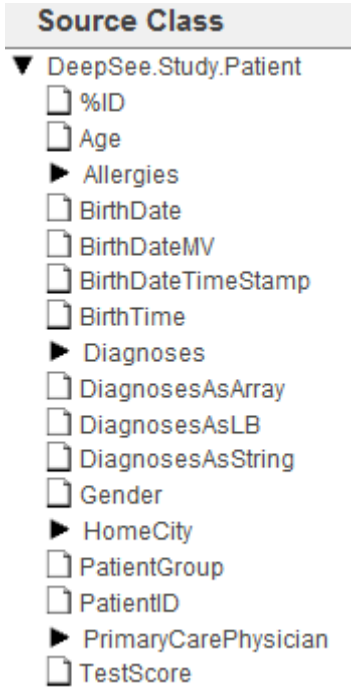
3.1 基本的なキューブの作成

1. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
2. **[ホーム]**→[Analytics]→**[アーキテクト]** に移動します。
3. **[新規]** をクリックします。ダイアログ・ボックスが表示されます。
4. このダイアログ・ボックスで、以下を指定します。
   - **[定義タイプ：キューブ]** ― これを選択します。
   - **[キューブ名]** ― `Tutorial`
   - **[ソースクラス]** ― **[参照]** ボタンをクリックし、`[BI.Study.Patient]` を選択して [OK] をクリックします。
   - **[キューブのクラス名]** ― `Tutorial.Cube`
5. [OK] をクリックします。

   キューブ・クラスが生成されます。これはスタジオでも表示および変更できます。
6. 中央の領域にある太字の最上行（`Tutorial` のラベル）をクリックします。これによって、キューブが選択され、右側で詳細を編集できます。
7. [詳細] ペインで、**[ヌル置換文字列]** に `None` と入力します。
8. **[保存]** をクリックして、[OK] をクリックします。

   キューブ・クラスが更新されます。

左領域にあるクラス・ビューワの表示が以下のようになります。



重要　クラス・ビューワは、ベース・クラスのクラス・プロパティ (リレーションシップ・プロパティ以外) の有益なビューを提供します。このビューを使用すると、これらのプロパティに基づく InterSystems IRIS Business Intelligence 要素を非常に簡単に作成できます。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。このチュートリアルでは、これらの点について後ほど説明します。

3.2 レベルおよびメジャーの追加

ここでは、Tutorial キューブにレベルとメジャーをいくつか追加します。

1. クラス・ビューワ (左領域) からモデル・ビューワの **[メジャー]** ヘッダ (中央の領域) に以下の項目をドラッグ・アンド・ドロップします。
   - Age
   - TestScore

   これにより、Age および TestScore という名前のメジャーが、それぞれの名前を持つクラス・プロパティに基づいて作成されます。
2. TestScore メジャーを以下のように変更します。
   a. **[メジャー]** ヘッダの下のメジャー名をクリックします。
   b. [詳細] ペイン (右領域) で、**[名前]** を Test Score に変更します。
   c. **[検索可能]** をクリックします。
3. 以下の手順で、Avg Age メジャーを作成します。

a. 再度、クラス・ビューワからモデル・ビューワの [**メジャー**] ヘッダに [Age] プロパティをドラッグ・アンド・ドロップします。

これによって、Age1 という名前の新しいメジャーが作成されます。

b. モデル・ビューワでメジャー名をクリックして、[詳細] ペインで以下の詳細を編集します。

- [**名前**] に Avg Age と指定します。
- [**集計**] に [AVG] を選択します。
- [**書式文字列**] に #.## を指定します。

4. 以下の手順で、Avg Test Score メジャーを作成します。

a. 再度、クラス・ビューワからモデル・ビューワの [**メジャー**] ヘッダに TestScore プロパティをドラッグ・アンド・ドロップします。これによって、TestScore1 という名前の新しいメジャーが作成されます。

b. モデル・ビューワでメジャー名をクリックして、[詳細] ペインで以下の詳細を編集します。

- [**名前**] に Avg Test Score と指定します。
- [**集計**] に [AVG] を選択します。
- [**書式文字列**] に #.## を指定します。

これで、4 つのメジャーが作成されました。

▼ Measures		
Age	measure	SUM Age
Avg Age	measure	AVG Age
Test Score	measure	SUM TestScore
Avg Test Score	measure	AVG TestScore

5. [**保存**] をクリックして、[OK] をクリックします。

キューブ・クラスが更新されます。

6. Age プロパティに基づいて、以下のようにディメンジョン、階層、およびレベルを追加します。

a. [Age] プロパティを [**ディメンジョン**] ヘッダにドラッグ・アンド・ドロップします。

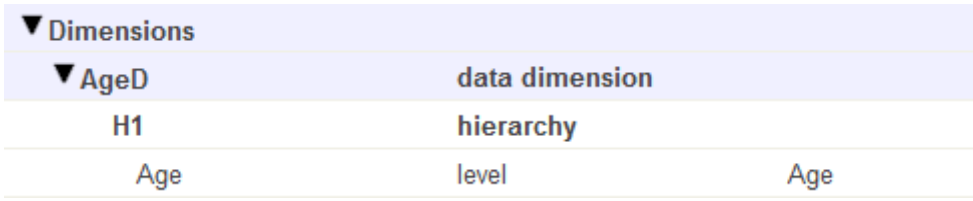
アーキテクトによって、ディメンジョン、階層、およびレベルが即座に作成され、モデル・ビューワの表示が以下のようになります。

▼ Dimensions			
▼ Age	data dimension		✖
H1	hierarchy		✖
Age	level 1	Age	✖

b. 最初の [Age] 項目をクリックします。この項目には [data**ディメンジョン**] のラベルが付けられています。

c. 右領域で、[**名前**] を編集して AgeD にします。

モデル・ビューワの表示は以下のようになります。



Business Intelligence の使用計画によっては、ディメンジョン名が表示されない場合もあります。このチュートリアルでは、Patients サンプルで使用される規約に従います。ここでは、ピボット・テーブルでディメンジョンを行または列に使用しないことを前提にしています (この代わりにレベルを行または列として使用します)。

d. オプション [**このディメンジョンの All レベルを有効にする**] を選択します。

e. [**All メンバに対するキャプション**] を編集して `All Patients` にします。

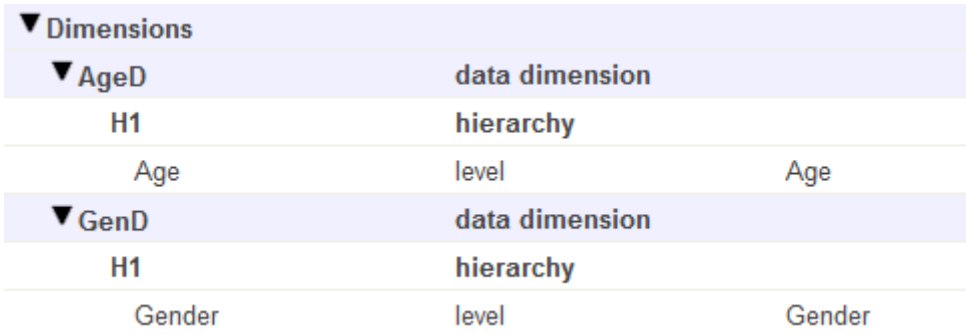
f. [**All メンバの表示名**] を編集して `All Patients` にします。

7. 以前と同じ方法でキューブ定義を保存します。

8. `Gender` プロパティに基づいて、ディメンジョン、階層、およびレベルを追加します。前述の手順を繰り返しますが、以下の点が異なります。

   - `[Gender]` プロパティをドラッグ・アンド・ドロップします。
   - ディメンジョンの名前を `GenD` に変更します。

   モデル・ビューワの表示は以下のようになります。



9. `HomeCity` プロパティに基づいて、ディメンジョン、階層、およびレベルを追加します。前述の手順を繰り返しますが、以下の点が異なります。

   - `[HomeCity]` プロパティを展開して、このフォルダ内の `[Name]` プロパティを [**ディメンジョン**] にドラッグ・アンド・ドロップします。
   - ディメンジョンの名前を `HomeD` に変更します。
   - レベルの名前を `City` に変更します。

   この新しいディメンジョン、階層、およびレベルについて、モデル・ビューワでは以下のように表示されます。

この場合、[**プロパティ**] オプションはドット構文を使用していることに注意してください。

10. プロパティを `City` レベルに追加します。

a. 左側 (クラス・ビューワ領域) で [HomeCity] を展開します。

b. [Population] をドラッグして、中央領域の [City] レベルにドロップします。

c. [PrincipalExport] をドラッグして、中央領域の [City] レベルにドロップします。

d. 新しい [PrincipalExport] プロパティを選択して、名前を Principal Export に変更します。

11. PrimaryCarePhysician プロパティに基づいて、ディメンジョン、階層、およびレベルを追加します。以下はその方法です。

a. [**要素を追加**] をクリックします。

b. [**新しい要素名を入力**] に、DocD と入力します。

c. [**データ・ディメンジョン**] をクリックします。

d. [OK] をクリックします。

e. モデル・ビューワ領域で、[New_Level1] をクリックします。

f. [**名前**] を Doctor に変更します。

g. [**ソース値**] ヘッダの下で [**表現**] ラジオ・ボタンを選択して、以下の ObjectScript 式を [**式**] にコピーします。

```
%source.PrimaryCarePhysician.LastName_", "_%source.PrimaryCarePhysician.FirstName
```

変数 %source は、現在のレコードを参照します。この式はキューブの構築時に評価されます。

代わりに、以前と同じドラッグ・アンド・ドロップ手順を使用して、定義を編集することもできます。

12. 以前と同じ方法でキューブ定義を保存します。

13. キューブをコンパイルおよび構築します。そのためには、以下の操作を実行します。

a. [**コンパイル**] をクリックすると、コンパイルが開始され、進捗状況がダイアログ・ボックスに表示されます。

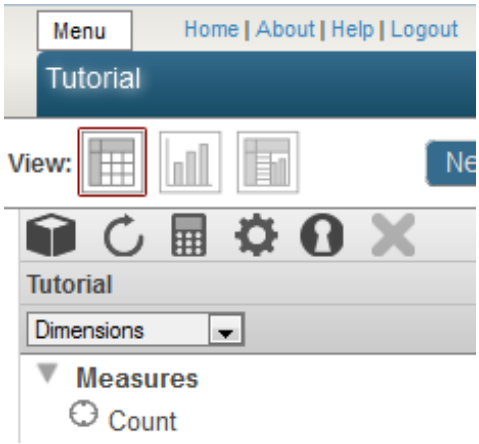
b. コンパイルが終了したら、[**完了**] をクリックします。

c. [**ビルド**] をクリックします。モーダル・ウィンドウで、[**選択構築**] が既定で選択された状態で表示されていること、および追加したメジャーとディメンジョンが下のリストに表示されていて、それぞれに 選択構築 プロセスの対象としてチェックが付いていることに注意してください。新しい列を追加するか、既存の列を変更すると、常にこのように 選択構築 のプロンプトが表示されます。[**ビルド**] をクリックします。

d. キューブおよびインデックスの構築が完了したら、[**完了**] をクリックします。

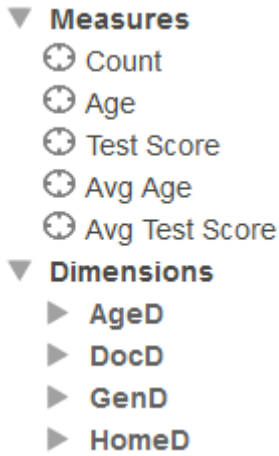
14. 別のブラウザ・タブまたはウィンドウで、アナライザを開きます。

左上の領域を確認します。この領域には、現在選択されているキューブまたはサブジェクト領域のタイトルが表示されます。以下のように表示されます。



このタイトルが Tutorial でない場合は、変更ボタン  をクリックし、[Tutorial] をクリックして [OK] をクリックします。

アナライザの左の領域では、このキューブの現在の内容が以下のように表示されます。



このように表示されない場合は、サンプルのデータが生成されていること、およびキューブのコンパイルとビルドが完了していることを確認してください。

3.3 初期キューブの検証

ここでは、キューブを検証して、変更の必要があるかどうかを調べます。

キューブを検証するには、キューブの要素を左の領域からピボット・ビルダ領域にドラッグ・アンド・ドロップして、単純なピボット・テーブルを作成します。これは、[行] 領域とその右にある 3 つのボックスで構成されます。

まず、定義していないメジャー (Count) がアナライザで表示されていることに注目してください。このメジャーは自動的に提供され、ベース・クラスのレコードをカウントします。

以下の手順を実行して、新しいキューブを十分に理解します。

1. 左の領域の各ディメンジョン名の横にある三角形をクリックします。

   この操作によって、以下のように表示されます。

2. [Age] レベルを [行] 領域にドラッグ・アンド・ドロップします。以下のような表示になります。

Age	
0	135
1	149
10	145
11	145
12	154
13	154

このレベルのメンバが文字列として並べ替えられていることに注意してください。このレベルでは、数値でメンバを並べ替えるほうが適しているため、調整する必要があります。

3. [Doctor] レベルを [行] 領域にドラッグ・アンド・ドロップして、[Age] のすぐ上に配置します (この操作で、Age が Doctor に置換されます)。これで、以下のような表示になります。

Doctor	
,	1,462
Adam, Susan	24
Adams, Elmo	26
Ahmed, Belinda	18
Ahmed, Edward	25

注釈　ここで作成される他のディメンジョンと異なり、Doctor ディメンジョンは、データ・セットのサイズに応じて非常に多くのメンバを所有できます。現実の実装では、このように低いレベルでディメンジョンを作成することはほとんどありません。このチュートリアルでは、このディメンジョンを使用して、いくつかのキー・ポイントを示します。

医師名 , は、記録されている一次診療医のない患者を参照します (このような患者は、PrimaryCarePhysician フィールドの姓と名前がいずれも Null になります)。これは、このチュートリアルの次の部分でレベルを再定義するときに変更します。

ID に基づくレベルを作成する場合は常に、その ID が一意であるかどうかを配慮することが重要です。多くの場合 (製品名、手順コード、部門名など)、ID は一意です。ただし、人名が一意であることを前提とすることは安全ではありません。このレベルは、直接医師名に基づいているため、システムによって同名の医師が結合されます。

例えば、ある患者には、Doctor テーブルの行 17 で表される Agnes Djakovic という名前の医師がいて、別の患者には、名前は同じでも同じテーブルの行 380 で表される医師がいる場合があります。Doctor レベルには、これらの患者を結合する Agnes Djakovic という名前のメンバが存在することになります。

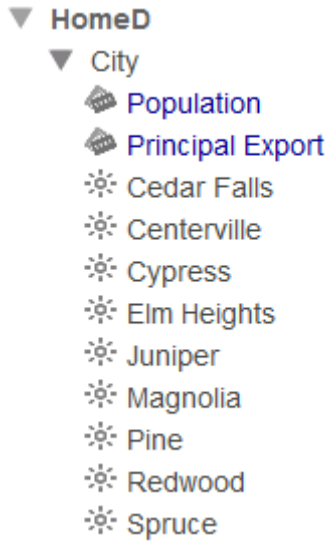
このチュートリアルの後続部分では、より堅牢なアプローチを使用します。

4. [Gender] レベルを [行] 領域にドラッグ・アンド・ドロップして、[Doctor] のすぐ上に配置します。この操作で、Doctor レベルが Gender レベルで置換されます。これで、以下のような表示になります。

Gender	
Female	5,112
Male	4,888

このレベルは特に変更する必要はありません。

5. 左側で [City] レベルを展開します。以下のように表示されます。



6. [City] レベルを [行] 領域にドラッグ・アンド・ドロップして、[Gender] のすぐ上に配置します。これで、以下のような表示になります。

City	
Cedar Falls	1,054
Centerville	1,136
Cypress	1,076
Elm Heights	1,179
Juniper	1,121
Magnolia	1,171
Pine	1,101
Redwood	1,110
Spruce	1,052

7. [Population] と [Principal Export] のプロパティを [列] にドラッグ・アンド・ドロップします。以下のように表示されます。

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

このレベルは特に変更する必要はありません。

3.4 キューブの調整

ここでは、キューブに以下の変更を加えます。

- `Age` のメンバの並べ替え方法を変更します。
- `Doctor` レベルが同名の医師を結合しないようにします。
- `Doctor` レベルに、`,` ではなく `None` (キューブの既定置換文字列) という名前のメンバが確実に存在するようにします。

1. アーキテクトにアクセスします。前回調べたキューブ定義が表示されます。
2. まず、`Age` レベルを調整して、メンバが数値で並べ替えられるようにします。そのためには、以下の操作を実行します。
   a. `[Age]` レベルをクリックします。
   b. **[要素を追加]** をクリックします。
   c. **[新しい要素名を入力]** に、`AgeSort` と入力します。
   d. **[プロパティ]** をクリックします。
   e. [OK] をクリックします。

      プロパティが追加され、アーキテクトでそれが選択されます。
   f. [詳細] ペインで **[表現]** を選択して以下を入力します。

      ```
      $CASE($LENGTH(%source.Age),2:%source.Age,:"0"_%source.Age)
      ```

      この式は、文字列の並べ替えで年齢が正しく並べ替えられるように、年齢の先頭にゼロを追加します。最初の年齢が 01 で、2 番目が 02 というように続きます (このサンプルの最高年齢は 99 歳であるため、年齢が 2 文字を超えることはありません)。
   g. **[プロパティ値でメンバを並べ替え]** で、[asc] を選択します。

      このオプションにより、メンバの並べ替え方法の制御にこのプロパティの値が使用されます。

h. キューブを保存します。

注釈　Patients サンプルでは別のアプローチを使用しますが、いずれのアプローチも有効です。

3. Doctor レベルを再定義して、同名の医師を結合できないようにします。そのためには、以下の操作を実行します。

   a. [Doctor] レベルをクリックします。

   b. **[表現]** フィールドで値を選択して、メモ帳またはその他の一時的な場所にこれをコピーします。

   c. **[プロパティ]** を選択して、PrimaryCarePhysician と入力します。

      これで、Doctor レベルが最低限の PrimaryCarePhysician プロパティに基づきます。これは OREF で医師ごとに一意です。

      これによって、このレベルでは偶然同名である異なる医師が結合されないようになります。

      また、この手順により、医師のいない患者に対する値が Null になります。これは、このレベルのそのようなメンバには、キューブの既定のヌル置換文字列が使用されることを意味します。

   d. [Doctor] レベルが選択された状態で、**[要素を追加]** をクリックします。

   e. **[新しい要素名を入力]** に、DoctorName と入力します。

   f. **[プロパティ]** をクリックします。

   g. [OK] をクリックします。

      プロパティが追加され、アーキテクトでそれが選択されます。

   h. [詳細] ペインで、**[表現]** を選択し、先ほどコピーした式を貼り付けます。

   i. **[メンバの名前として使用]** を選択します。

      このオプションにより、このプロパティの値が各メンバの名前として使用されます。

   j. **[プロパティ値でメンバを並べ替え]** で、[asc] を選択します。

      このオプションにより、このプロパティの値を基準にしてメンバが昇順で並べ替えられることになります。

4. キューブをコンパイルします。

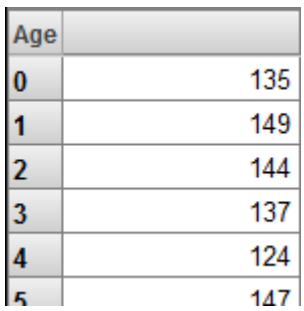
   コンパイルすると、アーキテクトでキューブが保存されます。

5. キューブを構築します。

6. アナライザに移動して、[Analytics]→**[アナライザ]** リンクをクリックし、最新のモデルで更新します。

7. 変更をダブル・チェックします。以下のように表示されます。

   - [Age] を [行] にドラッグ・アンド・ドロップすると、メンバが数値順に並べ替えられることがわかります。

Age	
0	135
1	149
2	144
3	137
4	124
5	147

   - [Doctor] を [行] にドラッグ・アンド・ドロップすると、None メンバが表示されます。

Doctor	
None	1,505
Adam, Dan	13
Adam, Danielle	21
Adam, Keith	24
Adam, Olga	31
Adam, Peter	23
Adam, Uma	21

生成されたデータによっては、重複している医師名も表示されます。以下はその例です。

Lee, Amanda	19
Lee, Bill	16
Lee, Chris	19
Lee, Chris	24

3.5 キューブへのリストの追加

リストを使用すると、ユーザには最下位レベルのデータから選択したフィールドが表示されます。これはさまざまなシナリオで有益です。この情報は、ユーザが異常値のレコードや、追跡処理の必要なレコードを特定する際に役立ちます。

1. まず、Patients テーブルで使用可能なフィールドを調べます。
   a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

      (これが別のブラウザ・タブで開かれている場合は、そのタブに切り替えます)。
   b. **[システムエクスプローラ→[SQL]** をクリックします。
   c. **[クエリ実行]** タブをクリックします。
   d. 以下のクエリを実行します。

      SQL

      ```
      select * from bi_study.patient
      ```

      これで、最初の 1000 人の患者が表示され、使用可能なフィールドが示されます。
   e. ここで以下のようなクエリを試行します。

      SQL

      ```
      select patientid, age,testscore,homecity->name as "City",
      primarycarephysician->lastname as "Doctor" from BI_Study.Patient
      ```

   f. このクエリをメモ帳やその他の使いやすい一時的な場所にコピーします。

   このブラウザ・タブまたはウィンドウは、後で使用できるように開いたままにしておきます。
2. 今実行したクエリのフィールドを使用するリストを追加します。
   a. アーキテクトにアクセスします

      (これが別のブラウザ・タブで開かれている場合は、そのタブに切り替えます)。

b. **[要素を追加]** をクリックします。

c. **[新しい要素名を入力]** に、SampleListing と入力します。

d. **[詳細リスト]** をクリックします。

e. **[OK]** をクリックします。

リストが追加されます。

f. [詳細] ペインで、以下に示すように、前の手順で保存したクエリから **[フィールドリスト]** 領域にフィールドのリストをコピーし、select を削除します。

```
patientid, age,testscore,homecity->name as "City", primarycarephysician->lastname as "Doctor"
```

システムでは、このフィールドのリストを使用して SQL クエリを構築します。

g. キューブをコンパイルします。

コンパイルすると、アーキテクトでキューブが保存されます。

キューブを再構築する必要はありません。

3. アナライザでこのリストにアクセスできることを確認します。そのためには、以下の操作を実行します。

a. アナライザにアクセスします。

(アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→**[アナライザ]** リンクをクリックし、最新のモデルで更新します。)

b. プレビュー領域に既に表示されているピボット・テーブル内のセルをクリックするか、単純なピボット・テーブルを作成してその中のセルをクリックします。

c. [リストを表示] ボタン  をクリックします。

以下のように表示されます。

#	PatientID	Age	TestScore	City	Doctor
1	SUBJ_100315	93	67	Cypress	Zevon
2	SUBJ_100325	36	75	Magnolia	Burroughs
3	SUBJ_100341	9	50	Spruce	Hammel
4	SUBJ_100356	8		Redwood	Novello
5	SUBJ_100385	32	70	Cedar Falls	Townsend
6	SUBJ_100403	11	51	Cedar Falls	Olsen
7	SUBJ_100426	29	72	Magnolia	Quine
8	SUBJ_100467	73		Pine	Quilty

注釈　既定では、最初の 1000 レコードが表示されます。これは、アナライザ内で変更できます。

リストがサポートされないことを示すメッセージが表示された場合は、キューブが保存され、リコンパイルされていることを確認します。

4. リストを変更して、レコードが別の方法で並べ替えられるようにします。

a. 再度アーキテクトにアクセスします。

b. [モデル・コンテンツ] 領域でリストをクリックします。

c. [詳細] ペインで、[Order By] に以下のように入力します。

```
age,homecity->name
```

d. キューブをコンパイルします。

コンパイルすると、アーキテクトでキューブが保存されます。

5. リストが、まず年齢で並べ替えられ、次に各年齢内で市区町村を基準にして並べ替えられることを確認します。

前と同様にリストを表示します。以下のような表示になります。

#	PatientID	Age	TestScore	City	Doctor
1	SUBJ_107966	0	99	Cedar Falls	Vonnegut
2	SUBJ_110142	0	88	Cedar Falls	Alton
3	SUBJ_109388	0	88	Cypress	
4	SUBJ_106292	0	56	Juniper	Young
5	SUBJ_104172	1	61	Centerville	Frith
6	SUBJ_102058	1		Cypress	Vivaldi
7	SUBJ_103519	1	79	Elm Heights	Koenig
8	SUBJ_104709	1	66	Elm Heights	Ingersol

下にスクロールして、患者が各年齢内で市区町村によって並べ替えられていることを確認します。

3.6 ファクト・テーブルおよびレベル・テーブルの確認

キューブ定義の作成を担当している場合は、システムがキューブ定義をどのように使用してシステムで直接使用されるテーブル (ファクト・テーブル と レベル・テーブル) を構築するのかを理解しておくと役立ちます。ここではこれらのテーブルを検証します。

1. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
2. **[システムエクスプローラ]→[SQL]** をクリックします。
3. **[クエリ実行]** タブをクリックします。
4. 以下の SQL クエリを実行します。これはキューブで使用されるベース・テーブルに対して実行されます。

SQL

```
select top 1 age,gender,homecity->name,primarycarephysician->lastname,
primarycarephysician->firstname, testscore from BI_Study.patient
```

詳細をメモしておきます。

#	Age	Gender	Name	LastName	FirstName	TestScore
1	13	F	Magnolia	Quince	Marvin	88
Complete						

5. 左側の領域で、テーブル `Tutorial_Cube.Fact` に移動します。
6. **[テーブルを開く]** をクリックします。

以下のように表示されます。

#	ID	%partition	%sourceId	DxAge	DxGender	DxNameViaHomeCity	DxPrimaryCarePhysician	MxAgeN	MxTestScoreN
1	1	1	1	1	1	1	1	13.00	88.00
2	2	1	2	2	1	2	2	27.00	
3	3	1	3	3	2	3	3	22.00	71.00
4	4	1	4	4	2	4	4	10.00	69.00
5	5	1	5	5	1	4	5	77.00	69.00
6	6	1	6	6	2	5	6	59.00	87.00
7	7	1	7	7	1	1	4	17.00	
8	8	1	8	8	1	6	7	39.00	
9	9	1	9	9	1	6	8	84.00	87.00
10	10	1	10	10	1	2	9	11.00	
11	11	1	11	11	1	7	10	25.00	
12	12	1	12	12	1	3	11	2.00	52.00
13	13	1	13	13	2	1	12	8.00	
14	14	1	14	14	1	3	13	19.00	

このテーブルは、キューブのコンパイル時に生成され、キューブの構築時にデータが入力されます。ファクト・テーブルには、ソース・テーブルから使用したレコードごとに、行 (ファクト) が 1 行あります。ここでは、ファクトのそれぞれが 1 人の患者に対応します。

このテーブルの最初の行は、ベース・テーブルの最初の行に対応します (13 歳で、テスト・スコアが 88 であった患者)。

7. 以下の点に注意してください。
   - `%sourceId` フィールドは、ファクトの基になったソース・レコードの ID を示します。
   - 名前の先頭が `Dx` である各フィールドは、定義したレベルに対応します。ファクト・テーブルのこれらのフィールドには整数が格納され、これらはレベル・テーブルのレコードを参照します。
   - 名前の先頭が `Mx` である各フィールドは、定義したメジャーに対応します。ファクト・テーブルのこれらのフィールドには、数値 (整数ではない) が格納されます (これがメジャーの既定のデータ型であるため)。
   - `MxTestScore` フィールドの値が Null のファクトもあります。
8. **[ウィンドウを閉じる]** をクリックします。
9. テーブル `[Tutorial_Cube.StarGender]` に移動します。
10. **[テーブルを開く]** をクリックします。表示は以下のようになります。

#	ID	DxGender
1	1	Female
2	2	Male
Complete		

このテーブルには、`Gender` レベルのメンバの名前が格納されます。ファクト・テーブルの `DxGender` フィールドは、このテーブルの行を参照します。

場合によって、`Male` が `Female` の前に表示されることもあります。

ここでは、システムで処理された最初の患者が女性であるために、`Female` メンバが先に表示されています。

システムは、これらのテーブルにデータを生成する際に、ベース・テーブルのレコードを繰り返し処理します。各レコードについて、それぞれのレベルの定義が確認され、値が決定されて、その値が (必要に応じて) 対応するレベル・テーブルに追加され、ファクト・テーブルのレベル・フィールドに検索値が書き込まれます。

11. **[ウィンドウを閉じる]** をクリックします。
12. テーブル `[Tutorial_Cube.StarAge]` に移動します。以下のように表示されます。

#	ID	Dx781900468	DxAge
1	1	13	13
2	2	27	27
3	3	22	22
4	4	10	10
5	5	77	77
6	6	59	59
7	7	17	17
8	8	39	39
9	9	84	84
10	10	11	11
11	11	25	25
12	12	02	2
13	13	08	8
14	14	19	19

Age レベルはベース・クラスの Age フィールドによって定義されます。この値は DxAge 列に表示されます。このレベルには、レベル・メンバの並べ替え順序の定義に使用されるレベル・プロパティがあります。この値は、Dx781900468 列に示されています。

このレベル・テーブルの最初のレコードは、13 歳のこの例で処理された最初の患者に対応します。

13. [ウィンドウを閉じる] をクリックします。
14. テーブル [Tutorial_Cube.StarNameViaHomeCity] に移動します。以下のように表示されます。

#	ID	Dx1438697606	DxNameViaHomeCity	DxPopulationViaHomeCity
1	1	bundt cake	Magnolia	4503
2	2	lettuce	Elm Heights	33194
3	3	spaghetti	Pine	15060
4	4	wheat	Juniper	10333
5	5	video games	Centerville	49000
6	6	mud	Spruce	5900
7	7	gravel	Cypress	3000
8	8	peaches	Redwood	29192
9	9	iron	Cedar Falls	90000
Complete				

City レベルはベース・クラスの HomeCity->Name フィールド によって定義されます。この値は DxNameViaHomeCity 列に表示されます。このレベルには、2 つのレベル・プロパティがあり、別の列に表示されています。

このテーブルの最初のレコードは Magnolia です。これは、ベース・テーブルの最初の患者の出身地です。

15. [ウィンドウを閉じる] をクリックします。
16. テーブル [Tutorial_Cube.StarPrimaryCarePhysician] に移動します。以下のように表示されます。

#	ID	Dx582175229	DxPrimaryCarePhysician
1	1	Quince, Marvin	232
2	2	Lopez, Kim	41
3	3	Drabek, Wolfgang	13
4	4	,	<null>
5	5	Cerri, Elvira	159
6	6	Sorenson, Hannah	12
7	7	Houseman, Janice	283

`Doctor` レベルはベース・クラスの `PrimaryCarePhysician` フィールドによって定義されます。これは `BI.Study.Doctor` クラスのインスタンスへの参照 (OREF) です。OREF は整数に変換され、それが `DxPrimaryCarePhysician` 列に書き込まれます。

このレベルでは、姓と名前をコンマを挟んで連結したレベル・プロパティによってメンバ名が定義されています。このレベル・プロパティの値は、`Dx582175229` 列に格納されます。

このテーブルの最初の医師は `Quince, Marvin` です。これは、ベース・テーブルの最初の患者の一次診療医です。

Null の医師の名前はコンマですが、これは表示されません。その代わり、このメンバには指定したヌル置換文字列が使用されます。

Tipヒント これらのテーブルのフィールド名を使いやすくするために、定義したレベルおよびメジャーに対してオプション **[ファクトテーブルのフィールド名]** を指定できます。このオプションは、特殊な内部処理が行われる時間レベル (次の項目で説明) には適用されないことに注意してください。

4

Business Intelligence チュートリアル：キューブ定義の拡張

チュートリアルの前のパートでは、単純なキューブを作成してテストしました。ここでは、そのキューブを展開して、Patients データのさらに多くの部分を使用し、さまざまな InterSystems IRIS Business Intelligence 機能を試してみます。

4.1 階層へのレベルの追加

これまでのところ、作成した各ディメンジョンには、レベルが 1 つある階層が 1 つ含まれています。ここでは、HomeD ディメンジョンの階層にレベルを追加します。

1. アーキテクトで、以下のように HomeD ディメンジョンにレベルを追加します。
   a. クラス・ビューワで [HomeCity] を展開します。
   b. [PostalCode] をドラッグして、[HomeD] ディメンジョン内の [H1] 階層にドロップします。
      この手順によって、City レベルの後に新規レベル PostalCode が追加されます。
   c. [PostalCode] をクリックします。
   d. [詳細] ペインで、[名前] を ZIP Code に変更します。
2. キューブをコンパイルします。
   コンパイルすると、アーキテクトでキューブが保存されます。
3. キューブを構築します。モーダル・ウィンドウで、[選択構築] が既定で選択された状態で表示されていること、および追加したレベルが下のリストに表示されていて、それぞれに 選択構築 プロセスの対象としてチェックが付いていることに注意してください。新しい列を追加するか、既存の列を変更すると、常にこのように 選択構築 のプロンプトが表示されます。
4. Analyzer にアクセスします。
   (アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[アナライザ] リンクをクリックし、最新のモデルで更新します。)
5. 左側の [HomeD] ディメンジョンを展開します。以下のように表示されます。

6. ZIP Code レベルを行として表示します。以下のような表示になります。

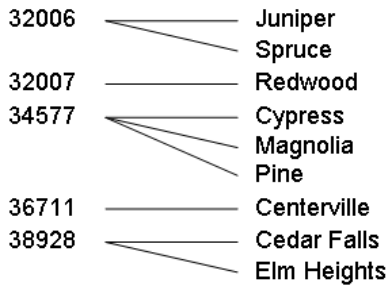
ZIP Code	
32006	1,051
32006	1,122
32007	1,159
34577	1,075
34577	1,138
34577	1,183
36711	1,106
38928	1,090
38928	1,076

一部のメンバに同じ名前があることに注意してください。同名で複数のメンバが存在することが適切である場合もあります。ただし、郵便番号は一意であるため、この場合はエラーです。

レベルに同じ名前の複数のメンバが存在する可能性がある状況は以下の 2 つのみです。

- レベル名が一意でないレベル・プロパティに基づいている場合 (この例については、前の項目で定義した Doctor レベルを参照してください)。
- レベルに親レベルがある場合。システムがレベルのメンバを作成するときは、ソース・プロパティまたは式だけではなく、親メンバも考慮されます。

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するため、どちらも他方の親にはなりません。Patients サンプルの郵便番号には、以下のように小さい市区町村が含まれています。



ZIP Code レベルを追加する際、このレベルを City レベルの後に配置しました。これは、City が ZIP Code の親であることを意味します。これが、ZIP Code のメンバの生成方法に影響していました。例えば、システムは Juniper 市の郵便番号 32006 と、Spruce 市の郵便番号 32006 を同じではないと見なしていました。

7. アーキテクトに戻り、HomeD ディメンジョンを修正します。
   a. [ZIP Code] レベルをクリックします。
   b. 上矢印ボタンをクリックします。
   c. キューブをコンパイルします。

      コンパイルすると、アーキテクトでキューブが保存されます。

d. キューブを構築します。

8. Analyzer にアクセスします。

(アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[**アナライザ**] リンクをクリックし、最新のモデルで更新します。)

9. 左側の `[HomeD]` ディメンジョンを展開します。以下のように、修正されて表示されます。

10. `ZIP Code` レベルを行として表示します。これで、以下のような適正な表示になります。

ZIP Code	
32006	2,173
32007	1,159
34577	3,396
36711	1,106
38928	2,166

11. 行 `[34577]` をダブルクリックします。これで、この郵便番号内に市区町村が表示されます。

ZIP Code:34577	
« Cypress	1,183
« Magnolia	1,075
« Pine	1,138

12. 必要に応じて、以下のように実行し、この変更がファクト・テーブルおよびレベル・テーブルにどのように影響したかを確認します。

a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

b. [**システムエクスプローラ**]→[SQL] をクリックします。

c. 左側の領域で、テーブル `Tutorial_Cube.Fact` に移動します。

このテーブル行に、`DxNameViaHomeCity` に加えてフィールド `DxPostalCodeViaHomeCity` が存在することに注意してください。つまり、このファクト・テーブルには、レベルが関連付けられたものであっても各レベルの値が格納されています。

d. 左側の領域で、テーブル StarNameViaHomeCity に移動し、これを開きます。

以下のように表示されます。

#	ID	Dx582175229	DxPrimaryCarePhysician
1	1	Uhles, Phil	7636
2	2	Isaksen, Violet	8196
3	3	Geoffrion, Andrew	9425
4	4	Quigley, Andrew	9887
5	5	Minichillo, Ed	1664
6	6	Orwell, Michelle	1527
7	7	Sato, Zeke	4373
8	8	Malkovich, Chris	2514
9	9	Gomez, Emilio	8479
10	10	Browne, Barb	1899

このテーブルには、市区町村ごとにその市区町村が属する郵便番号が格納されていることに注意してください。

e. このテーブルを閉じて、テーブル `[Tutorial_Cube.StarPostalCodeViaHomeCity]` に移動します。

以下のように表示されます。

#	ID	DxPostalCodeViaHomeCity
1	1	38928
2	2	32007
3	3	34577
4	4	32006
5	5	36711

このレベル・テーブルは、他のレベル・テーブルと同様で、レベル・メンバごとに行が 1 行あります。

4.2 時間レベルの追加

ここでは、キューブに時間レベルを追加します。

Patients クラスには、患者の生年月日が複数の形式で含まれています (このため Business Intelligence の各種形式を試してみることができます)。

```
Property BirthDate As %Date;
Property BirthDateTimeStamp As %TimeStamp;
```

Business Intelligence には、$HOROLOG 形式などだけではなく、これら 3 つすべての形式の組み込みサポートが用意されています (詳細は ″InterSystems Business Intelligence のモデルの定義″ を参照してください)。

このクラスには、`BirthDateTimeStamp` プロパティの一部として、または以下のプロパティとして、患者の出生時間も含まれます。

Class Member

```
Property BirthTime As %Time;
```

最も柔軟性の高いプロパティは `BirthDateTimeStamp` です。このプロパティには出生日付と出生時間の両方が格納されているため、時間レベルの基礎として使用します。

1. アーキテクトにアクセスし、`Tutorial` キューブを表示します。

2. [**要素を追加**] をクリックします。
3. [**新しい要素名を入力**] に、BirthD と入力します。
4. [**時間ディメンジョン**] をクリックします。
5. [OK] をクリックします。

   ディメンジョン、階層およびレベルが生成されます。
6. ディメンジョンを以下のように変更します。
   - [**プロパティ**] の横の検索ボタンをクリックし、[BirthDateTimeStamp] をクリックしてから [OK] をクリックします。
7. レベルを以下のように変更します。
   - レベルの名前を Year に変更します。
   - [**関数で値を抽出**] で [Year] を選択します。

     このオプションは、このレベルが患者の誕生年にのみ基づくことを意味します。
8. 以下のように別のレベルを追加します。
   a. このディメンジョンの階層 [H1] をクリックします。
   b. [**要素を追加**] をクリックします。
   c. [**新しい要素名を入力**] に、Month Year と入力します。
   d. [**レベル**] をクリックします。
   e. [OK] をクリックします。

      階層 H1 で、既存の Year レベルの後に新しいレベルが作成されます。
9. Month Year レベルを、以下のように変更します。
   - [**関数で値を抽出**] で [MonthYear] を選択します。

     このオプションは、このレベルが誕生年と月を結合したものに基づくことを意味します。
10. 以下のように BirthD ディメンジョンにさらに階層とレベルを追加します。
    a. ディメンジョン名をクリックします。
    b. [**要素を追加**] をクリックします。
    c. [**新しい要素名を入力**] に、H2 と入力します。
    d. [**階層**] をクリックします。
    e. [OK] をクリックします。

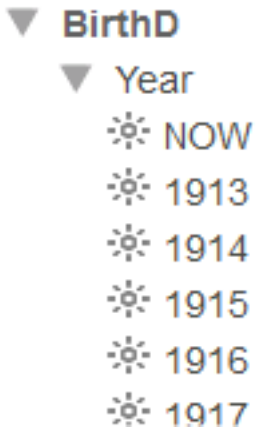
       新しい階層とレベルが生成されます。
    f. 新しいレベルを、以下のように変更します。
       - レベルの名前を Time に変更します。
       - [**関数で値を抽出**] で [HourNumber] を選択します。

         このオプションは、このレベルが患者が生まれた日の時間に基づくことを意味します。
11. キューブをコンパイルします。

コンパイルすると、アーキテクトでキューブが保存されます。

12. キューブを構築します。
13. Analyzer にアクセスします。

    (アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[**アナライザ**] リンクをクリックし、最新のモデルで更新します。)
14. 新しいレベルを使用してみます。以下のように表示されます。
    - 左の領域で `[Year]` を展開すると、以下のように表示されます。



      NOW は、現在の年 (このコンテキストの場合) を参照する特殊なメンバです。
    - `Month Year` レベルにも NOW メンバが存在し、これは現在の年および月を参照します。
    - `Year` を行として使用すると、以下のように表示されます。

Year	
1911	8
1912	4
1913	6
1914	14
1915	4
1916	10
1917	8

    - `Month Year` を行として使用すると、以下のように表示されます。

Month Year	
Feb-1911	2
Jul-1911	3
Oct-1911	1
Nov-1911	1
Dec-1911	1
Mar-1912	1
Jun-1912	1

    - `Time` を行として使用すると、以下のように表示されます。

Time	
12am	438
1am	421
2am	425
3am	419
4am	414
5am	410
6am	411

時間レベルにはテーブルが生成されません。このレベルでは特殊な内部処理が行われます。

4.3 コレクション・プロパティの使用

レベルは、コレクション・プロパティに基づいて作成できます。具体的には、システムは、$LIST によって返されるタイプのリスト %List、または文字区切りリストのいずれかを直接使用することができます。コレクション・プロパティで、データが別の方法で格納されている場合は、必要なデータを抽出して、サポートされるリスト・タイプのいずれかを作成する必要があります。

`BI.Study.Patient` クラスには、`Allergies`、`DiagnosesAsLB` など、いくつかのコレクション・プロパティがあります。`DiagnosesAsLB` プロパティは以下のように定義されます。

Class Member

```
Property DiagnosesAsLB As %List;
```

`Allergies` プロパティは以下のように定義されます。

Class Member

```
Property Allergies As list Of BI.Study.PatientAllergy;
```

ここでは、これらのプロパティを使用するレベルおよびメジャーの作成方法を説明します。

1. アーキテクトにアクセスし、`Tutorial` キューブを表示します。
2. `DiagnosesAsLB` プロパティを使用するディメンジョン、階層およびレベルを以下のように追加します。
   a. **[要素を追加]** をクリックします。
   b. **[新しい要素名を入力]** に、`DiagD` と入力します。
   c. **[データ・ディメンジョン]** をクリックします。
   d. [OK] をクリックします。

      ディメンジョン、階層およびレベルが生成されます。
   e. レベルの名前を `Diagnoses` に変更します。
   f. レベルが選択されている状態で、**[プロパティ]** の検索ボタンをクリックし、`[DiagnosesAsLB]` プロパティを選択して、[OK] をクリックします。
   g. **[ソース値リストタイプ]** で、**[$List 構造]** をクリックします。このタイプは、$LIST 関数によって返される形式、または %List タイプを持つデータを参照します。

h. キューブ・クラスを保存します。

3. アーキテクトで、前出の手順と同様にディメンジョン、階層、およびレベルを追加しますが、以下の変更点があります。

   - ディメンジョン名は `AllerD` にします。
   - レベル名は `Allergies` にします。
   - **[プロパティ]** の値は指定しないでください。

     直接使用できるプロパティはありません。式を使用してアレルギーのリストを抽出する必要があります。
   - **[表現]** に以下の値を指定します。

     ```
     ##class(Tutorial.Cube).GetAllergies(%source.%ID)
     ```

     キューブを構築する際に、この式がファクト・テーブルの行ごとに 1 回ずつ評価されます。

     変数 %source は、現在のレコードを参照します。この式は、患者の ID を取得し、ユーティリティ・メソッド (これから作成) を呼び出して、患者のアレルギーのリストを返します。
   - 忘れずに **[ソース値リストタイプ]** で **[$List 構造]** を選択してください。

   キューブ・クラスを保存します。

   次の手順では、このユーティリティ・メソッドを作成します。

4. スタジオを開き、サンプルをインストールしたネームスペースにアクセスします。
5. 作成したキューブ・クラス `[Tutorial.Cube]` を開きます。
6. 以下のように、`GetAllergies()` という名前のメソッドを追加します。

   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("")}
       Set list=""
       For i=1:1:allergies.Count() {
           Set $LI(list,i)=allergies.GetAt(i).Allergen.Description
           }
       Quit list
   }
   ```

   このメソッドは、患者の ID が指定されると、作成したレベルで期待される形式でその患者のアレルギーのリストを返します。

   %OpenId() の 2 番目の引数によって、使用する同時処理ロックのレベルが指定されます。オブジェクトからのデータの読み取りのみが必要であるため、この値を 0 に指定して同時処理ロックを設定しないことで、より高速に実行します。

7. スタジオでキューブ・クラスを保存して、コンパイルします。
8. 患者が持っているアレルギーの数を含むメジャーを追加します。このためには、以下のように `Allergies` プロパティを使用します。

   a. アーキテクトに戻ります。

   b. **[要素を追加]** をクリックします。

   c. **[新しい要素名を入力]** に、`Avg Allergy Count` と入力します。

   d. **[メジャー]** をクリックします。

e. [OK] をクリックします。

新しいメジャーがテーブルに追加されます。

f. [モデル・コンテンツ] 領域でメジャーをクリックします。

g. [**集計**] で [AVG] をクリックします。

h. [**表現**] に、以下を入力します。

```
##class(Tutorial.Cube).GetAllergyCount(%source.%ID)
```

このメソッドは、後で作成する必要があります。

i. アーキテクトでキューブ・クラスを保存します。

j. クラスをスタジオで編集したため、アーキテクトでは保存された定義をオーバーライドするかどうかを尋ねるダイアログ・ボックスが表示されます。[OK] をクリックします。アーキテクトでは、クラス定義のうちアーキテクトで編集可能な部分のみをオーバーライドします。このため、クラスに追加したメソッドはオーバーライドしません。

k. スタジオで、以下のメソッドをキューブ・クラスに追加します。

Class Member

```
ClassMethod GetAllergyCount(ID As %Numeric) As %Numeric
{
     Set allergies=##class(BI.Study.Patient).%OpenId(ID,0).Allergies
     Quit allergies.Count()
}
```

l. スタジオでキューブ・クラスを保存して、コンパイルします。

9. キューブを再構築します。

そのためには、アーキテクトに戻り、以前と同じ方法で再構築します。

または、ターミナル・ウィンドウを開き、サンプルをインストールしたネームスペースで以下のコマンドを入力することもできます。

ObjectScript

```
 do ##class(%DeepSee.Utils).%BuildCube("tutorial")
```

メソッドはキューブの論理名 (クラス名ではなく) を使用します。また、キューブ名では大文字と小文字は区別されません。

10. アナライザにアクセスします。

(アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[**アナライザ**] リンクをクリックし、最新のモデルで更新します。)

11. `Diagnoses` レベルを行として表示します。以下のように表示されます。

Diagnoses	
None	8,499
asthma	647
CHD	318
diabetes	506
osteoporosis	176

データには、もっとまれなてんかんの診断も表示される場合があります。

代わりに以下のような表示になることがあります。

Diagnoses	
CHD	247
CHD osteoporosis	25
asthma	618
asthma CHD	16
asthma CHD osteoporosis	2

これは、[**ソース値リストタイプ**] に適切なタイプを指定しなかった場合に発生します。

12. [**新規**] をクリックします。

13. 新しい `Allergies` レベルを行として表示し、`Count` メジャーと `Avg Allergy Count` メジャーを表示します。表示は以下のようになります。

Allergies	Count	Avg Allergy
None	3,879	0
additive/coloring agent	431	1.78
animal dander	454	1.73
ant bites	433	1.77
bee stings	453	1.75
dairy products	402	1.70
dust mites	447	1.66
eggs	393	1.75
fish	459	1.75
mold	421	1.79
nil known allergies	1,485	1
peanuts	431	1.72
pollen	468	1.73
shellfish	430	1.75
soy	450	1.71
tree nuts	442	1.65

`nil known allergies` メンバは、既知のアレルギーのない患者を表します。一部の医療情報システムでは、以下の技法を使用して患者に既知のアレルギーがないことを記録します。

- システムには `nil known allergies` と呼ばれる特別なアレルゲンが含まれます。
- システムのユーザが患者にアレルギーがあるかどうか質問して、その回答がいいえの場合、ユーザは値 `nil known allergies` を選択します。

システムは、この文字列に特別な意味を割り当てません。このディメンジョンは、このアレルゲンをその他のアレルゲンと同じように処理します。

Null メンバ (`None` と呼ばれる) は、`Allergies` プロパティが Null である患者を表します。これらの患者にアレルギーがないと見なすことは不適切であるため、このメンバの名前では語弊があります。`No Data Available` といった名前のほうが適しています。

Null メンバに属する患者については、Avg Allergy Count メジャーは 0 です。これらの患者の場合は Avg Allergy Count メジャーが Null である必要があります。

また、既知のアレルギーがない患者の場合は Avg Allergy Count メジャーが 1 になります。これは、Allergies プロパティに特殊な nil known allergies アレルゲンが含まれているためです。これらの患者の場合は Avg Allergy Count メジャーが 0 である必要があります。

このセクションの後続部分で、Null メンバの名前を修正し、Avg Allergy Count メジャーのロジックを調整します。

14. アーキテクトに戻ります。
15. [Allergies] レベルをクリックします。
16. [ヌル置換文字列] に No Data Available を指定します。
17. キューブ・クラスを保存します。
18. スタジオで、メソッド GetAllergyCount() を以下のように編集します。

Class Member

```
ClassMethod GetAllergyCount(ID As %Numeric)
{
    Set allergies=##class(BI.Study.Patient).%OpenId(ID,0).Allergies
    //check to see if patient has any recorded allergy data
    //if not, count is null

    If allergies.Count()=0 {
        Set allcount=""
                }
                 //check to see if patient has "Nil known allergies"
                 //in this case, the patient has one "allergen" whose code is 000
                Elseif ((allergies.Count()=1) && (allergies.GetAt(1).Allergen.Code="000")) {
                        Set allcount=0
                        }
                Else {
         Set allcount=allergies.Count()
         }

     Quit allcount
}
```

19. キューブ・クラスを保存します。
20. スタジオまたはアーキテクトでキューブ・クラスをコンパイルします。
21. アーキテクトでキューブを構築します。
22. アナライザにアクセスします。

(アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[アナライザ] リンクをクリックし、最新のモデルで更新します。)

23. Allergies を行として表示し、Count および Avg Allergy Count メジャーを表示します。以下のような画面が表示されているはずです。

Allergies	Count	Avg Allergy
No Data Available	3,879	
additive/coloring agent	431	1.78
animal dander	454	1.73
ant bites	433	1.77
bee stings	453	1.75
dairy products	402	1.70
dust mites	447	1.66
eggs	393	1.75
fish	459	1.75
mold	421	1.79
nil known allergies	1,485	0
peanuts	431	1.72
pollen	468	1.73
shellfish	430	1.75
soy	450	1.71
tree nuts	442	1.65

24. 必要に応じて、以下のように実行して、リストベースのレベルがファクト・テーブルおよびレベル・テーブルでどのように表されるのかを確認します。

    a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

    b. **[システムエクスプローラ]→[SQL]** をクリックします。

    c. 左側の領域で、テーブル `Tutorial_Cube.Fact` に移動してこれを開き、フィールド `DxDiagnosesAsLB` にスクロールします。

    以下のように表示されます。

DxDiagnosesAsLB
1
1
1
1
1
1
2
1
3,4
2

    このフィールドには、患者の診断が格納されます。複数の値が格納される場合もあります。

    このテーブルには、以下のようなアレルギー・レベルも表示されます。

Dx1208719676
1
2,3
4
5
6
5
6
6
5
7

レベル自体が式に基づいていて、このフィールドの名前は生成されたものであるため、明確性に欠けています。

これもリストベースのレベルであるため、複数の値が格納される場合があります。

d. テーブル `Tutorial_Cube.StarDiagnosesAsLB` に移動して、これを開きます。

#	ID	DxDiagnosesAsLB
1	1	<null>
2	2	asthma
3	3	CHD
4	4	osteoporosis
5	5	diabetes

このレベル・テーブルは、他のレベル・テーブルと同様で、レベル・メンバごとに行が 1 行あります。

アレルギーのレベル・テーブルも同様で、レベル・メンバごとに行が 1 行あります。

`Avg Allergy Count` に使用したメソッドはかなり単純なものでした。以下のメソッドについて考えてみます。

```
ClassMethod GetScore(ID As %Numeric) As %String
{
    //get customer rating data & call duration from source record
    set call=##class(MyPackage.MyClass).%OpenId(ID,0)
    set professionalism=call.Professionalism
    set knowledge=call.Knowledge
    set speed=call.OpenDuration

    If ...
        //logic to check for nulls and combine these values into weighted overall score
    Quit score
}
```

総合的なスコアを示すメジャーを定義するには、このようなメソッドを使用できます。

4.4 置換の定義

ここでは、レベルの元の値を別の値に変換するオプションを使用します。ここでは患者の `Age` プロパティを使用します。患者を 1 年より大きいバケットに配置するレベルを定義します。

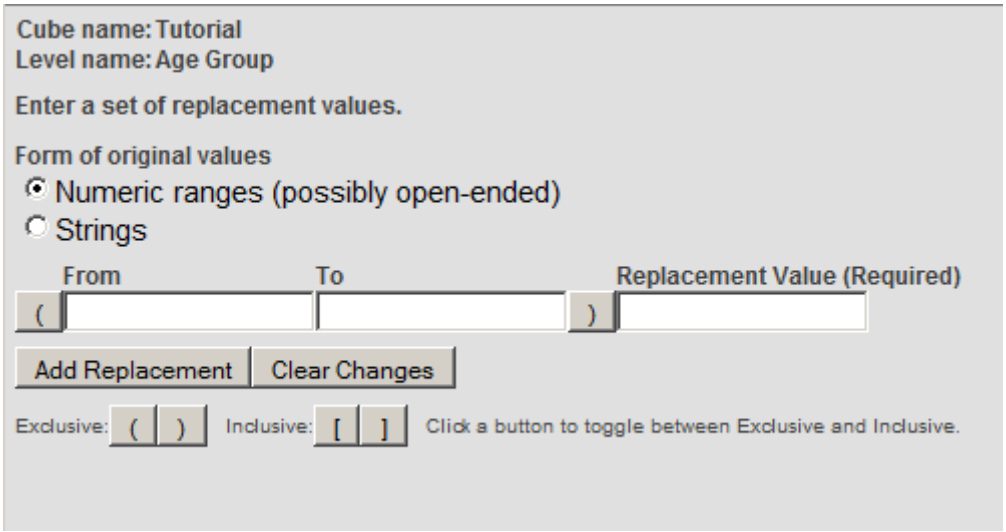
`Age Group` レベルには以下のメンバが含まれます。

- `0 to 29` メンバは 30 歳未満の患者で構成されます。
- `30 to 59` メンバは 30 歳から 59 歳まで (端点を含む) の患者で構成されます。
- `60+` メンバは 60 歳以上の患者で構成されます。

同様に、Age Bucket レベルには、0 to 9、10 to 19 などのメンバが含まれます。

1. アーキテクトにアクセスします。
2. 以下のように AgeD ディメンジョンにもう 1 つレベルを追加します。そのためには、以下の操作を実行します。
   a. [Age] レベルをクリックします。これによって、粒度の粗い新しいレベルが Age レベルの前に追加されます。
   b. [**要素を追加**] をクリックします。
   c. [**新しい要素名を入力**] に、Age Group と入力します。
   d. [**レベル**] をクリックします。
   e. [OK] をクリックします。
3. 範囲式を含むように、新しい Age Group レベルを以下のように再定義します。
   a. 新しい [Age Group] レベルをクリックします。
   b. [**プロパティ**] に、Age と入力します。
   c. [**範囲表現**] の横にある検索ボタンをクリックします。

      置換セットを指定するダイアログ・ボックスが表示されます。このダイアログ・ボックスの最初の表示は以下のようになります。



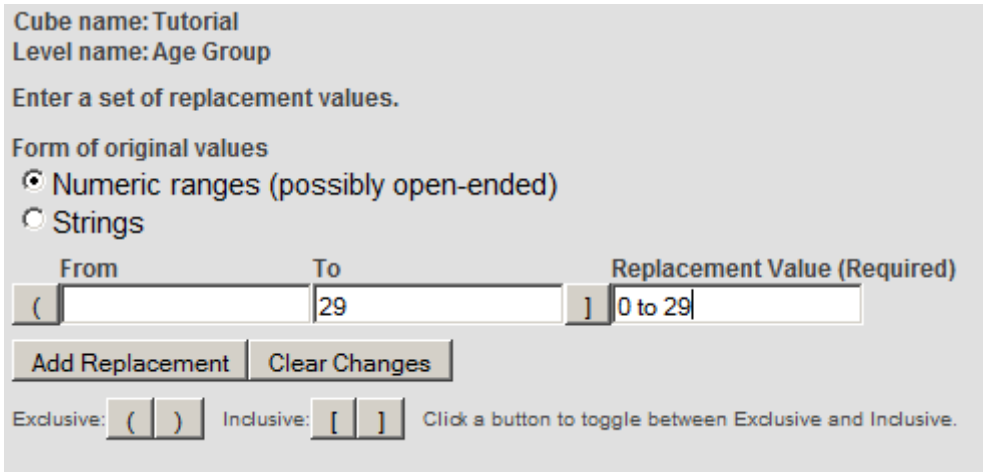
      数値データの場合は、置換ごとに、代わりに使用する新しい値のほかに元の値の範囲も指定します。

   d. [To] に 29 と入力します。

      [To] の右にあるボタンの初期表示は次のようになります : 

   e. このボタンをクリックすると表示がこのように変わります : 

   f. [**置換する値**] に 0 to 29 と入力します。この結果は次のようになります。



この範囲の下限値は指定されていないため、[From] の横にあるボタンはどれでもかまいません。

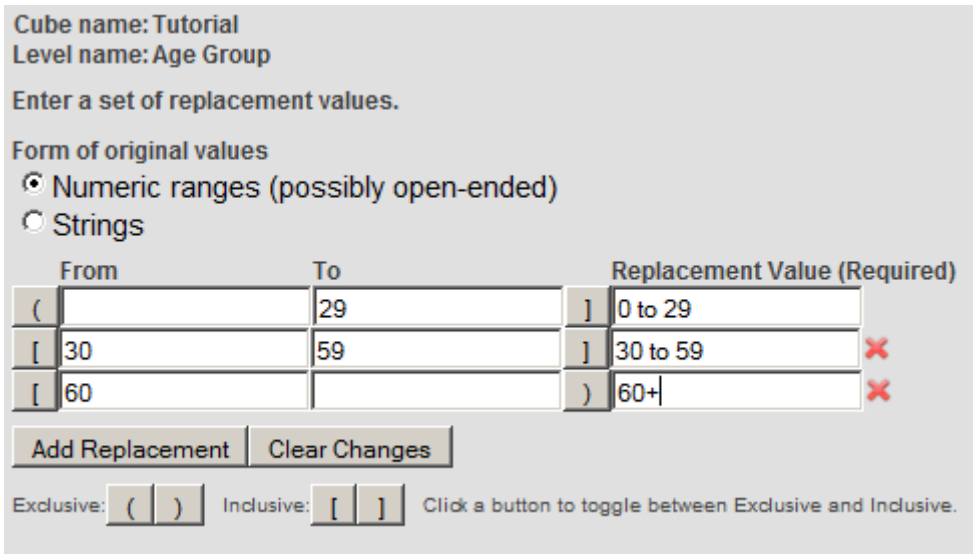
g. **[置換を追加]** をクリックします。

h. 新しい行で、[From] および [To] 横にある切り替えボタンをクリックします。

i. [From] には `30`、[To] には `59` と入力します。

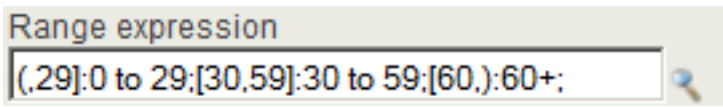
j. **[置換する値]** に `30 to 59` と入力します。

k. **[置換を追加]** をクリックして、最終行を追加すると、結果は以下のようになります。



l. [OK] をクリックします。

ダイアログ・ボックスが閉じられ、**[範囲表現]** フィールドに以下のように値が表示されます。



この値は、指定した置換の表記にシステムが内部的に使用する構文を示します。

m. **[ファクトテーブルのフィールド名]** に `DxAgeGroup` を指定します。

これで、このレベル定義が生成されたテーブルにどのように影響するかを理解しやすくなります。

4. キューブを保存します。

   Age Bucket レベルでも同じ方法を使用できます。ただし、このチュートリアルでは別の方法、つまり、年単位の年齢を適切な 10 年バケットに対応する文字列に変換するソース式を使用します。

5. スタジオでクラス [BI.Model.PatientsCube] を開きます。

6. メソッド GetAgeBucket() の定義を確認します。これは次のようになります。

   Class Member

   ```
   ClassMethod GetAgeBucket(age As %Numeric) As %String
   {
       If (age="") {Set return=""}
       ElseIf (age<10) {Set return="0 to 9"}
       ElseIf (age<20) {Set return="10 to 19"}
       ElseIf (age<30) {Set return="20 to 29"}
       ElseIf (age<40) {Set return="30 to 39"}
       ElseIf (age<50) {Set return="40 to 49"}
       ElseIf (age<60) {Set return="50 to 59"}
       ElseIf (age<70) {Set return="60 to 69"}
       ElseIf (age<80) {Set return="70 to 79"}
       ElseIf (age>=80) {Set return="80+"}
       Else {Set return=""}
       Quit return
   }
   ```

   このメソッドへの入力は、患者の ID ではなく単なる数値であることに注意してください。

7. アーキテクトで、AgeD に以下のように別のレベルを追加します。

   a. [Age] レベルをクリックします。これによって、粒度の粗い新しいレベルが Age レベルの前に追加されます。

   b. [**要素を追加**] をクリックします。

   c. [**新しい要素名を入力**] に、Age Bucket と入力します。

   d. [**レベル**] をクリックします。

   e. [OK] をクリックします。

      Age の前、Age Group の後ろに新しいレベルが追加されます。

   f. [**表現**] に、以下を入力します。

      ```
      ##class(BI.Model.PatientsCube).GetAgeBucket(%source.Age)
      ```

   g. [**ファクトテーブルのフィールド名**] に DxAgeBucket を指定します。

      これで、このレベル定義が生成されたテーブルにどのように影響するかを理解しやすくなります。

   注釈　実際には、(この例のように別のキューブではなく) ユーティリティ・メソッドを使用するキューブ・クラスなどの一元的な場所にユーティリティ・メソッドを組み込む傾向にあります。この演習のポイントの 1 つは、このネームスペースでアクセス可能などのクラス・メソッドも呼び出しが可能であることを示すことにあります。同様に、任意のルーチンやシステム関数も呼び出すことができます。

8. キューブを保存します。

   クラスをスタジオで編集したため、アーキテクトでは保存された定義をオーバーライドするかどうかを尋ねるダイアログ・ボックスが表示されます。[OK] をクリックします。アーキテクトでは、クラス定義のうちアーキテクトで編集可能な部分のみをオーバーライドします。このため、クラスに追加したメソッドはオーバーライドしません。

9. キューブをコンパイルします。

10. キューブを再構築します。

11. アナライザにアクセスします。

(アナライザが別のブラウザ・タブで開かれている場合は、そのタブに切り替えて、[Analytics]→[**アナライザ**] リンクをクリックし、最新のモデルで更新します。)

12. 新しい `Age Group` レベルを行として表示します。以下のような画面が表示されているはずです。

Age Group	
0 to 29	4,260
30 to 59	4,130
60+	1,610

13. 新しい `Age Bucket` レベルを行として表示します。以下のような画面が表示されているはずです。

Age Bucket	
0 to 9	1,413
10 to 19	1,452
20 to 29	1,395
30 to 39	1,548
40 to 49	1,498
50 to 59	1,084
60 to 69	749
70 to 79	553
80+	308

14. 新しいレベル・テーブルの 1 つを調べて、システムの動作を理解します。
    a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。
    b. [**システムエクスプローラ**]→[SQL] をクリックします。
    c. 左側の領域で、テーブル `Tutorial_Cube.Fact` に移動して、これを開きます。

       現在このテーブルには、`AgeD` 階層のレベルの値を格納する 3 つのフィールドがあります。

DxAge	DxAgeBucket	DxAgeGroup
1	1	1
2	2	1
3	3	2
4	3	2
5	4	3
6	5	2
7	6	3
8	7	2
9	6	3
10	2	1

    d. テーブル `Tutorial_Cube.DxAgeGroup` に移動して、これを開きます。

       以下のように表示されます。

#	ID	DxAgeGroup
1	1	0 to 29
2	2	30 to 59
3	3	60+

Complete

       システムでは指定した範囲式を使用してこのデータが作成されています。

e. テーブル [Tutorial_Cube.DxAgeBucket] を開きます。

以下のように表示されます。

#	ID	DxAgeBucket	DxAgeGroup
1	1	20 to 29	1
2	2	10 to 19	1
3	3	40 to 49	2
4	4	70 to 79	3
5	5	50 to 59	2
6	6	60 to 69	3
7	7	30 to 39	2
8	8	0 to 9	1
9	9	80+	3

Complete

このレベルは階層の最上位ではないため、各要素について Age Group レベル内の親メンバへの参照が含まれます。DxAgeGroup 列を参照してください。

システムは GetAgeBucket() メソッドを使用して、このデータを生成します。

これらの 2 レベルは、同等の方法で定義されています。つまり、[**範囲表現**] オプションを使用することは、変換を行う独自のメソッドを実行することと同等です。メソッドには単純な置換よりはるかに複雑なロジックを組み込むことができます。以下のメソッドについて考えてみます。

```
ClassMethod GetClassification(ID As %Numeric) As %String
{
    //get customer rating data & call duration from source record
    set customer=##class(MyPackage.MyClass).%OpenId(ID,0)
    set detail1=customer.Detail1
    set detail2=customer.Detail2
    set detail3=customer.Detail3
    ...

    If ...
        //logic to use these details and return a string, either "A", "B", or "C"
    Quit classification
}
```

このようなメソッドを使用すると、顧客に関する複数の情報を使用するアルゴリズムに基づいて顧客をグループ化するレベルにデータを入力することができます。

4.5 他のクラスへのアクセス

アーキテクトでは、ベース・クラス内のほとんどのプロパティに簡単にアクセスすることができますが、SQL を介してのみアクセス可能なクラスのプロパティなど、その他のプロパティも使用できます。ここでは、BI.Study.PatientDetails クラスのデータをキューブのレベルとして使用します。

BI.Study.Patient クラスと BI.Study.PatientDetails クラスはクラス・プロパティによる接続はなく、仮接続もありません。ただし、双方のテーブルに PatientID プロパティがあり、規約によってこれらを接続します。つまり、指定の患者の情報を検出するには、これら 2 つのテーブルで同じ PatientID を持つレコードを検索する必要があります。

この演習では、BI.Study.PatientDetails のデータを検証し、さまざまな SQL クエリを試行して、レベルの定義に使用できるようにメソッド内にクエリをラップします。SQL に習熟している場合は、最初のほうの手順を省略してもかまいません。

1. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

2. [システムエクスプローラ]→[SQL] をクリックします。
3. [クエリ実行] タブをクリックします。
4. 以下のクエリを実行します。

SQL

```
SELECT PatientID FROM BI_Study.Patient
```

5. 後で参照できるように、いずれかの PatientID 値をメモしておきます。
6. 以下のクエリを実行します。

SQL

```
SELECT * FROM BI_Study.PatientDetails WHERE PatientID='SUBJ_100301'
```

以下のように表示されます。

ID	FavoriteColor	PatientID	Profession
1	Purple	SUBJ_100301	

7. 以下のクエリを実行します。

SQL

```
SELECT FavoriteColor FROM BI_Study.PatientDetails WHERE PatientID='SUBJ_100301'
```

以下のように表示されます。

FavoriteColor
Purple

次に、同様のクエリを実行し、クエリによって取得された値を返すクラス・メソッドを作成する必要があります。

このメソッドにはクエリが組み込まれ、sql() 内にラップされます。クエリを以下のように変更する必要があります。

- FavoriteColor の代わりに FavoriteColor INTO :ReturnValue を使用して、返り値が ReturnValue という名前のホスト変数に書き込まれるようにします。
- 'SUBJ_100301' を使用する代わりに、ベース・クラスの PatientID フィールドを渡す必要があります。

埋め込み SQL の実行後に、メソッドは変数 SQLCODE をチェックする必要があります。この変数は正常に行われたクエリの場合のみ 0 になります。レコードが見つからなかった場合、このクエリは失敗です。その場合は、空の文字列を返すことが適切です。

8. スタジオで、以下のメソッドをキューブ・クラス Tutorial.Cube に追加します。

Class Member

```
ClassMethod GetFavoriteColor(patientID As %String) As %String
{
 &sql(SELECT FavoriteColor INTO :ReturnValue FROM BI_Study.PatientDetails WHERE PatientID=:patientID)

 If (SQLCODE'=0) {
  Set ReturnValue=""
  }
 Quit ReturnValue
}
```

注釈　BI.Study.PatientDetails の PatientID フィールドにはインデックスがあります。このことにより、クエリがより早く実行されます。

この例のように、SQL クエリを使用することで最も簡単に関連付けることが可能なテーブルがアプリケーションに含まれている場合、既に関連フィールドにインデックスが存在している可能性が大きくなります。ただし、このようなメソッドを作成する場合は常に、適切なインデックスが存在することを確認する必要があります。

9. クラスを保存して、コンパイルします。
10. ターミナルで、以下のようにメソッドをテストします。

```
write ##class(Tutorial.Cube).GetFavoriteColor("SUBJ_100301")
```

11. アーキテクトにアクセスします。
12. 次のようにして、新しいディメンジョン、階層およびレベルを作成します。
    a. [**要素を追加**] をクリックします。
    b. [**新しい要素名を入力**] に、ColorD と入力します。
    c. [**データ・ディメンジョン**] をクリックします。
    d. [OK] をクリックします。

       ディメンジョン、階層およびレベルが生成されます。
    e. レベルの名前を Favorite Color に変更します。
    f. レベルには、[**表現**] に以下のように入力します。

       ```
       ##class(Tutorial.Cube).GetFavoriteColor(%source.PatientID)
       ```

       この式は、インデックス構築時に実行されます。前の手順のパフォーマンスについての注意を参照してください。
    g. [**ファクトテーブルのフィールド名**] に DxFavColor を指定します。

       これで、このレベル定義が生成されたテーブルにどのように影響するかを理解しやすくなります。
13. キューブを保存します。

    クラスをスタジオで編集したため、アーキテクトでは保存された定義をオーバーライドするかどうかを尋ねるダイアログ・ボックスが表示されます。[OK] をクリックします。アーキテクトでは、クラス定義のうちアーキテクトで編集可能な部分のみをオーバーライドします。このため、クラスに追加したメソッドはオーバーライドしません。
14. キューブをコンパイルします。
15. キューブを再構築します。

    ベース・テーブルの各レコードに対して 1 回ずつ、作成したメソッドおよび埋め込み SQL が実行されます。
16. アナライザを開き、新しいレベルを行として表示します。以下のような画面が表示されているはずです。

Favorite Color	
None	2,380
Blue	1,273
Green	1,268
Orange	1,278
Purple	1,308
Red	1,249
Yellow	1,244

17. 必要に応じて、このレベルのレベル・テーブルを開きます。

    a. 前述のように、管理ポータルにアクセスし、サンプルをインストールしたネームスペースに移動します。

    b. **[システムエクスプローラ]→[SQL]** をクリックします。

    c. 左側の領域で、テーブル `Tutorial_Cube.DxFavColor` に移動して、これを開きます。

    以下のように表示されます。

#	ID	DxFavColor
1	1	Purple
2	2	<null>
3	3	Red
4	4	Yellow
5	5	Blue
6	6	Green
7	7	Orange

Complete

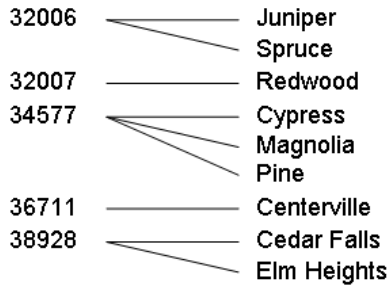
5

Business Intelligence Tutorial : サブジェクト領域の作成

Business Intelligence 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、セキュリティ上の理由およびその他の理由により、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。

5.1 はじめに

このチュートリアルでは、患者のデータを郵便番号によって分割する 2 つのサブジェクト領域を作成します。Patients サンプルの郵便番号には、以下のように小さい市区町村が含まれています。



以下のサブジェクト領域を作成します。

サブジェクト領域名	コンテンツ
Patient Set A	郵便番号 32006、32007、または 36711 に居住する患者
Patient Set B	郵便番号 34577、または 38928 に居住する患者

5.2 サブジェクト領域の作成

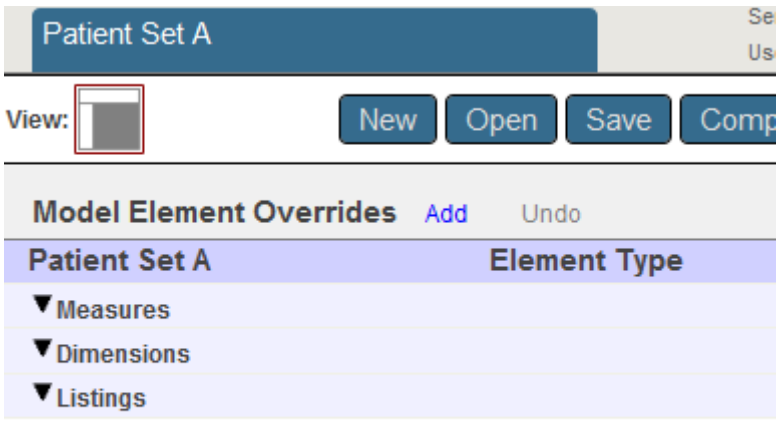
サブジェクト領域を作成する手順は以下のとおりです。

1. アーキテクトで、[新規] 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. [サブジェクト領域の名前] に Patient Set A と入力します。
4. [ベースキューブ] で、[参照] をクリックして [Tutorial] を選択します。
5. [サブジェクト領域のクラス名] に Tutorial.SubjectA と入力します。
6. [OK] をクリックします。

   サブジェクト領域が作成され、クラスが保存されます。

   以下のように表示されます。



   アーキテクトには、フィルタを定義するユーザ・インタフェースがありません。適切なフィルタ式を入力するか、アナライザからコピーして貼り付ける必要があります。

7. 別のブラウザ・タブまたはウィンドウで、アナライザにアクセスして、以下を実行します。

   a. [HomeD] を展開します。

   b. [ZIP Code] を [フィルタ] ボックスにドロップします。これによって、フィルタ・ボックスがピボット・テーブルのすぐ上に追加されます。

   c. そのフィルタ・ボックスで、検索ボタンをクリックし、[32006]、[32007]、および [36711] を選択します。

      チェック・マークをクリックします。

      この操作により、ピボット・テーブルがフィルタ処理されます。

      重要　[32006]、[32007]、および [36711] を個別に [フィルタ] ボックスにドラッグ・アンド・ドロップしないでください。代わりに、ここでの説明に従ってレベルをドラッグし、メンバを選択します。

   d. クエリ文字列ボタン  をクリックします。

      これで、アナライザで使用している MDX クエリを示すダイアログ・ボックスが表示されます。

```
SELECT  FROM [Patients]
%FILTER %OR({[HOMED].[H1].[ZIP Code].&[32006],[HOMED].[H1].[ZIP Code].&[32007],[HOMED].[H1].[ZIP
 Code].&[36711]})
```

   e. %FILTER 以降のテキストをシステムのクリップボードにコピーします。

   f. [OK] をクリックします。

8. アーキテクトで、Patient Set A のラベルが付けられた行をクリックします。
9. [詳細] ペインで、コピーしたテキストを [**フィルタ**] に貼り付けます。

```
%OR({[HOMED].[H1].[ZIP Code].&[32006],[HOMED].[H1].[ZIP Code].&[32007],[HOMED].[H1].[ZIP
Code].&[36711]})
```

10. [**保存**] をクリックして、[OK] をクリックします。
11. サブジェクト領域をコンパイルします。
12. 2 番目のサブジェクト領域についても、前述の手順を繰り返しますが、以下の変更点があります。
    - [**サブジェクト領域の名前**] に Patient Set B と入力します。
    - [**サブジェクト領域のクラス名**] に Tutorial.SubjectB と入力します。
    - 他の 2 つの郵便番号についても前述の手順を繰り返します。そのため、[**フィルタ**] には以下を使用します。

      ```
      %OR({[HOMED].[H1].[ZIP Code].&[34577],[HOMED].[H1].[ZIP Code].&[38928]})
      ```

5.3 サブジェクト領域の検証

次に、作成したサブジェクト領域を検証します。実際に表示される数は、ここでの表示と異なる場合もあります。

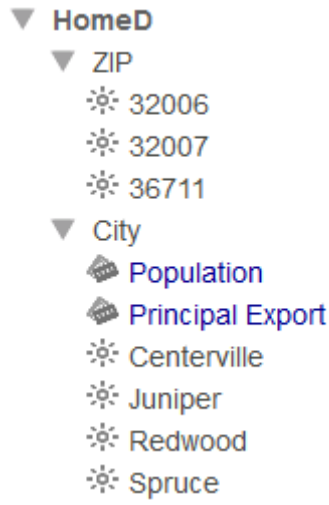
1. アナライザの変更ボタン  をクリックします。
2. [Patient Set A] をクリックします。
3. [OK] をクリックします。

   選択したサブジェクト領域の内容がアナライザに表示されます。

   合計レコード数が、ベース・キューブほど多くないことに注意してください。

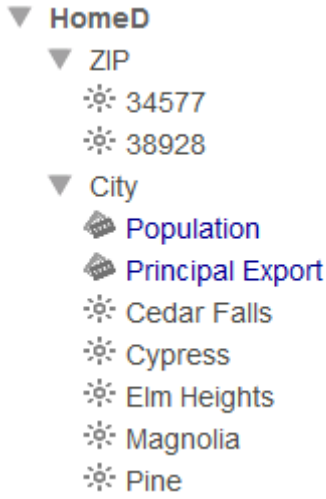
	All
Count	4,490

4. [モデル・コンテンツ] 領域で、[HomeD] ディメンジョン、[ZIP Code] レベル、および [City] レベルを展開します。以下のように表示されます。



5. `Patient Set B` についても前述の手順を繰り返します。

   `[HomeD]` ディメンジョン、`[ZIP Code]` レベル、および `[City]` レベルを展開します。以下のように表示されます。



5.4 一般的なフィルタ式

ここでは、アナライザで一般的なフィルタを試して、生成されたクエリに対する影響を確認します。

1. アナライザで `[Tutorial]` キューブを開きます。

   アナライザでは、キューブとサブジェクト領域の両方をサブジェクト領域と呼びます。これらの形式的な区別は、作成時にのみ関係します。

2. **[新規]** をクリックします。

   アナライザにカウント (レコード数) が表示されます。

	All
Count	10,000

   フィルタを追加する前に、クエリが現在どのように定義されているかを確認し、比較の基盤とします。

3. クエリ文字列ボタン  をクリックします。

   これで、アナライザで使用している MDX クエリを示すダイアログ・ボックスが表示されます。

   ```
   SELECT FROM [TUTORIAL]
   ```

4. [OK] をクリックします。
5. [ColorD] および [Favorite Color] を展開します。
6. [Orange] を [**フィルタ**] にドラッグ・アンド・ドロップします。

   これでアナライザは、好きな色がオレンジの患者のみを使用します。これは、以下のようになります。

	All
Count	1,303

7. クエリ文字列ボタン  をクリックします。

   これで、以下のクエリが表示されます。

   ```
   SELECT FROM [TUTORIAL] %FILTER [COLORD].[H1].[FAVORITE COLOR].&[ORANGE]
   ```

   %FILTER キーワードはクエリを制限します。%FILTER より後の部分はフィルタ式です。

   ```
   [COLORD].[H1].[FAVORITE COLOR].&[ORANGE]
   ```

   このフィルタ式は、メンバ (Favorite Color レベルの Orange メンバ) を参照するため、メンバ式です。メンバはレコード・セットであり、メンバ式はそのレコード・セットを参照します。

   この式はディメンジョン、階層およびレベルの名前を使用します。&[ORANGE] フラグメントは、Orange メンバのキーを参照します。アナライザは名前ではなくキーを使用しますが、メンバ名が一意であれば、どちらを使用してもかまいません。

8. [OK] をクリックします。
9. フィルタに別の色を追加します。そのためには、以下の操作を実行します。
   a. [**フィルタ**] で [Orange] の横にある [X] をクリックします。

      これによって、そのフィルタが削除されます。

   b. [Favorite Color] を [**フィルタ**] にドラッグ・アンド・ドロップします。これによって、フィルタ・ボックスがピボット・テーブルのすぐ上に追加されます。

   c. そのフィルタ・ボックスで、検索ボタンをクリックし、[Orange] と [Purple] を選択します。

   d. チェック・マークをクリックします。

      この操作により、ピボット・テーブルがフィルタ処理されます。

      重要　　[Orange] および [Purple] を個別に [**フィルタ**] ボックスにドラッグ・アンド・ドロップしないでください。代わりに、ここでの説明に従ってレベルをドラッグし、メンバを選択します。

      アナライザの表示は以下のようになります。

	All
Count	2,558

システムは、好きな色がオレンジまたは紫の患者のみを使用するようになりました（オレンジのみの場合よりカウントが増えていることに注意してください）。

10. クエリ・テキストを再度表示します。以下のように表示されます。

```
SELECT FROM [TUTORIAL]
%FILTER %OR({[COLORD].[H1].[FAVORITE COLOR].&[ORANGE],[COLORD].[H1].[FAVORITE COLOR].&[PURPLE]})
```

この場合、フィルタ式は以下のようになります。

```
%OR({[COLORD].[H1].[FAVORITE COLOR].&[ORANGE],[COLORD].[H1].[FAVORITE COLOR].&[PURPLE]})
```

%OR 関数はインターシステムズの最適化関数です。この関数の引数はセットです。

セットは中括弧 {} で囲まれ、要素のコンマ区切りリストから構成されます。この場合、セットには 2 つのメンバ式が含まれています。セット式は、セットの要素で示されたすべてのレコードを参照します。この場合、セットは、好きな色がオレンジであるすべての患者と、好きな色が紫であるすべての患者を参照します。

11. [OK] をクリックします。
12. フィルタのドロップダウン・リストを使用して、[Purple] の横にあるチェック・ボックスのチェックを外します。

これでアナライザは、好きな色がオレンジの患者のみを使用します。

13. [AllerD] および [Allergies] を展開します。
14. [mold] を [**フィルタ**] の [Orange] の下にドラッグ・アンド・ドロップします。

アナライザの表示は以下のようになります。

	All
Count	52

Orange のみを使用した場合よりカウントが少ないことに注意してください。このピボット・テーブルには、好きな色がオレンジで、カビに対するアレルギーがある患者のみが表示されます。

15. クエリ・テキストを再度表示します。以下のように表示されます。

```
SELECT FROM [TUTORIAL]
%FILTER NONEMPTYCROSSJOIN([ALLERD].[H1].[ALLERGIES].&[MOLD],[COLORD].[H1].[FAVORITE COLOR].&[ORANGE])
```

この場合、フィルタ式は以下のようになります。

```
NONEMPTYCROSSJOIN([ALLERD].[H1].[ALLERGIES].&[MOLD],[COLORD].[H1].[FAVORITE COLOR].&[ORANGE])
```

MDX 関数 NONEMPTYCROSSJOIN は、2 つのメンバを結合して、生成されたタプルを返します。このタプルは、指定されたメンバの両方に属するレコードにのみアクセスします。

これで、最も一般的な以下の 3 つのフィルタ式を確認しました。

メンバ式

メンバ式をフィルタとして使用すると、システムはそのメンバに属するレコードにのみアクセスします。

メンバ式は以下のように作成できます。

```
[dimension name].[hierarchy name].[level name].&[member key]
```

または以下のようにします。

```
[dimension name].[hierarchy name].[level name].[member name]
```

以下はその説明です。

- dimension name は、ディメンジョン名です。
- hierarchy name は階層名です。階層名は省略できます。省略すると、クエリでは、このディメンジョン内の定義に従って、指定された名前の最初のレベルが使用されます。
- level name は、その階層内のレベル名です。レベル名は省略できます。省略すると、クエリでは、このディメンジョン内の定義に従って、指定された名前の最初のメンバが使用されます。
- member key は、指定されたレベル内のメンバのキーです。多くの場合、これはメンバ名と同じです。
- member name は、指定されたレベル内のメンバの名前です。

セット式

メンバ・セットをフィルタとして使用すると、システムは指定のメンバのいずれかに属するレコードにアクセスします。つまり、メンバは論理 OR で結合されています。

メンバを参照するセット式は、以下のように作成できます。

```
{member_expression,member_expression,member_expression...}
```

member_expression はメンバ式です。

タプル式

タプルをフィルタとして使用すると、システムは指定のメンバのすべてに属するレコードにアクセスします。つまり、メンバは論理 AND で結合されています。

タプル式は以下のように作成できます。

```
NONEMPTYCROSSJOIN(member_expression,member_expression)
```

または以下のようにします。

```
(member_expression,member_expression)
```

その他のバリエーションについては、"InterSystems MDX の使用法" および "InterSystems MDX リファレンス" を参照してください。

6

Business Intelligence チュートリアル：ピボット・テーブルおよびダッシュボードの作成とパッケージ化

1 つ以上の Business Intelligence キューブを作成したら、通常は、最初のピボット・テーブルおよびダッシュボードのセットを作成およびパッケージ化します。また、通常ユーザは、必要に応じて新しいピボット・テーブルとダッシュボードを作成します。

ここでは、ピボット・テーブルとダッシュボードの作成プロセスを簡単に説明します。

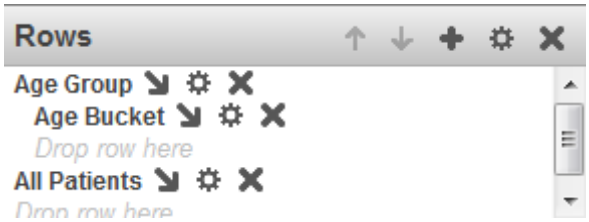
ピボット・テーブルおよびダッシュボードの作成方法の詳細は、"アナライザの使用法" および "ダッシュボードの作成" を参照してください。

6.1 ピボット・テーブルの作成

このチュートリアルで前述した手順では、Patients キューブを使用するピボット・テーブルを作成しました。ここでは、作成した新しいキューブを使用するピボット・テーブルを作成します。

1. アナライザにアクセスします。
2. [Tutorial] キューブに移動します。
3. [モデル・コンテンツ] ペインで、[AgeD] ディメンジョンを展開します。
4. [Age Group] を [行] にドラッグ・アンド・ドロップします。
5. [Age Bucket] を [行] にドラッグして、内訳を示すボタン  上にドロップします。
6. [Count] を [列] にドラッグ・アンド・ドロップします。
7. [All Patients] を下部の [行] にドラッグ・アンド・ドロップします。

   すると、[行] ボックスは以下のようになります。



ピボット・テーブルは以下のようになります。

Age Group		Count
0 to 29	0 to 9	1,459
	10 to 19	1,428
	20 to 29	1,335
30 to 59	30 to 39	1,524
	40 to 49	1,520
	50 to 59	1,090
60+	60 to 69	688
	70 to 79	604
	80+	352
All Patients		10,000

8. **[保存]** をクリックします。

   ピボット・テーブルの名前を指定できるダイアログ・ボックスが表示されます。

   名前を付けて、ピボット・テーブルを保存します。ここでは、データ取得の基本となるクエリを、選択した方法での表示に必要な情報と共に保存します。データを保存するわけではありません。

9. **[フォルダ]** に、`Tutorial` と入力します。
10. **[ピボット名]** に `Patients by Age Group` と入力します。
11. [OK] をクリックします。
12. 以下のようにもう 1 つピボット・テーブルを作成します。
    - **[行]** には `[Diagnoses]` を使用します。
    - **[列]** には `[Count]` および `[Avg Age]` を使用します。
    - **[フォルダ]** には、`[Tutorial]` を選択するか、入力します。
    - **[ピボット名]** に `Patients by Diagnosis` と入力します。

6.2 ダッシュボードの作成

ここでは、単純なダッシュボードを作成します。

1. ページ上部の [Analytics] リンクをクリックします。
2. **[ホーム]**→[Analytics]→**[ユーザポータル]** をクリックします。

   ユーザ・ポータルが表示され、既存のパブリック・ダッシュボードおよびピボット・テーブルがリストされます。

3. **[メニュー]→[新規ダッシュボード]** をクリックします。

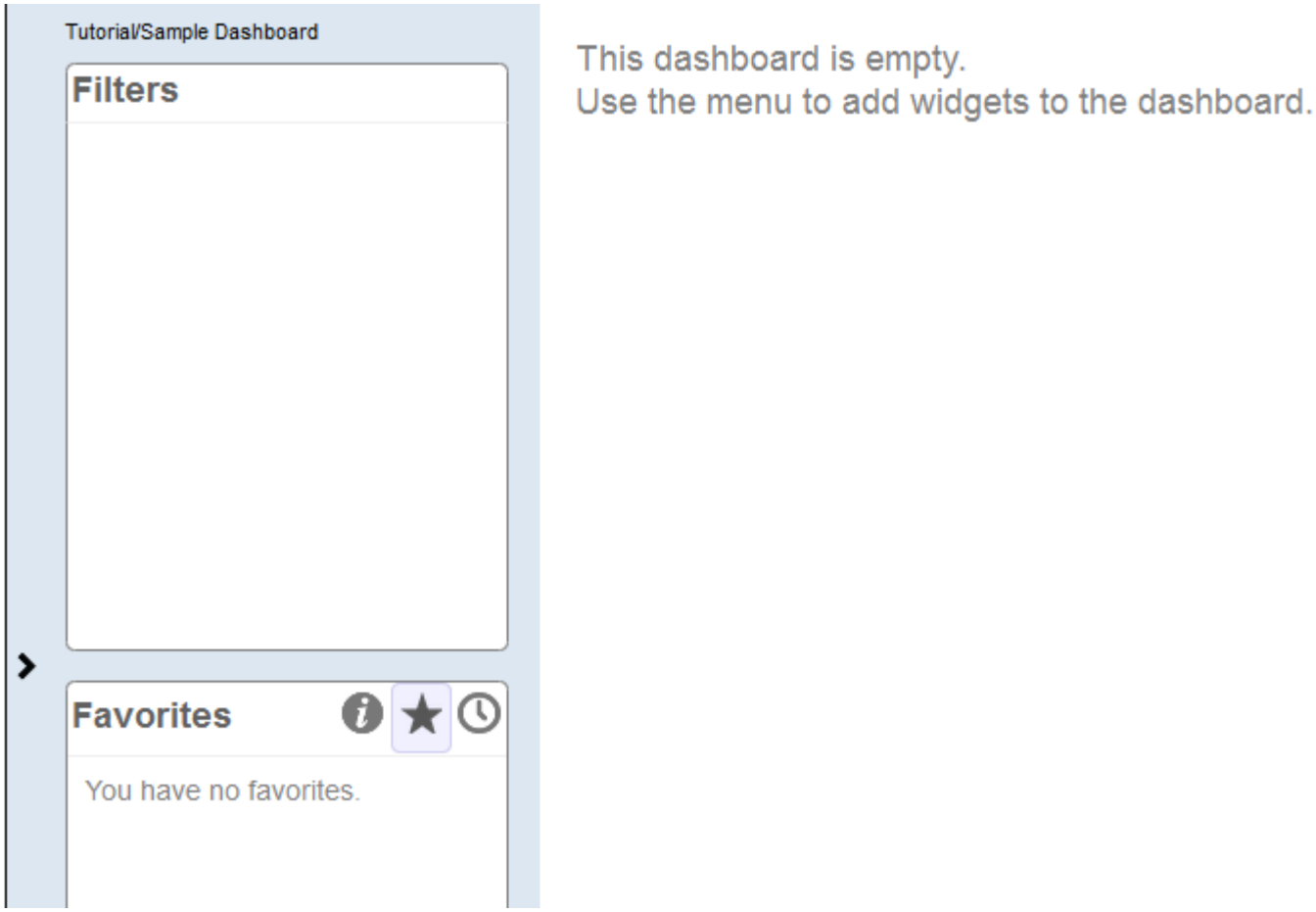
   新しいダッシュボードに関する基本情報の入力を求めるダイアログ・ボックスが表示されます。

4. **[フォルダ]** には、`[Tutorial]` を選択するか、入力します。

5. **[ダッシュボード名]** に、`Sample Dashboard` と入力します。

6. [OK] をクリックします。

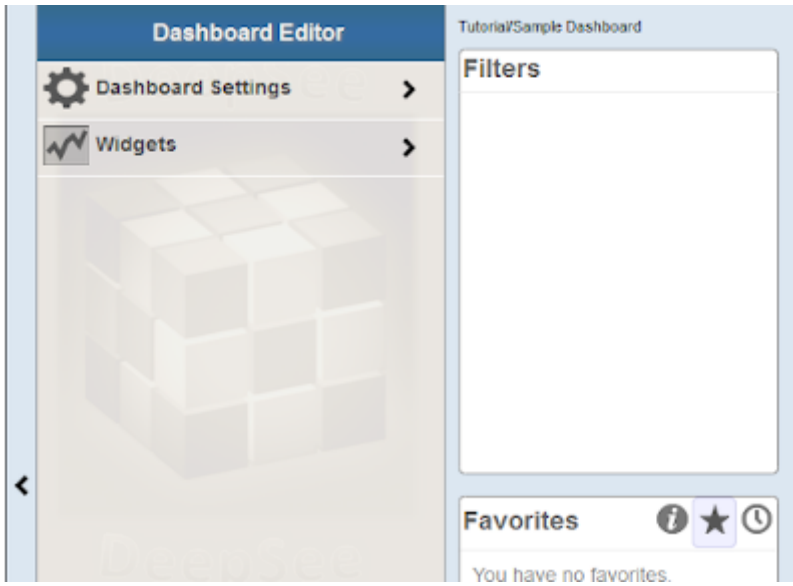
   システムによってダッシュボードが作成され、保存されて、表示されます。最初は空の状態です。



   ダッシュボードの左側にある [>] ボタンを確認します。

7. [>] ボタンをクリックします。

   すると、以下のようにダッシュボード・エディタが展開されます。



8. **[ウィジェット]** をクリックします。

   すると、現在このダッシュボードに含まれているすべてのウィジェットのリストが表示されます (この場合はなし)。

9. [+] ボタンをクリックします。

   すると、ダイアログ・ボックスが表示され、ここから初期オプションを選択できます。

10. ダイアログ・ボックスで、**[ピボットとグラフ]** が展開されていない場合はこれをクリックします。リストが展開され、選択肢リストが表示されます。

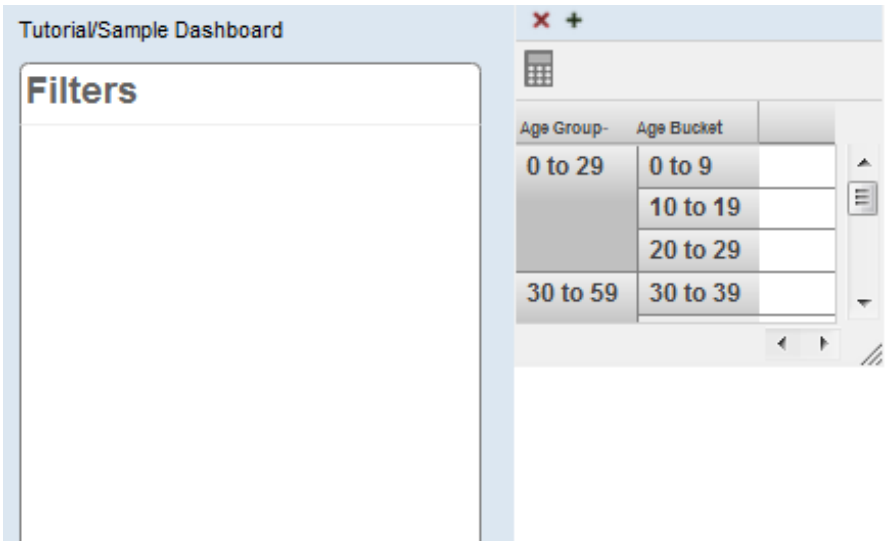
11. このリストで、**[テーブル]** をクリックします。

12. **[データソース]** の横にある検索ボタン  をクリックします。

13. `[Tutorial]→[Patients by Age Group]` をクリックします。

14. [OK] をクリックして、ウィジェットを追加します。

    ダッシュボードが以下のように表示されます。



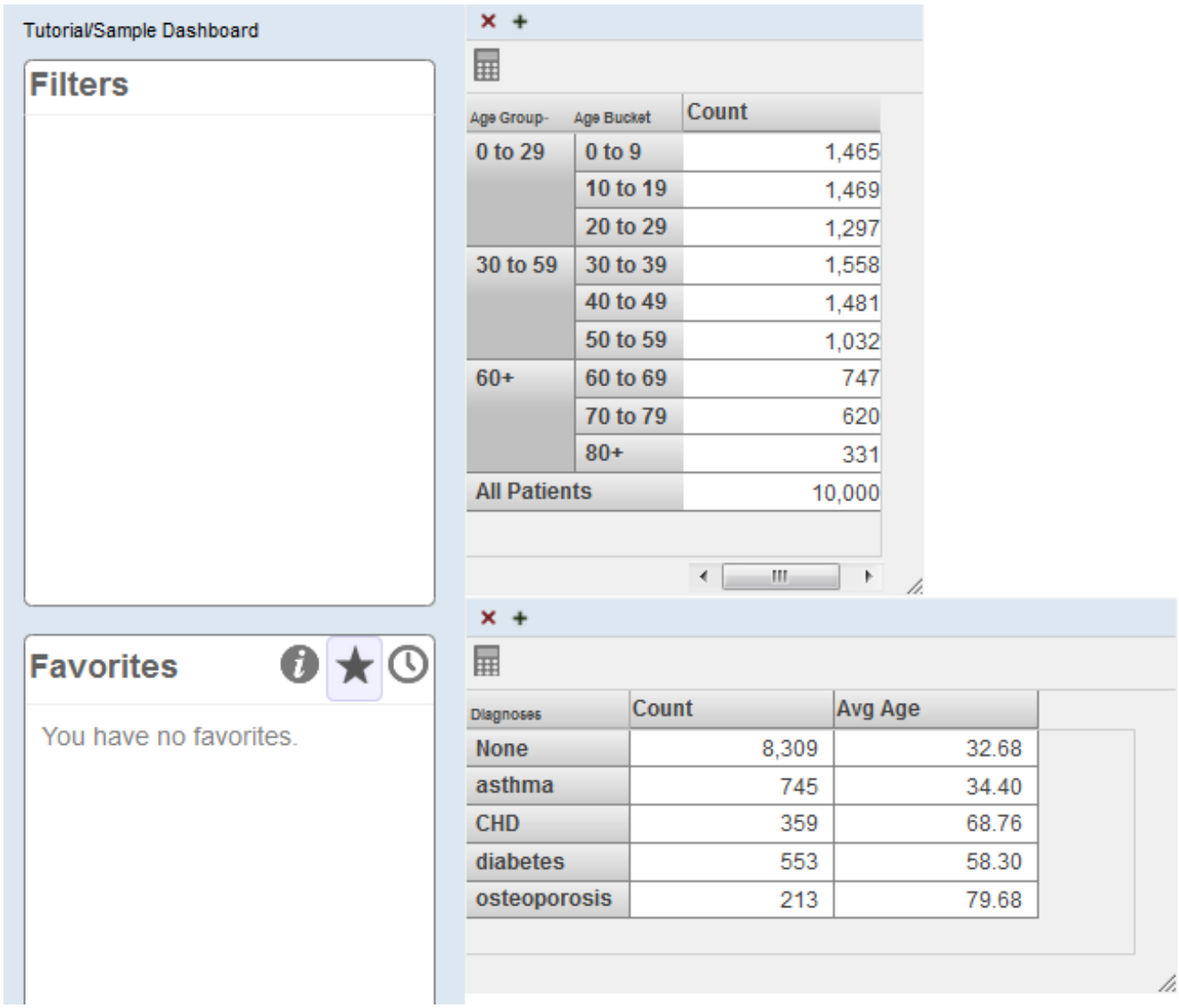
15. 前述の手順を繰り返して、前のセクションで作成したその他のピボット・テーブルを追加します。

新しく追加したウィジェットは、同じ既定の位置 (左上) に置かれるため、前に追加したウィジェットを覆うようになります。

16. 新しく追加したウィジェットのタイトル・バーにカーソルを置いて、他のウィジェットの下にこのウィジェットをドラッグ・アンド・ドロップします。

    サイズを変更できるように、上方のウィジェットの下にスペースをとることをお勧めします。

17. 各ウィジェットの右下角にあるサイズ変更コントロールを使用して、スクロールしなくてもすべての行が表示されるようにサイズを変更します。



18. **[メニュー]→[保存]** をクリックします。

    ダッシュボードが即座に保存されます。

19. 以下の手順を実行して、このダッシュボードにフィルタ・コントロールを追加します。

    a. ダッシュボード・エディタを開きます。

    b. **[ウィジェット]** をクリックします。

    c. `Patients by Age Group` ウィジェットをクリックします。

    d. **[コントロール]** をクリックします。

ダッシュボード・エディタに、選択したウィジェットで定義されているすべてのコントロールのリスト（現在は空）が表示されます。

e. リストの上の [+] ボタンをクリックします。

コントロールを指定するダイアログ・ボックスが表示されます。

f. **[場所]** で、**[ダッシュボード]** を選択します。

g. **[ターゲット]** に、* と入力します。

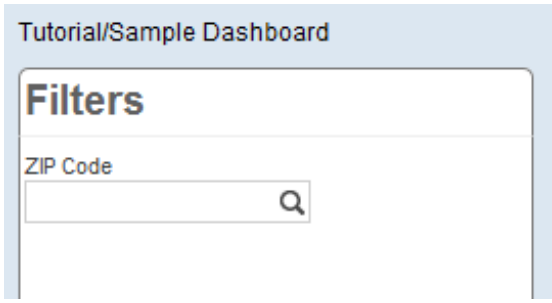
h. **[アクション]** で、**[フィルタを適用]** を選択します。

i. **[フィルタ]** で、[ZIP Code] を選択します。

j. **[ラベルまたはアイコンを制御]** に、ZIP Code と入力します。

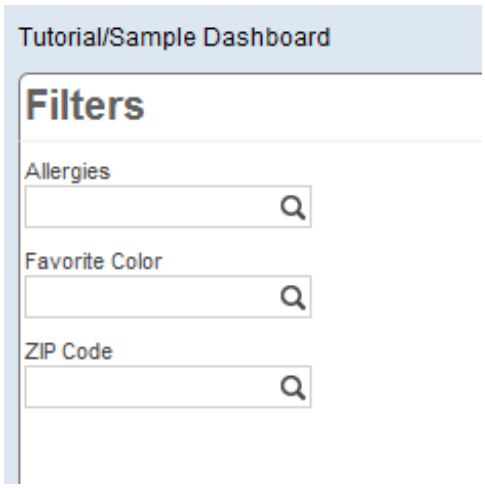
k. [OK] をクリックして、コントロールを追加します。

これでダッシュボードの左上角の表示が以下のようになります。



20. 前述の手順を繰り返し、Allergies および Favorite Color のフィルタを追加します。

これでダッシュボードの左上角の表示が以下のようになります。



21. これらのフィルタ・コントロールを再構成して、それぞれに既定値を設定します。ZIP Code フィルタを再構成する手順は以下のとおりです。

a. ダッシュボード・エディタを開きます。

b. Patients by Age Group ウィジェットの定義にアクセスします。

c. **[コントロール]** をクリックします。

d. [ZIP Code] コントロールをクリックします。

e. [デフォルト値] に、32007 と入力し、チェック・マークをクリックします。

f. [OK] をクリックします。

Allergies には既定値 soy を使用します。

Favorite Color には既定値 blue を使用します。

これらの変更を行うと、変更内容が自動的に保存されます。

22. ダッシュボードをテストして、以下を検証します。

- ブラウザの更新ボタンを使用すると、各フィルタに正しい既定値が表示されます。
- 各フィルタが両方のウィジェットに影響します。

6.3 ピボット・テーブルおよびダッシュボードのエクスポートとパッケージ化

1. [メニュー]→[管理ポータル] をクリックします。

   このオプションが表示されない場合は、現在、InterSystems IRIS Business Intelligence 開発ツールに直接アクセスできないユーザとしてログインしていることになります。この場合は、Business Intelligence に再度ログインします。

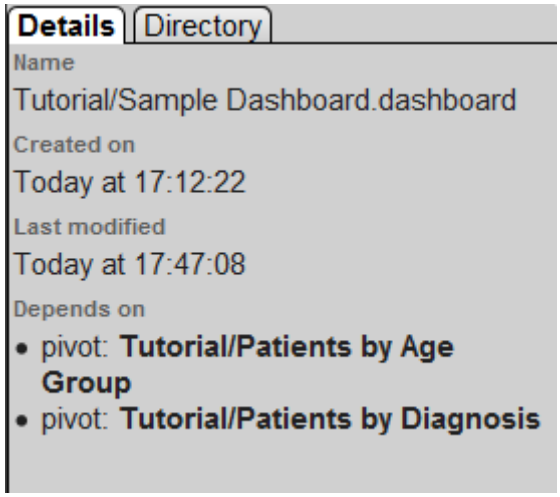
2. [Analytics]→[管理]→[フォルダマネージャ] をクリックします。

   [フォルダマネージャ] が表示されます。

3. [Tutorial] フォルダにスクロールします。これは以下のように表示されます。

   右側の列は、これらの項目を作成した InterSystems IRIS® データ・プラットフォーム・ユーザを示します。ここに表示される値は異なる場合もあります。

4. Sample Dashboard の左側にあるチェック・ボックスにチェックを付けます。これでページの左の領域に、このダッシュボードで使用されるピボット・テーブルがリストされます。



これらのピボット・テーブルはいずれもダッシュボード自体と同じフォルダにありますが、このことは必須ではありません。

5. 2 つの新しいピボット・テーブルの左側にあるチェック・ボックスにもチェックを付けます。
6. **[サーバーディレクトリ]** に、書き込み権限がある既存のディレクトリの名前を入力します。
7. **[エクスポート]** をクリックします。

   そのディレクトリに 3 つのファイルが書き込まれます。

   これらのファイルは定義した項目のパッケージとして使用できますが、次の手順では、より便利なアプローチを使用します。
8. スタジオで `Tutorial.DashboardsAndPivots` という新しいクラスを作成します。このクラスは **%DeepSee.UserLibrary.Container** を拡張する必要があります。
9. この新しいクラスで、以下のように `Contents` という名前の XData ブロックを追加します。

   Class Member

   ```
   XData Contents [ XMLNamespace = "http://www.intersystems.com/deepsee/library" ]
   {
   <items>
   </items>
   }
   ```

10. 新しいクラスを保存します。
11. ダッシュボードおよびピボット・テーブルの定義を、エクスポートしたファイルから XData ブロックにコピーします。
    a. テキスト・エディタを使用してダッシュボード・ファイル (`Tutorial-Sample_Dashboard.xml`) を開きます。これは以下のように表示されます。

       ```
       <?xml version="1.0" encoding="UTF-8"?>
       <dashboard xmlns="http://www.intersystems.com/deepsee/library" folderName="Tutorial" name="Sample
        Dashboard" ...
       ...
       </dashboard>
       ```

    b. このファイルのコンテンツを、2 行目からファイルの最後までコピーします。つまり、1 行目はコピーしません。これは XML 宣言です。

c. コピーしたテキストを、XData ブロックの <items> と </items> の間に貼り付けます。これで、以下のように表示されます。

```
XData Contents [ XMLNamespace = "http://www.intersystems.com/deepsee/library" ]
{
<items>
   <dashboard xmlns="http://www.intersystems.com/deepsee/library" folderName="Tutorial"
name="Sample Dashboard" ...
   ...
   </dashboard>
</items>
}
```

d. テキスト・エディタでピボット・テーブル・ファイルのいずれかを開きます。これは（<dashboard> ではなく <pivot> のある）類似した構造を持っています。

e. このファイルのコンテンツを、2 行目からファイルの最後までコピーします。

f. コピーしたテキストを、XData ブロックの </dashboard> と </items> の間に貼り付けます。これで、以下のような表示になります。

```
XData Contents [ XMLNamespace = "http://www.intersystems.com/deepsee/library" ]
{
<items>
   <dashboard xmlns="http://www.intersystems.com/deepsee/library" folderName="Tutorial"
name="Sample Dashboard" ...
   ...
   </dashboard>
  <pivot xmlns="http://www.intersystems.com/deepsee/library" folderName="Tutorial" name="Patients
 by Age Group" ...
   ...
   </pivot>
</items>
}
```

注釈　代わりに、コピーしたテキストを <items> と <dashboard> の間に挿入することもできます。項目の順序はどこにも影響を与えません。

g. その他のピボット・テーブル・ファイルについても前述の手順を繰り返します。

h. クラス定義を保存します。

12. フォルダマネージャに戻って、ページを更新します。

    これで、選択していたものがすべてクリアされます。

13. Tutorial フォルダの 3 つの項目のそれぞれの横にあるチェック・ボックスにチェックを付けます。

14. [削除] をクリックして [OK] をクリックし、確定します。

15. スタジオで、作成したクラスをコンパイルします。

    これを実行すると、そのクラスに格納されているダッシュボードとピボット・テーブルの定義がインポートされます。

16. フォルダマネージャに戻って、ページを更新します。チュートリアルのダッシュボードおよびピボット・テーブルが再度使用可能になっています。

このようなコンテナ・クラスには必要な数だけダッシュボードとピボット・テーブルを組み込むことができます。また、複数のコンテナ・クラスを持つこともできます。これらのコンテナでは、ダッシュボードおよびピボット・テーブルを特定の方法で構成する必要はありません。例えば、ピボット・テーブルを、それを使用するダッシュボードと同じコンテナに配置する必要はありません。また、これらのコンテナ・クラスを作成する必要はまったくありません。