Глава 3

Введение в MDX

В главе 2 вы запускали простой MDX-запрос для извлечения данных из Analysis Services 2008. В этой главе вы изучите фундаментальные концепции, лежащие в основе языка MDX, а также научитесь управлять многомерными объектами в Analysis Services с помощью конструкций языка MDX. Текущая глава заложит фундамент для большинства последующих глав книги; фактически в некоторых разделах этой главы и на протяжении всей книги показывается, как взаимодействия между клиентскими инструментами и экземпляром Analysis Services приводят к генерированию кода MDX. Вы не только увидите генерируемый в этих случаях код MDX, но также сможете понять, что выполняет этот код.

SQL Server 2008 предлагает пример проекта Analysis Services, который содержит большинство функций, поддерживаемых Analysis Services 2008. В данной главе для изучения MDX вам придется использовать пример проекта Analysis Services, который можно получить на веб-сайте `www.codeplex.com`. Предлагаемые вам иллюстрации ограничены тремя измерениями, чтобы вам легче было понять концепции. Вы можете расширить эти концепции, если хотите просматривать данные с использованием дополнительных измерений. Кроме того, вы узнаете, как создавать MDX-выражения и MDX-запросы для анализа данных из баз данных Analysis Services.

Что такое MDX

Подобно тому, как SQL (Structured Query Language — язык структурированных запросов) представляет собой язык создания запросов для извлечения данных из реляционных баз данных, MDX (Multi-Dimensional eXpressions — язык многомерных выражений) является языком запросов, используемым для извлечения данных из многомерных баз данных. Точнее, MDX используется для запрашивания данных из баз данных OLAP с помощью Analysis Services и поддерживает два особых режима. При использовании в качестве MDX-выражений позволяет определять многомерные объекты и данные для вычисления значений, а также управлять ими. Как язык запросов он используется для извлечения данных из баз данных Analysis Services. Изначально MDX был разработан компанией Microsoft и был введен вместе с SQL Server Analysis Services 7.0 в 1998 году.

Использование языка MDX не ограничено авторскими правами на продукт Analysis Services. Этот язык используется для извлечения информации из баз данных OLAP; и он основан на стандартах. Язык является частью спецификации OLEDB для OLAP, финансируемой Microsoft; он поддерживается также многими другими провайдерами OLAP, включая Intelligence Server компании Microstrategy, Essbase Server компании Hyperion и Enterprise BI Server от SAS. Некоторые компании стремятся расширить стандарт и обеспечить дополнительную функциональность, и вследствие этого расширения MDX действительно разработаны индивидуальными производителями. В расширениях MDX реализованы функции, которых в стандарте нет, но предполагается, что все компоненты таких расширений MDX разработаны согласно стандарту. Analysis Services 2008 предлагает несколько

расширений к стандарту MDX, определяемых спецификацией OLE DB для OLAP. В этой книге вы узнаете о форме MDX, поддерживаемой Analysis Services 2008.

Ссылка на MDX может означать либо язык MDX-запросов, либо выражения MDX. Хотя язык MDX-запросов использует синтаксис, подобный синтаксису языка SQL, они значительно отличаются. Однако мы будем использовать SQL, чтобы пояснить вам некоторые понятия из MDX. Прежде чем вы начнете углубляться в освоение языка MDX-запросов и MDX-выражений, вам необходимо изучить некоторые фундаментальные концепции.

Фундаментальные концепции

Основой многомерной базы данных является куб. Каждый куб обычно содержит более двух измерений. Куб Adventure Works в учебной базе данных содержит 21 измерение. Примеры SQL Server 2008 можно загрузить из `http://www.codeplex.com/MSFTDBProdSamples`. Найдите и загрузите файл `SQL2008.AdventureWorks_DW_BI_v2008<`*архитектура*`>.msi` и инсталлируйте на вашей машине в зависимости от архитектуры вашей машины x86, x64 или ia64. Этот пакет содержит пример реляционной БД `AdventureWorksDW2008` и проект `AdventureWorks` для Analysis Services. Для установки примера необходимо задать экземпляр реляционной базы данных SQL Server 2008. Используя BIDS, откройте учебный проект Adventure Works из папки `Program Files\Microsoft SQL Server\100\Tools\Samples\AdventureWorks 2008 Analysis Services Project\Enterprise` и разверните его на своем экземпляре службы анализа. Если ваш реляционный сервер SQL Server 2008 и/или экземпляр Analysis Services являются именованными экземплярами, необходимо внести изменения в источник данных и целевой сервер Analysis Services, упомянутый в главе 2. Если вы откроете куб Adventure Works в BIDS, то на вкладке Cube Structure (Структура куба) увидите размерности и измерения, которые составляют этот куб (рис. 3.1).

Объект Measures (размерности), по существу, представляет собой специальное измерение, которое является набором размерностей. Размерности являются количественными сущностями, которые используются для анализа. Каждая размерность представляет собой часть категории, называющейся *размерной группой* (*measure group*). Размерные группы являются наборами связанных размерностей, и каждая размерность может быть частью только одной размерной группы. Чаще всего вы будете использовать по одной размерной группе для каждой таблицы фактов в хранилище данных. Размерные группы используются инструментами разработки или клиентскими инструментами главным образом для навигационных целей, чтобы улучшить читабельность или облегчить использование конечным пользователям. Они никогда не используются в MDX-запросах при обращении к размерностям. Тем не менее они могут использоваться в определенных MDX-функциях, которые, кстати, будут описаны в этой главе и главе 10. По умолчанию служба анализа генерирует размерную группу для каждой таблицы фактов, поэтому вы не должны беспокоиться об изменении структуры размерной группы. Конечно, вы можете ее изменить, если хотите.

На рис. 3.1 вы можете увидеть измерения, которые являются частями куба `Adventure Works`. Для пояснения фундаментальных концепций MDX с помощью иллюстраций мы будем использовать три измерения: `Product` (Товар), `Customer` (Клиент) и `Date` (Дата). Каждое измерение имеет одну или несколько иерархий, а каждая иерархия содержит один или несколько уровней. Подробно об измерениях, иерархиях и уровнях вы узнаете в главах 5 и 8. Для пояснения фундаментальных концепций MDX мы будем использовать иерархии `Calendar` (Календарь),

Product Line (Товарная линия) и Country (Страна) из измерений Date, Product и Customer соответственно. На рис. 3.2 показан раздел куба Adventure Works, использующий три иерархии: Calendar, Product Line и Country. Иерархия Calendar измерения Date содержит пять уровней: Calendar Year (Календарный год), Calendar Semester (Календарный семестр), Calendar Quarter (Календарный квартал), Month (Месяц) и Date (Дата). В иллюстрационных целях на рис. 3.2 не отображены все члены или уровни иерархий Calendar, Product Line и Country, и, следовательно, рис. 3.2 не отражает реальных данных, хранящихся в примере куба.

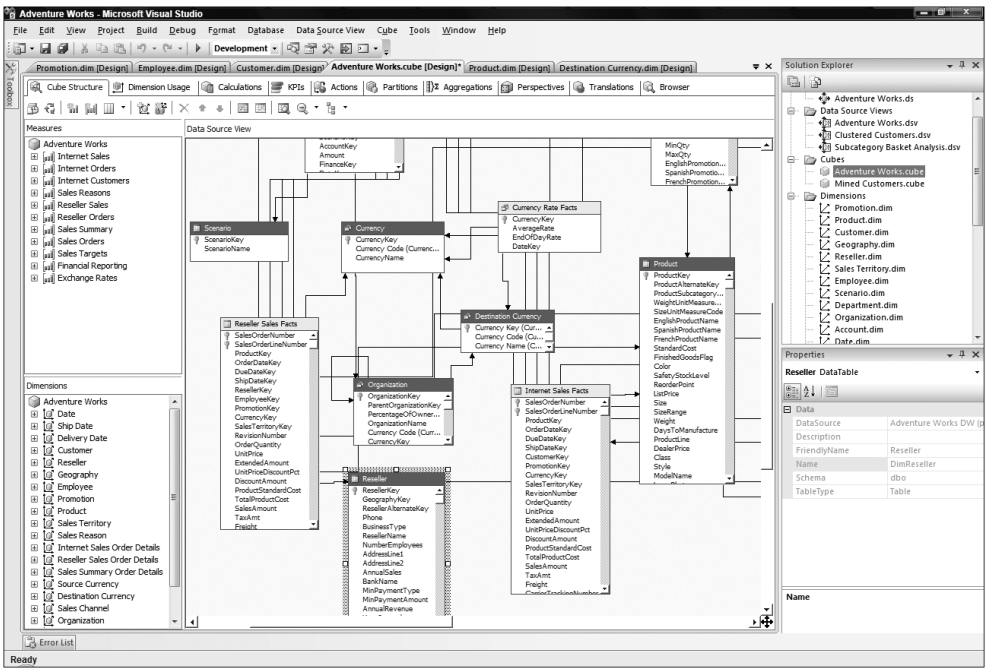

Рис. 3.1. Структура куба *Adventure Works*

Члены

Каждая иерархия измерения содержит один или несколько элементов, называемых *членами* (*members*). Каждый член соответствует одному или нескольким вхождениям этого значения в базовую таблицу измерений. На рис. 3.3 показаны члены иерархии Calendar (Календарь) в измерении Date (Дата). В иерархии Calendar элементы CY 2003, H1 CY 2003, H2 CY 2003, Q1 CY 2003, Q2 CY 2003, Q3 CY 2003 и Q4 CY 2003 являются членами. Обратите внимание на то, что элементы на каждом уровне вместе формируют набор членов иерархии. Вы также можете запрашивать информацию для членов определенного уровня. Например, элементы Q1 CY 2003, Q2 CY 2003, Q3 CY 2003 и Q4 CY 2003 являются членами уровня Calendar Quarter (Календарный квартал) для календарного года CY 2003.

В MDX каждый член иерархии представлен уникальным именем. Уникальные имена помогают идентифицировать определенные члены. Уникальное имя члена определяется свойствами измерения внутри куба, такими как MemberUniqueNameStyle и HierarchyUniqueNameStyle. Алгоритм определения уникального имени члена иерархии в данной книге не рассматривается. Вы можете получить доступ

к члену измерения, используя путь по имени (т.е. с использованием имени члена) или путь по ключу (т.е. с использованием ключа члена). Используя для создания кубов и измерений в BIDS задаваемые по умолчанию свойства, вы сможете получить доступ к члену измерения с помощью имени этого измерения, имени иерархии и имени уровня. Например, следующая строка представляет член `Q1 CY 2004` в иерархии `Calendar`.

```
[Date].[Calendar].[Calendar Quarter].[Q1 CY 2004]
```

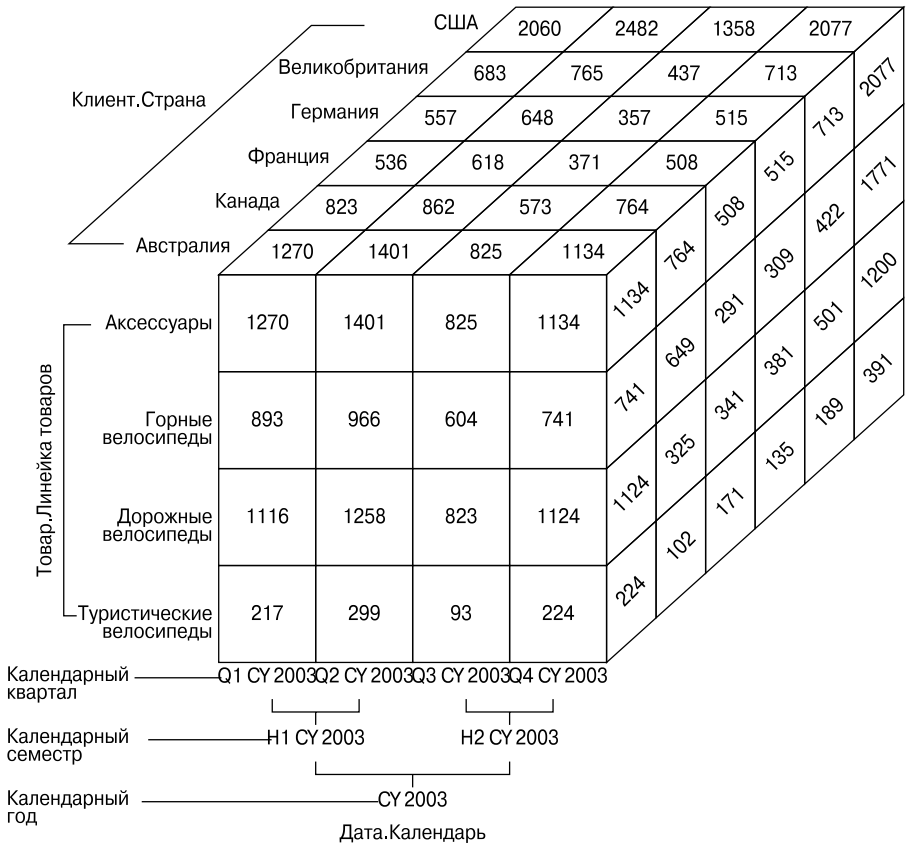

Рис. 3.2. Раздел куба *Adventure Works*, использующий три иерархии: *Calendar*, *Product Line* и *Country*

Имена измерений, иерархий, уровней и членов заключены в квадратные скобки (`[]`). В принципе вы не всегда обязаны заключать эти имена в квадратные скобки, но если в имени встречается пробел, цифра или имя представляет собой ключевое слово MDX, то квадратные скобки необходимо использовать. В предыдущем примере использовалось имя `Date`, которое является ключевым словом MDX и поэтому должно быть заключено в квадратные скобки.

Ниже приведены еще три корректных представления для члена `Q1 CY 2004`.

```
[Date].[Calendar].[Q1 CY 2004]                                    (1)
[Date].[Calendar].[CY 2004].[H1 CY 2004].[Q1 CY 2004]             (2)
[Date].[Calendar].[Calendar Quarter].&[2004]&[1]                   (3)
```

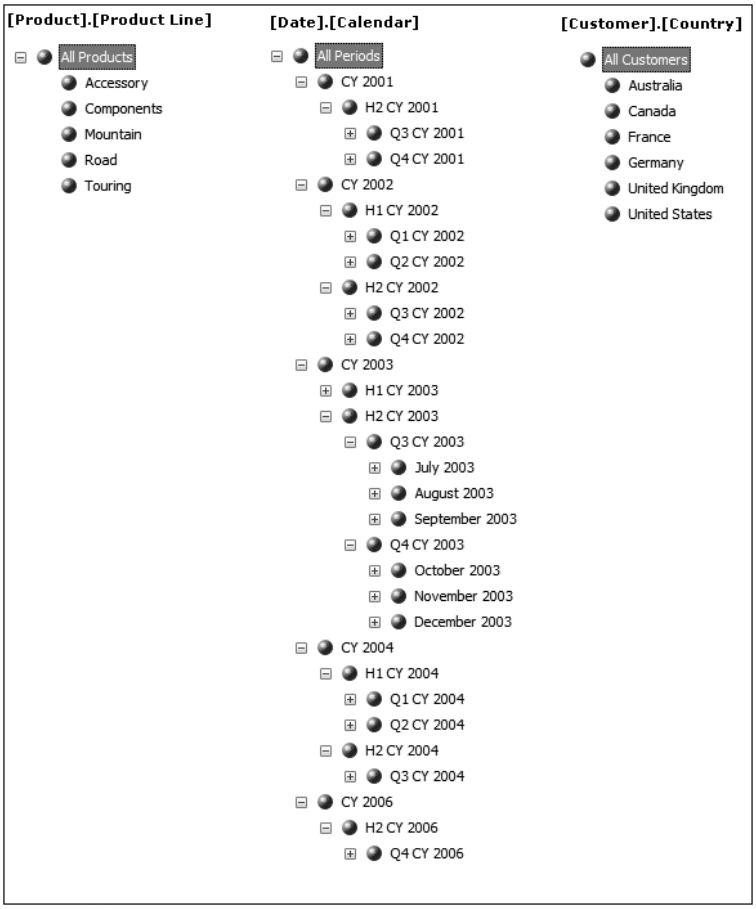

Рис. 3.3. Иерархии *Product Line*, *Calendar* и *Country*

В первом случае член иерархии представлен в формате *Измерение.Иерархия.ИмяЧлена*. Вы можете использовать данный формат в том случае, если не существует двух членов с совпадающими именами. Например, если первый квартал каждого года называется Q1, то вы не сможете использовать рассмотренный выше формат; вам необходимо будет уточнить MDX-выражение, используя в нем имя уровня. Если вы все же попытаетесь использовать данный формат обращения к члену иерархии, он всегда будет извлекать член Q1 для первого из представленных в иерархии годов. Второй вариант формата позволяет вам ясно видеть навигационный путь к члену иерархии, поскольку в нем представлены все элементы пути. Все рассмотренные форматы доступа к членам иерархии использовали в своем составе полное имя члена. Последний (третий) вариант представления члена иерархии использует путь по ключу, в котором вы можете видеть ключевые составляющие имени члена, представленные в форме `&[ИмяЧлена]`. При использовании пути по ключу всегда используется символ `&` перед ключевыми элементами имени члена.

Рассмотрим другой пример — член `Australia` (Австралия) иерархии `Country` (Страна) в измерении `Customer` (Клиент) может быть представлен следующим образом.

```
[Customer].[Country].Australia
```

Обратите внимание на тот факт, что в данном примере имя члена Australia не заключено в квадратные скобки. В данном случае квадратные скобки не обязательны, поскольку имя члена представляет собой одно слово и не содержит цифр. В общем случае вы можете использовать для доступа к члену иерархии следующий формат.

```
[Имя измерения].[Имя иерархии].[Имя уровня].[Имя члена]
```

В этой главе, а также всей книге преимущественно используется именно вышеуказанный формат. Если вы разрабатываете клиентские инструменты, мы рекомендуем приложить определенные усилия и изучить документацию, поставляемую с продуктом, чтобы понять алгоритм формирования уникального имени на основе установленных для измерений свойств. Таким образом, вы сможете использовать наилучший формат доступа к члену иерархии в MDX-запросах вместо прямого упоминания уникального имени в клиентском инструментарии.

Ячейки

На рис. 3.2 показаны три грани куба. Как вы можете видеть, передняя грань куба разделена на 16 маленьких квадратов, каждый из которых содержит число. Эти числа являются значениями размерности Internet Sales Amount (Сумма продаж через Интернет) куба данных AdventureWorksDW. Если вы посмотрите на остальные видимые грани куба данных, то поймете, что каждый из маленьких квадратов на фронтальной грани куба на самом деле является гранью маленького куба. Квадрат, расположенный в правом верхнем углу передней грани большого куба, содержит число 1134; обратите внимание на тот факт, что это число представлено и на других сторонах малого куба. Такой маленький куб принято называть *ячейкой* (*cell*).

Ячейка представляет собой элемент, из которого вы можете извлечь данные, соответствующие пересечению членов измерения. Число ячеек внутри куба данных определяется числом иерархий в каждом из измерений куба данных и числом членов каждой иерархии. Как вы можете себе представить, ячейки хранят значения всех размерностей в кубе. Если для какой-нибудь размерности недоступно значение данных в ячейке, считается, что соответствующим значением размерности является значение Null (т.е. пустое значение).

Если вы знакомы с понятием трехмерной системы координат, используемой для описания объемных геометрических тел, то осведомлены о назначении координатных осей X, Y и Z. Положение каждой точки в трехмерном геометрическом пространстве описывается значениями, отсчитываемыми по осям X, Y и Z. Подобно этому каждая ячейка внутри куба данных задается членами измерений. На иллюстрации, представленной на рис. 3.4, показаны три измерения: Product (Товар), Customer (Клиент) и Date (Дата). Предположим, что каждое из этих измерений имеет только по одной иерархии (как показано на иллюстрации), называющейся Product Line (Товарная линия), Country (Страна) и Calendar (Календарь). Как видно на рис. 3.4, иерархия Product Line включает 4 члена, иерархия Calendar также включает 4 члена (принимая во внимание только календарные кварталы) и, наконец, иерархия Country включает 6 членов. Таким образом, общее количество ячеек в этом кубе данных равно 96 (4*4*6=96).

После того как вы познакомились с понятием ячеек, следует разобраться в том, как извлекать из этих ячеек данные. Предположим, вы желаете извлечь данные из ячейки, закрашенной серым цветом в показанном на рис. 3.4 кубе данных. В этой ячейке представлено значение объема продаж, равное 966. Данная ячейка соответствует пересечению члена Product=Mountain, члена Date=Quarter2 и члена Customer=Australia.

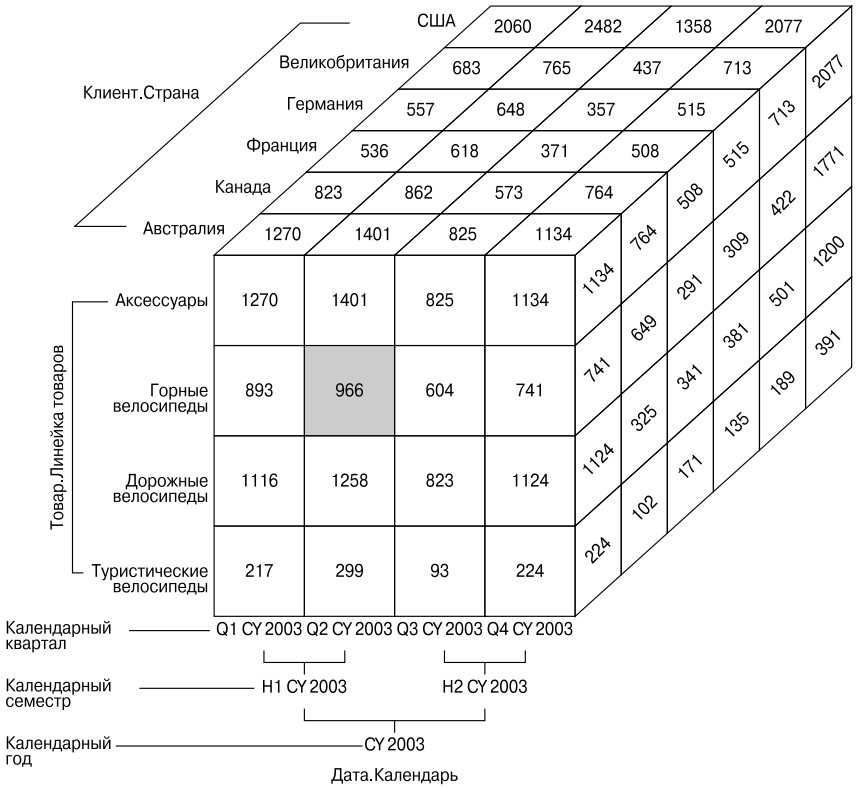

Рис. 3.4. Раздел куба, образованный измерениями `Product` (Товар), `Customer` (Клиент) и `Date` (Дата)

Чтобы извлечь данные из куба, вам необходимо отправить MDX-запрос к Analysis Services. Все, что вам необходимо выполнить, — это извлечь из куба данных величину "Internet Sales Amount" ("Объемы продаж через Интернет"), основываясь на условии, которое уникально идентифицирует ячейку, содержащую значение 966. Ниже приведен пример такого запроса.

```
SELECT Measures.[Internet Sales Amount] on COLUMNS
FROM [Adventure Works]
WHERE ( [Date].[Calendar].[Calendar Quarter].&[2003]&[2],
   [Product].[Product Line].[Mountain],
   [Customer].[Country].[Australia])
```

В этом запросе из куба данных `Adventure Works` на основе определенного условия, указанного в разделе `WHERE` MDX-запроса, выбирается значение `Measures.[Internet Sales Amount]`. Условие, заданное в предложении `WHERE`, уникально идентифицирует ячейку куба данных. В условии должны быть перечислены члены иерархии (о них рассказывалось в предыдущем разделе), разделенные запятыми. Подобное выражение на языке MDX, которое уникально идентифицирует ячейку, называют *кортежем*.

Кортежи

Как упоминалось в предыдущем разделе, *кортежем* (*tuple*) уникально идентифицирует ячейку или раздел куба данных. Кортеж представлен членами измерений куба данных (по одному члену из каждого измерения), разделенными запятыми. Кроме

того, кортеж заключается в круглые скобки. Кортеж совсем необязательно должен явно содержать члены всех измерений куба данных. Ниже приведено несколько примеров кортежей, основанных на кубе данных Adventure Works.

1. `([Customer].[Country].[Australia])`
2. `([Date].[Calendar].[2004].[H1 CY 2004].[Q1 CY 2004], [Customer].[Country].[Australia])`
3. `([Date].[Calendar].[2004].[H1 CY 2004].[Q1 CY 2004], [Product].[Product Line].[Mountain], [Customer].[Country].[Australia])`

В приведенных выше примерах кортежи 1 и 2 содержат члены не всех измерений рассматриваемого куба данных. Таким образом, они представляют разделы куба. Раздел куба, идентифицируемый кортежем, называют *слоем* (*slice*), или *срезом* данных. Впоследствии вы будете разделять куб данных на слои (или срезы), основываясь на определенных членах измерений.

В нашем примере, показанном на рис. 3.4, кортеж `([Customer].[Country].[Australia])` на самом деле ссылается на шестнадцать ячеек, соответствующих стране Австралия (ячейки, представляющие передний слой рассматриваемого куба данных). Таким образом, когда вы извлекаете данные из ячеек, на которые ссылается данный кортеж, на самом деле вы получаете данные о величине продаж через Интернет для всех клиентов из Австралии. Величина Интернет-продаж (`Internet Sales Amount`) для кортежа `([Customer].[Country].[Australia])` представляет собой результат агрегации значений ячеек из фронтальной грани куба данных, т.е. результат вычисления по всем значениям этих ячеек. Ниже приведен MDX-запрос, который позволяет извлечь данные, представляемые этим кортежем.

```
SELECT Measures.[Internet Sales Amount] on COLUMNS
FROM [Adventure Works]
WHERE ([Customer].[Country].[Australia])
```

Результатом данного запроса является значение $9061000,58.

Порядок, в котором представлены члены измерений в кортеже, не имеет значения. Это означает, что приведенные ниже примеры кортежей идентичны и уникально идентифицируют одну и ту же ячейку.

1. `([Date].[Calendar].[2005].[H1 CY 2004].[Q1 CY 2004], [Product].[Product Line].[Mountain], [Customer].[Country].[Australia])`
2. `([Product].[Product Line].[Mountain], [Customer].[Country].[Australia], ([Date].[Calendar].[2005].[H1 CY 2004].[Q1 CY 2004])`
3. `([Customer].[Country].[Australia], [Date].[Calendar].[2005].[H1 CY 2005].[Q1 CY 2005], [Product].[Product Line].[Mountain])`

Поскольку кортеж уникально идентифицирует ячейку, он может содержать не более одного члена от каждого измерения.

Кортеж, представленный единственным членом, называют *простым кортежем*. Простой кортеж можно не заключать в круглые скобки. Например, кортеж `([Customer].[Country].[Australia])` является простым кортежем и может быть представлен в виде `[Customer].[Country].[Australia]` или просто `Customer.Country.Australia`. Если кортеж представлен членами нескольких измерений, то такой кортеж требуется заключать в круглые скобки. Совокупности кортежей формируют новые объекты (называемые наборами), которые часто используются в запросах и выражениях MDX.

Наборы

Набор (*set*) — это совокупность кортежей, которые определены с использованием одинакового количества одних и тех же измерений. Набор обычно заключается в фигурные скобки ({ }). Далее приведены примеры, иллюстрирующие понятие набора.

- **Пример 1.** Кортежи `(Customer.Country.Australia)` и `(Customer.Country.Canada)` ссылаются на одно и то же измерение. Совокупность этих двух кортежей является корректным набором, для определения которого используется следующая запись.

```
{(Customer.Country.Australia), (Customer.Country.Canada)}
```

- **Пример 2.** Кортежи `(Customer.Country.Australia, [Product].[Product Line].[Mountain])` и `(Product.Country.Canada, [Date].[Calendar].[2004].[H1 CY 2004].[Q1 CY 2004])` нельзя объединить для формирования набора. Хотя оба они основаны на двух иерархиях, размерности, использующие кортеж, различаются. Оба кортежа имеют размерность иерархии `Customer.Country`, но вторые иерархии различаются.

- **Пример 3.** В каждом из приведенных ниже кортежей используется по три измерения: `Date`, `Product` и `Customer`.

```
1. ([Date].[Calendar].[2004].[H1 CY 2004].[Q1 CY 2004],
[Product].[Product Line].[Mountain],
[Customer].[Country].[Australia])
2. ([Product].[Product Line].[Mountain],
[Customer].[Country].[Australia],
([Date].[Calendar].[2002].[H1 CY 2002].[Q1 CY 2002])
3. ([Customer].[Country].[Australia], Date].[Calendar].[2003].[H1
CY 2003].[Q1 CY 2003], [Product].[Product Line].[Mountain])
```

  Члены измерений в приведенных выше кортежах отличаются, и таким образом эти кортежи ссылаются на различные ячейки куба данных. Однако согласно определению набора совокупность этих кортежей является правильным набором, который можно представить следующей записью.

```
{ ([Date].[Calendar Time].[2004].[H1 CY 2004].[Q1 CY 2004],
[Product].[Product Line].[Mountain],
[Customer].[Country].[Australia]),
([Product].[Product Line].[Mountain],
[Customer].[Country].[Australia],
([Date].[Calendar].[2002].[H1 CY 2002].[Q1 CY 2002]),
([Customer].[Country].[Australia], [Date].[Calendar].[2003].[H1
CY 2003].[Q1 CY 2003], [Product].[Product Line].[Mountain] )}
```

Набор может содержать ноль, один или несколько кортежей. Набор с нулевым количеством кортежей принято называть *пустым набором*. Пустому набору соответствует следующая запись.

```
{ }
```

Набор может содержать дублирующиеся кортежи. Ниже приведен пример такого набора.

```
{Customer.Country.Australia, Customer.Country.Canada,
Customer.Country.Austrailia}
```

Данный набор содержит два экземпляра кортежа `Customer.Country.Australia`. Поскольку кортежи формируются из членов измерений, то их можно исполь-

зовать в MDX-запросах. Если для указания ячейки используется кортеж, включающий только одно измерение, то нет необходимости заключать такой кортеж в круглые скобки. Аналогичным образом при использовании такого кортежа в запросе не требуется заключать его в фигурные скобки. При выполнении MDX-запроса этот кортеж неявно преобразуется в набор.

Теперь, когда вы ознакомились с ключевыми концепциями, которые помогут лучше понять язык MDX, пора перейти к изучению синтаксиса MDX-запросов, а также операторов, используемых в MDX-запросах и MDX-выражениях.

MDX-запрос

В главе 2 вы начали знакомство с инструкцией `SELECT` языка MDX. Для MDX-запроса используется следующий синтаксис.

```
[WITH <formula_expression> [, <formula_expression> ...]]
SELECT [<axis_expression>, [<axis_expression>...]]
FROM [<cube_expression>]
[WHERE [slicer_expression]]
```

Возможно, вас удивит, насколько формат инструкций `SELECT`, `FROM` и `WHERE` подобен языку структурированных запросов (Structured Query Language — SQL). Кроме того, эти инструкции служат для тех же целей, что и в SQL. Однако несмотря на идентичность вышеупомянутых инструкций в языках MDX и SQL, язык MDX позволяет вам выполнять более сложные операции. Вы более подробно узнаете о таких операциях в этой и последующих главах данной книги.

Ключевые слова `WITH`, `SELECT`, `FROM` и `WHERE` в сочетании с выражениями, следующими за ними, принято называть *предложениями* (*clause*). В представленном выше шаблоне MDX-запроса в квадратные скобки заключены необязательные элементы, которые могут присутствовать в одних запросах и отсутствовать в других.

Как вы можете видеть, предложения `WITH` и `WHERE` являются необязательными, поскольку они заключены в квадратные скобки (`[]`). Таким образом, вы можете прийти к выводу, что простейший запрос выглядит следующим образом.

```
SELECT
FROM [Adventure Works]
```

И вы абсолютно правы! Этот MDX-запрос возвращает одно-единственное значение. Вы интересуетесь, какое именно? Фактические данные содержатся в специальном измерении, которое называется `Measures`. Если приведенный выше запрос отправить к экземпляру Analysis Services, то в результате вы получите заданный по умолчанию член измерения `Measures`, который для куба `Adventure Works` является размерностью `Reseller Sales Amount` из размерной группы `Reseller Sales`. Результатом данного запроса является значение, полученное путем агрегации значений всех относящихся к этой размерности ячеек куба для заданных по умолчанию значений каждого измерения куба.

Предложение `WITH`, как правило, используется для пользовательских вычислений и операций. Более подробно о нем речь пойдет далее в этой главе. Но сначала мы рассмотрим предложения `SELECT`, `FROM` и `WHERE`.

Инструкция SELECT и определение оси

Инструкция `SELECT` языка MDX используется для извлечения подмножества многомерных данных из сервера OLAP. В языке SQL инструкция `SELECT` применяется для извлечения данных столбцов — т.е. извлекаемые данные представлены в двумерном ви-

де. SQL-запросы извлекают данные в виде строк и столбцов, что можно рассматривать как двумерные данные. Двумерная координатная система образуется осями X и Y. В данном контексте ось X используется для представления столбцов, ось Y — для представления строк. Однако многомерные данные описываются таким способом, который обеспечивает извлечение данных с помощью нескольких осей. И действительно, язык MDX предоставляет вам возможность извлекать данные по нескольким осям.

В языке MDX используется следующий синтаксис для инструкции SELECT.

```
SELECT [<axis_expression>, [<axis_expression>...]]
```

Выражение *axis_expression*, указанное после ключевого слова SELECT, ссылается на измерение, представляющее те данные, которые вы желаете извлечь. Эти измерения называют *осевыми измерениями* (axis dimensions), поскольку представленные ими данные проецируются на соответствующие оси. Для выражения *axis_expression* используется следующий синтаксис.

```
<axis_expression> := <набор> ON (ось) | Axis (номер оси) | (номер оси)
```

Осевые измерения используются для извлечения многомерных наборов данных. Набор (упорядоченную совокупность кортежей) определяют для того, чтобы сформировать осевое измерение. Язык MDX предоставляет возможность указать до 128 осей в инструкции SELECT. Первые пять осей имеют псевдонимы. Это оси COLUMNS (столбцы), ROWS (строки), PAGES (страницы), SECTIONS (разделы) и CHAPTERS (главы). Последующие оси указываются с помощью слова Axis, за которым следует номер оси. Рассмотрим следующий пример.

```
SELECT   Measures.[Internet Sales Amount] ON COLUMNS,
        [Customers].[Country].MEMBERS on ROWS,
        [Product].[Product Line].MEMBERS on PAGES
FROM [Adventure Works]
```

В приведенной выше инструкции SELECT указаны три оси. В этом примере данные из измерений Measures, Customers и Product отображаются на трех осях, чтобы сформировать осевые измерения. Этот же оператор можно представить следующим образом.

```
SELECT   Measures.[Internet Sales Amount] ON 0,
         [Customers].[Country].MEMBERS on 1,
         [Product].[Product Line].MEMBERS on 2
FROM [Adventure Works]
```

Осевое измерение

При определении инструкции SELECT вы создаете осевое измерение. Инструкция SELECT назначает набор осям COLUMNS и ROWS (а также дополнительным осям, если в вашем запросе используется более двух осей). В отличие от измерения среза данных (slicer dimension), которое описывается далее в этой главе, осевое измерение извлекает и содержит данные для нескольких членов, а не для единственного члена. Следует заметить, что когда мы ссылаемся на осевое измерение, в действительности это относится к иерархии Analysis Services 2008, так как в оператор MDX будут включены иерархии.

> В MDX нельзя создать работоспособный запрос, в котором опущены "низшие" оси (т.е. оси с меньшими порядковыми номерами). Если необходимо указать в запросе ось PAGES, то вы также должны указать оси COLUMNS и ROWS.

Предложение FROM и определение куба

Предложение FROM в MDX-запросе определяет куб, из которого вы извлекаете данные для анализа. Оно напоминает предложение FROM языка SQL, в котором указывается имя таблицы. Предложение FROM обязательно для любого MDX-запроса. Для предложения FROM используется следующий синтаксис.

```
FROM <cube_expression>
```

Выражение *cube_expression* обозначает имя всего куба или его подраздела, из которого вы желаете извлечь данные. Язык SQL допускает использование в предложении FROM нескольких таблиц, однако в запросе MDX можно указать имя только одного куба данных. Куб, указанный в предложении FROM, называют *контекстом куба* (*cube context*), и MDX-запрос выполняется внутри этого контекста. То есть, каждая часть выражения *axis_expression* будет извлекаться из контекста куба, указанного в предложении FROM.

```
SELECT [Measures].[Internet Sales Amount] ON COLUMNS
FROM [Adventure Works]
```

Выше приведен пример правильного MDX-запроса, который извлекает размерность [Internet Sales Amount] по оси X. Указанная размерность извлекается из контекста куба [Adventure Works]. Несмотря на то, что предложение FROM ограничивает вас возможностью извлечения данных только из одного куба или раздела куба, вы все же можете получать данные из других кубов, используя функцию LookupCube языка MDX. Если имеется два (или более) куба с общими членами измерений, то функция LookupCube поможет вам с помощью этих членов извлечь размерности, не входящие в текущий контекст куба.

Предложение WHERE и определение среза данных

При работе практически с любой реляционной базой данных вам необходимо выполнять запросы, которые возвращают только определенную часть из всех данных, доступных в указанной таблице или наборе объединенных таблиц (и/или объединенных баз данных). Это достигается благодаря использованию инструкций SQL, которые позволяют вам планировать, какие данные следует возвращать в результатах выполняемого запроса, а какие не следует. Ниже приведен пример не ограниченного условием WHERE SQL-запроса к таблице Product, которая содержит информацию о продажах определенных товаров.

```
SELECT *
FROM Product
```

Результатом выполнения этого запроса станет таблица из пяти столбцов и нескольких строк. (То есть запрос вернет всю таблицу Products. — *Примеч. ред.*)

Product ID	Product Line	Color	Weight	Sales
1	Accessories	Silver	5,00	200,00
2	Mountain	Grey	40,35	1000,00
3	Road	Silver	50,23	2500,00
4	Touring	Red	45,11	2000,00

Значок * после ключевого слова SELECT означает, что запрос вернет все столбцы и все таблицы. Если вам требуется извлечь только информацию о цвете (столбец

Color) для каждой линейки товаров (столбец Product Line), необходимо ограничить запрос так, чтобы он возвращал только необходимую информацию. Следующий запрос сконструирован так, что он возвращает только два столбца таблицы Product.

```
SELECT ProductLine, Color
FROM Product
```

Данный запрос возвращает следующий набор результатов.

Product Line	Color
Accessories	Silver
Mountain	Grey
Road	Black
Touring	Red

Концепция создания SQL-запросов, возвращающих только требуемые данные, напрямую экстраполируется на MDX-запросы. Фактически в MDX-запросах используются аналогичные инструкции, задающие условие отбора, что обеспечивает возвращение запросами только требуемых данных. Для задания условия отбора применяется предложение WHERE. Рассмотрев вкратце использование инструкции WHERE в SQL-запросах, обратимся к изучению того, как данная концепция применяется в конструкциях языка MDX. Ниже приведен пример SQL-запроса, использующего предложение WHERE для возвращения записей двух столбцов только для тех товаров из таблицы Product, которые имеют серебристый (Silver) цвет.

```
SELECT ProductLine, Sales
FROM Product
WHERE Color = 'Silver'
```

Данный запрос возвращает следующий набор результатов.

Product Line	Sales
Accessories	200,00
Road	2500,00

Подобные концепции применяются и в MDX. Действительно, в языке MDX используются обе инструкции, SELECT и WHERE. Инструкция SELECT определяет измерения и члены, возвращаемые запросом, а инструкция WHERE ограничивает набор результатов запроса с помощью некоторого критерия. В предыдущем примере ограничивающим условием являлось выражение Color='Silver'. Заметьте, что члены представляют собой элементы, которые составляют иерархию измерения. Таблица Product (Товар), представленная как куб данных, будет содержать две размерности — Sales (Продажи) и Weight (Вес), а также измерение Product с иерархиями Product ID (Код товара), Product Line (Линейка товаров) и Color (Цвет). В этом примере таблица Product используется и в качестве таблицы фактов, и в качестве таблицы измерений. Ниже приведен MDX-запрос к кубу данных, который возвращает те же результаты, что и рассмотренный выше запрос SQL.

```
SELECT Measures.[Sales] ON COLUMS,
[Product].[Product Line].MEMBERS on ROWS
FROM ProductsCube
WHERE ([Product].[Color].[Silver])
```

Два столбца, выбираемых запросом SQL, теперь представлены по осям COLUMNS и ROWS. Условие в предложении WHERE SQL-запроса (которое представляет собой выражение, сравнивающее строковые значения) теперь преобразовано в предложение WHERE MDX-запроса, являющееся срезом куба, содержащим информацию о товарах серебристого цвета. Несмотря на то, что запросы SQL и MDX выглядят похоже, операции, выполняемые в SQL Server и Analysis Services, заметно отличаются.

Измерение среза

Измерение среза (*slicer dimension*) создается при определении предложения WHERE; по сути, это фильтр, который исключает нежелательные измерения и члены. Как упоминалось в этой главе ранее, в контексте Analysis Services 2008, измерения будут в действительности иерархиями в Analysis Services 2008. Еще более интересен тот факт, что измерение среза включает и все те оси куба, которые не включены явно в оси, указанные в определении запроса. Заданные по умолчанию члены иерархий, которые не включены в оси запроса, используются в осях среза данных. Независимо от способа получения данных, измерение среза работает только с MDX-выражениями, результатом которых является один кортеж. (MDX-выражениями будут подробно рассмотрены далее в этой главе.) Если в оси среза определено несколько кортежей, то они будут обработаны как набор, а их значения — агрегированы с использованием размерности, заданной в запросе, и функции агрегации из этой размерности.

Предложение WITH и вычисляемые члены

Зачастую бизнес-процессы требуют использования вычислений, которые должны быть сформулированы в рамках специфического запроса. Предложение WITH в MDX-запросе предоставляет вам возможность выполнять такие вычисления непосредственно в контексте запроса. Кроме того, с помощью функции LookupCube языка MDX вы также можете получить результаты, включающие данные, не входящие в контекст текущего указанного куба.

Типичными вычислениями, которые создаются с использованием предложения WITH, являются именованные наборы и вычисляемые члены. Кроме этого, предложение WITH обеспечивает возможность определять вычисления по ячейкам, загружать куб в кэш Analysis Services для улучшения выполнения запроса, изменять содержимое ячеек с помощью вызова функций из внешних библиотек, а также позволяет реализовать некоторые сложные концепции типа порядка вычисления и очередности прохода. Вы узнаете об именованных наборах, вычисляемых членах и вычисляемых размерностях в этой главе.

Для предложения WITH используется следующий синтаксис.

```
[WITH <formula_expression> [, <formula_expression> ...]]
```

Предложение WITH позволяет определять несколько вычислений внутри одной инструкции. Выражение *formula_expression* варьируется в зависимости от типа вычислений. При использовании в предложении WITH нескольких вычислений они отделяются друг от друга запятыми.

Именованные наборы

Как уже упоминалось в этой главе, набор — это совокупность кортежей. Выражение, описывающее кортеж (даже вполне простой), зачастую оказывается достаточно длинным, а это приводит к тому, что запрос выглядит сложным и нечитабельным. MDX позволяет динамически определять наборы с помощью специфического имени, а затем это имя может использоваться внутри запроса. Представьте себе это имя как

псевдоним, назначенный для совокупности кортежей, представляющей собой набор, который вы хотите использовать для извлечения данных. Набор, для которого назначен псевдоним, называют *именованным набором* (*named set*). Таким образом, именованный набор — это просто псевдоним для обычного MDX-выражения, описывающего набор. Такой псевдоним можно использовать в любом месте внутри запроса вместо того, чтобы вводить реальное выражение, описывающее набор.

Рассмотрим случай, когда ваш куб содержит данные о продажах для клиентов из разных стран. Предположим, что вы хотите извлечь информацию для клиентов из Европы. В таком случае ваш MDX-запрос может выглядеть следующим образом.

```
SELECT Measures.[Internet Sales Amount] on COLUMNS,
{[Customer].[Country].[Country].&[France],
[Customer].[Country].[Country].&[Germany],
[Customer].[Country].[Country].&[United Kingdom]} ON ROWS
FROM [Adventure Works]
```

Этот запрос не слишком длинный, но вы можете представить себе запрос, который содержит большое количество членов и функций, применяемых к этому специфическому набору несколько раз. Вместо того чтобы каждый раз использовать полное описание набора, вы можете создать именованный набор и затем использовать в запросе псевдоним, как показано в следующем примере.

```
WITH SET [EUROPE] AS '{[Customer].[Country].[Country].&[France],
[Customer].[Country].[Country].&[Germany],[Customer].[Country].
[Country].&[United Kingdom]} '

SELECT Measures.[Internet Sales Amount] on COLUMNS,
[EUROPE] ON ROWS
FROM [Adventure Works]
```

Для именованного набора выражение *formula_expression* в предложении WITH выглядит следующим образом.

```
Formula_expression := [DYNAMIC] SET <псевдоним_набора> AS ['']<набор>['']
```

В качестве псевдонима можно использовать любое имя, которое обычно заключают в квадратные скобки. Обратите внимание, что в выражении, определяющем именованный набор, используются ключевые слова SET и AS. Ключевое слово DYNAMIC является необязательным. На самом деле набор кортежей не обязательно заключать в одинарные кавычки (''). Одинарные кавычки используются лишь для обеспечения обратной совместимости, поскольку они требуются для версии Analysis Services 2000.

Вычисляемые члены

Вычисляемые члены представляют собой вычисления, определяемые MDX-выражениями. Таким образом вычисляемые члены позволяют получить результат, основанный на вычислении MDX-выражений, а не просто извлечь исходные фактические данные. Типичным примером вычисляемого члена может служить вычисление суммы продаж товаров за период с начала года по настоящий момент. Предположим, что фактические данные содержат информацию только о продажах товаров за каждый месяц и вам необходимо вычислить сумму продаж с начала года по настоящий момент. С помощью MDX-выражения, использующего предложение WITH, вы сможете извлекать информацию о продажах не только за конкретный месяц, но и за период с начала года по данный месяц.

Выражение *formula_expression* для вычисляемого члена в предложении WITH выглядит следующим образом.

```
Formula_expression := MEMBER <ИмяЧлена> AS ['']<Выражение_MDX>[''],
            [ , SOLVE_ORDER = <целое число>]
            [ , <СвойствоЯчейки> = <ВыражениеСвойства>]
```

В языке MDX для создания вычисляемых членов в предложении `WITH` используются ключевые слова `MEMBER` и `AS`. *ИмяЧлена* — это полностью определенное имя члена, которое включает измерение, иерархию и уровень, для которого необходимо создать вычисляемый член. Выражение *Выражение_MDX* должно возвращать значение, которое соотносится с членом. Необязательному параметру `SOLVE_ORDER` необходимо задать положительное целое значение (если данный параметр используется). Параметр `SOLVE_ORDER` определяет порядок, в котором должны вычисляться члены, если задано несколько вычисляемых членов. Параметр *СвойствоЯчейки* также является необязательным; он используется для указания свойств ячейки для вычисляемого члена (таких, как форматирование текста содержимого ячейки, включая цвет фона).

Все размерности куба хранятся в специальном измерении `Measures`. Для данного измерения также можно создавать вычисляемые члены. Фактически большинство вычисляемых членов, применяемых в реальных бизнес-процессах, обычно создается именно для измерения `Measures`. Вычисляемые члены, созданные для измерения `Measures`, принято называть вычисляемыми размерностями. Далее приведено несколько примеров, использования вычисляемых членов.

- **Пример 1.**

```
WITH MEMBER [MEASURES].[Profit] AS '([Measures].[Internet Sales
Amount] - [Measures].[Internet Standard Product Cost])'
SELECT measures.profit ON COLUMNS,
   [Customer].[Country].MEMBERS ON ROWS
FROM [Adventure Works]
```

В примере 1 вычисляемый член `Profit` (Прибыль) определен как разность размерностей `[Internet Sales Amount]` (Сумма Интернет-продаж) и `[Internet Standard Product Cost]` (Стандартные затраты на товар). При выполнении данного запроса для каждой страны вычисляемый член будет получен на основе вычисления MDX-выражения.

- **Пример 2.**

```
WITH
SET [ProductOrder] AS 'Order([Product].[Product Line].MEMBERS,
 [Internet Sales Amount], BDESC)'
MEMBER [Measures].[ProductRank] AS 'Rank([Product].[Product
 Line].CURRENTMEMBER, [ProductOrder])'
SELECT {[ProductRank],[Sales Amount]} ON COLUMNS,
[ProductOrder] on ROWS
FROM [Adventure Works]
```

В примере 2 внутри запроса создаются именованный набор и вычисляемый член. Запрос сортирует товары на основе значения суммы Интернет-продаж `(Internet Sales Amount)` и возвращает сумму продаж для каждого товара вместе с его рангом. Именованный набор `[ProductOrder]` создается таким образом, что члены в этом наборе отсортированы по убыванию в соответствии с величиной суммы продаж. Эта задача выполняется с помощью MDX-функции, которая называется `Order` (вы узнаете более подробно о функции `Order` в приложении А). Чтобы извлечь ранг каждого товара, создается вычисляемый член `[ProductRank]`, для которого используется MDX-функция под названием `Rank`.

Далее приведены результаты рассмотренного выше запроса в случае обращения его к учебной базе данных Adventure Works из примера базы данных в Adventure Works DW 2008.

	Product Rank	Sales Amount
All Products	1	$109,809,274.20
Road	2	$48,262,055.15
Mountain	3	$42,456,731.56
Touring	4	$16,010,837.10
Accessory	5	$2,539,401.59
Components	6	$540,248.80

- **Пример 3.**

```
WITH MEMBER Measures.[Cumulative Sales] AS 'SUM(YTD(),[Internet
  Sales Amount])'
SELECT {Measures.[Internet Sales Amount], Measures.[Cumulative
  Sales]} ON 0,
[Date].[Calendar].[Calendar Semester].MEMBERS on 1
FROM [Adventure Works]
```

В примере 3 вычисляемый член создается так, что вы сможете проанализировать сумму продаж за каждый месяц наряду с накопительной суммой продаж. Для решения этой задачи используются две MDX-функции: SUM и YTD. MDX-функция — YTD — вызывается без указания каких-либо параметров, поэтому в вычислениях используется заданный по умолчанию на данном уровне член временного измерения. Функция SUM используется для объединения сумм продаж для указанного специфического уровня. Результаты выполнения рассмотренного выше запроса при его обращении к учебной базе данных Analysis Services под названием Adventure Works показаны в приведенной ниже таблице. Как можно увидеть в данной таблице, накопительные суммы продаж (Cumulative Sales), соответствующие членам H2 CY 2002, H2 CY 2003 и H2 CY 2004, представляют собой сумму величины Интернет-продаж для данного члена иерархии времени и величины Интернет-продаж за предыдущее полугодие соответствующего года.

	Internet Sales Amount	Cumulative Sales
H2 CY 2001	$3,266,373.66	$3,266,373.66
H1 CY 2002	$3805710,59	$3805710,59
H2 CY 2002	$2724632,94	$6530343,53
H1 CY 2003	$3037501,36	$3037501,36
H2 CY 2003	$6753558,94	$9791060,30
H1 CY 2004	$9720059,11	$9720059,11
H2 CY 2004	$50840,63	$9770899,74
H2 CY 2006	null	null

- **Пример 4.**

```
WITH MEMBER [Date].[Calendar].[%Change] AS
100* (([Date].[Calendar].[Calendar Quarter].[Q2 CY 2002] -
   [Date].[Calendar].[Calendar Quarter].[Q1 CY 2002])    /
   [Date].[Calendar].[Calendar Quarter].[Q2 CY 2002])
```

```
SELECT {[Date].[Calendar].[Calendar Quarter].[Q1 CY 2002],
[Date].[Calendar].[Calendar Quarter].[Q2 CY 2002],
[Date].[Calendar].[%Change]} ON COLUMNS,
Measures.[Internet Sales Amount] ON ROWS
FROM [Adventure Works]
```

Приведенный выше запрос демонстрирует пример использования вычисляемого члена, который определен в измерении `Date` (Дата) для возвращения результатов сравнения поквартальных сумм продаж. В данном примере используются первый и второй кварталы 2002 года. Рассматриваемый запрос вернет следующие результаты.

	Q1 CY 2002	Q2 CY 2002	%Change
Internet Sales Amount	$1791698,45	$2014012,13	11,038

MDX-выражения

MDX-выражения представляют собой инструкции языка MDX, которые вычисляют определенные значения. Обычно они используются для вычисления или определения значений для таких объектов, как заданный по умолчанию член и заданная по умолчанию размерность, либо применяются при определении выражений безопасности, позволяющих или запрещающих доступ к некоторой информации. Обычно MDX-выражения используют в качестве параметра член, кортеж или набор и возвращают некоторое значение. Если в результате выполнения выражения не получено никакое значение, то возвращается значение `Null`. Далее приведены несколько примеров MDX-выражений.

- **Пример 1.**

```
Customer.[Customer Geography].DEFAULTMEMBER
```

Данный пример возвращает член, заданный по умолчанию для иерархии `Customer Geography` (Месторасположение клиента) измерения `Customer` (Клиент).

- **Пример 2.**

```
(Customer.[Customer Geography].CURRENTMEMBER, Measures.[Sales
 Amount]) - (Customer.[Customer Geography].Australia,
 Measures.[Sales Amount])
```

Это MDX-выражение используется для сравнения сумм продаж, осуществленных для клиентов из различных стран, с суммой продаж для клиентов из Австралии.

Подобное выражение обычно используется в вычислении, называемом вычисляемой размерностью. Сложные MDX-выражения могут включать различные операторы языка MDX наряду с комбинациями функций, доступных в этом языке. Ниже приведен пример такого сложного MDX-выражения.

- **Пример 3.**

```
COUNT(INTERSECT( DESCENDANTS( IIF( HIERARCHIZE(EXISTS[Employee].
          [Employee].MEMBERS,
STRTOMEMBER("[Employee].[login].[login].&["+USERNAME+"]")),
          POST).ITEM(0).ITEM(0).PARENT.DATAMEMBER is
          HIERARCHIZE(EXISTS([Employee].[Employee].MEMBERS,
          STRTOMEMBER("[Employee].[login].[login].&["+USERNAME+"]")),
          POST).ITEM(0).ITEM(0),
HIERARCHIZE(EXISTS([Employee].[Employee].MEMBERS,
```

```
          STRTOMEMBER("[Employee].[login].[login].&["+username+"]")
),           POST).ITEM(0).ITEM(0).PARENT,
HIERARCHIZE(EXISTS([Employee].[Employee].MEMBERS,
          STRTOMEMBER("[Employee].[login].[login].&["+USERNAME+"]")
),           POST).ITEM(0).ITEM(0))).ITEM(0),
Employee.Employee.CURRENTMEMBER))  > 0
```

Приведенный выше пример является MDX-выражением, обеспечивающим защиту ячейки. Оно позволяет просматривать информацию о продажах только тем сотрудникам, которые вносили эту информацию (или сотрудникам, отчитывающихся перед ними). Данное MDX-выражение использует несколько функций (о некоторых из них вы узнаете более подробно позже в этой главе). Как видите, это довольно непростое MDX-выражение. Оно возвращает значение `TRUE` (Истина) или `FALSE` (Ложь), в зависимости от того, какой сотрудник пытается получить доступ к базе данных, и служба Analysis Services, основываясь на возвращаемых этим выражением результатах, обеспечивает доступ к соответствующим ячейкам для данного сотрудника. Более подробно этот пример будет рассмотрен в главе 22.

Язык MDX значительно усовершенствовался с момента своего появления, и вы довольно быстро можете столкнуться со сложными запросами или MDX-выражениями, подобными тому, которое продемонстрировано в примере 3. Над реализацией определенных решений могут работать сразу несколько специалистов, и поэтому было бы полезно иметь некоторую документацию, поясняющую назначение запросов или выражений. Подобно языкам программирования, которые позволяют внедрять в программный код комментарии, язык MDX также поддерживает возможность размещения комментариев в MDX-запросах и MDX-выражениях. На текущий момент существует три различных способа добавления комментариев в код MDX. Эти способы продемонстрированы ниже.

```
// (две косые черты) здесь следует комментарий
-- (два дефиса) здесь следует комментарий
/* здесь следует комментарий */ (две пары символов косой черты
                                  и звездочки)
```

Мы настоятельно рекомендуем вам добавлять комментарии в свои MDX-выражения и MDX-запросы так, чтобы вы смогли через определенное время вспомнить или понять, для чего же собственно предназначен тот или иной MDX-запрос (либо MDX-выражение).

Операторы

Язык MDX, подобно другим языкам запросов (например, SQL) или языкам программирования, включает несколько операторов. Оператор представляет собой функцию, которая выполняет специфическое действие и использует аргументы. В MDX используются операторы нескольких типов. Подобно другим языкам, MDX содержит арифметические операторы, логические операторы и специальные операторы MDX.

Арифметические операторы

Обычные арифметические операторы, такие как `+`, `–`, `*` и `/`, входят в состав арифметических операторов MDX. Как и в языках программирования, эти операторы могут применяться для выполнения арифметических операций с двумя числами. Операторы `+` и `–` также могут использоваться в качестве унарных операторов

для чисел. Понятие *унарный оператор* означает, что данный оператор может использоваться с единственным операндом (единственным числом) в MDX-выражениях, например `+100` или `–100`.

Операторы наборов

Операторы +, – и * помимо того, что являются арифметическими операторами, также могут использоваться для выполнения операций над наборами кортежей MDX. Оператор + применяется для объединения двух наборов, оператор – используется для вычисления разности двух наборов, а оператор * позволяет найти векторное произведение двух наборов. Результатом векторного произведения двух наборов являются все возможные комбинации кортежей в каждом наборе. Векторное произведение позволяет извлечь данные в матричном формате. Например, если имеются два набора {Мужчина, Женщина} и {2003, 2004, 2005}, то результатом векторного произведения этих двух наборов, представленного как `{Мужчина, Женщина}*{2003, 2004, 2005}`, будет набор {(Мужчина, 2003), (Мужчина, 2004), (Мужчина, 2005), (Женщина, 2003), (Женщина, 2004), (Женщина, 2005)}. Далее приведены примеры, демонстрирующие использование операторов набора для выполнения некоторых операций с наборами кортежей.

- **Пример 1.** Результатом следующего MDX-выражения

```
{[Customer].[Country].[Australia]} +
{[Customer].[Country].[Canada]}
```

является объединение двух наборов, приведенное ниже.

```
{[Customer].[Country].[Australia], [Customer].[Country].[Canada]}
```

- **Пример 2.** Результатом следующего MDX-выражения

```
{[Customer].[Country].[Australia],[Customer].[Country].[Canada]}*
{[Product].[Product Line].[Mountain],[Product].
           [Product Line].[Road]}
```

является векторное произведение двух наборов, приведенное ниже.

```
{([Customer].[Country].[Australia],[Product].[Product
 Line].[Mountain])
([Customer].[Country].[Australia],[Product].[Product
 Line].[Road])
([Customer].[Country].[Canada],[Product].[Product
 Line].[Mountain])
([Customer].[Country].[Canada],[Product].[Product
 Line].[Road])}
```

Операторы сравнения

Язык MDX поддерживает работу с такими операторами сравнения, как <, <=, >, >=, = и <>. Эти операторы используют два MDX-выражения в качестве аргументов и возвращают значения `TRUE` (Истина) или `FALSE` (Ложь) в зависимости от результатов сравнения величин, полученных в результате вычисления MDX-выражений.

Пример. Следующее MDX-выражение использует оператор сравнения > (больше чем).

```
Count (Customer.[Country].members) > 3
```

В приведенном выше примере функция Count используется для подсчета количества членов в иерархии Country измерения Customer. Поскольку иерархия Country содержит больше трех членов, результатом рассматриваемого MDX-выражения будет значение TRUE.

Логические операторы

Язык MDX поддерживает такие логические операторы, как AND, OR, XOR, NOT и IS, которые используются для логической конъюнкции (операция И), логической дизъюнкции (операция ИЛИ), логического отрицания и сравнения соответственно. Эти операторы используют два выражения MDX в качестве аргументов и возвращают значение TRUE или FALSE как результат логической операции. Данные логические операторы обычно используются в MDX-выражениях, предназначенных для обеспечения защиты ячейки или измерения от несанкционированного доступа (подробнее об этом вы узнаете в главе 22).

Специальные операторы MDX — фигурные скобки, запятые и двоеточия

В фигурные скобки ({}) заключается кортеж или несколько кортежей, чтобы сформировать набор MDX. В том случае, когда используется один кортеж, фигурные скобки не обязательны, поскольку Analysis Services автоматически преобразует единственный кортеж в набор. Если же необходимо представить в виде набора несколько кортежей или вы имеете дело с пустым набором, то в таком случае требуется использовать фигурные скобки.

Вам уже доводилось видеть, как используется символ запятой (,) в нескольких предыдущих примерах. Символ запятой применяется при формировании кортежа, который содержит более одного члена. Кроме того, символ запятой используется при построении наборов, включающих несколько кортежей (кортежи в наборе отделяются друг от друга запятыми). В наборе {(Мужчина, 2003), (Мужчина, 2004), (Мужчина, 2005), (Женщина, 2003), (Женщина, 2004), (Женщина, 2005)} символ запятой используется не только для формирования самих кортежей, но также и для создания набора кортежей.

Символ двоеточия (:) используется для определения диапазона членов внутри набора. Члены внутри набора упорядочиваются на основе ключа или имени члена, в зависимости от того, что было задано при создании набора. Символ двоеточия используется между двумя непоследовательными членами в наборе, чтобы включить все остальные расположенные между ними члены. Предположим, что имеется приведенный ниже набор.

```
{[Customer].[Country].[Australia], [Customer].[Country].[Canada],
 [Customer].[Country].[France], [Customer].[Country].[Germany],
 [Customer].[Country].[United Kingdom],
 [Customer].[Country].[United States]}
```

В таком случае результатом приведенного ниже MDX-выражения

```
{[Customer].[Country].[Canada] : [Customer].[Country].[United Kingdom]}
```

будет следующий набор.

```
{[Customer].[Country].[Canada], [Customer].[Country].[France],
[Customer].[Country].[Germany], [Customer].[Country].[United Kingdom]}
```

MDX-функции

MDX-функции могут использоваться в MDX-выражениях или MDX-запросах. Язык MDX составляет основу Analysis Services 2008; BIDS создает MDX-выражения, которые обычно включают MDX-функции, для извлечения информации из базы данных Analysis Services, основываясь на ваших действиях, таких как просмотр измерений или кубов. MDX-функции помогают обращаться к некоторым общим операциям, которые требуются в ваших MDX-запросах или MDX-выражениях (включая упорядочение кортежей в наборе, подсчет количества членов в измерении и манипуляции со строковыми значениями, необходимые при преобразовании введенной пользователем информации в соответствующие объекты MDX).

В данном разделе вы узнаете о различных категориях MDX-функций и увидите некоторые простые примеры использования функций. Лучший способ изучить MDX-функции — рассмотреть их использование в реальных бизнес-сценариях, так чтобы вы смогли применить соответствующие MDX-функции в аналогичных случаях. В этой книге будут приводиться примеры кода MDX, генерируемого реальными продуктами. Для того чтобы вы не просто ознакомились с основами Analysis Services, а достигли достаточной степени мастерства в организации систем анализа данных, чрезвычайно важно уделять особое внимание изучению таких примеров кода MDX и самостоятельному экспериментированию с ними. И хотя это очень сложная задача, но цель все же достижима. Вы сможете стать профессионалом, если очень захотите. Кроме того, когда вы выполняете срез измерения в любой программе просмотра кубов данных (подобной Office Web Components), на самом деле используется код MDX, который генерируется и выполняется приложением для заполнения значений в созданном срезе данных. При создании отчета на основе куба данных (UDM — унифицированной модели измерений) с помощью Excel (этот вопрос рассматривается в главе 17) или с помощью Reporting Services (глава 20) также используется код MDX, создаваемый для извлечения данных, по которым должен быть составлен отчет. Почти все MDX-запросы или MDX-выражения, генерируемые BIDS или клиентскими инструментами, используют различные MDX-функции; о некоторых из них вы подробно узнаете по мере прочтения данной книги.

В главе 11 вы узнаете о новых хранимых процедурах, поддерживаемых в Analysis Services 2008, а также о том, как написать на языках программирования .NET свои собственные пользовательские функции, которые можно будет вызывать в ваших MDX-запросах или MDX-выражениях. Например, следующий запрос MDX содержит пользовательскую функцию `MyStoredProc`, которая принимает два аргумента и возвращает объект MDX.

```
SELECT MyStoredProc (arg1, arg2) ON COLUMNS FROM CorporateCube
```

Еще один важный факт, который, как мы ожидаем, еще больше подогреет ваш интерес к главе 11, заключается в том, что конструкции .NET сами по себе могут содержать MDX-выражения, благодаря объектной модели, которая позволяет обращаться к объектам MDX! Очевидно, что если вы имеете опыт работы с Analysis Services, то новая версия откроет целый набор новых подходов к решению проблем бизнес-анализа. Поскольку MDX-функции так важны для успешного использования Analysis Services 2008, лучше сразу же перейти к изучению некоторых из этих функций. Несколько позже в данной книге эти функции будут рассмотрены в контексте решения более конкретных задач. А пока просто начните знакомство с MDX-функциями.

Категории MDX-функций

MDX-функции используются для оперирования многомерными базами данных программным способом; перед вами широкое поле для исследований — от просмотра иерархий измерения до выполнения числовых вычислений с данными. В этом разделе MDX-функции будут разбиты на категории определенным образом, чтобы позволить вам наиболее эффективно их изучить. Кроме того, функции которые вам, вероятно, придется использовать чаще других, будут рассмотрены более подробно. Все MDX-функции рассматриваются в приложении к данной книге. Мы разделили MDX-функции на несколько категорий подобно тому, как это сделано в поставляемой с программным продуктом документации по MDX-функциям. Наибольшее число функций мы отнесли к категории *функций набора*. В каждом из последующих разделов рассматривается соответствующая отдельная категория функций. Вызвать функцию MDX можно несколькими способами.

- *.ИмяФункции* (читается функция *точка*)

  Пример. Функция *<Измерение>*.Name возвращает имя объекта, к которому выполняется обращение (это может быть выражение для иерархии или уровня/члена). Возможно, это напомнило вам использование оператора . в VB.NET или C# — проще говоря, идея та же.

```
WITH MEMBER measures.LocationName AS
[Customer].[Country].CurrentMember.Name
SELECT measures.LocationName ON COLUMNS,
Customer.country.members on ROWS
FROM [Adventure Works]
```

- *ИмяФункции*

  Пример. Функция Username используется для получения пользовательского имени, вошедшего в систему пользователя. Она возвращает строковое значение в следующем формате: доменное имя/пользовательское имя. Наиболее часто эта функция используется в MDX-выражениях, связанных с защитой измерения или ячейки от несанкционированного доступа. Далее приведен пример того, как функция Username может использоваться в MDX-выражении.

```
WITH MEMBER Measures.User AS USERNAME
SELECT Measures.User ON 0 FROM [Adventure Works]
```

- *ИмяФункции*()

  Пример. Функция CalculationCurrentPass( ) требует использования круглых скобок, но не использует аргументов. Подробнее о функции CalculationCurrentPass( ) вы можете узнать из приложения (имеется на веб-узле www.wrox.com).

- *ИмяФункции*(*аргументы*)

  Пример. Функция OpeningPeriod([*Выражение_уровня* [, *Выражение_члена*]]) может использовать два аргумента (*Выражение_уровня* и *Выражение_члена*) или только один аргумент *Выражение_члена*. Она наиболее часто используется с измерениями типа Time (Время), но может работать и с другими типами измерений. Данная функция возвращает первый член уровня, заданного выражением *Выражение_уровня*, для члена, определенного выражением *Выражение_члена*. Приведенный ниже пример возвращает первого члена уровня Day члена April временного измерения, принятого по умолчанию.

```
OpeningPeriod(Day, [April])
```

Функции набора

Функции набора, как и предполагает название данной категории, применяются для выполнения операций с наборами кортежей. Такие функции используют в качестве аргументов наборы кортежей, и зачастую результатом их выполнения также является набор. Наиболее широко применяемыми из функций набора являются функции `CrossJoin` и `Filter`, которые вам наверняка придется использовать в своих MDX-запросах. Поэтому мы рассмотрим использование этих функций на примерах.

Функция `CrossJoin` возвращает все возможные комбинации членов наборов, указанных в качестве ее аргументов. Если в функции `Crossjoin` задано N наборов, это приведет к комбинации всех возможных членов в пределах одной оси. Рассмотрим следующий пример.

```
CrossJoin(Выражение_набора [, Выражение_набора ...])


SELECT Measures.[Internet Sales Amount] ON COLUMNS,
CrossJoin( {Product.[Product Line].[Product Line].MEMBERS},
{[Customer].[Country].MEMBERS}) on ROWS
FROM [Adventure Works]
```

Этот запрос создает векторное произведение каждого члена в измерении `Product` (Товар) с каждым членом измерения `Customer` (Клиент) и возвращает для каждой полученной пары значение размерности `Internet Sales Amount` (Сумма Интернет-продаж). Ниже приведены первые несколько строк результатов, получаемых при выполнении рассматриваемого запроса.

		Internet Sales Amount
Accessory	All Customers	$604,053.30
Accessory	Australia	$127,128.61
Accessory	Canada	$82,736.07
Accessory	France	$55,001.21
Accessory	Germany	$54,382.29
Accessory	United Kingdom	$67,636.33
Accessory	United States	$217,168.79
Components	All Customers	(null)
...	...	...

Иногда результаты комбинаций членов набора могут возвращать пустое значение (`Null`). Например, предположим, что имеется один товар, который продается только в Австралии. Тогда в качестве суммы продаж этого товара в других странах будет возвращено значение `Null`. Очевидно, что вам не интересны такие пустые результаты. Они никак не помогают ни в каких бизнес-решениях. Вместо того чтобы извлекать все результаты и затем проверять их на наличие `Null`-значений, лучше воспользоваться специальным методом, который позволяет ограничить (на серверной стороне экземпляра Analysis Services) возвращаемые результаты, исключив из них пустые значения. Кроме того, Analysis Services оптимизирует запрос так, что извлекается и передается только подходящий результат. Для этого используется функция `NonEmptyCrossJoin` или `NonEmpty`. Для этих двух функций используется следующий синтаксис.

```
NonEmptyCrossJoin(
Выражение_набора [,Выражение_набора ...][, Счетчик_наборов])

NonEmpty(Выражение_набора [, Выражение_фильтра_набора])
```

Поэтому для того, чтобы удалить пустые ячейки в рассмотренном ранее запросе на основе функции CrossJoin, вы можете использовать один из следующих запросов, использующих функции NonEmptyCrossJoin и NonEmpty. При использовании функции NonEmptyCrossJoin необходимо применить условие фильтрации по размерности [Internet Sales Amount] и затем извлечь векторное произведение членов первых двух наборов. В данном случае требуется явно задать размерность [Internet Sales Amount] в параметрах функции NonEmptyCrossJoin. Дело в том, что размерность Internet Sales Amount в кубе Adventure Works не является заданной по умолчанию размерностью, а если размерность не указана в качестве параметра функции NonEmptyCrossJoin, то функция будет использовать заданную по умолчанию размерность. При использовании функции NonEmpty сначала выполняется функция CrossJoin, а затем отфильтровываются кортежи, содержащие пустые значения (Null) для сумм Интернет-продаж, как показано во втором из двух приведенных ниже запросе. MDX-функция NonEmpty является новинкой Analysis Services 2005.

```
SELECT Measures.[Internet Sales Amount] ON COLUMNS,
NONEMPTYCROSSJOIN({Product.[Product Line].[Product Line].MEMBERS},
{[Customer].[Country].MEMBERS},Measures.[Internet Sales Amount],2 )
 ON ROWS
FROM [Adventure Works]

SELECT Measures.[Internet Sales Amount] ON COLUMNS,
NONEMPTY(CROSSJOIN( {Product.[Product Line].[Product Line].MEMBERS},
{[Customer].[Country].MEMBERS}), Measures.[Internet Sales Amount])
 ON ROWS
FROM [Adventure Works]
```

Большинство пользовательских и клиентских инструментов, взаимодействующих с экземпляром Analysis Services, широко использует функцию NonEmptyCrossJoin. Более подробно вы узнаете об этой функции в последующих главах данной книги.

Еще одной весьма полезной MDX-функцией является функция Filter. Эта функция позволяет ограничить результаты запроса на основе одного или нескольких условий. Функция Filter использует два аргумента: выражение, задающее набор, и логическое выражение, определяющее условие отбора. Логическое выражение применяется к каждому элементу набора и возвращает набор элементов, удовлетворяющий логическому условию. Для функции Filter используется следующий синтаксис.

```
Filter( Выражение_набора, {Логическое_выражение |
 [CAPTION | KEY | NAME] = Строковое_выражение})
```

Результат показанного ранее запроса на основе функции CrossJoin содержит 35 ячеек. Если вас интересуют только товары, для которых сумма продаж в каждой из стран превышает определенное значение, то можете воспользоваться функцией Filter, как показано ниже.

```
SELECT Measures.[Internet Sales Amount] ON COLUMNS,
FILTER(CROSSJOIN( {Product.[Product Line].[Product Line].MEMBERS},
{[Customer].[Country].MEMBERS}),[Internet Sales Amount] >2000000)
 ON ROWS
FROM [Adventure Works]
```

Этот запрос возвращает информацию только о тех товарах, для которых суммы продаж по каждой из стран (и общая сумма по всем странам) превышают $2000000. Результаты выполнения данного запроса приведены в следующей таблице.

		Internet Sales Amount
Mountain	All Customers	$10251183,52
Mountain	Australia	$2906994,45
Mountain	United States	$3547956,78
Road	All Customers	$14624108,58
Road	Australia	$5029120,41
Road	United States	$4322438,41
Touring	All Customers	$3879331,82

Функции члена измерения

Функции члена измерения используются для выполнения операций с членами, таких как извлечение текущего члена, предыдущего, родительского, дочернего, следующего члена и т.д. Все функции данной категории возвращают член. Одна из наиболее часто используемых функций этой категории называется `ParallelPeriod`. Функция `ParallelPeriod` позволяет извлечь член измерения типа `Time` (Время), основываясь на заданном члене и определенных условиях. Для функции `ParallelPeriod` используется следующий синтаксис.

```
ParallelPeriod([Выражение_уровня [, Числовое_выражение [,
Выражение_члена]]])
```

На рис. 3.5 проиллюстрирована работа функции `ParallelPeriod`. Эта функция возвращает член измерения типа `Time` (вы узнаете больше об измерении типа `Time` в главе 5) относительно указанного члена для определенного временного периода. Например, выражение `ParallelPeriod([Quarter], 1, [April])` возвращает член `[January]`. Возможно, вы удивлены результатом и не вполне понимаете, как он был получен. В последующих пошаговых инструкциях показано выполнение функции `ParallelPeriod` и объясняется, как Analysis Services достигает такого результата.

1. Функция `ParallelPeriod` может использоваться только в сочетании с временными измерениями (т.е. типа `Time`). Для приведенной на рис. 3.5 иллюстрации предположим, что мы имеем дело с временным измерением с иерархией `Calendar` (Календарь), которая содержит уровни `Year` (Год), `Semester` (Семестр), `Quarter` (Квартал) и `Month` (Месяц).
2. Сначала функция `ParallelPeriod` находит родительский член для своего последнего аргумента (заданного как `April` (Апрель)) на указанном уровне Quarter (который задан в качестве первого аргумента). Она определяет, что родительский член для члена `April` — это `Quarter 2`.
3. Затем на основании числового выражения (заданного в качестве второго аргумента функции) определяется родственный член для члена `[Quarter 2]` (т.е. член того же уровня). Положительные значения означают, что искомый родственный член является предшествующим по отношению к текущему члену в совокупности членов данного уровня. Отрицательные значения показывают, что искомый родственный член является последующим по отношению к текущему члену. В данном примере искомый член того же уровня —

это [Quarter 1], поскольку в качестве второго аргумента функции ParallelPeriod задано значение 1.

4. Далее для члена [Quarter 1] находится дочерний член, который расположен на той же позиции, что и член [April] (т.е. на первой). Таким членом является January (Январь).

Функция ParallelPeriod используется для сравнения значений размерности относительно различных временных периодов. Обычно пользователь заинтересован в сравнении сумм продаж за разные кварталы или годы, и в таком случае данная функция реально окажется очень полезной. Большинство клиентских инструментов, взаимодействующих с Analysis Services, активно использует эту функцию.

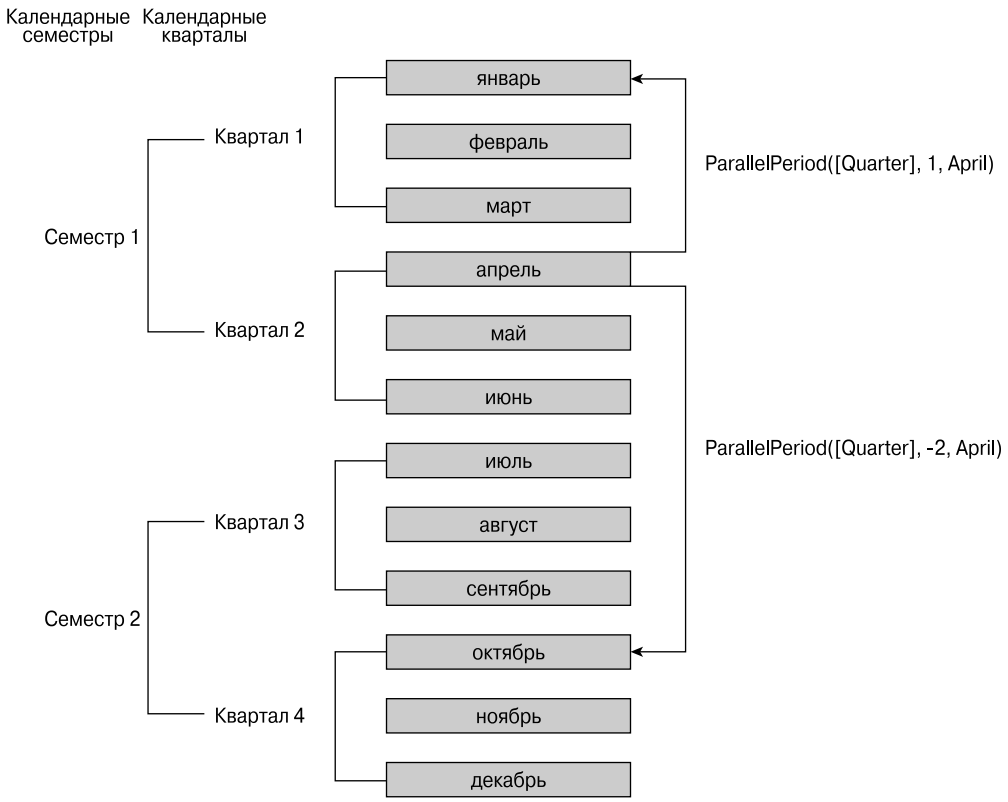

Рис. 3.5. Иллюстрация работы функции ParallelPeriod

Числовые функции

Числовые функции чрезвычайно полезны при определении параметров MDX-запроса или создании вычисляемых размерностей. Обратите внимание на тот факт, что в данной группе представлено множество статистических функций, включая функции для расчета стандартного среднеквадратичного отклонения, выборочной дисперсии и корреляции. Наиболее часто применяемыми из числовых функций являются простая функция под названием Count и ее "близкая родственница" DistinctCount. Функция Count используется для подсчета количества элементов в таком объекте, как измерение, кортеж, набор или уровень. Функция Dis-

tinctCount в свою очередь использует в качестве аргумента выражение, задающее набор кортежей, и возвращает число индивидуальных (недублирующихся) элементов в указанном наборе, а не общее число элементов. Ниже показан синтаксис, используемый обеими упомянутыми функциями.

```
Count(Измерение | Кортеж | Набор| Уровень)
DistinctCount(Выражение_набора)
```

Теперь рассмотрим следующий запрос.

```
WITH MEMBER Measures.CustomerCount AS DistinctCount(
Exists([Customer].[Customer].MEMBERS,
       [Product].[Product Line].Mountain,"Internet Sales"))
SELECT Measures.CustomerCount ON COLUMNS
FROM [Adventure Works]
```

Функция DistinctCount подсчитывает число неповторяющихся членов измерения Customer (Клиент), которые соответствуют клиентам, приобретавшим товары товарной линии Mountain (Горные). Если клиент приобретал товар дважды, то функция DistinctCount посчитает этого клиента только один раз. В данном примере MDX-функция Exists используется для отбора только тех клиентов, которые приобретали товары товарной линии Mountain через Интернет. Более подробно о функции Exists вы узнаете в главе 10. Результатом выполнения функции Exists является набор записей о клиентах, которые приобрели товары товарной линии Mountain. Результатом приведенного выше запроса будет число 9590.

Функции измерения, функции уровня и функции иерархии

Функции, которые относят к данной группе, обычно используют для навигации и манипуляций. Ниже приведен пример использования подобной функции (а конкретно функции Level).

```
SELECT [Date].[Calendar].[Calendar Quarter].[Q1 CY 2004].LEVEL
 ON COLUMNS
FROM [Adventure Works]
```

Данный запрос на самом деле возвращает список всех кварталов. Дело в том, что выражение [Date].[Calendar].[Calendar Quarter].[Q1 CY 2004].LEVEL трактуется программой как [Date].[Calendar Year].[Calendar Semester].[Calendar Quarter]. В результате вы получаете список всех кварталов соответствующего календарного года.

Функции обработки строковых значений

Чтобы извлечь имена наборов, кортежей и членов в форме строки, можно использовать специальные функции, например MemberToStr(<*Выражение_члена*>). А для выполнения обратного преобразования, т.е. преобразования строкового значения в выражение, возвращающее член измерения, можно использовать функцию StrToMember(<*Строка*>). Рассмотрим следующий случай: предположим, имеется клиентское приложение, которое отображает информацию о продажах для всех стран. Когда пользователь выбирает определенную страну, необходимо извлечь информацию о продажах в этой стране из Analysis Services. Поскольку в клиентском приложении страны представлены строковыми значениями, необходимо преобразовать эту строку в соответствующий член измерения, а затем вы сможете извлечь данные. Функции, позволяющие обрабатывать строковые значения, при-

годятся вам во всех тех ситуациях, когда требуется принять некоторые введенные пользователем параметры и преобразовать их в соответствующие объекты MDX. Однако применение функций обработки строковых значений зачастую оборачивается снижением производительности системы. Поэтому мы рекомендуем вам стараться использовать эти функции только тогда, когда это действительно необходимо.

```
SELECT STRTOMEMBER ('[Customer].[Country].[Australia]')
 ON COLUMNS FROM [Adventure Works]
```

Другие функции

Существует еще четыре категории функций. Категории функций `Subcube` (подкуб) и `Array` (массив) содержат по одной функции каждая. Последние две категории это логические функции, которые позволяют выполнять логические (булевы) операции с многомерными объектами, и функции кортежа, с помощью которых осуществляется доступ к кортежам. Кроме того, в Analysis Services 2005 и Analysis Services 2008 введено несколько новых MDX-функций. Более подробно о некоторых из них (таких, как `NonEmpty` и `Exists`) речь пойдет в главе 10 и приложении, представленном в конце этой книги.

Резюме

Примите поздравления, вы освоили первые три главы! В принципе с этого места вы можете переходить с изучению любой другой главы этой книги в произвольном порядке. Но все же хочется порекомендовать сразу же перейти к главе 4. Теперь вы знакомы с фундаментальными элементами MDX — ячейками, членами, кортежами и наборами. Более того, вы узнали, что язык MDX применяется в двух формах: в виде запросов и в форме выражений.

Вы заметили, что MDX-запросы, которые используются для извлечения информации из баз данных Analysis Services, сохраняют внешнее сходство с SQL-запросами, но это сходство выглядит уже не таким явным при более глубоком рассмотрении данного вопроса. MDX-выражения в свою очередь являются простыми и к тому же мощными конструкциями, которые могут включать различные операторы и MDX-функции. Сами по себе выражения не возвращают конечных результатов, как это делают запросы. Выражения позволяют вам определять многомерные объекты и управлять ими, а также данными (посредством вычислений).

Для закрепления базовых знаний об MDX вы изучили общие инструкции запросов (`WITH`, `SELECT`, `FROM` и `WHERE`) и рассмотрели операторы MDX, которые позволяют осуществлять операции добавления, вычитания, умножения и деления, а также логические операторы `AND` и `OR`. Полученные вами знания очень важны для эффективного использования языка MDX. Мы в общих чертах ознакомили вас с одиннадцатью категориями MDX-функций; показали вам четыре способа применения MDX-функций и даже более подробно рассмотрели примеры использования некоторых наиболее часто применяемых функций (таких, как `Filter`, `ParallelPeriod`, `MemberToStr` и `StrToMember`). Более сложные концепции языка MDX и его функций будут рассмотрены в главах 8, 9, 10, 11 и 12. Все поддерживаемые Analysis Services 2008 функции (вместе с примерами их использования) перечислены в приложении, которое можно загрузить из Интернета. В главе 4 вы подробно ознакомитесь с созданием источника данных, представлением источника данных, а также узнаете, как работать с несколькими представлениями источника данных в едином проекте.