**Реферат: BD\_\_DW\_\_OLAP**

* [Реферат: работа по дисциплине “Базы данных” "Хранилища данных и технология olap "](https://domashke.net/referati/ostalnye-referaty/referat-rabota-po-discipline-bazy-dannyh-hranilishha-dannyh-i-tehnologiya-olap)

**Реферат: работа по дисциплине “Базы данных” "Хранилища данных и технология olap ":**

Московский государственный институт электроники и математики

Кафедра электронно-вычислительной аппаратуры

Курсовая работа по дисциплине “Базы данных”

**"Хранилища данных и технологияOLAP"**

Выполнили: студенты группы № С-54

Баталов А.В.

Воробьев Д.Н.

Новиков Р.О.

Овсиенко А.Ю.

Руководитель:

д.т.н. профессор Зарудный Д.И.

Москва 2006

**Аннотация**

В представленной курсовой работе рассматриваются проблемы и подходы к проектированию многомерных баз данных. В работе освещена технология многомерного анализа данных OLAP, рассматриваются концепции хранилищ данных, требования к многомерным базам данных и OLAP-средствам, логическую организацию OLAP-данных, применение OLAP технологий, а также основные термины и понятия, применяемые при обсуждении многомерного анализа.

**Содержание**

|  |
| --- |
| стр. |
| Аннотация | 2 |
| Содержание | 3 |
| Введение | 4 |
| 1. Постановка задачи | 5 |
| 2. Краткие сведения | 6 |
| 3. Основы концепции OLAP | 8 |
| 4. Проектирование многомерных баз данных | 20 |
| 5. Пример создания многомерной базы данных с помощью Microsoft Analysis Services | 33 |
| 6. OLAP клиенты | 47 |
| 7. Применение OLAP технологий | 57 |
| Заключение | 60 |
| Литература | 61 |

**Введение**

На сегодняшний день многие организации пришли к пониманию того факта, что наличие своевременной и объективной информации о рынке, прогнозе его перспектив, оценки деятельности фирмы и взаимоотношений с партнерами, является ключевым фактором в конкурентной борьбе. Исходя из этого, все больше и больше внимания уделяется средствам реализации и построения систем аналитической обработки информации. В первую очередь это касается систем управления многомерными базами данных, которые рассмотрены в данной работе.

**1. Постановка задачи**

Исследование технологии OLAP, её возможностей и применения многомерных хранилищ данных. Рассматриваются основные подходы к проектированию и созданию хранилищ данных с помощью Oracle Express. В качестве практической части рассматривается создания OLAP-куба средствами Microsoft Analysis Services. Также рассмотрены возможности Microsoft Excel, как OLAP клиента.

**2. Краткие сведения**

В отличие от традиционных реляционных СУБД, концепция OLAP не так широко известна. OLAP - это не отдельно взятый программный продукт, не язык программирования и даже не конкретная технология. Если постараться охватить OLAP во всех его проявлениях, то это совокупность концепций, принципов и требований, лежащих в основе программных продуктов, облегчающих аналитикам доступ к данным. Термин "OLAP" неразрывно связан с термином "хранилище данных" (Data Warehouse).Назначение хранилищ данных - предоставление пользователям информации для статистического анализа и принятия управленческих решений.

Приведем определение, сформулированное основателем концепции хранилищ данных Биллом Инмоном: "Хранилище данных - это предметно-ориентированное, привязанное ко времени и неизменяемое собрание данных для поддержки процесса принятия управляющих решений". Ральф Кимбалл (Ralph Kimball), один из авторов концепции хранилищ данных, описывал хранилище данных как "место, где люди могут получить доступ к своим данным. 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 удается. Поэтому для реализации хранилищ данных обычно используется несколько продуктов, одни их которых представляют собой собственно средства хранения данных, другие — средства их извлечения и просмотра, третьи — средства их пополнения и т.д.Хранилища данных должны обеспечивать высокую скорость получения данных, возможность получения и сравнения так называемых срезов данных, а также непротиворечивость, полноту и достоверность данных.

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.

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 - причем 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). 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. Эта технология основана на построении многомерных наборов данных — OLAP-кубов, оси которого содержат параметры, а ячейки — зависящие от них агрегатные данные. Концепция OLAP была описана в 1993 году Эдгаром Коддом, известным исследователем баз данных и автором реляционной модели данных.

В 1995 году на основе требований, изложенных Коддом, был сформулирован так называемый тест FASMI (Fast Analysis of Shared Multidimensional Information — быстрый 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.Приложения с OLAP-функциональностью должны предоставлять пользователю результаты анализа за приемлемое время, осуществлять логический и статистический анализ, поддерживать многопользовательский доступ к данным, осуществлять многомерное концептуальное представление данных и иметь возможность обращаться к любой нужной информации.

**3. Основы концепцииOLAP**

Рассмотрим подробнее 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 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.CompanyNameAS

CustomerName, Invoices.Salesperson,

Invoices.OrderDate, Categories.CategoryName,

Invoices.ProductName,

Invoices.Shippers.CompanyNameAS

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.



**Рис. 3.1. 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.

|  |  |
| --- | --- |
| **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 |
| 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):

|  |
| --- |
| **ShipperName** |
| **Country** | **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 |
| 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. 3.2. Трехмерный набор агрегатных данных**

Ячейки куба, 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. 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. 3.3. Иерархия в измерении, связанном с географическим положением клиентов**

Отметим, что иерархии могут быть сбалансированными (balanced), как, например, 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):



**Рис. 3.4. 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. 3.5. "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.

Все, что говорилось выше про OLAP, по сути, относилось к многомерному представлению данных. То, как данные хранятся, грубо говоря, не волнует ни конечного пользователя, ни разработчиков инструмента, которым клиент пользуется.

Многомерность в OLAP-приложениях может быть разделена на три уровня:

* Многомерное представление данных - средства конечного пользователя, обеспечивающие многомерную визуализацию и манипулирование данными; слой многомерного представления абстрагирован от физической структуры данных и воспринимает данные как многомерные.
* Многомерная обработка - средство (язык) формулирования многомерных запросов (традиционный реляционный язык SQL здесь оказывается непригодным) и процессор, умеющий обработать и выполнить такой запрос.
* Многомерное хранение - средства физической организации данных, обеспечивающие эффективное выполнение многомерных запросов.

Первые два уровня в обязательном порядке присутствуют во всех OLAP-средствах. Третий уровень, хотя и является широко распространенным, не обязателен, так как данные для многомерного представления могут извлекаться и из обычных реляционных структур; процессор многомерных запросов в этом случае транслирует многомерные запросы в SQL-запросы, которые выполняются реляционной СУБД. Конкретные OLAP-продукты, как правило, представляют собой либо средство многомерного представления данных, OLAP-клиент (например, Pivot Tables в Excel 2000 фирмы Microsoft или ProClarity фирмы Knosys), либо многомерную серверную СУБД, OLAP-сервер (например, Oracle Express Server или Microsoft OLAP Services). Слой многомерной обработки обычно бывает встроен в OLAP-клиент и/или в OLAP-сервер, но может быть выделен в чистом виде, как, например, компонент Pivot Table Service фирмы Microsoft.

Средства OLAP-анализа могут извлекать данные и непосредственно из реляционных систем. Такой подход был более привлекательным в те времена, когда OLAP-серверы отсутствовали в прайс-листах ведущих производителей СУБД. Но сегодня и Oracle, и Informix, и Microsoft предлагают полноценные OLAP-серверы. OLAP-серверы, или серверы многомерных БД, могут хранить свои многомерные данные по-разному. Прежде чем рассмотреть эти способы, нам нужно поговорить о таком важном аспекте, как хранение агрегатов. Дело в том, что в любом хранилище данных - и в обычном, и в многомерном - наряду с детальными данными, извлекаемыми из оперативных систем, хранятся и суммарные показатели (агрегированные показатели, агрегаты), такие, как суммы объемов продаж по месяцам, по категориям товаров и т. п. Агрегаты хранятся в явном виде с единственной целью - ускорить выполнение запросов. Ведь, с одной стороны, в хранилище накапливается, как правило, очень большой объем данных, а с другой - аналитиков в большинстве случаев интересуют не детальные, а обобщенные показатели. И если каждый раз для вычисления суммы продаж за год пришлось бы суммировать миллионы индивидуальных продаж, скорость, скорее всего, была бы неприемлемой. Поэтому при загрузке данных в многомерную БД вычисляются и сохраняются все суммарные показатели или их часть.

Но за скорость обработки запросов к суммарным данным приходится платить увеличением объемов данных и времени на их загрузку. Причем увеличение объема может стать буквально катастрофическим - в одном из опубликованных стандартных тестов полный подсчет агрегатов для 10 Мб исходных данных потребовал 2,4 Гб, т. е. данные выросли в 240 раз. Степень "разбухания" данных при вычислении агрегатов зависит от количества измерений куба и структуры этих измерений, т. е. соотношения количества "отцов" и "детей" на разных уровнях измерения. Для решения проблемы хранения агрегатов применяются подчас сложные схемы, позволяющие при вычислении далеко не всех возможных агрегатов достигать значительного повышения производительности выполнения запросов. Теперь о различных вариантах хранения информации. Как детальные данные, так и агрегаты могут храниться либо в реляционных, либо в многомерных структурах. Многомерное хранение позволяет обращаться с данными как с многомерным массивом, благодаря чему обеспечиваются одинаково быстрые вычисления суммарных показателей и различные многомерные преобразования по любому из измерений. Некоторое время назад OLAP-продукты поддерживали либо реляционное, либо многомерное хранение. Сегодня, как правило, один и тот же продукт обеспечивает оба этих вида хранения, а также третий вид - смешанный. Применяются следующие термины:

* **MOLAP**(Multidimensional OLAP) - и детальные данные, и агрегаты хранятся в многомерной БД. В этом случае получается наибольшая избыточность, так как многомерные данные полностью содержат реляционные.
* **ROLAP**(Relational OLAP) - детальные данные остаются там, где они "жили" изначально - в реляционной БД; агрегаты хранятся в той же БД в специально созданных служебных таблицах.
* **HOLAP**(Hybrid OLAP) - детальные данные остаются на месте (в реляционной БД), а агрегаты хранятся в многомерной БД.

Каждый из этих способов имеет свои преимущества и недостатки и должен применяться в зависимости от условий - объема данных, мощности реляционной СУБД и т. д. При хранении данных в многомерных структурах возникает потенциальная проблема "разбухания" за счет хранения пустых значений. Ведь если в многомерном массиве зарезервировано место под все возможные комбинации меток измерений, а реально заполнена лишь малая часть (например, ряд продуктов продается только в небольшом числе регионов), то бо/льшая часть куба будет пустовать, хотя место будет занято. Современные OLAP-продукты умеют справляться с этой проблемой.

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.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.



**Рис. 3.6. Структура базы данных Northwind**

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

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.





**Рис. 3.6. Пример таблицы фактов**

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. Заметим, 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 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) 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, содержавшиеся 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.





**Рис. 3.7. 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).



**Рис. 3.8. 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).



**Рис. 3.9. 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 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. 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.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 умещаться 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.

**4. Проектирование многомерных баз данных**

Сегодня все большее число организаций приходит к пониманию того, что без наличия своевременной и объективной информации о состоянии рынка, прогнозирования его перспектив, постоянной оценки эффективности функционирования собственных структур и анализа взаимоотношений с бизнес-партнерами и конкурентами их дальнейшее развитие становится практически невозможным. Поэтому не удивительно то внимание, которое сегодня уделяется средствам реализации и концепциям построения информационных систем, ориентированных на аналитическую обработку данных. И в первую очередь это касается систем управления базами данных, основанными на многомерном подходе (далее МСУБД).

Следует заметить, что МСУБД не являются изобретением девяностых годов, а сам многомерный подход возник практически одновременно и параллельно с реляционным. Еще в начале семидесятых годов консалтинговой фирмой Management Decision System (позже преобразованной в IRI Software) были реализованы первые версии многомерных инструментальных средств. Позднее эти средства стали известны как IRI Multidimensional DBMS, IRI Express Server и с 1995 г. - Oracle Express Server. И хотя к 1995 г. у фирмы IRI Software было уже более тысячи корпоративных пользователей, и она имела более двадцати представительств в Европе, Азии и Латинской Америке, все же МСУБД долгое время оставались в тени своего "старшего" собрата РСУБД. И только начиная с середины девяностых годов, а точнее с 1993 г., интерес к МСУБД начал приобретать всеобщий характер. Именно в этом году появилась новая программная статья одного из основоположников реляционного подхода Э. Кодда, в которой он сформулировал 12 основных требований к средствам реализации OLAP и произвел анализ некоторых как субъективных, так и вполне объективных недостатков реляционного подхода, затрудняющих его использование в задачах, требующих сложной аналитической обработки данных.

|  |  |
| --- | --- |
| **Многомерное представление данных** | Средства должны поддерживать многомерный на концептуальном уровне взгляд на данные. |
| **Прозрачность** | Пользователь не должен знать о том, какие конкретные средства используются для хранения и обработки данных, как данные организованы и откуда они берутся. |
| **Доступность** | Средства должны сами выбирать и связываться с наилучшим для формирования ответа на данный запрос источником данных. Средства должны обеспечивать автоматическое отображение их собственной логической схемы в различные гетерогенные источники данных. |
| **Согласованная производительность** | Производительность практически не должна зависеть от количества Измерений в запросе. |
| **Поддержка архитектуры клиент-сервер** | Средства должны работать в архитектуре клиент-сервер. |
| **Равноправность всех измерений** | Ни одно из измерений не должно быть базовым, все они должны быть равноправными (симметричными). |
| **Динамическая обработка разреженных матриц** | Неопределенные значения должны храниться и обрабатываться наиболее эффективным способом. |
| **Поддержка многопользовательского режима работы с данными** | Средства должны обеспечивать возможность работать более чем одному пользователю. |
| **Поддержка операций на основе различных измерений** | Все многомерные операции (например Агрегация) должны единообразно и согласованно применяться к любому числу любых измерений. |
| **Простота манипулирования данными** | Средства должны иметь максимально удобный, естественный и комфортный пользовательский интерфейс. |
| **Развитые средства представления данных** | Средства должны поддерживать различные способы визуализации (представления) данных. |
| **Неограниченное число измерений и уровней агрегации данных** | Не должно быть ограничений на число поддерживаемых Измерений. |

**Таблица 4.1. Двенадцать правил оценки средств для OLAP**

Многомерное представление данных и OLAP уже стали сегодня одними из наиболее широко распространенных концепций построения аналитических систем.При первом знакомстве с многомерным подходом к организации данных достаточно часто возникают два противоречивых вопроса: Для чего собственно нужны МСУБД и нужно ли тратить время и средства на их освоение и приобретение, если все те же задачи можно решить и средствами традиционных РСУБД? И обратный: Почему МСУБД ограничивают себя исключительно приложениями, ориентированными на анализ данных и почему бы на их основе не реализовывать традиционные системы оперативной обработки данных? И, несмотря на то, что эти вопросы выражают противоположные точки зрения, ответ на них звучит приблизительно одинаково: "Главное достоинство МСУБД состоит именно в том, что они узко специализированны и область их применения - интерактивная аналитическая обработка агрегированных исторических и прогнозируемых данных".

Пользователя, занимающегося анализом, редко интересуют детализированные данные. Более того, чем выше уровень пользователя (руководителя, управляющего, аналитика), тем выше уровень агрегации данных, используемых им для принятия решения. Рассмотрим в качестве примера фирму по продаже автомобилей. Коммерческого директора такой фирмы мало интересует вопрос: "Какого цвета "Жигули" успешнее всего продает один из ее менеджеров - Петров: белого или красного?" Для него важно, какие модели и какие цвета предпочитают в данном регионе. Его также мало интересует детализация на уровне контракта, часа или даже дня. Например, если выяснится, что "ВАЗ2108 Красного цвета" чаще покупают в утренние часы, этот факт скорее заинтересует психиатра, а не коммерческого аналитика. Для правильного формирования склада ему важна и необходима информация на уровне декады, месяца или даже квартала.

Важнейшим свойством данных в аналитических задачах является их Исторический характер. После того как зафиксировано, что Петров в июне 1996 г. продал 2 автомобиля "Волга" и 12 автомобилей "Жигули", данные об этом событии становятся историческим (свершившимся) фактом. И после того, как информация об этом факте получена, верифицирована и заведена в БД, она может быть сколько угодно раз считана оттуда, но уже не может и не должна быть изменена. Историчность данных предполагает не только высокий уровень статичности (неизменности) как собственно данных (например:Петров продал в 1995 г. 51 автомобиль "Жигули ВАЗ2105"), так и их взаимосвязей (например:в 1995 г. Петров работал в Восточном Регионе; в 1995 г. продавались автомобили модели ВАЗ2105). А это, в свою очередь, дает возможность использовать специализированные, основанные на предположении о статичности данных и их взаимосвязей методы загрузки, хранения, индексации и выборки.

Другим неотъемлемым свойством Исторических данных является обязательная спецификация Времени, которому эти данные соответствуют. Причем Время является не только наиболее часто используемым критерием выборки, но и одним из основных критериев, по которому данные упорядочиваются в процессе обработки и представления пользователю. А это накладывает соответствующие требования как на используемые механизмы хранения и доступа:

· для уменьшения времени обработки запросов желательно, чтобы уже в БД данные хранились (были предварительно отсортированы) в том порядке, в котором они наиболее часто запрашиваются;

так и на языки описания и манипулирования данными, например:

· во многих организациях используются как общепринятые, так и собственные календарные циклы (финансовый год может начинаться не в январе как календарный, а, например, в июне);

· время является стандартным параметром практически любой аналитической, статистической или финансовой функции (прогноз, нарастающий итог, переходящий запас, скользящее среднее и т.д.).

Когда говорится о неизменности и статичности данных в аналитических системах, имеется в виду неизменность исключительно Исторических данных (данных, описывающих уже произошедшие события). Такое предположение ни в коем случае не распространяется на Прогнозируемые данные (данные о событии, которое еще не происходило). И этот момент является весьма существенным.

Например, если мы строим прогноз об объеме продаж на июнь 1997 г. для менеджера Петрова, то, по мере поступления фактических (Исторических) данных за 1996 г., эта цифра может и будет многократно изменяться и уточняться. Более того, достаточно часто прогнозирование и моделирование затрагивает не только будущие, еще не произошедшие, но и прошлые, уже свершившиеся события. Например, анализ: "а, что будет (было бы)... если (бы)..?", строится на предположении о том, что значения некоторых данных, в том числе и из прошлого, отличны от реальных. И для ответа на вопрос: "Какой был бы Прогноз по объему продаж автомобилей "Волга" для менеджера Петрова на июнь 1997 г., если бы объем продаж "Волг" в июне 1996 г. у него возрос на тот же процент, что объем продаж "Жигулей"?". Потребуется не только вычислить новое, еще не существующее значение Объема Продаж, для еще не наступившего июня 1997 г., но и предварительно вычислить гипотетическое значение Объема продаж, за уже прошедший июнь 1996 г.

На первый взгляд, мы сами противоречим себе, говоря о неизменности данных, как основополагающем свойстве аналитической системы. Но это не так. Это кажущееся противоречие наоборот подчеркивает и усиливает значимость требований к неизменности Исторических данных. Сколько бы мы не упражнялись (например, при анализе: "а что... если..?") со значением объема продаж за июнь 1996 г., значения Исторических (реальных) данных должны оставаться неизменными. Конечно, предположение о неизменности не означает невозможности исправления ошибок, если они были обнаружены в Исторических данных. В свою очередь, к оперативным данным, отражающим состояние некоторой предметной области в данный текущий момент времени, не применимы такие понятия, как прошлое или будущее. Для них существует единственное понятие - сейчас, а их основное назначение - адекватное детализированное отображение текущих событий (изменений), происходящих в реальном мире.Например:

· менеджер Петров продал еще одни "Жигули ВАЗ2106";

· менеджера Петрова перевели из Восточного филиала фирмы в Западный.

Вместе с тем изменчивость Оперативных данных ни в коем случае не подразумевает их близость по свойствам к Прогнозируемым данным. Между ними существует коренное различие. Оперативным данным, в отличие от Прогнозируемых, присуще свойство общезначимости, и обычно все пользователи работают с одним и тем же экземпляром данных. После того как в оперативную систему заведены данные о том, что Петров продал еще один автомобиль, эта информация сразу же должна стать доступной всем заинтересованным в ней пользователям. Причем до тех пор, пока это изменение не зафиксировано, ни какой другой пользователь не имеет права изменять строку с информацией о продажах Петрова.

Существенно иная ситуация с Прогнозируемыми данными. Они носят, скорее, личностный (индивидуальный) характер. Вполне реальна ситуация, когда коммерческий директор фирмы и управляющий региональным отделением одновременно решили получить прогноз возможного объема продаж на 1997 г. для Петрова. Однако каждый из них делает собственный прогноз. Каждый из них может использовать свои функции прогнозирования, и, даже если применяется один и тот же метод (или функция), прогноз может основываться на различных исторических интервалах, и результаты, по всей вероятности, будут различны. Поэтому каждый из них работает с собственным экземпляром Прогнозируемых данных (хотя эти данные и относятся формально к одной и той же личности, виду деятельности и времени), и эти данные не должны смешиваться. Конечно, вполне вероятно, что один из этих вариантов будет принят в качестве плановых показателей для Петрова. Но после того, как Прогноз утвержден в качестве Плана, данные просто перейдут в другую категорию и станут Историческими. Следует заметить, что в области информационных технологий всегда существовало два взаимодополняющих друг друга направления развития:

· системы, ориентированные на оперативную (транзакционную или операционную) обработку данных;

· системы, ориентированные на анализ данных - системы поддержки принятия решений.

И практически до настоящего времени, когда говорилось о стремительном росте числа реализаций информационных систем, прежде всего имелись в виду системы, предназначенные исключительно для оперативной обработки данных. Именно для этого изначально и создавались и на это были ориентированы РСУБД, которые сегодня стали основным средством построения информационных систем самого различного масштаба и назначения. Но, являясь высокоэффективным средством реализации систем оперативной обработки данных, РСУБД оказались менее эффективными в задачах аналитической обработки. Конечно, средствами традиционных РСУБД и на основании данных, хранящихся в реляционной БД, можно построить заранее регламентированный аналитический отчет и даже Прогноз об ожидаемом объеме продаж автомобилей на следующий год.

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Статический анализ** | **Динамический анализ** |
| **Типы вопросов** | Сколько? Как? Когда? | Почему? Что будет если? |
| **Время отклика** | Не регламентируется | Секунды |
| **Типичные операции** | Регламентированный отчет, диаграмма | Последовательность интерактивных отчетов, диаграмм, экранных форм; динамическое изменение уровней агрегации и срезов данных |
| **Уровень аналитических требований** | Средний | Высокий |
| **Тип экранных форм** | В основном определенный заранее, регламентированный | Определяемый пользователем |
| **Уровень агрегации данных** | Детализированные и суммарные | В основном суммарные |
| **Возраст данных** | Исторические и текущие | Исторические, текущие и прогнозируемые |
| **Типы запросов** | В основном предсказуемые | Непредсказуемые, от случаю к случаю |
| **Назначение** | Работа с историческими и текущими данными, регламентированная аналитическая обработка и построение прогнозов | Работа с историческими, текущими и прогнозируемыми данными. Многопроходный анализ, моделирование |

**Таблица 4.2.Сравнение характеристик статического (регламентированного) и динамического анализа**

Но, как правило, после просмотра такого отчета у пользователя (аналитика) появится не готовый ответ, а новая серия вопросов. Однако, если бы ему захотелось получить ответ на новый вопрос, он может ждать его часы, а иногда и дни. Обычно каждый новый непредусмотренный заранее запрос должен быть сначала формально описан, передан программисту, запрограммирован и, наконец, выполнен. Но после того, как аналитик получит долгожданный ответ, достаточно часто оказывается, что решение не могло ждать и оно уже принято, или что случается еще чаще, произошло взаимное непонимание и получен ответ на не совсем тот вопрос. Впрочем, не намного меньшее время затрачивается и на получение ответа и на заранее описанный и запрограммированный запрос.

Более того, для решения большинства аналитических задач, скорее всего, потребуется использование внешних по отношению к РСУБД, специализированных инструментальных средств. Выполнение большинства аналитических функций (например построение прогноза) невозможно без предположения об упорядоченности данных. Но в РСУБД предполагается, что данные в БД не упорядочены (или, более точно, упорядочены случайным образом). Естественно, здесь имеется возможность после выборки данных из БД выполнить их сортировку и затем аналитическую функцию. Но это потребует дополнительных затрат времени на сортировку. Сортировка должна будет проводиться каждый раз при обращении к этой функции, и, самое главное, такая функция может быть определена и использована только во внешнем по отношению к РСУБД пользовательском приложении и не может быть встроенной функцией языка SQL.

Не менее важно и то, что многие критически необходимые для оперативных систем функциональные возможности, реализуемые в РСУБД, являются избыточными для аналитических задач. Например, в аналитических системах данные обычно загружаются достаточно большими порциями из различных внешних источников (оперативных БД, заранее подготовленных плоских файлов, электронных таблиц). И, как правило, время и последовательность работ по загрузке, резервированию и обновлению данных могут быть спланированы заранее. Поэтому в таких системах обычно не требуются и, соответственно, не предусматриваются, например, развитые средства обеспечения целостности, восстановления и устранения взаимных блокировок и т.д. А это не только существенно облегчает и упрощает сами средства реализации, но и значительно снижает внутренние накладные расходы и, следовательно, повышает производительность при выполнении их основной целевой функции - поиске и выборке данных.

|  |  |  |
| --- | --- | --- |
| **Характеристика** | Оперативные | Аналитические |
| **Частота обновления** | Высокая частота, маленькими порциями | Малая частота, большими порциями |
| **Источники данных** | В основном внутренние | В основном внешние (по отношению к аналитической системе) |
| **Возраст данных** | Текущие (за период от нескольких месяцев до одного года) | В основном исторические (за период в несколько лет, десятки лет) и прогнозируемые |
| **Уровень агрегации данных** | Детализированные данные | В основном агрегированные данные |
| **Назначение** | Фиксация, оперативный поиск и обработка данных | Работа с историческими данными, аналитическая обработка, прогнозирование и моделирование |

**Таблица 4.3.Характеристики данных в системах, ориентированных на оперативную и аналитическую обработку данных.**

"Многомерный взгляд на данные наиболее характерен для пользователя, занимающегося анализом данных" - это утверждение сегодня стало уже почти аксиомой. Однако, что такое многомерное представление, откуда появляется многомерность в трехмерном мире, чем оно отличается и чем оно лучше ставшего уже привычным реляционного представления? И наконец, откуда среди нас появились люди, мыслящие в четырех и более измерениях, и как это им удается - именно эти вопросы возникают практически у любого, впервые прочитавшего это утверждение. На самом деле все сказанное в этом утверждении - чистая правда, и пользователю, занимающемуся анализом, действительно присуща многомерность мышления. Весь вопрос в том, что понимать под Измерением.

Достаточно очевидно, что даже при небольших объемах данных отчет, представленный в виде двухмерной таблицы (Модели автомобиля по оси Y и Время по оси X), нагляднее и информативнее отчета с реляционной построчной формой организации.

**Реляционная модель**

|  |  |  |
| --- | --- | --- |
| **Модель** | **Месяц** | **Объем** |
| "Жигули""Жигули""Жигули""Москвич""Москвич""Волга" | ИюньИюльАвгустИюньИюльИюль | 1224521819 |

**Многомерная модель**

|  |  |  |
| --- | --- | --- |
| **Июнь** | **Июль** | **Август** |
| "Жигули""Москвич""Волга" | 122No | 241819 | 5NoNo |

**Рис. 4.1.Реляционная и многомерная модели представления данных.**

А теперь представим, что у нас не три модели, а 30 и не три, а 12 различных месяцев. В случае построчного (реляционного) представления мы получим отчет в 360 строк (30х12), который займет не менее 5-6 страниц. В случае же многомерного (в нашем случае двухмерного) представления мы получим достаточно компактную таблицу 12 на 30, которая вполне уместится на одной странице и которую, даже при таком объеме данных, можно реально оценивать и анализировать.

И когда говорится о многомерной организации данных, вовсе не подразумевается то, что данные представляются конечному пользователю (визуализируются) в виде четырех или пятимерных гиперкубов. Это невозможно, да и пользователю более привычно и комфортно иметь дело с двухмерным табличным представлением и двухмерной бизнес-графикой.

Закономерен вопрос: "Где же здесь многомерность, откуда она берется и куда исчезает?" Ответ прост. Когда говорится о многомерности, имеется в виду не многомерность визуализации, а многомерное представление при описании структур данных и поддержка многомерности в языках манипулирования данными.

Основными понятиями, с которыми оперирует пользователь и проектировщик в многомерной модели данных, являются:

· измерение (Dimension);

· ячейка (Cell).

Иногда вместо термина "Ячейка" используется термин "Показатель" (Measure).

Измерение - это множество однотипных данных, образующих одну из граней гиперкуба. Например - Дни, Месяцы, Кварталы, Годы - это наиболее часто используемые в анализе временные Измерения. Примерами географических измерений являются: Города, Районы, Регионы, Страны и т.д. В многомерной модели данных Измерения играют роль индексов, используемых для идентификации конкретных значений (Показателей), находящихся в Ячейках гиперкуба. В свою очередь, Показатель - это поле (обычно цифровое), значения которого однозначно определяются фиксированным набором Измерений. В Oracle Express Server, в зависимости от того, как формируются его значения, Показатель может быть определен, как:

· Переменная (Variable) - значения таких Показателей один раз вводятся из какого-либо внешнего источника или формируются программно и затем в явном виде хранятся в многомерной базе данных (МБД);

· Формула (Formula) - значения таких Показателей вычисляются по некоторой заранее специфицированной формуле. То есть для Показателя, имеющего тип Формула, в БД хранится не его значения, а формула, по которой эти значения могут быть вычислены.

Заметим, что это различие существует только на этапе проектирования и полностью скрыто от конечных пользователей. В примере каждое значение поля Объем продаж однозначно определяется комбинацией полей:

· Модель автомобиля;

· Месяц продаж.

Но в реальной ситуации для однозначной идентификации значения Показателя, скорее всего, потребуется большее число измерений, например:

· Модель автомобиля;

· Менеджер;

· Время (например Год).

**Измерения:**

· Время (Год) - 1994, 1995, 1995

· Менеджер - Петров, Смирнов, Яковлев

**Показатель:**

· Объем Продаж



**Рис. 4.2.Измерения и Показатели (Ячейки)**

И в терминах многомерной модели речь будет идти уже не о двухмерной таблице, а о трехмерном гиперкубе:

· первое Измерение - Модель автомобиля;

· второе Измерение - Менеджер, продавший автомобиль;

· третье Измерение - Время (Год);

на пересечении граней которого находятся значения ПоказателяОбъем продаж.

Заметим, что, в отличие от Измерений, не все значения Показателей должны иметь и имеют реальные значения. Например, МенеджерПетровв 1994 г. мог еще не работать в фирме, и в этом случае все значения ПоказателяОбъем продажза этот год будут иметь неопределенные значения.



**Рис. 5.3.Неопределенные значения показателей**

В различных МСУБД используются два основных варианта организации данных:

· Гиперкубическая модель;

· Поликубическая модель.

В чем состоит разница? Системы, поддерживающие Поликубическую модель (примером является Oracle Express Server), предполагают, что в МБД может быть определено несколько гиперкубов с различной размерностью и с различными Измерениями в качестве их граней. Например, значение Показателя Рабочее Время Менеджера, скорее всего, не зависит от Измерения Модель Автомобиля и однозначно определяется двумя Измерениями: День и Менеджер. В Поликубической модели в этом случае может быть объявлено два различных гиперкуба:

· Двухмерный - для Показателя Рабочее Время Менеджера;

· Трехмерный - для Показателя Объем Продаж.

В случае же Гиперкубической модели предполагается, что все Показатели должны определяться одним и тем же набором Измерений. То есть только из-за того, что Объем Продаж определяется тремя Измерениями, при описании Показателя Рабочее Время Менеджера придется также использовать три Измерения и вводить избыточное для этого Показателя Измерение Модель Автомобиля.

Пользователя редко интересуют все потенциально возможные комбинации значений Измерений. Более того, он практически никогда не работает одновременно сразу со всем гиперкубом данных. Подмножество гиперкуба, получившееся в результате фиксации значения одного или более Измерений, называется Срезом (Slice). Например, если мы ограничим значение Измерения Модель Автомобиля = "ВАЗ2108", то получим подмножество гиперкуба (в нашем случае - двухмерную таблицу), содержащее информацию об истории продаж этой модели различными менеджерами в различные годы. Изменение порядка представления (визуализации) Измерений (обычно применяется при двухмерном представлении данных) называется Вращением (Rotate). Эта операция обеспечивает возможность визуализации данных в форме, наиболее комфортной для их восприятия. Например, если менеджер первоначально вывел отчет, в котором Модели автомобилей были перечислены по оси X, а Менеджеры по оси Y, он может решить, что такое представление мало наглядно, и поменять местами координаты (выполнить Вращение на 90 градусов). В нашем примере значения Показателей определяются только тремя измерениями. На самом деле их может быть гораздо больше и между их значениями обычно существуют множество различных Отношений (Relation) типа "один ко многим". Например, каждый Менеджер может работать только в одном подразделении, а каждой модели автомобиля однозначно соответствует фирма, которая ее выпускает:

· Менеджер ->Подразделение;

· Модель Автомобиля ->Фирма-Производитель.

Заметим, что для Измерений, имеющих тип Время (таких как День, Месяц, Квартал, Год), все Отношения устанавливаются автоматически, и их не требуется описывать. В свою очередь, множество Отношений может иметь иерархическую структуру - Иерархические Отношения (Hierarchical Relationships). Вот только несколько примеров таких Иерархических Отношений:

· День -> Месяц -> Квартал -> Год;

· Менеджер -> Подразделение -> Регион -> Фирма -> Страна;

· Модель Автомобиля -> Завод-Производитель -> Страна.

И часто более удобно не объявлять новые Измерения и затем устанавливать между ними множество Отношений, а использовать механизм Иерархических Отношений. В этом случае все потенциально возможные значения из различных Измерений объединяются в одно множество. Например, мы можем добавить к множеству значений Измерения Менеджер ("Петров", "Сидоров", "Иванов", "Смирнов"), значения Измерения Подразделение ("Филиал 1", "Филиал 2", "Филиал 3") и Измерения Регион ("Восток", "Запад") и затем определить между этими значениями Отношение Иерархии. Например:

|  |  |
| --- | --- |
| "Восток""Запад""Филиал 1""Филиал 2""Филиал 3""Петров""Сидоров""Иванов""Смирнов" | "NA""NA""Восток""Восток""Запад""Филиал 1""Филиал 1""Филиал 2""Филиал 3" |

С точки зрения пользователя, Подразделение, Регион, Фирма, Страна являются точно такими же Измерениями, как и Менеджер. Но каждое из них соответствует новому, более высокому уровню агрегации значений Показателя Объем продаж. В процессе анализа пользователь не только работает с различными Срезами данных и выполняет их Вращение, но и переходит от детализированных данных к агрегированным, т.е. производит операцию Агрегации (Drill Up). Например, посмотрев, насколько успешно в 1995 г. Петров продавал модели "Жигули" и "Волга", управляющий может захотеть узнать, как выглядит соотношение продаж этих моделей на уровне Подразделения, где Петров работает. А затем получить аналогичную справку по Региону или Фирме. Переход от более агрегированных к более детализированным данным называется операцией Детализации (Drill Down). Например, начав анализ на уровне Региона, пользователь может захотеть получить более точную информацию о работе конкретного Подразделения или Менеджера.Основное назначение МСУБД - реализация систем, ориентированных на динамический, многомерный анализ исторических и текущих данных, анализ тенденций, моделирование и прогнозирование будущего. Причем такие системы в большой степени ориентированы на обработку произвольных, заранее не регламентированных запросов, и при их разработке фактически отсутствует этап проектирования регламентированных пользовательских приложений (наиболее ответственный и трудоемкий в традиционных оперативных системах).

Проектирование МБД обычно начинается с определения вопросов, с которыми конечные пользователи хотели бы обратиться к системе. Причем на этом этапе интерес представляют даже не сами тексты вопросов, а понимание того, о каких личностях, местах, событиях и объектах в них спрашивается.

|  |  |  |  |
| --- | --- | --- | --- |
| **Подразделение** | **Менеджер** | **Временной интервал** | **Вопрос** |
| Отдел | Петров | 3 года | На сколько процентов увеличились продажи "Жигулей" в Западном регионе после январской рекламной кампании в еженедельнике "Западный Вестник"? |
| Финансовый отдел | Смирнов | 5 лет | Какие региональные подразделения превысили в третьем квартале запланированные расходы на командировки и как это соотносится с ростом их прибыли (в абсолютных и относительных величинах)? |
| Коммерческий отдел | Левшин | 10 лет | Какие два варианта скидок наиболее эффективны в Западном регионе в летний период при продаже автомобилей "Жигули", на основе данных за последние 10 лет? |
| Отдел развития бизнеса | Васильева | 5 лет | Как повлияло на объемы продаж открытие двух новых отделений в Южном регионе и на какой процент могут увеличиться продажи в Северном регионе, если в этом году там будет открыто 3 новых офиса? |

**Таблица 4.4.Список потенциальных вопросов менеджеров фирмы**

Рассмотрим в качестве примера вопрос сотрудника коммерческого отдела ("Какие два варианта скидок наиболее эффективны в Западном регионе в летний период при продаже автомобилей "Жигули", на основе данных за последние 10 лет?"). Как было сказано выше, на этом этапе мы не собираемся программировать этот вопрос, тем более, что инструментальные средства конечного пользователя, входящие в состав Oracle Express, позволят легко сформулировать его в интерактивном режиме, без написания строк кода. Сейчас нам важнее понять, какие данные должны быть в МБД, оценить временные интервалы, которые должны отражаться, понять трудоемкость и реальность подготовки и загрузки этих данных. После того как первичный анализ вопросов выполнен, и получено представление о том, какие данные потенциально могут выступать в качестве Показателей и Измерений, можно переходить к проектированию ее структуры - определению конкретных Измерений, их взаимосвязей и уровней агрегации хранимых данных.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование информации** | **Временной интервал** | **Количество строк** | **Тип** | **Источник** |
| Месяц | 10 лет | 12 \* 10 | Измерение | Оперативная система "Продажи", архив |
| Регион | 10 лет | 5 | Измерение | Оперативная система "Продажи", архив |
| Модель автомобиля | 10 лет | 200 | Измерение | Оперативная система "Продажи", архив |
| Типы скидок | 10 лет | 4 | Измерение | Оперативная система "Продажи", архив |
| Объем продаж в USD | 10 лет | 200 \* 12 \* 10 \* 5 \* 4 | Показатель | Оперативная система "Продажи", архив |

**Таблица 4.5.Данные, необходимые для ответа на вопрос аналитика коммерческого отдела.**

Если спросить пользователя, какой уровень детализации ему желателен, он не задумываясь ответит - максимально возможный. Однако стоит оценить, сколько такое решение может стоить, и попытаться определить возможный экономический эффект от наличия данных на каждом новом уровне детализации.

Например, выбрав в качестве уровня агрегации Год, вы получите возможность проанализировать общие тенденции автомобильного рынка и спрогнозировать динамику его развития. Выбрав же в качестве уровня агрегации Месяц или Неделю, вы, кроме того, сможете спрогнозировать спрос на конкретные модели в конкретные моменты времени. И хотя автомобили - товар не сезонный, скорее всего, весной и летом их покупают больше, чем осенью и зимой. Это позволит отследить возможные сезонные колебания, рациональнее формировать свой склад и более эффективно проводить политику формирования сезонных скидок и распродаж. А если в систему введена информация о затратах на маркетинг, появится возможность проследить эффект от каждого конкретного маркетингового мероприятия.

Выбор в качестве уровня агрегации Номер Контракта/Счета позволит перейти на качественно новый уровень анализа. На этом уровне можно будет учитывать взаимосвязи между конкретным Автомобилем, Менеджером и Покупателем. А поскольку при покупке автомобиля заполняется множество документов, то доступна достаточно детальная информация о каждом конкретном Покупателе (Возраст, Пол, Место жительства, Вид оплаты и т.д.). Теперь вы сможете проанализировать не только рынок, но и заглянуть внутрь своей фирмы и всесторонне проанализировать эффективность работы каждого Менеджера и Подразделения. Но наиболее ценное, что вы получаете, - это информация о Регионах и Покупателях. Например, вы не только сможете оценить, какие Модели автомобилей пользуются наибольшим спросом в конкретном регионе сегодня, но на основе анализа истории и структуры автомобильного рынка в более развитых, с точки зрения автомобилизации, регионах попытаться оценить динамику спроса и перспективы различных Моделей в остальных регионах.

Однако переход на каждый следующий уровень детализации и добавление новых источников данных могут привести к увеличению, иногда более чем на порядок, размера целевой МБД и соответствующему удорожанию и усложнению аппаратного решения.

Рассмотрим в качестве примера Показатель Объем продаж. Анализ предметной области показывает, что он однозначно определяется комбинацией четырех Измерений:

· {Год | Полугодие | Квартал | Месяц | Неделя | День | Счет}

· {Страна | Регион | Филиал | Менеджер}

· {Фирма-Производитель | Завод-Производитель | Модель Автомобиля}

· {Тип скидки}

Выбрав уровень детализации:

· День (365 \* 10 = 3650 различных значений),

· Менеджер (300 различных значений),

· Модель Автомобиля (100 различных значений),

· Тип Скидки (4 различных значения),

получим куб, состоящий из 438000000 ячеек. Но в основе используемого в МСУБД способа хранения данных лежит предположение о том, что внутри, в данном случае четырехмерного гиперкуба, нет пустот. Данные в МСУБД представлены в виде разреженных матриц с заранее фиксированной размерностью. При этом значения Показателей хранятся в виде множества логически упорядоченных блоков (массивов), имеющих фиксированную длину, причем именно блок является минимальной индексируемой единицей.

Таким образом, в нашей БД будет сразу же зарезервировано место для всех 438 млн. значений Показателя Объем Продаж. Причем цифры "300 менеджеров" и "100 моделей автомобилей" вовсе не означают того, что сегодняшняя номенклатура фирмы - 100 различных моделей, которые продают 300 человек. Цифра 300 говорит о том, что в фирме за 10 лет ее существования работало 300 различных менеджеров. Сегодня же их может быть, например, всего 30.

Попробуем оценить, какой процент ячеек в нашем случае будет содержать реальные значения. Предположим, что в среднем в фирме постоянно работает около 30 менеджеров, менеджер продает в день 10 различных моделей и при продаже каждого автомобиля может быть использован только один вариант скидки. Тогда 3650 \* 30 \* 10 \* 1 = 1095000. То есть только 0,25% ячеек куба будет содержать реальные значения данных. И хотя в МСУБД обычно предполагается, что блоки, полностью заполненные неопределенными значениями, не хранятся, как правило, это не обеспечивает полного решения проблемы.

Однако предположим, что мы решили остановиться на уровне детализации (агрегации) в один день, и теперь можем приступить к непосредственному описанию структур данных.

|  |  |  |
| --- | --- | --- |
| **Имя** | **Дескриптор** | **Тип** |
| DAYMANAGERMODEL\_CARFIRMA\_CARDEPARTMENTREGIONMONTHYEARTIP\_DISCOUNTTOTAL\_COSTINT\_COSTQUANTITYPROFITWORK\_TIME | ДеньМенеджерМодель АвтомобиляФирма-ПроизводительПодразделение (Филиал)РегионМесяцГодТип СкидкиОбъем продаж в USDСебестоимость проданных автомобилейКоличество проданных единицДоходКоличество рабочих часов менеджера | ИзмерениеИзмерениеИзмерениеИзмерениеИзмерениеИзмерениеИзмерениеИзмерениеИзмерениеПоказательПоказательПоказательПоказательПоказатель |

**Таблица 5.6.Перечень Измерений и Показателей, используемых в описываемом примере.**

В Oracle Express Server предоставляются два альтернативных варианта описания структур данных.

· В интерактивном режиме средствами Oracle Express EDDiE (интерактивных средств проектирования, загрузки и администрирования).

· Средствами языка описания данных Data Definition Language - DDL.

Здесь для лаконичности описания в текстовом виде мы будем говорить исключительно о средствах DDL, хотя на практике эта работа обычно выполняется средствами Oracle Express EDDiE (что существенно проще, нагляднее и не требует запоминания языковых конструкций).

Define MANAGER Dimension Text

Define MODEL\_CAR Dimension Text

Define DAY Dimemsion Day

Define MONTH Dimension Month

Define YEAR Dimension Year

Define FIRMA\_CAR Dimension Text

Define TIP\_DISCOUNT Dimension Text

Объявление показателей:

Define TOTAL\_COST Variable Decimal <DAY MANAGER MODEL\_CAR TIP\_DISCOUNT>

В этой строке мы определили Показатель (Переменную) - TOTAL\_VALUE. Указали, что данный Показатель определяется четырьмя измерениями. И более того, порядком, в котором перечислены Измерения, задали порядок, в котором будут отсортированы значения TOTAL\_VALUE в БД (первым изменяется значение Измерения - День, Менеджер, Модель Автомобиля, Тип Скидки). Заметим, что при этом мы сгруппировали данные таким образом, что неопределенные значения TOTAL\_COST, появившиеся из-за того, что Менеджер в данный момент времени не работал в фирме, будут идти подряд. И тем самым существенно снизили непроизводительные затраты на хранение неопределенных значений в БД.

Определим остальные показатели:

Define INT\_COST Variable Decimal <DAY MANAGER MODEL\_CAR>

Define QUANTITY Variable Decimal <DAY MANAGER MODEL\_CAR TIP\_DISCOUNT>

Define WORK\_TIME Variable Decimal <DAY MANAGER>

Define PROFIT Formula TOTAL\_COST - INT\_COST Decimal

Показатель PROFIT имеет тип Формула и вычисляется как разница между TOTAL\_COST и INT\_COST. При объявлении Показателей мы использовали не все ранее объявленные Измерения. Это вовсе не означает, что Измерение FIRMA\_CAR не будет использоваться, и пользователь не сможет получать итоговые результаты не на уровне отдельных моделей, а на более высоком уровне агрегации - фирмы-производителя. Конечно, можно объявить Измерение Фирма-Производитель, в качестве пятого уровня иерархии при определении Показателей, но это, соответственно, увеличило бы число ячеек в гиперкубе, а следовательно, и объем БД. Более простой и эффективный способ определить эту взаимосвязь через соответствующее Отношение между Измерениями Фирма-Производитель и Модель Автомобиля.

Define FIRMA.CAR Relation FIRMA\_CAR <MODEL\_CAR>

Таким образом, мы объявили, что каждой модели автомобиля (MODEL\_CAR) однозначно соответствует выпускающая ее фирма (FIRMA\_CAR). Но пока мы только сказали, что такое отношение существует, и не более того. Для того чтобы окончательно определить отношение, нам еще потребуется ввести соответствующие пары значений, соотносимых этим Отношением Измерений, например:

АВТОВАЗ ВАЗ 2106

АВТОВАЗ ВАЗ 2108

АВТОВАЗ ВАЗ 2121

АЗЛК М2141

И хотя при объявлении Показателя Объем Продаж мы явно не указываем его взаимосвязь с Фирмой-Производителем, это не означает, что пользователь не сможет получать итоговые результаты не на уровне отдельных моделей, а на более высоком уровне агрегации - по каждой фирме. В дальнейшем, при любых операциях манипулирования данными (выборках, вычислениях и т.д.) пользователь имеет возможность ссылаться как на Модель Автомобиля, так и непосредственно Фирму-Производителя. И для того, чтобы получить Срез по всем моделям, производимым на конкретной фирме, будет достаточно просто указать имя фирмы (например "АВТОВАЗ"), а не перечислять все наименования моделей (ВАЗ2106, ВАЗ2108, ВАЗ2108), выпускаемых на ней.

Oracle Express обеспечивает возможность реализации чрезвычайно широкого спектра аналитических задач, начиная от простейшего статического DSS и заканчивая достаточно сложной статистической обработкой данных. Это обеспечивается за счет:

· развитых языковых средств манипулирования данными;

· встроенных функций агрегирования данных;

· встроенные функции математической, финансовой, статистической, логической обработок данных;

· встроенных средств для работы с датами и временными рядами.

Язык манипулирования данными Oracle Express - развитый процедурный язык, обеспечивающий возможность написания различных пользовательских программ и моделей. Причем в языке поддерживаются операции как над обычными скалярными переменными, так и непосредственно над многомерными структурами.

Кроме функций агрегации в Oracle Express включены обширные библиотеки встроенных функций для математической, финансовой, статистической обработки данных, а также обеспечивается возможность определения и подключения собственных, определяемых пользователем функций и написания пользовательских программ и моделей.

|  |
| --- |
| **Функции, основанные на временных рядах** |
| CUMSUMLAGLAGABSPCTLAGDIFLAGPSTLEADMOVINGAVERAGEMOVINGMAXMOVINGMINMOVINGTOTAL | Вычисляет нарастающий итогВычисляет значения за предшествующий временной периодВычисляет процентную разницу между текущим и предыдущим значениеми индикацию направления измененияВычисляет разность между текущим и предшествующим значениемВычисляет процентную разницу между текущим и предыдущим значениемВычисляет значения за последующий временной периодВычисляет скользящее среднееВозвращает серию максимальных значенийВозвращает серию минимальных значенийВозвращает серию итогов |
| **Функции агрегации** |
| AVERAGEAVVCOUNTEVERYLARGESTNONESMALLESTSTDEVROLLUPTOTAL | Возвращает среднее значениеПроверяет, есть ли значения, соответствующие заданному критериюПодсчитывает количество значений, соответствующих заданному критериюПроверяет все значения на соответствие заданному критериюВычисляет наибольшее значениеПроверяет на отсутствие значений, соответствующих заданному критериюВычисляет наименьшее значениеВычисляет стандартное отклонениеВычисляет сумму значений на основе Иерархических ОтношенийВычисляет сумму значений |
| **Финансовые функции** |
| DEPRDECL, DEPRSL, DEPSOYDFINTSCHED, FPMTSCHEDIRR | Вычисляют амортизационные отчисленияВычисляют расписание выплаты процентов по займамВычисляет внутреннюю процентную ставку |
| **Прочие функции** |
| REGRESSFORECAST TRENDFORECAST EXPONENTIALFORECAST WINTERS | Вычисляет линейную регрессиюВычисляет линейный прогнозВычисляет экспоненциальный прогнозВычисляет прогноз на основе алгоритма Holt-Winters |

**Таблица 4.7.Список встроенных функций ORACLE Express.**

Как уже было сказано выше, основное назначение МСУБД - работа с достаточно стабильными во времени данными, и данные в таких системах достаточно редко вводятся в интерактивном режиме. Обычно загрузка выполняется из внешних источников: оперативных БД, электронных таблиц или из заранее подготовленных плоских файлов. В Oracle Express Server загрузка данных может производиться практически из внешнего источника данных.

Следует заметить, что в Oracle Express данные могут храниться как на постоянной основе, так и загружаться динамически, в тот момент, когда к ним обратится пользователя. Таким образом, имеется возможность постоянно хранить в МБД только ту информацию, которая наиболее часто запрашивается пользователями. Для всех остальных данных хранятся только описания их структуры и программы их выгрузки из центральной (обычно реляционной) БД. И хотя при первичном обращении к таким виртуальным данным, время отклика может оказаться достаточно продолжительным, такое решение обеспечивает высокую гибкость и требует более дешевых аппаратных средств. А если впоследствии оказывается, что интенсивность обращения к данным, имеющим статус временных, высока, их статус может быть легко изменен.

Семейство программных продуктов - Oracle Express включает в себя следующие основные компоненты.

· Система управления базами данных, основанная на многомерном подходе (Oracle Express Server).

· Интерактивные средства администрирования (Oracle Express EDDiE).

· Средства для построения и реализации интерактивных On-Line Analitical Processing (OLAP) приложений (Oracle Express Analyzer, Oracle Express Objects).

Oracle Express Server- масштабируемый (может работать на большинстве аппаратных платформ, начиная с персональных компьютеров и заканчивая Unix-серверами и большими машинами) сервер многомерных данных, обеспечивающий возможность работы и хранения больших объемов (до нескольких десятков гигабайт) данных. Express Server легко интегрируется в существующую вычислительную среду Oracle7 и может быть установлен на той же машине, что Oracle7 RDBMS или может связан с ней через локальную или удаленную сеть. Oracle Express Server включает в свой состав все средства, необходимые для реализации и эксплуатации самых различных аналитических приложений. Разработка приложений осуществляется на специализированном 4GL-языке, обеспечивающем матрично-ориентированные средства манипулирования данными и включающем широкий спектр математических, финансовых, статистических, логических и строчных функций.

Oracle Express EDDiE- интерактивные средства описания структуры БД, проектирования процедур загрузки, загрузки и администрирования данных.

Oracle Express Analyzer- инструментальные средства, ориентированные на широкий круг пользователей (разработчиков, операторов, аналитиков), обеспечивающие возможность динамического формирования запросов, отчетов и бизнес-диаграмм. Oracle Express Analyzer может быть использован как в качестве непосредственного инструментария, предоставляемого конечному пользователю (руководителю, аналитику), так и в качестве среды разработчика, для реализации регламентированных приложений и их прототипов. Разработанные на его основе приложения представляются в виде многостраничных книг (Briefing Books), страницами которых являются интерактивные таблицы (формы) и бизнес-диаграммы.

Oracle Express Objects- инструментальные средства, ориентированные на профессиональных разработчиков OLAP-приложений. С помощью Oracle Express Objects разработчики имеют возможность создавать все элементы аналитического приложения: таблицы, бизнес-диаграммы, кнопки, меню и т.д. Разработанные приложения полностью совместимы и могут выполняться с помощью Oracle Express Analyzer, так как оба средства базируются на одних и тех же объектах и основываются на одном и том же программном ядре.

Было бы не совсем правильно противопоставлять или говорить о какой-либо серьезной взаимной конкуренции реляционного и многомерного подходов. Правильнее сказать, что эти два подхода взаимно дополняют друг друга. Реляционный подход никогда не предназначался для решения на его основе задач, требующих синтеза, анализа и консолидации данных. И изначально предполагалось, что такого рода функции должны реализовываться с помощью внешних по отношению к РСУБД, инструментальных средств. Но именно на решение таких задач и ориентированы МСУБД. Область, где они наиболее эффективны, это хранение и обработка высоко агрегированных и стабильных во времени данных. И их применение оправдано только при выполнении двух требований.

· Уровень агрегации данных в БД достаточно высок, и, соответственно, объем БД не очень велик (не более пары гигабайт).

· В качестве граней гиперкуба выбраны достаточно стабильные во времени Измерения (с точки зрения неизменности их взаимосвязей), и, соответственно, число несуществующих значений в ячейках гипрекуба относительно невелико.

Поэтому уже сегодня МСУБД все чаще используются не только как самостоятельный программный продукт, но и как аналитические средства переднего плана, к системам Хранилищ Данных или традиционным оперативным системам, реализуемым средствами РСУБД.



**Рис. 4.4.Многоуровневая архитектура.**

Причем такое решение позволяет наиболее полно реализовать и использовать достоинства каждого из подходов: компактное хранение детализированных данных и поддержка очень больших БД, обеспечиваемые РСУБД и простота настройки и хорошие времена отклика, при работе с агрегированными данными, обеспечиваемые МСУБД.

**5. Пример создания многомерной базы данных с помощьюMicrosoft Analysis Services**

Рассмотрим создание многомерного OLAP-куба на основании хранилища данных Northwind\_Mart, которое мы создали и заполнили в третьей главе. 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 и запустить утилиту Analysis Manager, с помощью которой обычно и создаются многомерные базы данных.

Прежде чем создавать OLAP-кубы, необходимо описать источники исходных данных для них. В нашем примере таким источником является созданное ранее хранилище Northwind\_Mart.



**Рис. 5.1. Создание источника данных**

Как мы уже знаем, 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.

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:



**Рис. 5.2. 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. 5.3. 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. 5.4. 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. 5.5. 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.

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. 5.6. Создание регулярного измерения в Dimension Editor**

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. 5.7. 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.

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. 5.8. 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. 5.9. 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.PC, и укажем в качестве свойства члена измерения поле Hire Date. В результате мы получим иерархию:



**Рис. 5.10. 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.

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.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. 5.11. 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. 5.12. Выбор мер 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. 5.13. Редактор кубов**

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 Сalculated 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 ведем Discounted Total.



**Рис. 5.14. 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.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. 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. 5.15. 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. 5.16. 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. 5.17. 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. О других способах чтения многомерных кубов мы расскажем в одной из следующих статей данного цикла.

**6.OLAPклиенты**

Проиллюстрируем возможности OLAP-клиентов на примере Microsoft Excel. Для начала 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-приложениях.

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. 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.

В качестве примера создадим сводную таблицу, содержащую данные OLAP-куба, создание которого мы рассматривали в предыдущей главе. 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. 6.1. 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. 6.2. 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. 6.3. Выбор OLAP-сервера**

И наконец, выберем имя многомерной базы данных, в которой хранится OLAP-куб.



**Рис. 6.4. Выбор многомерной базы данных**

Определив источник данных, выберем куб, который мы будем отображать в сводной таблице.



**Рис. 6.5. Выбор куба для отображения в сводной таблице**

После этого можно нажать кнопку OK. В результате мы получим пустую сводную таблицу, вид которой в Excel 2000:



**Рис. 6.6. 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. 6.7. 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.



**Рис. 6.8. 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.



**Рис. 8.9. Готовая сводная таблица**

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

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:



**Рис. 6.10. 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. 6.11. 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. 6.12. 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. 6.13. 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.

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. 6.14. 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. 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. 6.15. Диалоговая панель Offline OLAP Settings**

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



**Рис. 6.16. 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. 6.17. 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.

**7. ПрименениеOLAPтехнологий**

Эффективное управление крупным и средним бизнесом сегодня не мыслимо без применения передовых информационных технологий - систем поддержки принятия решений (далее, СППР). Для сложных систем характерно то, что управлять ими приходится, как правило, в условиях не полной информации, незнания закономерностей функционирования и постоянного изменения внешних факторов. Поэтому процесс управления имеет итерационный характер. После принятия решения и применения управляющего воздействия необходимо вновь оценить состояние, в котором находится система, и решить вопрос, о том правильно ли мы движемся по намеченному пути. Если отклонения нас не удовлетворяют, то необходимо переопределить процесс управления. Современные информационные технологии при поиске ответов на поставленные вопросы позволяют аналитику формулировать и решать следующие классы задач:

· Аналитические - вычисление заданных показателей и статистических характеристик бизнес деятельности на основе ретроспективной информации из баз данных.

· Визуализация данных - наглядное графическое и табличное представление имеющейся информации.

· Добыча знаний (data mining) - определение взаимосвязей и взаимозависимостей бизнес процессов на основе существующей информации. К данному классу можно отнести задачи: проверки статистических гипотез, кластеризации, нахождения ассоциаций и временных шаблонов. Например, путем анализа экономических и финансовых показателей деятельности компаний, которые затем обанкротились, банк может выявить некоторые стереотипы, которые можно будет учесть при оценке степени риска кредитования.

· Имитационные - проведение на ЭВМ экспериментов с математическими моделями, описывающими поведение сложных систем в течение заданного или формируемого интервала времени. Задачи этого класса применяются для анализа возможных последствий принятия того или иного управленческого решения (анализ "Что, если?...").

· Синтез управления - используется для определения допустимых управляющих воздействий, обеспечивающих достижение заданной цели. Задачи этого типа применяются для оценки достижимости намеченных целей, определения множества возможных управляющих воздействий, приводящих к заданной цели.

· Оптимизационные - основаны на интеграции имитационных, управленческих, оптимизационных и статистических методов моделирования и прогнозирования. Задачи данного класса позволяют выбрать на множестве возможных управлений те из них, которые обеспечивают наиболее эффективное (с точки зрения определенного критерия) продвижение к поставленной цели.

Вряд ли следует надеяться, что когда-либо будет создано комплексное программное обеспечение (ПО) СППР, которое будет реализовывать все, или хотя бы относительно большую часть алгоритмов, применяемых при решении перечисленных классов задач. Не стоит также полагать, что в ближайшие годы могут появиться на рынке программного обеспечения универсальные имитационные модели функционирования коммерческого предприятия, которые можно бы было адаптировать и использовать в СППР конкретной фирмы. Однако весьма вероятно, что ряд фирм для того, чтобы добиться преимущества в конкурентной борьбе, сделает ставку на разработку собственных имитационных моделей. Исходя из личного опыта участия в создании моделей сложных систем, могу утверждать, что даже при наличие квалифицированной команды из бизнес аналитиков, математиков и программистов срок реализации подобного проекта составит не менее чем полтора-два года. До тех пор о реальном использовании информационных технологий при решении задач пятого и шестого классов говорить, на мой взгляд, преждевременно, поскольку существующие в этих областях алгоритмы подразумевают наличие адекватной модели управляемой системы.

Бизнес представляет собой сложный объект, который строится из множества различных по свойствам подсистем, между которыми действует большое число разнородных связей. Предпринимательская деятельность состоит из целого ряда бизнес - процессов, которые существенно зависят от множества внешних факторов: законодательных, экономических, социальных, политических. В кибернетике такие объекты получили название сложные системы, а методы их изучения - системный анализ. Хотя эта наука развивается с начала 40-х годов, с тех пор, когда в армиях США и Великобритании стали привлекать ученых к разработке рекомендаций по проведению боевых операций в ходе Второй мировой войны, существенные практические результаты получены лишь в исследовании операций - применении количественных математических методов для обоснования решений. Но математика начинает работать лишь тогда, когда исследователь имеет модель системы. Математиков, как правило, мало интересует откуда берется формальная модель. Они справедливо считают, что построение модели относится к компетенции специалистов в конкретной прикладной области.

Откуда же берутся модели и почему их практически нет в системах управления бизнесом? Общая с точки зрения теории познания триада имеет вид: Гипотеза - Модель - Решение. Появлению любой модели предшествует гипотеза о взаимосвязях явлений и объектов реального мира. Гипотеза является открытием, принципиально новым знанием. Открытие не может быть вычислено, или получено путем логического вывода из уже существующих знаний, поскольку в этом случае оно будет лишь тавтологией, повторением уже пройденного. С точки зрения современной психологии открытие совершается при помощи интуиции, которая уходит корнями в область человеческого подсознания и опирается на огромный личный опыт. Георг Ом, прежде чем открыл свой закон, несколько лет проводил эксперименты с проводниками разного диаметра, разной длины, изготовленными из разных металлов, анализировал и накапливал результаты опытов. Открытию, как правило, предшествует не одна, а множество гипотез. По каждой гипотезе строится модель - формальное математическое описание - и находится решение. Полученное решение проверяется затем в эксперименте и отвергается, если не находит подтверждения.

Вторая проблема заключается в том, что в предпринимательской деятельности да еще в условиях свободного рынка отсутствует возможность проведения целенаправленных экспериментов, которые предшествуют открытию гипотезы и позволяют проверять ее на практике. В настоящее время в бизнесе реально следует рассчитывать лишь на использование комплексного ПО, реализующего алгоритмы решения задач первого, второго и частично третьего из перечисленных классов. Сегодня мы являемся свидетелями стремительного прогресса в создании подобного ПО под общим названием OLAP (On-line Analytical Processing). Предназначение новых технологий хранилищ данных и OLAP - заполнить объективно существующие разрывы познавательной деятельности. Более100 крупнейших производителей программ включились в конкуренцию на данном секторе рынка.

Хранилища данных по своей сути больше идея, чем технология. Идея заключается в том, чтобы собрать в едином , по крайней мере с точки зрения пользователя, месте - супербазе всю информацию, которая может понадобиться управляющему при принятии решения. Источниками данных для информационного хранилища служат в первую очередь данные из разрозненных транзактных информационных систем, основанных на различных реляционных СУБД, которые обслуживают повседневную бизнес-деятельность. Следует особо подчеркнуть, что хранилище данных не предназначается для замены существующих систем, а является как бы надстройкой над ними. В хранилище данных могут быть включены сведения о клиентах, о штатном персонале, о конкурентах, о демографической ситуации, о показателях экономики и проч. Источниками необходимой информации могут быть газеты, радио, телевидение, Интернет и любые другие. При этом предполагается, что данные предварительно должны быть приведены к единым стандартам, очищены от противоречий, структурированы и обобщены с требуемым уровнем детализации. Прогнозируемый объем хранилищ данных оценивается в десятки терабайт (1терабайт примерно равен 1012байт, для наглядности можете представить себе стопку книг, энциклопедического формата высотой порядка километра). Сама идея хранилищ данных, не является новой. Отличительной особенностью хранилищ являются лишь прогнозируемые объемы хранимой информации, которые позволяют надеяться получить качественно новое свойство БД - восполнить описанные в предыдущем разделе разрывы познавательной деятельности аналитика, которые состоят в ограниченности личного опыта и невозможности проводить целенаправленные эксперименты. Не без основания предполагается, что в процессе анализа показателей собственной коммерческой деятельности и деятельности конкурентов в их взаимосвязи с внутренними и внешними факторами аналитик выработает интуицию, необходимую для выдвижения гипотез, а затем сумеет проверить справедливость найденных закономерностей, но не в ходе проведения целенаправленного эксперимента, а опять же при помощи информации из хранилища, которая уже содержит результаты требуемых опытов, поставленных самой жизнью.

Программные средства OLAP это инструмент оперативного анализа данных, содержащихся в хранилище. Главной особенностью, на мой взгляд, является то, что эти средства ориентированы на использование не специалистом в области информационных технологий, не экспертом-статистиком, а профессионалом в прикладной области - менеджером кредитного отдела, менеджером бюджетного отдела, наконец, директором. Они предназначены для общения аналитика с проблемой, а не с компьютером.

Имея гибкие механизмы манипулирования данными и визуального отображения, исследователь, как правило, сначала рассматривает с разных сторон данные, которые могут быть (а могут и не быть) связаны с решаемой проблемой, не имея при этом в голове никаких идей, просто пытаясь заметить какие-либо особенности. Сопоставляет различные показатели бизнеса между собой, стараясь выявить скрытые взаимосвязи. Заинтересовавшись какой либо позицией, он может рассмотреть данные боле пристально, детализировав их, например, разложив на составляющие по времени, по регионам или по клиентам, или наоборот еще более обобщить представление информации, чтобы убрать отвлекающие подробности. У него , например, может зародиться гипотеза о том, что разброс роста активов в различных филиалах банка, зависит от соотношения в них специалистов с техническим и гуманитарным образованием. Тогда аналитик может запросить из хранилища (а не у отдела информатизации!) и отобразить на одном графике интересующее его соотношение для тех филиалов, у которых за текущий квартал рост активов снизился, по сравнению с прошлым годом, более чем на 10% и для тех, у которых повысился боле чем на 25%. Для этого исследователь должен иметь возможность использовать не изощренный SQL-запрос, а простой выбор из предлагаемого меню. Если полученные результаты ощутимо распадутся на две соответствующие группы, то это должно стать стимулом для дальнейшей проверки выдвинутой гипотезы. А может быть полученные результаты извлекут из подсознания какие-то новые ассоциации и поиск начнет продвигаться в другом направлении. Возьму на себя смелость утверждать, что сегодня мы являемся свидетелями достаточно редкого явления - широкого внедрения элементов искусственного интеллекта в практическую деятельность, да еще в такой заповедной области как бизнес. В отличие от традиционных систем искусственного интеллекта технология OLAP, не пытается моделировать естественный интеллект, а усиливает его возможности мощностью современных вычислительных серверов и хранилищ данных.

Универсальность законов психологии, положенных в основу OLAP, позволяет разработчику приложений мало заботиться о характере возможных запросов данных конечным пользователем. Законы человеческого мышления мало изучены. Вместе с тем, вряд ли кто-нибудь сомневается в том, что общие законы мышления существуют и действуют. К признакам OLAP, основанным на законах психологии, следует отнести:

· Разделение данных на показатели (переменные) и измерения, определяющие соответственно состояние и пространство бизнеса.

· Логическое представление значений показателей в виде многомерных кубов, упорядоченных по равноправным измерениям.

· Неограниченное число и количество уровней иерархических связей между значениями измерениями.

· Гибкое манипулирование данными. Возможность построение подмножества значений показателя по любому дискриминирующему правилу, определенному на множестве значений его измерений. Возможность построения подмножества значений измерения по любому дискриминирующему правилу, определенному на множестве значений любой из переменных, связанной с ней. Логические операции над полученными множествами.

· Неограниченные возможности агрегирования заданного подмножества значений показателя. При этом должна предоставляться возможность вычислять не только сумму значений, но и любой другой определенный пользователем функционал, например, минимум, максимум, среднее, медиану и проч.

· Возможность обработки запросов в "реальном времени" - в темпе процесса аналитического осмысления данных пользователем.

· Развитые средства табличного и, главное, графического представления данных конечному пользователю.

Важность гибкого графического представления, хотелось бы подчеркнуть особо. Такие выдающиеся ученые, как Жак Адамар и Жуль Пуанкаре, которые пытались при помощи самоанализа изучить творческий процесс математического открытия, сошлись во мнении, что мыслят при решении сложной задачи не словами, не математическими знаками а некоторыми геометрическими образами и, когда воображаемые образы соединяются в решение, то остается только формализовать его в символьном виде, чтобы донести это решение до остальных. Современная психология так же утверждает, что творческое мышление - образное. Она называет его правополушарным. Не углубляясь в психологические аспекты проблемы, приведем известный факт, что человеческий мозг способен воспринимать и анализировать информацию, которая представлена в виде геометрических образов, в объеме на несколько порядков большем, чем информацию, представленную в алфавитно-цифровом виде.

Наглядные геометрические образы, связанные с решаемой проблемой, колоссально стимулируют творческое мышление и приводят к открытиям даже в такой формальной области, как теория чисел. Один из ведущих российский специалистов в области искусственного интеллекта, профессор Д. Поспелов назвал системы подобные OLAP новым окном в мир познания. Важность развитого графического представления информации и его влияние на интуицию исследователя, подчеркивают и зарубежные специалисты.

Технология OLAP призвана повысить эффективность информационно-аналитической и управленческой деятельности руководящего персонала. Используя эти средства, можно быстрее и более обоснованно принимать оперативные и стратегические решения. Открытые при помощи OLAP закономерности реализуются затем в экономические модели, позволяющие заглянуть в будущее, которое, по словам Нейла Рейдена президента Archer Decision Sciences Inc., "принадлежит тому, кто сможет его предвидеть и первым к нему приблизится".

**Заключение**

Концепция хранилищ данных не хранит в себе что-то принципиально новое, о чём не говорилось ранее и чему нельзя найти аналогий в прошлом. Подобно многим новациям в технологиях, этот термин «хранилища данных» используется для того, чтобы описать основу, которая имеет потенциал развиться со временем во что-то более сложное и значительное. Уже сегодня можно говорить о том, что появление этой концепции послужило серьёзным стимулом для развития внутренней архитектуры современных СУБД, их программного окружения, инструментальных средств конечного пользователя, различных межкорпоративных стандартов. Несмотря на то, что стоимость аналитических систем даже сегодня остается достаточно высокой, а методологии и технологии реализации таких систем находятся ещё в стадии их становления, уже сегодня, экономический эффект обеспечиваемый ими существенно превышает эффект от традиционных оперативных систем. Эффект от правильной организации, стратегического и оперативного планирования развития бизнеса трудно заранее оценить в цифрах, но очевидно, что он в десятки и даже сотни раз может превзойти затраты на реализацию таких систем. Однако не следует и заблуждаться. Эффект обеспечивает не сама система, а люди с ней работающие. Современные аналитические системы не являются системами искусственного интеллекта и они не могут ни помочь, ни помешать в принятии решения. Их цель своевременно обеспечить менеджера всей информацией необходимой для принятия решения. А какая информация будет запрошена и какое решение будет принято на её основе, зависит только от конкретного человека

**Литература**

1. “Принципы проектирования и использования многомерных баз данных (на примере Oracle Express Server) ” А.А.Сахаров, СУБД, №3, 1996

2. Кречетов Н., Иванов П. Продукты для интеллектуального анализа данных, ComputerWeek-Москва. - 1997. - № 14-15.

3. Сахаров А. А. Концепция построения и реализации информационных систем, ориентированных на анализ данных, СУБД. - 1996. - № 4.

4. Сахаров А. А. Принципы проектирования и использования многомерных баз данных (на примере Oracle Express Server). СУБД. - 1996. - № 3.

* [**Условия использования**](https://domashke.net/articles/pravila)

* [**Родителям**](https://domashke.net/articles/roditelyam)

* [**Реклама**](https://domashke.net/articles/reklama)

* [**Правообладателям**](https://domashke.net/articles/pravo)