

# Введение в OLAP

Алексей Федоров, Наталия Елманова

## Часть 1. Основы OLAP

Что такое хранилище данных

Что такое OLAP

Многомерные кубы

Некоторые термины и понятия

В цикле статей «Введение в базы данных», публиковавшемся в последнее время, мы обсуждали различные технологии и программные средства, применяемые при создании информационных систем — настольные и серверные СУБД, средства проектирования данных, средства разработки приложений, а также Business Intelligence — средства анализа и обработки данных масштаба предприятия, которые в настоящее время становятся все более популярными в мире, в том числе и в нашей стране. Отметим, однако, что вопросы применения средств Business Intelligence и технологии, используемые при создании приложений такого класса, в отечественной литературе пока еще освещены недостаточно. В новом цикле статей мы попробуем восполнить этот пробел и рассказать о том, что представляют собой технологии, лежащие в основе подобных приложений. В качестве примеров реализации мы будем использовать в основном OLAP-технологии фирмы Microsoft (главным образом Analysis Services в Microsoft SQL Server 2000), но надеемся, что основная часть материала будет полезна и пользователям других средств.

Первая статья в данном цикле посвящена основам OLAP (On-Line Analytical Processing) — технологии многомерного анализа данных. В ней мы рассмотрим концепции хранилищ данных и OLAP, требования к хранилищам данных и OLAP-средствам, логическую организацию OLAP-данных, а также основные термины и понятия, применяемые при обсуждении многомерного анализа.

### Что такое хранилище данных

Информационные системы масштаба предприятия, как правило, содержат приложения, предназначенные для комплексного многомерного анализа данных, их динамики, тенденций и т.п. Такой анализ в конечном итоге призван содействовать принятию решений. Нередко эти системы так и называются — системы поддержки принятия решений.

Принять любое управленческое решение невозможно не обладая необходимой для этого информацией, обычно количественной. Для этого необходимо создание хранилищ данных (Data warehouses), то есть процесс сбора, отсеивания и предварительной обработки данных с целью предоставления результирующей информации пользователям для статистического анализа (а нередко и создания аналитических отчетов).

Ральф Кимбалл (Ralph Kimball), один из авторов концепции хранилищ данных, описывал хранилище данных как «место, где люди могут получить доступ к своим данным» (см., например, Ralph Kimball, «The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses», John Wiley & Sons, 1996 и «The Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse», John Wiley & Sons, 2000). Он же сформулировал и основные требования к хранилищам данных:

- поддержка высокой скорости получения данных из хранилища;
- поддержка внутренней непротиворечивости данных;
- возможность получения и сравнения так называемых срезов данных (slice and dice);
- наличие удобных утилит просмотра данных в хранилище;
- полнота и достоверность хранимых данных;
- поддержка качественного процесса пополнения данных.

Удовлетворять всем перечисленным требованиям в рамках одного и того же продукта зачастую не удастся. Поэтому для реализации хранилищ данных обычно используется несколько продуктов, одни из которых представляют собой собственно средства хранения данных, другие — средства их извлечения и просмотра, третьи — средства их пополнения и т.д.

Типичное хранилище данных, как правило, отличается от обычной реляционной базы данных. Во-первых, обычные базы данных предназначены для того, чтобы помочь пользователям выполнять повседневную работу, тогда как хранилища данных предназначены для принятия решений. Например, продажа товара и выписка счета производятся с использованием базы данных, предназначенной для обработки транзакций, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, — с помощью хранилища данных.

Во-вторых, обычные базы данных подвержены постоянным изменениям в процессе работы пользователей, а хранилище данных относительно стабильно: данные в нем обычно обновляются согласно расписанию (например, еженедельно, ежедневно или ежечасно — в зависимости от потребностей). В идеале процесс пополнения представляет собой просто добавление новых данных за определенный период времени без изменения прежней информации, уже находящейся в хранилище.

И в-третьих, обычные базы данных чаще всего являются источником данных, попадающих в хранилище. Кроме того, хранилище может пополняться за счет внешних источников, например статистических отчетов.

## **Что такое OLAP**

Системы поддержки принятия решений обычно обладают средствами предоставления пользователю агрегатных данных для различных выборок из исходного набора в удобном для восприятия и анализа виде. Как правило, такие агрегатные функции образуют многомерный (и, следовательно, нереляционный) набор данных (нередко называемый гиперкубом или метакубом), оси которого содержат параметры, а ячейки — зависящие от них агрегатные данные<sup>1</sup>. Вдоль каждой оси данные могут быть организованы в виде иерархии, представляющей различные уровни их детализации. Благодаря такой модели данных пользователи могут формулировать сложные запросы, генерировать отчеты, получать подмножества данных.

Технология комплексного многомерного анализа данных получила название OLAP (On-Line Analytical Processing). OLAP — это ключевой компонент организации хранилищ данных. Концепция OLAP была описана в 1993 году Эдгаром Коддом, известным исследователем баз данных и автором реляционной модели данных (см. E.F. Codd, S.B. Codd, and C.T. Salley, Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical report, 1993). В 1995 году на основе требований, изложенных Коддом, был сформулирован так называемый тест FASMI (Fast Analysis of Shared Multidimensional Information — быстрый

---

<sup>1</sup> Храниться такие данные могут и в реляционных таблицах, но в данном случае мы говорим о логической организации данных, а не о физической реализации их хранения.

анализ разделяемой многомерной информации), включающий следующие требования к приложениям для многомерного анализа:

- предоставление пользователю результатов анализа за приемлемое время (обычно не более 5 с), пусть даже ценой менее детального анализа;
- возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;
- многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;
- многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий (это — ключевое требование OLAP);
- возможность обращаться к любой нужной информации независимо от ее объема и места хранения.

Следует отметить, что OLAP-функциональность может быть реализована различными способами, начиная с простейших средств анализа данных в офисных приложениях и заканчивая распределенными аналитическими системами, основанными на серверных продуктах. Но прежде чем говорить о различных реализациях этой функциональности, давайте рассмотрим, что же представляют собой кубы OLAP с логической точки зрения.

## Многомерные кубы

В данном разделе мы более подробно рассмотрим концепцию OLAP и многомерных кубов. В качестве примера реляционной базы данных, который мы будем использовать для иллюстрации принципов OLAP, воспользуемся базой данных Northwind, входящей в комплекты поставки Microsoft SQL Server или Microsoft Access и представляющей собой типичную базу данных, хранящую сведения о торговых операциях компании, занимающейся оптовыми поставками продовольствия. К таким данным относятся сведения о поставщиках, клиентах, компаниях, осуществляющих доставку, список поставляемых товаров и их категорий, данные о заказах и заказанных товарах, список сотрудников компании. Подробное описание базы данных Northwind можно найти в справочных системах Microsoft SQL Server или Microsoft Access — здесь за недостатком места мы его не приводим.

Для рассмотрения концепции OLAP воспользуемся представлением Invoices и таблицами Products и Categories из базы данных Northwind, создав запрос, в результате которого получим подробные сведения о всех заказанных товарах и выписанных счетах:

```
SELECT dbo.Invoices.Country,
       dbo.Invoices.City,
       dbo.Invoices.CustomerName,
       dbo.Invoices.Salesperson,
       dbo.Invoices.OrderDate,
       dbo.Categories.CategoryName,
       dbo.Invoices.ProductName,
       dbo.Invoices.ShipperName,
       dbo.Invoices.ExtendedPrice
FROM dbo.Products INNER JOIN
     dbo.Categories ON dbo.Products.CategoryID = dbo.Categories.CategoryID INNER
JOIN
     dbo.Invoices ON dbo.Products.ProductID = dbo.Invoices.ProductID
```

В Access 2000 аналогичный запрос имеет вид:

```
SELECT Invoices.Country, Invoices.City,
Invoices.Customers.CompanyName AS
CustomerName, Invoices.Salesperson,
```

```

Invoices.OrderDate, Categories.CategoryName,
Invoices.ProductName,
Invoices.Shippers.CompanyName AS
ShipperName, Invoices.ExtendedPrice
FROM Categories INNER JOIN (Invoices INNER
JOIN Products ON Invoices.ProductID =
Products.ProductID) ON Categories.CategoryID =
Products.CategoryID;

```

Этот запрос обращается к представлению Invoices, содержащему сведения обо всех выписанных счетах, а также к таблицам Categories и Products, содержащим сведения о категориях продуктов, которые заказывались, и о самих продуктах соответственно. В результате этого запроса мы получим набор данных о заказах, включающий категорию и наименование заказанного товара, дату размещения заказа, имя сотрудника, выписавшего счет, город, страну и название компании-заказчика, а также наименование компании, отвечающей за доставку.

Для удобства сохраним этот запрос в виде представления, назвав его Invoices1. Результат обращения к этому представлению приведен на рис. 1.

Column Alias Table Output Sort Type Sort Order Criteria Or... Or... Or...

Column	Alias	Table	Output	Sort Type	Sort Order	Criteria	Or...	Or...	Or...
Country		Invoices	✓						
City		Invoices	✓						
CustomerName		Invoices	✓						
Salesperson		Invoices	✓						
OrderDate		Invoices	✓						
CategoryName		Categories	✓						
ProductName		Products	✓						
ShipperName		Invoices	✓						

SELECT dbo.Invoices.Country, dbo.Invoices.City, dbo.Invoices.CustomerName, dbo.Invoices.Salesperson, dbo.Invoices.OrderDate, dbo.Categories.CategoryName, dbo.Invoices.ProductName, dbo.Invoices.ShipperName, dbo.Invoices.ExtendedPrice FROM dbo.Products INNER JOIN dbo.Categories ON dbo.Products.CategoryID = dbo.Categories.CategoryID INNER JOIN dbo.Invoices ON dbo.Products.ProductID = dbo.Invoices.ProductID

Country	City	CustomerName	Salesperson	OrderDate	CategoryName	ProductName	ShipperName	ExtendedPrice
France	Reims	Vins et alcools Chevalier	Steven Buchanan	04.07.1996	Dairy Products	Queso Cabrales	Speedy Express	168
France	Reims	Vins et alcools Chevalier	Steven Buchanan	04.07.1996	Grains/Cereals	Singaporean Hokkien Fried Mee	Speedy Express	98
France	Reims	Vins et alcools Chevalier	Steven Buchanan	04.07.1996	Dairy Products	Mozzarella di Giovanni	Speedy Express	174
Germany	Munster	Toins Spezialitäten	Michael Suyama	05.07.1996	Produce	Tofu	Speedy Express	167.4
Germany	Munster	Toins Spezialitäten	Michael Suyama	05.07.1996	Produce	Manjimup Dried Apples	Speedy Express	1696
Brazil	Rio de Janeiro	Hanari Carnes	Margaret Peacock	08.07.1996	Seafood	Jack's New England Clam Chowder	United Package	77
Brazil	Rio de Janeiro	Hanari Carnes	Margaret Peacock	08.07.1996	Produce	Manjimup Dried Apples	United Package	1261.4
Brazil	Rio de Janeiro	Hanari Carnes	Margaret Peacock	08.07.1996	Condiments	Louisiana Fiery Hot Pepper Sauce	United Package	214.2

Рис. 1. Результат обращения к представлению Invoices1

Какие агрегатные данные мы можем получить на основе этого представления? Обычно это ответы на вопросы типа:

- Какова суммарная стоимость заказов, сделанных клиентами из Франции?
- Какова суммарная стоимость заказов, сделанных клиентами из Франции и доставленных компанией Speedy Express?
- Какова суммарная стоимость заказов, сделанных клиентами из Франции в 1997 году и доставленных компанией Speedy Express?

Переведем эти вопросы в запросы на языке SQL<sup>2</sup> (табл. 1).

Таблица 1

Вопрос	SQL-запрос
Какова суммарная стоимость заказов, сделанных клиентами из Франции?	SELECT SUM (ExtendedPrice) FROM invoices1 WHERE Country='France'
Какова суммарная стоимость заказов, сделанных клиентами из Франции и доставленных компанией Speedy Express?	SELECT SUM (ExtendedPrice) FROM invoices1 WHERE Country='France' AND ShipperName='Speedy Express'
Какова суммарная стоимость заказов, сделанных клиентами из Франции в 1996 году и доставленных компанией Speedy Express?	SELECT SUM (ExtendedPrice) FROM Ord_pmt WHERE CompanyName='Speedy Express' AND OrderDate BETWEEN 'December 31, 1995' AND 'April 1, 1996' AND ShipperName='Speedy Express'

Результатом любого из перечисленных выше запросов является число. Если в первом из запросов заменить параметр 'France' на 'Austria' или на название иной страны, можно снова выполнить этот запрос и получить другое число. Выполнив эту процедуру со всеми странами, мы получим следующий набор данных (ниже показан фрагмент):

Country	SUM (ExtendedPrice)
Argentina	7327.3
Austria	110788.4
Belgium	28491.65
Brazil	97407.74
Canada	46190.1
Denmark	28392.32
Finland	15296.35
France	69185.48
Germany	209373.6
...	...

Полученный набор агрегатных значений (в данном случае — сумм) может быть интерпретирован как одномерный набор данных. Этот же набор данных можно получить и в результате запроса с предложением GROUP BY следующего вида:

```
SELECT Country, SUM (ExtendedPrice) FROM invoices1
GROUP BY Country
```

Теперь обратимся ко второму из приведенных выше запросов, который содержит два условия в предложении WHERE. Если выполнять этот запрос, подставляя в него все возможные значения параметров Country и ShipperName, мы получим двухмерный набор данных следующего вида (ниже показан фрагмент):

Country	ShipperName	
	Federal Shipping	Speedy Express United Package
Argentina	1 210.30	1 816.20 5 092.60
Austria	40 870.77	41 004.13 46 128.93

<sup>2</sup> Аналогичные вопросы, сформулированные на английском языке, можно превратить в SQL-запросы с помощью Microsoft English Query, однако рассмотрение подобных средств выходит за рамки данной статьи.

Belgium	11 393.30	4 717.56	17 713.99
Brazil	16 514.56	35 398.14	55 013.08
Canada	19 598.78	5 440.42	25 157.08
Denmark	18 295.30	6 573.97	7 791.74
Finland	4 889.84	5 966.21	7 954.00
France	28 737.23	21 140.18	31 480.90
Germany	53 474.88	94 847.12	81 962.58
...	...	...	...

Такой набор данных называется сводной таблицей (pivot table) или кросс-таблицей (cross table, crosstab). Создавать подобные таблицы позволяют многие электронные таблицы и настольные СУБД — от Paradox для DOS до Microsoft Excel 2000. Вот так, например, выглядит подобный запрос в Microsoft Access 2000:

```
TRANSFORM Sum(Invoices1.ExtendedPrice) AS SumOfExtendedPrice
SELECT Invoices1.Country
FROM Invoices1
GROUP BY Invoices1.Country
PIVOT Invoices1.ShipperName;
```

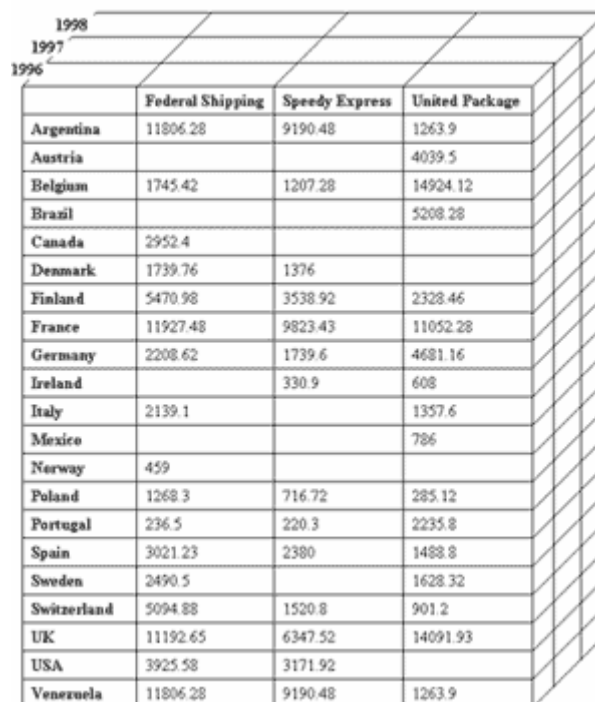
Агрегатные данные для подобной сводной таблицы можно получить и с помощью обычного запроса GROUP BY:

```
SELECT Country,ShipperName, SUM (ExtendedPrice) FROM invoices1
GROUP BY COUNTRY,ShipperName
```

Отметим, однако, что результатом этого запроса будет не сама сводная таблица, а лишь набор агрегатных данных для ее построения (ниже показан фрагмент):

Country	ShipperName	SUM (ExtendedPrice)
Argentina	Federal Shipping	845.5
Austria	Federal Shipping	35696.78
Belgium	Federal Shipping	8747.3
Brazil	Federal Shipping	13998.26
...	...	...

Третий из рассмотренных выше запросов имеет уже три параметра в условии WHERE. Варьируя их, мы получим трехмерный набор данных (рис. 2).



	Federal Shipping	Speedy Express	United Package
Argentina	11806.28	9190.48	1263.9
Austria			4039.5
Belgium	1745.42	1207.28	14924.12
Brazil			5208.28
Canada	2952.4		
Denmark	1739.76	1376	
Finland	5470.98	3538.92	2328.46
France	11927.48	9823.43	11052.28
Germany	2208.62	1739.6	4681.16
Ireland		330.9	608
Italy	2139.1		1357.6
Mexico			786
Norway	459		
Poland	1268.3	716.72	285.12
Portugal	236.5	220.3	2235.8
Spain	3021.23	2380	1488.8
Sweden	2490.5		1628.32
Switzerland	5094.88	1520.8	901.2
UK	11192.65	6347.52	14091.93
USA	3925.58	3171.92	
Venezuela	11806.28	9190.48	1263.9

Рис. 2. Трехмерный набор агрегатных данных

Ячейки куба, показанного на рис. 2, содержат агрегатные данные, соответствующие находящимся на осях куба значениям параметров запроса в предложении WHERE.

Можно получить набор двухмерных таблиц с помощью сечения куба плоскостями, параллельными его граням (для их обозначения используют термины cross-sections и slices).

Очевидно, что данные, содержащиеся в ячейках куба, можно получить и с помощью соответствующего запроса с предложением GROUP BY. Кроме того, некоторые электронные таблицы (в частности, Microsoft Excel 2000) также позволяют построить трехмерный набор данных и просматривать различные сечения куба, параллельные его грани, изображенной на листе рабочей книги (workbook).

Если в предложении WHERE содержится четыре или более параметров, результирующий набор значений (также называемый OLAP-кубом) может быть 4-мерным, 5-мерным и т.д.

Рассмотрев, что представляют собой многомерные OLAP-кубы, перейдем к некоторым ключевым терминам и понятиям, используемым при многомерном анализе данных.

## Некоторые термины и понятия

Наряду с суммами в ячейках OLAP-куба могут содержаться результаты выполнения иных агрегатных функций языка SQL, таких как MIN, MAX, AVG, COUNT, а в некоторых случаях — и других (дисперсии, среднеквадратичного отклонения и т.д.). Для описания значений данных в ячейках используется термин summary (в общем случае в одном кубе их может быть несколько), для обозначения исходных данных, на основе которых они вычисляются, — термин measure, а для обозначения параметров запросов — термин dimension (переводимый на русский язык обычно как «измерение», когда речь идет об OLAP-кубах, и как «размерность», когда речь идет о хранилищах данных). Значения, откладываемые на осях, называются членами измерений (members).

Говоря об измерениях, следует упомянуть о том, что значения, наносимые на оси, могут иметь различные уровни детализации. Например, нас может интересовать суммарная стоимость заказов, сделанных клиентами в разных странах, либо суммарная стоимость заказов, сделанных иногородними клиентами или даже отдельными клиентами. Естественно, результирующий набор агрегатных данных во втором и третьем случаях будет более детальным, чем в первом. Заметим, что возможность получения агрегатных данных с различной степенью детализации соответствует одному из требований, предъявляемых к хранилищам данных, — требованию доступности различных срезов данных для сравнения и анализа.

Поскольку в рассмотренном примере в общем случае в каждой стране может быть несколько городов, а в городе — несколько клиентов, можно говорить об иерархиях значений в измерениях. В этом случае на первом уровне иерархии располагаются страны, на втором — города, а на третьем — клиенты (рис. 3).

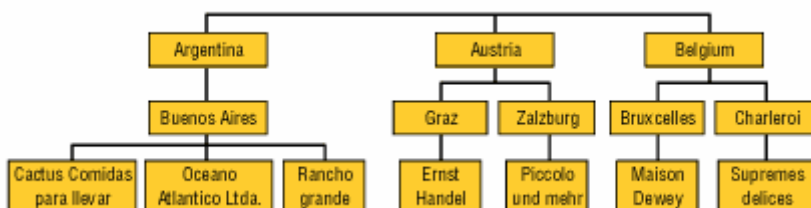


Рис. 3. Иерархия в измерении, связанном с географическим положением клиентов

Отметим, что иерархии могут быть сбалансированными (balanced), как, например, иерархия, представленная на рис. 3, а также иерархии, основанные на данных типа «дата—время», и несбалансированными (unbalanced). Типичный пример несбалансированной иерархии — иерархия типа «начальник—подчиненный» (ее можно построить, например, используя значения поля Salesperson исходного набора данных из рассмотренного выше примера), представлен на рис. 4.

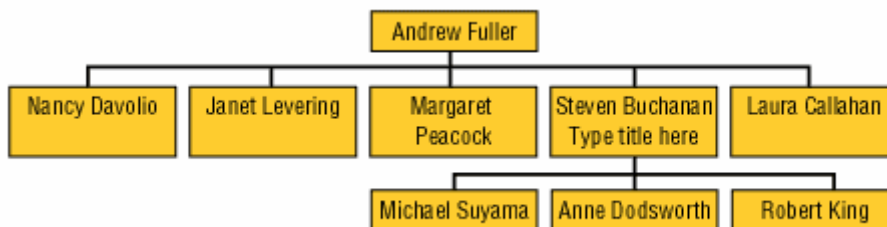
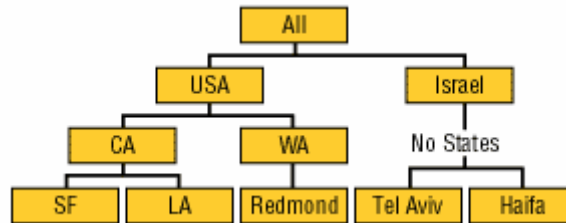


Рис. 4. Несбалансированная иерархия

Иногда для таких иерархий используется термин Parent-child hierarchy.

Существуют также иерархии, занимающие промежуточное положение между сбалансированными и несбалансированными (они обозначаются термином ragged — «неровный»). Обычно они содержат такие члены, логические «родители» которых находятся не на непосредственно вышестоящем уровне (например, в географической иерархии есть уровни Country, City и State, но при этом в наборе данных имеются страны, не имеющие штатов или регионов между уровнями Country и City; рис. 5).





*Рис. 5. «Неровная» иерархия*

Отметим, что несбалансированные и «неровные» иерархии поддерживаются далеко не всеми OLAP-средствами. Например, в Microsoft Analysis Services 2000 поддерживаются оба типа иерархии, а в Microsoft OLAP Services 7.0 — только сбалансированные. Различным в разных OLAP-средствах может быть и число уровней иерархии, и максимально допустимое число членов одного уровня, и максимально возможное число самих измерений.

## Часть 2. Хранилища данных

Типичная структура хранилищ данных

Таблица фактов

Таблицы измерений

OLAP на клиенте и на сервере

Технические аспекты многомерного хранения данных

Первая статья данного цикла была посвящена основам OLAP (On-Line Analytical Processing) — технологии многомерного анализа данных. В ней мы обсудили концепции хранилищ данных и OLAP, требования к хранилищам данных и OLAP-средствам, логическую организацию OLAP-данных, а также основные термины и понятия, относящиеся к многомерному анализу.

В настоящей статье мы рассмотрим типичную структуру хранилищ данных, поговорим о том, что представляет собой OLAP на клиенте и на сервере, а также обсудим некоторые технические аспекты многомерного хранения данных.

### Типичная структура хранилищ данных

Как мы уже знаем, конечной целью использования OLAP является анализ данных и представление результатов этого анализа в виде, удобном для восприятия и принятия решений. Основная идея OLAP заключается в построении многомерных кубов, которые будут доступны для пользовательских запросов. Однако исходные данные для построения OLAP-кубов обычно хранятся в реляционных базах данных. Нередко это специализированные реляционные базы данных, называемые также хранилищами данных (Data Warehouse). В отличие от так называемых оперативных баз данных, с которыми работают приложения, модифицирующие данные, хранилища данных предназначены исключительно для обработки и анализа информации, поэтому проектируются они таким образом, чтобы время выполнения запросов к ним было минимальным. Обычно данные копируются в хранилище из оперативных баз данных согласно определенному расписанию.

Типичная структура хранилища данных существенно отличается от структуры обычной реляционной СУБД. Как правило, эта структура денормализована (это позволяет повысить скорость выполнения запросов), поэтому может допускать избыточность данных.

Для дальнейших примеров мы снова воспользуемся базой данных Northwind, входящей в комплекты поставки Microsoft SQL Server и Microsoft Access. Ее структура данных приведена на рис. 1.

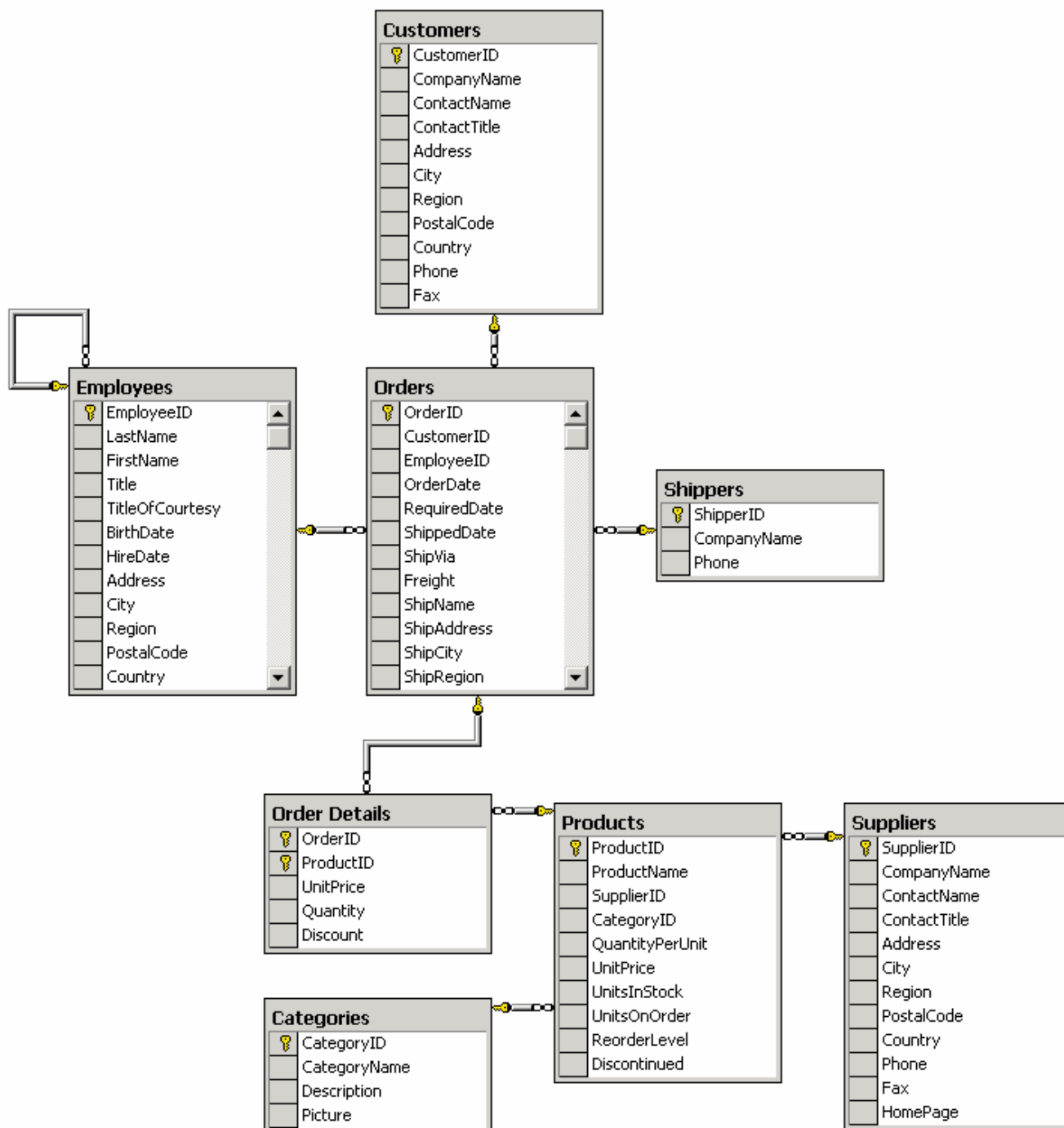


Рис. 1. Структура базы данных Northwind

Основными составляющими структуры хранилищ данных являются таблица фактов (fact table) и таблицы измерений (dimension tables).

## Таблица фактов

Таблица фактов является основной таблицей хранилища данных. Как правило, она содержит сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться. Обычно говорят о четырех наиболее часто встречающихся типах фактов. К ним относятся:

- факты, связанные с транзакциями (Transaction facts). Они основаны на отдельных событиях (типичными примерами которых являются телефонный звонок или снятие денег со счета с помощью банкомата);

- факты, связанные с «моментальными снимками» (Snapshot facts). Основаны на состоянии объекта (например, банковского счета) в определенные моменты времени, например на конец дня или месяца. Типичными примерами таких фактов являются объем продаж за день или дневная выручка;
- факты, связанные с элементами документа (Line-item facts). Основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, количестве, цене, проценте скидки);
- факты, связанные с событиями или состоянием объекта (Event or state facts). Представляют возникновение события без подробностей о нем (например, просто факт продажи или факт отсутствия таковой без иных подробностей).

Для примера рассмотрим факты, связанные с элементами документа (в данном случае счета, выставленного за товар).

Таблица фактов, как правило, содержит уникальный составной ключ, объединяющий первичные ключи таблиц измерений. Чаще всего это целочисленные значения либо значения типа «дата/время» — ведь таблица фактов может содержать сотни тысяч или даже миллионы записей, и хранить в ней повторяющиеся текстовые описания, как правило, невыгодно — лучше поместить их в меньшие по объему таблицы измерений. При этом как ключевые, так и некоторые неключевые поля должны соответствовать будущим измерениям OLAP-куба. Помимо этого таблица фактов содержит одно или несколько числовых полей, на основании которых в дальнейшем будут получены агрегатные данные.

Пример таблицы фактов, которая может быть построена на основе базы данных Northwind, приведен на рис. 2.

SQL Server Enterprise Manager - [3:Data in Table 'Sales\_Fact' in 'Northwind\_Mart' on 'MAINDESK']

Console Window Help

A toolbar containing various icons for database management, including a server icon, a folder icon, a table icon, a query icon, a refresh icon, a save icon, a delete icon, a print icon, a help icon, and a search icon.

	TimeKey	CustomerKey	ShipperKey	ProductKey	EmployeeKey	RequiredDate	LineItemFreight	LineItemTotal	LineItemQuantity	LineItemDiscount
3	85	4	11	5	01.08.1996	14.3904	168	12	0	
5	85	4	42	5	01.08.1996	11.992	98	10	0	
5	85	4	72	5	01.08.1996	5.996	174	5	0	
1	79	1	14	6	16.08.1996	2.1321	167.4	9	0	
1	79	1	51	6	16.08.1996	9.476	1696	40	0	
3	34	2	41	4	05.08.1996	10.971	77	10	0	
3	34	2	51	4	05.08.1996	38.3985	1484	35	222.6	
3	34	2	65	4	05.08.1996	16.4565	252	15	37.8	
4	84	1	22	3	05.08.1996	6.0492	100.8	6	5.04	
4	84	1	57	3	05.08.1996	15.123	234	15	11.7	
4	84	1	65	3	05.08.1996	20.164	336	20	0	
2	76	2	20	4	06.08.1996	19.54	2592	40	129.6	
2	76	2	33	4	06.08.1996	12.2125	50	25	2.5	
2	76	2	60	4	06.08.1996	19.54	1088	40	0	
5	34	2	31	3	24.07.1996	11.404	200	20	0	

Рис. 2. Пример таблицы фактов

Sales_Fact	
Key	TimeKey
Key	CustomerKey
Key	ShipperKey
Key	ProductKey
Key	EmployeeKey
	RequiredDate
	LineItemFreight
	LineItemTotal
	LineItemQuantity
	LineItemDiscount

В данном примере измерениям будущего куба соответствуют первые шесть полей, а агрегатным данным — последние четыре.

Отметим, что для многомерного анализа пригодны таблицы фактов, содержащие как можно более подробные данные (то есть соответствующие членам нижних уровней иерархии соответствующих измерений). В данном случае предпочтительнее взять за основу факты продажи товаров отдельным заказчикам, а не суммы продаж для разных стран — последние все равно будут

вычислены OLAP-средством. Исключение можно сделать, пожалуй, только для клиентских OLAP-средств (о них мы поговорим чуть позже), поскольку в силу ряда ограничений они не могут манипулировать большими объемами данных.

Отметим, что в таблице фактов нет никаких сведений о том, как группировать записи при вычислении агрегатных данных. Например, в ней есть идентификаторы продуктов или клиентов, но отсутствует информация о том, к какой категории относится данный продукт или в каком городе находится данный клиент. Эти сведения, в дальнейшем используемые для построения иерархий в измерениях куба, содержатся в таблицах измерений.

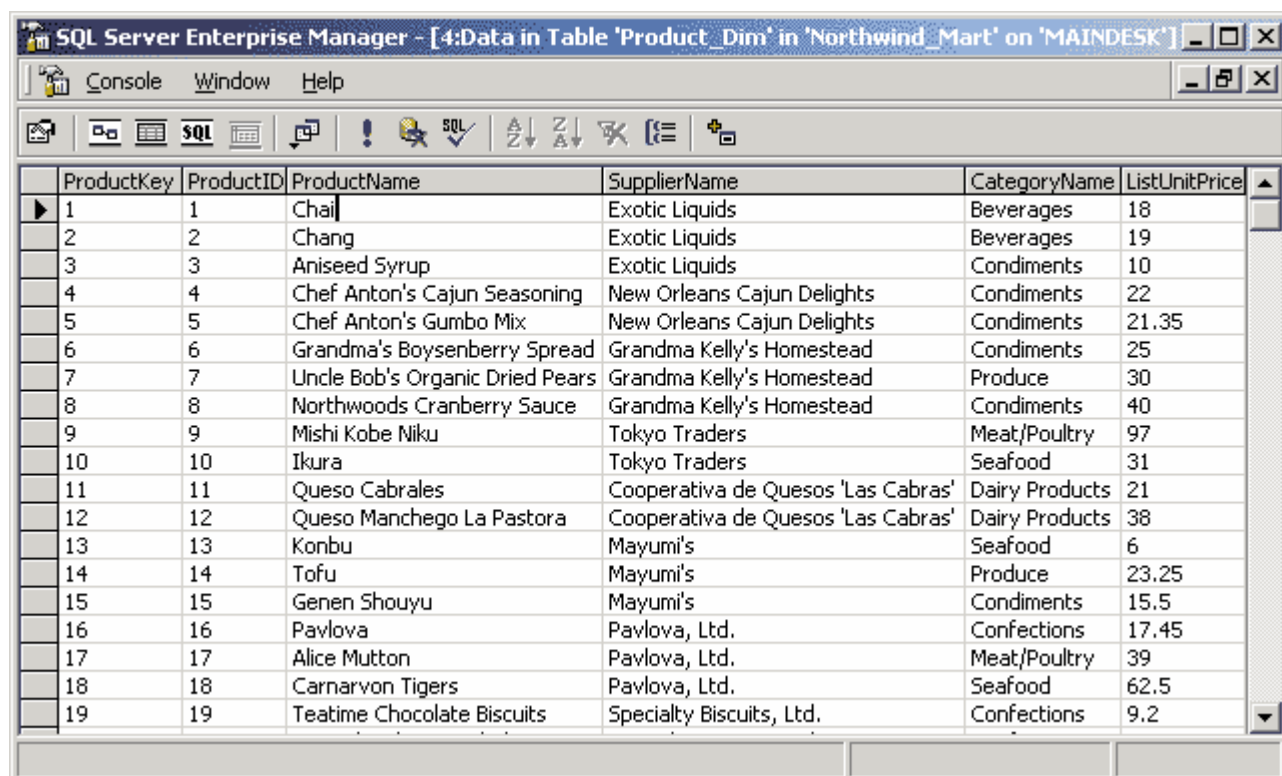
## Таблицы измерений

Таблицы измерений содержат неизменяемые либо редко изменяемые данные. В подавляющем большинстве случаев эти данные представляют собой по одной записи для каждого члена нижнего уровня иерархии в измерении. Таблицы измерений также содержат как минимум одно описательное поле (обычно с именем члена измерения) и, как правило, целочисленное ключевое поле (обычно это суррогатный ключ<sup>3</sup>) для однозначной идентификации члена измерения. Если будущее измерение, основанное на данной таблице измерений, содержит иерархию, то таблица измерений также может содержать поля, указывающие на «родителя» данного члена в этой иерархии. Нередко (но не всегда) таблица измерений может содержать и поля, указывающие на «прародителей», и иных «предков» в данной иерархии (это обычно характерно для сбалансированных иерархий), а также дополнительные атрибуты членов измерений, содержащиеся в исходной оперативной базе данных (например, адреса и телефоны клиентов).

Каждая таблица измерений должна находиться в отношении «один ко многим» с таблицей фактов.

Отметим, что скорость роста таблиц измерений должна быть незначительной по сравнению со скоростью роста таблицы фактов; например, добавление новой записи в таблицу измерений, характеризующую товары, производится только при появлении нового товара, не продававшегося ранее.

Пример таблицы измерений приведен на рис. 3.



ProductKey	ProductID	ProductName	SupplierName	CategoryName	ListUnitPrice
1	1	Chai	Exotic Liquids	Beverages	18
2	2	Chang	Exotic Liquids	Beverages	19
3	3	Aniseed Syrup	Exotic Liquids	Condiments	10
4	4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	Condiments	22
5	5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights	Condiments	21.35
6	6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments	25
7	7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce	30
8	8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments	40
9	9	Mishi Kobe Niku	Tokyo Traders	Meat/Poultry	97
10	10	Ikura	Tokyo Traders	Seafood	31
11	11	Queso Cabrales	Cooperativa de Quesos 'Las Cabras'	Dairy Products	21
12	12	Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'	Dairy Products	38
13	13	Konbu	Mayumi's	Seafood	6
14	14	Tofu	Mayumi's	Produce	23.25
15	15	Genen Shouyu	Mayumi's	Condiments	15.5
16	16	Pavlova	Pavlova, Ltd.	Confections	17.45
17	17	Alice Mutton	Pavlova, Ltd.	Meat/Poultry	39
18	18	Carnarvon Tigers	Pavlova, Ltd.	Seafood	62.5
19	19	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.	Confections	9.2

<sup>3</sup> Суррогатный ключ — искусственно созданный первичный ключ, обычно генерируемый автоматически.

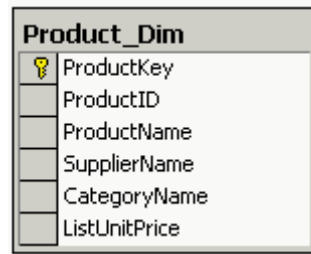


Рис. 3. Пример таблицы измерений

Одно измерение куба может содержаться как в одной таблице (в том числе и при наличии нескольких уровней иерархии), так и в нескольких связанных таблицах, соответствующих различным уровням иерархии в измерении. Если каждое измерение содержится в одной таблице, такая схема хранилища данных носит название «звезда» (star schema). Пример такой схемы приведен на рис. 4.

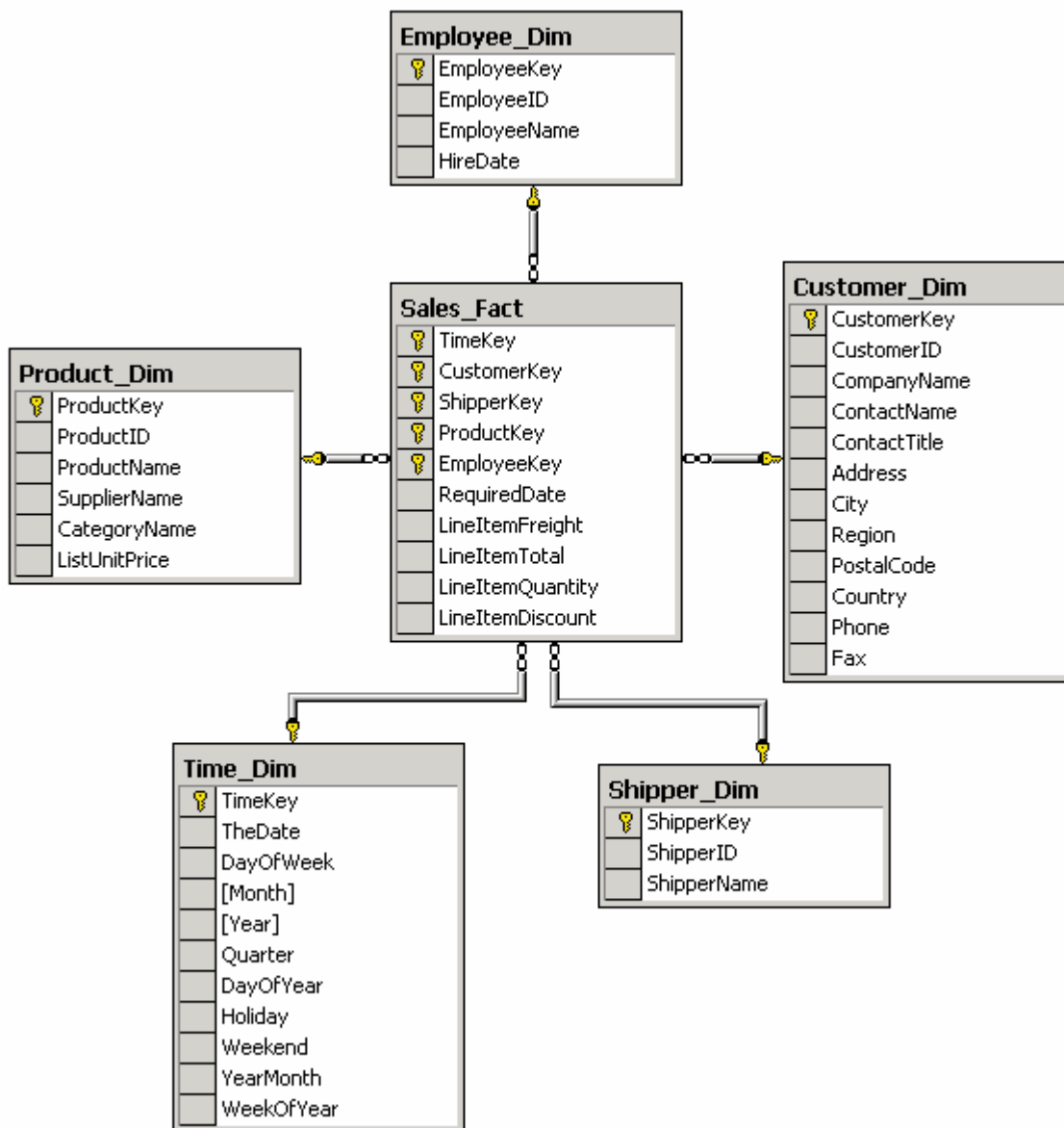


Рис. 4. Пример схемы «звезда»

Если же хотя бы одно измерение содержится в нескольких связанных таблицах, такая схема хранилища данных носит название «снежинка» (snowflake schema). Дополнительные

таблицы измерений в такой схеме, обычно соответствующие верхним уровням иерархии измерения и находящиеся в соотношении «один ко многим» в главной таблице измерений, соответствующей нижнему уровню иерархии, иногда называют консольными таблицами (outrigger table). Пример схемы «снежинка» приведен на рис. 5.

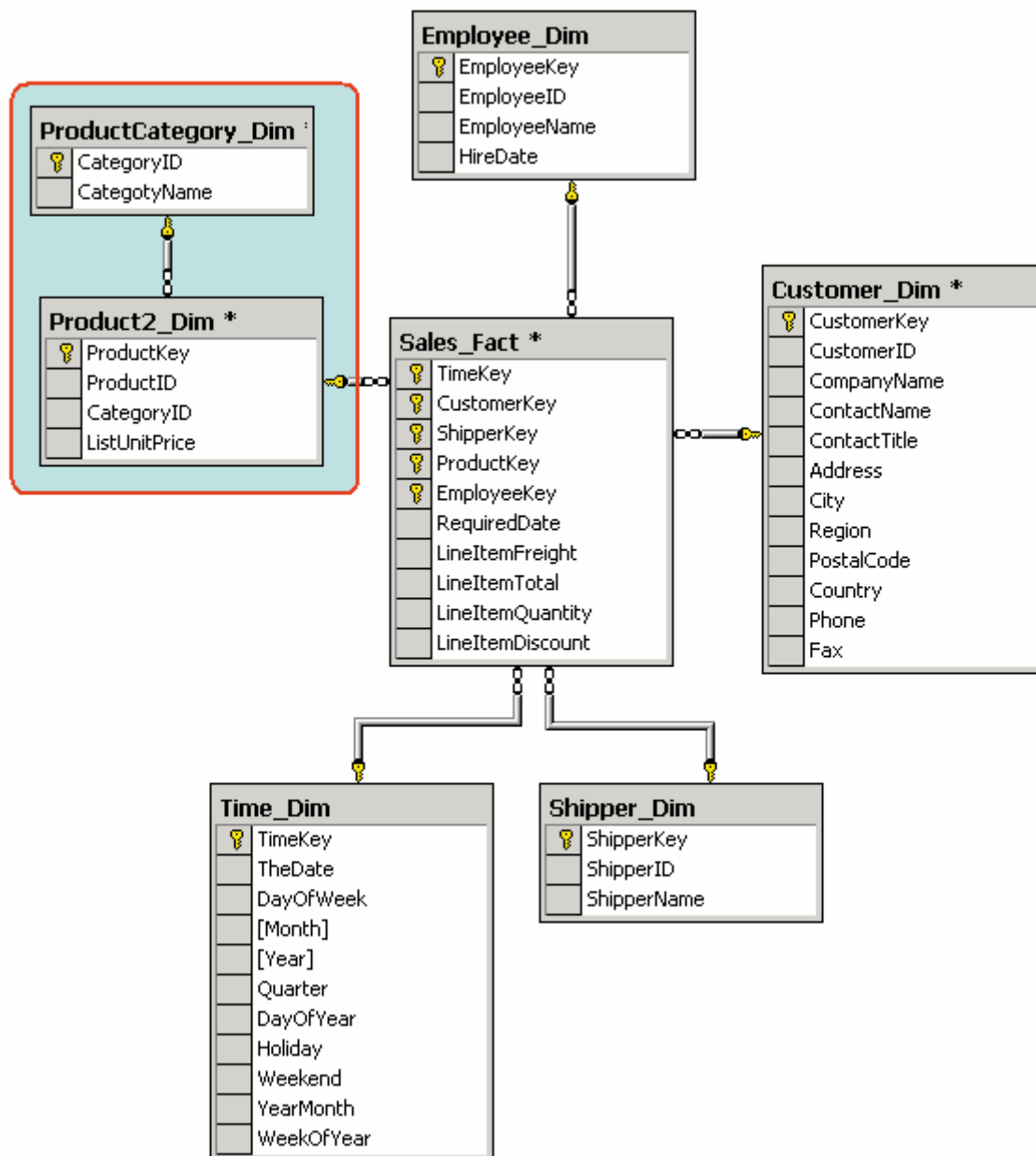


Рис. 5. Пример схемы «снежинка»

Отметим, что даже при наличии иерархических измерений с целью повышения скорости выполнения запросов к хранилищу данных нередко предпочтение отдается схеме «звезда».

Однако не все хранилища данных проектируются по двум приведенным выше схемам. Так, довольно часто вместо ключевого поля для измерения, содержащего данные типа «дата», и соответствующей таблицы измерений сама таблица фактов может содержать ключевое поле типа «дата». В этом случае соответствующая таблица измерений просто отсутствует.

В случае несбалансированной иерархии (например, такой, которая может быть основана на таблице Employees базы данных Northwind, имеющей поле EmployeeID, которое одновременно является и первичным, и внешним ключом и отражает подчиненность одних сотрудников другим (см. рис. 1) в схему «снежинка» также следует вносить коррективы. В этом

случае обычно в таблице измерений присутствует связь, аналогичная соответствующей связи в оперативной базе данных.

Еще один пример отступления от правил — наличие нескольких разных иерархий для одного и того же измерения. Типичные примеры таких иерархий — иерархии для календарного и финансового года (при условии, что финансовый год начинается не с 1 января), или с различными способами группировки членов измерения (например, группировать товары можно по категориям, а можно и по компаниям-поставщикам). В этом случае таблица измерений содержит поля для всех возможных иерархий с одними и теми же членами нижнего уровня, но с разными членами верхних уровней (пример такой таблицы приведен на рис. 3).

Как мы уже отмечали выше, таблица измерений может содержать поля, не имеющие отношения к иерархиям и представляющие собой просто дополнительные атрибуты членов измерений (*member properties*). Иногда такие атрибуты могут быть использованы при анализе данных.

Следует сказать, что для создания реляционных хранилищ данных нередко применяются специализированные СУБД, хранение данных в которых оптимизировано с точки зрения скорости выполнения запросов. Примером такого продукта является Sybase Adaptive Server IQ, реализующий нетрадиционный способ хранения данных в таблицах (не по строкам, а по столбцам). Однако создавать хранилища можно и в обычных реляционных СУБД.

Итак, обсудив типичную структуру хранилища данных, на основе которых обычно строятся OLAP-кубы, вернемся к созданию OLAP-кубов и поговорим о том, какими бывают OLAP-инструменты.

## **OLAP на клиенте и на сервере**

Многомерный анализ данных может быть произведен с помощью различных средств, которые условно можно разделить на клиентские и серверные OLAP-средства.

Клиентские OLAP-средства представляют собой приложения, осуществляющие вычисление агрегатных данных (сумм, средних величин, максимальных или минимальных значений) и их отображение, при этом сами агрегатные данные содержатся в кэше внутри адресного пространства такого OLAP-средства.

Если исходные данные содержатся в настольной СУБД, вычисление агрегатных данных производится самим OLAP-средством. Если же источник исходных данных — серверная СУБД, многие из клиентских OLAP-средств посылают на сервер SQL-запросы, содержащие оператор GROUP BY, и в результате получают агрегатные данные, вычисленные на сервере.

Как правило, OLAP-функциональность реализована в средствах статистической обработки данных (из продуктов этого класса на российском рынке широко распространены продукты компаний StatSoft и SPSS) и в некоторых электронных таблицах. В частности, неплохими средствами многомерного анализа обладает Microsoft Excel 2000. С помощью этого продукта можно создать и сохранить в виде файла небольшой локальный многомерный OLAP-куб и отобразить его двух- или трехмерные сечения.

Многие средства разработки содержат библиотеки классов или компонентов, позволяющие создавать приложения, реализующие простейшую OLAP-функциональность (такие, например, как компоненты DecisionCube в Borland Delphi и Borland C++Builder). Помимо этого многие компании предлагают элементы управления ActiveX и другие библиотеки, реализующие подобную функциональность.



Отметим, что клиентские OLAP-средства применяются, как правило, при малом числе измерений (обычно рекомендуется не более шести) и небольшом разнообразии значений этих параметров, — ведь полученные агрегатные данные должны уместиться в адресном пространстве подобного средства, а их количество растет экспоненциально при увеличении числа измерений. Поэтому даже самые примитивные клиентские OLAP-средства, как правило, позволяют произвести предварительный подсчет объема требуемой оперативной памяти для создания в ней многомерного куба.

Многие (но не все!) клиентские OLAP-средства позволяют сохранить содержимое кэша с агрегатными данными в виде файла, что, в свою очередь, позволяет не производить их повторное вычисление. Отметим, что нередко такая возможность используется для отчуждения агрегатных данных с целью передачи их другим организациям или для публикации. Типичным примером таких отчуждаемых агрегатных данных является статистика заболеваемости в разных регионах и в различных возрастных группах, которая является открытой информацией, публикуемой министерствами здравоохранения различных стран и Всемирной организацией здравоохранения. При этом собственно исходные данные, представляющие собой сведения о конкретных случаях заболеваний, являются конфиденциальными данными медицинских учреждений, которые ни в коем случае не должны попадать в руки страховых компаний и тем более становиться достоянием гласности.

Идея сохранения кэша с агрегатными данными в файле получила свое дальнейшее развитие в серверных OLAP-средствах, в которых сохранение и изменение агрегатных данных, а также поддержка содержащего их хранилища осуществляются отдельным приложением или процессом, называемым OLAP-сервером. Клиентские приложения могут запрашивать подобное многомерное хранилище и в ответ получать те или иные данные. Некоторые клиентские приложения могут также создавать такие хранилища или обновлять их в соответствии с изменившимися исходными данными.

Преимущества применения серверных OLAP-средств по сравнению с клиентскими OLAP-средствами сходны с преимуществами применения серверных СУБД по сравнению с настольными: в случае применения серверных средств вычисление и хранение агрегатных данных происходят на сервере, а клиентское приложение получает лишь результаты запросов к ним, что позволяет в общем случае снизить сетевой трафик, время выполнения запросов и требования к ресурсам, потребляемым клиентским приложением. Отметим, что средства анализа и обработки данных масштаба предприятия, как правило, базируются именно на серверных OLAP-средствах, например, таких как Oracle Express Server, Microsoft SQL Server 2000 Analysis Services, Hyperion Essbase, продуктах компаний Crystal Decisions, BusinessObjects, Cognos, SAS Institute. Поскольку все ведущие производители серверных СУБД производят (либо лицензировали у других компаний) те или иные серверные OLAP-средства, выбор их достаточно широк и почти во всех случаях можно приобрести OLAP-сервер того же производителя, что и у самого сервера баз данных.

Отметим, что многие клиентские OLAP-средства (в частности, Microsoft Excel 2000, Seagate Analysis и др.) позволяют обращаться к серверным OLAP-хранилищам, выступая в этом случае в роли клиентских приложений, выполняющих подобные запросы. Помимо этого имеется немало продуктов, представляющих собой клиентские приложения к OLAP-средствам различных производителей.

OLAP-серверы могут хранить многомерные данные разными способами, которые мы и обсудим в следующем разделе.

## Технические аспекты многомерного хранения данных

В многомерных хранилищах данных содержатся агрегатные данные различной степени подробности, например, объемы продаж по дням, месяцам, годам, по категориям товаров и т.п. Цель хранения агрегатных данных — сократить время выполнения запросов, поскольку в большинстве случаев для анализа и прогнозов интересны не детальные, а суммарные данные. Поэтому при создании многомерной базы данных всегда вычисляются и сохраняются некоторые агрегатные данные.

Отметим, что сохранение всех агрегатных данных не всегда оправданно. Дело в том, что при добавлении новых измерений объем данных, составляющих куб, растет экспоненциально (иногда говорят о «взрывном росте» объема данных). Если говорить более точно, степень роста объема агрегатных данных зависит от количества измерений куба и членов измерений на различных уровнях иерархий этих измерений. Для решения проблемы «взрывного роста» применяются разнообразные схемы, позволяющие при вычислении далеко не всех возможных агрегатных данных достичь приемлемой скорости выполнения запросов.

Как исходные, так и агрегатные данные могут храниться либо в реляционных, либо в многомерных структурах. Поэтому в настоящее время применяются три способа хранения данных:

- MOLAP (Multidimensional OLAP) — исходные и агрегатные данные хранятся в многомерной базе данных. Хранение данных в многомерных структурах позволяет манипулировать данными как многомерным массивом, благодаря чему скорость вычисления агрегатных значений одинакова для любого из измерений. Однако в этом случае многомерная база данных оказывается избыточной, так как многомерные данные полностью содержат исходные реляционные данные.
- ROLAP (Relational OLAP) — исходные данные остаются в той же реляционной базе данных, где они изначально и находились. Агрегатные же данные помещают в специально созданные для их хранения служебные таблицы в той же базе данных.
- HOLAP (Hybrid OLAP) — исходные данные остаются в той же реляционной базе данных, где они изначально находились, а агрегатные данные хранятся в многомерной базе данных.

Некоторые OLAP-средства поддерживают хранение данных только в реляционных структурах, некоторые — только в многомерных. Однако большинство современных серверных OLAP-средств поддерживают все три способа хранения данных. Выбор способа хранения зависит от объема и структуры исходных данных, требований к скорости выполнения запросов и частоты обновления OLAP-кубов.

Отметим также, что подавляющее большинство современных OLAP-средств не хранит «пустых» значений (примером «пустого» значения может быть отсутствие продаж сезонного товара вне сезона).

## Часть 3. Архитектура Microsoft Analysis Services

Что представляют собой аналитические службы

Что хранится в многомерной базе данных

Технологии доступа к аналитическим службам из клиентских приложений

SQL DSO

PivotTable Service, OLE DB for OLAP и ADO MD

Клиенты аналитических служб

Analysis Manager

Приложения Microsoft Office

### Службы преобразования данных

Источником данных для кубов OLAP, как правило, является реляционное хранилище данных. Сами по себе аналитические службы не содержат средств для пополнения реляционного хранилища данными из оперативных баз данных, с которыми работают пользователи. Однако такие средства содержат многие современные серверные СУБД.

В частности, в Microsoft SQL Server средства для переноса данных из одной реляционной СУБД в другую с возможным их преобразованием, которые можно применять и для пополнения хранилищ данных, называются службами преобразования данных (Data Transformation Services, DTS). Службы преобразования данных могут быть использованы не только с Microsoft SQL Server, но и с любыми другими источниками данных, доступными через универсальный механизм доступа к данным OLE DB (более подробное описание OLE DB и ADO вы найдете в книге «Базы данных для всех», которая будет выпущена издательством «КомпьютерПресс» этим летом).

Отметим, что DTS позволяют использовать дополнительные модули расширения (plugins) и в состав аналитических служб входит одно из таких расширений, позволяющее обновлять OLAP-кубы.

### Репозиторий аналитических служб

При создании описаний OLAP-кубов с помощью библиотек SQL DSO эти описания, или метаданные, сохраняются в репозитории. Сами же данные сохраняются в каталоге, указанном при установке Analysis Services (впоследствии его местоположение можно изменить).

По умолчанию репозиторий аналитических служб представляет собой базу данных Access msmdrep.mdb, расположенную в каталоге Microsoft Analysis Services\Bin, который при необходимости можно перенести и в базу данных Microsoft SQL Server. Текущая версия аналитических служб не поддерживает сохранение репозитория в базах данных других СУБД, в то время как само исходное реляционное хранилище данных может содержаться в любой СУБД, доступной с помощью универсальных механизмов доступа к данным OLE DB и ODBC.

Создание приложений для чтения и записи в репозиторий с помощью средств, отличных от библиотек SQL DSO, не рекомендуется, так как структура репозитория не документирована и может быть изменена в последующей версии аналитических служб.

В предыдущих статьях данного цикла мы рассказали об основах OLAP (On-Line Analytical Processing) — технологии многомерного анализа данных, а также рассмотрели типичную структуру хранилищ данных и некоторые технические аспекты многомерного

хранения данных. Настоящая статья посвящена типичной архитектуре OLAP-служб, рассматриваемой на примере Microsoft Analysis Services — OLAP-сервера фирмы Microsoft, входящего в комплект поставки Microsoft SQL Server 2000 Enterprise Edition и на сегодняшний день признанного аналитиками Gartner Group одним из наиболее популярных продуктов этого класса.

## **Что представляют собой аналитические службы**

Как мы уже знаем, конечной целью использования хранилищ данных и OLAP являются анализ данных и представление результатов этого анализа в удобном для восприятия и принятия решений виде. Непосредственное обращение клиентского приложения, отвечающего за представление результатов анализа данных, к хранилищу данных в принципе возможно. Однако в этом случае в нем должны быть реализованы средства такого анализа, то есть по существу оно должно быть клиентским OLAP-средством. При всей простоте такого подхода к реализации OLAP он не лишен недостатков, связанных с ограничениями, налагаемыми на число измерений и количество членов в них (подробное рассмотрение этого вопроса можно найти в предыдущей статье данного цикла). Как мы знаем, у серверных OLAP-средств таких недостатков нет. Поэтому более прогрессивным представляется подход, основанный на применении серверных OLAP-средств в качестве промежуточного звена между хранилищем данных в виде реляционной СУБД и клиентским приложением. В этом случае OLAP-сервер должен превращать данные из реляционного хранилища в форму, более удобную для создания аналитических отчетов, — в OLAP-кубы.

Как уже было сказано выше, в качестве примера серверного OLAP-средства мы рассмотрим аналитические службы Microsoft (Microsoft Analysis Services), входящие в состав Microsoft SQL Server 2000 Enterprise Edition.

Основным компонентом аналитических служб является Analysis Server — сервис операционной системы Windows NT/2000. Этот сервер предназначен для создания OLAP-кубов на основе реляционных хранилищ данных, а также для предоставления доступа к ним из клиентских приложений. Ниже мы рассмотрим, какими именно объектами манипулирует этот сервер и с помощью каких механизмов это происходит.

## **Что хранится в многомерной базе данных**

Теоретически OLAP-куб, созданный с помощью аналитических служб Microsoft, может содержать все данные из таблицы фактов плюс агрегатные значения для тех групп записей из этой таблицы, которые соответствуют верхним уровням иерархии измерений. При необходимости можно производить динамическое обновление куба, если в таблицу фактов были добавлены новые записи, а также выбрать, будут ли данные с нижних уровней иерархии храниться в самом кубе, что соответствует способу хранения данных Multidimensional OLAP, или они будут считываться из таблицы фактов хранилища данных, что соответствует способам хранения данных Relational OLAP и Hybrid OLAP (способы хранения данных мы рассматривали в предыдущей статье данного цикла). С точки зрения пользователя различий между этими способами хранения нет, не считая разницы в производительности обращающихся к этим кубам приложений.

Аналитические службы сохраняют агрегатные данные только для простейших агрегатных функций (сумм, числа записей, максимальных и минимальных значений). Однако в случае необходимости можно создавать так называемые вычисляемые члены (calculated members) для получения других типов агрегатных значений (средних, средневзвешенных, смещенных и несмещенных дисперсий и т.д.). При этом, помимо применения встроенных средств создания агрегатных данных, Analysis Services позволяет использовать для вычисления агрегатных данных функции VBA или Excel, а также создавать собственные.

Так, для создания нескольких кубов, имеющих одинаковые измерения, можно сгруппировать их в одну многомерную базу данных, а сами эти измерения поместить в библиотеку (library), сделав их коллективными, то есть общедоступными для всех кубов, содержащихся в базе данных (shared dimensions). Можно также создавать измерения, принадлежащие только одному кубу (private dimensions).

И наконец, аналитические службы Microsoft позволяют создавать так называемые виртуальные кубы (virtual cubes), которые в определенной степени являются аналогами представлений (view) реляционных СУБД. Виртуальные кубы не содержат данных, но позволяют представить в виде единого куба данные из нескольких кубов, имеющих хотя бы одно общее коллективное измерение.

## Технологии доступа к аналитическим службам из клиентских приложений

Пользователей аналитических служб можно условно разделить на две группы: администраторов, создающих или модифицирующих OLAP-кубы, и обычных пользователей-аналитиков, читающих данные из OLAP-кубов с целью создания аналитических отчетов. Эти группы используют разные технологии доступа к OLAP-данным. Ниже мы выясним, какие технологии предназначены для этих двух категорий пользователей.

### SQL DSO

Decision Support Objects (DSO) — это набор библиотек, содержащих COM-объекты, позволяющие создавать и модифицировать многомерные базы данных и содержащиеся в них объекты (кубы, коллективные измерения и т.д.). Отметим, что Analysis Manager — приложение, использующее SQL DSO, — входит в состав аналитических служб.

Эти библиотеки можно использовать для разработки собственных приложений, в которых осуществляется создание или модификация многомерных баз данных, в том числе и для реализации действий, не предусмотренных в клиентских утилитах, входящих в состав аналитических служб (рис. 1).

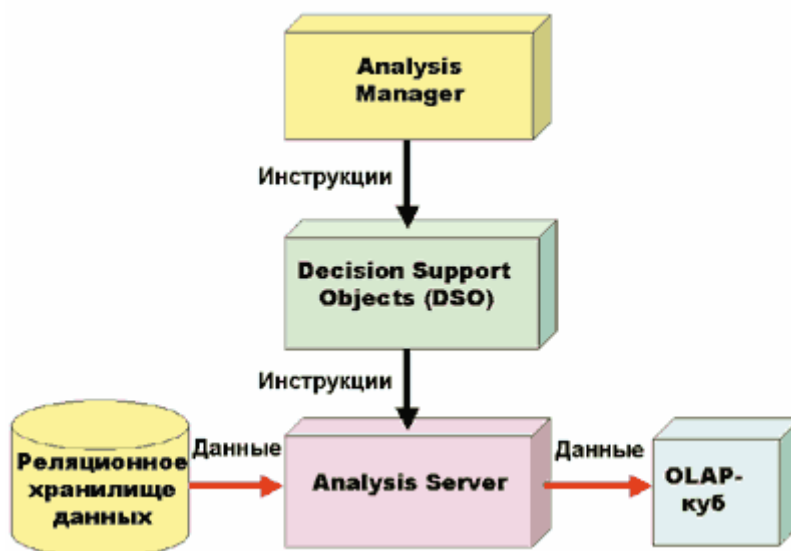


Рис. 1. Приложение, использующее SQL DSO

Отметим, что SQL DSO можно использовать только для доступа к аналитическим службам Microsoft. Ни к каким другим OLAP-серверам с помощью этих библиотек обратиться нельзя.

Более подробно о применении SQL DSO мы расскажем в отдельной статье.

## **PivotTable Service, OLE DB for OLAP и ADO MD**

Приложения, предназначенные для чтения OLAP-данных, при взаимодействии с аналитическими службами обязательно используют PivotTable Service — библиотеки, загружаемые в адресное пространство клиентского приложения. Эти библиотеки автоматически устанавливаются вместе с аналитическими службами (независимо от того, какая именно их часть установлена — клиентская или серверная), а также вместе с Microsoft Office 2000. В состав Microsoft SQL Server 2000 входит также инсталляционное приложение для установки PivotTable Service на компьютер, на котором не установлены ни аналитические службы, ни Microsoft Office.

PivotTable Service можно использовать в любой 32-разрядной версии Windows для просмотра серверных OLAP-кубов, а также для создания, модификации и чтения локальных OLAP-кубов, созданных в клиентском приложении, реализуя таким образом клиентскую OLAP-функциональность. Эти библиотеки реализуют кэширование в клиентском приложении данных, полученных как с OLAP-сервера, так и из реляционных источников данных. Помимо этого они позволяют осуществлять кэширование данных и на OLAP-сервере, повышая тем самым производительность работы с ним в случае обращения к одним и тем же данным нескольких пользователей.

Для взаимодействия с PivotTable Service клиентское приложение может использовать OLE DB for OLAP — расширение универсального механизма доступа к данным OLE DB, позволяющее обращаться к многомерным данным, а также ADO MD — библиотеки, представляющие собой надстройку над OLE DB for OLAP и являющиеся COM-серверами для доступа к многомерным данным, удобными для применения в клиентских приложениях.

Отметим, что спецификация OLE DB for OLAP является открытой. Это означает, что можно создавать и другие OLAP-серверы, поддерживающие OLE DB for OLAP (либо разрабатывать OLE DB-провайдеры к уже имеющимся OLAP-средствам), а также создавать клиентские приложения, обращающиеся к любым таким источникам данных с помощью PivotTable Service, OLE DB for OLAP и ADO MD.

## **Клиенты аналитических служб**

Описанные выше технологии доступа к многомерным данным можно применять в собственных приложениях. Однако для наиболее часто встречающихся задач, таких как создание и просмотр кубов, в состав аналитических служб входят клиентские утилиты, которые мы рассмотрим ниже.

## **Analysis Manager**

Analysis Manager представляет собой утилиту, входящую в состав аналитических служб и предназначенную главным образом для администраторов баз данных OLAP (рис. 2).

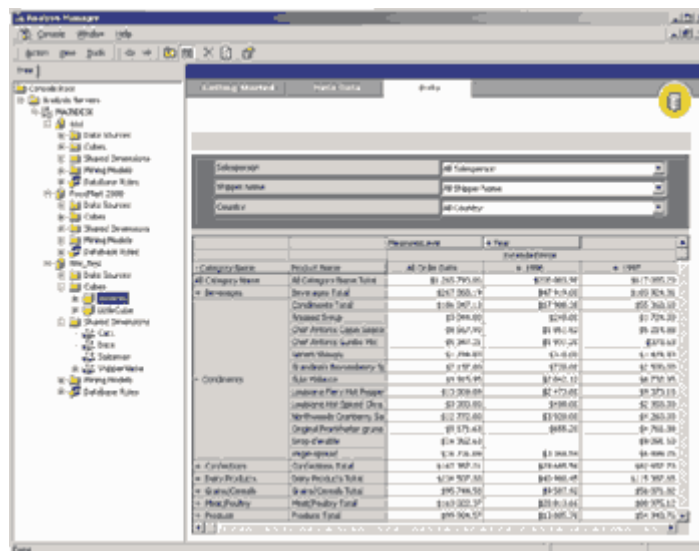


Рис. 2. Analysis Manager

В составе Analysis Manager имеется простейшее средство просмотра многомерных данных, представляющее собой элемент управления ActiveX, использующий для доступа к данным OLE DB for OLAP.

Analysis Manager использует библиотеки SQL DSO для создания и модификации объектов многомерной базы данных и OLE DB для доступа к исходным реляционным хранилищам данных. Что касается доступа к самим многомерным данным, то, повторимся, Analysis Manager использует для этой цели OLE DB for OLAP.

## Приложения Microsoft Office

Из других клиентских приложений, не входящих в состав аналитических служб, но часто используемых для просмотра OLAP-кубов, следует назвать приложения Microsoft Office, в частности Microsoft Excel. С помощью Excel можно обращаться к серверным OLAP-кубам, получая их двух- и трехмерные сечения на листах рабочих книг Excel в виде сводных таблиц, а также создавать локальные OLAP-кубы в виде файлов на основе реляционных данных, доступных с помощью OLE DB (рис. 3).

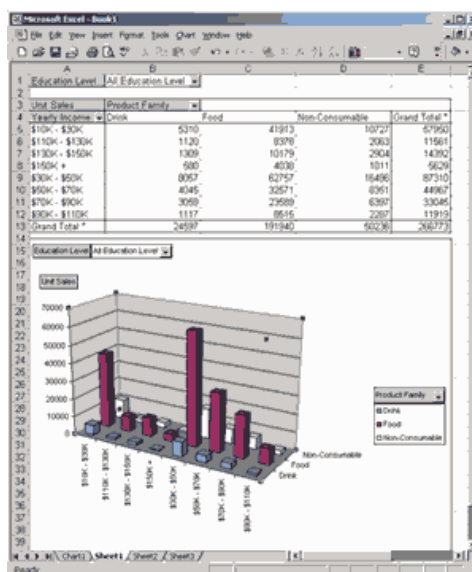


Рис. 3. Применение Excel в качестве OLAP-клиента

Кроме того, в состав Microsoft Office Web Components входит элемент управления ActiveX PivotTable List, позволяющий реализовать сходную функциональность как в обычном Windows-приложении, так и на HTML-странице, предназначенной для применения внутри корпоративной сети (рис. 4).

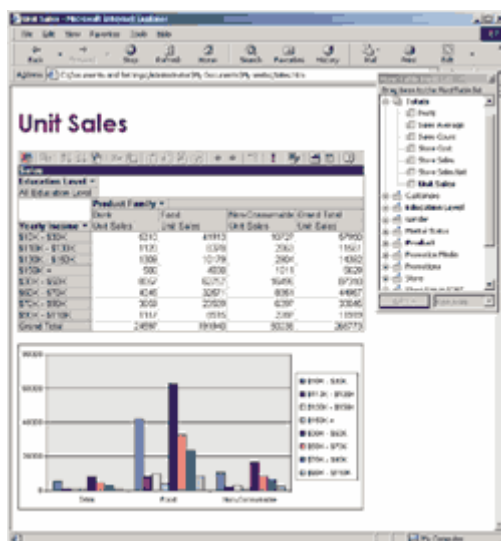


Рис. 4. Применение элемента управления PivotTable List в качестве OLAP-клиента

Подробнее о применении Microsoft Office в качестве клиентских приложений для аналитических служб мы расскажем в одной из следующих статей данного цикла.

Отметим, что помимо Microsoft Office существуют и другие коммерческие продукты, предназначенные для обращения к OLAP-данным и создания OLAP-кубов. Кроме того, как мы уже говорили, можно создавать свои собственные приложения, использующие PivotTable Service, OLE DB for OLAP и ADO MD (рис. 5).

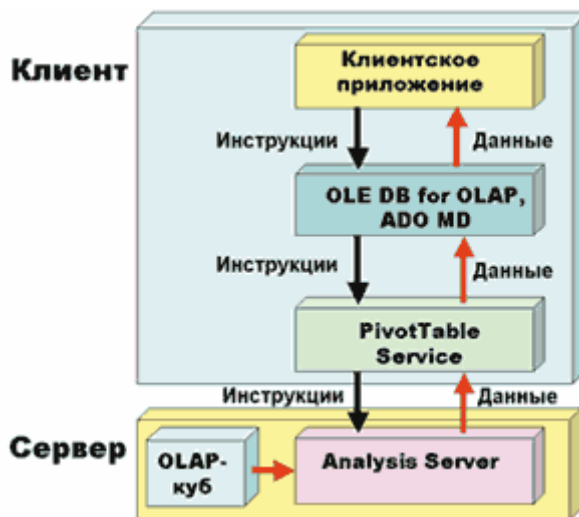


Рис. 5. Приложение, использующее PivotTable Service и OLE DB for OLAP



## **Часть 4. Создание и заполнение хранилищ данных с помощью Data Transformation Services**

Создание хранилищ данных

Заполнение хранилища данных с помощью Data Transformation Services

Что представляет собой DTS

Описание источников данных

Описание потоков данных и последовательности выполнения задач

Описание преобразования данных

Выполнение пакетов DTS

В предыдущей статье данного цикла мы рассмотрели архитектуру OLAP-служб на примере Microsoft Analysis Services — OLAP-сервера фирмы Microsoft, входящего в комплект поставки Microsoft SQL Server 2000 Enterprise Edition и на сегодняшний день признанного аналитиками Gartner Group одним из наиболее популярных продуктов этого класса. Среди ближайших тем, которые мы планируем рассмотреть в данном цикле, будет создание многомерных баз данных и OLAP-кубов с помощью Microsoft Analysis Services.

Прежде чем обсуждать создание OLAP-кубов, вспомним, что, как правило, исходные данные для их создания хранятся не в оперативной базе данных, с которой работают пользователи, а в специализированном хранилище данных. Настоящая статья будет посвящена вопросам заполнения хранилищ данных и синхронизации их с содержимым оперативной базы данных.

### **Создание хранилищ данных**

Как мы уже знаем, типичная структура хранилища данных существенно отличается от структуры традиционной реляционной СУБД. Как правило, эта структура денормализована с целью повышения скорости выполнения запросов, поэтому она может допускать избыточность данных. Типичное хранилище данных содержит таблицу фактов со сведениями об объектах или событиях, совокупность которых будет в дальнейшем анализироваться, и несколько таблиц измерений, содержащих неизменяемые либо редко изменяемые данные

В качестве оперативной базы данных для нашего примера мы будем использовать базу данных Northwind из комплекта поставки Microsoft SQL Server 2000. На ее основе мы построим хранилище данных, использующее схему «звезда». Структура данных этого хранилища приведена на рис. 1,

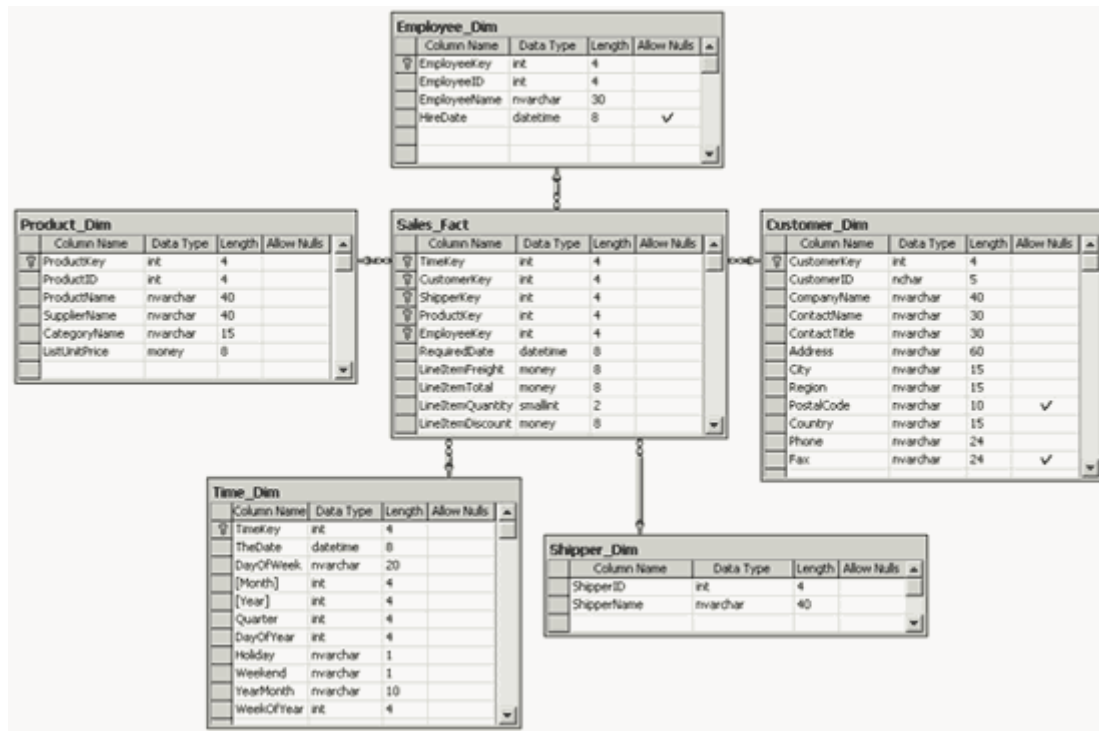


Рис. 1. Структура хранилища данных *Northwind\_Mart*, созданного на основе базы данных *Northwind*

а скрипт для создания базы данных с такой структурой (назовем ее *Northwind\_Mart*) — в листинге 1.

## Листинг 1

```
CREATE DATABASE Northwind_Mart ON PRIMARY
(
    NAME=Northwind_Mart_Data,
    FILENAME='d:\Program Files\Microsoft SQL
Server\mssql\data\Northwind_Mart_Data.MDF',
    SIZE=5MB,
    FILEGROWTH=10%
)
LOG ON
(
    NAME=Northwind_Mart_Log,
    FILENAME='d:\Program Files\Microsoft SQL
Server\mssql\data\Northwind_Mart_Log.LDF',
    SIZE=2MB,
    FILEGROWTH=10%
)
GO

USE Northwind_Mart
GO

CREATE TABLE [dbo].[Customer_Dim] (
    [CustomerKey] [int] IDENTITY (1, 1) NOT NULL ,
    [CustomerID] [nchar] (5) NOT NULL ,
    [CompanyName] [nvarchar] (40) NOT NULL ,
    [ContactName] [nvarchar] (30) NOT NULL ,
    [ContactTitle] [nvarchar] (30) NOT NULL ,
    [Address] [nvarchar] (60) NOT NULL ,
    [City] [nvarchar] (15) NOT NULL ,
    [Region] [nvarchar] (15) NOT NULL ,
    [PostalCode] [nvarchar] (10) NULL ,
```

```

        [Country] [nvarchar] (15) NOT NULL ,
        [Phone] [nvarchar] (24) NOT NULL ,
        [Fax] [nvarchar] (24) NULL
    ) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Employee_Dim] (
    [EmployeeKey] [int] IDENTITY (1, 1) NOT NULL ,
    [EmployeeID] [int] NOT NULL ,
    [EmployeeName] [nvarchar] (30) NOT NULL ,
    [HireDate] [datetime] NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Product_Dim] (
    [ProductKey] [int] IDENTITY (1, 1) NOT NULL ,
    [ProductID] [int] NOT NULL ,
    [ProductName] [nvarchar] (40) NOT NULL ,
    [SupplierName] [nvarchar] (40) NOT NULL ,
    [CategoryName] [nvarchar] (15) NOT NULL ,
    [ListUnitPrice] [money] NOT NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Sales_Fact] (
    [TimeKey] [int] NOT NULL ,
    [CustomerKey] [int] NOT NULL ,
    [ShipperKey] [int] NOT NULL ,
    [ProductKey] [int] NOT NULL ,
    [EmployeeKey] [int] NOT NULL ,
    [RequiredDate] [datetime] NOT NULL ,
    [LineItemFreight] [money] NOT NULL ,
    [LineItemTotal] [money] NOT NULL ,
    [LineItemQuantity] [smallint] NOT NULL ,
    [LineItemDiscount] [money] NOT NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Shipper_Dim] (
    [ShipperKey] [int] IDENTITY (1, 1) NOT NULL ,
    [ShipperID] [int] NOT NULL ,
    [ShipperName] [nvarchar] (40) NOT NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Time_Dim] (
    [TimeKey] [int] IDENTITY (1, 1) NOT NULL ,
    [TheDate] [datetime] NOT NULL ,
    [DayOfWeek] [nvarchar] (20) NOT NULL ,
    [Month] [int] NOT NULL ,
    [Year] [int] NOT NULL ,
    [Quarter] [int] NOT NULL ,
    [DayOfYear] [int] NOT NULL ,
    [Holiday] [nvarchar] (1) NOT NULL ,
    [Weekend] [nvarchar] (1) NOT NULL ,
    [YearMonth] [nvarchar] (10) NOT NULL ,
    [WeekOfYear] [int] NOT NULL
) ON [PRIMARY]
GO

```

```

ALTER TABLE [dbo].[Customer_Dim] WITH NOCHECK ADD
    CONSTRAINT [PK_Customer_Dim] PRIMARY KEY NONCLUSTERED
        ([CustomerKey]) ON [PRIMARY]
GO

```

```

ALTER TABLE [dbo].[Employee_Dim] WITH NOCHECK ADD

```

```

        CONSTRAINT [PK_Employee_Dim] PRIMARY KEY NONCLUSTERED
        ([EmployeeKey]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Product_Dim] WITH NOCHECK ADD
    CONSTRAINT [PK_Product_Dim] PRIMARY KEY NONCLUSTERED
    ([ProductKey]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Sales_Fact] WITH NOCHECK ADD
    CONSTRAINT [PK_Sales_Fact] PRIMARY KEY NONCLUSTERED
    (
        [TimeKey],
        [CustomerKey],
        [ShipperKey],
        [ProductKey],
        [EmployeeKey]
    ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Shipper_Dim] WITH NOCHECK ADD
    CONSTRAINT [PK_Shipper_Dim] PRIMARY KEY NONCLUSTERED
    ([ShipperKey]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Time_Dim] WITH NOCHECK ADD
    CONSTRAINT [PK_Time_Dim] PRIMARY KEY NONCLUSTERED
    ([TimeKey]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Sales_Fact] ADD
    CONSTRAINT [FK_Sales_Fact_Customer_Dim] FOREIGN KEY
    ([CustomerKey]) REFERENCES [dbo].[Customer_Dim] ([CustomerKey]),
    CONSTRAINT [FK_Sales_Fact_Employee_Dim] FOREIGN KEY
    ([EmployeeKey]) REFERENCES [dbo].[Employee_Dim] ([EmployeeKey]),
    CONSTRAINT [FK_Sales_Fact_Product_Dim] FOREIGN KEY
    ([ProductKey]) REFERENCES [dbo].[Product_Dim] ([ProductKey]),
    CONSTRAINT [FK_Sales_Fact_Shipper_Dim] FOREIGN KEY
    ([ShipperKey]) REFERENCES [dbo].[Shipper_Dim] ([ShipperKey]),
    CONSTRAINT [FK_Sales_Fact_Time_Dim] FOREIGN KEY
    ([TimeKey]) REFERENCES [dbo].[Time_Dim] ([TimeKey])
GO

```

Отметим, однако, что проектирование хранилища и создание базы данных соответствующей структуры — лишь первый шаг к созданию хранилища данных. Далее нам следует позаботиться о том, чтобы таблицы этого хранилища были заполнены данными, соответствующими текущему состоянию оперативной базы данных.

## Заполнение хранилища данных с помощью Data Transformation Services

### Что представляют собой DTS

Data Transformation Services (DTS) — это набор служб SQL Server, предназначенных для организации импорта, экспорта, преобразования данных и переноса их между любыми источниками, доступными через интерфейсы OLE DB. С их помощью можно копировать структуры данных и сами данные из одной базы данных в другую, создавать средства для переноса данных, встроенные в приложения, а также пополнять хранилища данных из разнообразных источников (которые в общем случае вовсе не обязательно должны быть базами данных SQL Server).

Для заполнения хранилища данных обычно требуется создать и выполнить так называемый пакет DTS (DTS package), содержащий описание последовательности всех действий, которые следует выполнить при переносе данных (включая преобразование типов данных, выполнение SQL-запросов и т.д.). Такой пакет можно выполнить с помощью SQL Server Enterprise Manager или утилиты dtsrun, сохранить его в службах метаданных (Meta Data Services; в прежних версиях SQL Server это хранилище называлось репозитарием) либо в виде структурированного файлового хранилища. Также возможно программное выполнение DTS-пакетов с помощью свойств и методов соответствующих объектов SQL DMO — для этого можно автоматически сгенерировать код на языке Visual Basic. В SQL Server 2000 также поддерживается возможность сохранения DTS-пакетов в формате XML.

Ниже мы рассмотрим процесс создания пакета DTS, заполняющего хранилище Northwind\_Mart данными из оперативной базы данных Northwind. На этом примере мы изучим разнообразные возможности сервисов преобразования данных, доступные с помощью DTS.

## **Описание источников данных**

Создать пакет DTS можно с помощью соответствующего редактора — DTS package editor. Для его запуска следует с помощью SQL Server Enterprise Manager соединиться с сервером, содержащим хранилище данных, найти в разделе Data Transformation Services элемент Meta Data Service Packages и выбрать опцию New Package из его контекстного меню.

Далее нам требуется описать базу данных, в которой находится наше хранилище. Для этого необходимо перенести на рабочее пространство редактора пакетов DTS пиктограмму Microsoft OLE DB Provider for SQL Server с палитры Data tool в левой части окна редактора. После этого появится диалоговая панель Connection Properties для описания источников данных OLE DB, в которой нужно выбрать базу данных Northwind\_Mart, указать параметры доступа к ней (например, Use Windows NT authentication). Присвоим этому источнику данных имя NW\_OLAP. Для наглядности создаваемой диаграммы сделаем копию этого же источника данных, перенеся на рабочее пространство редактора пакетов DTS еще одну такую же пиктограмму, отметив в диалоговой панели Connection Properties опцию Existing Connection и выбрав из списка имеющихся источников данных NW\_OLAP.

Тем же способом опишем источник исходных данных — базу данных Northwind, присвоим ему имя NW и создадим еще пять его копий, так как в нашем хранилище данных содержится шесть таблиц, и нам потребуется шесть отдельных операций по их заполнению.

## **Описание потоков данных и последовательности выполнения задач**

В нашем примере перед заполнением таблиц в хранилище данных мы будем полностью очищать их содержимое. Для этой цели мы перенесем в рабочее пространство редактора пиктограмму Execute SQL Task. При этом на экране появится диалоговая панель Execute SQL Task Properties, в которой мы заполним поля Description (описание задачи) и SQL Statement (сюда мы добавим операторы для удаления данных из всех таблиц хранилища данных, рис. 2).

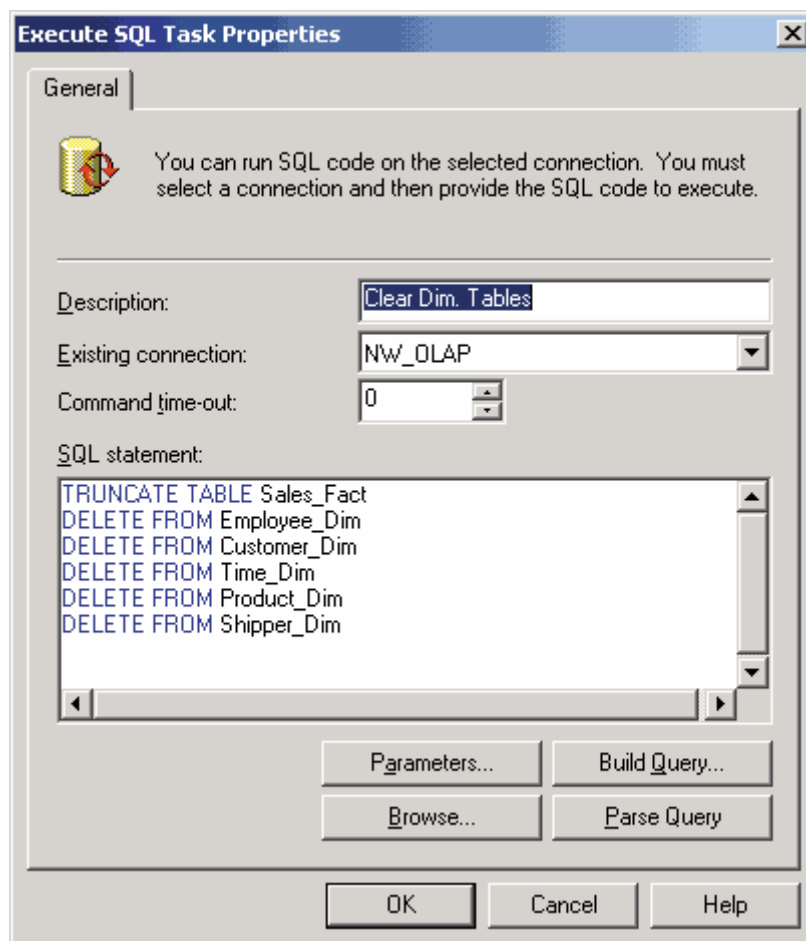


Рис. 2. Диалоговая панель *Execute SQL Task Properties*

Отметим, что при большом объеме данных удаление данных из хранилища обычно не применяется — в этом случае к уже существующим данным добавляются новые.

Далее нам следует определить, какие потоки данных нужны для заполнения хранилища. С этой целью с помощью щелчков мыши при нажатой клавише Ctrl выберем один из шести экземпляров источника данных NW и один из двух экземпляров источника данных NW\_OLAP. Когда обе пиктограммы будут выделены, следует выбрать опцию WorkFlow из контекстного меню источника данных NW\_OLAP, и тогда пиктограммы окажутся соединенными стрелкой, соответствующей одной из задач преобразования и переноса данных. Далее повторим эту же операцию с четырьмя другими экземплярами источника данных NW и с тем же самым экземпляром источника данных NW\_OLAP.

Таким образом, мы создали задания для переноса данных в пять таблиц измерений нашего хранилища. Эти задачи могут выполняться параллельно, ведь таблицы измерений в нашем хранилище не связаны друг с другом. Однако они могут быть выполнены только после полной очистки всего хранилища. Чтобы описать это условие (такие условия определяются словосочетанием *precedence constraint*), нам следует одновременно выбрать пиктограмму Execute SQL Task и одну из пяти уже задействованных пиктограмм источника данных NW, а затем из контекстного меню источника данных NW выбрать опцию Workflow | On Success. Появившаяся зеленая пунктирная стрелка между пиктограммами означает, что перенос данных в соответствующую таблицу изменений будет осуществлен только после успешного завершения очистки хранилища. Далее следует повторить это действие с оставшимися четырьмя используемыми экземплярами источника данных NW.

Что же касается задачи заполнения данными таблицы фактов, она может быть выполнена только после того, как будут заполнены все таблицы измерений. Поэтому сначала

мы выделим оставшиеся экземпляры источника данных NW и источника данных NW\_OLAP, затем выберем опцию Workflow из контекстного меню этого экземпляра источника данных NW\_OLAP — при этом пиктограммы окажутся соединены стрелкой, соответствующей задаче заполнения таблицы фактов. Далее нам следует одновременно выбрать пиктограмму NW\_OLAP, участвующую в описании пяти задач заполнения таблиц измерений, и пиктограмму NW, участвующую в описании задачи заполнения таблицы фактов, а затем из контекстного меню выделенного источника данных NW выбрать опцию Workflow | On Success. Таким образом, мы указали, что заполнение таблицы фактов осуществляется только после успешного заполнения таблиц измерений (рис. 3).

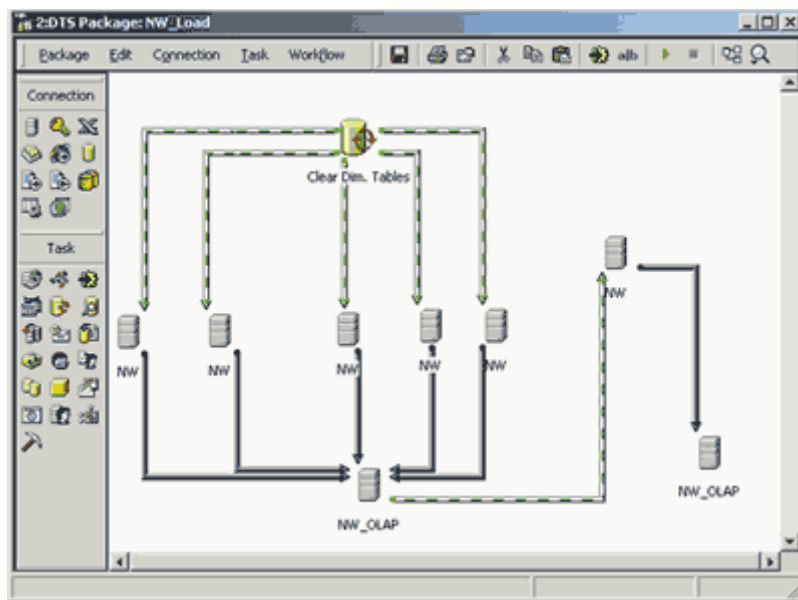


Рис. 3. Описание последовательности выполнения задач заполнения хранилища данных

## Описание преобразования данных

Далее нам следует описать, откуда берутся и как преобразовываются данные при переносе из оперативной базы данных в хранилище. Мы начнем с таблицы Time\_Dim. Для этой цели дважды щелкнем мышью по одной из пяти стрелок, соответствующих задачам заполнения таблиц измерений. В появившейся диалоговой панели заполним поле Description, выберем опцию SQL Query и введем текст SQL-запроса, результат которого должен быть помещен в таблицу Time\_Dim:

```
SELECT DISTINCT
  S.ShippedDate AS TheDate,
  DateName(dw, S.ShippedDate) AS DayOfWeek,
  DatePart(mm, S.ShippedDate) AS [Month],
  DatePart(yy, S.ShippedDate) AS [Year],
  DatePart(qq, S.ShippedDate) AS [Quarter],
  DatePart(dy, S.ShippedDate) AS DayOfYear,
  'N' AS Holiday,
  case DatePart(dw, S.ShippedDate)
  when (1) then 'Y'
  when (7) then 'Y'
  else 'N'
  end
AS Weekend,
  DateName(month, S.ShippedDate) +
  '_' + DateName(year, S.ShippedDate) AS YearMonth,
  DatePart(wk, S.ShippedDate) AS WeekOfYear
FROM Orders S
WHERE S.ShippedDate IS NOT NULL
```

Щелкнем по закладке Destination и выберем из списка таблиц хранилища данных таблицу Time\_Dim. Далее можно перейти на страницу Transformations и проверить правильность соответствий между полями исходного набора данных и таблицы Time\_Dim. Если они не соответствуют желаемым, их можно отредактировать с помощью кнопок New, Edit, Delete (рис. 4).

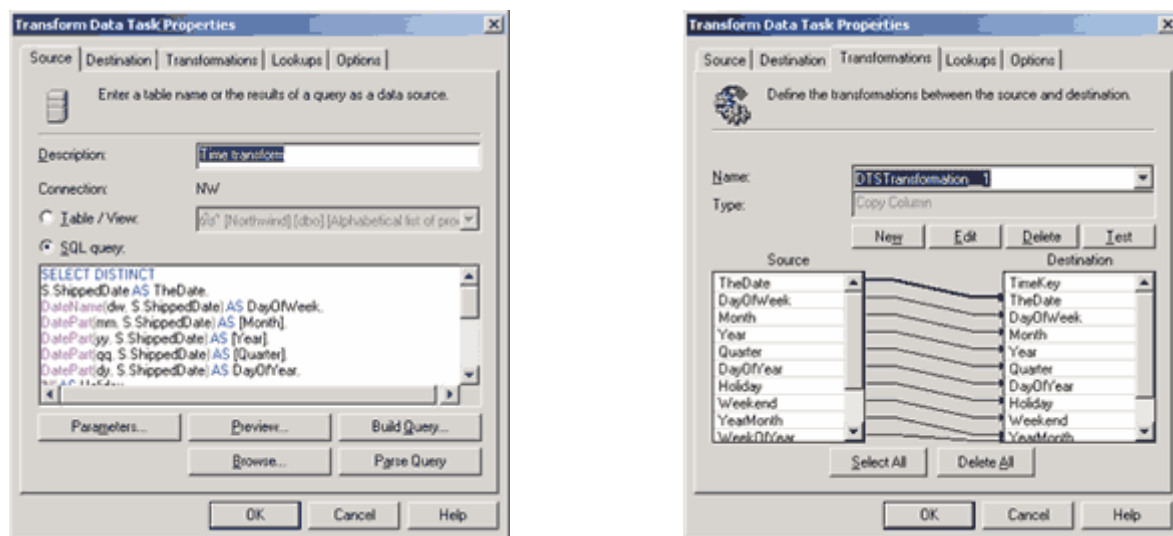


Рис. 4. Описание преобразования данных для таблицы Time\_Dim

Следующая таблица измерений, Customer\_Dim, будет заполняться не результатами запроса, а данными из таблицы Customers. Поэтому на странице Source следует отметить опцию Table/View, выбрать таблицу Customers в списке таблиц базы данных Northwind, на странице Destination выбрать таблицу Customer\_Dim и проверить правильность соответствий между полями исходного набора данных и таблицы Customer\_Dim. Однако в этом случае было бы желательно преобразовать некоторые значения, содержащиеся в поле Region исходной таблицы (для одних стран это поле не содержит данных, тогда как для других может потребоваться анализ продаж по регионам или другим административным единицам). С этой целью мы удалим соответствие между полем Region обеих таблиц, нажмем кнопку New, из списка в диалоговом окне New Transformation выберем опцию ActiveX Script и в появившейся диалоговой панели ActiveX Script Transformation Properties отредактируем код на языке VBScript, описывающий преобразование данных в этом поле:

```
Function Main()  
If IsNull(DTSSource("Region")) Then  
    DTSDestination("Region") = "Other"  
Else  
    DTSDestination("Region") = DTSSource("Region")  
End If  
Main = DTSTransformStat_OK  
End Function
```

Здесь курсивом выделены фрагменты, добавленные к коду, сгенерированному по умолчанию (рис. 5).



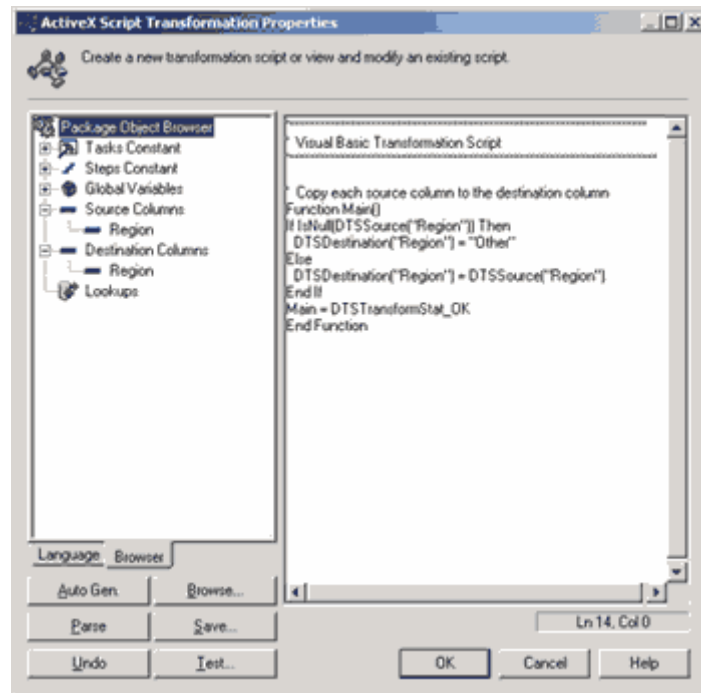


Рис. 5. Описание преобразования данных для таблицы Customer\_Dim с помощью скрипта

Для таблицы измерений Product\_Dim последовательность действий сходна с той, что мы применяли при создании таблицы Time\_Dim. Однако здесь, выбрав на странице Source диалоговой панели Transform Data Task Properties опцию SQL Query, мы нажмем кнопку Build Query и создадим запрос с помощью DTS Query Designer (рис. 6).

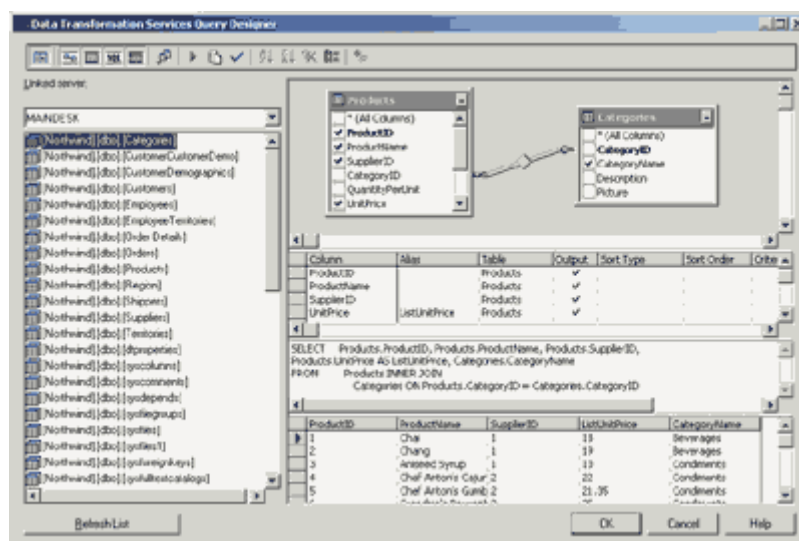


Рис. 6. Создание запроса к оперативной базе данных с помощью DTS Query Designer

В запросе используются таблицы Products и Categories базы данных Northwind, при этом поле UnitPrice таблицы Products переименовывается в ListUnitPrice.

Выбрав на странице Destination таблицу Product\_Dim, проверим корректность соответствий между полями исходного набора данных и таблицы Product\_Dim. В данном случае мы видим, что поля SupplierID и SupplierName – разных типов, при этом то и другое поле описывают, по существу, одно и то же свойство члена измерения. В этой ситуации нам поможет подстановка значений (lookup). Перейдем на страницу Lookups, нажмем кнопку Add, придумаем имя для подстановки, например SupplierLookup, щелкнем по кнопке Query и в появившемся редакторе DTS Query Designer введем текст запроса:

```

SELECT CompanyName
FROM Suppliers
WHERE (SupplierID = ?)

```

Далее на странице Transformations опишем соответствие между полями SupplierId и SupplierName. С этой целью нажмем кнопку New, из списка в диалоговом окне New Transformation выберем опцию ActiveX Script, укажем имена исходного и получаемого полей и в появившейся диалоговой панели ActiveX Script Transformation Properties отредактируем код на языке VBScript, описывающий преобразование данных в этом поле:

```

Function Main()
DTSDestination("SupplierName") = _
DTSLookups("SupplierLookup").Execute(DTSSource("SupplierID").Value)
Main = DTSTransformStat_OK
End Function

```

Для заполнения данными следующей таблицы измерений, Employee\_Dim, нам нужно указать, что два поля исходной таблицы Customers, FirstName и LastName, соответствуют одному полю EmployeeName таблицы Customer\_Dim. Для этого нажмем кнопку New на странице Transformations, выберем опцию ActiveX Script и отметим оба поля, FirstName и LastName, в качестве исходных. Далее модифицируем код в диалоговой панели панели ActiveX Script Transformation Properties:

```

Function Main()
DTSDestination("EmployeeName") = DTSSource("FirstName") & _
" " & DTSSource("LastName")
Main = DTSTransformStat_OK
End Function

```

И наконец, при описании преобразования данных для таблицы Shipper\_Dim нам нужно проверить соответствие между полем CompanyName таблицы Shippers базы данных Northwind и полем ShipperName таблицы Shipper\_Dim.

Завершив работу с таблицами измерений, займемся преобразованием данных для таблицы фактов. В данном случае исходный набор данных, преобразуемый в таблицу Sales\_Fact, представляет собой результат следующего запроса:

```

SELECT
    Northwind_Mart.dbo.Time_Dim.TimeKey,
    Northwind_Mart.dbo.Customer_Dim.CustomerKey,
    Northwind_Mart.dbo.Shipper_Dim.ShipperKey,
    Northwind_Mart.dbo.Product_Dim.ProductKey,
    Northwind_Mart.dbo.Employee_Dim.EmployeeKey,
    Northwind.dbo.Orders.RequiredDate,
    Orders.Freight * [Order Details].Quantity /
        (SELECT SUM(Quantity)
         FROM [Order Details] od
         WHERE od.OrderID = Orders.OrderID) AS LineItemFreight,
    [Order Details].UnitPrice * [Order Details].Quantity
    AS LineItemTotal,
    [Order Details].Quantity AS LineItemQuantity,
    [Order Details].Discount * [Order Details].UnitPrice *
    [Order Details].Quantity AS LineItemDiscount
FROM Orders
INNER JOIN [Order Details]
    ON Orders.OrderID = [Order Details].OrderID
INNER JOIN Northwind_Mart.dbo.Product_Dim
    ON [Order Details].ProductID =
    Northwind_Mart.dbo.Product_Dim.ProductID
INNER JOIN Northwind_Mart.dbo.Customer_Dim
    ON Orders.CustomerID =

```

```

Northwind_Mart.dbo.Customer_Dim.CustomerID
INNER JOIN Northwind_Mart.dbo.Time_Dim
ON Orders.ShippedDate = Northwind_Mart.dbo.Time_Dim.TheDate
INNER JOIN Northwind_Mart.dbo.Shipper_Dim
ON Orders.ShipVia = Northwind_Mart.dbo.Shipper_Dim.ShipperID
INNER JOIN Northwind_Mart.dbo.Employee_Dim
ON Orders.EmployeeID =
Northwind_Mart.dbo.Employee_Dim.EmployeeID
WHERE (Orders.ShippedDate IS NOT NULL)

```

Итак, мы описали все шесть преобразований данных, необходимых для заполнения хранилища данными. Осталось их выполнить, о чем мы и расскажем в следующем разделе.

## Выполнение пакетов DTS

Созданный пакет DTS следует сохранить, выбрав опцию **Package | Save** из меню редактора пакетов DTS. Выполнить его можно, выбрав пункт меню **Package | Execute**. После этого начнется процесс преобразования данных и заполнения ими таблиц хранилища данных.

Для того чтобы данные в хранилище соответствовали текущему или недавнему состоянию оперативной базы данных, можно создать расписание, согласно которому будет автоматически выполняться данный пакет. Для этого следует выбрать его в Enterprise Manager и опцию **Schedule Package** — из контекстного меню. Далее следует выбрать нужный режим обновления данных в диалоговой панели **Edit Recurring Job Schedule** (рис. 7).

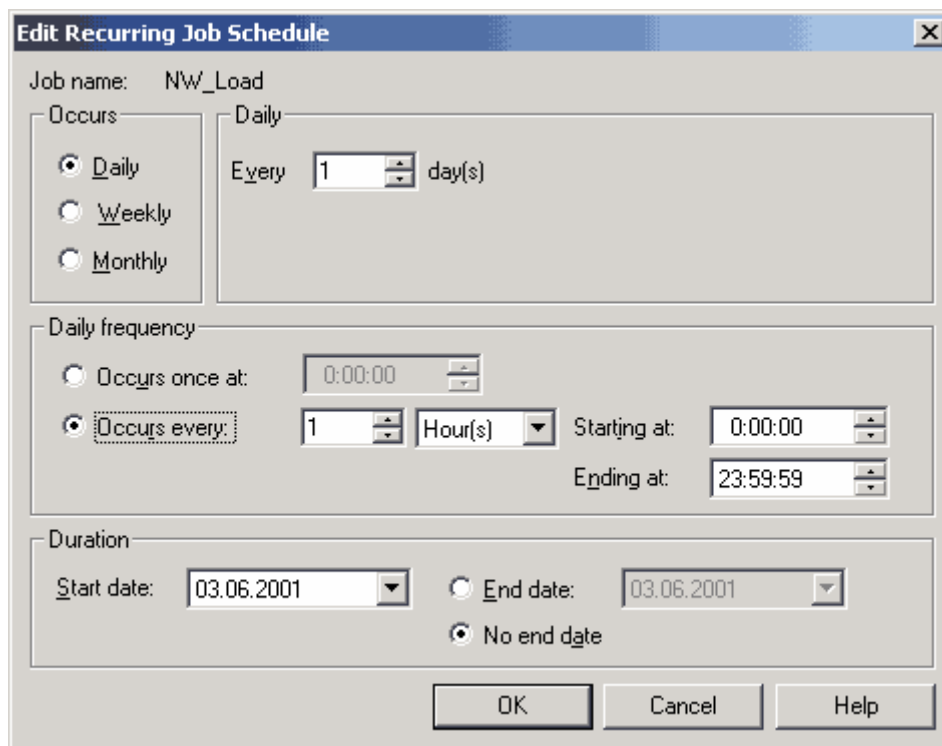


Рис. 7. Создание расписания выполнения пакета DTS

Отметим, что для запуска пакета по расписанию необходимо, чтобы был запущен SQL Server Agent — служба, инициирующая выполнение различных заданий по расписанию.

## Часть 5. Создание многомерных баз данных

Создание многомерных баз данных и описание источников данных

Создание коллективных измерений

- Создание измерения типа «дата/время»

- Создание регулярного измерения

- Создание измерения с несбалансированной иерархией

- Создание измерения типа «родитель-потомок»

Создание OLAP-кубов

- Создание описания куба

- Создание вычисляемых выражений

- Создание многомерного хранилища данных

В предыдущей статье данного цикла мы обсудили вопросы заполнения хранилищ данных и синхронизации их с содержимым оперативной базы данных. На этот раз мы рассмотрим, как на основании хранилищ данных можно создавать многомерные базы данных и OLAP-кубы с помощью Microsoft Analysis Services — аналитических сервисов, с архитектурой которых мы уже знакомы.

### Создание многомерных баз данных и описание источников данных

Рассмотрим создание многомерного OLAP-куба на основании хранилища данных Northwind\_Mart, которое мы создали и заполнили в предыдущей статье. Напомним, что это хранилище содержит таблицу фактов Sales\_Fact и таблицы измерений Employee\_Dim, Customer\_Dim, Product\_Dim, Time\_Dim, Shipper\_Dim. Отметим, что в процессе создания куба нам придется несколько модифицировать наше хранилище данных, с тем чтобы оно позволяло производить некоторые специальные виды анализа данных.

Для выполнения этого примера следует установить аналитические службы Microsoft SQL Server (напоминаем, что они входят в комплект поставки Microsoft SQL Server Enterprise Edition, Standard Edition, Developer Edition и Personal Edition) и запустить утилиту Analysis Manager, с помощью которой обычно и создаются многомерные базы данных.

Прежде всего следует зарегистрировать в Analysis Manager OLAP-сервер (он может находиться как на локальном компьютере, так и на другом компьютере в рамках локальной сети), выбрав пункт Register Server... из контекстного меню элемента Analysis Servers в левой части главного окна Analysis Manager. Затем нужно соединиться с OLAP-сервером, выбрав пункт Connect контекстного меню соответствующего элемента.

Поскольку OLAP-кубы хранятся в многомерных базах данных, создадим таковую, выбрав пункт New Database... из контекстного меню элемента, соответствующего OLAP-серверу, и введем имя базы данных и ее описание.

Прежде чем создавать OLAP-кубы, необходимо описать источники исходных данных для них. В нашем примере таким источником является созданное ранее хранилище Northwind\_Mart. Для описания источника данных выберем из контекстного меню элемента Data Sources пункт New Data Source... и заполним поля стандартной диалоговой панели Data Link Properties: в качестве провайдера данных укажем OLE DB Provider for SQL Server и выберем базу данных Northwind\_Mart (рис. 1).

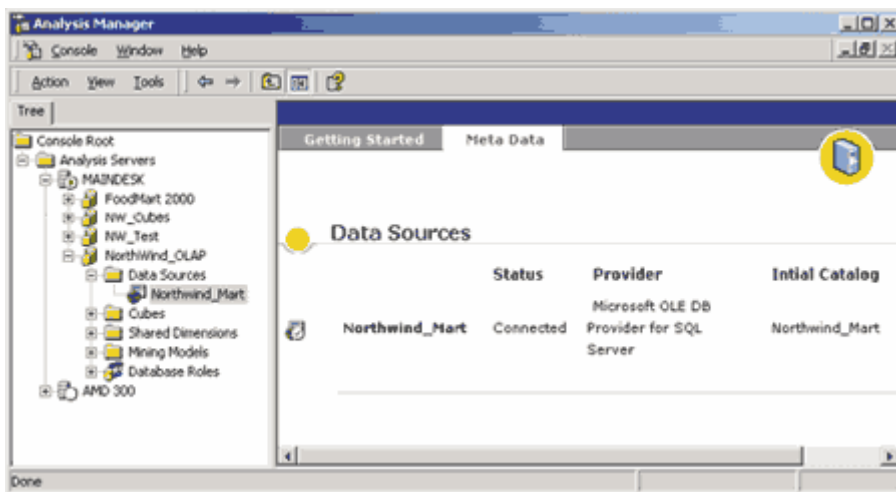


Рис. 1. Создание источника данных

Теперь можно приступить к созданию измерений и кубов.

## Создание коллективных измерений

Как мы уже знаем из предыдущих статей данного цикла, у OLAP-куба должно быть как минимум одно измерение. В Microsoft SQL Server Analysis Services измерения делятся на коллективные (shared dimensions) и частные (private dimensions).

Коллективные измерения — это измерения, которые могут быть использованы одновременно в нескольких кубах. Их применение удобно в том случае, когда измерение основано на стандартных данных, применимых при анализе различных предметных областей. Типичным примером создания таких измерений может быть, например, список сотрудников компании. Коллективные измерения принадлежат самой многомерной базе данных и не зависят от того, какие кубы имеются в многомерной базе данных и есть ли они там вообще.

Частные измерения принадлежат конкретному кубу и создаются вместе с ним. Они применяются в том случае, когда данное измерение имеет смысл только в одной конкретной предметной области.

Создать как коллективное, так и частное измерение можно двумя способами: с помощью соответствующего мастера и с помощью редактора измерений.

## Создание измерения типа «дата/время»

В качестве примера создадим коллективное измерение, основанное на таблице хранилища данных Time\_Dim, воспользовавшись мастером создания измерений (Dimension wizard). Запустить его можно с помощью команды New Dimension | Wizard из контекстного меню элемента Shared Dimensions. Затем необходимо ответить на вопросы мастера создания измерений. В первую очередь следует выбрать, на основании чего мы создаем измерение. Поскольку исходное хранилище данных основано на схеме «звезда», выберем в мастере создания измерений опцию Star Schema: a single dimension table, а затем — имя таблицы, служащей источником данных для создаваемого измерения (в нашем примере — Time\_Dim, рис. 2):

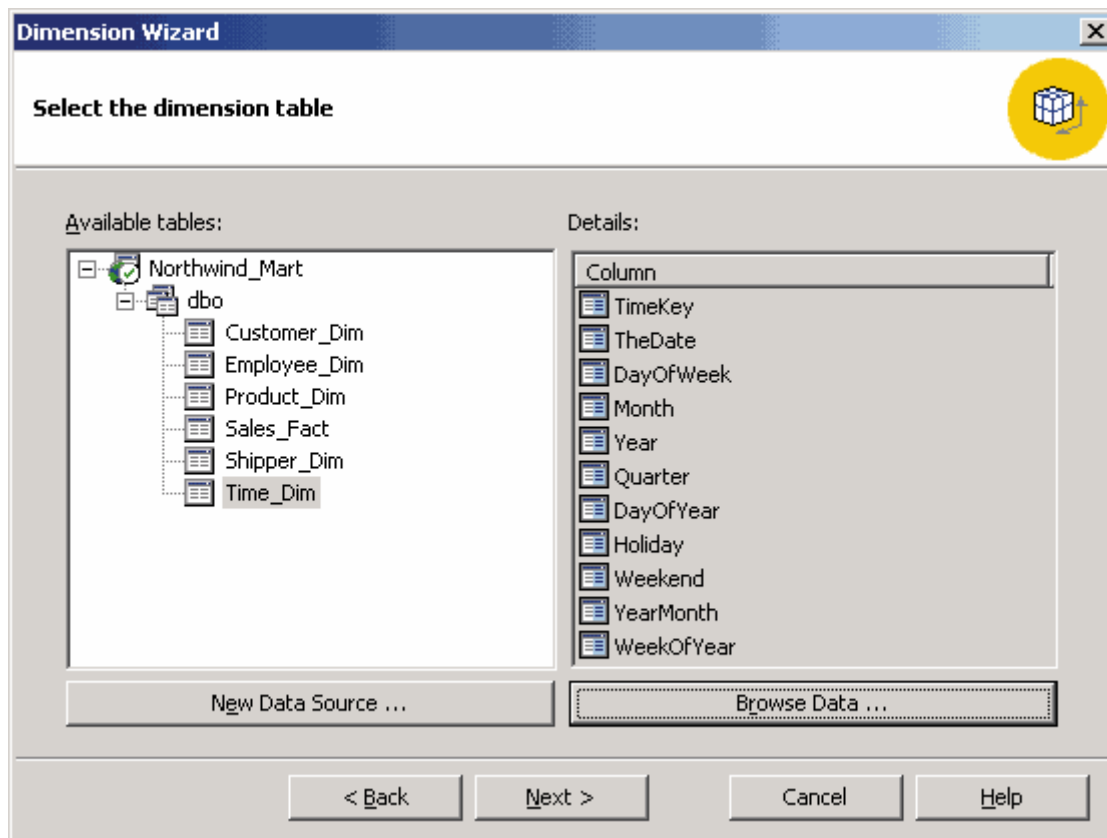


Рис. 2. Выбор таблицы для создания измерения

Иерархия данных в измерениях, основанных на данных типа «дата/время», подчиняется определенным стандартным правилам — ведь время измеряется в годах, месяцах, днях, часах, минутах независимо от того, какую предметную область мы анализируем. Поэтому измерения в OLAP-средствах обычно делятся на стандартные (не имеющие отношения ко времени) и временные. Поскольку наше измерение относится к последним, в диалоговой панели Select the dimension type выберем опцию Time Dimension и в качестве колонки, в которой содержатся данные типа «дата/время», укажем поле TheDate.

Теперь нам необходимо выбрать уровни иерархии измерений (например, решить, интересна ли нам информация о часах и минутах, нужны ли нам номера недель года и т.д.), а также определить, когда начинается год с точки зрения данного измерения. Это довольно важная возможность — ведь во многих странах начало финансового года не совпадает с началом года календарного. В нашем случае выберем уровни Year, Quarter, Month, Day и согласимся с тем, что год начинается 1 января (рис. 3).

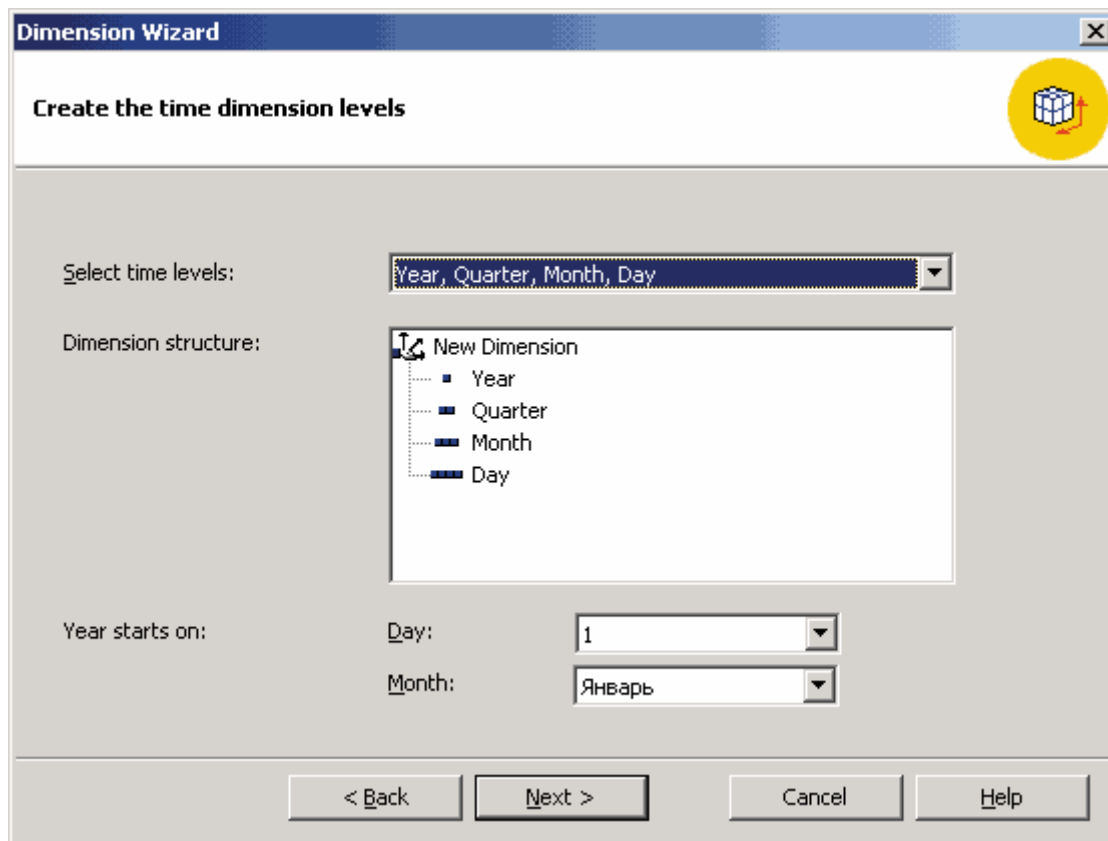


Рис. 3. Создание измерения типа «дата/время»

Далее нам предстоит выбрать, является ли измерение изменяющимся (changing dimension). В изменяющихся измерениях (новинка в SQL Server 2000) можно перемещать члены измерений между уровнями без перерасчета данных измерения, что во многих случаях бывает удобно. Однако измерения типа Time, как правило, не делают изменяющимися — обычно никто не перемещает месяцы из одного года в другой. Поэтому в данном случае мы не будем выбирать эту опцию.

В заключительной диалоговой панели мы должны ввести имя будущего измерения и, если есть необходимость, создать иерархию в измерении и задать ее имя. Дело в том, что при необходимости можно создать еще одно измерение, основанное на тех же данных, с тем же именем, но с другой иерархией, например Year, Week, Day; в этом случае мы имеем разное представление одних и тех же данных. Присвоим созданной иерархии имя YQMD (рис. 4).

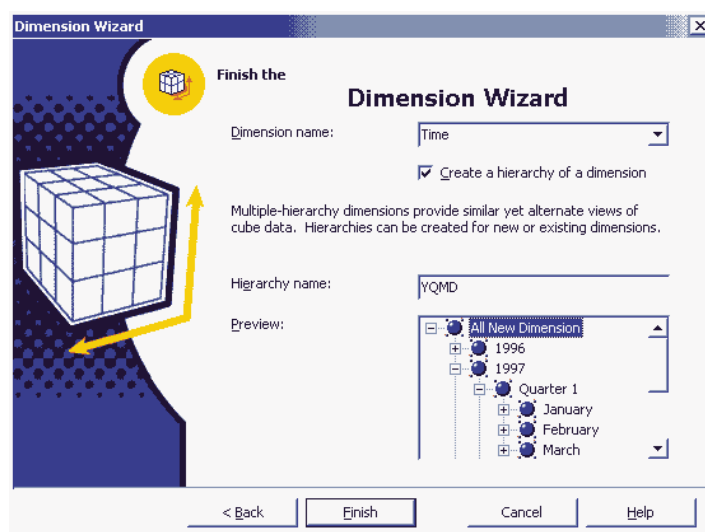


Рис. 4. Создание источника данных

Создание измерения заканчивается запуском редактора измерений — Dimension Editor. В нем при необходимости можно внести изменения в структуру измерения, например добавив дополнительные уровни или свойства членов измерения. Так, если мы планируем анализировать зависимость продаж от дня недели или сравнивать продажи в выходные, праздничные и будние дни, можно перенести в раздел Member Properties уровня Day поля Day of Week, Holiday и Weekend исходной таблицы Time\_Dim (рис. 5).

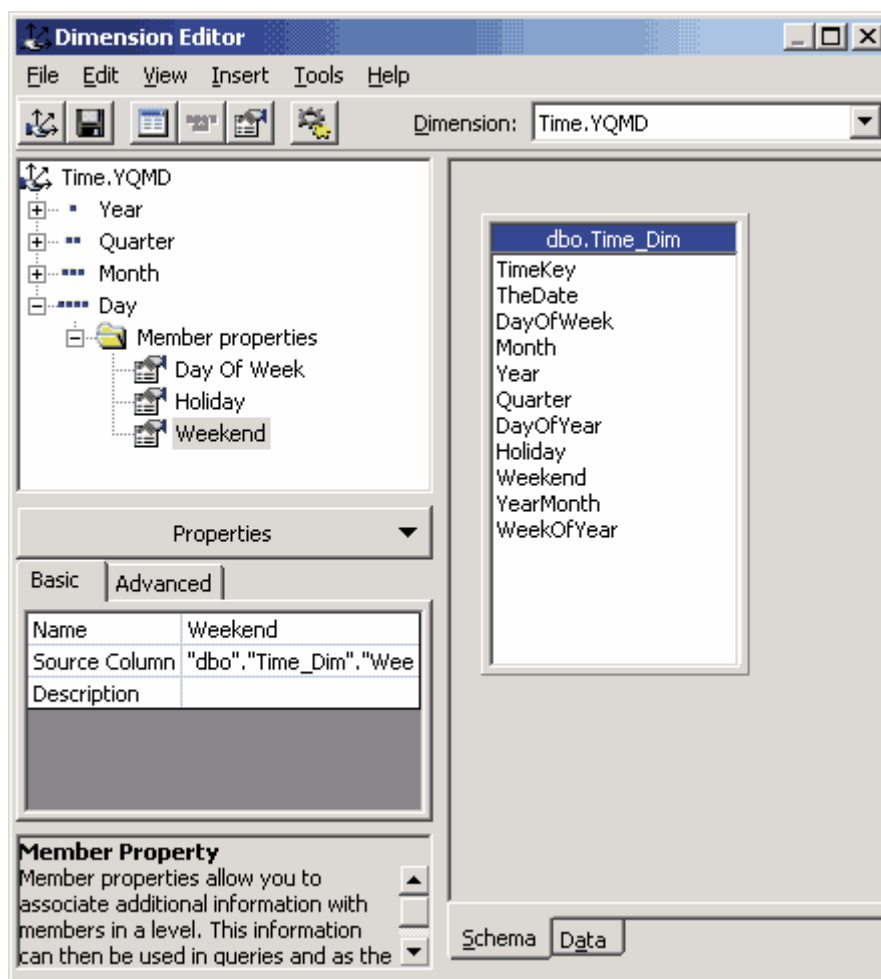


Рис. 5. Dimension Editor

Теперь можно сохранить созданное измерение, выбрав пункт меню File | Save, и закрыть редактор измерений.

Повторим все указанные действия, выбрав при этом другую иерархию — Year, Week, Day, и назовем вновь созданное измерение Time.YWD.

## Создание регулярного измерения

Следующее коллективное измерение создадим с помощью редактора измерений. Запустить его можно с помощью команды New Dimension | Editor из контекстного меню элемента Shared Dimensions. Далее в диалоговой панели Select the dimension table выберем таблицу Product\_Dim. В редакторе измерений создадим два уровня иерархии этого измерения — CategoryName и ProductName — и перенесем мышью соответствующие имена полей в левую часть редактора измерений. В качестве свойств членов измерения уровня ProductName выберем поля SupplierName и ListUnitPrice. Поскольку переносить продукты из одной категории в другую представляется более разумным, чем переносить месяцы из одного



года в другой, сделаем это измерение изменяющимся — соответствующее свойство доступно на вкладке Advanced панели Properties в левой нижней части редактора измерений. Сохраним созданное измерение под именем Product (рис. 6).

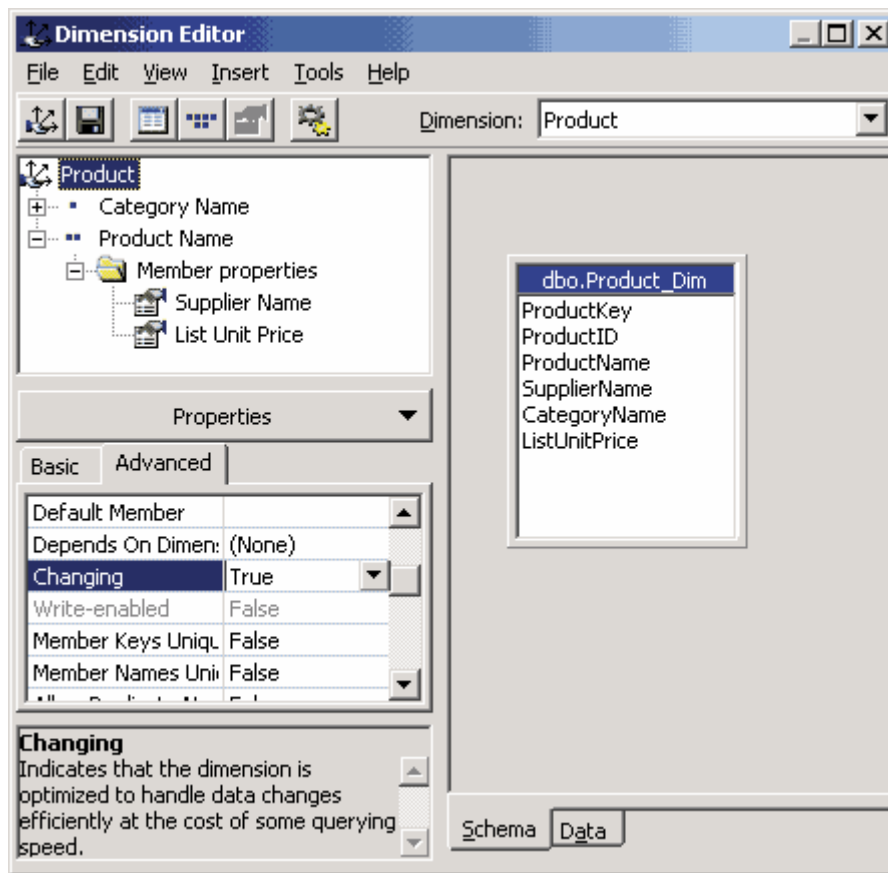


Рис. 6. Создание регулярного измерения в Dimension Editor

## Создание измерения с несбалансированной иерархией

Следующее измерение будет содержать географические сведения. Такие измерения являются типичными кандидатами для создания так называемых неровных (ragged) иерархий — частного случая несбалансированных (unbalanced) иерархий. Как известно, административно-территориальное деление в разных странах осуществляется по разным правилам: в некоторых странах есть регионы, штаты, административные округа, а в некоторых достаточно указать населенный пункт, и в этом случае сведения о штате или регионе могут отсутствовать.

Чтобы создать измерение с несбалансированной иерархией, добавим в хранилище данных представление, которое будет содержать исходные данные для создания этого измерения:

```
CREATE VIEW dbo.CustomerView_Dim
AS
SELECT
CustomerKey, CustomerID, CompanyName, ContactName, ContactTitle, Address, City,
Region,
REPLACE(Region, 'Other', Country) AS Region1,
PostalCode, Country, Phone, Fax
FROM dbo.Customer_Dim
```

Это представление содержит данные из таблицы Customer\_Dim, а также вычисляемое поле Region1, содержащее название страны вместо строки «Other» в тех случаях, когда

сведения о клиенте не содержат данных о регионе или штате. Это поле нам потребуется в дальнейшем для создания несбалансированной иерархии.

Теперь создадим измерение, основанное на вновь созданном представлении CustomerView\_Dim хранилища данных. Последовательность действий в этом случае сходна с предыдущим примером. В качестве уровней иерархии этого измерения мы выберем поля Country, Region1, City, CompanyName. Добавим в раздел Member Properties уровня Company Name поля Contact Name и Contact Title.

Несбалансированные иерархии обычно базируются на сокрытии членов измерения, содержащих избыточные сведения. В данном случае таковым является уровень Region1. Выберем его в редакторе измерений и на странице Advanced раздела Properties установим свойство Hide Member If равным Parent's name. В этом случае все члены уровня Region1, содержащие названия стран, будут скрыты (рис. 7).

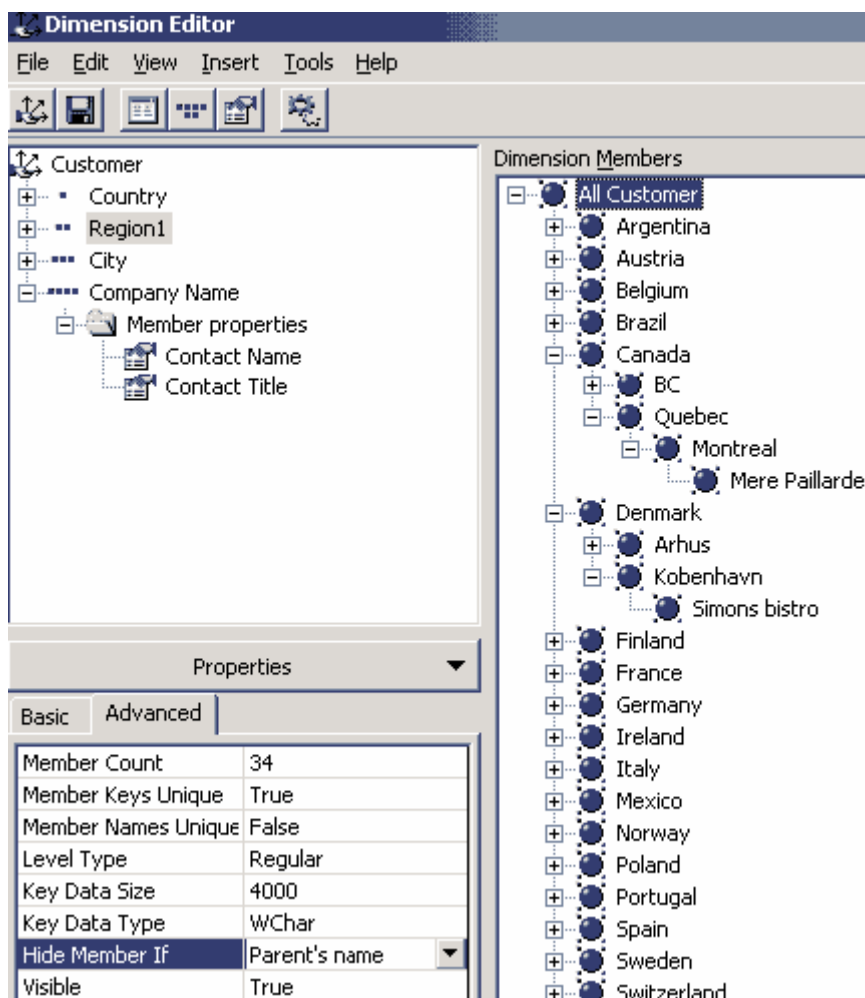


Рис. 7. Несбалансированная иерархия

Именно для этого мы и создавали представление в хранилище данных — строка «Other», вполне устраивавшая нас при обращении к самому хранилищу данных, будучи именем члена измерения, не может выступать в качестве условия его скрытия.

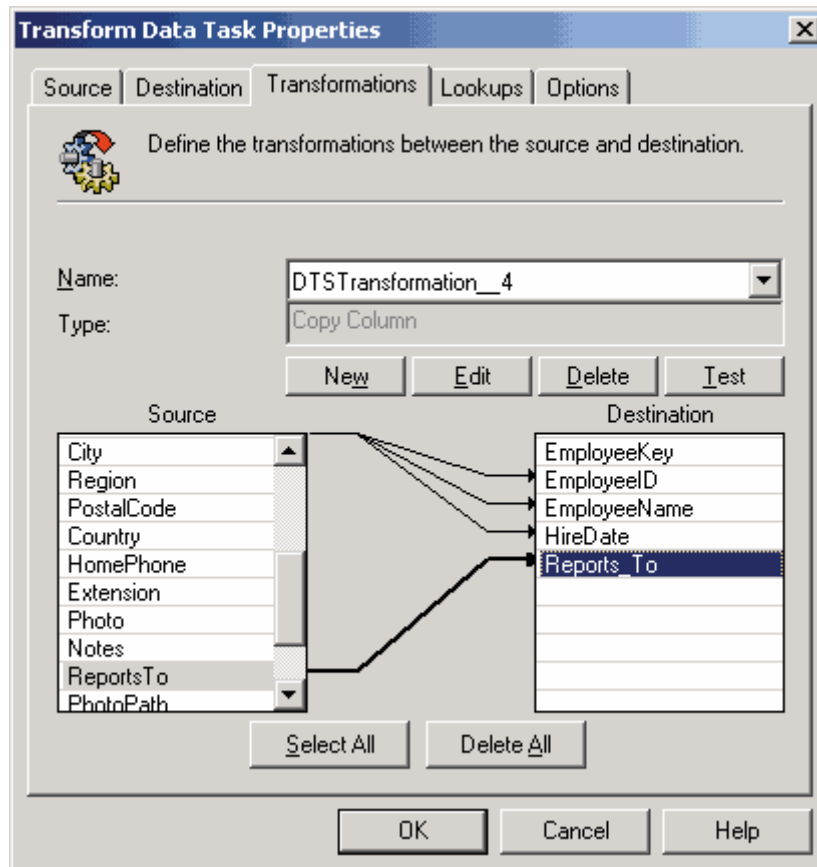
## Создание измерения типа «родитель-потомок»

Следующее измерение, которое мы создадим, будет основано на таблице Employee\_Dim хранилища данных. Обычно измерения, содержащие сведения об административной подчиненности сотрудников, содержат еще один тип несбалансированных иерархий —

иерархии типа «родитель-потомок» (parent-child). Такие иерархии нередко основаны на таблицах, где первичный ключ является одновременно и внешним ключом. Исходная таблица Employees базы данных Northwind действительно содержит сведения об административной подчиненности сотрудников (и имеет соответствующий внешний ключ), а таблица Employee\_Dim — нет. Поэтому в первую очередь модифицируем ее, добавив к ней поле Reports\_To:

```
ALTER TABLE dbo.Employee_Dim ADD Reports_To int
```

Затем с помощью DTS добавим в это поле данные из таблицы Employees (рис. 8).



	EmployeeKey	EmployeeID	EmployeeName	HireDate	Reports_To
	55	1	Nancy Davolio	01.05.1992	2
	56	2	Andrew Fuller	14.08.1992	<NULL>
	57	3	Janet Leverling	01.04.1992	2
	58	4	Margaret Peacock	03.05.1993	2
	59	5	Steven Buchanan	17.10.1993	2
	60	6	Michael Suyama	17.10.1993	5
	61	7	Robert King	02.01.1994	5
	62	8	Laura Callahan	05.03.1994	2
	63	9	Anne Dodsworth	15.11.1994	5
*					

Рис. 8. Добавление данных в таблицу Employee\_Dim

Теперь можно создать измерение на основе таблицы Employee\_Dim с помощью Dimension Wizard. В его первой диалоговой панели выберем опцию Parent-Child: Two related columns in a single dimension table. Далее опишем связь между двумя полями таблицы

Employee\_Dim, указав имя поля EmployeeID в качестве свойства Member\_Key создаваемого измерения, Reports\_To — в качестве его свойства Parent\_Key, EmployeeName — в качестве его свойства Member Name (рис. 9).

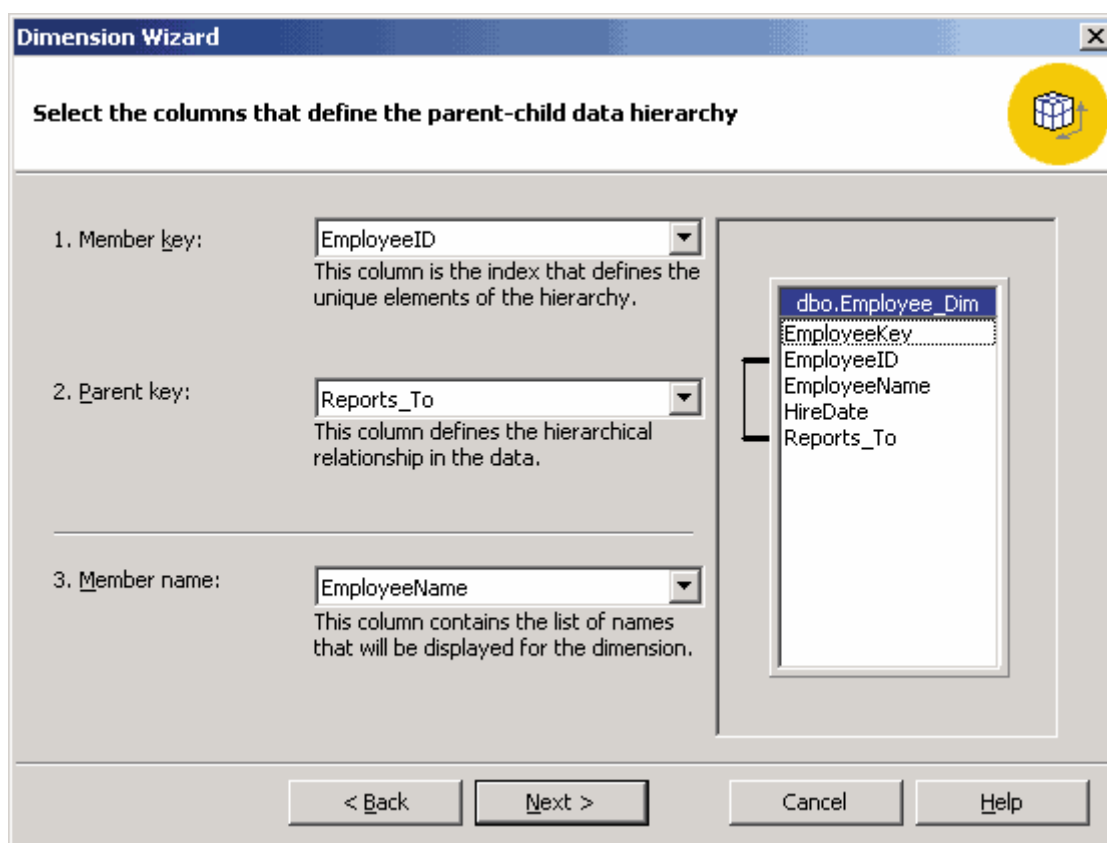


Рис. 9. Определение параметров иерархии «родитель-потомок»

В диалоговой панели Select advanced options следует выбрать опцию Members with data, а в панели Set members with data property — опции Nonleaf members have associated data и Data members are visible. Это позволит анализировать как собственные результаты работы сотрудников, имеющих подчиненных, так и результаты работы их подчиненных. Создадим иерархию в этом измерении, назвав ее Employee.РС, и укажем в качестве свойства члена измерения поле Hire Date. В результате мы получим иерархию, показанную на рис. 10.

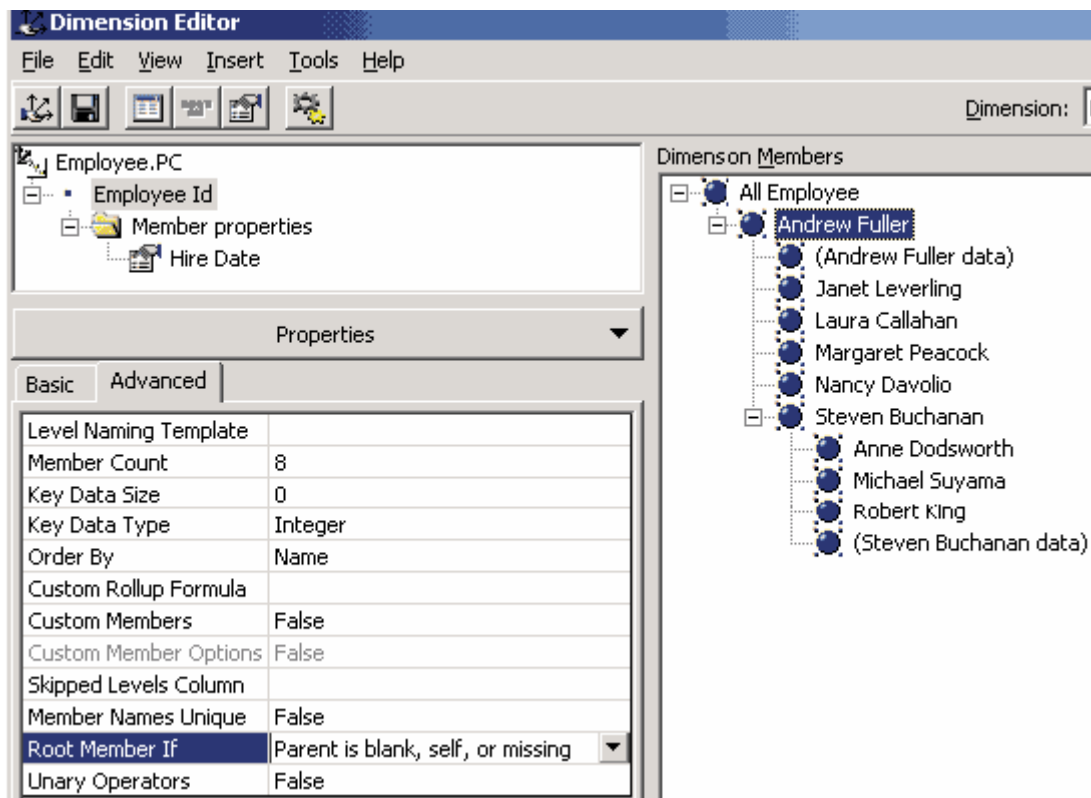


Рис. 10. Иерархия «родитель-потомок»

В качестве альтернативы создадим еще одну иерархию — Employee.Regular, содержащую один уровень Employee\_Name; в качестве свойств члена этого уровня выберем поля Hire Date и Reports\_To.

На этом мы закончим создание коллективных измерений и приступим к созданию куба.

## Создание OLAP-кубов

Как и измерение, куб можно создать с помощью соответствующего мастера или непосредственно в редакторе кубов. В качестве примера создадим куб, основанный на нашем хранилище данных Northwind\_Mart и использующий созданные выше измерения. Запустить мастер создания кубов можно командой New Cube | Wizard из контекстного меню элемента Cubes.

## Создание описания куба

Первое, что следует сделать после запуска мастера, — выбрать таблицу фактов для будущего куба. В нашем случае это таблица Sales\_Fact. Далее из таблицы фактов следует выбрать одно или несколько полей, на основе которых вычисляются меры куба (то есть поля, данные которых подлежат суммированию либо обработке с помощью других агрегатных функций). Выберем поля Line Item Total, Line Item Quantity и Line Item Discount (рис. 11).

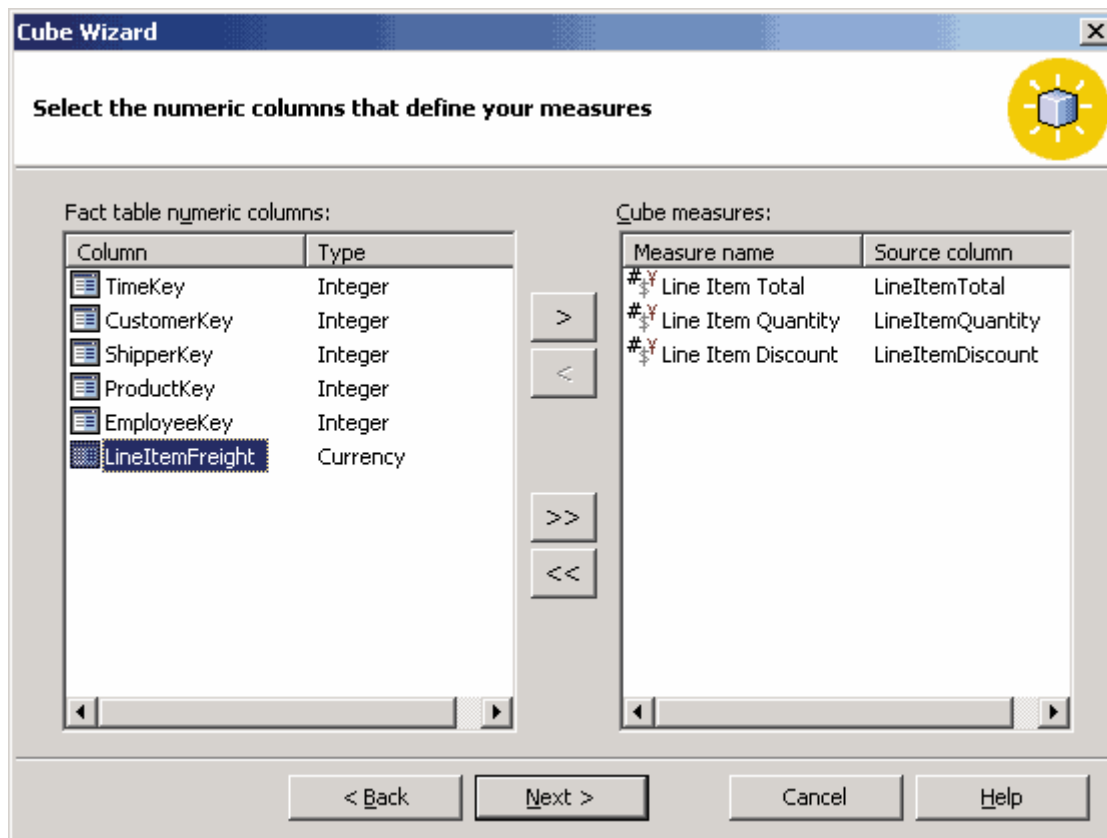


Рис. 11. Выбор мер куба

Следующим шагом будет выбор коллективных измерений, используемых в этом кубе, а также создание недостающих частных измерений. Выберем коллективные измерения Employee.PC, Employee.Regular, Time.YQMD, Product и Customer. Кроме того, добавим новое измерение Shipper, нажав кнопку New Dimension в диалоговой панели выбора измерений. Это приведет к запуску уже знакомого нам мастера создания измерений; в последней из диалоговых панелей мастера в этом случае мы можем выбрать, каким будет создаваемое измерение — частным или коллективным (рис. 12).

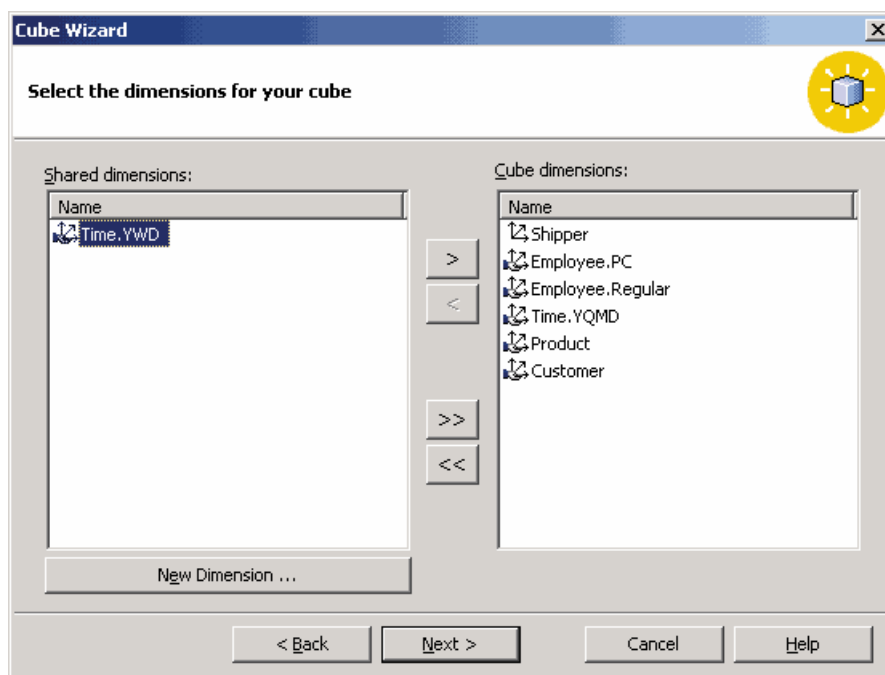


Рис. 12. Выбор измерений куба

Таким образом, мы определили метаданные куба. По окончании работы мастера будет запущен редактор кубов, в котором при необходимости можно внести исправления в определение куба, например добавить или удалить измерения и меры, создать вычисляемые значения и т.д. (рис. 13).

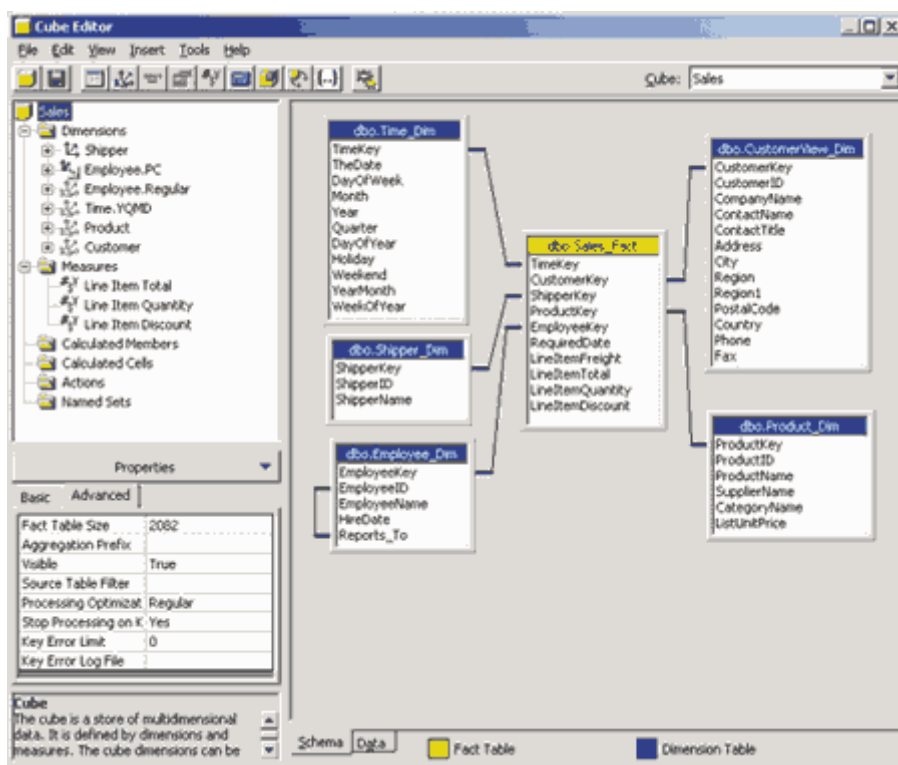


Рис. 13. Редактор кубов

## Создание вычисляемых выражений

Теперь попробуем добавить к нашему кубу вычисляемые значения, то есть значения, которые не хранятся в самом кубе, а вычисляются «на лету». Типичным примером такого значения может быть дополнительная мера, вычисленная на основе уже имеющихся. При вычислениях можно использовать как функции из библиотеки, входящей в состав Analysis Services, так и выражения VBA, а также собственные библиотеки функций (последние следует зарегистрировать в Analysis Services).

Для создания вычисляемых выражений следует выбрать раздел Calculated Members и из контекстного меню выбрать опцию New Calculated Member. После этого будет запущен построитель выражений (Calculated Member Builder), в котором можно создавать и редактировать выражения, перетаскивая мышью имена измерений и их уровней, мер, имена функций. Например, перенесем в поле для выражения имена мер [Measures].[Line Item Total] и [Measures].[Line Item Discount], поставим между ними знак вычитания, а в качестве значения Member Name введем Discounted Total (рис. 14).

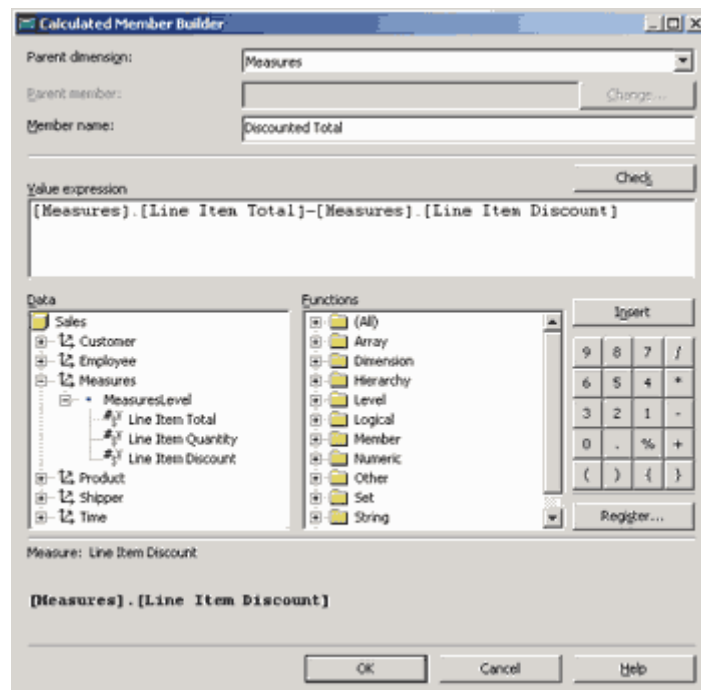


Рис. 14. Редактор вычисляемых выражений

В результате мы получили еще одну меру — значение суммы, вырученной за товар, с учетом скидки.

Теперь можно сохранить определение куба, выбрав пункт меню File | Save редактора кубов.

## Создание многомерного хранилища данных

Процесс создания куба на этом не завершен — мы создали его определение, но не производили никаких вычислений. Прежде чем произвести вычисления, напомним, что существует несколько способов хранения агрегатных данных (мы уже обсуждали эти способы в предыдущих статьях данного цикла). Для данного примера вполне подойдет хранение всех данных в многомерной базе данных (MOLAP), так как объем исходных данных невелик. Однако в других случаях следует оценить, какой способ хранения наиболее выгоден для данной задачи.

Еще один вопрос, который следует решить при создании многомерного хранилища данных — сколько агрегатов следует хранить? Агрегаты — это заранее вычисленные агрегатные данные, соответствующие ячейкам куба. Чем их больше, тем быстрее выполняются запросы к многомерному хранилищу и тем больше объем самого хранилища. Поэтому в общем случае требуется некое их количество, позволяющее осуществить разумный баланс между компактностью и производительностью.

Для определения количества агрегатов и их вычисления следует запустить Storage Design wizard — мастер создания многомерного хранилища. Для этого в редакторе кубов следует выбрать пункт меню Tools | Design Storage.

В первой диалоговой панели следует указать способ хранения данных — MOLAP, ROLAP или HOLAP (в нашем примере мы выберем MOLAP). Затем выбрать, какова должна быть производительность при выполнении запросов (либо будущий максимальный объем хранилища). После этого можно нажать на кнопку Start и получить зависимость производительности от объема хранилища (рис. 15).



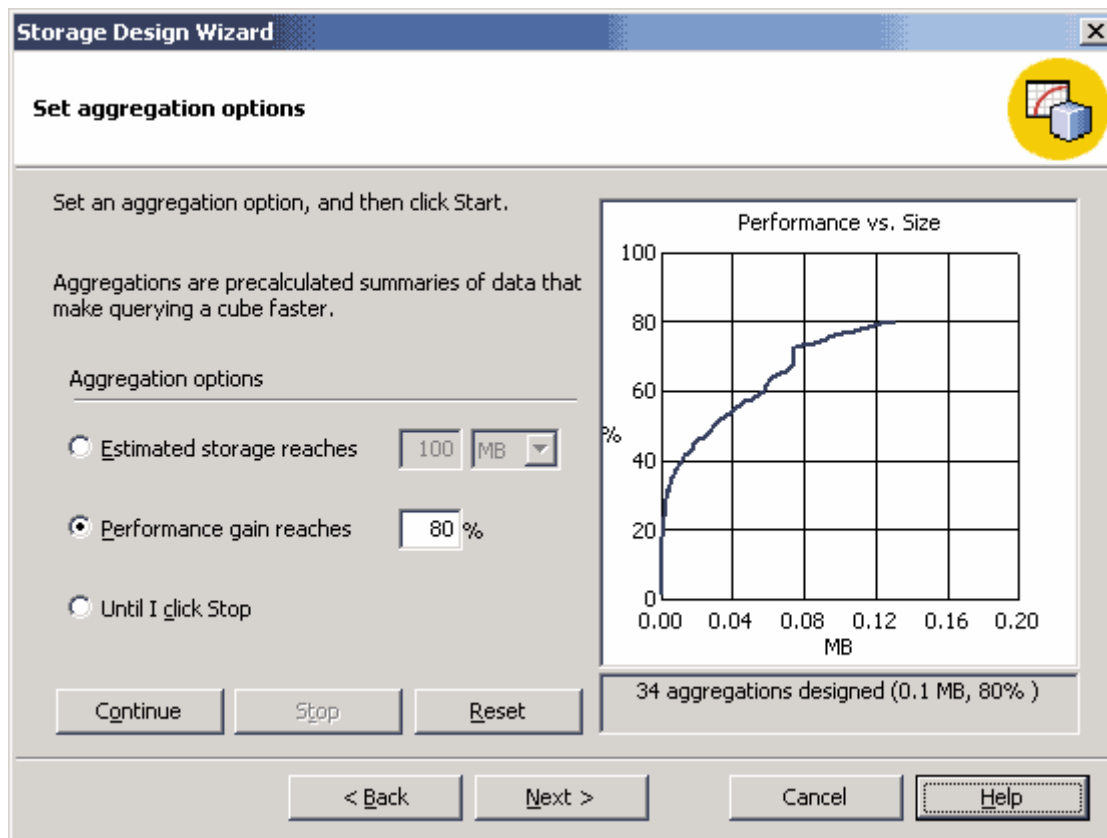


Рис. 15. Определение количества агрегатов

И наконец, нам необходимо вычислить сами агрегатные данные. Это можно сделать как в том же мастере создания хранилища, так и в редакторе кубов (команда Tools | Process Cube, рис. 16).

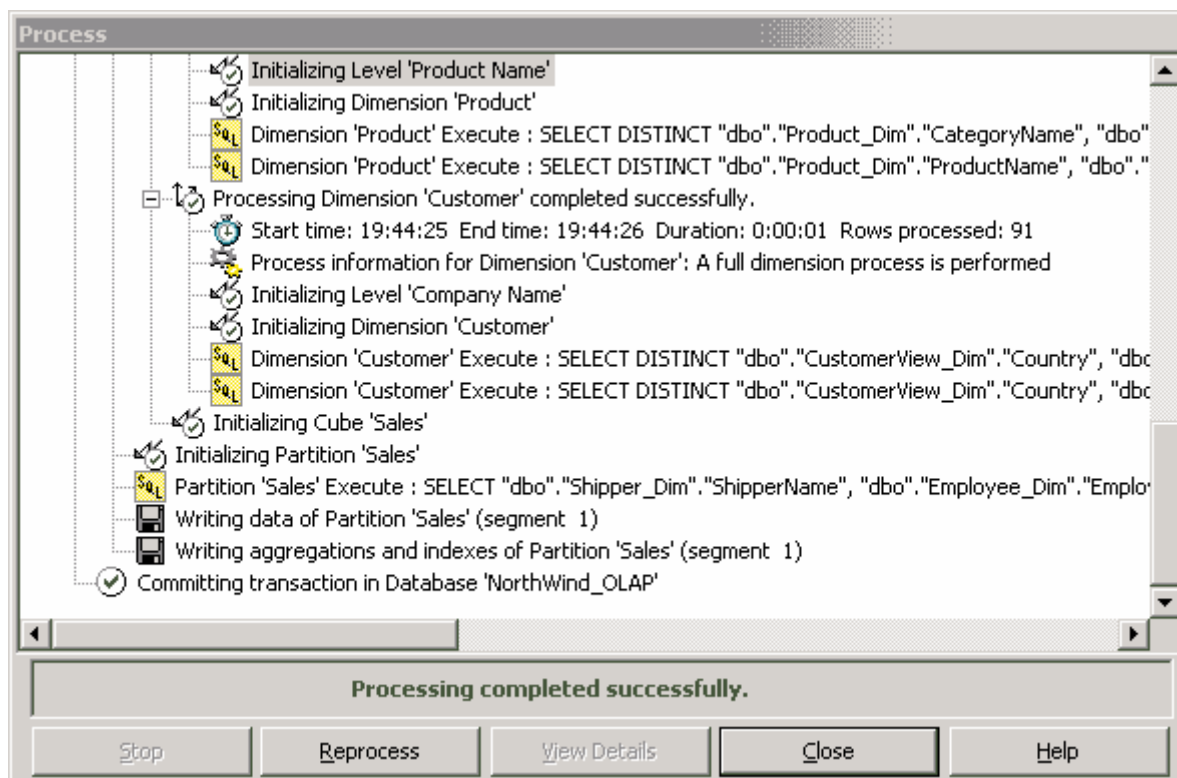


Рис. 16. Вычисление агрегатных данных

Теперь, когда куб готов, можно заняться его просмотром в редакторе кубов (для этого нужно выбрать закладку Data в нижней части экрана, рис. 17).

Measures	Discounted Total				
Employee.Regular	All Employee				
Product	All Product				
Customer	All Customer				
Time.YQMD	All Time				

- Level 02	- Level 03	Level 04	Shipper Name	Federal Shipping	Speedy Express
All Employee	All Employee Total		All Shipper		
			\$1 239 855.61	\$376 604.57	\$346 231.04
	Andrew Fuller Total		\$1 239 855.61	\$376 604.57	\$346 231.04
	(Andrew Fuller data)		\$162 769.78	\$41 277.99	\$61 669.89
	Janet Leverling		\$202 812.84	\$64 357.33	\$50 455.80
	Laura Callahan		\$123 842.68	\$35 529.22	\$33 468.56
	Margaret Peacock		\$225 763.70	\$67 026.43	\$58 511.89
- Andrew Fuller	Nancy Davolio		\$187 277.38	\$64 357.90	\$53 958.58
		Steven Buchanan Total	\$337 389.23	\$104 055.71	\$88 166.25
		Anne Dodsworth	\$76 450.07	\$21 860.20	\$17 694.56
	- Steven Buchanan	Michael Suyama	\$72 527.63	\$23 433.29	\$23 809.22
		Robert King	\$119 619.25	\$45 713.12	\$22 039.44
		(Steven Buchanan data)	\$68 792.28	\$13 049.11	\$24 623.00

Рис. 17. Просмотр сечений куба

В редакторе кубов мы можем просматривать различные двухмерные сечения куба, перемещая имена измерений на горизонтальную и вертикальную оси, а также скрывая и раскрывая уровни. Это самый простой из способов просмотра кубов.

## Часть 6. Microsoft Excel как OLAP-клиент

Средства чтения OLAP-данных в Microsoft Office

Манипуляция OLAP-данными в Microsoft Excel

Создание сводной таблицы с данными OLAP-кубов

Манипуляция отображением данных в сводной таблице

Создание сводных диаграмм с данными OLAP-кубов

Создание локальных OLAP-кубов

В предыдущей части данной статьи мы рассмотрели процесс создания многомерных баз данных для Microsoft Analysis Services и содержащихся в них объектов, а также ознакомились с простейшим средством просмотра сечений кубов, встроенным в Analysis Manager. Этот способ работы с OLAP-данными — не единственный (и далеко не самый удобный, по крайней мере для конечного пользователя) из возможных на сегодняшний день. Помимо него существует немало других средств просмотра этих данных — от приложений Microsoft Office и входящих в его состав компонентов до многочисленных средств просмотра OLAP-данных, предлагаемых сторонними производителями. Разработчики могут создавать собственные приложения для работы с OLAP-данными — как с применением компонентов Microsoft Office, так и без них.

Мы начнем с рассмотрения одного из самых простейших способов работы с OLAP-данными — использования Microsoft Excel.

### Средства чтения OLAP-данных в Microsoft Office

Прежде чем обсуждать возможности Microsoft Excel как OLAP-клиента, кратко остановимся на компонентах Microsoft Office, используемых для работы с OLAP-данными, — это позволит нам в дальнейшем избежать терминологической путаницы. Тем более что все эти компоненты содержат в своем названии словосочетание PivotTable.

Первым из компонентов Microsoft Office, предназначенных для создания OLAP-клиентов, является набор библиотек PivotTable Service. С одной стороны, он является составной частью Analysis Services и выполняет роль связующего звена между Analysis Services и их клиентами (не обязательно имеющими отношение к Microsoft Office). PivotTable Service может быть установлен отдельно на компьютер, на котором эксплуатируются какие-либо клиенты Analysis Services; для его установки в состав Analysis Services входит отдельный дистрибутив. С другой стороны, PivotTable Service входит и в состав Microsoft Office 2000/XP и при этом может быть использован не только для работы с данными Analysis Services, но и для создания и чтения локальных OLAP-кубов, не имеющих отношения к Analysis Services, как с помощью Microsoft Excel, так и без него.

Вторым компонентом, который может быть использован для просмотра OLAP-кубов, является служба, называемая PivotTable Reports, — средство создания сводных таблиц Microsoft Excel. Это средство позволяет получать, сохранять в кэше в оперативной памяти и отображать на листах рабочих книг двухмерные и трехмерные наборы агрегатных данных на основе данных из реляционных СУБД и рабочих книг Excel. PivotTable Reports входит в Excel начиная с версии 5.0, но возможность считывать с помощью него данные из OLAP-кубов Analysis Services, равно как и создавать локальные OLAP-кубы, впервые появилась в Excel 2000. Отметим, что средство создания сводных таблиц Excel использует библиотеки PivotTable Services.

И наконец, третьим компонентом, применяемым при создании OLAP-клиентов, является PivotTable List — элемент управления ActiveX, входящий в состав Microsoft Office Web Components и предназначенный для просмотра сечений OLAP-кубов. Применяется он главным образом на Web-страницах, а иногда и в обычных Windows.

Выяснив, что представляют собой средства чтения OLAP-кубов Microsoft Office, мы можем перейти к более детальному рассмотрению процесса чтения и отображения OLAP-кубов с помощью Microsoft Excel.

## Манипуляция OLAP-данными в Microsoft Excel

Как было отмечено выше, средства создания сводных таблиц Microsoft Excel хранят в кэше агрегатные данные, вычисленные на основе данных из реляционных СУБД или полученные от OLAP-серверов. Манипулируя сводной таблицей, пользователь может управлять отображением данных из этого кэша.

Прежде чем приступить к созданию примера, заметим, что посредством Microsoft Excel 2000 можно корректно отображать данные из OLAP-кубов, созданных с помощью Microsoft SQL Server 7.0 OLAP Services. Что касается OLAP-кубов, созданных с помощью Microsoft SQL Server 2000 Analysis Services, по большей части посредством Microsoft Excel 2000, то они также отображаются корректно, однако имеются и некоторые ограничения. Например, при создании локальных кубов OLAP или при сохранении сводной таблицы в виде Web-страницы с помощью соответствующих мастеров автоматически выбирается OLE DB-провайдер предыдущей версии (версии 7.0), не поддерживающий несбалансированные измерения. Это приводит к сообщениям об ошибках и к игнорированию таких измерений или даже всего источника данных.

При использовании же Microsoft Excel 2002 эти проблемы не возникают.

## Создание сводной таблицы с данными OLAP-кубов

В качестве примера создадим сводную таблицу, содержащую данные OLAP-куба, созданного ранее (см. часть 5 этой статьи в КомпьютерПресс № 8'2001). Для этого запустим Microsoft Excel и из меню Data выберем PivotTable and PivotChart Report. После этого управление будет передано мастеру PivotTable and PivotChart Wizard. В первой диалоговой панели этого мастера укажем, что для построения сводной таблицы выбирается внешний источник данных, для чего выберем опцию External data source. Затем укажем, что это за источник, нажав кнопку Get Data в следующей диалоговой панели, что приведет к запуску приложения Microsoft Query. Далее выберем закладку OLAP Cubes и, если в операционной системе еще нет описания соответствующего источника данных, создадим его (рис. 1).

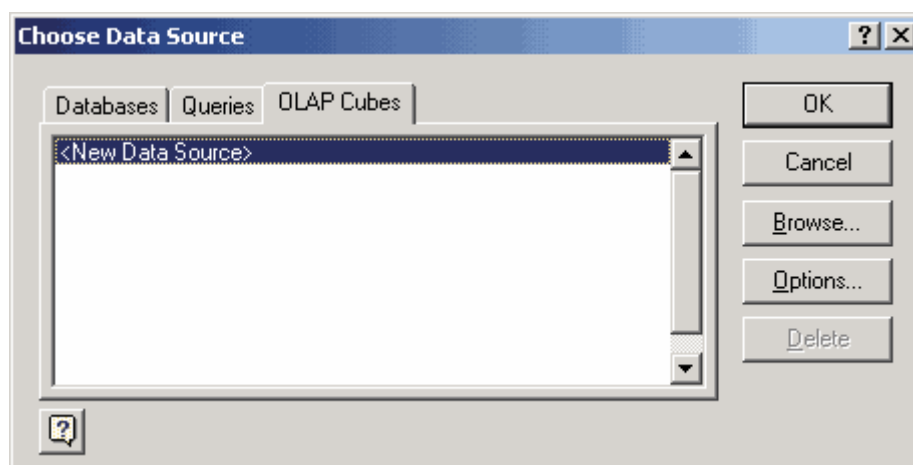
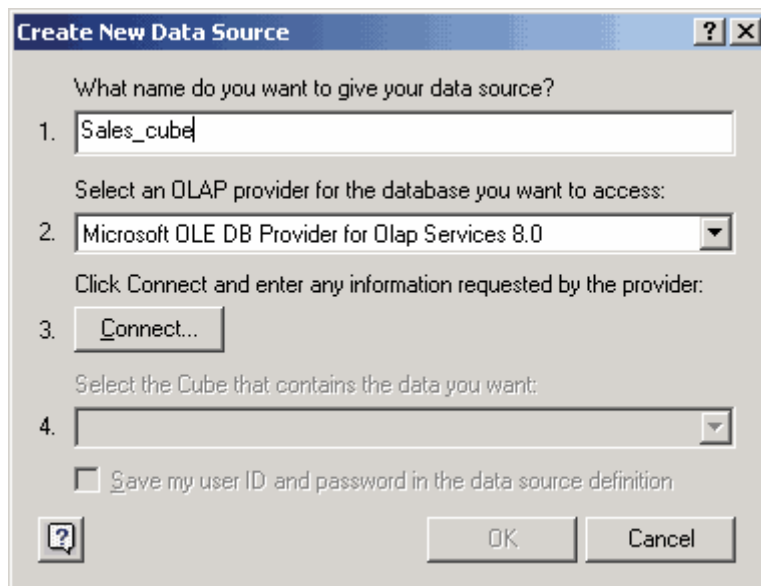


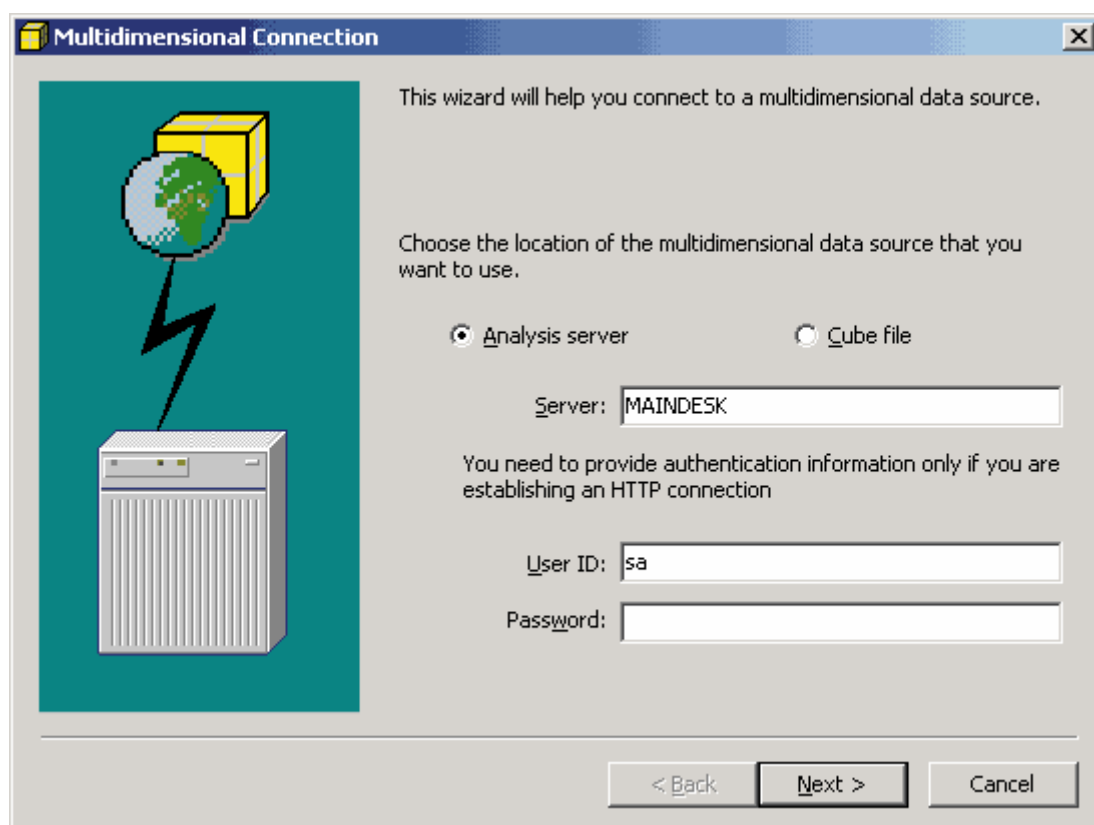
Рис. 1. Описание источника данных

В процессе создания источника данных укажем его имя, выберем OLE DB-провайдер (в нашем случае — Microsoft OLE DB Provider for OLAP Services 8.0, поскольку мы используем Microsoft SQL Server 2000 Analysis Services) и нажмем на кнопку Connect (рис. 2).



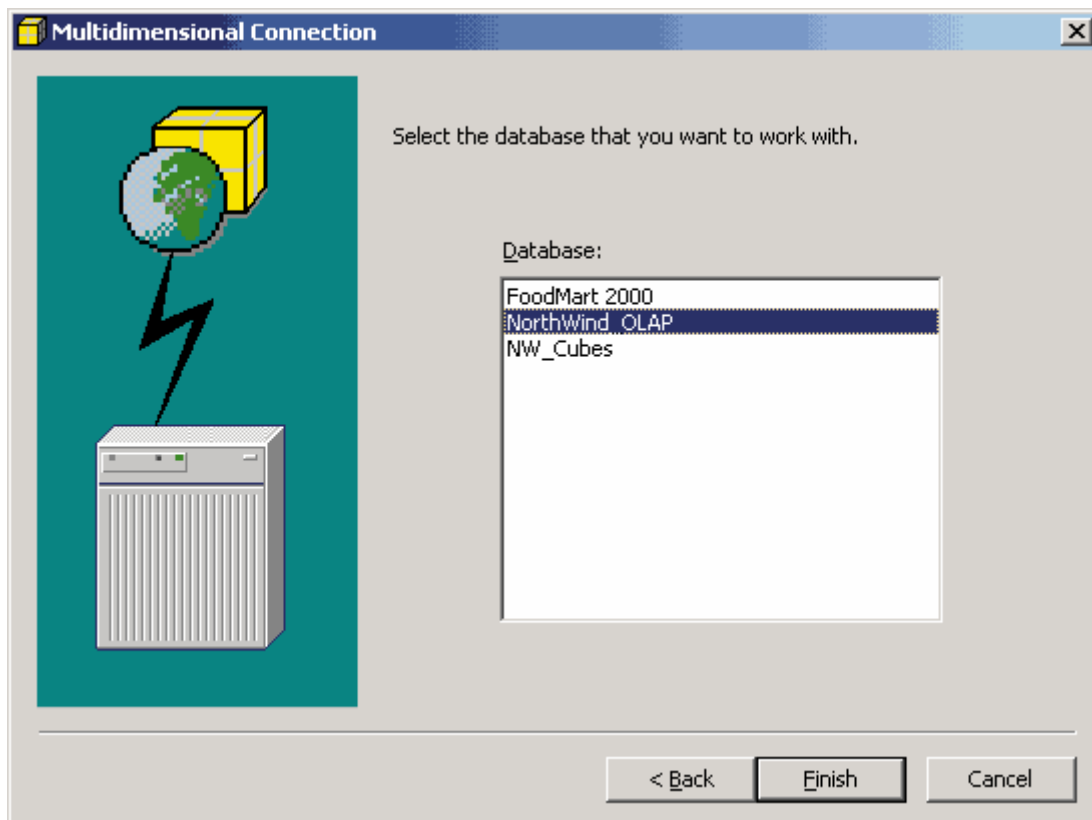
*Рис. 2. Выбор провайдера данных*

В диалоговой панели Multidimensional Connection укажем имя компьютера (если это локальный компьютер, можно использовать имя localhost), на котором расположен OLAP-сервер, а также данные для аутентификации пользователя, которые понадобятся только в том случае, если для связи с OLAP-сервером мы используем HTTP-протокол (рис. 3).



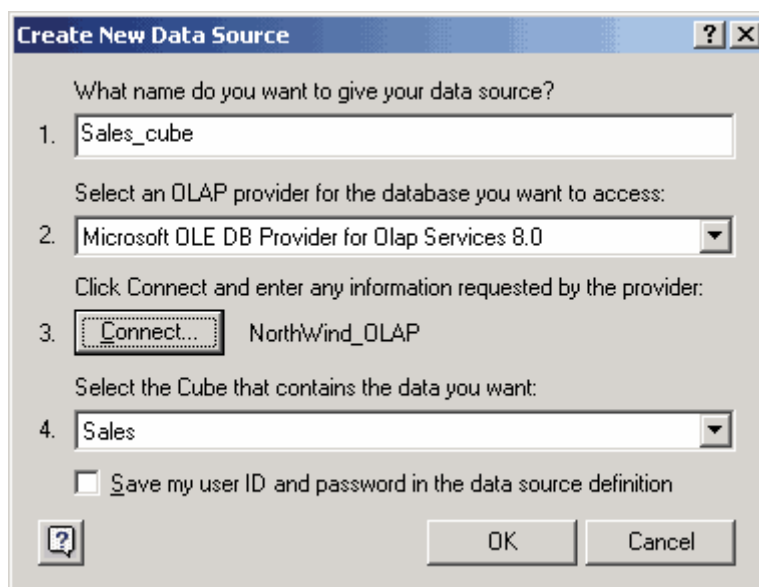
*Рис. 3. Выбор OLAP-сервера*

И наконец, выберем имя многомерной базы данных, в которой хранится OLAP-куб (рис. 4).



*Рис. 4. Выбор многомерной базы данных*

Определив источник данных, выберем куб, который мы будем отображать в сводной таблице (рис. 5).



*Рис. 5. Выбор куба для отображения в сводной таблице*

После этого можно нажать кнопку ОК. В результате мы получим пустую сводную таблицу, вид которой в Excel 2000 показан на рис. 6.

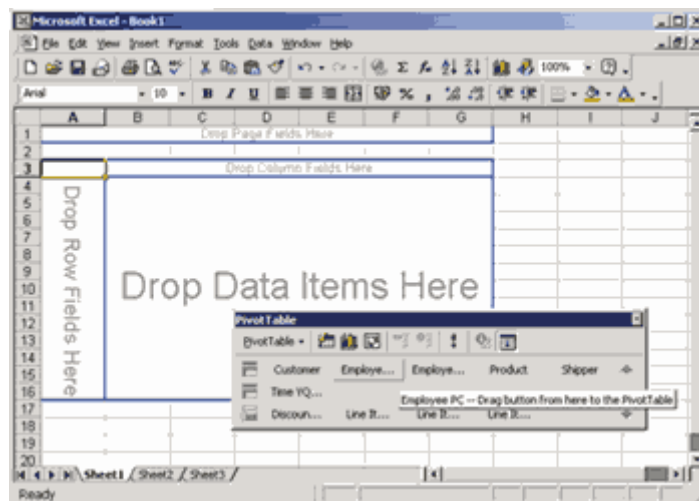


Рис. 6. Сводная таблица в Excel 2000

Для дальнейших манипуляций нам потребуется панель инструментов PivotTable. В случае с Excel 2000 пользоваться ею удобнее, если она не закреплена у края окна Excel, а свободно перемещается по экрану, в противном случае некоторые нужные нам элементы этой панели окажутся недоступны.

Следует отметить, что, когда фокус ввода находится на самой сводной таблице (для чего достаточно щелкнуть по ней мышью), панель PivotTable в Excel 2000 содержит кнопки с названиями измерений и мер куба. Отметим, что они обозначаются пиктограммами разного вида и, если их названия не умещаются на кнопке, их можно увидеть на всплывающих подсказках.

При смещении фокуса ввода в другое место листа эти кнопки исчезают.

В Excel 2002 диалоговая панель PivotTable выглядит иначе — она не содержит кнопок с именами измерений и мер. Их список предоставляется в отдельной панели PivotTable Field List (рис. 7).

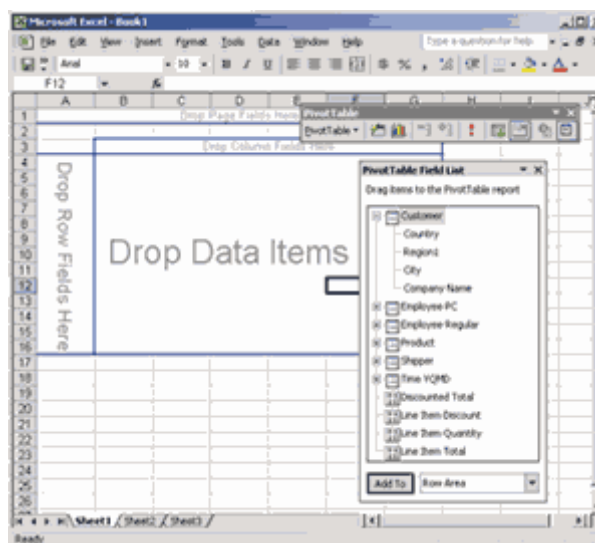


Рис. 7. Сводная таблица в Excel 2002

Теперь нам необходимо определить, какие из мер мы хотим отобразить в сводной таблице. Для этого достаточно перенести мышью кнопку (в случае Excel 2002 — соответствующий элемент из списка) с наименованием нужной меры в область данных (Data

Area; на рис. 7 она обозначена надписью Drop Data Items Here). Результат этой манипуляции представлен на рис. 8.

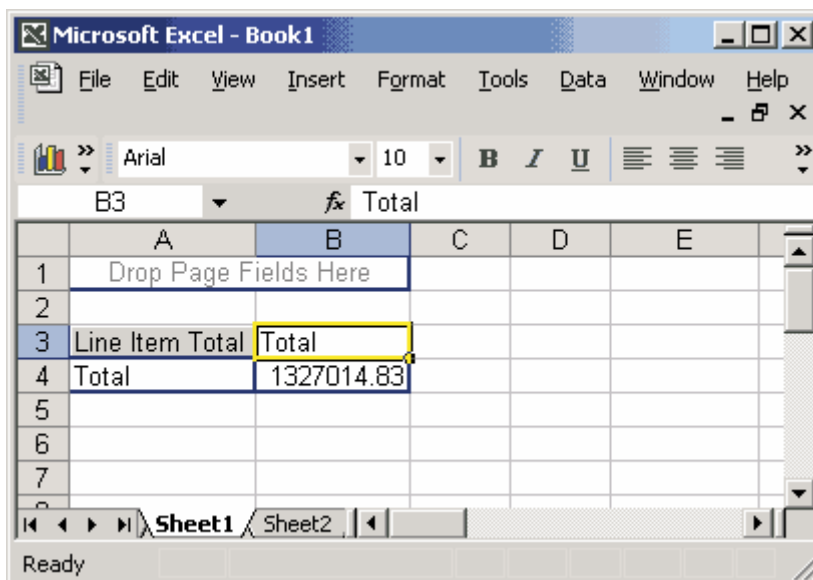


Рис. 8. Выбор меры для отображения в сводной таблице

Теперь требуется определить, какие из полей будут участвовать в формировании строк, столбцов и страниц (иногда последние называются фильтрами). В общем случае сводная таблица является трехмерной, и можно считать, что третье измерение расположено перпендикулярно экрану, а мы наблюдаем сечения, параллельные плоскости экрана и определяемые тем, какая «страница» выбрана для отображения. Осуществить фильтрацию можно путем перетаскивания мышью соответствующих кнопок с панели инструментов PivotTable (в случае Excel 2002 — соответствующих элементов с панели PivotTable Field List) на области строк, столбцов и страниц сводной таблицы — Row Area, Column Area и Page Area. Результат этой манипуляции показан на рис. 9.

	A	B	C	D	E
1	Employee PC	All Employee			
2					
3	Line Item Total	Year			
4	Category Name	1996	1997	1998	Grand Total
5	Beverages	52203.2	106545	119634.75	269582.95
6	Condiments	19312.7	59588	33606.05	112506.75
7	Confections	26616.2	85682.19	59997.51	174295.9
8	Dairy Products	40330.4	122970.2	84465.9	247766.5
9	Grains/Cereals	9529.6	59275.2	26955	95759.8
10	Meat/Poultry	25630.2	87648.12	63917.48	177195.8
11	Produce	13589.8	55549.8	29419.95	98559.55
12	Seafood	20900.2	69780.3	49667.08	140347.58
13	Grand Total	210112.3	649038.81	467863.72	1327014.83

Рис. 9. Готовая сводная таблица

Итак, мы отобразили в сводной таблице Excel содержимое OLAP-куба. Теперь этим отображением можно манипулировать.

## Манипуляция отображением данных в сводной таблице

Если нас интересуют более подробные данные, связанные с одним из членов одного из отображаемых измерений, можно дважды щелкнуть по ячейке с этим значением и отобразить



члены следующего уровня данного измерения (эта операция называется drill-down). То, что получится, если дважды щелкнуть на ячейке A5, показано на рис. 10.

Line Item Total	Product Name	1996	1997	1998	Grand Total
Beverages	Chai	1000	5295.6	6462	13657.6
	Chang	3055.2	7600	6726	17381.2
	Chateau verte	3030.4	4920.4	4356	13114.8
	Côte de Blaye	29512	49327.2	71145	149984.2
	Guaraná Fantástica	560.8	1666.8	2412	4640.6
	Ispoh Coffee	5004.8	11518.4	6900	23423.2
	Lakkakikōn	1598.4	8211.6	6948	16758
	Laughing Lumberjack Lager	56	910	1316	2282
	Outback Lager	1512	6240	3540	11292
	Rhinobrau Klosterbier	744	4457.8	3417.75	8619.55
	Sasquatch Ale	1008	2100	2310	5418
	Steelhead Stout	3513.6	6289.2	4302	14104.8
Beverages Total		52203.2	108545	119034.75	289582.95
Condiments		19312.7	59598	33606.05	112516.75
Confections		28616.2	85682.19	59997.51	174295.9
Dairy Products		40330.4	122970.2	84465.9	247766.5
Grains/Cereals		9529.6	59275.2	26955	95760
Meat/Poultry		25630.2	87648.12	63917.40	177195.7
Produce		13589.8	55549.8	29419.95	98559.55
Seafood		20900.2	69780.3	49667.05	140347.55
Grand Total		210112.3	649038.81	462963.72	1327014.83

Рис. 10. Результат операции drill-down

Если же нас интересуют более подробные данные, нежели представленные в данный момент в сводной таблице, следует выбрать ячейку с именем соответствующего измерения (например, ячейку A4) и нажать на панели инструментов PivotTable кнопку Show Detail (рис. 11).

Line Item Total	Product Name	1996	1997	1998	Grand Total
Condiments	Amazón Syrup	240	1762	1542	3544
	Chef Antonio's Cajun Seasoning	1883.2	5737.6	1782	9402.8
	Chef Antonio's Gumbo Mix	2193	405.65	3002.5	5801.15
	Green Shoyu	310	1503.5	3600	5413.5
	Grandma's Boysenberry Spread	720	2500	3600	6820
	Guia Mielace	2139	5987.65	1487.65	10614.3
	Louisiana Fire Hot Pepper Sauce	2604	9855.9	2147.1	14607
	Louisiana Hot Spiced Okra	408	3094		3502
	Northwest's Cranberry Sauce	4480	4590	4640	13710
	Original Frankfurt grüne Soße	509.6	5327.4	3450	9287
	Simp. diable		10539.3	5785.5	16324.8
	Vierge spread	3826.9	7417.1	6453.3	17700.3
Condiments Total		19312.7	59598	33606.05	112516.75
Confections	Chocolate		1440.75	52	1542.75
	Gourmet Gumbarchen	2241	12648.54	7245.36	21534.9

Рис. 11. Отображение следующего уровня иерархии измерения

При необходимости можно вручную определить, какие члены измерения должны быть отображены в сводной таблице; для этого можно нажать кнопку вывода соответствующего выпадающего списка в правой части ячейки с именем измерения (рис. 12).

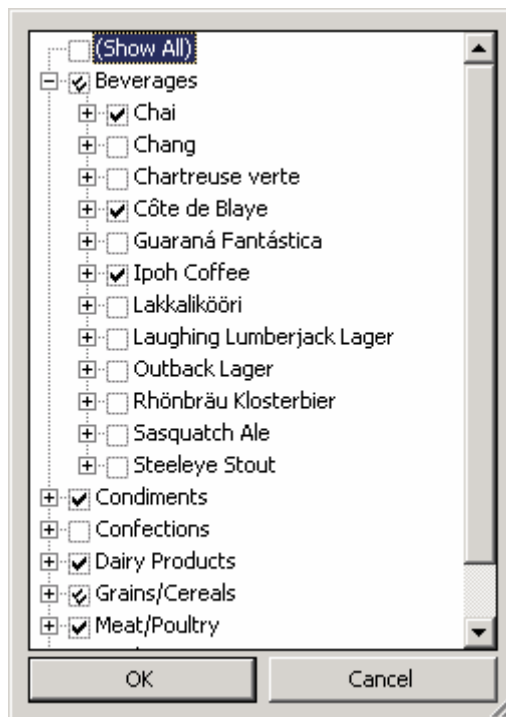


Рис. 12. Выбор отображаемых членов измерения

Если в сводной таблице отображается несколько мер, они формируют отдельное дополнительное измерение Data. По умолчанию оно располагается на оси строк, но может быть перенесено и на ось столбцов (рис. 13).

Category Name	Line Item Total	Discounted Total	1997	1998	Grand Total
Beverages	Line Item Total	Discounted Total	52203.2	108649	119854.75
Condiments	Line Item Total	Discounted Total	19912.7	59500	33606.05
Confections	Line Item Total	Discounted Total	28616.2	85882.19	89997.51
Dairy Products	Line Item Total	Discounted Total	40330.4	122870.2	84465.5
Grains/Cereals	Line Item Total	Discounted Total	9529.6	58275.2	26955
Meat/Poultry	Line Item Total	Discounted Total	25630.2	87648.12	62617.48
Produce	Line Item Total	Discounted Total	12651.18	53019.8875	21958.65
Total Line Item Total			240112.3	649030.81	867063.72
Total Discounted Total			193216.56	808646.3749	419557.1741

Рис. 13. Отображение нескольких мер в сводной таблице

Если в сводной таблице оставить только одну меру, перенеся оставшиеся обратно на панель инструментов PivotTables, измерение Data исчезнет.

Отметим, что с помощью одного из доступных в Excel шаблонов оформления можно изменить оформление сводной таблицы. Кроме того, можно выбрать на панели инструментов PivotTables пункты меню PivotTable | Table Options или PivotTable | Field Settings и изменить другие параметры отображения данных в сводной таблице.

Применяя Excel в качестве OLAP-клиента, следует помнить, что объем данных, отображаемых в сводной таблице, ограничен — ведь все эти данные хранятся в оперативной памяти клиентского компьютера.

## Создание сводных диаграмм с данными OLAP-кубов

При необходимости в Excel можно построить сводную диаграмму, синхронизированную со сводной таблицей. Для этого достаточно нажать соответствующую кнопку на панели инструментов PivotTables и, если нужно, отредактировать внешний вид диаграммы (рис. 14).

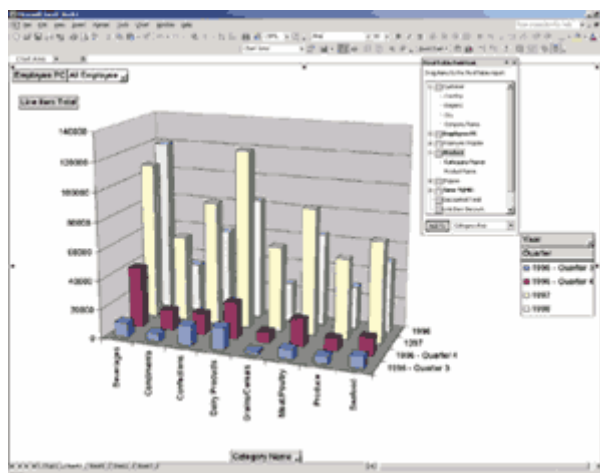


Рис. 14. Сводная диаграмма с данными OLAP-куба

Отметим, что с помощью панелей инструментов PivotTable и PivotTable FieldList, а также выпадающих списков на осях и легенде можно управлять отображением данных на сводной диаграмме, например выполнять операцию drill-down; при этом сводная таблица будет меняться синхронно с диаграммой.

## Создание локальных OLAP-кубов

Как уже было отмечено выше, Microsoft Excel позволяет создавать локальные OLAP-кубы, представляющие собой подмножества данных серверных OLAP-кубов. Локальные кубы хранятся в файлах с расширением \*.cub. Напомним, что для корректного создания локального куба на основе серверного куба, содержащего несбалансированные измерения, рекомендуется применять Microsoft Excel 2002. Поэтому все последующие примеры выполнены в этой версии Microsoft Excel.

Чтобы создать локальный OLAP-куб на основе серверного куба, следует на панели инструментов PivotTables выбрать пункт меню PivotTable | Offline OLAP в Excel 2002 (в Excel 2000 ему соответствовал пункт меню PivotTable | Client-Server Settings) и нажать кнопку Create offline data file (рис. 15; в Excel 2000 — Create Local Cube).

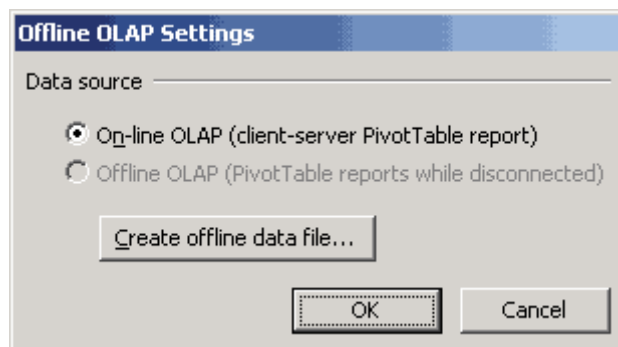


Рис. 15. Диалоговая панель Offline OLAP Settings

Далее следует выбрать измерения и их уровни, а также меры, которые будут присутствовать в локальном кубе (рис. 16).

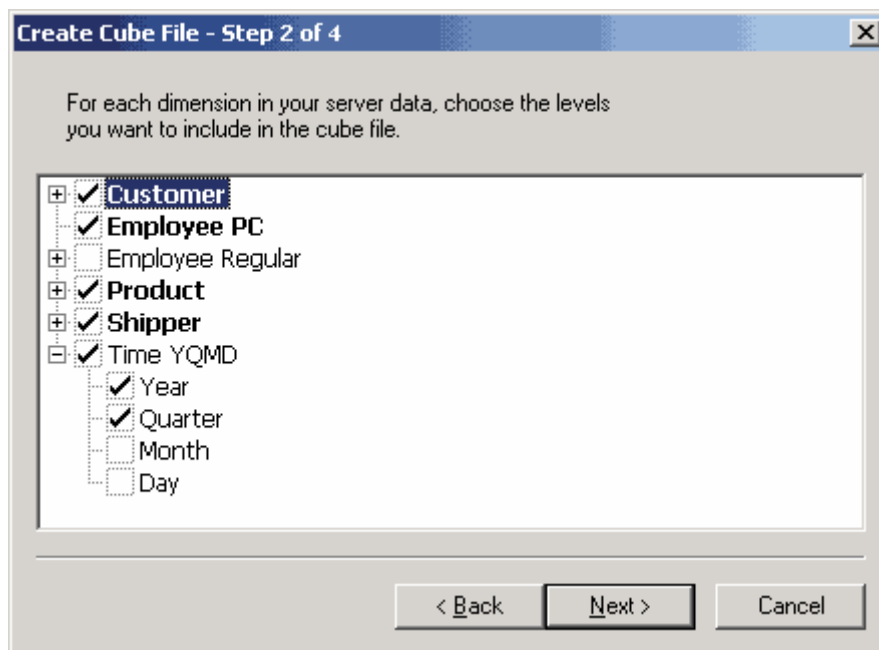


Рис. 16. Выбор измерений и мер для локального куба

Помимо выбора измерений, их уровней и мер можно внести и другие ограничения в набор данных, который будет содержаться в локальном кубе, выбрав набор членов изменений, участвующих в его формировании (рис. 17).

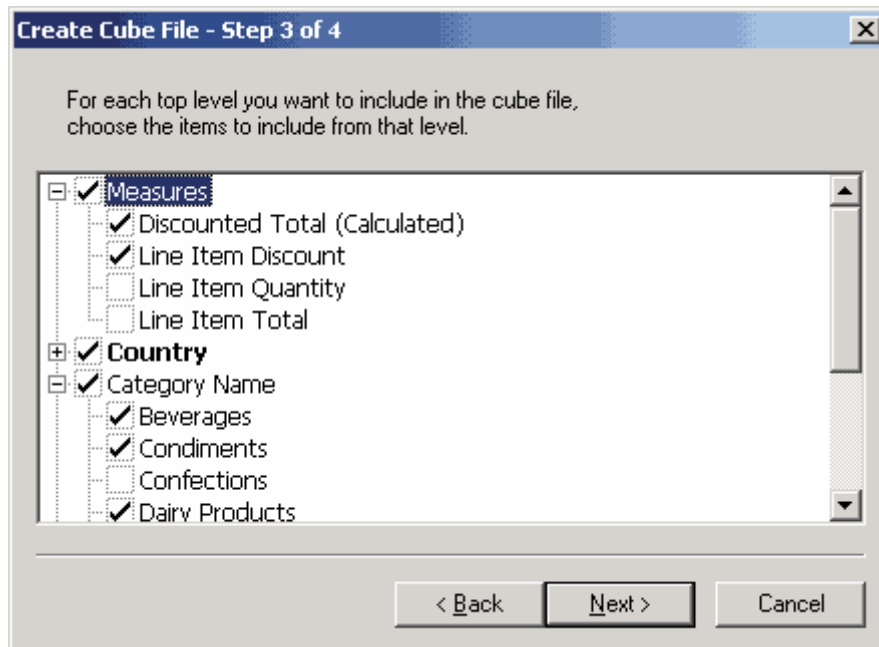


Рис. 17. Выбор членов измерений для локального куба

Теперь осталось только сохранить локальный куб в файле с расширением \*.cub. Отметим, что этот файл является отчуждаемым: его можно просматривать на любом компьютере, оснащенном как Microsoft Excel 2002, так и Microsoft Excel 2000, независимо от наличия на нем Microsoft SQL Server Analysis Services или их клиентской части.

## Часть 7. Применение компонента PivotTable List для отображения OLAP-данных

Публикация сводных таблиц на Web-страницах  
Возможности компонента PivotTable List  
Создание Web-страниц со сводными диаграммами

В предыдущей статье данного цикла мы рассмотрели один из простейших способов работы с OLAP-данными — использование Microsoft Excel в качестве OLAP-клиента. На этот раз мы поговорим о просмотре OLAP-данных с помощью компонента PivotTable List — элемента управления ActiveX, входящего в состав Microsoft Office Web Components и предназначенного для создания сводных таблиц и просмотра сечений OLAP-кубов. Этот компонент применяется главным образом на Web-страницах, а иногда и в обычных Windows-приложениях.

Для выполнения примеров данной статьи нам потребуется созданная ранее сводная таблица Microsoft Excel 2002, содержащая сечение созданного ранее OLAP-куба.

### Публикация сводных таблиц на Web-страницах

Самый простой способ воспользоваться компонентом PivotTable List — сохранить сводную таблицу Microsoft Excel как Web-страницу. Для этого выберем в Microsoft Excel пункт меню File | Save As Web Page, в появившейся диалоговой панели нажмем кнопку Publish, в диалоговой панели выберем из выпадающего списка Choose опцию Items on Sheet1, затем — PivotTable, отметим опцию Add interactivity with и выберем из списка PivotTable functionality (рис. 1).

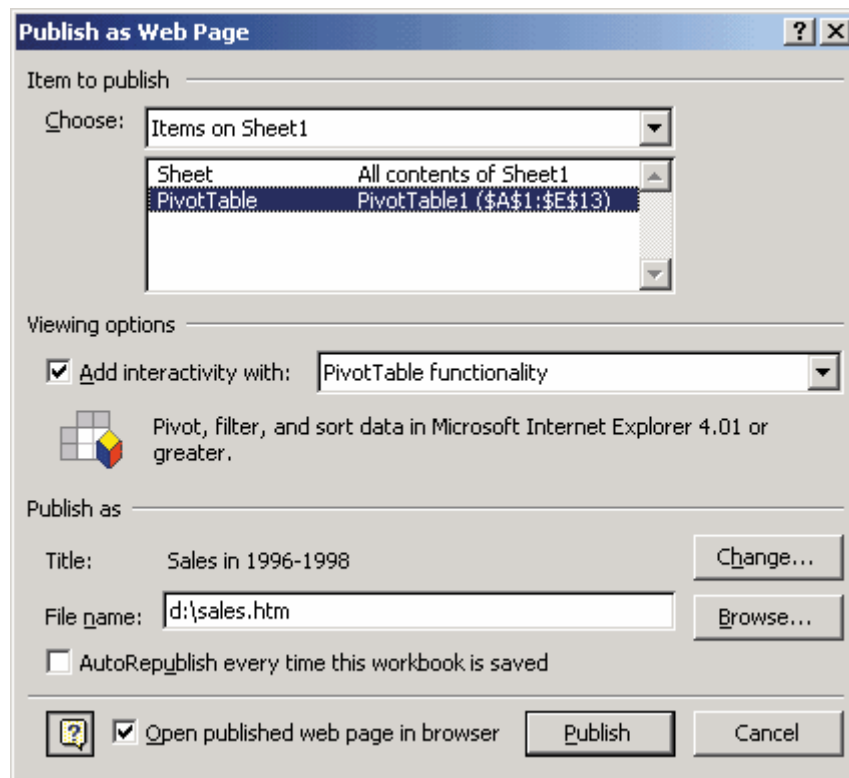


Рис. 1. Публикация сводной таблицы Excel на Web-странице

Далее при необходимости изменим заголовок, который появится на будущей Web-странице, и сохраним ее. Если открыть эту страницу в Microsoft Internet Explorer версии 4.01

или выше, мы увидим, что она содержит PivotTable List — элемент управления ActiveX, предназначенный для просмотра OLAP-данных и сводных таблиц на Web-страницах или в Windows-приложениях (рис. 2).

Category Name	1996		1997		1998		Grand Total
	Line Item Total	Line Item Total	Line Item Total	Line Item Total	Line Item Total	Line Item Total	
Beverages	52203.2	100545	119834.75	292562.95			
Condiments	19912.7	59580	33606.05	112506.75			
Confections	28616.2	85682.19	69997.51	174295.9			
Dairy Products	40330.4	122970.2	84465.9	247766.5			
Grains/Cereals	9529.6	59275.2	26955	95759.8			
Meat/Poultry	25630.2	87648.12	63917.48	177195.8			
Produce	13589.8	55549.8	25419.95	98559.55			
Seafood	20900.2	69780.3	49667.08	140347.58			
Grand Total	210112.3	649038.81	467863.72	1327014.83			

Рис. 2. Web-страница с компонентом PivotTable List

Сразу же заметим, что этот элемент управления можно применять только в локальных сетях на компьютерах, для которых приобретена лицензия на Microsoft Office; другие способы его применения, например на Web-страницах, доступных в Интернете, запрещены лицензионным соглашением.

## Возможности компонента PivotTable List

В этом разделе мы кратко рассмотрим возможности, предоставляемые компонентом PivotTable List.

Пользователь, манипулирующий этим компонентом в браузере или в Windows-приложении, может, как и в сводной таблице Excel, перемещать данные в область строк, столбцов и страниц (в Microsoft Office Web Components приняты термины Row Area, Column Area и Filter Area) с диалоговой панели, напоминающей панель PivotTable Field List из Excel 2002. Диалоговая панель со списком измерений и мер выводится на экран по нажатию кнопки Field List на инструментальной панели компонента PivotTable List (рис. 3).

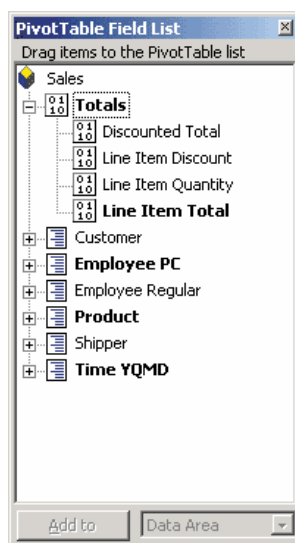


Рис. 3. Диалоговая панель PivotTable Field List

Пользователь может также выполнять операцию drill-down, щелкая мышью на значках «+» (рис. 4).

Sales in 1996-1998

Category Name	Year				Grand Total
	1996	1997	1998		
<b>Beverages</b>	62273.2	109645	118634.75	290553.95	
<b>Confections</b>	19312.7	69608	33600.05	112520.75	
<b>Dairy Products</b>	40330.4	122970.2	84455.9	247756.5	
<b>Groceries</b>	9628.6	69275.2	26955	95759.8	
<b>Meat/Produce</b>	25630.2	87648.12	63917.48	177195.8	
<b>Produce</b>	13660.8	55649.8	29412.05	98522.65	
<b>Seafood</b>	2000.2	67730.3	48627.00	140347.50	
<b>Grand Total</b>	210112.3	649030.51	457862.22	1320143.83	

Рис. 4. Операция Drill-Down в компоненте PivotTable Field List

Компонент PivotTable List позволяет сортировать и фильтровать данные. Во-первых, фильтрация данных может быть осуществлена с помощью отображения только выбранных членов измерений, которые могут быть отмечены в выпадающем списке, сходном с соответствующим списком Excel (рис. 5).

**Commands and Options**

**Filter and Group**

**Filtering**

Display the: **Top**

Items: **5** %

Based on: **Line Item Total**

☒ Allow selecting multiple items when in filter area

**Grouping**

Group items by: **(No grouping)**

Interval:

☐ Start at: **(Automatic)**

☐ End at: **(Automatic)**

Рис. 5. Выбор членов измерения в компоненте PivotTable Field List

Во-вторых, с помощью диалоговой панели Commands and Options (ее можно вывести на экран с помощью соответствующей кнопки инструментальной панели компонента PivotTable List) можно выбрать способы фильтрации и группировки данных (например, выводить определенное количество наибольших или наименьших значений — Top 5, Top 10, Bottom 25 и т.п.; рис. 6).

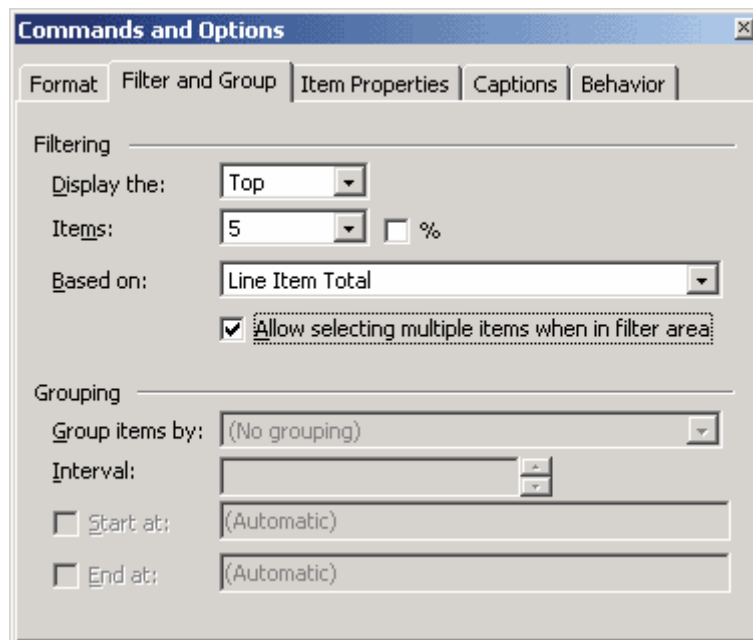
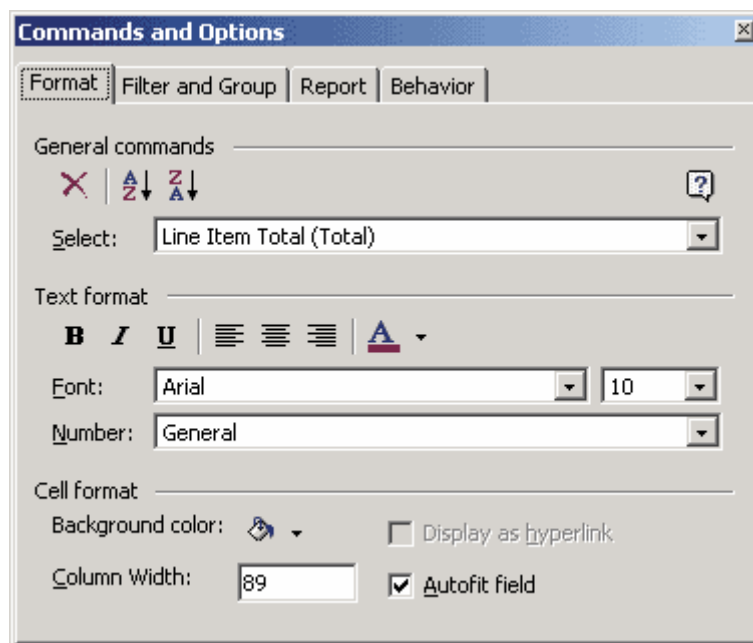


Рис. 6. Установка правил фильтрации и группировки данных

Помимо этого пользователь может изменять атрибуты отображения данных — цвет и шрифт текста, цвет фона, выравнивание текста, отображение и т.д. Для этого достаточно поместить курсор на один из элементов данных, атрибуты которых нужно изменить (например, на наименование члена измерения, на ячейку с суммарными данными или с итоговыми значениями), и выбрать новые атрибуты отображения данных этого типа в той же диалоговой панели Commands and Options (рис. 7).





**Sales in 1996-1998**

		Year - Quarter				
		1996		1997		1998
		Line Item	Total	Line Item	Total	Line Item
Category Name	Product Name	Line Item	Total	Line Item	Total	Line Item
Beverages		52203.2	106645	119334.75	289582.95	
Confections		19312.7	69688	33906.05	112506.75	
Confections	Gumbär Gumbärchen	2241	12048.54	7245.36	21534.9	
Confections	Pavlova	3832.8	7916.25	6526.3	17985.35	
Confections	Schoggi Schokolade	1404	10874	2853.5	15231.5	
Confections	Sir Rodney's Marmalade	6968.8	7776	8910	23504.8	
Confections	Tarte au sucre	9456	21736.5	18635.4	49827.9	
Confections	Total *	28616.2	65602.19	59957.51	174295.9	
Dairy Products		40330.4	122870.2	84465.9	247766.5	
Grains/Cereals		9629.8	69275.2	26965	95759.8	
Meat/Poultry		25620.2	87648.12	63917.40	177495.8	
Produce		13689.8	56549.8	29419.95	98559.55	
Seafood		20900.2	68780.3	49667.08	148347.58	
Grand Total *		210172.3	649038.81	467803.72	1327836.83	

Рис. 7. Установка атрибутов отображения данных

Помимо этого компонент PivotTable List позволяет на основе агрегатных данных вычислять доли или проценты общей суммы или суммы, соответствующей родительскому члену измерения (например, процент от годовой прибыли, полученный в данном квартале), — соответствующие опции можно найти в контекстных меню элементов данных.

Пользователю также доступен специально предназначенный для него файл справки (на русском языке, если используются Web-компоненты из комплекта поставки русской версии Microsoft Office XP). Однако пользователь не может изменить источник данных и отобразить на Web-странице другой OLAP-куб, поскольку право сделать это есть только у разработчика Web-страницы (и для него имеется отдельный файл справки, существенно отличающийся от того, что предназначен для пользователя, — он содержит, в частности, сведения об объектной модели этого компонента).

Отметим, что подобную Web-страницу можно создать и с помощью Microsoft FrontPage. Для вставки PivotTable List в Web-страницу, создаваемую в FrontPage, следует выбрать пункт меню Insert | Web component и в появившейся диалоговой панели выбрать Office PivotTable из раздела Spreadsheets and Charts (рис. 8).

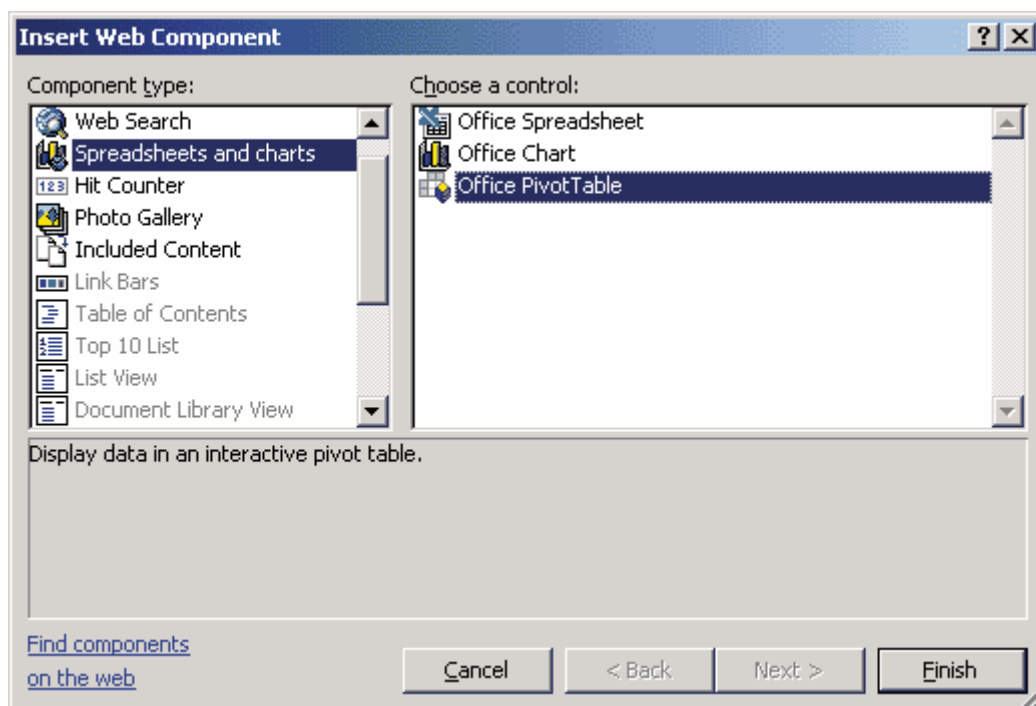


Рис. 8. Выбор компонента PivotTable List в Microsoft FrontPage

После появления компонента PivotTable List на Web-странице следует щелкнуть мышью на гипертекстовой ссылке, предлагающей определить источник данных, а затем выбрать ODBC-источник из предлагаемого списка (либо описать его, если он еще отсутствует в списке; как это сделать, было рассказано в предыдущей статье данного цикла). В качестве источника данных можно использовать как серверный OLAP-куб, так и локальный, созданный с помощью Excel (а также результат запроса к любому ODBC-источнику, возвращающего обычный «плоский» набор данных). И наконец, в случае необходимости можно вывести на экран диалоговую панель PivotTable Field List и перенести имена измерений и мер в соответствующие области этого компонента (рис. 9).

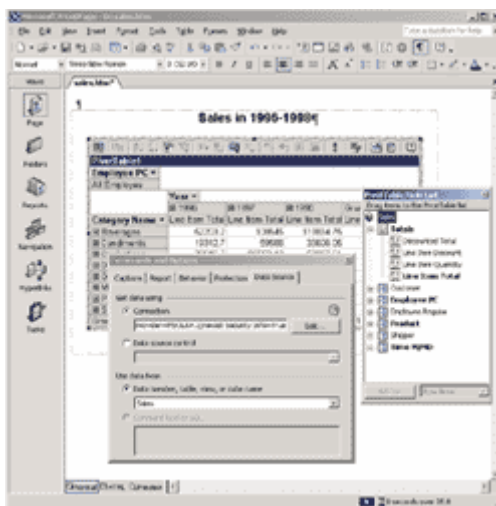


Рис. 9. Создание Web-страницы с компонентом PivotTable List с помощью Microsoft FrontPage

Отметим, что страница Data Source диалоговой панели Commands and Options доступна только на этапе разработки (то есть в FrontPage или, если компонент PivotTable List используется не на Web-странице, а в Windows-приложении, то в средстве разработки, с помощью которого создается это приложение). Иными словами, конечный пользователь не имеет возможности изменить источник данных, это может сделать только разработчик.

Теперь созданную нами страницу можно сохранить — она готова к применению.

## Создание Web-страниц со сводными диаграммами

Microsoft Office Web Components позволяют построить и сводную диаграмму на основе данных, отображенных в компоненте PivotTable List. Для этой цели применяется элемент управления ChartSpace, также входящий в комплект поставки Microsoft Office Web Components. Чтобы поместить его на Web-страницу, следует из меню FrontPage выбрать пункт меню Insert | Web component и в появившейся диалоговой панели выбрать Office Chart из раздела Spreadsheets and Charts.

Следующий этап создания диаграммы заключается в выборе источника данных для ее построения. В нашем случае это будет уже имеющийся компонент PivotTable List (рис. 10).



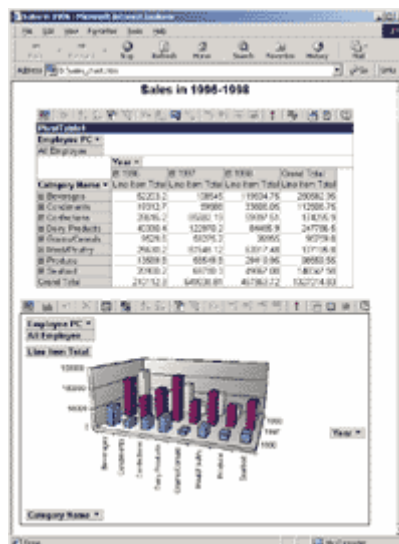


Рис. 12. Web-страница с компонентами PivotTable List и ChartSpace в Microsoft Internet Explorer

Сводную диаграмму на основе OLAP-куба можно построить и непосредственно с помощью компонента ChartSpace. Для этого в процессе создания Web-страницы следует описать источник данных на странице Data Details в диалоговой панели Commands and Options (рис. 13).

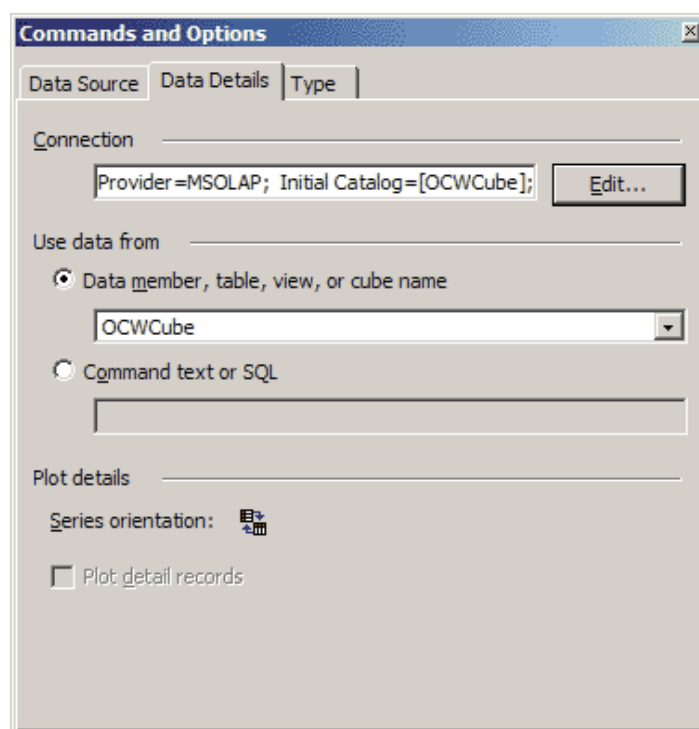


Рис. 13. Установка свойств источника данных для компонента ChartSpace

И наконец, еще один способ создания Web-страницы со сводной диаграммой. Он заключается в сохранении в виде Web-страницы сводной диаграммы Excel. Однако в этом случае на эту же страницу будет автоматически добавлен компонент PivotTable List, связанный с создаваемой диаграммой.

Как уже было сказано выше, компоненты PivotTable List и ChartSpace можно применять и в приложениях. Для этого потребуется средство разработки, поддерживающее применение элементов управления ActiveX на формах (например, Microsoft Visual Basic, Microsoft Visual C++, Borland Delphi, Borland C++Builder).

## Часть 8. Обзор MDX

### Язык MDX

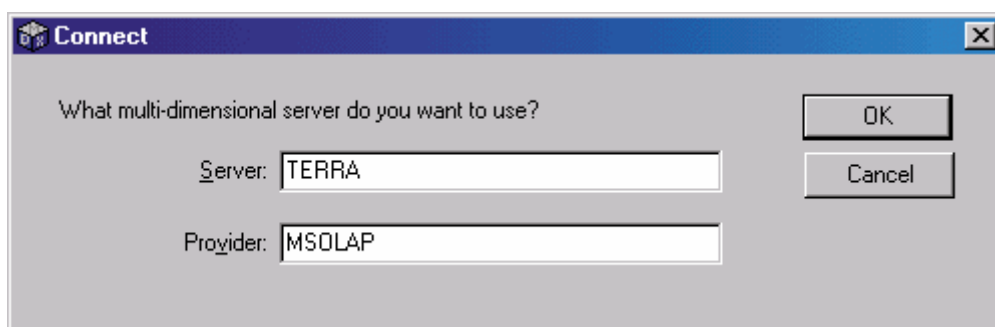
#### Использование языка MDX

##### MDX Sample Application

##### Функции языка MDX

Предыдущая часть данной статьи была посвящена просмотру OLAP-данных с помощью компонента PivotTable List. Как вы помните, данный элемент управления ActiveX входит в состав Microsoft Office Web Components и позволяет создавать сводные таблицы, просматривать сечения OLAP-кубов, а также строить простейшие сечения многомерных кубов. Однако запросы к кубам не всегда могут быть представлены в виде простейших сечений.

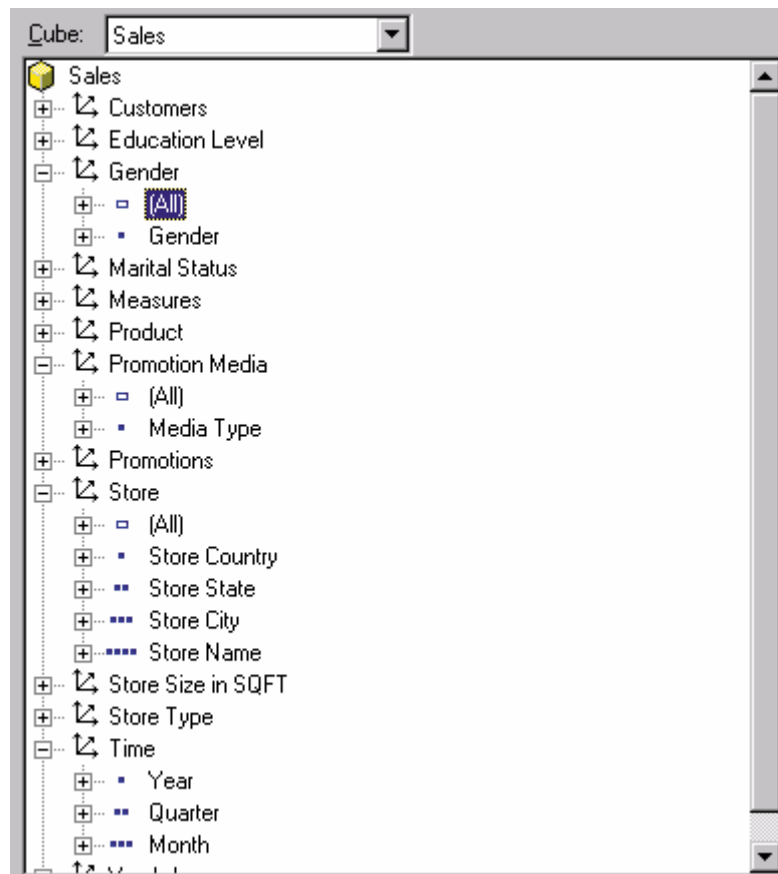
На практике нередко требуется получить сведения о конкретном подмножестве данных (например, узнать о заказах только в одной стране или сравнить данные о продажах только в конкретном месяце, но в разные годы) и отобразить не просто суммарные значения, а какие-то другие статистические данные (например, среднеквадратичное отклонение). Далеко не всегда подобные срезы данных можно получить, просто манипулируя имеющимся кубом в рассмотренных нами OLAP-клиентах (равно как не всегда можно получить нужные сведения из реляционной базы данных, просто обратившись к одной таблице с помощью простейшей утилиты просмотра данных).



Практически во всех промышленных OLAP-средствах, включая и Analysis Services, для получения нестандартных срезов данных требуется отдельный непроцедурный язык для формулирования запросов к многомерным базам данных. Одному из таких языков, MDX, посвящена данная статья.

### Язык MDX

Назначение языка MDX (Multidimensional Expressions) — предоставить в распоряжение разработчиков средство для более простого и эффективного доступа к многомерным структурам данных. В Microsoft SQL Server 2000 Analysis Services язык MDX используется для формирования запросов и описания алгоритмов получения вычисляемых значений.



Следует сказать, что язык MDX никак не связан с Microsoft SQL Server 2000 Analysis Services, а является частью спецификации OLE DB for OLAP и, таким образом, поддерживается на уровне провайдера доступа к данным (OLE DB-провайдера), а не самого OLAP-хранилища. Этот язык можно сравнить с языком SQL. Но если SQL используется для извлечения реляционных данных, то MDX служит для извлечения многомерных данных. Естественно, что, как и в случае с языком SQL, возможны некоторые отклонения от стандарта. В этой статье мы рассмотрим язык MDX применительно к Microsoft SQL Server 2000 Analysis Services.

## Использование языка MDX

### MDX Sample Application

Для выполнения запросов на языке MDX мы будем использовать утилиту MDX Sample Application, входящую в состав Microsoft SQL Server 2000 Analysis Services. При запуске этой утилиты появляется диалоговая панель Connect, в которой следует указать имя сервера (имя компьютера, на котором установлен Microsoft SQL Server 2000 Analysis Services) и тип провайдера для связи с этим сервером — в нашем примере это будет MSOLAP.

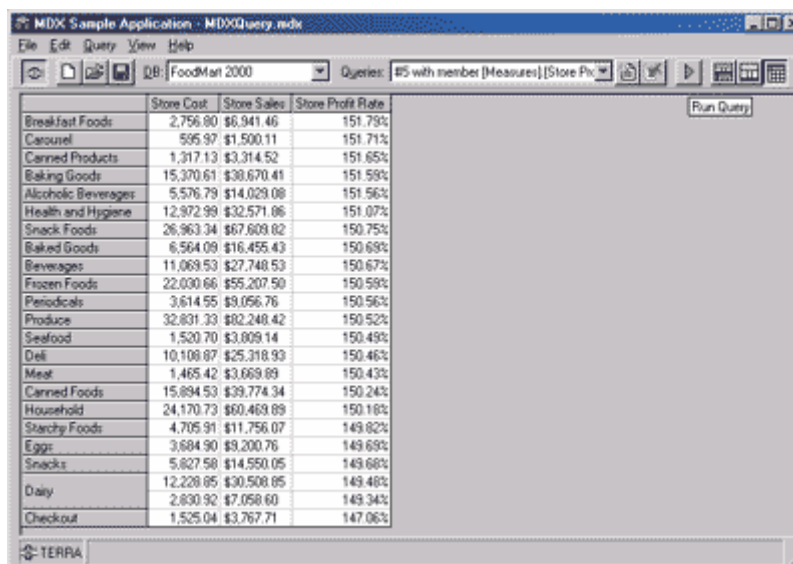


После этого можно нажать кнопку ОК. Нажатие кнопки Cancel отменяет данную диалоговую панель — в этом случае для связи с сервером следует воспользоваться командой Connect из меню File.

После соединения с сервером можно начинать создание MDX-запросов к многомерной базе данных и их выполнение. Часто бывает полезно знать структуру куба, к которому вы собираетесь обратиться. Утилита MDX Sample Application позволяет просматривать метаданные куба. Для этого необходимо выбрать интересующий вас куб в списке Cube и изучить его размерности и меры в древовидном представлении содержимого куба.

Верхняя панель утилиты MDX Sample Application предназначена для задания MDX-запросов. Мы можем либо выбрать один из predefined запросов (список Query), либо создать собственный. По умолчанию утилита использует запросы, находящиеся в файле MDXQuery.mdx, расположенном в папке MDXSample. Для загрузки другого файла или сохранения текущего используются команды меню File.

Можно вводить MDX-команды непосредственно в панели запросов или конструировать запрос, перетаскивая измерения и меры куба в панель запросов. Помимо этого можно использовать примеры функций из панели Syntax Examples. В этом случае в панель переносится копия примера, в котором следует изменить аргументы функций и разделители на нужные значения.



The screenshot shows the MDX Sample Application window with the title bar 'MDX Sample Application - MDXQuery.mdx'. The menu bar includes 'File', 'Edit', 'Query', 'View', and 'Help'. The 'Query' menu is open, showing a list of queries. The 'Queries' list includes 'F5 with member [Measures] [Store Profit Rate]'. The 'Run Query' button is visible. The main area displays a table with the following data:

	Store Cost	Store Sales	Store Profit Rate
Breakfast Foods	2,756.80	\$6,941.46	151.73%
Carourel	595.97	\$1,500.11	151.71%
Canned Products	1,317.13	\$3,314.52	151.65%
Baking Goods	15,370.61	\$38,670.41	151.58%
Alcoholic Beverages	5,576.79	\$14,029.08	151.56%
Health and Hygiene	12,972.99	\$32,571.06	151.07%
Snack Foods	26,963.34	\$67,609.02	150.75%
Baked Goods	6,564.09	\$16,455.43	150.68%
Beverages	11,069.53	\$27,748.53	150.67%
Frozen Foods	22,030.66	\$55,207.50	150.58%
Periodicals	3,614.55	\$9,056.76	150.56%
Produce	32,831.33	\$82,248.42	150.52%
Seafood	1,530.70	\$3,809.14	150.49%
Deli	10,108.87	\$25,318.93	150.46%
Meat	1,465.42	\$3,663.89	150.43%
Canned Foods	15,894.53	\$39,774.34	150.24%
Household	24,170.73	\$60,469.89	150.16%
Starchy Foods	4,705.91	\$11,756.07	149.82%
Eggs	3,684.90	\$9,200.76	149.69%
Snacks	5,827.58	\$14,550.05	149.68%
Dairy	12,228.85	\$30,508.85	149.48%
Daily	2,830.92	\$7,058.60	149.34%
Checkout	1,525.04	\$3,767.71	147.06%

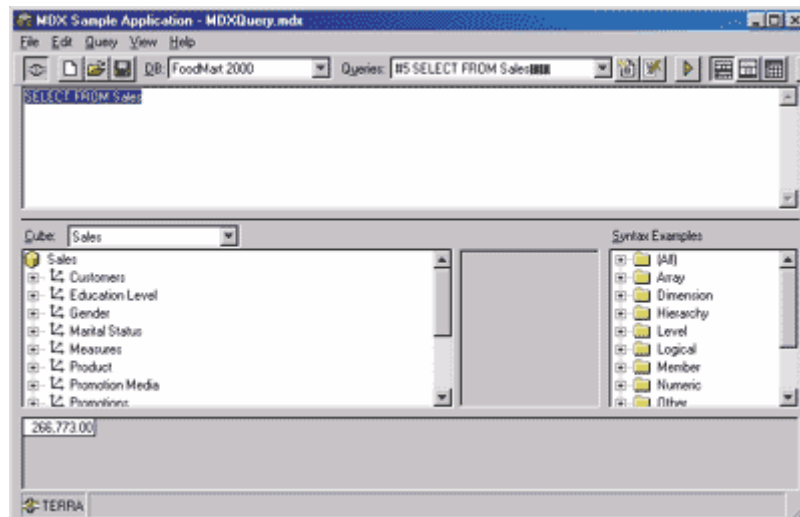
*Выполнить запрос можно одним из трех способов:*

- выбрав команду Run в меню Query;
- нажав клавиши F5;
- нажав кнопку Run Query на панели инструментов.

Результат выполнения запроса отображается в нижней части экрана или, если выбрана опция View | Results, во весь экран.

Отметим, что MDX Sample Application не выполняет проверку корректности введенного MDX-запроса перед отсылкой его серверу на обработку.





Теперь вы готовы приступить к изучению языка MDX. Этой теме будет посвящена следующая часть данной статьи. В качестве исходных данных для выполнения примеров мы воспользуемся многомерной базой данных FoodMart, входящей в комплект поставки Microsoft SQL Server Analysis Services.

## Синтаксис языка MDX

Запрос на языке MDX представляет собой набор команд, который выглядит следующим образом:

```
SELECT [<axis_specification>
      [, <axis_specification>...]]
FROM [<cube_specification>]
[WHERE
< slicer_specification>]]
```

где:

- axis\_specification — содержит описание осей куба;
- cube\_specification — содержит название куба;
- slicer\_specification — содержит описание срезов куба.

В языке MDX выражение SELECT используется для задания набора данных, содержащего подмножество многомерных данных. Простейший SELECT-запрос может выглядеть так:

```
SELECT FROM Sales
```

В этом примере мы получили общее число продаж (Unit Sales) для всего куба. Поскольку в запросе мы не указали имена членов измерений, были выбраны члены по умолчанию из каждого измерения. Наш запрос эквивалентен следующему:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS
FROM SALES
```

Более полный SELECT-запрос должен содержать следующую информацию:

- число осей (в одном запросе можно указать до 128 осей);
- список членов измерения, которые должны быть включены для каждой оси;
- имя куба, к которому производится запрос;
- список членов среза.



Рассмотрим более сложный пример, который позволит нам разобраться с различными элементами MDX-запроса:

```
SELECT
  { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,
  { [Time].[1997], [Time].[1998] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

SELECT определяет используемые оси. В нашем примере их две: одна — задает значения для колонок, другая — для рядов:

```
{ [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,
{ [Time].[1997], [Time].[1998] } ON ROWS
```

Выражение FROM определяет источник многомерных данных, к которому обращен наш запрос. В данном примере — это куб Sales.

Выражение WHERE задает размерности или члены, используемые в качестве среза. В нашем примере мы ограничили данные размерностью Store.

Рассмотрим еще несколько запросов. В первом запросе мы получаем данные по продажам всех товаров для всех лет:

```
SELECT [Product].[Product Category].Members ON COLUMNS FROM Sales
```

Данные для целого года (1997):

```
SELECT [Product].[Product Category].Members ON COLUMNS,
  {[Time].[1997]} ON ROWS
FROM Sales
```

Данные по кварталам:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
  {[Time].[Quarter].Members} ON ROWS
FROM Sales
```

Данные для первого квартала:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
  {[Time].[1997].[Q1]} ON ROWS
FROM Sales
```

Данные для первого месяца первого квартала:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
  {[Time].[1997].[Q1].[1]} ON ROWS
FROM Sales
```

Продажи для всех клиентов в США:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
  {[Customers].[All Customers].[USA]} ON ROWS
FROM Sales
```

Продажи для всех клиентов в штате Калифорния:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
{[Customers].[All Customers].[USA].[CA]} ON ROWS
FROM Sales
```

Продажи товаров по категориям для всех клиентов в штате Калифорния:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
{[Product].[Product Family].Members} ON ROWS
FROM Sales
WHERE [Customers].[All Customers].[USA].[CA]
```

Продажи товаров по категориям для всех клиентов в штате Калифорния в первом квартале 1997 года:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
{[Product].[Product Family].Members} ON ROWS
FROM Sales
WHERE ([Customers].[All Customers].[USA].[CA], [Time].[1997].[Q1])
```

Продажи морепродуктов для всех клиентов в штате Калифорния в первом квартале 1997 года:

```
SELECT {[Measures].[Unit Sales]} ON COLUMNS,
{[Product].[Product Department].[Seafood]} ON ROWS
FROM Sales
WHERE ([Customers].[All Customers].[USA].[CA], [Time].[1997].[Q1])
```

## Функции языка MDX

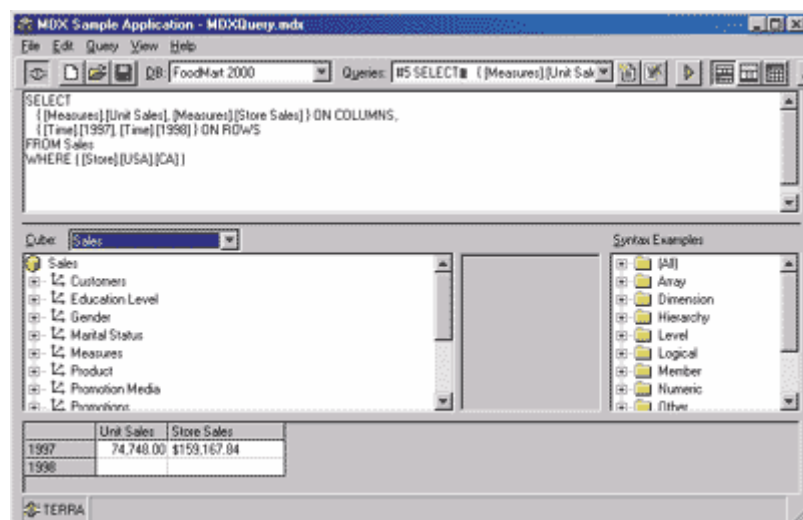
Функции, реализованные в языке MDX, разделяются на несколько групп, как показано в таблице.

Группы функций и входящие в них функции

Array Functions				
SetToArray				
Dimension, Hierarchy and Level Functions				
Dimension	Dimensions	Hierarchy	Level	Levels
Logical Functions				
Is	IsAncestor	IsEmpty	IsGeneration	IsLeaf
IsSibling				
Member Functions				
Ancestor	ClosingPeriod	Cousin	CurrentMember	DataMember
DefaultMember	FirstChild	FirstSibling	Ignore	Item
Lag	LastChild	LastSibling	Lead	LinkMember
Members	NextMember	OpeningPeriod	ParallelPeriod	Parent
PrevMember	StrToMember	ValidMeasure		
Numeric Functions				
Aggregate	Avg	CalculationCurrentPass	CalculationPassValue	CoalesceEmpty
Correlation	Count	Covariance	CovarianceN	DistinctCount
IIf	LinRegIntercept	LinRegPoint	LinRegR2	LinRegSlope
LinRegVariance	LookupCube	Max	Median	Min
Ordinal	Predict	Rank	RollupChildren	Stddev
StddevP	Stdev	StdevP	StrToValue	Sum

Value	Var	Variance	VarianceP	VarP
Set Functions				
AddCalculatedMembers	AllMembers	Ancestors	Ascendants	Axis
BottomCount	BottomPercent	BottomSum	Children	Crossjoin
Descendants	Distinct	DrilldownLevel	DrilldownLevelBottom	DrillDownLevelTop
DrilldownMember	DrilldownMemberBottom	DrilldownMemberTop	DrillupLevel	DrillupMember
Except	Extract	Filter	Generate	Head
Hierarchize	Intersect	LastPeriods	Members	Mtd
NameToSet	NonEmptyCrossjoin	Order	PeriodToDate	Qtd
Siblings	StripCalculatedMembers	StrToSet	Subset	Tail
ToggleDrillState	TopCount	TopPercent	TopSum	Union
VisualTotals	Wtd	Ytd		
String Functions				
CalculationPassValue	CoalesceEmpty	Generate	IIf	LookupCube
MemberToStr	Name	Properties	SetToStr	TupleToStr
UniqueName	UserName			
Tuple Functions				
Current	Item	StrToTuple		
Other Functions				
Call				

Кроме того, перечисленные функции языка MDX могут использоваться при вычислении различных выражений, в том числе при создании вычисляемых измерений. Кроме того, при необходимости для применения в MDX можно создавать функции, определенные пользователем (User Defined Functions, UDF), с помощью средств разработки, поддерживающих создание COM DLL.



Более подробные сведения о синтаксисе этого языка вы можете найти в Microsoft SQL Server Books Online, а примеры его практического применения — в статье «Введение в MDX».

# Часть 9. Создание OLAP-клиентов с помощью ADO и ADOMD

## Применение ADO в OLAP-клиентах

- Чтение метаданных

- Выполнение MDX-запросов

## Применение ADO MD в OLAP-клиентах

- Чтение метаданных

- Выполнение MDX-запросов

## Применение средств разработки для создания OLAP-клиентов

- Visual Basic for Applications

- Visual Studio .Net

- Delphi и C++Builder

Предыдущая часть данной статьи (КомпьютерПресс № 11'2001), а также статья «Введение в MDX», вторую часть которой вы найдете в настоящем номере нашего журнала, посвящены построению запросов к OLAP-кубам с помощью языка MDX (Multidimensional Expressions). Мы убедились, что этот язык позволяет получить из многомерных баз данных любые сведения, содержащиеся в них, и теперь можем приступить к созданию собственных OLAP-клиентов. В качестве средства создания OLAP-кубов мы будем использовать Visual Basic 6. В конце статьи мы также затронем вопрос об использовании других средств разработки.

Существует два способа получения данных из OLAP-кубов Microsoft SQL Server 2000 Analysis Services. Первый основан на применении ADO (ActiveX Data Objects) и OLEDB-провайдера для доступа к многомерным данным — Microsoft OLE DB Provider for OLAP Services 8.0. При втором способе используется ADO MD (ADO Multidimensional) — технология, позволяющая обращаться как к метаданным многомерных баз данных, так и к результатам MDX-запросов. Мы начнем с применения ADO, а затем перейдем к ADO MD.

## Применение ADO в OLAP-клиентах

Вобщем случае OLAP-клиент должен уметь не только выполнять MDX-запросы, но и получать сведения о метаданных многомерных баз данных, то есть имена содержащихся в них кубов, их измерений, иерархий, уровней и членов. С точки зрения приложений, не создающих кубы, а только читающих содержащиеся в них данные, члены измерений вполне можно отнести к метаданным, поскольку с точки зрения таких приложений они неизменны. Но, как только мы начинаем создавать приложения, модифицирующие или создающие OLAP-кубы, члены измерения (а в случае некоторых нестандартных типов измерений — и их уровни) к метаданным отнести уже нельзя — их значения определяются в исходном источнике данных, который обычно подвержен изменениям. Впрочем, с помощью ADO и ADO MD кубы обычно не создаются — для этого существуют другие технологии, такие как PivotTable Services и SQL Decision Support Objects (DSO), о которых будет говориться в последующих частях данной статьи.

## Чтение метаданных

Итак, первая задача, которую следует решить, — как получить доступ к метаданным с помощью ADO. Для ее решения в случае реляционных баз данных обычно применяется метод OpenSchema объекта ADO Connection, возвращающий объект ADO Recordset, содержащий сведения о метаданных указанной базы данных. Этот же метод применим и к многомерным базам данных, если обращаться к ним с помощью провайдера Microsoft OLE DB Provider for OLAP Services (или Microsoft OLE DB Provider for OLAP Services 8.0).

Метод OpenSchema имеет три параметра — QueryType, Criteria и SchemaID, из которых обязательным является лишь первый. Среди значений параметра QueryType, имеющих отношение к многомерным базам данных, наибольший интерес представляют значения adSchemaCubes, adSchemaDimensions, adSchemaLevels, adSchemaMembers и adSchemaMeasures, позволяющие получить объект ADO Recordset со сведениями о кубах, измерениях, уровнях, членах измерений и мерах соответственно.

Создадим простейшее приложение для отображения метаданных с помощью Visual Basic 6. Начнем с описания глобальной переменной, которая будет содержать значение QueryType метода OpenSchema:

```
Public ads As Integer
```

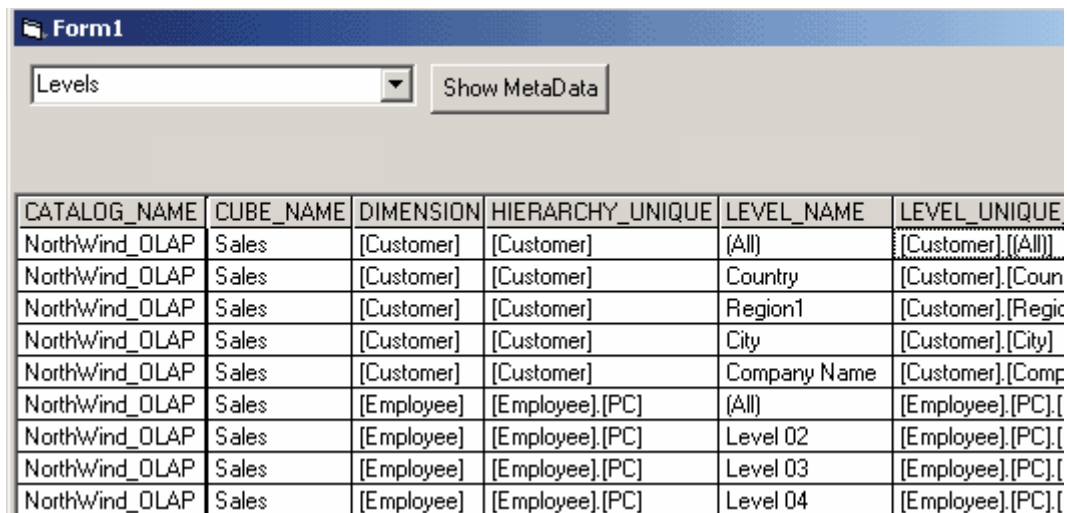
Далее поместим на главную форму приложения компоненты MSHFlexGrid для отображения результатов, возвращаемых методом OpenSchema, и кнопку для инициирования его выполнения. Поместим на форму также компонент ComboBox, заполним его свойство List значениями Cubes, Dimensions, Levels, Members и Measures и создадим обработчик события Click, связанного с выбором пользователем одного из значений списка:

```
Private Sub Combo1_Click()  
    Select Case Combo1.ListIndex  
        Case 0  
            ads = adSchemaCubes  
        Case 1  
            ads = adSchemaDimensions  
        Case 2  
            ads = adSchemaLevels  
        Case 3  
            ads = adSchemaMembers  
        Case 4  
            ads = adSchemaMeasures  
    End Select  
    Command1.Enabled = True  
End Sub
```

Далее создадим обработчик события Click для кнопки:

```
Private Sub Command1_Click()  
    Dim RS As New ADODB.Recordset  
    Dim Cnn As New ADODB.Connection  
    Cnn.ConnectionString = "Provider=MSOLAP.2;" & _  
        "Persist Security Info=True;User ID=sa;Data Source=MAINDESK;" & _  
        "Initial Catalog=NorthWind_OLAP"  
    Cnn.Open  
    Set RS = Cnn.OpenSchema(ads)  
    Set MSHFlexGrid1.Recordset = RS  
    Set Cnn = Nothing  
    Set RS = Nothing  
End Sub
```

В этом обработчике события мы обращаемся к методу OpenSchema объекта Connection с параметром QueryType, соответствующим значению, выбранному из выпадающего списка, и отображаем полученный набор данных в объекте MSHFlexGrid1 (рис. 1).



CATALOG_NAME	CUBE_NAME	DIMENSION	HIERARCHY_UNIQUE	LEVEL_NAME	LEVEL_UNIQUE
NorthWind_OLAP	Sales	[Customer]	[Customer]	(All)	[Customer].[All]
NorthWind_OLAP	Sales	[Customer]	[Customer]	Country	[Customer].[Country]
NorthWind_OLAP	Sales	[Customer]	[Customer]	Region1	[Customer].[Region1]
NorthWind_OLAP	Sales	[Customer]	[Customer]	City	[Customer].[City]
NorthWind_OLAP	Sales	[Customer]	[Customer]	Company Name	[Customer].[Company Name]
NorthWind_OLAP	Sales	[Employee]	[Employee].[PC]	(All)	[Employee].[PC].[All]
NorthWind_OLAP	Sales	[Employee]	[Employee].[PC]	Level 02	[Employee].[PC].[Level 02]
NorthWind_OLAP	Sales	[Employee]	[Employee].[PC]	Level 03	[Employee].[PC].[Level 03]
NorthWind_OLAP	Sales	[Employee]	[Employee].[PC]	Level 04	[Employee].[PC].[Level 04]

Рис. 1. Получение сведений о метаданных с помощью метода OpenSchema объекта ADO Connection

Таким образом, мы получили доступ к метаданным многомерной базы данных с помощью метода ADO OpenSchema. Нашей следующей задачей будет выполнение MDX-запроса к соответствующим данным. Зная, каковы метаданные, мы вполне можем это сделать.

## Выполнение MDX-запросов

Для выполнения MDX-запросов немного модифицируем наше приложение, добавив еще одну кнопку и компонент TextBox для ввода текста MDX-запроса. Создадим обработчик события Click для второй кнопки:

```
Private Sub Command2_Click()
    Dim RS As New ADODB.Recordset
    Dim Cnn As New ADODB.Connection
    On Error GoTo Err1
    Cnn.ConnectionString = "Provider=MSOLAP.2;" & _
        "Persist Security Info=True;User ID=sa;Data Source=MAINDESK;" & _
        "Initial Catalog=NorthWind_OLAP"
    Cnn.Open
    Set RS = Cnn.Execute(Text1.Text)
    Set MSHFlexGrid1.Recordset = RS
    Set Cnn = Nothing
    Set RS = Nothing
    Exit Sub
Err1:
    MsgBox ("Неверный запрос")
End Sub
```

В этом обработчике события мы инициируем выполнение запроса с помощью метода Execute объекта ADO Connection. Результат его выполнения показан на рис. 2.

Levels

Show MetaData

Show Data

```
SELECT measures.members
ON COLUMNS, Product.Children
ON ROWS FROM Sales
```

[Product].[Category Name]	[Measures].[Line Item Total]	[Measures].[Line Item Quant]	[Measures].[Line Item Discount]
Beverages	280582.95	9210	18010.4701
Condiments	112506.75	5235	7459.565
Confections	174295.9	7742	9623.848
Dairy Products	247766.5	8997	16815.335
Grains/Cereals	95759.8	4398	4980.2175
Meat/Poultry	177195.8	4155	15063.5905
Produce	98559.55	2831	4928.7525
Seafood	140347.58	7551	10277.4425

Рис. 2. Результат выполнения MDX-запроса с помощью метода Execute объекта ADO Connection

Нужно иметь в виду, что при применении метода Execute объекта ADO Connection результат любого MDX-запроса можно представить в виде двумерного набора данных, даже если этот запрос возвращает набор данных, число измерений в котором более двух. Например, результат запроса:

```
SELECT measures.members ON columns, product.children ON rows, shipper.children
ON pages FROM sales
```

представляющий собой трехмерный набор данных, отображается на обычный объект ADO Recordset, содержащий набор последовательных сечений этого трехмерного набора данных (рис. 3).

Dimensions

Show MetaData

Show Data

```
select measures.members on columns,
product.children on rows, shipper.children on
pages from sales
```

[Shipper].[Shipper Name]	[Product].[Category Name]	[Measures].[Line Item Total]	[Measures].[Line Item Quantity]
Federal Shipping	Beverages	95016,45	2681
Federal Shipping	Condiments	29585,3	1487
Federal Shipping	Confections	55532,11	2604
Federal Shipping	Dairy Products	66083	2486
Federal Shipping	Grains/Cereals	26032,45	1225
Federal Shipping	Meat/Poultry	51756,41	1403

Рис. 3. Результат выполнения MDX-запроса, возвращающего трехмерный набор данных

Таким же образом можно обращаться и к локальным OLAP-кубам, созданным с помощью Microsoft Excel и сохраненным в файлах \*.cub. В этом случае свойство ConnectionString объекта Connection может выглядеть так:

```
Cnn.ConnectionString =
"Provider=MSOLAP.2;Data Source=C:\Data\Cubes\NW1.cub"
```

Отметим, что рассмотренный способ создания OLAP-клиентов — не единственный из возможных. Есть еще один способ, довольно часто используемый в коммерческих OLAP-клиентах, — это применение ADO MD. Об этой технологии пойдет речь в следующем разделе.

## Применение ADO MD в OLAP-клиентах

ADO MD — это расширение ADO, реализованное в библиотеке msadomd.dll и содержащее объектную модель, позволяющую обращаться как к метаданным многомерных баз данных, так и к результатам MDX-запросов.

Объектная модель ADO MD, представленная на рис. 4, состоит из двух «ветвей» объектов: первая из них используется для доступа к метаданным многомерной базы данных, а вторая — для извлечения данных с помощью запросов к OLAP-кубам.

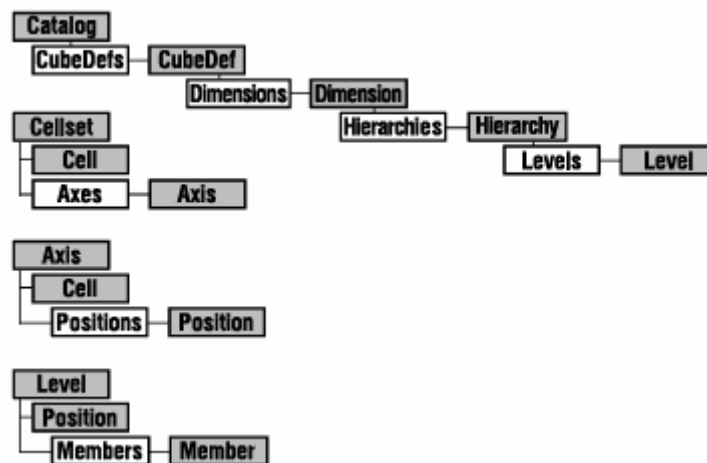


Рис. 4. Объектная модель ADO MD

Первый из объектов ADO MD для доступа к метаданным, объект Catalog, представляет многомерное хранилище данных, которое может содержать ноль, один или более кубов. Следовательно, одним из свойств объекта Catalog является коллекция CubeDefs. Каждый элемент этой коллекции представляет собой объект CubeDef, описывающий конкретный куб в хранилище. Имя куба — это значение свойства Name соответствующего объекта CubeDef.

Каждый объект CubeDef может содержать коллекцию Dimensions, состоящую из объектов Dimension. Каждый объект Dimension, в свою очередь, представляет конкретное измерение куба, а имя этого измерения содержится в свойстве Name.

Все объекты Dimension включают в себе коллекцию Hierarchies, которая может содержать один или более объектов Hierarchy. Иерархия же может содержать один или несколько уровней, поэтому объект Hierarchy содержит коллекцию Levels, состоящую из объектов Level. Каждый из уровней иерархии содержит один или более членов, поэтому объект Level содержит коллекцию Members, состоящую из объектов Member.

## Чтение метаданных

Используя Visual Basic 6, создадим пример, выводящий сведения о метаданных многомерной базы данных с помощью ADO MD. Для этой цели поместим на форму компонент TreeView и кнопку, создадим обработчик события Click этой кнопки, в котором заполним компонент TreeView сведениями о метаданных (см. листинг 1).

### Листинг 1

```
Private Sub Command1_Click()
    Dim Nod, Nod1 As Node, i, j, k, l As Integer
    Dim Cat As New ADOMD.Catalog
    Dim Cnn As New ADODB.Connection
```



```

Conn.ConnectionString = "Provider=MSOLAP.2;Persist Security Info=True;"_
+"User ID=sa;Data Source=MAINDESK;Initial Catalog=NorthWind_OLAP"
Conn.Open
Set Cat.ActiveConnection = Conn
If Cat.CubeDefs.Count > 0 Then
Set Nod = TreeView1.Nodes.Add(, , "Catalog", "Catalog")
For i = 0 To Cat.CubeDefs.Count - 1
Set Nod1 = TreeView1.Nodes.Add("Catalog", tvwChild, _
Cat.CubeDefs(i).Name, Cat.CubeDefs(i).Name)
For j = 0 To Cat.CubeDefs(i).Dimensions.Count - 1
Set Nod1 = TreeView1.Nodes.Add(Cat.CubeDefs(i).Name, tvwChild, _
Cat.CubeDefs(i).Dimensions(j).UniqueName, _
Cat.CubeDefs(i).Dimensions(j).UniqueName)
For k = 0 To (Cat.CubeDefs(i).Dimensions(j).Hierarchies.Count - 1)
For l = 0 To _
(Cat.CubeDefs(i).Dimensions(j).Hierarchies(k).Levels.Count - 1)
Set Nod1 = _
TreeView1.Nodes.Add(Cat.CubeDefs(i).Dimensions(j).UniqueName,_
tvwChild, _
Cat.CubeDefs(i).Dimensions(j).Hierarchies(k).Levels(l).UniqueName,_
Cat.CubeDefs(i).Dimensions(j).Hierarchies(k).Levels(l).UniqueName)
Next l
Next k
Next j
Next i
Command1.Enabled = False
End If
End Sub

```

Результат работы этого приложения показан на рис. 5.

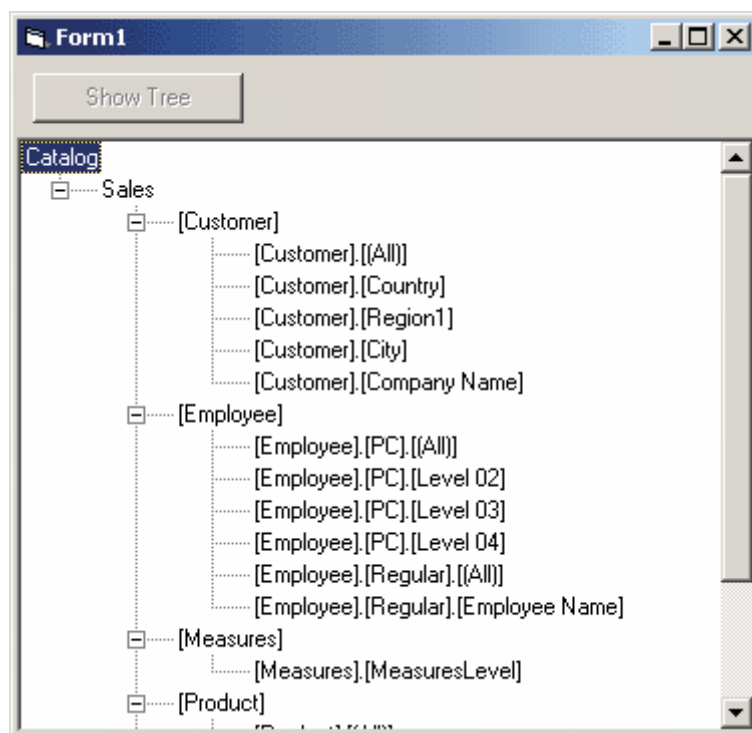


Рис. 5. Приложение для чтения метаданных с помощью ADO MD

Теперь нам осталось решить последнюю задачу — выполнение MDX-запросов с помощью ADO MD.

## Выполнение MDX-запросов

В объектной модели ADO MD результат MDX-запроса представлен объектом `CellSet`. Этот объект позволяет осуществить доступ к объектам `Cells`, каждый из которых представляет конкретные ячейки в наборе данных, получающемся в результате запроса. Помимо этого объект `CellSet` содержит коллекцию `Axes`, состоящую из объектов `Axis` (число членов этой коллекции равно числу измерений в результирующем наборе данных). Как объект `Cell`, так и объект `Axis` обладают коллекцией `Positions`, состоящей из объектов `Position`, соответствующих позициям вдоль оси. Объект `Position` обладает коллекцией `Members`, члены которой представляют конкретные значения данных на оси.

Добавим в уже созданное приложение возможность выполнять MDX-запросы. Для этого поместим на форму приложения еще одну кнопку, два компонента `TextBox` и создадим обработчик события `Click` для этой кнопки (см. листинг 2).

### Листинг 2

```
Private Sub Command2_Click()  
    Dim Cst As New ADOMD.Cellset  
    Dim Cnn As New ADODB.Connection  
    Cnn.ConnectionString = "Provider=MSOLAP.2;Persist Security Info=True;" _  
        + "User ID=sa;Data Source=MAINDESK;Initial Catalog=NorthWind_OLAP"  
    Cnn.Open  
    Text2.Text = vbTab & vbTab  
    Set cst.ActiveConnection = cnn  
    cst.Source = Text1.Text  
    On Error GoTo Err  
    cst.Open  
    For i = 0 To cst.Axes(0).Positions.Count - 1  
        Text2.Text = Text2.Text & cst.Axes(0).Positions(i).Members(0).Caption _  
            & vbTab  
    Next i  
    Text2.Text = Text2.Text & vbCrLf & vbCrLf  
    For j = 0 To cst.Axes(1).Positions.Count - 1  
        Text2.Text = Text2.Text & cst.Axes(1).Positions(j).Members(0).Caption _  
            & vbTab  
        For k = 0 To cst.Axes(0).Positions.Count - 1  
            Text2.Text = Text2.Text & vbTab & cst(k, j).FormattedValue  
        Next k  
        Text2.Text = Text2.Text & vbCrLf  
    Next j  
Exit Sub  
Err:  
    MsgBox ("Invalid Query")  
End Sub
```

В данном фрагменте кода мы создаем объект `CellSet` и используем его метод `Open`. Если MDX-запрос корректен, то выполняется цикл перебора ячеек объекта `CellSet` и происходит вывод их содержимого в компонент `Text2`. Результат работы созданного приложения приведен на рис. 6.

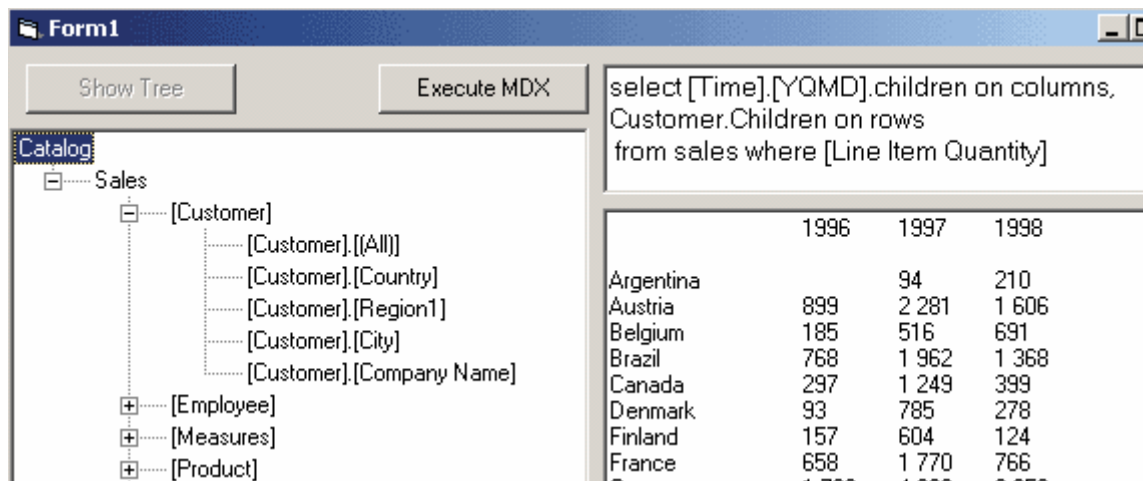


Рис. 6. Приложение для выполнения MDX-запросов с помощью ADO MD

Обратите внимание: данный фрагмент кода написан при допущении, что объект Cellset содержит двумерный набор данных. В общем же случае число измерений этого набора данных может превышать указанное значение, и это обстоятельство следует учитывать при создании OLAP-клиентов.

## Применение средств разработки для создания OLAP-клиентов

### Visual Basic for Applications

Примеры, использующие ADO, почти без изменений можно выполнить в Access VBA, создав форму Access и разместив на ней примерно те же интерфейсные элементы (за исключением TextBox — в Access он предназначен для отображения полей таблиц или запросов, поэтому рекомендуется заменить его на какой-нибудь элемент управления для редактирования текста, например RichEdit Control). Используя позднее связывание, можно решить эту задачу и с помощью других приложений MS Office, например:

```
Dim cnnConn, rs As Object
Set cnnConn = CreateObject("ADODB.Connection")
Cnn.ConnectionString = "Provider=MSOLAP.2;Persist Security Info=True;" _
    + "User ID=sa;Data Source=MAINDESK;Initial Catalog=NorthWind_OLAP"
Source=e:\old_data\d\whistler\sales.cub"
cnnConn.Open
Set rs = CreateObject("ADODB.Recordset")
Set rs = cnnConn.OpenSchema(ads)
```

Что касается примеров, использующих ADO MD, то их воспроизведение с помощью VBA также возможно посредством позднего связывания.

### Visual Studio .Net

Поскольку Visual Studio .Net позволяет применять COM и ADO, все описанные выше примеры можно перенести и в это средство разработки, получив в результате проекты Visual Basic .Net, использующие либо ADO, либо ADO MD, равно как и создать их непосредственно с помощью этого средства разработки. Если же мы хотим создать с нуля подобный проект посредством ADO .Net, то можем столкнуться с тем, что метод GetOleDbSchemaTable, являющийся ADO.Net-аналогом метода OpenSchema, в версии Visual Studio .Net Beta 2 не поддерживает многомерные базы данных и что с его помощью можно получить только наименования кубов, но не сведения об изменениях, уровнях, членах и мерах.

Что касается MDX-запросов, их вполне можно выполнять, используя объекты OleDbConnection и OleDbCommand. В дальнейшем мы планируем посвятить ряд статей технологии ADO .Net, где рассмотрим эти объекты более подробно.

## Delphi и C++Builder

Пример, аналогичный первому из рассмотренных, можно создать с помощью редакции Enterprise обоих этих средств разработки начиная с версии 5. Для получения сведений о метаданных можно воспользоваться методом OpenSchema компонента TADOConnection:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SI : TSchemaInfo;
  I : Integer;
begin
  case ComboBox1.ItemIndex of
    0: SI := siCubes;
    1: SI := siDimensions;
    2: SI := siHierarchies;
    3: SI := siLevels;
    4: SI := siMeasures;
    5: SI := siProperties;
    6: SI := siMembers;
  end;
  ADOConnection1.OpenSchema(SI, EmptyParam, EmptyParam, ADODataSet1);
  ADODataSet1.Open;
end;
```

Для получения же самих данных можно воспользоваться методом Open компонента TADODataSet:

```
procedure TForm1.Button1Click(Sender: TObject);
var i:integer;
begin
  If Memo1.Lines.Count > 0 Then
  try
    With ADOQuery1 do
      begin
        Close;
        SQL := Memo1.Lines;
        Open;
      end;
    except
      ShowMessage('Invalid MDX query');
    end;
  end;
```

Создание OLAP-клиентов с применением ADO MD с помощью этих двух средств разработки также вполне возможно, поскольку оба они поддерживают все способы создания COM-клиентов начиная с версии 3.0.

## **Часть 10 Применение PivotTable Service для создания локальных OLAP-кубов**

Microsoft PivotTable Service  
Расширения DDL и DML для создания локальных кубов  
Предложение CREATE CUBE  
Предложение INSERT INTO  
Свойства Source\_DSN, Data Source и Provider  
Пример создания локального OLAP-куба  
Поставка приложений, использующих PivotTable Service

Предыдущая статья данного цикла была посвящена созданию собственных OLAP-клиентов с помощью Visual Basic 6 и других средств разработки. В ней мы обсуждали два способа чтения данных из OLAP-кубов Microsoft SQL Server 2000 Analysis Services: способ, основанный на применении ADO (ActiveX Data Objects) и Microsoft OLE DB Provider for OLAP Services 8.0, и способ, основанный на применении ADO MD (ADO Multidimensional).

Отметим, что помимо непосредственного доступа к OLAP-данным из приложений можно создавать контроллеры автоматизации Excel или применять в приложениях компоненты PivotTable List и ChartSpace (об этом мы уже упоминали в предыдущих статьях данного цикла; см. КомпьютерПресс № 9, 10'2001). Подобные решения нередко оказываются весьма выгодными с точки зрения трудозатрат на их создание, если, конечно, пользователи созданных приложений обладают лицензиями Microsoft Office и установленными Office Web Components или Excel, — ведь значительная часть кода, требующегося для реализации манипуляций с отображением OLAP-данных, в Microsoft Office уже реализована.

Тем не менее просмотр OLAP-данных — не единственная задача, которую можно реализовать в клиентских приложениях. Нередко в них требуется также возможность создания или обновления OLAP-кубов. Этой теме будут посвящены настоящая и следующая статьи данного цикла. В этот раз мы рассмотрим программное создание OLAP-кубов с помощью Microsoft PivotTable Service, а в следующей статье — программное создание серверных OLAP-кубов с помощью Decision Support Objects.

### **Microsoft PivotTable Service**

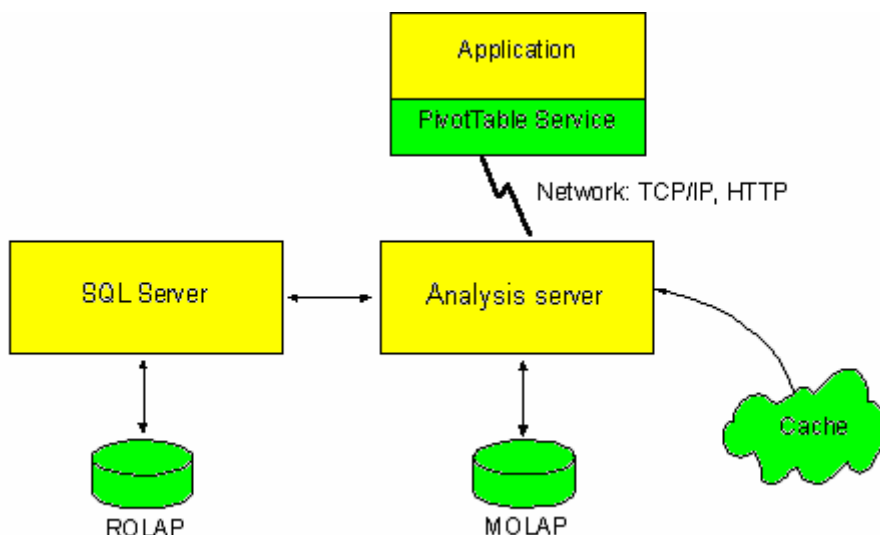
Microsoft PivotTable Service представляет собой OLE DB-провайдер, поддерживающий расширения OLE DB for OLAP, выполняющий роль интерфейса к Analysis Services, а также осуществляющий кэширование данных Analysis Services в клиентском приложении.

Помимо этого PivotTable Service является и OLAP-сервером, реализованным в виде клиентских библиотек. Этот сервер используется для локального анализа данных, находящихся в кэше клиентского приложения, манипуляции ими и создания локальных OLAP-кубов.

PivotTable Service поддерживает язык Multidimensional Expressions (MDX) и подмножество языка SQL, а также некоторые дополнительные расширения подмножеств языка SQL — DDL (Data Definition Language) и DML (Data Manipulation Language), необходимые для описания структуры локальных OLAP-кубов. О них мы поговорим ниже.

С помощью PivotTable Service можно создавать локальные кубы на основе данных из реляционных СУБД и серверных OLAP-кубов и затем формировать запросы к таким кубам, не обращаясь непосредственно к Analysis Services, что во многих случаях позволяет снизить сетевой трафик и повысить производительность приложений, а также, используя возможность

записи в ячейки таких кубов, проводить анализ возможных вариантов. Отметим, что соединение с Analysis Services можно осуществить с помощью протоколов TCP/IP или HTTP.



*Microsoft PivotTable Service*

Microsoft PivotTable Service используется в клиентах Microsoft SQL Server 2000 Analysis Services. Он используется, в частности, приложениями Microsoft Office (например, при создании сводных таблиц и диаграмм Microsoft Excel), равно как и другими клиентскими приложениями. PivotTable Service входит в состав как Analysis Services, так и Microsoft Excel и может быть включен в состав других клиентских приложений.

Microsoft PivotTable Service можно использовать в различных средствах разработки, поддерживающих создание COM-клиентов, в частности Visual Basic, Microsoft Visual C++, Borland Delphi, Borland C++Builder.

## Расширения DDL и DML для создания локальных кубов

Прежде чем приступить к программному созданию локальных OLAP-кубов с помощью Microsoft PivotTable Service, нам следует обсудить, какие расширения DDL и DML для этого требуются. Как минимум, нам понадобится описать структуру будущего куба (этой цели служит свойство PivotTable Service CreateCube, содержащее предложение CREATE CUBE), описать, какие данные следует поместить в этот куб (этой цели служит свойство PivotTable Service InsertInto, содержащее предложение INSERT INTO), а также указать, откуда взять эти данные и в каком файле сохранить локальный куб.

## Предложение CREATE CUBE

Это предложение представляет собой расширение DDL, предназначенное для описания структуры будущего локального куба. Полный синтаксис этого предложения можно найти в SQL Server Books Online, мы же ограничимся наиболее часто встречающимся видом этого предложения:

```
CREATE CUBE <Cube Name>
(
    DIMENSION <Dimension Name> [TYPE <Dimension Type>]
    LEVEL <Level Name>[TYPE <Level Type>],
    [LEVEL <Level Name> [TYPE <Level Type>]...],
    [[DIMENSION <Dimension Name> [TYPE <Dimension Type>]
    LEVEL <Level Name>[TYPE <Level Type>],
```

```

[LEVEL <Level Name> >[TYPE <Level Type>...],...],
MEASURE <Measure Name> FUNCTION <Function Name>,
[MEASURE <Measure Name> FUNCTION <Function Name>,...]
)

```

Например, создавая локальный куб на основе базы данных NorthWind, входящей в комплект поставки Microsoft SQL Server, мы можем написать следующее предложение CREATE CUBE:

```

CREATE CUBE Sales(
    DIMENSION [Country], LEVEL [All] TYPE ALL, LEVEL [Country],
        LEVEL [City], LEVEL [CustomerID],
    DIMENSION [Salesperson], LEVEL [All] TYPE ALL, LEVEL [Salesperson],
    DIMENSION [ShipperName], LEVEL [All] TYPE ALL, LEVEL [ShipperName],
    DIMENSION [CategoryName], LEVEL [All] TYPE ALL, LEVEL [CategoryName],
        LEVEL [ProductName],
    DIMENSION [OrderDate] TYPE TIME, LEVEL [All] TYPE ALL,
        LEVEL [Year] TYPE YEAR,
        LEVEL [Quarter] TYPE QUARTER,
        LEVEL [Month] TYPE MONTH,
        LEVEL [Day] TYPE DAY,
    MEASURE [Sum Of ExtendedPrice] FUNCTION SUM,
    MEASURE [Sum Of Quantity] FUNCTION SUM
)

```

## Предложение INSERT INTO

Это предложение представляет собой расширение DML, предназначенное для заполнения локального куба данными. Полный синтаксис этого предложения можно найти в SQL Server Books Online. Рассмотрим наиболее часто встречающийся вид этого предложения:

```

InsertInto=INSERT INTO <Cube name>
(
    <Dimension Name>.<Level Name>,
    [<Dimension Name>.<Level Name>,...]
    <Measure Name>,
    [<Measure Name>,... ]
)
OPTIONS <option list>
SELECT <Column list>
FROM <table list>
WHERE <joins list>

```

Например, в случае заполнения данными локального куба, описанного выше, это предложение может выглядеть так:

```

InsertInto=INSERT INTO Sales
[Country].[Country], [City], [CustomerID],
[Sum Of ExtendedPrice], [Sum Of Quantity],
[ShipperName].[ShipperName], [Salesperson].[Salesperson],
[CategoryName].[CategoryName],
[ProductName], [OrderDate])
OPTIONS ATTEMPT_ANALYSIS
SELECT Invoices.Country, Invoices.City, Invoices.CustomerID,
    Invoices.ExtendedPrice, Invoices.Quantity, Invoices.ShipperName,
    Invoices.Salesperson, Categories.CategoryName,
    Products.ProductName, Invoices.OrderDate
FROM Northwind.dbo.Categories Categories,
    Northwind.dbo.Invoices Invoices,
    Northwind.dbo.Products Products
WHERE Categories.CategoryID = Products.CategoryID
AND Invoices.ProductID = Products.ProductID

```

Параметр OPTIONS предназначен для описания различных способов создания кубов (как и серверные кубы, локальные кубы могут быть созданы как ROLAP- или MOLAP-хранилища, при этом во втором случае при работе с кубом не требуется обращаться к реляционному источнику, а сам куб может быть отчуждаемым). В данном примере мы создаем именно такой куб.

## Свойства Source\_DSN, Data Source и Provider

Эти свойства описывают OLEDB-источник данных для создания локального куба, в каком файле его сохранить и с помощью какого OLEDB-провайдера к нему потом обращаться. В случае создания локального куба на основе базы данных Northwind свойство Source\_DSN может выглядеть, например, так:

```
SOURCE_DSN="DRIVER=SQL Server;SERVER=MAINDESK;UID=sa;DATABASE=Northwind"
```

Свойство Data Source может принять следующий вид:

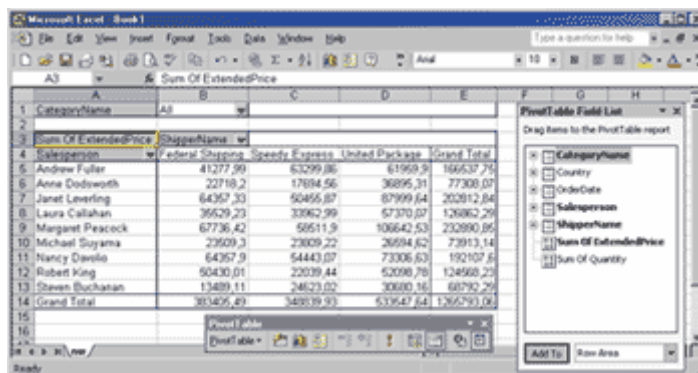
```
DATA_SOURCE=d:\nw.cub
```

Свойство Provider может выглядеть так:

```
Provider = MSOLAP.2
```

Таким образом, мы перечислили минимальный набор свойств, необходимых для программного создания локального куба. Полный список свойств можно найти в Microsoft SQL Server Books Online.

## Пример создания локального OLAP-куба



*Локальный OLAP-куб, созданный программно с помощью PivotTable Service*

Описав минимальный набор свойств локального куба, попробуем создать его программным способом. Воспользуемся для этой цели Visual Basic 6. Учитывая все сказанное выше, создать соответствующий код будет несложно (листинг 1).

### Листинг 1

```
Private Sub Command1_Click()  
Dim cnCube As ADODB.Connection  
Dim s As String  
Dim Prov As String  
Dim DS As String  
Dim SourceDSN As String  
Dim SourceDSNSuffix As String  
Dim CreateCube As String
```



```

Dim InsertInto As String
Dim Sel As String

Prov = "PROVIDER=MSOLAP"
DS = "DATA SOURCE=d:\nw.cub"
SourceDSN = "SOURCE_DSN=""DRIVER=SQL
Server;SERVER=MAINDESK;UID=sa;DATABASE=Northwind""
CreateCube = "CREATECUBE=CREATE CUBE Sales( "
CreateCube = CreateCube & " DIMENSION [Country], LEVEL [All] TYPE ALL, LEVEL
[Country], LEVEL [City], LEVEL [CustomerID], "
CreateCube = CreateCube & " DIMENSION [Salesperson], LEVEL [All] TYPE ALL, LEVEL
[Salesperson], "
CreateCube = CreateCube & " DIMENSION [ShipperName], LEVEL [All] TYPE ALL, LEVEL
[ShipperName], "
CreateCube = CreateCube & " DIMENSION [CategoryName],LEVEL [All] TYPE ALL, LEVEL
[CategoryName], LEVEL [ProductName], "
CreateCube = CreateCube & " DIMENSION [OrderDate] TYPE TIME, LEVEL [All] TYPE
ALL, LEVEL [Year] TYPE YEAR, LEVEL [Quarter] TYPE QUARTER, LEVEL [Month] TYPE
MONTH, LEVEL [Day] TYPE DAY, "
CreateCube = CreateCube & " MEASURE [Sum Of ExtendedPrice] FUNCTION SUM,
MEASURE [Sum Of Quantity] FUNCTION SUM )"

InsertInto = InsertInto & "InsertInto=INSERT INTO Sales ("
InsertInto = InsertInto & "[Country].[Country], [City], [CustomerID], [Sum Of
ExtendedPrice], [Sum Of Quantity], [ShipperName].[ShipperName],
[Salesperson].[Salesperson], "
InsertInto = InsertInto & "[CategoryName].[CategoryName], [ProductName],
[OrderDate]) OPTIONS ATTEMPT_ANALYSIS "

Sel = " SELECT Invoices.Country, Invoices.City, Invoices.CustomerID,
Invoices.ExtendedPrice, Invoices.Quantity, Invoices.ShipperName, "
Sel = Sel + "Invoices.Salesperson, Categories.CategoryName, Products.ProductName,
Invoices.OrderDate "
Sel = Sel + "FROM Northwind.dbo.Categories Categories, Northwind.dbo.Invoices
Invoices, Northwind.dbo.Products Products "
Sel = Sel + "WHERE Categories.CategoryID = Products.CategoryID AND
Invoices.ProductID = Products.ProductID"

InsertInto = InsertInto & Sel

Set cnCube = New ADODB.Connection
s = Prov & ";" & DS & ";" & SourceDSN & ";" & CreateCube & ";" & InsertInto & ";"
cnCube.Open s

End Sub

```

В этом примере кода создание куба происходит в тот момент, когда становится активным объект ADO Connection, свойство ConnectionString которого содержит перечисленные выше пять свойств PivotTable Service, разделенные точкой с запятой.

Созданный куб можно открыть, например, с помощью Microsoft Excel, просто перетащив имя соответствующего файла из Windows Explorer на лист открытой рабочей книги или в окно самого Excel.

Отметим, что аналогичный код можно создать и с помощью других средств разработки, например с помощью Borland Delphi.

## Поставка приложений, использующих PivotTable Service

Как было сказано выше, PivotTable Service может быть включен в состав других клиентских приложений; для этой цели в комплект поставки Microsoft SQL Server 2000 входят соответствующие дистрибутивы:

- Ptslite.exe — приложение для установки только файлов PivotTable Service;
- Ptsfull.exe — приложение для установки файлов PivotTable Service и Microsoft Data Access Components.

Оба эти приложения находятся в каталоге \Msolap\Install\Pts дистрибутива SQL Server 2000.

Итак, мы обсудили создание локальных OLAP-кубов с помощью Microsoft PivotTable Service, а также основные возможности PivotTable Service и основные свойства, необходимые для программного создания локального куба.

# Часть 11 Применение SQL DSO для создания серверных OLAP-кубов

Введение в объектную модель SQL DSO

Типичные задачи

- Обновление ранее созданных кубов

Работа с коллективными объектами

- Создание и удаление многомерных баз данных

- Создание и удаление источников данных

- Создание коллективных измерений

- Создание кубов

Поставка приложений, использующих SQL DSO

Предыдущая статья данного цикла (КомпьютерПресс № 1'2002) была посвящена созданию локальных OLAP-кубов с помощью Microsoft PivotTable Service. В ней мы обсудили главные возможности PivotTable Service, а также основные свойства, необходимые для программного создания локальных кубов.

Отметим, однако, что иногда требуется программное создание или, что встречается гораздо более часто, обновление серверных OLAP-кубов. Эту задачу нельзя решить с помощью PivotTable Service, так как эти библиотеки предназначены для управления клиентским кэшем. Подобные задачи обычно решаются с помощью SQL DSO (Decision Support Objects) — COM-серверов, предназначенных для управления метаданными и для иных манипуляций с объектами многомерных баз данных Analysis Services. Эти же библиотеки используются и самим приложением Analysis Manager.

## Введение в объектную модель SQL DSO

Объекты SQL DSO представляют собой COM-серверы, предназначенные для управления объектами многомерных баз данных Microsoft SQL Server Analysis Services. С их помощью можно создавать многомерные базы данных, измерения, кубы и другие объекты, содержащиеся в таких базах данных.

DSO состоят из иерархически организованных групп объектов, соответствующих основным объектам многомерных баз данных. Иерархия этих объектов в основном соответствует той, что можно увидеть в левой части окна Analysis Manager (рис. 1).

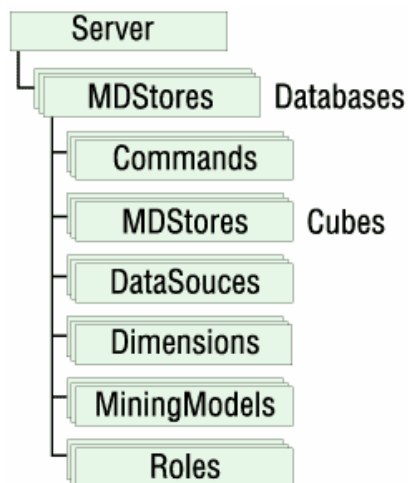


Рис. 1. Объектная модель SQL DSO (фрагмент)

Основным в объектной модели SQL DSO является объект DSO.Server, представляющий сервер Analysis Services. Он, в свою очередь, обладает свойством Databases — коллекцией объектов, представляющих многомерные базы данных, которыми управляет данный сервер. Каждый такой объект, в свою очередь, обладает коллекциями объектов, представляющих кубы различных типов, коллективные измерения, источники данных, роли.

С точки зрения DSO базы данных (объекты типа DSO.Database) и OLAP-кубы (объекты типа DSO.Cubes) обладают коллекциями MDStores, и создание баз данных, кубов, агрегатов производится путем добавления новых элементов в соответствующие коллекции.

Мы не будем подробно описывать объектную модель SQL DSO (найти ее подробное описание можно в Microsoft SQL Server Books On-Line), а рассмотрим несколько наиболее типичных задач, которые решаются с помощью этих объектов. Для этой цели мы воспользуемся Borland Delphi — пользователи Visual Basic смогут создать такие же примеры самостоятельно.

## Типичные задачи

Прежде чем обсуждать манипуляцию объектами в многомерных базах данных с помощью SQL DSO, отметим, что библиотека типов SQL DSO содержится в файле msmdso80.dll, который находится в каталоге Program Files\Common Files\Microsoft Shared\Dsso. В случае применения в качестве средства разработки Visual Basic следует просто сослаться на библиотеку Microsoft Decision Support Objects (это, по существу, и есть тот же самый файл), а вот в случае применения Delphi импортировать эту библиотеку не удастся. Поэтому при создании приложений — клиентов SQL DSO с помощью Delphi следует использовать позднее связывание, а значения констант, содержащихся в этой библиотеке, получать с помощью утилиты OLEView (рис. 2) — редактор библиотеки типов Delphi тут, к сожалению, бесполезен.

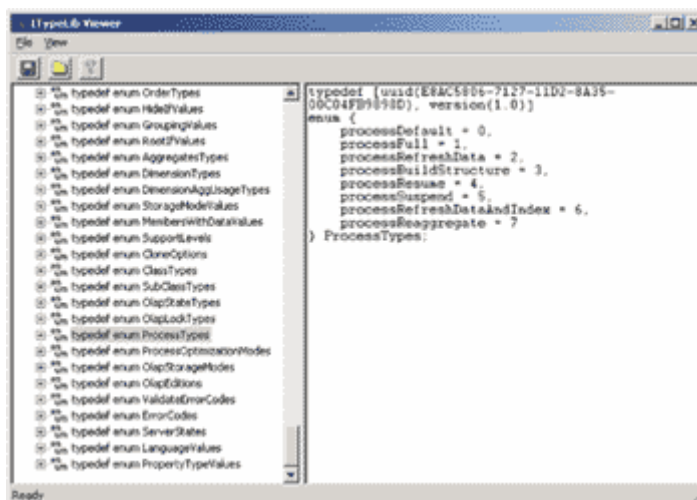


Рис. 2. Поиск констант SQL DSO с помощью утилиты OLEView

Начнем с простейшей задачи — обновления данных в уже имеющемся кубе.

## Обновление ранее созданных кубов

Обновление данных в готовых кубах, созданных с помощью Analysis Manager, — наиболее типичная задача, решаемая с помощью SQL DSO. Такая задача может быть частью аналитического приложения, предназначенного для просмотра OLAP-данных и требующего, чтобы куб содержал агрегатные данные, актуальные в момент просмотра.

Пересчет куба, содержащего только частные измерения, очень прост — обновление данных куба автоматически предваряется обновлением всех частных измерений. Фрагмент кода, приведенный ниже, иллюстрирует, как это можно сделать:

```
const
    //обновлять данные куба полностью
    ProcessFull = 1;
    //блокировать куб во время обновления
    olapLockProcess = 4;
var   Srv,dsoDB,dsoCube,dsoDim: variant;
      I           : Integer;
      OldCursor   : TCursor;
      dimName     : String;
begin
    Srv := CreateOleObject('DSO.Server');
    // соединяемся с сервером
    Srv.Connect(Edit1.Text);
    // указываем имя базы данных
    dsoDB := srv.MDStores.Item(Edit2.Text);
    // и имя куба
    dsoCube := dsoDB.Cubes.Item(Edit3.Text);
    OldCursor := Form1.Cursor;
    Form1.Cursor := crHourGlass;
    // обновляем данные куба
    dsoCube.Process (ProcessFull);
    // снимаем блокировку
    dsoCube.UnlockObject;
    Form1.Cursor := OldCursor;
    // возвращаем курсор в первоначальное состояние
    dsoCube := Unassigned;
    dsoDb := Unassigned;
    Srv := Unassigned;
end;
```

В этом фрагменте кода мы обновляем данные уже имеющегося куба. Обратите внимание на то, что мы блокируем куб, чтобы предотвратить попытку одновременного обновления его несколькими пользователями.

Если OLAP-куб использует не только частные, но и коллективные измерения, последние должны быть обновлены до того, как начнется обновление самого куба. Дело в том, что коллективные измерения принадлежат самой базе данных, а не кубу. Если набор членов измерения изменился, но это еще не отражено в самом измерении из-за того, что его данные не обновлялись, то во время обновления куба произойдет исключение, связанное с несовпадением ожидаемого и реально имеющегося набора членов в измерении.

Ниже приведен фрагмент кода, иллюстрирующего сказанное выше:

```
procedure TForm1.Button1Click(Sender: TObject);
const
    //обновлять данные куба полностью
    ProcessFull = 1;
    //блокировать куб во время обновления
    olapLockProcess = 4;
var   Srv,dsoDB,dsoCube,dsoDim: variant;
      I           : Integer;
      OldCursor   : TCursor;
      dimName     : String;
begin
    Srv := CreateOleObject('DSO.Server');
    // соединяемся с сервером
    Srv.Connect(Edit1.Text);
    // указываем имя базы данных
```

```

dsoDB := srv.MDStores.Item(Edit2.Text);
// и имя куба
dsoCube := dsoDB.Cubes.Item(Edit3.Text);
OldCursor := Form1.Cursor;
Form1.Cursor := crHourGlass;
for I := 1 to dsoCube.Dimensions.Count do
begin
    dsoDim := dsoCube.Dimensions.Item(I);
    dimName := dsoCube.Dimensions.Item(I).Name;
    //если измерение коллективное, оно принадлежит базе данных, а не кубу
    if dsoDim.IsShared then begin
        //блокируем измерение
        dsoDB.Dimensions.Item(dimName).LockObject(olapLockProcess, 'Dimension is
processed');
        //обновляем измерение
        dsoDB.Dimensions.Item(dimName).Process(ProcessFull);
        //снимаем блокировку с измерения
        dsoDB.Dimensions.Item(dimName).UnlockObject;
    end;
end;
// блокируем куб
dsoCube.LockObject(olapLockProcess, 'Cube is processed');
// обновляем данные куба
dsoCube.Process (ProcessFull);
// снимаем блокировку
dsoCube.UnlockObject;
Form1.Cursor := OldCursor;
// возвращаем курсор в первоначальное состояние
dsoCube := Unassigned;
dsoDb := Unassigned;
Srv := Unassigned;
end;

```

Обратите внимание на то, что коллективные измерения также следует блокировать на время обновления во избежание одновременного обновления их несколькими пользователями.

## Работа с коллективными объектами

Еще одной задачей является создание объектов многомерных баз данных — самих баз данных, принадлежащих им описаний источников данных, коллективных измерений, кубов.

### Создание и удаление многомерных баз данных

Чтобы создать новую базу данных, следует применять метод AddNew коллекции MDStores объекта DSO Server, например:

```

Var
    dsoServer, dsoDB: Variant ;
    DbName: string;

...

InputQuery('New Database', 'Input Database Name', dbname);
DsoDB := dsoServer.MDStores.AddNew(dbName);

```

Для удаления базы данных применяется метод Remove коллекции MDStores объекта Server:

```

dsoServer.MDStores.Remove(dbName);

```

После создания многомерной базы данных необходимо определить для нее хотя бы один реляционный источник данных — в противном случае мы не сможем создавать измерения и кубы, поскольку их будет просто не из чего создать.

## Создание и удаление источников данных

Analysis Services для доступа к данным используют уже знакомый нам механизм ADO. Именно поэтому основной характеристикой источника данных является ADO Connection String.

Для создания описания источника данных следует использовать метод Add коллекции DataSources объекта Database:

```
var
  CS,DSname, DBname: string;
  DS, dsoDB: variant;

...

dsoDB := dsoServer.MDStores.Item(DBname);
// Стандартный диалог для получения ADO Connection string
CS := PromptDataSource(Form1.Handle, '');
InputQuery('New Datasource','Input Datasource Name',DSname);
// создаем объект DataSource
DS := CreateOleObject('DSO.DataSource');
// указываем его имя
DS.Name := DSname;
// устанавливаем его свойство ConnectionString
DS.ConnectionString := CS;
// добавляем описание источника данных
dsoDB.DataSources.Add(DS);
```

Обратите внимание: прежде чем добавлять объект DataSource в коллекцию DataSources объекта Database, следует определить его свойство ConnectionString.

Для удаления описания источника данных следует использовать метод Remove коллекции DataSources объекта DSO Database.

```
dsoDB.DataSources.Remove(DS);
```

Для соединения с источником данных следует использовать метод IsConnected объекта DSO DataSource. Если же соединение уже установлено, этот метод вернет True. Если же соединение не установлено, этот метод попытается осуществить соединение и вернет True в случае удачи и False в случае неудачи. При необходимости DSO автоматически устанавливает соединение с источником данных (например, когда требуется получить данные из него).

## Создание коллективных измерений

Коллективные измерения принадлежат многомерной базе данных, и каждое такое измерение в общем случае может быть использовано несколькими кубами. В объектной модели SQL DSO им соответствует коллекция Dimensions объекта Database. Для создания коллективного измерения следует использовать Add коллекции Dimensions объекта Database. Во фрагменте кода, приведенном ниже, предполагается, что список таблиц исходной реляционной базы данных, на основе которой формируется измерение, содержится в компоненте TableListBox, а источник данных, соответствующий этой базе данных, является I-м элементом в коллекции DataSources:

```
Var dimName : string;
    DsoDim : Variant;
```

```

I      : Integer;
...
DS := CreateOleObject('DSO.DataSource');
...
dsoDim := CreateOleObject('DSO.Dimension');
dimName := InputBox('New Shared Dimension', 'Input Dimension Name', 'Dimension1');
dsoDim.Name := dimName;
//устанавливаем свойство DataSource
dsoDim.DataSource := dsoDB.DataSources.Item(I+1);
//указываем размерную таблицу
dsoDim.FromClause := TableListBox.Items[TableListBox.ItemIndex];
//устанавливаем свойство JoinClause
dsoDim.JoinClause:= InputBox('Join clause', 'Input Join clause', '');
//добавляем измерение в базу данных
dsoDB.Dimensions.AddNew(dimname);

```

Прежде чем создать измерение, необходимо определить следующие свойства объекта Dimension:

DataSource — один из элементов коллекции DataSources объекта Database;

FromClause — имя размерной таблицы;

JoinClause — это свойство важно, если размерная таблица не совпадает с таблицей фактов, которая будет использована для создания куба. В этом случае нужно указать, как эти таблицы связаны друг с другом. В общем случае это свойство указывается в следующей форме:

```

<fact_table_name>.<field_name> =
<dimension_table_name>.<linked_field_name>

```

Иными словами, свойство JoinClause должно содержать описание связи между таблицами так, как описывается связь типа INNER JOIN в диалекте SQL, поддерживаемом данным OLE DB-провайдером.

В общем случае могут потребоваться значения и других свойств, например, при создании измерения с несбалансированной иерархией.

Любое измерение содержит как минимум один уровень. Соответственно объект Dimension обладает коллекцией Levels. Для добавления уровня к измерению следует применять метод AddNew этой коллекции. В приведенном ниже фрагменте кода предполагается, что компонент FieldListBox содержит список полей размерной таблицы:

```

dsoLev := dsoDim.Levels.AddNew(FieldListBox.
Items[FieldListBox.ItemIndex]);
dsoLev.MemberKeyColumn :=
TableListBox.Items[TableListBox.ItemIndex]+'.'+
//указываем поле, содержащее члены будущего
//измерения
FieldListBox.Items[FieldListBox.ItemIndex];
dsoLev.ColumnSize := 255;
dsoLev.ColumnType := adWChar;
dsoLev.EstimatedSize := 1;

```

Константа adWChar определена в файле ADODB\_TLB.pas, который можно получить, импортировав библиотеку типов msado15.dll.

Для создания уровня иерархии нам нужно определить следующие свойства объекта Level:



- MemberKeyColumn — имя поля размерной таблицы, которое содержит члены данного измерения;
- ColumnSize и ColumnType — размер и тип данных членов измерения;
- EstimatedSize — свойство, используемое DSO при создании агрегатов, в которых хранится данное измерение.

Если для создания измерения использовалось несколько таблиц, для объекта Level следует также определить свойство JoinClause в той же форме, в которой оно было определено для объекта Dimension.

После того как все уровни измерения определены, можно произвести заполнение его данными:

```
DsoDim.Process;
```

При изменении структуры измерения (например, при добавлении или удалении уровней) следует вызвать метод Update объекта Dimension для сохранения изменений, а затем — метод Process для обновления членов измерения.

## **Создание кубов**

Чтобы добавить к многомерной базе данных новый OLAP-куб, следует использовать метод Add коллекции Cubes объекта Database:

```
dsoDB.Cubes.AddNew(cubename);
```

Далее необходимо определить все его частные измерения и их уровни так, как это было описано в предыдущем разделе, и добавить эти измерения к кубу с помощью метода AddNew коллекции Dimensions объекта Cube например:

```
Var dsoCube: Variant;  
NewCube.Dimensions.AddNew(SharedDimListBox.  
Items[SharedDimListBox.ItemIndex]);
```

Создание частных измерений практически ничем не отличается от создания коллективных измерений. Единственное различие заключается в том, что в случае коллективного измерения соответствующий объект Dimension добавляется в коллекцию Dimensions объекта Database, а в случае частного измерения — в коллекцию Dimensions объекта Cube.

Наконец, следует определить меры куба. Для этой цели используется коллекция Measures объекта Cube:

```
NewMeasure:=CreateOleObject('DSO.Measure');  
NewMeasure.SourceColumn :=  
FieldListBox.Items[FieldListBox.ItemIndex];  
Mname := InputBox('New Measure',  
'Input Measure Name',NewMeasure.SourceColumn);  
NewMeasure.Name := Mname;  
dsoCube.Measures.AddNew(Mname);
```

Прежде чем добавить измерение, следует определить свойство SourceColumn объекта Measure. Это свойство должно указывать на числовое поле, на основании значений которого вычисляются агрегатные данные.

Определив метаданные куба, мы можем вычислить его данные с помощью уже знакомого нам метода Process объекта Cube.

Напомним, что при первоначальном вычислении или обновлении данных куба, использующего коллективные измерения, следует предварительно произвести ту же операцию с этими измерениями.

При изменении структуры куба (например, при добавлении или удалении измерений) необходимо вызвать метод Update объекта Cube для сохранения изменений, а затем — метод Process для обновления данных куба.

## **Поставка приложений, использующих SQL DSO**

Файлы библиотек DSO могут поставляться с использующими их приложениями. Ниже приведен их список:

- Msmddo80.dll — библиотека DSO;
- Msmdso.rll — файл ресурсов DSO;
- Msmdnet.dll — сетевой интерфейс Analysis Services;
- Msmdlock.dll — менеджер блокировок Analysis Services;
- Msmdso.dll — файл, требующийся для совместимости с Microsoft SQL Server version 7.0 OLAP Services.

Все файлы, за исключением Msmdso.rll, следует поместить в каталог

C:\Program Files\Common Files\Microsoft Shared\DSO

Все библиотеки должны быть зарегистрированы в реестре с помощью утилиты Regsvr32. Кроме того, каждый из файлов должен быть описан в ключе реестра:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SharedDLLs

Перед установкой DSO следует также установить Microsoft Data Access Components и PivotTable Services. Для этой цели на диске с дистрибутивом Microsoft SQL Server Analysis Services в каталоге \Msolap\Install\Pts имеются две программы установки. Одна из них, Ptslite.exe, предназначена для установки только PivotTable Service, а другая, Ptsfull.exe, — для установки PivotTable Service и Microsoft Data Access Components.