Сайты про MDX:

mdx.far.ru

sqlmdx.narod.ru

ЯЗЫК ЗАПРОСОВ MDX

Калюжный О.Н.,

oleg212121@yandex.ru

Введение

Язык MDX был впервые представлен как составляющая *OLE DB for OLAP* в 1997г. компанией Microsoft. MDX создан группой разработчиков *SQL Server* (Mosha Pasumansky и другие). Вскоре последовала коммерческая реализация языка в *Microsoft OLAP Services 7.0* (1998г.), затем - в *Microsoft Analysis Services*. Последняя версия *OLE DB for OLAP* была выпущена компанией Microsoft в 1999г.

Несмотря на то, что MDX является не общим стандартом, а только внутренней спецификацией Microsoft, он был принят многими ведущими распространителями технологии OLAP. В их числе разработчики серверных приложений, такие, как Applix, Microstrategy, SAS, SAP, Whitelight, NCR, а также разработчики клиентских приложений: Panorama Software, Proclarity, AppSource, Cognos, Business Objects, Brio Technology, Crystal Reports, Microsoft Excel, Microsoft Reporting Services и другие.

С появлением *XML for Analysis*, в котором MDX был принят как стандартный язык запросов, все большее число компаний (в их числе, например, Hyperion Solutions), стали поддерживать MDX.

XML for Analysis обращается к *OLE DB for OLAP* с помощью языка запросов MDX. В приложении *Analysis Services 2005* компанией Microsoft были добавлены некоторые расширения MDX, такие, как подзапросы (*subselects*). В таких продуктах, как *Microsoft Excel 2007*, эти расширения уже применяются.

(введение переведено с Wikipedia.org)

Основные объекты многомерных баз данных

Основные элементы структуры многомерных баз данных – ***Кубы (Cubes), Измерения (Dimensions) и Меры (Measures).***

Куб (Cube) — совокупность данных, имеющая многомерную структуру. Куб представлен набором мер и измерений. Измерений может быть много, но, т.к. нам легче представить себе трехмерный объект, такой объект назвали Кубом.

Рассмотрим эти понятия на примере базы данных клиентов.

Пусть наш куб содержит данные клиентов: ФИО, дату рождения, место рождения, пол, место жительства, а также календарь и список населенных пунктов. В случае реляционной базы данных можно было организовать 3 таблицы: [Населенные пункты], [Календарь] и [Паспортные данные]; причем в [Паспортных данных] присутствовали бы поля, связанные с соответствующими полями из [Календаря] и [Населенных пунктов]. В многомерном случае можно задать, например, такие ***измерения***: [Клиенты], [Дата], [Место], [Тип места] *(рождения или жительства)*, [Пол].

На пересечении этих измерений зададим некоторые агрегированные величины - ***меры***. Например: [Количество клиентов], [Максимальный возраст].

Пример (*): Сколько клиентов мужского пола проживает в Твери? Ответ можно получить, задав следующий MDX-запрос:

```
SELECT { [Место].[РФ].[Тверь] } ON COLUMNS,
    { [Пол].[M] } ON ROWS
    FROM [Наш куб данных]
    WHERE ([Measures].[Количество клиентов] , [Тип места].[Место жительства])
```

В данном запросе присутствуют лишь 3 измерения: [Место], [Тип места] и [Пол]; все измерения, не указанные в запросе ([Клиенты] и [Дата]), присутствуют в нем неявно.
Следует отметить, что совокупность мер является, по сути, еще одним измерением.

Как видим, синтаксис MDS очень похож на синтаксис SQL.
К синтаксису запросов вернемся позднее, а пока рассмотрим другие важные понятия.

Члены измерений (Members). Иерархии (Hierarchies).

В рассмотренном выше примере мы пользовались измерением [Место], детализируя его с помощью заданной на нем иерархии. Делается это с помощью ***членов измерения***, т.е. единиц описания на различных уровнях иерархии.
Примеры членов измерения: [РФ], [Белоруссия], [Минск], [ул. Северная], [дом №70], [дом №95] и т.д.

До каждого члена можно «добраться», описав путь к нему по иерархической структуре.
Наример:
[Место]. [Иерархия_Место].[РФ].[Тверь].[Петербургское шоссе].[дом №777].[квартира №8888],
[Место]. [Иерархия_Место].[РФ].[Москва]

Замечание: в приведенных примерах использована подробная запись:
[<Измерение>].[<Иерархия>].[<Член верхнего уровня>]. … .[<Член нижнего уровня>]
В некоторых случаях *уникальные в пределах измерения* элементы такой записи можно опускать.

Можно организовать ***несколько иерархий*** для одного измерения, *например*:

[Дата].[Иерархия1].[<Год>].[<Месяц>].[<Число>] и

[Дата].[Иерархия2].[<№ недели>].[<День недели>]

Замечание: в случае единственной иерархии на нее можно сослаться по имени измерения, то есть:
[Дата].[Иерархия1].Members эквивалентно [Дата].Members ,
[Место].[Иерархия_Место].[Города].Members эквивалентно [Место].[Города].Members .

Каждому ***уровню иерархии (Level)*** присвоено свое имя. Имена уровней применяются в конструкциях вида:
[<Измерение>].[<Иерархия>].[<Уровень>].Members .

Например: [Дата].[Иерархия1].[Месяца].Members – все возможные месяца;
[Место].[Иерархия_Место].[Улицы].Members – ***все*** улицы, независимо от страны и города.

Кортежи (Tuples) и множества (Sets)

Кортеж — это набор членов одного или нескольких разных измерений. Задавая кортеж, мы указываем на конкретную ячейку или набор ячеек внутри куба. Таким образом, кортеж – это декартово произведение (т.е. *пересечение*) множеств, определенных на различных измерениях куба.

Кортежи задаются с помощью круглых скобок.

Примеры:

([Место].[РФ].[Рязань].[улица Есенина], [Пол].[Ж], [Тип места].[Место жительства]) – все дамы с улицы Есенина в Рязани;

([Место].[РФ].[Тамбов], [Тип места].[Место рождения]) – все клиенты, родившиеся в Тамбове.

В языке MDX пересечения множеств реализуются с помощью конструкций CrossJoin <Кортеж>, Where <Кортеж> и других, что обусловлено необходимостью представить результат в удобном для восприятия виде.

Множество (или Набор) — это совокупность (*объединение*) кортежей, определенных с использованием одинакового количества одних и тех же измерений.

Примеры:

{ [Дата].[1960].[Январь], [Дата].[1960].[Февраль] } – все клиенты, родившиеся в январе ***или*** в феврале 1960г.

{ ([Место].[РФ].[Ростов] , [Тип места].[Место жительства] , [Дата],[1980]) , ([Место].[РФ].[Воронеж] , [Тип места].[Место жительства] , [Дата],[1980]) } - все клиенты 1980 года рождения, проживающие в Ростове или в Воронеже.

Множество заключается в фигурные скобки.

Пересечение кортежа или множества с какой-либо мерой дает значение меры на данном множестве. В первом разобранном нами примере (*) мы имели пересечение кортежа ([Место].[РФ].[Тверь], [Пол].[М]) с мерой [Measures].[Количество клиентов].

Член по умолчанию (default member)

Как уже упоминалось, что в большинстве случаев в кортежах явно присутствуют не все измерения. Чтобы получить значение меры на кортеже, нужно учесть все прочие измерения. Для этого у каждого измерения существует член по умолчанию. Как правило, в роли Default Member выступает *единственный* член специального уровня иерархии [All], автоматически создаваемого при создании измерения. Этот уровень содержит совокупные результаты по всему измерению.

Пример:

([Клиенты].[All].Members, [Measures].[Максимальный возраст]) – этот кортеж возвращает возраст самого старшего клиента.

Эквивалентное указание на член по умолчанию можно записать короче и корректнее с помощью выражения DefaultMember:

[Клиенты].DefaultMember .

Если уровень [All] отсутствует, в его роли выступает первый член следующего уровня. Например, для нашего измерения Дата это будет [Дата].[1900], если в кубе нет более ранних годов.

В некоторых системах членом по умолчанию можно назначить любой член или MDX-выражение измерения.

Поскольку совокупность мер ([Measures]) тоже является одним из измерений, то и среди мер есть элемент по умолчанию.

Предположим, что в нашем случае мерой по умолчанию является [Количество клиентов]. Тогда, если из нашего запроса (*) удалить конструкцию WHERE ([Measures].[Количество клиентов]), результат его выполнения останется прежним.

Общий вид запроса

Простейший вид запроса MDX выглядит следующим образом:

```
SELECT <множество1> ON COLUMNS,
<множество2> ON ROWS
FROM <куб>
WHERE <кортеж>
```

На каждой из ***осей*** (columns, rows и др.) можно располагать несколько измерений, например:

```
SELECT { CrossJoin ( { [Место].[РФ].[Урюпинск] }, { [Дата].[1970].[Февраль]  } ) } ON COLUMNS,
{ [Клиенты].Children } ON ROWS
FROM [Наш куб данных]
WHERE ([Тип места].[Место жительства], [Measures].[Количество клиентов])
```

Этот запрос возвращает всех клиентов, проживающих в Урюпинске, родившихся в феврале 1970г.

Кортежи (множества), указанные на осях и после WHERE, образуют декартово произведение, которое и является результатом выполнения запроса.

Множества, входящие в кортеж после слова WHERE, называются ***Срезами (Slicers)***: (Slicer1, … , SlicerN).

Располагать члены одного измерения по разным осям отчета нельзя, такая операция лишена смысла.

Слово CrossJoin не несет логической нагрузки, оно лишь указывает на порядок вывода данных разных измерений.

Функции навигации

Members, AllMembers

Функция Members применяется к иерархиям *или* к уровням.

При применении *к иерархии* функция возвращает набор *всех* членов иерархии, *независимо от уровня*.
Пример: [Дата].[Иерархия1].Members – возвращает полный перечень всех годов, месяцев и дней.

При применении *к уровню* функция возвращает набор всех членов измерения, находящихся *на данном уровне*.
Пример: [Место].[Иерархия_Место].[Города].Members – возвращает полный перечень всех городов.

Функция AllMembers работает аналогично функции Members, но Members возвращает все элементы иерархии, кроме вычисляемых, а AllMembers возвращает также и вычисляемые элементы.

PrevMember, NextMember

Для перемещения в пределах одного уровня, используются функции PrevMember и NextMember:

[Дата].[2009].[Март].NextMember – возвращает апрель 2009 года,

[Дата].[2009].[Март].PrevMember – возвращает февраль 2009 года,

[Дата].[2009].[Март].PrevMember.PrevMember – возвращает январь 2009 года.

Для более компактной записи применяются функции ***Lag(.) и Lead(.)***:

[Дата].[2009].[Март]. Lag(2) – возвращает январь 2009 года,

[Дата].[2009].[Март]. Lead(5) – возвращает август 2009 года,

[Дата].[2009].[Март]. Lag(-1) – возвращает апрель 2009 года.

Children, Parent

Для перемещения вверх и вниз по уровням используются функции Children и Parent:

[Дата].[2009].[Март].Children – возвращает все дни марта,

[Дата].[2009].[Март].Parent – возвращает [Дата].[2009],

[Дата].[2009].[Март].[25].Parent.Parent – возвращает [Дата].[2009] .

Функции ***FirstChild, LastChild*** возвращают первого (последнего) потомка данного элемента.