ХРАНИЛИЩА ДАННЫХ
учебное пособие

Томск – 2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)
Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав. Кафедрой АОИ
Д.т.н., профессор
_____
Ю. П. Ехлаков
«___»
_____ 2015 г.

УЧЕБНОЕ ПОСОБИЕ

по дисциплине «Хранилища данных»
для студентов направления
230102 – Бизнес информатика

Разработчик
Доцент каф. АОИ
К.т.н., с.н.с.

О.И. Жуковский

Томск 2015

Аннотация

«Хранилища данных» (ХД) – учебная дисциплина, в которой изучаются современные технологии, методы и средства проектирования и построения автоматизированных информационных систем, ориентированных на анализ данных. Специалисты в области ХД должны владеть знаниями и умениями, позволяющими принимать обоснованные решения на всех стадиях и этапах проектирования и построения автоматизированных информационных систем, ориентированных на анализ данных.

Пособие подготовлено в соответствии с требованиями Государственного образовательного стандарта высшего профессионального образования.

Учебное пособие предназначено для студентов факультета дистанционного обучения ТУСУР.

Оглавление

Введение

«Хранилища данных» (ХД) – учебная дисциплина, в которой изучаются современные технологии, методы и средства проектирования и построения автоматизированных информационных систем, ориентированных на анализ данных. Специалисты в области ХД должны владеть знаниями и умениями, позволяющими принимать обоснованные решения на всех стадиях и этапах проектирования и построения автоматизированных информационных систем, ориентированных на анализ данных.

В пособии представлены такие базовые темы, относящиеся к области проектирования и использования Хранилищ Данных как:

Основные принципы построения систем, ориентированных на анализ данных;

Модели данных, используемые для построения хранилищ;

Особенности построения систем на основе хранилищ данных;

Основные бизнес-функции процесса разработки и проектирования хранилища данных;

Основные принципы OLAP.

Назначение и область применения методов интеллектуального анализа данных;

1. Системы поддержки принятия решений

1.1. Задачи систем поддержки принятия решений

В настоящее время современные вычислительные системы и компьютерные сети позволяют накапливать большие массивы данных для решения задач обработки и анализа. К сожалению, сама по себе машинная форма представления данных содержит информацию, необходимую человеку, в скрытом виде, и для ее извлечения нужно использовать специальные методы анализа данных.

Большие объемы информации, с одной стороны, позволяют получить более точные расчеты и анализ, с другой — превращают поиск решений в сложную задачу. Неудивительно, что первичный анализ данных был переложен на компьютер. В результате появился целый класс программных систем, призванных облегчить работу людей, выполняющих анализ (аналитиков). Такие системы принято называть системами поддержки принятия решений — СППР (DSS, Decision Support System).

Для выполнения анализа СППР должна накапливать информацию, обладая средствами ее ввода и хранения. Можно выделить три основные задачи, решаемые в СППР: ввод данных; хранение данных; анализ данных. Таким образом, СППР — это системы, обладающие средствами ввода, хранения и анализа данных, относящихся к определенной предметной области, с целью поиска решений.

Ввод данных в СППР осуществляется либо автоматически от датчиков, характеризующих состояние среды или процесса, либо человеком-оператором. В первом случае данные накапливаются путем циклического опроса или по сигналу готовности, возникающему при появлении информации. Во втором случае СППР должны предоставлять пользователям удобные средства ввода данных, контролирующие корректность вводимых данных и выполняющие сопутствующие вычисления. Если ввод осуществляется одновременно несколькими операторами, то система должна решать проблемы параллельного доступа и модификации одних и тех же данных разными пользователями.

Постоянное накопление данных приводит к непрерывному росту их объема. В связи с этим на СППР ложится задача обеспечить надежное хранение больших

объемов данных. На СППР также могут быть возложены задачи предотвращения несанкционированного доступа, резервного хранения данных, архивирования и т. п.

Основная задача СППР — предоставить аналитикам инструмент для выполнения анализа данных. Необходимо отметить, что для эффективного использования СППР ее пользователь-аналитик должен обладать соответствующей квалификацией. Система не генерирует правильные решения, а только предоставляет аналитику данные в соответствующем виде (отчеты, таблицы, графики и т. п.) для изучения и анализа, именно поэтому такие системы обеспечивают выполнение функции поддержки принятия решений. И если с одной стороны качество принятых решений зависит от квалификации аналитика, то с другой стороны рост объемов анализируемых данных, высокая скорость обработки и анализа, а также сложность использования машинной формы представления данных стимулируют исследования и разработку интеллектуальных СППР. Для таких СППР характерно наличие функций, реализующих отдельные умственные возможности человека [1].

По степени "интеллектуальности" обработки данных при анализе выделяют три класса задач анализа:

информационно-поисковый — СППР осуществляет поиск необходимых данных. Характерной чертой такого анализа является выполнение заранее определенных запросов;

оперативно-аналитический — СППР производит группирование и обобщение данных в любом виде, необходимом аналитику. В отличие от информационно-поискового анализа в данном случае невозможно заранее предсказать необходимые аналитику запросы;

интеллектуальный — СППР осуществляет поиск функциональных и логических закономерностей в накопленных данных, построение моделей и правил, которые объясняют найденные закономерности и/или прогнозируют развитие некоторых процессов (с определенной вероятностью).

Обобщенная архитектура СППР может быть представлена следующим образом (рис. 1.1)

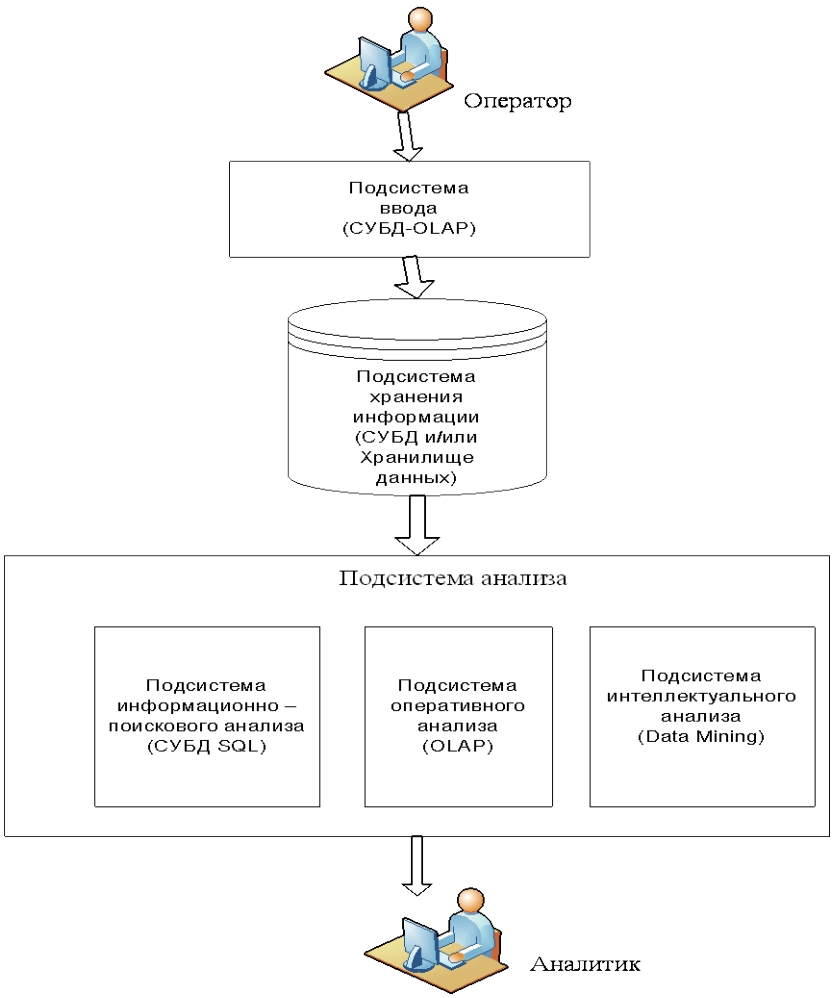

Рис 1.1 Обобщенная архитектура СППР.

Рассмотрим отдельные подсистемы более подробно.

Подсистема ввода данных. В таких подсистемах, относящихся к классу OLTP-систем (On-line transaction processing), выполняется операционная (транзакционная) обработка данных. Для реализации этих подсистем используют обычные системы управления базами данных (СУБД).

Подсистема хранения. Для реализации данной подсистемы используют современные СУБД и системы, отвечающие концепции хранилищ данных.

Подсистема анализа. Данная подсистема может быть построена на основе:

- подсистемы информационно-поискового анализа на основе реляционных СУБД и статических запросов с использованием языка структурных запросов SQL;

- подсистемы оперативного анализа. Для реализации таких подсистем применяется технология оперативной аналитической обработки данных OLAP (On-line analytical processing), использующая концепцию многомерного представления данных;
- подсистемы интеллектуального анализа. Данная подсистема реализует методы и алгоритмы Data Mining (добыча данных).

1.2 OLTP-системы

OLTP-системы оперативной обработки транзакций характеризуются большим количеством изменений хранимых данных, одновременным обращением множества пользователей к одним и тем же данным для выполнения разнообразных операций, таких как чтение, запись, удаление и модификация данных. Для нормальной работы множества пользователей применяются блокировки и транзакции. Эффективная обработка транзакций и поддержка блокировок входят в число важнейших требований к системам оперативной обработки транзакций.

К этому классу систем относятся и первые СППР — информационные системы руководства (ИСР или EIS - Executive Information Systems). Такие системы, как правило, строятся на основе реляционных СУБД, включают в себя подсистемы сбора, хранения и информационно-поискового анализа информации, а также содержат в себе предопределенное множество запросов для повседневной работы. Каждый новый запрос, непредусмотренный при проектировании такой системы, должен быть сначала формально описан, закодирован программистом и только затем выполнен. Время ожидания в этом случае может составлять часы и дни, что неприемлемо для оперативного принятия решений.

Как уже было сказано выше основной логической единицей функционирования систем операционной обработки данных является транзакция. Транзакцией называют неделимую с позиции воздействия на БД последовательность операции манипулирования данными. Транзакция может состоять из операции чтения, удаления, вставки, модификации данных. В OLTP-системах транзакция реализует

некоторое осмысленное, с точки зрения пользователя, действие, например, перевод денег со счета на счет в платежной системе банка, резервирование места в поезде системой оформления железнодорожных билетов, резервирование места в кинотеатре и т.п.

Традиционно понятие "обработка транзакций" использовалось применительно к крупномасштабным системам обработки данных - системам, осуществлявшим международные банковские операции и др. Теперь ситуация меняется. Информационные системы в различных областях человеческой деятельности становятся все более распределенными и неоднородными, в них остро стоят проблемы сохранения целостности данных и разграничения доступа. Одно из направлений решения этих проблем - использование средств обслуживания транзакций в информационных системах [1].

Чтобы использование механизмов обработки транзакций позволило обеспечить целостность данных и изолированность пользователей, транзакция должна обладать четырьмя основными свойствами: атомарности (atomicity), согласованности (consistency), изолированности (isolation), долговечности (darability). Транзакции, обладающие перечисленными свойствами, иногда называют ACID-транзакциями по первым буквам их английских названий.

Свойство атомарности означает, что транзакция должна выполняться как единая операция доступа к БД. Она должна быть выполнена полностью либо не выполнена совсем. То есть должны быть выполнены все операции манипулирования данными, которые входят в транзакцию, либо, если по каким-то причинам выполнение части операций невозможно, ни одна из операций не должна выполняться. Свойство атомарности обычно коротко выражают фразой: "все или ничего".

Свойство согласованности гарантирует взаимную целостность данных, то есть выполнение ограничений целостности БД после окончания обработки транзакции. Следует отметить, что база данных может обладать такими ограничениями целостности, которые сложно не нарушить в случае выполнения каждый раз только одного оператора ее изменения. Например, если в отношении A хранится число кортежей отношения B, то добавить новый кортеж в отношение B, не нарушив

ограничений целостности, невозможно. Поэтому такое нарушение внутри транзакции допускается, но к моменту ее завершения база данных должна быть в целостном состоянии. Несоблюдение в системах со средствами контроля целостности этого условия приводит к отмене всех операции транзакции.

В многопользовательских системах с одной БД одновременно могут работать несколько пользователей или прикладных программ. Поскольку каждая транзакция может изменять разделяемые данные, данные могут временно находиться в несогласованном состоянии. Доступ к этим данным другим транзакциям должен быть запрещен, пока изменения не будут завершены; Свойство изолированности транзакций гарантирует, что они будут выполняться отдельно друг от друга.

Свойство долговечности означает, что если транзакция выполнена успешно, то произведенные в ходе ее выполнения изменения в данных не будут потеряны ни при каких обстоятельствах.

Результатом выполнения транзакции может быть ее фиксация или откат. Фиксация транзакции - это действие, обеспечивающее запись в БД всех изменений, которые были произведены в процессе ее выполнения. До того как транзакция зафиксирована, возможна отмена всех сделанных изменений и возврат базы данных в то состояние, в котором она была до начала выполнения транзакции. Фиксация транзакции означает, что все результаты ее выполнения становятся видимыми другим транзакциям. Для фиксации транзакции необходимо успешное выполнение всех ее операторов.

Если нормальное завершение транзакции невозможно, например, нарушены ограничения целостности БД, или пользователь выдал специальную команду, происходит откат транзакции. База данных возвращается в исходное состояние, все изменения аннулируются.

Механизм корректного отката и фиксации транзакций основан на использовании журнала транзакций. Для того чтобы иметь возможность сделать откат, СУБД должна сохранять все изменения, которые транзакция внесла в БД. Однако нет необходимости каждый раз сохранять всю информацию базы данных. Реляционные операции изменяют строки отношений БД, поэтому, чтобы обеспечить возможность

отката, СУБД должна хранить те строки, которые были модифицированы. При выполнении любой операции, изменяющей базу данных, СУБД автоматически сохраняет в журнале транзакций состояние модифицируемых строк до операции и после нее. Только после этого изменения вносятся в БД. Если по окончании обработки транзакция фиксируется, то в журнале делается соответствующая отметка. Если же производится откат транзакции, то СУБД по журналу восстанавливает те строки отношений, которые были модифицированы, отменяя, таким образом, все изменения.

Для того чтобы оперировать транзакцией как единой логической единицей, СУБД должна уметь определять ее границы, то есть первую и последнюю входящую в нее операции. Стандарт языка SQL предусматривает следующий принцип выделения транзакции как некоторой законченной последовательности действий. Предполагается, что транзакция начинается с первого SQL-оператора, вводимого пользователем или содержащегося в прикладной программе. Все следующие далее операторы составляют тело транзакции. Тело транзакции завершается SQL-операторами COMMIT WORK или ROLLBACK WORK. Выполнение транзакции заканчивается также при завершении программы, которая сгенерировала транзакцию. Транзакция фиксируется, если ее тело оканчивается оператором COMMIT WORK, либо при успешном завершении программы, сформировавшей транзакцию. Откат транзакции производится при достижении оператора ROLLBACK WORK, либо в случае, когда приложение, сгенерировавшее транзакцию, завершилось с ошибкой.

Рассмотрим пример транзакции, модифицирующей телефон (атрибут Phone) сотрудника с фамилией (атрибут Name) "Петров" в отношении Отдел (Department). Транзакция завершается фиксацией по достижении оператора COMMIT WORK.

UPDATE Department SET Phone = "5388" WHERE Name = "Петров" COMMIT WORK

Некоторые диалекты языка SQL, например, диалект, принятый в СУБД Sybase, включают специальные операторы, позволяющие производить промежуточную фиксацию транзакции. В теле транзакции могут быть определены точки, в которых

сохраняется состояние базы данных. Откат в этом случае может производиться как к одной из точек промежуточной фиксации, так и к состоянию до начала выполнения транзакции. Точки промежуточной фиксации применяются в "длинных" транзакциях. Они позволяют разделить ее на несколько отдельных фрагментов.

Применение транзакций - эффективное механизм организации многопользовательского доступа к БД. Однако при реализации этого механизма СУБД приходится сталкиваться с целым рядом проблем. Во-первых, необходимо избежать потери изменений БД в ситуации, когда несколько программ читают одни и те же данные, изменяют их и пытаются записать результат на прежнее место. В БД могут быть сохранены изменения, выполненные только одной программой, результаты работы всех остальных программ будут потеряны. Во-вторых, требуется исключить возможность чтения незафиксированных изменений. Это может произойти в случае, когда одна транзакция вносит изменения в БД, они тут же считываются в другой транзакции, но затем другая транзакция прерывается оператором ROLLBACK WORK.

Чтобы избежать этих проблем, должна быть использована специальная дисциплина совместной обработки (сериализации) транзакций. В ее основе лежат следующие принципы.

1) Транзакция не может получить доступ к незафиксированным данным, то есть к данным, в которых произведены изменения, но эти изменения еще не зафиксированы.

2) Результат совместного выполнения транзакций должен быть эквивалентен результату их последовательного выполнения. То есть, если две транзакции выполняются параллельно, то полагается, что результат будет такой же, как если бы сначала выполнялась первая, а затем вторая транзакция, или сначала вторая, а потом первая.

В современных СУБД сериализация транзакций реализуется через механизм блокировок. На время выполнения транзакции СУБД блокирует часть БД (отношение, строку или группу строк), к которой транзакция обращается.

Блокировка сохраняется до момента фиксации транзакции. Если в процессе параллельной обработки другой транзакции делается попытка обратиться к блокированным данным, обработка транзакции приостанавливается и возобновляется только после завершения транзакции, заблокировавшей данные и снятия блокировки.

При выполнении транзакции современные СУБД могут блокировать всю БД, отношение, группу строк или отдельную строку. Очевидно, что чем больше блокируемый элемент данных, тем медленнее СУБД обрабатывает транзакции - велико время ожидания снятия блокировок. При работе в режиме оперативного доступа к БД, как правило, реализуется блокировка на уровне отдельных строк. В этом случае можно добиться максимальной производительности за счет того, что блокируемый объект - минимальная структурная единица БД.

Транзакции могут попасть в ситуацию взаимоблокировки. Для предотвращения таких ситуаций СУБД периодически проверяет блокировки, установленные выполняющимися транзакциями. Если обнаруживается ситуация взаимоблокировки, то одна из транзакций, вызвавших эту ситуацию, прерывается. Это разрешает тупиковые ситуации. Программа, которая сгенерировала прерванную транзакцию, получает сообщение об ошибке. Для того чтобы избежать взаимоблокировок, стараются в каждой транзакции обновление отношений делать в одной и той же последовательности.

Современные информационные системы работают с распределенными БД, поэтому в одной транзакции могут модифицироваться отношения, физически хранящиеся на удаленных вычислительных системах. Транзакция, обновляющая данные на нескольких узлах сети, называется распределенной. Если транзакция работает с БД, расположенной на одном узле, то ее называют локальной. Таким образом, логически распределенная транзакция состоит из нескольких локальных.

С точки зрения пользователя, локальные и глобальные транзакции должны обрабатываться одинаково, то есть СУБД должна организовать процесс выполнения распределенной транзакции так, чтобы все локальные транзакции, входящие в нее, синхронно фиксировались на затрагиваемых ими узлах распределенной системы.

Однако распределенная транзакция должна фиксироваться только в случае, когда зафиксированы все локальные транзакции, ее составляющие. Если прерывается хотя бы одна из локальных транзакций, должна быть прервана и распределенная транзакция.

Для практической реализации этих требований в СУБД используют механизм двухстадийной фиксации транзакций (two phase commit). При его использовании фиксация распределенных транзакций осуществляется в два этапа (стадии). На первой стадии сервер БД, фиксирующий распределенную транзакцию, посылает команду "приготовиться к фиксации" всем узлам сети (серверам БД), задействованным для выполнения локальных транзакций, инициированных распределенной транзакцией. Все серверы локальных БД должны в ответ сообщить, что они готовы к фиксации. Если хотя бы от одного из серверов ответ не получен, например, если имела место программная или аппаратная ошибка при выполнении локальной транзакции, то сервер распределенной БД производит откат локальных транзакций на всех узлах, включая те, которые прислали оповещение о готовности к фиксации.

Вторая стадия начинается, когда все локальные СУБД готовы к фиксации. Сервер, обрабатывающий распределенную транзакцию, заканчивает ее фиксацию, посылая команду "зафиксировать транзакцию" всем локальным серверам.

Описанный механизм фиксации гарантирует синхронную фиксацию распределенной транзакции на всех узлах сети.

Тиражирование данных. Описанный подход выполнения транзакций в распределенных системах не единственно возможный. Альтернатива ему - технология **тиражирования данных**. Эта технология предполагает отказ от распределенности данных - во всех узлах вычислительной системы должна находиться своя полная копия БД. Средства тиражирования автоматически поддерживают согласованное состояние информации в нескольких БД посредством копирования изменений, вносимых в любую из них. Любая транзакция в такой системе выполняется локально, поэтому нет необходимости в сложной процедуре фиксации.

Узкое место такого подхода - обеспечение тождественности данных в узлах сети. Процесс переноса изменений исходной БД в базы, принадлежащие различным узлам распределенной системы, принято называть тиражированием данных. Функции тиражирования данных выполняет специальный модуль СУБД - сервер тиражирования данных (репликатор): При любых изменениях в тиражируемых данных репликатор копирует их на все остальные узлы системы. Схема тиражирования может быть построена на полном обновлении содержимого таблицы на удаленных серверах (схема с полным обновлением) или же на обновлении только изменившихся записей (быстрое обновление). Если в системе нет необходимости поддерживать постоянную идентичности данных, и БД узлов должны согласовываться лишь периодически, репликатор накапливает изменения и в нужные моменты времени копирует их на другие узлы. Процесс тиражирования данных скрыт от прикладных программ пользователей, репликатор автоматически поддерживает БД в согласованном состоянии.

При использовании технологии тиражирования уменьшается график, так как все запросы обрабатываются локальной СУБД, а на другие узлы передаются только изменения в данных, увеличивается скорость доступа к данным. Кроме того, обрыв связи между узлами не останавливает обработку данных. Однако эта технология не лишена недостатков. Так, невозможно полностью исключить конфликты, возникающие при одновременном изменении одних и тех же данных на разных узлах. При переносе этих изменений в узлах вычислительной системы могут оказаться несогласованные копии БД, в результате чего пользователи различных узлов распределенной БД будут получать разные ответы на одни и те же запросы.

Одно из основных требований к современным OLTP-системам - надежность хранения данных. СУБД должна уметь восстанавливать согласованное состояние базы данных после любых аппаратных и программных сбоев. Для восстановления после сбоев СУБД использует журнал транзакций, который содержит последовательность записей, описывающих изменения в БД.

Общий принцип восстановления после сбоя таков - результаты выполнения транзакций, зафиксированных до сбоя, должны присутствовать в восстановленной БД, результаты незафиксированных транзакций в ней должны отсутствовать. То есть восстанавливается последнее до сбоя согласованное состояние базы данных. Процесс восстановления основан на механизме отката незавершенных транзакций, который описан ранее.

Конечно, журнал транзакций не поможет, если содержимое внешней памяти системы физически уничтожено, утеряна вся информация БД. Для того чтобы избежать подобных ситуаций, реализуют дублированное хранение данных, например, зеркалирование дискового пространства - запись данных одновременно на несколько устройств. После сбоя копируется содержимое БД, а затем, как и в первом случае, на основе журнала откатываются все незавершенные транзакции.

Мониторы транзакций. С ростом сложности распределенных вычислительных систем возникают проблемы эффективного использования их ресурсов. Для решения этих проблем в состав распределенных OLTP-систем вводят дополнительный компонент - монитор транзакций (TPM - transaction processing monitor). Мониторы транзакций выполняют две основные функции: динамическое распределение запросов в системе (выравнивание нагрузки) и оптимизация числа выполняющихся серверных приложений. Кратко рассмотрим эти функции.

Если в системе функционирует несколько серверов, предоставляющих одинаковый сервис, например, доступ к БД, то для оптимизации пропускной способности системы (числа обрабатываемых запросов в единицу времени) необходимо добиться сбалансированной их загрузки. То есть необходимо обеспечить, чтобы на каждый из них поступало примерно равное число пользовательских запросов. При распределении запросов может учитываться также удаленность серверов, их готовность, содержимое запроса. Реализуется функция выравнивания нагрузки следующим образом (см. рис. 1.2).

Запрос клиентского приложения поступает монитору транзакций, который, действуя от имени клиентского приложения, определяет получателя этого запроса.

Для этого он обращается к динамической маршрутной таблице, по которой определяет систему, предоставляющую соответствующий сервис. Если нужный сервис предлагают несколько систем, то в зависимости от используемого алгоритма маршрутизации выбирается одна из них, после чего ей перенаправляется запрос клиентского приложения. Маршрутизация может быть произвольной, когда система выбирается случайным образом, циклической, когда запросы посылаются системам по очереди, либо определяться содержимым запроса, если, например, серверы БД обслуживают разные подмножества данных. Результат выполнения запроса через монитор транзакций перенаправляется приложению, пославшему запрос. Клиентские приложения не знают о том, какой системе будут направлены их запросы, предлагается ли нужный им сервис одним или несколькими серверами, расположен ли нужный сервер локально, удаленно или одновременно локально и удаленно, - в любом случае их запрос будет обработан оптимальным образом/Подобную схему обработки запросов называют "прозрачность местонахождения серверов" (service location transparency).

Скорость обработки транзакций напрямую зависит от числа запущенных серверных приложений. Чем больше приложений одновременно обслуживает запросы, тем выше пропускная способность вычислительной системы. Это увеличение наиболее заметно на многопроцессорных системах, где каждое приложение может работать на отдельном процессоре. В идеале для эффективного использования системных ресурсов нужно по мере необходимости увеличивать или уменьшать число серверных приложений в зависимости от числа обрабатываемых запросов.

Для решения этой задачи мониторы транзакций периодически измеряют отношение числа запросов в очереди к числу работающих серверных приложений. Если это отношение превышает некоторое максимальное пороговое значение (maximum watermark), то запускается дополнительная копия серверного приложения. Если это отношение падает ниже минимального порогового значения (minimum watermark), то одна из копий завершается.

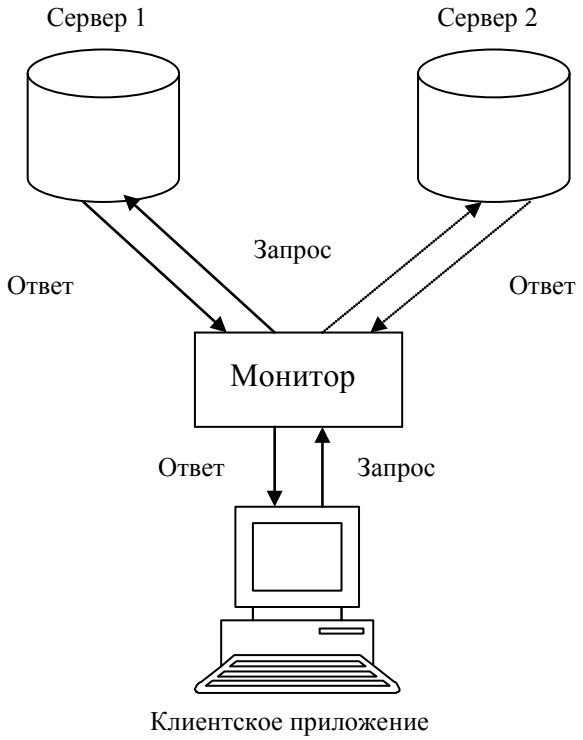

Рис. 1.2. Упрощенная схема работы монитора

На рынке мониторов транзакций доступно довольно много продуктов. В числе наиболее известных: TUXEDO фирмы USL, TOP END фирмы NCR, CICS фирмы IBM, ENCINA фирмы Transarc, ACMS фирмы DEC.

1.3 Неэффективность использования OLTP-систем для анализа данных

Практика использования OLTP-систем показала неэффективность их применения для полноценного анализа информации. Такие системы достаточно успешно решают задачи сбора, хранения и поиска информации, но они не удовлетворяют требованиям, предъявляемым к современным СППР. Подходы, связанные с наращиванием функциональности OLTP-систем, не дали удовлетворительных результатов. Основной причиной неудачи является противоречивость требований, предъявляемых к системам OLTP и СППР. Рассмотрим основные требования, предъявляемые к системам OLTP и СППР.

Степень детализации хранимых данных. Типичный запрос в OLTP-системе, как правило, выборочно затрагивает отдельные записи в таблицах, которые эффективно извлекаются с помощью индексов. В системах анализа, наоборот,

требуется выполнять запросы сразу над большим количеством данных с широким применением группировок и обобщений (суммирования, агрегирования и т. п.).

Например, в стандартных системах складского учета наиболее часто выполняются операции вычисления текущего количества определенного товара на складе, продажи и оплаты товаров покупателями и т.д. В системах анализа выполняются запросы, связанные с определением общей стоимости товаров, хранящихся на складе, категорий товаров, пользующихся наибольшим и наименьшим спросом, обобщение по категориям и суммирование по всем продажам товаров и т.д.

Качество данных. OLTP-системы, как правило, хранят информацию, вводимую непосредственно пользователями систем (операторами ЭВМ). Присутствие "человеческого фактора" при вводе повышает вероятность ошибочных данных и может создать локальные проблемы в системе. При анализе ошибочные данные могут привести к неправильным выводам и принятию неверных стратегических решений. В системах анализа не допускаются ошибки в данных.

Формат хранения данных. OLTP-системы, обслуживающие различные участки работы, не связаны между собой. Они часто реализуются на разных программно-аппаратных платформах. Одни и те же данные в разных базах могут быть представлены в различном виде и могут не совпадать (например, данные о клиенте, который взаимодействовал с разными отделами компании, могут не совпадать в базах данных этих отделов). В процессе анализа такое различие форматов чрезвычайно затрудняет совместный анализ этих данных. Поэтому к системам анализа предъявляется требование единого формата. Как правило, необходимо, чтобы этот формат был оптимизирован для анализа данных (нередко за счет их избыточности).

Допущение избыточных данных. Структура базы данных, обслуживающей OLTP-систему, обычно довольно сложна. Она может содержать многие десятки и даже сотни таблиц, ссылающихся друг на друга. Данные в такой БД сильно нормализованы для оптимизации занимаемых ресурсов. Аналитические запросы к БД очень трудно формулируются и крайне неэффективно выполняются, поскольку

содержат в себе представления, объединяющие большое количество таблиц. При проектировании систем анализа стараются максимально упростить схему БД и уменьшить количество таблиц, участвующих в запросе. С этой целью часто допускают денормализацию (избыточность данных) БД.

Управление данными. Основное требование к OLTP-системам — обеспечить выполнение операций модификации над БД. При этом предполагается, что они должны выполняться в реальном режиме, и часто очень интенсивно. Например, при оформлении розничных продаж в систему вводятся данные о соответствующих документах. Очевидно, что интенсивность ввода зависит от интенсивности покупок и в случае ажиотажа будет очень высокой, а любое промедление ведет к потере клиента. В отличие от OLTP-систем данные в системах анализа меняются редко. Единожды попав в систему, данные уже практически не изменяются. Ввод новых данных, как правило, носит эпизодический характер и выполняется в периоды низкой активности системы (например, раз в неделю на выходных).

Количество хранимых данных. Как правило, системы анализа предназначены для анализа временных зависимостей, в то время как OLTP-системы обычно имеют дело с текущими значениями каких-либо параметров. Например, типичное складское приложение OLTP оперирует с текущими остатками товара на складе, в то время как в системе анализа может потребоваться анализ динамики продаж товара. По этой причине в OLTP-системах допускается хранение данных за небольшой период времени (например, за последний квартал). Для анализа данных, наоборот, необходимы сведения за максимально большой интервал времени.

Характер запросов к данным. В OLTP-системах из-за нормализации БД составление запросов является достаточно сложной работой и требует необходимой квалификации. Поэтому для таких систем заранее составляется некоторый ограниченный набор статических запросов к БД, необходимый для работы с системой (например, наличие товара на складе, размер задолженности покупателей и т. п.). Для СППР невозможно заранее определить необходимые запросы, поэтому к ним предъявляется требование обеспечить формирование произвольных запросов к БД аналитиками.

Время обработки обращений к данным. OLTP-системы, как правило, работают в режиме реального времени, поэтому к ним предъявляются жесткие требования по обработке данных. Например, время ввода документов продажи товаров (расходных накладных) и проверки наличия продаваемого товара на складе должно быть минимально, т. к. от этого зависит время обслуживания клиента. В системах анализа, по сравнению с OLTP, обычно выдвигают значительно менее жесткие требования ко времени выполнения запроса. При анализе данных аналитик может потратить больше времени для проверки своих гипотез. Его запросы могут выполняться в диапазоне от нескольких минут до нескольких часов.

Характер вычислительной нагрузки на систему. Как уже отмечалось ранее, работа с OLTP-системами, как правило, выполняется в режиме реального времени. В связи с этим такие системы нагружены равномерно в течение всего интервала времени работы с ними. Документы продажи или прихода товара оформляются в общем случае постоянно в течение всего рабочего дня. Аналитик при работе с системой анализа обращается к ней для проверки некоторых своих гипотез и получения отчетов, графиков, диаграмм и т. п. При выполнении запросов степень загрузки системы высокая, т. к. обрабатывается большое количество данных, выполняются операции суммирования, группирования и т. п. Таким образом, характер загрузки систем анализа является пиковым.

Приоритетность характеристик системы. Для OLTP-систем приоритетным является высокая производительность и доступность данных, т. к. работа с ними ведется в режиме реального времени. Для систем анализа более приоритетными являются задачи обеспечения гибкости системы и независимости работы пользователей, т. е. то, что необходимо аналитикам для анализа данных.

Противоречивость требований к OLTP-системам и системам, ориентированным на глубокий анализ информации, усложняет задачу их интеграции как подсистем единой СППР. В настоящее время наиболее популярным решением этой проблемы является подход, ориентированный на использование концепции хранилищ данных.

Общая идея хранилищ данных заключается в разделении БД для OLTP-систем и БД для выполнения анализа и последующем их проектировании с учетом соответствующих требований.

Контрольные вопросы

1. В силу каких причин первичный анализ данных был переложен на компьютер?
2. Что означает аббревиатура DSS?
3. Какие классы задач выделяют по степени "интеллектуальности" обработки данных при их анализе?
4. Чем характеризуются OLTP-системы оперативной обработки транзакций?
5. Приведите основные особенности и назначение OLTP-систем.
6. Охарактеризуйте обобщенную архитектуру СППР.
7. В чем заключается основная задача СППР?
8. Охарактеризуйте недостатки OLTP-систем для эффективного анализа данных.

2 Хранилище данных

2.1 Концепция хранилища данных

Стремление объединить в одной архитектуре СППР возможности OLTP-систем и систем анализа, требования к которым во многом противоречивы, привело к появлению концепции хранилищ данных (ХД).

Концепция ХД, так или иначе, обсуждалась специалистами в области информационных систем достаточно давно. Первые статьи, посвященные именно ХД, появились в 1988 г., их авторами были Б. Девлин и П. Мэрфи. В 1992 г. У. Инмон подробно описал данную концепцию в своей монографии "Построение хранилищ данных" ("Building the Data Warehouse, second edition — QED Publishing Group, 1996).

В основе концепции ХД лежит идея разделения данных, используемых для оперативной обработки и для решения задач анализа. Это позволяет применять структуры данных, которые удовлетворяют требованиям их хранения с учетом использования в OLTP-системах и системах анализа. Такое разделение позволяет оптимизировать как структуры данных оперативного хранения (оперативные БД, файлы, электронные таблицы и т. п.) для выполнения операций ввода, модификации, удаления и поиска, так и структуры данных, используемые для анализа (для выполнения аналитических запросов). В СППР - эти два типа данных называются соответственно оперативными источниками данных (ОИД) и хранилищем данных.

В своей работе Инмон дал следующее определение ХД.

Хранилище данных — предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.

Рассмотрим свойства ХД более подробно.

Предметная ориентация. Это фундаментальное отличие ХД от ОИД. Разные ОИД могут содержать данные, описывающие одну и ту же предметную область с разных точек зрения (например, с точки зрения бухгалтерского учета, складского учета, планового отдела и т. п.). Решение, принятое на основе только одной точки зрения, может быть неэффективным или даже неверным. ХД позволяют интегрировать информацию, отражающую разные точки зрения на одну предметную область. Предметная ориентация позволяет также хранить в ХД только те данные, которые нужны для их анализа (например, для анализа нет смысла хранить информацию о номерах документов купли-продажи, в то время как их содержимое — количество, цена проданного товара — необходимо). Это существенно сокращает затраты на носители информации и повышает безопасность доступа к данным.

Интеграция. ОИД, как правило, разрабатываются в разное время несколькими коллективами с собственным инструментарием. Это приводит к тому, что данные, отражающие один и тот же объект реального мира в разных системах, описывают его по-разному. Обязательная интеграция данных в ХД позволяет решить эту проблему, приведя данные к единому формату.

Поддержка хронологии. Данные в ОИД необходимы для выполнения над ними операций в текущий момент времени. Поэтому они могут не иметь привязки ко времени. Для анализа данных часто бывает важно иметь возможность отслеживать хронологию изменений показателей предметной области. Поэтому все данные, хранящиеся в ХД, должны соответствовать последовательным интервалам времени.

Неизменяемость. Требования к ОИД накладывают ограничения на время хранения в них данных. Те данные, которые не нужны для оперативной обработки, как правило, удаляются из ОИД для уменьшения занимаемых ресурсов. Для анализа, наоборот, требуются данные за максимально большой период времени. Поэтому, в отличие от ОИД, данные в ХД после загрузки только читаются. Это позволяет существенно повысить скорость доступа к

данным, как за счет возможной избыточности хранящейся информации, так и за счет исключения операций модификации.

При реализации в СППР концепции ХД данные из различных ОИД загружаются в единое хранилище. Собранные данные приводятся к единому формату, согласовываются и обобщаются. Аналитические запросы адресуются к ХД.

Такая модель неизбежно приводит к дублированию информации в ОИД и в ХД. Однако Инмон в своей работе утверждает, что избыточность данных, хранящихся в СППР, не превышает 1 %! Это можно объяснить следующими причинами. При загрузке информации из ОИД в ХД данные фильтруются. Многие из них не попадают в ХД, поскольку лишены смысла с точки зрения использования в процедурах анализа. Информация в ОИД носит, как правило, оперативный характер, и данные, потеряв актуальность, удаляются. В ХД, напротив, хранится историческая информация. С этой точки зрения дублирование содержимого ХД данными ОИД оказывается весьма незначительным. В ХД хранится обобщенная информация, которая в ОИД отсутствует. Во время загрузки в ХД данные очищаются (удаляется ненужная информация), и после такой обработки они занимают гораздо меньший объем.

Избыточность информации можно свести к нулю, используя виртуальное ХД. В данном случае в отличие от классического (физического) ХД данные из ОИД не копируются в единое хранилище. Они извлекаются, преобразуются и интегрируются непосредственно при выполнении аналитических запросов в оперативной памяти компьютера. Фактически такие запросы напрямую адресуются к ОИД. Основными достоинствами виртуального ХД являются минимизация объема памяти, занимаемой на носителе информацией и работа с текущими, детализированными данными.

Однако такой подход обладает многими недостатками. Время обработки запросов к виртуальному ХД значительно превышает соответствующие показатели для физического хранилища. Кроме того, структуры оперативных БД, рассчитанные на интенсивное обновление одиночных записей, в высокой

степени нормализованы. Для выполнения же аналитического запроса требуется объединение большого числа таблиц, что также приводит к снижению быстродействия.

Интегрированный взгляд на виртуальное хранилище возможен только при выполнении условия постоянной доступности всех ОИД. Таким образом, временная недоступность хотя бы одного из источников может привести либо к невыполнению аналитического запроса, либо к неверным результатам.

Выполнение сложных аналитических запросов над ОИД требует значительных ресурсов компьютеров. Это приводит к снижению быстродействия OLTP-систем, что недопустимо, т. к. время выполнения операций в таких системах часто весьма критично.

Различные ОИД могут поддерживать разные форматы и кодировки данных. Часто на один и тот же вопрос может быть получено несколько вариантов ответа. Это может быть связано с несинхронностью моментов обновления данных в разных ОИД, отличиями в описании одинаковых объектов и событий предметной области, ошибками при вводе, утерей фрагментов архивов и т.д. В таком случае цель — формирование единого непротиворечивого взгляда на объект управления, может быть не достигнута.

Главным же недостатком виртуального хранилища следует признать практическую невозможность получения данных за долгий период времени. При отсутствии физического хранилища доступны только те данные, которые на момент запроса есть в ОИД. Основное назначение OLTP-систем — оперативная обработка текущих данных, поэтому они не ориентированы на хранение данных за длительный период времени. По мере устаревания данные выгружаются в архив и удаляются из оперативной БД.

Несмотря на преимущества физического ХД перед виртуальным его реализация представляет собой достаточно трудоемкий процесс. Остановимся на основных проблемах создания ХД:

необходимость интеграции данных из неоднородных источников в распределенной среде;

потребность в эффективном хранении и обработке очень больших объемов информации;

необходимость наличия многоуровневых справочников метаданных;

повышенные требования к безопасности данных.

Рассмотрим эти проблемы более подробно.

Необходимость интеграции данных из неоднородных источников в распределенной среде. ХД создаются для интегрирования данных, которые могут поступать из разнородных ОИД, физически размещающихся на разных компьютерах: БД, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников, сайтов различного назначения. При создании ХД приходится решать задачу построения системы, согласованно функционирующей с неоднородными программными средствами и решениями. При выборе средств реализации ХД приходится учитывать множество факторов, включающих уровень совместимости различных программных компонентов, легкость их освоения и использования, эффективность функционирования и т. д.

Потребность в эффективном хранении и обработке очень больших объемов информации. Свойство неизменности ХД предполагает накопление в нем информации за долгий период времени, что должно поддерживаться постоянным ростом объемов дисковой памяти. Ориентация на выполнение аналитических запросов и связанная с этим денормализация данных приводят к нелинейному росту объемов памяти, занимаемой ХД при возрастании объема данных. Исследования, проведенные на основе тестового набора TPC-D, показали, что для баз данных объемом в 100 Гбайт требуется память, в 4,87 раза большая объемом, чем нужно для хранения только полезных данных [1].

Необходимость многоуровневых справочников метаданных. Для систем анализа наличие развитых метаданных (данных о данных) и средств их предоставления конечным пользователям является одним из основных условий успешной реализации ХД. Метаданные необходимы пользователям СППР для понимания структуры информации, на основании которой принимается

решение. Например, прежде чем менеджер корпорации задаст системе свой вопрос, он должен понять, какая информация имеется, насколько она актуальна, можно ли ей доверять, сколько времени может занять формирование ответа и т. д. При создании ХД необходимо решать задачи хранения и удобного представления метаданных пользователям.

Повышение требований к безопасности данных. Собранная вместе и согласованная информация об истории развития корпорации, ее успехах и неудачах, о взаимоотношениях с поставщиками и заказчиками, об истории и состоянии рынка дает возможность анализа прошлой и текущей деятельности корпорации и построения прогнозов для будущего. Очевидно, что подобная информация является конфиденциальной и доступ к ней ограничен в пределах самой компании, не говоря уже о других компаниях. Для обеспечения безопасности данных приходится решать вопросы аутентификации пользователей, защиты данных при их перемещении в хранилище данных из оперативных баз данных и внешних источников, защиты данных при их передаче по сети и т. п.

Снижения затрат на создание ХД можно добиться, создавая его упрощенный вариант — витрину данных (Data Mart).

Витрина данных (ВД) — это упрощенный вариант ХД, содержащий только тематически объединенные данные.

ВД максимально приближена к конечному пользователю и содержит данные, тематически ориентированные на него (например, ВД для работников отдела маркетинга может содержать данные, необходимые для маркетингового анализа). ВД существенно меньше по объему, чем ХД, и для ее реализации не требуется больших затрат. Они могут быть реализованы как самостоятельно, так и вместе с ХД.

Самостоятельные ВД часто появляются в организации исторически и встречаются в крупных организациях с большим количеством независимых подразделений, решающих собственные аналитические задачи.

Достоинствами такого подхода являются:

- проектирование ВД для ответов на определенный круг вопросов;
- быстрое внедрение автономных ВД и получение отдачи;
- упрощение процедур заполнения ВД и повышение их производительности за счет учета потребностей определенного круга пользователей.

Недостатками автономных ВД являются:

- многократное хранение данных в разных ВД, что приводит к увеличению расходов на их хранение и потенциальным проблемам, связанным с необходимостью поддержания непротиворечивости данных;
- отсутствие консолидированности данных на уровне предметной области, и, следовательно — отсутствие единой картины.

В последнее время все более популярной становится идея совместить ХД и ВД в одной системе. В этом случае ХД используется в качестве единственного источника интегрированных данных для всех ВД.

ХД представляет собой единый централизованный источник информации для всей предметной области, а ВД являются подмножествами данных из хранилища, организованными для представления информации по тематическим разделам данной области. Конечные пользователи имеют возможность доступа к детальным данным хранилища, если данных в витрине недостаточно, а также для получения более полной информационной картины. Достоинствами такого подхода являются:

- простота создания и наполнения ВД, поскольку наполнение происходит из единого стандартизованного надежного источника очищенных данных — из ХД;
- простота расширения СППР за счет добавления новых ВД;
- снижение нагрузки на основное ХД.

К недостаткам относятся:

- избыточность (данные хранятся как в ХД, так и в ВД);
- дополнительные затраты на разработку СППР с ХД и ВД.

2.2 Организация хранилища данных

Все данные в ХД делятся на три основные категории (рис. 2.1):

1. детальные данные;
2. агрегированные данные;
3. метаданные.

Детальными являются данные, переносимые непосредственно из ОИД. Они соответствуют элементарным событиям, фиксируемым OLTP-системами (например, продажи, эксперименты и др.). Принято разделять все данные на измерения и факты. *Измерениями* называются наборы данных, необходимые для описания событий (например, города, товары, люди и т. п.). *Фактами* называются данные, отражающие сущность события (например, количество проданного товара, результаты экспериментов и т. п.). Фактические данные могут быть представлены в виде числовых или категориальных значений.

В процессе эксплуатации ХД необходимость в ряде детальных данных может снизиться. Ненужные детальные данные могут храниться в архивах в сжатом виде на более емких накопителях с более медленным доступом (например, на магнитных лентах). Данные в архиве остаются доступными для обработки и анализа. Регулярно используемые для анализа данные должны храниться на накопителях с быстрым доступом (например, на жестких дисках).

На основании детальных данных могут быть получены *агрегированные* (обобщенные) данные. Агрегирование происходит путем суммирования числовых фактических данных по определенным измерениям.

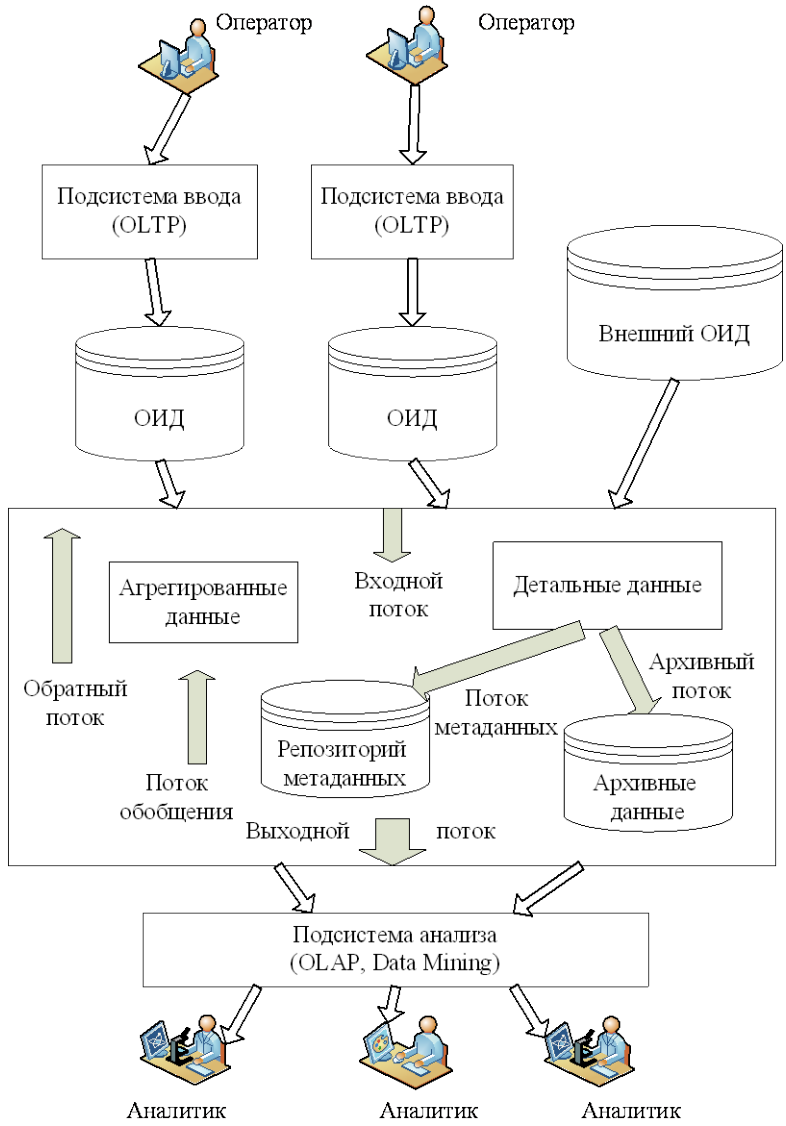

Рис. 2.1 Потоки данных в Хранилище Данных

В зависимости от возможности агрегировать данные они подразделяются на следующие типы:

- аддитивные — числовые фактические данные, которые могут быть просуммированы по всем измерениям;
- полуаддитивные — числовые фактические данные, которые могут быть просуммированы только по определенным измерениям;
- неаддитивные — фактические данные, которые не могут быть просуммированы ни по одному измерению.

Большинство пользователей СППР работают не с детальными, а с агрегированными данными. Архитектура ХД должна предоставлять быстрый и удобный

способ получать интересующую пользователя информацию. Для этого необходимо часть агрегированных данных хранить в ХД, а не вычислять их при выполнении аналитических запросов. Очевидно, что это ведет к избыточности информации и увеличению размеров ХД. Поэтому при проектировании таких систем важно добиться оптимального соотношения между вычисляемыми и хранящимися агрегированными данными. Те данные, к которым редко обращаются пользователи, могут вычисляться в процессе выполнения аналитических запросов. Данные, которые требуются более часто, должны храниться в ХД.

Для удобства работы с ХД необходима информация о содержащихся в нем данных. Такая информация называется ***метаданными*** (данные о данных). Согласно концепции Дж. Захмана, метаданные должны отвечать на следующие вопросы — что, кто, где, как, когда и почему:

что (описание объектов) — метаданные описывают объекты предметной области, информация о которых хранится в ХД. Такое описание включает: атрибуты объектов, их возможные значения, соответствующие поля в информационных структурах ХД, источники информации об объектах и т. п.;

кто (описание пользователей) — метаданные описывают категории пользователей, использующих данные. Они описывают права доступа к данным, а также включают в себя сведения о пользователях, выполнявших над данными различные операции (ввод, редактирование, загрузку, извлечение и т. п.);

где (описание места хранения) — метаданные описывают местоположение серверов, рабочих станций, ОИД, размещенные на них программные средства и распределение между ними данных;

как (описание действий) — метаданные описывают действия, выполняемые над данными. Описываемые действия могли выполняться как в процессе переноса из ОИД (например, исправление ошибок, расщепление полей и т. п.), так и в процессе их эксплуатации в ХД;

когда (описание времени) — метаданные описывают время выполнения разных операций над данными (например, загрузка, агрегирование, архивирование, извлечение и т. п.);

почему (описание причин) — метаданные описывают причины, повлекшие выполнение над данными тех или иных операций. Такими причинами могут быть требования пользователей, статистика обращений к данным и т. п.

Так как метаданные играют важную роль в процессе работы с ХД, то к ним должен быть обеспечен удобный доступ. Для этого они сохраняются в репозитории метаданных с удобным для пользователя интерфейсом. В последующих главах нашего пособия мы более детально остановимся на процессах определения и использования метаданных ХД [1].

Данные, поступающие из ОИД в ХД, перемещаемые внутри ХД и поступающие из ХД к аналитикам, образуют следующие информационные потоки (рис. 2.1):

входной поток (Inflow) — образуется данными, копируемыми из ОИД в ХД;

поток обобщения (Upflow) — образуется агрегированием детальных данных и их сохранением в ХД;

архивный поток (Downflow) — образуется перемещением детальных данных, количество обращений к которым снизилось;

поток метаданных (Meta Flow) образуется переносом информации о данных в репозиторий данных;

выходной поток (Outflow) — образуется данными, извлекаемыми пользователями;

обратный поток (Feedback Flow) — образуется очищенными данными, записываемыми обратно в ОИД.

Самый мощный из информационных потоков — входной, связан с переносом данных из ОИД. Обычно информация не просто копируется в ХД, а подвергается обработке: данные очищаются и обогащаются за счет добавления новых атрибутов. Исходные данные из ОИД объединяются с информацией из внешних источников — текстовых файлов, сообщений электронной почты,

электронных таблиц и др. При разработке ХД не менее 60% всех затрат связано с переносом данных.

Процесс переноса, включающий в себя этапы извлечения, преобразования и загрузки, называют ETL-процессом (E — extraction, T — transformation, L — loading: извлечение, преобразование и загрузка, соответственно). Программные средства, обеспечивающие его выполнение, называются ETL-системами. Традиционно ETL-системы использовались для переноса информации из устаревших версий информационных систем в новые. В настоящее время ETL-процесс находит все большее применение для переноса данных из ОИД в ХД и ВД.

Рассмотрим более подробно этапы ETL-процесса (рис. 2.2).

Извлечение данных. Чтобы начать ETL-процесс, необходимо извлечь данные из одного или нескольких источников и подготовить их к этапу преобразования. Можно выделить два способа извлечения данных:

1. Извлечение данных вспомогательными программными средствами непосредственно из структур хранения информации (файлов, электронных таблиц, БД и т.п.). Достоинствами такого способа извлечения данных являются:

отсутствие необходимости расширять OLTP-систему (это особенно важно, если ее структура закрыта);

данные могут извлекаться с учетом потребностей процесса переноса.

2. Выгрузка данных средствами OLTP-систем в промежуточные структуры. Достоинствами такого подхода являются:

возможность использовать средства OLTP-систем, адаптированные к структурам данных;

средства выгрузки изменяются вместе с изменениями OLTP-систем и ОИД;

возможность выполнения первого шага преобразования данных за счет определенного формата промежуточной структуры хранения данных.

Преобразование данных. После того как сбор данных завершен, необходимо преобразовать их для размещения на новом месте. На этом этапе выполняются следующие процедуры:

обобщение данных (aggregation) — перед загрузкой данные обобщаются. Процедура обобщения заменяет многочисленные детальные данные относительно небольшим числом агрегированных данных. Например, предположим, что данные о продажах за год занимают в нормализованной базе данных несколько тысяч записей. После обобщения данные преобразуются в меньшее число кратких записей, которые будут перенесены в ХД;

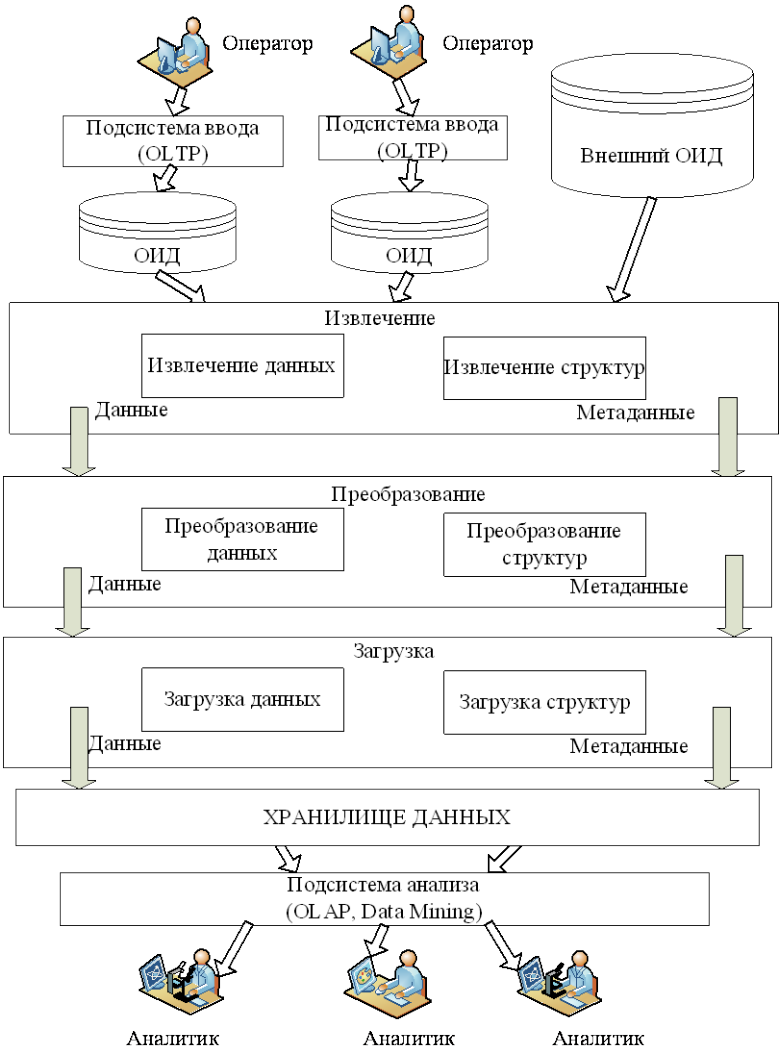

Рис. 2.2. Основные этапы ETL-процесс

перевод значений (value translation) — в ОИД данные часто хранятся в закодированном виде для того, чтобы сократить избыточность данных и память для их хранения. Например, названия товаров, городов, специальностей и т. п.

могут храниться в сокращенном виде. Поскольку ХД содержат обобщенную информацию и рассчитаны на простое использование, закодированные данные обычно заменяют на более понятные описания;

создание полей (field derivation) — при создании полей для конечных пользователей создается и новая информация. Например, ОИД содержит одно поле для указания количества проданных товаров, а второе — для указания цены одного экземпляра. Для исключения операции вычисления стоимости всех товаров можно создать специальное поле для ее хранения во время преобразования данных;

очистка данных (cleaning) — направлена на выявление и удаление ошибок и несоответствий в данных с целью улучшения их качества. Проблемы с качеством встречаются в отдельных ОИД, например, в файлах и БД могут быть ошибки при вводе, отдельная информация может быть утрачена, могут присутствовать "загрязнения" данных и др. Очистка также применяется для согласования атрибутов полей таким образом, чтобы они соответствовали атрибутам базы данных назначения.

Загрузка данных. После того как данные преобразованы для размещения в ХД, осуществляется этап их загрузки. При загрузке выполняется запись преобразованных детальных и агрегированных данных. Кроме того, при записи новых детальных данных часть старых данных может переноситься в архив.

2.3 Метаданные в хранилищах данных

Рассмотрим более подробно задачу создания и ведения метаданных в хранилищах данных. К сожалению, часто встречающееся на страницах компьютерной литературы определение (метаданные - это "данные о данных") вносит более путаницы в толкование термина "метаданные" (metadata), чем поясняет его смысл. Образное определение (метаданные - это "тень данных"), принадлежащее Б. Инмону и понятное ИТ-специалистам по системам складирования данных, также не вносит большей ясности.

Попробуем уточнить смысл термина "метаданные". Метаданные есть в любой информационной системе, будь то OLTP-система, система складирования данных или корпоративный портал. Чтобы осознать этот факт, нужно вспомнить о том, что любая ИС реализует "вопросно-ответное" отношение на конечном алфавите. Метаданные дают возможность пользователям понять, на какие вопросы может отвечать данная ИС. С технической точки зрения метаданные - это совокупность спецификаций и данных, которая в целом дает ответы на вопросы, какова степень охвата предметной области в ИС, какие данные в ней представлены, какова архитектура системы и т. д.

В частности, метаданные содержат семантическую интерпретацию или толкование содержания элементов данных, циркулирующих в ИС. Но это далеко не все. В метаданные включается также описание вычислительной среды, предметных областей, информационной безопасности и многое другое, что непосредственно влияет на эффективность применения ИС, в первую очередь, конечными пользователями и разработчиками.

Далее под метаданными будет пониматься совокупность элементов данных и спецификаций, содержащих описание данных ИС и процессов их обработки.

Неоднозначность толкования термина "метаданные" определяется тем обстоятельством, что последние должны удовлетворить технические и семантические потребности всех групп пользователей ИС. Каждая группа пользователей имеет свои требования к метаданным.

Так, руководство компании (основные пользователи информационных систем руководителей) хочет знать, что оно получит от системы, как быстро получить ответ на интересующий вопрос, и все это в терминах, понятных лицам, принимающим решения. Руководство организации не имеет времени (да и не понимает смысла) для изучения объемных инструкций по эксплуатации ИС.

В то же время специалисты организации, например, пользователи бухгалтерских систем, хотят, чтобы такая система "разговаривала на их профессиональном языке", вплоть до того, что разработчики таких систем (1С-

бухгалтерия) оснащают свои системы специальным формальным языком. Бухгалтеры вряд ли будут основательно изучать SQL.

Аналитиков компании занимают более сложные вопросы, в частности о происхождении и достоверности данных. Руководство требует от них информации для поддержки принятия обоснованных решений. Цена ошибки может быть велика - значительные убытки или сорванные контракты.

Разработчиков приложений интересует информации о модели данных системы для создания или внедрения дополнительных бизнес-приложений. Они хотят знать, что находится в таблицах БД системы и в каком формате.

Реализация столь многообразных интересов в метаданных и порождает большое число компонентов, которые составляют метаданных. Проектирование и разработка метаданных являются одной из самых сложных и трудоемких задач проектирования и разработки ИС.

Разработку метаданных можно отнести к ИТ-дисциплине, которую называют *управлением данными*. Решение задач управления данными часто возлагается на администраторов данных, которых не следует путать с администраторами баз данных и компьютерных сетей.

Нужно отметить, что некоторые принципы управления данными в ХД и БД OLTP-систем имеют существенные отличия. Так, например, характерный для OLTP-систем принцип, состоящий в том, что существует только одно правильно отображающее семантику определение данных в системе, не является верным ХД. Это связано с различными временными горизонтами данных в этих системах.

Задача проектировщика ХД при проектировании метаданных состоит:

- в идентификации объектов ХД и их атрибутов;
- идентификации источников данных;
- описании семантики данных источников и ХД;
- описании алгоритмов преобразования и агрегации данных;
- описании путей доступа к данным и т. п.

Весь этот комплекс вопросов, которые должен решить проектировщик ХД при проектировании метаданных, требует от него тщательной их проработки еще на начальных стадиях проектирования. Проектирование ХД должно начинаться с проектирования метаданных и должно заканчиваться им же.

Рассмотрим основные функции метаданных и их состав, характерный для ХД.

2.3.1. Функции метаданных в хранилище данных

Роль метаданных в системах складирования данных значительно важнее, чем в системах оперативной обработки информации. Если в системах оперативной обработки информации интерфейс системы является настроенным на бизнес-процессы и бизнес-процедуры обработки данных конкретных специалистов и понятным им после специального обучения, то интерфейс систем складирования данных конструируется таким образом, чтобы помимо всего прочего отвечать на непредопределенные вопросы (ad hoc). Как правило, такие вопросы формулируются в терминах предметной области и специалистами, для которых ИТ-технологии не являются основной профессией, аналитиками, менеджерами среднего и высшего уровня [2].

Таким образом, одним из главных аспектов использования метаданных в ХД является их предметная ориентация. Основные вопросы, на которые должны ответить метаданные, - это какие данные представлены в системе и как их получить в нужном для анализа виде.

Первой основной функцией метаданных в ХД является представлением соответствия данных источников и данных ХД. Как правило, это описание представляет собой фиксацию взаимосвязи атрибутов данных источника и атрибутов данных ХД, правила преобразования первых во вторые, изменение в наименовании данных, их физических характеристиках и т. д.

Такая информация позволяет идентифицировать источники данных для ХД, правильность данных в ХД и их корректность.

Вторая основная функция метаданных в ХД является управление данными во времени. Время жизни данных в ХД, как правило, 5-10 лет, а то и более. Для систем оперативной обработки информации это от нескольких дней до нескольких месяцев. Затем данные архивируются в случае необходимости.

Таким образом, временной горизонт данных в ХД гораздо больше, и это обстоятельство изменяет коренным образом некоторые принципы управления данными. Например, в системах оперативной обработки данных в одно и то же время существует только одно корректное определение данных. Для ХД это не так.

Структура данных (схема или модель данных) в системах оперативной обработки данных меняется во времени, т. е. данные в таких системах в разное время имеют различные формы представления. Эти изменения должны быть зафиксированы в ХД.

Таким образом, в ХД в одно и то же время может существовать несколько схем данных, отвечающих различным периодам эволюции источников данных. Запись о таких структурных изменениях сохраняется в метаданных ХД. На основании таких записей аналитики получают ответы на вопросы, какими данными и за какие периоды они располагают.

Третья, и немаловажная, функция метаданных в ХД - это поддержка версионности. Эта функция тесно связана с управлением данными с большим временным горизонтом. Метаданные должны отражать изменения внутренней структуры данных источников и, следовательно, должны сами изменяться, для того чтобы обеспечить непрерывность истории изменения структуры данных ХД. Поддержка версионности метаданных позволяет в каждый момент времени в прошлом обеспечить правильное описание модели данных, а аналитики получают возможность знать, какие данные, когда и как попали в ХД.

Четвертая основная функция метаданных в ХД - это интерпретация данных в терминах бизнес-пользователей. Метаданные должны поддерживать в запросах понятную для пользователя терминологию, независимо от того, какие правила наименования атрибутов было использовано проектировщиком ХД.

Пятая основная функция метаданных - это обеспечение открытости (доступности другим информационным системам) системы складирования данных для ее интеграции с другими аналитическими системами компании. Опрос метаданных ХД другой системой позволяет последней выяснить структуру данных ХД и поддерживать обмен данными между системами.

2.3.2 Состав метаданных в хранилище данных

В настоящее время нет строго определенных требований к составу метаданных информационных систем. О предлагаемых стандартах метаданных, и в частности для ХД, мы поговорим несколько позже. Сейчас остановимся на перечне тех компонентов метаданных, которые обязательно должны присутствовать в ХД.

Базовые компоненты метаданных ХД не сильно отличаются от базовых компонентов систем оперативной обработки данных. Это описание таблиц, их атрибутов и ключей. Существенное отличие для ХД - поддержка версионности метаданных. Базовые компоненты говорят нам, какие данные сохраняются в ХД. Следующая, характерная для ХД, группа компонентов метаданных - это описание преобразований. Как правило, описание преобразований данных для ХД включает в себя:

идентификацию полей источников данных;

соответствие между атрибутами сущностей источников данных и атрибутами объектов ХД;

преобразования атрибутов;

физические характеристики преобразований;

преобразования таблиц кодировки и ссылочных таблиц;

изменения наименований (соответствие имен источников и объектов ХД);

изменение ключевых атрибутов;

значение полей по умолчанию;

логика (алгоритмы) формирования данных ХД из нескольких источников (приоритетность источников);

алгоритмы трансформации данных и т. д.

Компоненты преобразования говорят нам о том, как данные в ХД были получены.

Немаловажным компонентом метаданных ХД является история поступления в него данных. Компонент метаданных история экстрагирования (поступления) данных, говорит нам о том, когда данные поступили в ХД, а также позволяет судить о полноте представления данных в ХД. Для проведения анализа данных такая информация является очень важной, поскольку на ее основе формируются утверждения пользователей о корректности анализа и надежности его результатов.

Информация о синонимии, или о терминологическом соответствии понятий, - это еще один компонент метаданных ХД. Он включает в себя альтернативные наименования для данных ХД. Такая информация, как правило, делает ХД более "дружелюбным" для пользователей.

Одним из важных компонентов метаданных является информация о состояниях и статистики использования данных ХД. Эта информация составляет основу для оптимизации производительности ХД. К такой информации относятся данные о числе строк в таблицах, скорости роста таблиц, статистических профилях использования таблиц (среднее и максимальное число запросов на день), статистика архивирования и удаления данных, индексирование таблиц, частота использования индексов в запросах и т. п.

Еще одним компонентом метаданных ХД являются алгоритмы агрегации и суммирования данных, критерии выборки из источников, правила преобразования данных источников перед загрузкой в ХД, описание взаимосвязей между объектами ХД, их кардинальность и т. п. Такая информация играет важную роль при проведении анализа данных и часто требуется аналитиками для решения вопросов надежности результатов анализа.

Информация о том, кто отвечает за содержание и актуальность различных источников данных, составляет еще один компонент метаданных. Эта ин-

формация важна для группы сопровождения ХД и позволяет организационно решать вопросы качества, точности и надежности данных в ХД.

Часто в метаданные включаются компоненты, описывающие шаблоны доступа к данным (когда и как данные мигрировали на другой уровень хранения). Они используются также для оптимизации физического потока данных в ХД и для оптимизации производительности. Иногда, алгоритмы обработки данных в ХД используют информацию из объектов внешних систем, так называемые таблицы расширения (таблицы кодировок и электронных справочников). В этом случае в метаданных ХД необходимо фиксировать описание таких таблиц и историю их изменения, поскольку в случае изменения кодов необходимо провести соответствующие изменения в обработке данных ХД, чтобы не потерять исторические связи в данных.

Часто проектировщики ХД включают в состав метаданных дополнительную информацию, важную с их точки зрения.

Из сказанного выше ясно, что проектирование метаданных ХД является достаточно сложной и креативной задачей для проектировщика, решение которой требует часто литературного мастерства, знания предметной области ХД и очень много времени.

2.3.3 Стандарты метаданных

Как правило, любая стандартизация начинается с построения классификации объектов, для которых разрабатывается стандарт. Стандарт в области метаданных не является исключением из этого правила. Второй аспект разработки любого промышленного стандарта - это учет предложений ведущих производителей. И третий момент - это концепция, которая позволяет связать разрабатываемый стандарт с уже действующими в данной предметной области стандартами.

Предметом обсуждения этого раздела является спецификация "Общая метамодель хранилища данных" (Common Warehouse Metamodel, CWM), которая является одним из стандартов, использующих XML-технологии. Этот

стандарт описывает обмен метаданными в информационных системах, использующих технологии ХД, системах деловой осведомленности BI (Business Intelligence) и системах управления знаниями KM (Knowledge Management).

Классификация метаданных. На очень высоком уровне метаданные могут быть разделены на 2 категории: *разделяемые метаданные* (Shared meta data) и *уникальные метаданные* (Unique meta data).

Элементы разделяемых метаданных необходимы для точного определения объектов и их семантики для ХД в целом. Так, определение объекта "Покупатель", например, должно быть согласованным и для источников и для ХД, т. е. иметь одинаковый смысл в рамках информационных систем масштаба организации. Уникальные метаданные описывают уникальные объекты системы, например атрибуты измерений или фактов.

Другой подход к построению классификации метаданных - это разделить элементы метаданных по их функциональному назначению: предметно-ориентированные метаданные (Business meta data), технические метаданные (Technical meta data), метаданные процесса обработки данных (Process meta data).Предметно-ориентированные метаданные содержат определения сущностей предметной области в терминах пользователей, логические отображения между данными на различных уровнях их представления в системе, описания словаря ХД как БД. Технические метаданные содержат определения и данные о физических объектах ХД. Это определения наименований таблиц и их колонок, ограничений, правил физических преобразований данных и т. п. Метаданные процесса обработки данных содержат информацию, связанную с процессом обработки данных, такую, как статистика загрузки, расписание загрузки данных и т. п.

Исследователями в области проектирования ХД предлагаются и другие подходы к построению классификации, но при этом, как правило, используются перечисленные выше два принципа.

Ведущие производители программного обеспечения в области хранилищ данных ведут жесткую конкурентную борьбу за лидерство. И выдвижение

своих решений в качестве промышленного стандарта для систем этого класса является неотъемлемой составляющей этой борьбы.

В середине 1998 г. корпорации IBM, Oracle, Unisys, Hyperion, SAS и ряд других поставщиков программного обеспечения представили в организацию Object Management Group (OMG) спецификацию стандарта "Обмен общими метаданными хранилища данных" (Common Warehouse Metadata Interchange, CWMI). Во второй половине 1999 г. корпорация Microsoft передала на рассмотрение в консорциум Meta Data Coalition (MDC) разработанный ею стандарт "Открытая информационная модель" (Open Information Model, OIM). В 2000 г. произошло слияние MDC и OMG, а спустя год, в начале февраля 2001 г., была опубликована первая версия спецификации "Общая метамодель хранилища данных"

CWMI определяет интерфейсы, которые могут быть использованы для обмена метаданными между ХД и аналитическими приложениями с помощью инструментальных средств ХД, программно-аппаратных платформ и репозиториев метаданных в распределенных гетерогенных вычислительных средах.

CWMI основывается на трех основных стандартах:

- UML – Unified Modeling Language (стандарт OMG для моделирования);
- MOF – Meta Object Facility (стандарт OMG для метамоделирования и репозитория метаданных);
- XMI – XML Metadata Interchange (стандарт OMG обмена метаданными).

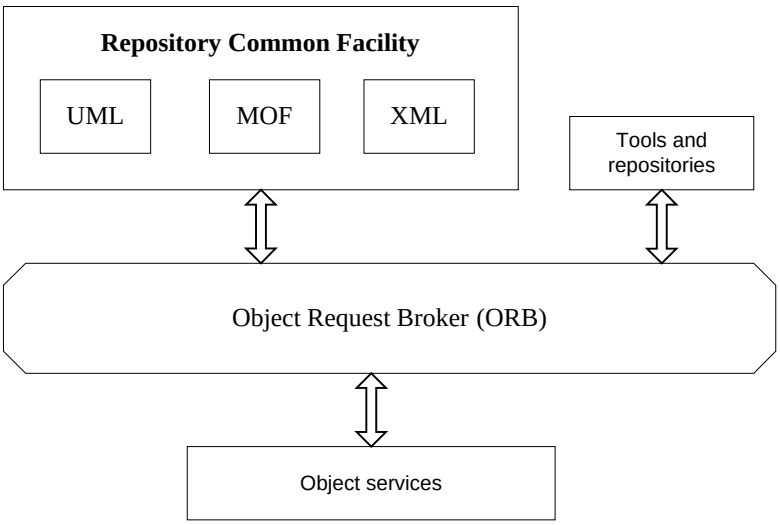

Рис. 2.3. Ядро архитектуры репозитория метаданных OMG.

Стандарт UML определяет язык объектно-ориентированного моделирования, который поддерживает ряд графических нотаций. Стандарт MOF устанавливает гибкие средства для определения модели метаданных и обеспечивает программные средства для хранения и доступа к метаданным в репозитории. Стандарт XMI определяет спецификации для обмена метаданными в формате стандарта XML. Использование этих стандартов не накладывает сильных ограничений при реализации модели метаданных в конкретных системах складирования данных.

Перечисленные выше стандарты формируют ядро архитектуры репозитория метаданных OMG, как показано на рис. 2.3.

Общая метамодель хранилища данных. Основные элементы общей метамодели хранилища данных (CWM) включают в себя:

• четырехуровневую архитектуру метамоделирования при работе с метаданными в распределенных репозиториях;

• использование нотации UML для представления метамодели и моделей данных;

• использование стандартных информационных моделей UML для описания семантики объектно-ориентированного анализа и проектирования моделей;

• использование MOF для определения и работы с метамоделями с использованием интерфейсов CORBA;

• использование XML для организации обмена метаданными.

Четырехуровневая архитектура метамоделирования аналогична общепринятой архитектуре моделирования. Стандарт расширяет базовую метамодель метамоделями для реляционных и многомерных данных, для преобразования, функций OLAP и ХД, включая процессы и операции. Спецификацию CWMI можно рассматривать как язык, предназначенный для определения моделей ХД. Спецификация CWM расширяет язык UML: каждый метакласс CWMI наследуется непосредственно или косвенно из метаклассов UML. Так, метакласс "Реляционная таблица (Relational Table) CWMI" является непосредственным наследником класса UML (UML Class), а "Реляционный столбец" (Relational Column) - прямой потомок атрибута UML (UML Attribute).

Таким образом, спецификация CWMI определяет метамодель и для предметно-ориентированных метаданных и для технических метаданных. Эта метамодель используется для обмена экземплярами метаданных между гетерогенным программным обеспечением, поставляемым различными производителями. Системы, которые поддерживают метамодель CWMI, обмениваются данными в форматах, которые согласуются с этой моделью.

Стандарт OMG "Средства метаобъекта" (Meta Object Facility, MOF) определяет общие интерфейсы и семантику для взаимодействующих метамоделей. Являясь подмножеством UML, он представляет собой пример метаметамодели, или модели метамодели (подмножество). В сферу действия этого стандарта входит определение языка описания интерфейса (Interface Definition Language), который устанавливает правила управления моделями с помощью программных API). Все модели CWMI выражаются на UML и реализуют семантику MOF.

Стандарт OMG "Обмен метаданными XML" (XML Metadata Interchchange, XMI) устанавливает правила преобразования метамоделей MOF в XML. XMI непосредственно задействован в обмене метамоделями. Метамодели MOF

транслируются в XML DTD, а модели - в XML-документы, которые согласуются со своими DTD.

Таким образом, стандарт CWM состоит из ряда составных метамоделей (субметамоделей), которые организованы в виде следующих четырех слоев: базовый слой (Foundation), источники данных (Resources), анализ (Analysis) и управление хранилищем (Management).

Базовый слой состоит из метамоделей, которые поддерживают моделирование таких различных элементов и сервисов, как типы данных, системное преобразование типов, абстрактные ключи и индексы, выражения, бизнес-информация и включения программного обеспечения, основанного на использовании компонентных объектов.

Слой источников данных предоставляет возможность моделировать существующие и новые источники данных, в том числе реляционные базы данных, ориентированные на запись базы данных, а также XML- и основанные на объектах источники данных.

Слой анализа предоставляет средства для моделирования сервисов информационного анализа, которые обычно используются в хранилище данных. Он определяет метамодель для преобразования данных, OLAP, визуализации информации и исследования данных (data mining).

Слой управления состоит из метамоделей, представляющих стандартные процессы и операции ХД, журнализации и планирования работ (например, ежедневной загрузки и выгрузки).

Набор метамоделей CWMI является достаточным для моделирования всего ХД.

Выбор метамодели при проектировании хранилища данных

Когда требования к метаданным собраны и формализованы, можно приступать к разработке метамодели. На практике следовать требованиям стандарта часто бывает сложно. Причиной этого является дефицит времени и недопонимание важности проработки метамодели руководством компании, особенно когда компания создает ХД силами своего ИТ-подразделения. Ру-

ководство по проектированию и разработке метамодели CWMI насчитывает более 700 страниц. Как правило, менеджменту ИТ-подразделения трудно объяснить руководству компании, что для собственной разработки ХД необходимо либо взять новую штатную единицу, либо отправить своего специалиста на обучение.

Заказывает ли компания разработку ХД третьей компании или собирается проводить ее самостоятельно, можно выделить следующие способы создания метамодели:

- построение метамодели ХД вручную;
- построение метамодели ХД на основе стандартов;
- построение как самой метамодели, так и репозитория метамодели на основе имеющихся инструментальных средств для интеграции метаданных источников.

Чтобы построить метамодель ХД вручную, необходимо собрать правильные определения сущностей, их атрибутов и взаимосвязей между сущностями. Для разработки такой метамодели может быть использовано либо объектно-ориентированное моделирование, либо ER-моделирование.

Если для построения метамодели ХД проектировщик ориентируется на использование стандарта, то у него есть возможность либо использовать спецификацию "Открытая информационная модель" (Open Information Model, OIM), либо спецификацию "Общая метамодель хранилища данных" (Common Warehouse Meta-Model, CWM). CWM описывает обмен метаданными в системах складирования данных, управления знаниями и деловой осведомленности. OIM является набором спецификаций метаданных для использования в разработке приложений ХД. Обе спецификации основываются на промышленных стандартах, таких, как UML, XML и SQL.

Выбор подхода к проектированию метаданных во многом определяется набором инструментальных средств проектировщика ХД и выбором несущей СУБД. Ясно, что разработанная вручную метамодель имеет важное преимущество: она, как правило, наиболее полно отражает представление метаданных

компании. Но у такой модели есть большой недостаток - ее нужно сопровождать и поддерживать постоянно в актуальном состоянии, как правило, вручную. Модели, разработанные с учетом стандартов, учитывают большинство требований по представлению метаданных компании в ХД. Кроме того, они расширяемы и поддерживаются ведущими производителями средств разработки (Oracle, IBM, Microsoft).

Репозиторий метаданных ХД следует поддерживать при использовании любого метода проектирования метаданных. При этом важно выбрать для него архитектуру (централизованную или распределенную) и способы поддержки его в актуальном состоянии (поскольку метаданные связывают между собой семантику всех компонентов системы складирования данных).

Программные компоненты системы складирования данных через репозиторий обмениваются метаданными в процессе своей работы. Для организации обмена метаданными стандарт CWMI позволяет детализировать архитектуру репозитория. При этом формат обмена метаданными есть XML-документ.

2.4 Очистка данных

Одной из важных задач, решаемых при переносе данных в ХД, является их очистка. С одной стороны, данные загружаются постоянно из различных источников, поэтому вероятность попадания "грязных данных" весьма высока. С другой стороны, ХД используются для принятия решений, и "грязные данные" могут стать причиной' принятия неверных решений. Таким образом, процедура очистки является обязательной при переносе данных из ОИД в ХД. Ввиду большого спектра возможных несоответствий в данных их очистка считается одной из самых крупных проблем в технологии ХД. Основные проблемы очистки данных можно классифицировать по следующим уровням: уровень ячейки таблицы; уровень записи; уровень таблицы БД; уровень одиночной БД; уровень множества БД.

Рассмотрим перечисленные уровни и соответствующие им проблемы более подробно.

Уровень ячейки таблицы. На данном уровне задача очистки заключается в анализе и исправлении ошибок в данных, хранящихся в ячейках таблиц БД. К таким ошибкам можно отнести следующие:

1. Орфографические ошибки (опечатки) — возникают при вводе информации. Они могут привести к неправильному пониманию, а также к искажению реальных данных. Например, при продаже товара вместо количества 1000 было введено 10 000 или вместо названия товара "Водка" было введено название "Вода".

2. Отсутствие данных — такие ошибки происходят из-за отсутствия у оператора соответствующих данных при вводе информации. Главной задачей OLTP-систем является обеспечение ежедневных операций с данными, поэтому оператор может пропустить ввод неизвестных ему данных, а не тратить время на их выяснение. Как следствие, в БД могут оставаться незаполненные ячейки (содержащие значение NULL).

3. Фиктивные значения — это значения, введенные оператором, но не имеющие смысла. Наиболее часто такая проблема встречается в полях, обязательных для заполнения, но при отсутствии у оператора реальных данных он вынужден вводить бессмысленные данные, например: номер социального страхования 999-99-9999, или возраст клиента 999, или почтовый индекс 99999. Проблема усугубляется, если существует вероятность появления реальных данных, которые могут быть приняты за фиктивные, например, номер социального страхования 888-88-8888 для указания на статус клиента-иностранца "нерезидент" или месячный доход в размере $99 999,99 для указания на то, что клиент имеет работу.

4. Логически неверные значения — значения, не соответствующие логическому смыслу, вкладываемому в данное поле таблицы. Например, в поле "Город" находится значение "Россия", или в поле "температура больного" значение 10.

5. Закодированные значения — сокращенная запись или кодировка реальных данных, используемая для уменьшения занимаемого места.

6. Составные значения — значения, содержащие несколько логических данных в одной ячейке таблицы. Такая ситуация возможна в полях произвольного формата (например, строковых или текстовых). Проблема усугубляется, если отсутствует строгий формат записи информации в такие поля.

Уровень записи. На данном уровне возникает проблема противоречивости значений в разных полях записи, описывающей один и тот же объект предметной области, например, когда возраст человека не соответствует его году рождения: age = 22, bdate = 12.02.50.

Уровень таблицы БД. На данном уровне возникают проблемы, связанные с несоответствием информации, хранящейся в таблице и относящейся к разным объектам. На этом уровне наиболее часто встречаются следующие проблемы.

Нарушение уникальности. Значения, соответствующие уникальным атрибутам разных объектов предметной области, являются одинаковыми.

Отсутствие стандартов. Из-за отсутствия стандартов на формат записи значений могут возникать проблемы, связанные с дублированием данных или с их противоречивостью:

• дублирующиеся записи (один и тот же человек записан в таблицу два раза, хотя значения полей уникальны):

emp1 = (name = «Петр Иванов»); emp2 = (name = «П. Иванов»);

• противоречивые записи (об одном человеке в разных случаях введена разная информация о дате рождения, хотя значения полей уникальны):

emp1 = (name = «И. Сидоров», bdate = 12.02.90);

emp2 = (name = «Иван Сидоров», bdate = 12.12.90);

Уровень одиночной БД. На данном уровне, как правило, возникают проблемы, связанные с нарушением целостности данных.

Уровень множества БД. На этом уровне возникают проблемы, связанные с неоднородностью как структур БД, так и хранящейся в них информации. Можно выделить следующие основные проблемы этого уровня:

различие структур БД (различие наименований полей, типов, размеров и др.);

в разных БД существуют одинаковые наименования разных атрибутов;

в разных БД одинаковые данные представлены по-разному;

в разных БД классификация элементов разная;

в разных БД временная градация разная;

в разных БД ключевые значения, идентифицирующие один и тот же объект предметной области, разные и т. п.

При решении задачи очистки данных, прежде всего, необходимо отдавать себе отчет в том, что не все проблемы могут быть устранены. Возможны ситуации, когда данные не существуют и не могут быть восстановлены, вне зависимости от количества приложенных усилий. Встречаются ситуации, когда значения настолько запутаны или найдены в стольких несопоставимых местах с такими на вид различными и противоположными значениями одного и того же факта, что любая попытка расшифровать эти данные может породить еще более неверные результаты, и, возможно, лучшим решением будет отказаться от их обработки. На самом деле не все данные нужно очищать, процесс очистки требует больших затрат, поэтому те данные, достоверность которых не влияет на процесс принятия решений, могут оставаться неочищенными.

Процесс очистки данных включает в себя несколько этапов:

- выявление проблем в данных;
- определение правил очистки данных;
- тестирование правил очистки данных;
- непосредственная очистка данных.

Выявление проблем в данных. Для выявления подлежащих удалению видов ошибок и несоответствий необходим подробный анализ данных. Наряду

с ручной проверкой следует использовать аналитические программы. Существует два взаимосвязанных метода анализа: профайлинг данных и Data Mining.

Профайлинг данных ориентирован на грубый анализ отдельных атрибутов данных. При этом происходит получение, например, такой информации, как тип, длина, спектр значений, дискретные значения данных и их частота, изменение, уникальность, наличие неопределенных значений, типичных строковых моделей (например, для номеров телефонов) и др., что позволяет обеспечить точное представление различных аспектов качества атрибута.

Data Mining помогает найти специфические модели в больших наборах данных, например отношения между несколькими атрибутами. Именно на это направлены так называемые описательные модели Data Mining, включая группировку, обобщение, поиск ассоциаций и последовательностей. При этом могут быть получены ограничения целостности в атрибутах, например, функциональные зависимости или характерные для конкретных приложений бизнес-правила, которые можно использовать для восполнения утраченных и исправления недопустимых значений, а также для выявления дубликатов записей в источниках данных.

Определение правил очистки данных. В зависимости от числа источников данных, степени их неоднородности и загрязненности, они могут требовать достаточно обширного преобразования и очистки. Первые шаги по очистке данных могут скорректировать проблемы отдельных источников данных и подготовить данные для интеграции. Дальнейшие шаги должны быть направлены на интеграцию данных и устранение проблем множественных источников.

На этом этапе необходимо выработать общие правила преобразования, часть из которых должна быть представлена в виде программных средств очистки.

Тестирование правил очистки данных. Корректность и эффективность правил очистки данных должны тестироваться и оцениваться, например, на

копиях данных источника. Это необходимо для выяснения целесообразности корректировки правил с целью их улучшения или исправления ошибок.

Этапы определения правил и их тестирование могут выполняться итерационно несколько раз, например, из-за того, что некоторые ошибки становятся заметны только после определенных преобразований.

Непосредственная очистка данных. На этом этапе выполняются преобразования в соответствии с определенными ранее правилами. Очистка выполняется в два приема. Сначала устраняются проблемы, связанные с отдельными источниками данных, а затем — проблемы множества БД.

Над отдельными ОИД выполняются следующие процедуры.

Расщепление атрибутов. Данная процедура извлекает значения из атрибутов свободного формата для повышения точности представления и поддержки последующих этапов очистки, таких как сопоставление элементов данных и исключение дубликатов. Необходимые на этом этапе преобразования перераспределяют значения в поле для получения возможности перемещения слов и извлекают значения для расщепленных атрибутов.

Проверка допустимости и исправления. Эта процедура исследует каждый элемент данных источника на наличие ошибок. Обнаруженные ошибки автоматически исправляются (если это возможно). Проверка на наличие орфографических ошибок выполняется на основе просмотра словаря. Словари географических наименований и почтовых индексов помогают корректировать адресные данные. Атрибутивные зависимости (дата рождения — возраст, общая стоимость — цена за шт., город — региональный телефонный код и т. д.) могут использоваться для выявления проблем и замены утраченных или исправления неверных значений

Стандартизация. Эта процедура преобразует данные в согласованный и унифицированный формат, что необходимо для их дальнейшего согласования и интеграции. Например, записи о дате и времени должны быть оформлены в специальном формате, имена и другие символьные данные должны конвертироваться либо в прописные, либо в строчные буквы и т. д. Текстовые

данные могут быть сжаты и унифицированы с помощью выявления основы (шаблона), удаления префиксов, суффиксов и вводных слов. Более того, аббревиатуры и зашифрованные схемы подлежат согласованной расшифровке с помощью специального словаря синонимов или применения предопределенных правил конверсии.

Противоток данных. После того как ошибки отдельных источников удалены, очищенные данные должны заменить загрязненные данные в исходных ОИД. Это необходимо для повышения качества данных в ОИД и исключения затрат на очистку при повторном использовании. После завершения преобразований над данными из отдельных источников можно приступать к их интеграции. При этом выполняются следующие процедуры.

Сопоставление данных, относящихся к одному элементу. Эта процедура устраняет противоречивость и дублирование данных из разных источников, относящихся к одному объекту предметной области. Для сопоставления записей из разных источников используются идентификационные атрибуты или комбинация атрибутов. Такими атрибутами могут выступать общие первичные ключи или другие общие уникальные атрибуты. К сожалению, без таких атрибутов процесс сопоставления данных затруднителен.

Слияние записей. Данная процедура объединяет интегрированные записи, относящиеся к одному объекту. Объединение выполняется, если информация из разных записей дополняет или корректирует одна другую

Исключение дубликатов. Данная процедура удаляет дублирующие записи. Она производится либо над двумя очищенными источниками одновременно, либо над отдельным, уже интегрированным набором данных. Исключение дубликатов требует, в первую очередь, выявления (сопоставления) похожих записей, относящихся к одному и тому же объекту реального окружения

Очищенные данные сохраняются в ХД и могут использоваться для анализа и принятия на их основе решений. За формирование аналитических запросов к данным и представление результатов их выполнения в СППР отвечают под-

системы анализа. От вида анализа также зависит и непосредственная реализация структур хранения данных в ХД.

2.5 Концепция хранилища данных и анализ

Концепция ХД не является законченным архитектурным решением СППР и тем более не является готовым программным продуктом. Цель концепции ХД — определить требования к данным, помещаемым в ХД, общие принципы и этапы построения ХД, основные источники данных, дать рекомендации по решению потенциальных проблем, возникающих при выгрузке, очистке, согласовании, транспортировке и загрузке данных.

Необходимо понимать, что концепция ХД:

- это не концепция анализа данных, скорее, это концепция подготовки данных для анализа;
- не предопределяет архитектуру целевой аналитической системы. Концепция ХД указывает на то, какие процессы должны выполняться в системе, но не где конкретно и как они будут выполняться.

Таким образом, концепция ХД определяет лишь самые общие принципы построения аналитической системы и в первую очередь сконцентрирована на свойствах и требованиях к данным, но не на способах организации и представления данных в целевой БД и режимах их использования. Концепция ХД описывает построение аналитической системы, но не определяет характер ее использования. Она не решает ни одну из следующих проблем:

выбор наиболее эффективного для анализа способа организации данных;

организация доступа к данным;

использование технологии анализа.

Проблемы использования собранных данных решают подсистемы анализа. Как отмечалось ранее, такие подсистемы используют следующие технологии:

- регламентированные запросы;
- оперативный анализ данных;

- интеллектуальный анализ данных.

Если регламентированные запросы успешно применялись еще задолго до появления концепции ХД, то оперативный и интеллектуальный анализы в последнее время все больше связывают с ХД.

Контрольные вопросы

1. Какая идея лежит в основе концепции ХД?
2. Приведите данное Инмоном определение хранилища данных.
3. Приведите основные преимущества виртуального хранилища данных.
4. Охарактеризуйте недостатки физического хранилища данных.
5. В чем заключаются основных проблемах создания классического (физического) ХД?
6. Для чего нужны метаданные пользователям СППР?
7. Что такое «витрина данных»?
8. На какие три основные категории делятся данные в ХД?
9. Охарактеризуйте основные этапы ETL процесса.
10. Опишите основные этапы процесса очистки данных.

3. Архитектура хранилищ данных

3.1 Факторы, определяющие архитектуру ХД

Одной из главных целей разработки ХД является информационное обеспечение компьютерной поддержки принятия решений по всем или основным видам деятельности организации**.** Каждый вид деятельности организации является отдельной задачей, решение которой может быть, а может и не быть увязано с решением других задач в рамках организации. Вид деятельности организации или направление бизнеса совместно со спектром соответствующих ему бизнес-задач определяют предметную область ХД.

Например, компания производит и продает оборудование для добычи угля, а с другой стороны, та же компания имеет подразделения, которые занимаются производством услуг в области автоматизации предприятий, в том числе и угледобывающих. Источники прибыли в этих случаях различны. Это два направления бизнеса компании (две предметных области). Общими задачами анализа данных для этих направлений бизнеса являются прибыль и бюджет.

ХД — это сложная компьютерная система. Под **архитектурой ХД** понимают совокупность программно-аппаратных компонент, совокупность технологических и организационных решений, предпринимаемых для создания, разработки и функционирования ХД, т.е. выбор аппаратного и программного обеспечения, выбор способов взаимодействия программно-аппаратных компонент, выбор способа решения проектной задачи по разработке и созданию ХД.

Из материалов предыдущего раздела следует, что, как правило, архитектуру ХД составляют следующие компоненты [3]:

• средства извлечения данных из различных БД OLTP-систем, унаследованных систем и других внешних источников данных;

• средства трансформации и очистки данных. Точность существующих данных доставляет немало хлопот организации. Поэтому перед тем как

поместить данные в хранилище их необходимо привести в порядок, иначе говоря — очистить;

• программное обеспечение БД. Как правило, это высокопроизводительная РСУБД, используемая для структуризации и хранения информации;

• средства для соединения источников данных с хранилищем и клиентов с сервером.

Кроме этого, необходимы специальные программные средства проектирования хранилища, средства работы с репозиторием *метаданных* и собственно средства оперативной аналитики, или OLAP-средства.

Все это — сложное специальное программное обеспечение, стоимость которого также может исчисляться десятками и сотнями тысяч долларов.

Характер и масштаб решаемых задач анализа данных организации оказывает решающее значение на выбор архитектуры ХД и методы его проектирования.

Проектировщик должен помнить, что, с одной стороны, ХД создается для решения конкретных, строго определенных задач анализа и воспроизводства новых данных, с другой — ХД должно обеспечивать корпоративную отчетность в рамках всей организации. Таким образом, определяющим моментом в построении ХД являются задачи обработки и анализа данных, производства и доставки отчетов.

Характер и масштаб решаемых задач анализа данных определяет и подходы к выбору архитектуры и проектированию ХД.

Желательно, чтобы выбор архитектуры ХД был сделан до начала его реализации, однако на практике не всегда следуют этому правилу. Задержка с выбором архитектуры ХД обычно приводит к пересмотру проделанной работы в свете новых принятых решений и, как правило, к увеличению объема работы.

Выбор архитектуры ХД относится к сфере компетенции руководителя ИТ-проекта по созданию системы поддержки принятия решений. На такой выбор влияют несколько различных факторов: инфраструктура организации, производственная и информационная среда организации, управление и

контроль, масштабы проекта, возможности аппаратно-технологического обеспечения, готовность персонала и имеющиеся ресурсы.

Выбор подхода к конкретной реализации ХД также лежит в области влияния руководителя ИТ-проекта. Правильный выбор архитектуры ХД обычно определяет успех конкретного проекта по созданию системы поддержки принятия решений.

Существует несколько факторов, влияющих на принятие решений о выборе способа реализации: ***время,*** отведенное на проект, ***возврат инвестиций, скорость ввода ХД в эксплуатацию, потребности пользователей, потенциальные угрозы по переделке, требования к ресурсам,*** необходимым в определенный момент времени, выбранная архитектура ХД, ***совокупная стоимость владения*** ХД.

Проектировщик ХД должен знать, какие возможные решения могут быть приняты по архитектуре ХД и какой объем работ по проектированию ХД они повлекут. Выбор архитектуры будет определять, где ХД и/или киоски данных будут расположены и как ими будут организационно-технологически управлять. Например, данные могут быть расположены в центральном офисе организации, т.е. будут поддерживаться централизованно. Данные могут быть распределены по офисам организации или располагаться в филиалах организации, и могут поддерживаться как централизованно, так и независимо друг от друга.

Далее приводится краткий обзор типовых архитектур систем с хранилищем данных и программных продуктов, наиболее часто используемых для реализации таких систем.

3.2 Основные типы программно-аппаратной архитектуры хранилища данных

Типовыми архитектурами для систем складирования данных принято считать следующие:

- системы с глобальным ХД;
- системы с независимыми киосками данных;
- системы с интегрированными киосками данных;
- системы, разработанные на основе комбинации из вышеперечисленных архитектур.

Глобальное хранилище данных (Global data warehouse), или хранилище данных масштаба организации, — это такое ХД, в котором будут поддерживаться все данные организации или большая их часть**.** Это наиболее полное интегрированное ХД с высокой степенью интенсивности доступа к консолидированным данным и использованием его всеми подразделениями организации или руководством организации в рамках основных направлений деятельности организации. Таким образом, глобальное ХД проектируется и конструируется на основе потребностей аналитической информационной поддержки организации в целом. Его можно рассматривать как общий репозиторий для данных, обеспечивающих принятие решений.

Глобальное ХД необязательно должно быть реализовано физически как централизованное. Термин «глобальное» используется для отражения масштаба использования и доступа к данным в рамках всей организации. Глобальное ХД может быть физически как централизованным, так и распределенным.

Централизованное глобальное ХД характерно для организаций, расположенных территориально в одном здании. Оно поддерживается отделом информационных систем организации. *Распределенное глобальное ХД* также может быть использовано в рамках организации в целом. Оно физически распределяется по подразделениям организации и также поддерживается отделом информационных систем.

Поддержка ХД отделом информационных систем вовсе не означает, что именно эта служба управляет ХД. Например, отдельные части распределенного ХД могут управляться в рамках подразделений или направлений бизнеса.

Управление ХД определяет, кто решает:

• какие данные должны поступать в ХД;

• когда данные должны поступать в ХД;

• когда данные должны обновляться;

• кому разрешен доступ к данным в ХД.

Таким образом, для глобального ХД существуют два основных архитектурных решения.

Данные для ХД обычно извлекаются из OLTP-систем организации, электронных документов организации и внешних источников данных. После фильтрации, очистки и преобразования они помещаются в ХД. Затем пользователи получают доступ к этим данным в соответствии с правилами управления доступом к данным, принятыми в организации.

Преимуществом глобального ХД является предоставление конечным пользователям доступа к информации в масштабах предприятия, недостатком — высокие затраты на реализацию, в том числе затраты времени на создание ХД.

Независимые киоски данных включают в себя автономные или независимые киоски данных (Stand-alone Data Marts), которые управляются рабочими группами, отделами или направлениями бизнеса и разрабатываются исключительно для реализации аналитических потребностей последних. Вполне возможно, что при этом не существует никакой связи между ними. Например, данные для таких киосков данных могут генерироваться непосредственно в самих подразделениях организации. Данные могут извлекаться из OLTP-систем, в частности, при помощи информационных служб организации. Информационные службы могут поддерживать вычислительную среду для киосков данных, но не управляют информацией в них. Данные в киоски могут поступать и из глобального ХД [3].

Для организации независимых киосков данных требуются некоторые профессиональные и технические навыки. Как правило, для их создания выделяются ресурсы и персонал в рамках того подразделения, для которого они создаются. Такой тип реализации ХД оказывает минимальное влияние на

информационные ресурсы организации и может быть выполнен очень быстро. В то же время максимальная независимость и минимальная интеграция, а также отсутствие глобального представления о данных организации могут стать ограничением такой архитектуры.

Киоски данных могут быть взаимозависимы или взаимосвязаны (так называемые *связанные киоски данных*)**.** Такая архитектура ХД включает в себя совокупность киосков данных, которые управляются рабочими группами, отделами или направлениями бизнеса, но разрабатываются в рамках единой для организации схемы удовлетворения информационных и аналитических потребностей**.** Для взаимосвязанных киосков данных типична распределенная архитектура реализации. Несмотря на то, что отдельные киоски данных реализуются в рамках рабочих групп, подразделений и направлений бизнеса, они могут быть интегрированы, т.е. взаимосвязаны, для того чтобы обеспечить представления данных в рамках организации в целом. Фактически, на наиболее высоком уровне интеграции, они могут стать глобальным ХД. В такой архитектуре пользователи одних подразделений могут получать доступ к данным других подразделений в рамках своих полномочий.

Требования интеграции данных в рамках архитектуры взаимосвязанных киосков данных делают реализацию ХД более сложной по сравнению с независимыми киосками данных. Например, необходимо решить вопрос, кто будет управлять данными в киосках данных и кто будет поддерживать вычислительную среду. Важным становится вопрос о том, что делать с данными, которые являются общими для нескольких киосков данных, а также как разработать схему разграничения доступа пользователей к киоскам данных в рамках всей организации.

Главным достоинством создания ХД такой архитектуры является более глобальное представление данных. Взаимосвязанные киоски данных могут управляться в рамках того подразделения, в котором они создаются.

Реализация такой архитектуры не выдвигает высоких требований к программно-аппаратному обеспечению, и стоимость ее может быть невысокой. Однако время реализации будет больше по сравнению с независимыми киосками данных. Возрастают также сложность и стоимость процедур проектирования.

В заключение раздела следует отметить, что развитие программно-вычислительных средств позволяет создавать так называемые виртуальные ХД, которые работают над OLTP-системами, ХД с многоуровневой архитектурой и так называемые встроенные ХД, которые встраиваются в существующую систему обработки данных организации.

3.3 Организация работ по созданию хранилища данных

Так же, как и для реализации любых типов информационных систем с базами данных, к созданию ХД применимы следующие основные методологические подходы:

- «сверху вниз» (Top down design);
- «снизу вверх» (Bottom down design);
- «из середины» (Middle of design).

На выбор подхода к реализации ХД оказывают влияние следующие факторы:

- состояние текущей информационной инфраструктуры организации;
- имеющиеся в наличии ресурсы;
- требования по возврату инвестиций;
- потребности организации в интегрированном представлении данных о своей деятельности;
- скорость реализации.

Выбор методологического подхода к реализации ХД влияет на объем и тщательность проектирования.

Подход «сверху вниз». Подход «сверху вниз» требует детального планирования и проектирования ХД в рамках ИТ-проекта до начала

выполнения проекта. Это связано с тем, что необходимо привлекать всех потенциальных пользователей ХД для выяснения их информационных потребностей в аналитической обработке данных, принимать решения об источниках данных, безопасности, структурах данных, качестве данных, стандартах данных. Все эти работы должны быть документированы и согласованы. При этом подходе модель ХД должна быть разработана до начала реализации.

Обычно такой подход практикуют при создании глобального ХД. Если киоски данных включаются в конфигурацию, то они могут быть построены позже.

Достоинством такого подхода является получение более согласованных определений данных и бизнес-правил организации в самом начале работы над созданием ХД. Стоимость начального планирования и проектирования может оказаться достаточно высокой. Для этого подхода характерны большие затраты времени, что откладывает начало реализации и задерживает возврат инвестиций. Подход «сверху вниз» хорошо применять в организациях с четко организованной информационно-вычислительной структурой, когда программно-аппаратная платформа определена и существуют слаженно работающие источники данных.

Подход «снизу вверх». При использовании подхода «снизу вверх» начинают с планирования и проектирования киосков данных подразделений без предварительной разработки глобальной информационно-вычислительной инфраструктуры организации. Это не означает, что такая глобальная инфраструктура не будет разработана позже. Такой подход является более приемлемым во многих случаях, поскольку он быстрее приводит к конечным результатам. У него есть и недостатки: данные могут дублироваться и быть несогласованными в разных киосках данных. Чтобы избежать этого, необходимо тщательное планирование и проектирование.

Подход «проектирование из середины». Подходы «снизу вверх» и «сверху вниз» могут комбинироваться в зависимости от поставленных перед

руководителем проекта по созданию ХД целей. Подход «проектирование из середины» представляет собой комбинацию вышеперечисленных подходов, которые применяются как бы по спирали. Сначала создается ядро системы (подход «сверху вниз»), а затем оно поэтапно наращивается за счет добавления новой или дополнительной функциональности (подход «снизу вверх»). Таким образом, на каждом витке спирали может быть использован каждый из двух указанных выше подходов.

Существуют и другие комбинации. Выбор подхода к реализации ХД наряду с выбором архитектуры ХД определяет тактические решения в проектировании и управлении проектом создания системы складирования данных. К таким решениям относятся планирование реализацией и управление проектом [3].

3.4 Характеристика решений ведущих производителей

В настоящем разделе приведем краткий обзор базовых решений основных производителей программного обеспечения для разработки ХД.

IBM. Решение компании IBM называется Data Warehouse Plus. Це¬лью компании в области разработки и поддержки систем складирования данных является обеспечение пользователя интегрированным набором программных продуктов и сервисов в рамках единой архитектуры.

IBM предлагает встроенную поддержку трех типов архитектурных решений для ХД:

- независимый киоск данных;
- взаимосвязанные киоски данных;
- глобальное ХД.

Несущая СУБД для ХД — семейство объектно-реляционных СУБД DB2. Язык манипулирования данными — SQL.

Преимущество решений IBM проявляется, когда и системы оперативной обработки данных, и ХД находятся на программном обеспечении IBM, т.е. предлагается так называемое замкнутое типовое решение.

С приобретением компании Informix Software IBM взяла под свое крыло ряд удачных решений этой компании в области систем поддержки принятия решений на основе хранилищ данных.

Oracle. Решения, предлагаемые компанией, преследуют две основные цели: предоставление пользователям широкого ассортимента программных продуктов самой компании и деятельность партнеров в рамках программы Warehouse Technology Initiative.

Компания Oracle не предлагает поддержку каких-либо встроенных архитектурных решений для ХД.

Несущая СУБД для ХД — семейство объектно-реляционных СУБД Oracle 11g/10g. Язык манипулирования данными — SQL. Начиная с версии 8i, диалект SQL существенно дополнен набором функций для аналитической обработки данных, вплоть до построения линейной регрессии.

Компания выпускает специальный CASE-инструментарий для проектирования ХД.

Конкурентные возможности Oracle определяются следующими факторами:

• имеется набор готовых приложений для разработки ХД, обеспечивающий полный жизненный цикл;

• компания является одним из лидеров по продажам в области анализа данных;

• совместимость с продуктами, производимыми другими компаниями.

NCR. Решение этой компании в области складирования данных ориентировано на организации, у которых имеются потребности в системах DSS (система поддержки и принятия решений) и системах OLAP. Предлагаемая архитектура называется Enterprise Information Factory (виртуальное предприятие).

Несущая СУБД для ХД — реляционная СУБД Teradata.

Конкурентным преимуществом решений компании является большой опыт применения СУБД Teradata и связанных с ней методов параллельной обработки данных.

SAS Institute. Компания считает себя поставщиком полного решения для организации ХД. Компания предлагает методологию Rapid Data Warehousing для быстрого создания и наполнения ХД. В основу этой методологии положено:

• обеспечение доступа к данным в ХД с возможностью их извлечения из разнообразных источников данных (интероперабельность);

• преобразование и манипулирование данными в рамках 4GL (Data Step);

• наличие у компании сервера многомерных БД;

• большой набор программных продуктов компании для аналитической обработки данных и статистического анализа.

Конкурентным преимуществом компании является наличие у нее длинной линейки программных продуктов для статистического и сравнительного анализа данных, который интегрирован в ее методологию построения и использования ХД.

Sybase. Стратегия компании в области ХД основывается на разработанной архитектуре Warehouse WORKS.

Несущая СУБД для ХД — реляционная СУБД Sybase System 11, средство подключения к базам данных OmniCONNECT. Язык манипулирования данными — SQL и средства быстрой разработки приложений.

Компания выпускает специальный CASE-инструментарий для проектирования ХД.

Конкурентным преимуществом компании является наличие набора программных продуктов для обеспечения полного жизненного цикла разработки ХД.

Microsoft. Компания сравнительно недавно стала активно предлагать комплексные решения в области ХД. Целью корпорации Microsoft является создание инструментальной и технологической среды, которая позволила бы минимизировать затраты на создание ХД и сделала бы этот процесс доступным для массового пользователя. Акцент предлагаемых компанией решений в области складирования данных концентрируется на развитии инструментальных средств OLAP.

Корпорация предлагает спецификации среды Microsoft Data Warehousing Framework для создания и использования ХД. Открытость среды Microsoft Data Warehousing Framework обеспечила ее поддержку многими производителями программного обеспечения.

Цель Microsoft Data Warehousing Framework состоит в том, чтобы упростить разработку, внедрение и администрирование решений на основе ХД. Эта спецификация призвана обеспечить:

• открытую архитектуру, которая интегрируется и расширяется третьими фирмами;

• экспорт и импорт гетерогенных данных наряду с их проверкой, очисткой и ведением истории накопления;

• доступ к разделяемым метаданным со стороны процессов разработки ХД.

Несущая СУБД для ХД — реляционная СУБД MS SQL Server 2005/2008. Язык манипулирования данными — SQL со встроенными средствами обработки многомерных кубов.

Конкурентным преимуществом компании является наличие у нее набора программных продуктов для обеспечения разработки и поддержки ХД, в том числе для очистки данных, при невысокой цене на эти продукты. Ориентация продукции компании на средний и малый бизнес позволяет ей увеличить свои конкурентные преимущества.

3.5 Типовые программно-аппаратные решения реализации ХД

Общие типовые решения. Из предыдущих разделов пособия следует, что существуют несколько вариантов реализации ХД в рамках типовой архитектуры. Рассмотрим особенности технологических решений некоторых из них.

*Виртуальное хранилище данных***.** Архитектура обеспечивает доступ к «живым» данным в режиме реального времени через программное обеспечение промежуточного слоя. В основе такого решения лежит репозиторий метаданных, который описывает источники данных, процедуры их предварительной обработки и форматы представления информации конечному пользователю. Недостатки такого решения — интенсивный сетевой трафик, снижение производительности несущей системы, угроза нарушения целостности данных в случае неудачных действий пользователей ХД.

*Киоски данных***.** Архитектура представляет собой облегченный вариант ХД тематической направленности. Бывают киоски данных, связанные с интегрированным ХД или несвязанные (автономные).

*Глобальное хранилище данных***.** Архитектура представляет собой единый источник интегрированных данных организации.

*Хранилища данных с многоуровневой (в основном трехзвенной) архитектурой, или корпоративные ХД***.** Архитектура является разновидностью глобального ХД, в которую технологически реализуются три уровня. На первом уровне располагается корпоративное ХД организации. На втором уровне поддерживаются связанные киоски данных тематической направленности на основе многомерной СУБД. На третьем уровне находятся клиентские приложения пользователей с установленными на них средствами анализа данных.

Встроенные (комбинированные) хранилища данных. Архитектура представляет собой ХД, которые органически встраиваются в виртуальное предприятие (Enterprise Information Factory, EIF) или используются как компонент аналитической поддержки в информационной реализации бизнес-функций.

Корпоративная информационная фабрика (Corporate Information Factory, CIF). Эта архитектура является развитием архитектуры корпоративного ХД (enterprise data warehouse, EDW). Ее использование предполагает скоординированное извлечение данных из источников, загрузку их в реляционную БД со структурой в третьей нормальной форме, использование построенного ХД для наполнения дополнительных репозиториев презентационных данных.

Хранилище данных с архитектурой шины данных (Data Warehouse Bus). В этой архитектуре ХД не является единым физическим репозиторием (в отличие от CIF). Это «виртуальное» ХД, представляющее коллекцию витрин данных, каждая из которых имеет архитектуру типа «звезда».

*Объединенное (федеративное) ХД***.** В этой архитектуре ХД состоит из ряда экземпляров ХД, которые функционируют на полуавтономной основе и, как правило, организационно или географически разнесены, однако могут рассматриваться и управляться как одно большое ХД.

Существенные различия в программном обеспечении у различных производителей определяются следующими факторами: 1) используемая модель данных; 2) степень охвата жизненного цикла; 3) встроенная поддержка различных архитектур; 3) возможности языка обработки данных. Можно обратить внимание на следующие две основные тенденции.

1. Производители предлагают комплексные решения по созданию хранилищ данных. Ведущие производители программного обеспечения в области проектирования и разработки информационных систем с базами данных стараются иметь свои собственные программы по системам складирования данных и обеспечивать полный жизненный цикл разработки и сопровождения таких систем.

2. Производители начинают предлагать готовые встроенные архитектурные решения для хранилищ данных. Это обстоятельство позволяет значительно сокращать время на проектирование и разработку ХД.

С точки зрения применения программно-аппаратных платформ решения в области создания СППР на основе хранилищ данных можно условно разбить на три класса.

1. Комбинация готовых продуктов (решений) разных фирм без непосредственного программирования.

2. Использование полной замкнутой цепочки продуктов (решений) одной фирмы-поставщика.

3. Использование контура продуктов (решений) одной фирмы поставщика с дополнением до замкнутой цепочки совместимыми продуктами третьих фирм.

Простое масштабируемое решение. Пример простого масштабируемого решения можно предложить, основываясь на использовании Crystal Enterprise и Crystal Reports (фирма Business Objects) как инструментов конечного пользователя. Подробнее о возможностях Crystal Enterprise и Crystal Reports можно прочитать в литературе к курсу настоящих лекций.

ХД реализуется на СУБД Oracle, DB2, MS SQL Server или других, имеющих ODBC-интерфейс или интерфейс прямого доступа с Crystal Enterprise. Обычно применяется классическая архитектура ХД без киосков данных. Для этого решения большое значение имеет тщательное проектирование структуры ХД и запросов. Необходимо разработать и создать приложения для очистки данных (или воспользоваться имеющими у поставщиков средствами).

Преимущества такого решения:

- Сводится к минимуму объем программирования, т.к. все стадии покрываются готовыми коробочными продуктами.
- Сокращается время разработки и создания ХД (за счет исключения трудоемкого процесса написания программ).
- Время разработки типового запроса — от 2-х до 6-ти часов, время разработки типового отчета — 1-2 дня.

- Такое решение хорошо для создания прототипов ХД, поскольку в данном случае отрабатываются практически все необходимые запросы и отчеты.
- Создается прекрасная инструментальная среда для использования нетиповых запросов.
- Такое решение прекрасно подходит и для создания виртуальных ХД.

Недостатки:

- Разработка сложных перекрестных запросов может занять много времени.
- Это решение не подходит для сложной аналитической обработки данных, требующей разработки специальных приложений для анализа.

Замкнутое типовое решение. Замкнутое типовое решение можно предложить на основе использования замкнутой цепочки продуктов одной фирмы-поставщика, например Microsoft, Oracle , SAS или Sybase.

Преимущества:

- Как правило, все бизнес-направления поддерживаются за счет готовых сервисов.
- Время разработки и создания ХД поддается строгому описанию и достаточно точной оценке.
- Такое решение хорошо для создания ХД, которые предполагается использовать в организации длительное время.
- Такие решения подходят для сложной аналитической обработки данных, требующей разработки специальных приложений для анализа.

Недостатки:

- Главным недостатком является высокий уровень затрат на разработку и создание, который при правильной организации проекта окупается.

- Кадровый вопрос: необходимо нанимать высококвалифицированные кадры, умеющие работать с набором продуктов выбранной компании. Как правило, обучение своих сотрудников по всем направлениям работы с ХД малоэффективно, хотя и привлекательно.

3.5 Области применения технологии хранилищ данных

Концепция хранилищ данных находит применение во многих сферах бизнеса, науки и управления. Типовые решения использования технологии хранилищ данных в бизнесе можно разделить на следующие основные группы:

- Разработка основы для создания аналитических подсистем сопровождения бизнеса.
- Разработка ХД как составной части виртуального предприятия.
- Разработка ХД для цифровых (электронных) библиотек и мультимедиа.

Имеется тенденция расширения проникновения концепции в те сферы бизнеса, где необходимо выполнять, с одной стороны, сравнительный анализ, искать зависимости в данных, выявлять тренды в рядах динамики, а с другой — использовать системы складирования данных в связке с системами операционной обработки.

Корпоративные информационные фабрики. В настоящее время в кругу бизнес-пользователей информационных технологий обсуждается предложенная Биллом Инмоном концепция так называемой корпоративной информационной фабрики (Corporate Information Factory, CIF) как одной из основополагающих вычислительных архитектур для производства информационных продуктов предприятия. Для любого предприятия реализацию такой концепции можно рассматривать как важную перспективную задачу, решение которой не только позволит повысить качество управления взаимоотношениями с внешними организациями (налоговыми и финансовыми государственными структурами) и партнерами, но и значительно увеличить производительность его

подразделений, поставляющих информацию, необходимую для принятия стратегических решений.

Рассмотрим более подробно концепцию CIF.

Корпоративная информационная фабрика — это логическая архитектура программно-аппаратного решения по производству, складированию, управлению и доставке данных для поддержки принятия стратегических и тактических решений в масштабе организации**.** Концепция CIF, предложенная классиком в области теории хранилищ данных Биллом Инмоном в серии его работ, подразумевала системно организованное взаимодействие репозиториев оперативных данных (Operational Data Store), центрального ХД, витрин данных и системы интеллектуального анализа данных (Data Mining) за счет создания технологических цепочек переработки и доставки данных.

В абстрактной форме процесс производства информации в CIF был представлен в аналогии с производством некоторого продукта. В соответствии с этим были выделены основные стадии производства информации (новых данных): получение исходных данных (сырья), их преобразование (производство отдельных деталей), складирование данных, создание информационных продуктов (из деталей готовой продукции) и доставка данных их потребителям (распределение конечной продукции).

Основная идея, положенная в основу концепции CIF, состоит в выделении элементов информационной архитектуры на основе их функционального назначения и регламентирования технологических процедур обработки данных.

Краеугольным камнем правильно спроектированной CIF являются, безусловно, метаданные. Задача этого слоя — описать в рамках единой терминологической базы (метаданные бизнес-пользователя) всю совокупность объектов управления средой CIF (метаданные администрирования). Только подход «от метаданных» позволяет из гетерогенного потока входной информации получить однородное описание среды и предметной области, что дает возможность одинаково легко обращаться к измерениям, кубам, отчетам и

бизнес-объектам на основе произвольных выборок. Таким образом, обеспечивается высокое качество циркулирующей в CIF информации.

Структурные компоненты CIF. В основе CIF лежит модель функционального разделения процессов производства новых данных (информационных продуктов) и доставки информационных продуктов их потребителям, а также управления этими процессами.

Производители информационного продукта собирают данные из доступных источников (чаще всего из оперативных систем ввода и обработки данных), преобразуют и интегрируют их, размещая в системе складирования данных в унифицированном регламентированном формате. Потребители информационных продуктов извлекают необходимые тематические выборки из системы складирования данных (через специализированные предварительно настроенные интерфейсы — витрины данных) и затем используют их в процессе принятия решений.

Логическая структура CIF включает в себя несколько типовых архитектурных элементов.

На предприятии производственные и финансовые потоки тесно взаимосвязаны с потоками информационными, которые отражают их динамические показатели и текущее состояние. Кроме того, такие информационные потоки являются источником данных для анализа при определении трендов изменений и их количественных характеристик.

Описанная выше в общих чертах схема превращения данных в информационные продукты и составляет суть концепции CIF на любом предприятии.

Ведение хранилища данных — это технология, с помощью которой можно оперативно собрать данные и на их основе решать разнообразные задачи по финансовому планированию, бюджетированию, риск-менеджменту, анализу взаимоотношений с партнерами, маркетинговому анализу и т.д. Однако самое главное преимущество отлаженной архитектуры CIF в другом: она позволяет адаптировать вычислительную среду как под четко определенные

информационные потоки небольшого предприятия, так и под сложные схемы консолидации, которые характерны для предприятий с развитой филиальной структурой и входящих в состав холдингов и отраслевых объединений предприятий.

Рассмотрим подробнее, как «фабрика управленческих данных» функционирует на предприятии.

ERP/MRP II системы как источники данных для CIF. Первоначальное наполнение корпоративного ХД и постоянное поддержание его в актуальном состоянии — это отнюдь не тривиальные задачи. Особые требования здесь предъявляются к качеству информации, кроме того, высока степень риска — ошибочные решения на основе неверных исходных посылок могут обернуться серьезными потерями.

На предприятиях основными источниками данных являются ERP-системы. Они представляют собой семейство оперативных приложений, обеспечивающих обработку производственных и финансовых данных, включая выполнение бухгалтерских проводок, логистических операций, генерацию текущей оперативной отчетности. Модули ERP ориентированы на те информационные продукты, которые они сопровождают или поддерживают. Разумеется, ERP не предназначены для обработки информации в историческом аспекте и не имеют развитого инструментария для агрегации и систематизации данных предприятия. Из-за строгой предметной направленности у подсистем ERP, как правило, слабо развиты взаимосвязи на уровне данных: обычно у них информационный обмен осуществляется небольшими объемами.

Таким образом, на первом шаге построения CIF-системы источники данных накапливают информацию в масштабе предприятия в «сыром» виде: она не подготовлена для анализа и компиляции аналитической отчетности.

Интеграция и преобразование данных. Организация процесса интеграции является еще одним фактором успеха в создании CIF: информация извлекается из разнородной вычислительной среды ERP, преобразуется с целью повышения

ее качества и складируется. Все это делается для того, чтобы системы поддержки и принятия решений могли в дальнейшем ее активно использовать.

Для наполнения корпоративного ХД в нем обычно предусматриваются инструментальные средства:

- для извлечения и доставки из различных оперативных БД и внешних источников;
- для очистки, преобразования и интеграции;
- для загрузки;
- для актуализации.

В конечном итоге информация должна попасть к потребителю в заданном виде, чтобы послужить базисом для принятия взвешенных управленческих решений. Логично на выходе CIF применять:

• средства для многомерного представления данных и манипулирования ими;

• средства для формирования отчетов;

• систему информационных запросов.

В качестве отличительных характеристик подхода Билла Инмона к архитектуре ХД можно назвать следующие.

1. Использование реляционной модели организации атомарных данных и многомерной модели — для организации суммарных данных.

2. Использование подхода «проектирование из середины» при создании больших ХД, что позволяет создавать ХД поэтапно.

3. Использование третьей нормальной формы для организации атомарных данных, что обеспечивает высокую степень детальности интегрированных данных и, соответственно, предоставляет корпорациям широкие возможности для манипулирования ими и изменения формата и способа представления данных по мере необходимости.

4. ХД — это проект корпоративного масштаба, охватывающий все отделы и обслуживающий нужды всех пользователей корпорации.

5. ХД — это не механическая коллекция витрин данных, а физически целостный объект.

Хранилища данных с архитектурой шины данных. В данной архитектуре ХД с архитектурой шины данных, предложенной Ральфом Кимболлом, первичные данные преобразуются в необходимые структуры на стадии подготовки данных [3]. При этом обязательно принимаются во внимание требования к скорости обработки информации и качеству данных. Подготовка данных начинается со скоординированного извлечения их из источников. Ряд операций совершается централизованно, например, поддержание и хранение общих справочных данных, другие действия могут быть распределенными.

ХД с архитектурой шины данных изначально ориентированы на использование многомерной модели данных. Поэтому, как правило, данные в его структуре денормализованы, чтобы оптимизировать выполнение запросов. Запросы в процессе выполнения обращаются к все более низкому уровню детализации без дополнительного перепрограммирования со стороны пользователей или разработчиков приложения.

В отличие от корпоративной информационной фабрики, в ХД с архитектурой шины данных чаще используются связанные киоски данных, которые разрабатываются для обслуживания бизнес-процессов (бизнес-показателей или бизнес-событий), а не направлений бизнеса. Например, данные о заказах, которые должны быть доступны для общекорпоративного использования, вносятся в ХД только один раз, в отличие от CIF, в котором их пришлось бы трижды копировать в витрины данных отделов маркетинга, продаж и финансов. После того, как в ХД появляется информация об основных бизнес-процессах, консолидированные киоски данных могут выдавать их перекрестные характеристики. Матрица шины данных корпоративного ХД с архитектурой шины выявляет и усиливает связи между показателями бизнес-процессов (фактами) и описательными атрибутами (измерениями).

ХД с архитектурой шины данных состоит из набора взаимосвязанных киосков данных, которые созданы для обслуживания бизнес-процессов организации.

Суммируя все вышесказанное, можно отметить типичные характеристики ХД с архитектурой шины данных.

- Использование многомерной модели организации данных с архитектурой «звезда» (star scheme).
- Использование двухуровневой архитектуры, которая включает стадию подготовки данных, недоступную для конечных пользователей, и собственно ХД с архитектурой шины. В состав последнего входят несколько киосков атомарных данных, несколько киосков агрегированных данных и персональный киоск данных, но оно не содержит одного физически целостного или централизованного ХД.
- ХД не является единым физическим репозиторием (в отличие от CIF). Это «виртуальное» ХД, представляющее коллекцию витрин данных, каждая из которых имеет архитектуру типа «звезда».

Отметим, что и корпоративная информационная фабрика, и ХД с архитектурой шины данных имеют своей целью создание корпоративного ХД. Соответственно, единство конечного объекта означает общность требований, которым должен удовлетворять любой подход для достижения искомого конечного результата, а это, в свою очередь, указывает на то, что и в самой архитектуре должны быть общие черты.

Обе эти архитектуры отличаются в основном способами представления данных. В CIF, они, как правило, нормализованы, а в ХД с архитектурой шины данных — нет.

Объединенное (федеративное) ХД. Для любой организации, особенно многофилиальной, наличие согласованной управленческой информации, необходимой для четкого понимания того, как функционирует бизнес, является одной из актуальных задач.

Обычной подход к улучшению информированности о бизнес-операциях — проведение стандартизации «сверху вниз» как структуры отчетности, так и модели данных. Однако с практической точки зрения стандартизация бизнес-структур оказывается для большинства организаций малоэффективной — требуется слишком много средств и времени.

В качестве одного из подходов для решения указанной проблемы может быть использована архитектура **федеративного ХД**. В этой архитектуре, на основе иерархии связанных ХД, можно обмениваться данными, бизнес-моделями и структурами отчетности, благодаря чему возможно, с одной стороны, осуществлять общий контроль и предусмотреть определенную степень стандартизации, а с другой — позволить региональным отделениям сохранить автономность и учесть местную специфику.

Система объединенных ХД характеризуется совместным использованием общих информационных точек, что устраняет, таким образом, избыточность и гарантирует достоверность информации по всей организации. Федеративное ХД состоит из ряда экземпляров ХД, которые функционируют на полуавтономной основе и, как правило, организационно или географически разнесены, однако могут рассматриваться и управляться как одно большое ХД. Применение такой архитектуры снижает риск неудачи при глобальном развертывании системы, поскольку каждое локальное ХД меньше по масштабу, отвечает местным требованиям бизнеса и может управляться сотрудниками регионального подразделения.

Каждый из экземпляров федеративного ХД хранит копию базовой бизнес-модели и общие основные данные (common master dat), причем каждое ХД более высокого уровня содержит итоговые транзакционные данные более низкого уровня. Общие основные данные — например, схема организационной структуры компании — отправляются «вниз», т.е. из корпоративного (глобального) ХД, а суммарные данные о транзакциях отправляются «верх», т.е. из локального ХД. Таким образом, «федерация» ХД может предоставить местным отделениям необходимую гибкость, а также обеспечить общий

контроль и согласованность; при этом каждое ХД функционирует независимо от всех других остальных.

Для федеративных ХД характерны общая семантика и бизнес-правила, стандартизованный набор процессов извлечения из бизнес-правил, децентрализованные ресурсы и управление, параллельная разработка.

При этом следует учитывать, что важна необходимость в координировании работ, требуется согласованность среди различных отделов по вопросам архитектуры, бизнес-правил и семантики, сложная технологическая информационно-вычислительная среда.

Таким образом**, к**омпонентами типовой архитектуры ХД являются:

- программное обеспечение промежуточного слоя. Основное назначение этих компонент состоит в обеспечении доступа к сети и доступа к данным;
- БД OLTP систем и данные внешних источников;
- предварительная обработка и загрузка данных;
- ХД, реализованное средствами СУБД;
- метаданные, которые играют роль справочника о данных;
- уровень доступа к данным — программное обеспечение, которое обеспечивает взаимодействие конечных пользователей с данными ХД;
- уровень информационного доступа, который обеспечивает непосредственное общение пользователя с ХД;
- уровень администрирования.

Отметим, что в последнее время возрастает практический интерес к использованию ХД при формировании информационной инфраструктуры организаций. Преимущества, которые получает организация от внедрения хранилищ данных, следующие.

- *Взгляд на данные организации, как на единое целое.* Это ответы на такие вопросы: сколько продуктов реально производится? Что влияет на изменение спроса? Какие товары или услуги приносят наибольший

доход? А также возможность учитывать особенности и предпочтения клиентов.

- *Возрастает надежность данных для принятия решений.* Данные, загружаемые в хранилище данных, подвергаются очистке — согласуются, проверяются, уточняются.
- *Геопространственный анализ данных.* Анализ такой информации имеет решающее значение в принятии решений по всем вопросам, связанным с географией бизнеса.
- *Исследование трендов и колебаний в бизнес-данных.* Позволяет достаточно надежно прогнозировать развитие бизнес-процессов организации во времени.

Контрольные вопросы

1. Какие компоненты составляют архитектуру ХД?
2. Что оказывает решающее значение на выбор архитектуры ХД и методы его проектирования?
3. Какие факторы влияют на принятие решений о выборе способа реализации ХД?
4. Какие основные методологические подходы применимы к созданию ХД?
5. Чем характеризуется «корпоративная информационная фабрика»?
6. Какие структурные компоненты CIF вам известны?
7. Чем характеризуется архитектура федеративного ХД?

4 Основные бизнес-функции процесса разработки и проектирования хранилища данных.

4.1 Задачи процесса проектирования хранилища данных

Системы поддержки принятия решения, как и всякие сложные компьютерные системы, к которым относятся информационные системы с ХД, создаются командами IT-специалистов. Работа команды является процессом, который требует тонкой психологической настройки, мобилизации усилий каждого члена команды для реализации поставленных в проекте целей. Каждый член такой команды должен знать, какова цель проекта, какие задачи решаются в рамках проекта, какие задачи и в какие сроки должен решить каждый член команды. Помимо решения своих непосредственных задач, проектировщик ХД должен четко представлять себе:

- от кого и какие данные он должен получить (оценить риск получения неполных данных);
- какие и кому передать результаты решения своих задач (оценить риск срыва сроков их решения);
- с кем и когда согласовывать свои проектные решения (оценить риск межличностных конфликтов в команде);
- в каких точках реализации проекта он может получить распоряжение пересмотреть результаты решения своих задач (оценить риск повторной работы) и т. д.

Плохо спроектированная структура ХД обычно приводит к сложности реализации ХД и зачастую к увеличению сроков проекта примерно в три раза. Плохо организованный проект создания ХД, даже при качественном проектировании, ведет в более чем 50% случаев к краху проекта. Нередко проекты создания ХД реализуются со второй и даже с шестой попыток.

Организация любого проекта разработки ХД (используемая бизнес-модель) во многом определяет его успешность. В основу организационной структуры проекта, как правило, закладывается некоторая модель жизненного цикла создаваемого объекта.

Главная цель создания ХД состоит в том, чтобы собрать вместе информацию из различных источников и представить эту информацию в формате, который является удобным для принятия решений по основным направлениям деятельности организации. Действия, необходимые для достижения поставленной цели, оказываются значительно более сложными, чем просто собрать данные и получать из них ответы на запланированные вопросы или получить ряд отчетов.

Подготовка данных для таких систем требует всесторонней экспертизы со стороны как технических специалистов, так и специалистов предметной области, что обычно включает в себя:

- точную идентификацию бизнес-информации, которая должна храниться в ХД;
- идентификацию предметных областей (бизнес-направлений деятельности организации) и определение их приоритетов, которые составят в итоге набор предметных областей ХД;
- управление границами каждой предметной области, которая будет представлена в ХД на интерактивной основе;
- разработку масштабируемой архитектуры для технического обслуживания ХД и его приложений, точное определение и обоснованный выбор структурных компонент (аппаратной, программной, лингвистической и организационной), которые будут реализовать ХД;
- определение процедур экстрагирования, очистки, агрегации, преобразования и проверки данных для обеспечения гарантии их точности и согласованности;

- определение корректных уровней суммирования числовых показателей для поддержки бизнес-решений;
- определение и разработку процедур своевременного обновления данных, которые отвечают производственным потребностям, временным и жизненным циклам бизнес-информации;
- разработку дружественных интерфейсов и мощных инструментов аналитической обработки данных для конечных пользователей ХД;
- разработку словаря метаданных для обеспечения общения с ХД широких групп пользователей;
- обучение персонала для работы с приложениями ХД, всестороннее освещение круга решения задач, которые возможно решить в системе складирования данных;
- определение и регламентирование процесса сопровождения и обслуживания ХД и т.д.

Это еще раз подчеркивает, что ХД создаются для информационного обеспечения в решении стратегических задач по определенным направлениям бизнеса (для удовлетворения потребностей руководства и бизнес-аналитиков организации), а не для удовлетворения потребностей в автоматизации ежедневных процедур обработки бизнес-информации сотрудников структурных подразделений организации. В этом и состоит главное и коренное отличие систем складирования данных и систем бизнес-аналитики, использующих технологию ХД, от систем оперативной обработки информации (OLTP-систем, или On-Line Trasactions Proccessing).

Рассмотрим факторы, влияющие на структуру проекта создания хранилища данных. В современных экономических условиях стратегический бизнес-план организации разрабатывается на два-три года. Бизнес-процедуры, как правило, также пересматриваются в организации каждые два-четыре года. Это приводит к пересмотру функций и реинжинирингу OLTP-систем организации. ХД создаются на пять и более лет, в них накапливается огромный объем информации. ХД предназначено жить, даже если половина бизнес

направлений организации будет закрыта, а на их смену придут новые. Информация не теряется, она остается и будет доступна для анализа предыдущего опыта организации. Весьма расточительно планировать ХД так, чтобы через два года его переделывать [3].

Если язык общения системы бизнес-аналитики с конечными пользователями будет им не понятен, весь смысл и экономический эффект от разработки и внедрения такой системы будет сведен на ноль. Поэтому сбор и систематизация бизнес-терминологии, разработка открытой и гибкой структуры лингвистического обеспечения, ее внедрение в структуру ХД (а не в аналитические бизнес-приложения) является важной задачей проекта создания ХД.

Системы бизнес-аналитики создаются для лиц, принимающих решения, или приравненных к ним ведущих специалистов компаний. Эти системы обслуживают в первую очередь стратегические решения. Ошибочно полагать, что любой персонал способен принимать подобные решения для организации. Однако такая постановка вопроса отнюдь не мешает пользоваться персоналу организации ХД как проверенным информационным источником накопленных компанией знаний.

Таким образом, в проекте ХД вес разработки каждого структурного компонента приблизительно одинаков, ничто: ни аппаратные решения, ни программное, ни сетевое, ни лингвистическое, ни технологическое обеспечение — не должно быть обойдено вниманием. Недооценка этого факта приводит к краху проекта создания ХД. Поэтому тщательное планирование и организационная структура проекта создания ХД очень важна.

4.2 Модель жизненного цикла хранилища данных

В основу организационной структуры проекта создания или разработки ХД закладывается определенная *модель* (или представление) ***жизненного цикла ХД*** как продукта. Под жизненным циклом продукта понимается набор

определенным образом расположенных во времени фаз (этапов), которые проходит продукт от момента его создания до момента его утилизации.

В зависимости от квалификации руководителя и масштаба проекта создания ХД используются модели жизненного цикла различной степени детализации. В нашем пособии мы не будем углубляться в анализ различных методологических подходов к построению моделей жизненного цикла ХД. Все они основаны на обоснованном выборе того или иного способа разбиения процесса разработки ХД на этапы (фазы), идентификации задач каждого этапа и расположения выбранных этапов в пространстве и во времени. Отметим, что в настоящее время различные исследователи предлагают разбиение процесса разработки ХД на три, четыре, пять и более этапов. Отметим также, что все модели жизненного цикла разработки ХД имеют общие (типовые) элементы, которые в разных моделях могут быть отнесены к различным этапам.

Рассмотрим типовые элементы процесса создания ХД, которые и определяют его жизненный цикл. При этом предлагаемая модель не является единственно верной, так как сам процесс разработки не является строго структурированной задачей, она отражает лишь общий взгляд на этот процесс.

После принятия решения о создании ХД, до начала выполнения этапа планирования формируется проектная команда, определяются роли и задачи ее участников. Поэтому проектировщику важно представлять, что должны делать другие члены команды, а для этого нужно понимать свое место и роль в процессе разработки ХД.

Процесс создания и разработки ХД (жизненный цикл разработки хранилища данных) в общем случае можно представить состоящим из следующих основных стадий:

- планирование;
- формулирование требований к СППР;
- анализ;
- проектирование;
- конструирование;

- внедрение;
- поддержка.

Этап поддержки ХД в жизнеспособном состоянии включен для полноты представления жизненного цикла разработки ХД. Как правило, поддержка ХД осуществляется ИТ-службами заказчика в рамках отдельного проекта и имеет свой жизненный цикл. Отметим также, что в настоящее время внедрение также стало рассматриваться как самостоятельный проект — совместный (корпоративный) проект разработчика и заказчика ХД. Охарактеризуем кратко задачи каждого из этих этапов.

4.2.1 Планирование

На **этапе планирования** решаются следующие задачи: выбор стратегии реализации и методологии разработки, анализ задач, для решения которых создается ХД, анализ ресурсов разработки с технологической точки зрения и с точки зрения задач бизнеса, выбор архитектуры ХД, определяется бюджет проекта, разрабатываются возможные сценарии использования ХД, начинается сбор метаданных для ХД.

Целью этого этапа является идентификация задач ХД, выбор способа решения этих задач, определения программно-технологического объекта и того, как, в какие сроки и за какие деньги этот объект будет реализоваться.

Выбор стратегии реализации определяет подход, который будет использован при создании ХД. Обычно используют следующие подходы: Top Down («сверху вниз»), Bottom Up («снизу вверх»), Middle In («из середины») и комбинированный подход, который в последнее время становится все более популярным. Подход «сверху вниз» выбирается для вновь создаваемого ХД, т.е. когда «с нуля» принимаются все решения о технологической реализации объекта (аппаратура, программное обеспечение и т.д.). Подход «снизу вверх» используется, когда уже есть определенная вычислительная среда и объекты, из которых можно построить новый объект. Подход «из середины» предполагает эволюционное, поэтапное создание объекта, когда сначала разрабатывается так

называемое ядро объекта, которое на следующих этапах наращивается новой функциональностью. Комбинированный подход применяет комбинацию выше перечисленных подходов.

Стратегия реализации определяет, какая концептуальная архитектура ХД будет использоваться, и как будет строиться ХД с точки зрения последовательности реализации выбранной концептуальной архитектуры.

Выбор методологии создания ХД определяет, образно говоря, язык проекта, на котором будут разговаривать члены проектной команды, как будет оформлена техническая документация, какие принципы разработки будут использоваться. Это могут быть метод структурного анализа и проектирования (SADT), спиральный метод, методы, основанные на применении UML (язык объектно-ориентированного моделирования), и т.п.

Методология разработки основывается на использовании типичной для БД концепции трех схем и включает методы анализа спецификаций, методы концептуального, логического и физического проектирования.

Метод структурного анализа и проектирования является хорошо разработанной методикой и опирается на использование стандартов и методик IDEF или аналогичных им. Методика создания ХД по спирали реализует концепцию эволюционного подхода к созданию системы. Использование методов объектно-ориентированного анализа для реляционных ХД потребует преобразования полученной объектной схемы несущей базы данных в реляционную схему.

Анализ задач ХД предполагает идентификацию и определение объектов бизнеса, информация о которых будет содержаться в ХД. Вот далеко не полный перечень вопросов, на которые следует получить ответ на этой стадии:

- Что является предметной областью для хранилища данных?
- Какие программно-аппаратные платформы используются или какие планируется использовать?
- Какие возможности планируются в терминах свойств, характеристик и функций?

- Что представляют собой источники данных, которые можно или нужно интегрировать в хранилище данных?
- Когда хранилище данных должно начать функционировать?

Определение предметной (тематической) ориентации ХД является одной из самых главных задач на этом этапе. В рамках каждого фрагмента предметной области определяется:

- количество и типы источников данных (подразделения организации);
- число выбранных источников данных;
- данные, которые будут храниться в ХД;
- цели использования данных;
- каким должен быть спектр вопросов, на который должно отвечать ХД;
- каков размер метамодели хранилища данных;
- каков размер данных в хранилище данных;
- каковы источники входных данных и их количество;
- насколько востребованными являются данные из источников данных;
- насколько хорошо документированы данные из источников данных;
- каковы уровни управления организации (IFC);
- какие инструментальные средства доступны для логического проектирования;
- доступна ли модель данных в масштабе организации;
- есть ли профессионально подготовленный персонал;
- будет ли хранилище данных реализоваться в рамках существующей программно-аппаратной платформы или на платформе, аналогичной существующей.

На этой стадии планирования решается вопрос, что должно быть реализовано сначала, что можно реализовать в дальнейшем и, самое главное,

что не будет реализовано в этом проекте создания хранилища данных. Задача состоит в том, чтобы четко ограничить предметную область хранилища данных и понять (переформулировать) спецификации для хранилища данных.

Выбор архитектурных решений должен задать такие ограничения:

- степень использования данных систем оперативной обработки;
- только хранилище данных;
- только киоски данных;
- хранилище и киоски данных;
- инфраструктура секционирования данных;
- использование архитектуры клиент-сервер (двухзвенная или трехзвенная архитектура).

Разработка программы проекта и бюджетного плана имеет целью составить план проекта, построить бюджет проекта, оценить стоимость и обеспечение методов для оценки окупаемости ХД в таких понятиях как:

- оценка стоимости восстановления и стоимости хранения;
- оценка возможности организации;
- оценка доходности организации;
- оценка расширения рынка;
- оценка конкурентного преимущества;
- оценка удовлетворения потребностей клиентов.

На этом этапе разработчики ХД должны работать в тесном взаимодействии с экспертами и экономистами.

Проработка сценариев типовой коммерческой деятельности для согласования с конечными пользователями является ответственной задачей, решение которой дает ответ на вопрос, *как пользователи видят использование ХД*. Решение задачи очень важно не только с точки зрения дальнейшей реализации проекта, но с точки зрения снижения риска неоправданных ожиданий от внедрения ХД.

Обычно анализ сценариев строится по следующей схеме. Каждый сценарий состоит из следующего:

- четко определенного конечного пользователя с точно определенной его ролью в бизнес-процессах;
- тематической области, которая представляется в хранилище или киоске данных и будет использоваться данным конечным пользователем;
- одного или более вопросов (задач) по ключевым проблемам тематической области, которые воплощены неудовлетворительно в существующих ИС и полезны для решения следующих задач:
- построение адекватных критериев запросов для ХД;
- идентификация метамодели ХД;
- идентификация объема необходимой хронологической информации;
- идентификация точек зрения пользователя на данные (измерений);
- идентификация потребностей по данным киосков и ХД.

Цель настоящего этапа — очертить круг вопросов, на которые должна отвечать СППР.

Цель следующей стадии, сбора метаданных, — сформировать спецификации по описанию данных и описать все необходимые элементы данных ХД в терминах бизнес-процессов и процедур бизнеса.

На этом этап планирования заканчивается, и команда проекта переходит к формированию каталога требований проекта. Без этого документа не имеет смысла продолжать проект, в намеченный срок он не закончится и в планируемый бюджет не вложится.

4.2.2 Разработка требований.

Этап разработки требований к ХД включает в себя следующие стадии:

- определение требований владельца ХД;
- определение требований конечных пользователей;
- определение технологических требований;

- определение архитектурных требований.

Требования владельца связаны с уточнением предметно-ориентированной направленности ХД, требованиями потенциального роста предметной области по таким направлениям как:

• цели бизнеса;

• масштаб и цели хранилища данных/киосков данных, по клиентам и посредникам;

• требования клиентов;

• источники данных;

• план — бюджет, календарный план-график, ресурсы;

• влияние на текущие инвестиции — люди, технология, обучение.

Часть требований бизнеса является спецификацией предметной области для ХД — спецификацией бизнес-функций направлений деятельности определенной категории потенциальных пользователей (идентификация интереса). Рассмотрим, например, отдел маркетинга компании. Отдел маркетинга может интересовать одна из следующих тем (направлений деятельности):

• маркетинговые исследования;

• анализ конкурентоспособности;

• анализ поведения покупателя;

• сегментация (рынка) реализации продукта;

• ценовые и бюджетные решения;

• решения по продукции;

• решения по продвижению;

• решения по каналам сбыта;

• прогнозирование тенденций;

• сертификация (Benchmarking).

Анализ выбранных выше направлений имеет дело со спецификацией предметных областей для отдела маркетинга, которыми в данном случае будут

являться «Заказы», «Рынки», «Продажи», «Продвижение продукции», «Временные циклы».

Здесь важен такой параметр, как **степень детализации** (granularity) — чем ниже степень детализации, тем больше количество деталей. Это ключевая информация для проектировщика ХД, поскольку она определяет потенциальный объем ХД, скорость его роста, потенциальную производительность обработки запросов и т.д.

На этой стадии идентифицируется, в каких разрезах будут исследоваться данные потенциальными пользователями ХД. Для примера с отделом маркетинга это может быть:

• временной цикл с детализацией «День», «Неделя», «Месяц», «Квартал», «Год»;

• группы клиентов с детализацией «Покупатели», «Сегмент рынка», «Рынок», «Отрасль промышленности»;

• семейство продуктов с детализацией «Продукция», «Семейство продуктов», «Линия продуктов»;

• география и месторасположение с детализацией «Район», «Регион страны», «Страна», «Международный регион»;

• структура организации с детализацией «Подразделение», «Отдел», «Филиал», «Организация».

Здесь может быть также учтена специфика как самой организации, так и сектора промышленности и рынка, на которых работает организация.

На стадии определения требований по архитектуре ХД проводится планирование архитектуры в масштабе предприятия и по его структурным подразделениям, определяется архитектура данных в рамках принятой модели (например, ER-модели и т.д.), определяется архитектура приложений ХД, разрабатывается каталог приложений и функции каждого из них.

На стадии определения требований разработки создаются:

- спецификация приложений, интерфейсов, компьютеров, баз данных, коммуникаций и форм для конечного пользователя;

- технологические требования;
- требования размещения ПО;
- требования по поддержке ХД в программных продуктах;
- требования для разработчиков и обслуживающего персонала по их квалификации.

На стадии определения требования конечных пользователей оценивается роль ХД в делопроизводстве, соответствие функциональности ХД ежедневной последовательности операций конечных пользователей, формулируются критерии запросов к данным, требования по корпоративной отчетности, требования анализа данных.

Например, классифицируются в операциях над ХД действия пользователей, такие как:

- разделение элементов данных по различным аспектам анализа (операции Slice and dice);
- развертывание и экспозиция элементов данных по уровням иерархии (операции Drill-down up);
- поиск скрытых закономерностей в данных (алгоритмы Data Mining);
- просмотр данных непредусмотренным образом (Datasurfing);
- загрузка и выполнение модификаций данных.

В заключение этой стадии разрабатывается также набор бизнес-моделей использования ХД, после чего принимаются решения по просмотру данных. Например, принимается решение о том, как будут просматриваться различные категории данных:

- с помощью стандартной визуализации в рамках реляционных СУБД (двумерные таблицы);
- с помощью многомерных кубов данных (OLAP);
- с помощью предусмотренных отчетов и диаграмм;
- на основе жизненных циклов элементов данных.

Главным результатом этого этапа является создание каталога требований.

4.2.3 Анализ

Располагая каталогом требований, можно приступить к реализации **этапа анализа** — получения согласованных по источникам логической модели и определения набора инструментальных средств для работы с ХД.

Цель первой стадии этого этапа состоит в разработке логической модели данных для ХД и киосков данных. Кроме создания логической модели, необходимо фиксировать модели логической структуры баз данных подающих систем.

На следующей стадии определяются процессы, которые необходимы для связи источников данных и ХД, процедур очистки, преобразования и агрегации данных источников, производится выбор инструментальных средств загрузки данных в ХД и выбор инструментальных средств конечного пользователя, которые должны быть размещены на клиентских компьютерах.

4.2.4 Проектирование

После завершения стадии анализа переходят к реализации **стадии проектирования**. Цель этого этапа — разработка физической модели ХД, проектирование процедур поступления данных в него и проектирование архитектуры приложений.

Проектирование ХД можно разложить на две равноценных стадии — проектирование архитектуры данных (логическое и физическое проектирование) и архитектуры приложений (анализ запросов и фиксация процессов взаимодействия хранилища данных с внешними источниками и пользователями).

Детальное проектирование архитектуры данных включает в себя:

- **логическое проектирование.** Разработка логических моделей данных для ХД и киосков данных в рабочем пространстве базы данных. Отображение логических моделей данных источников данных в логические модели ХД и киосков данных;

- **физическое проектирование.** Отображение логических объектов в физическое описание ХД. Денормализация логической модели. Создание табличного пространства, секционирование, создание индексов и ограничений. Оценка объема физического ввода-вывода.

На стадии логического проектирования рассматриваются логические взаимоотношения между объектами предметной области. Особенностью логического проектирования ХД является тотальная ориентация на потребности и задачи конечного пользователя — бизнес-аналитиков и руководителей организации различных уровней. Они будут анализировать данные и работать с агрегированными данными, а не с данными конкретной бизнес-операции. Этот класс пользователей, как правило, обычно не знает точно, что им все-таки нужно от данных. И здесь важно оценить уровень претензий к данным и очертить круг задач, которые могут возникнуть.

В процессе логического проектирования выделяется набор объектов предметной области с их атрибутами. Объект представляет собой фрагмент информации о предметной области. В реляционных базах данных объект отображается в таблицу, а его атрибуты отображаются в колонки такой таблицы. В логическом проектировании ХД для создания ER-диаграмм используется многомерное моделирование, которое, грубо говоря, сводится к идентификации информации об объекте в виде фактов (таблицы фактов) и идентификации информации, с помощью которой на эти факты можно посмотреть (таблицы измерений). Результатом стадии логического проектирования является логическая схема ХД и, возможно, отображение логической схемы источников данных (подающих систем) на логическую схему ХД.

На стадии физического проектирования рассматриваются задачи размещения данных в БД для эффективной их выборки. На этой стадии (для реляционных хранилищ данных) реально пишутся SQL-операторы. Данные, собранные на стадии логического проектирования, превращаются в описание физической БД, включая таблицы, уникальные идентификаторы объектов — в

первичные ключи, ограничения по значениям данных, взаимоотношения между объектами — во внешние ключи, индексы, табличные пространства, разбиения и представления.

Назначение табличного пространства состоит в отделении таблиц от их индексов, маленьких таблиц от больших таблиц, т.е. является механизмом для решения задачи оптимального размещения данных (некоторые СУБД решают эту задачу сами, без вмешательства пользователя).

Секционирование больших таблиц необходимо для увеличения производительности обработки запросов, т.е. является одним из механизмов решения задачи оптимизации выборки, так же, как и создание индексов.

Ограничения в ХД отличаются от ограничений в OLTP-системах. В системах складирования данных целостность и проверка данных обеспечивается на стадии загрузки данных. Поэтому роль ограничений в ХД не столь уж велика. Типичным ограничением в ХД является ограничение NOT NULL.

Параллельно с проектирование архитектуры данных начинается стадия детального проектирования структуры приложений, которая включает в себя определение:

- процессов, которые являются внутренними для источников данных и связаны с процедурами очистки и частичной экстракции информации;
- процессов, которые связывают источники данных с ХД и киосками данных;
- процессов, которые являются внутренними по отношению к ХД и предназначены исключительно для обслуживания данных в хранилище;
- процессов, которые связывают ХД и киоски данных;
- процессов, которые являются внутренними по отношению к киоскам данных и предназначены исключительно для обслуживания данных в них;
- процессов, которые связывают хранилище и киоски данных со средствами конечного пользователя;

- процессов, которые являются внутренними на рабочих станциях конечных пользователей и используются для установки связи с ХД и запуском средств анализа;
- процессов для поддержки управления и администрирования внутренних задач ХД как системы.

Целью этой стадии является получение спецификаций всех приложений ХД. Все результаты стадии проектирования тщательно документируются, поскольку являются исходными и рабочими документами команды проекта и постоянно используются в ходе дальнейшей работы над проектом.

4.2.5 Построение хранилища данных

Следующий этап проекта создания ХД — это **построение ХД**. Цель этого этапа состоит в разработке программ и собственно физической БД под ХД. Выполнение этого этапа проекта, помимо создания собственно ХД, включает в себя разработку и отладку приложений ХД, а именно следующих групп программ:

- программы, которые создают и модифицируют БД для ХД и киосков данных;
- программы, которые экстрагируют данные из источников данных;
- программы, которые выполняют преобразования данных, такие как интеграция, суммирование и агрегация;
- программы, которые выполняют обновление реляционных БД;
- программы, которые реализуют поиск в очень больших БД. Результатом выполнения этого этапа является комплекс программ,
- работающих с ХД.

4.2.6 Внедрение

Следующим этапом реализации проекта создание является **внедрение** в опытную эксплуатацию. Это очень ответственный и трудоемкий этап. Начинается он со стадии начальной инсталляции, включающей начальную загрузку хранилища из источников данных, и тестирования процедур обновления и синхронизации данных.

Далее составляются и утверждаются планы постадийного тестирования, тренировки и обучения персонала для всех классов пользователей, реализации обновления несущих платформ для ХД, информационных потребностей и т. д.

Выполняется обеспечение администрирования системы и данных о пользователях, возможностей архивирования и резервного копирования, возможностей восстановления, управления доступом и безопасностью, полных возможностей и процесса управления системой и ее структурными компонентами, информационного каталога/директории. Проводится целостная интеграция хранилища данных в существующую инфраструктуру организации. Результатом выполнения этого этапа является всесторонняя подготовка перехода ХД в промышленную эксплуатацию.

4.2.7 Поддержка

Этап поддержки ХД в работоспособном состоянии, как уже отмечалось выше, является самостоятельным проектом. Это последний этап жизненного цикла ХД. По его завершении происходит либо уничтожение ХД как продукта, либо его реинжиниринг. Это связано с быстрыми изменениями сферы бизнеса организации, которые приводят к необходимости пересмотра стратегических и тактических планов организации. ХД, так же, как и другие элементы инфраструктуры организации, должно соответствовать генеральному бизнес-плану организации. В зависимости от структуры генерального бизнес-плана организации, эксплуатирующей ХД, продолжительность этого этапа может составлять два-четыре года, реже пять и более лет.

На этом этапе ИТ-служба организации обеспечивает:

- поддержку работоспособности и масштабируемости программно-аппаратного обеспечения ХД;
- сбор, очистку, преобразование, загрузку и актуализацию данных в соответствии с установленными бизнес-процедурами;
- поддержку автоматизированных мест пользователей.

Этот этап может также включать в себя техническую поддержку со стороны разработчика ХД, в частности консультирование, обучение пользователей или абонентское обслуживание.

Этот этап является также испытанием ХД «на прочность». Конечные пользователи, как правило, не являются ИТ-профессионалами. Они решают свои задачи. Они будут использовать ХД только в том случае, если оно будет эффективно помогать им решать их задачи. Поэтому ИТ-служба организации должна постоянно отслеживать активность пользователей на ХД и вести постоянный аудит мнений пользователей о полезности ХД в их профессиональной деятельности.

Временные затраты на реализацию этапов. Таким образом, учитывая замечание о поддержке ХД как о самостоятельном проекте, в жизненном цикле создания ХД можно выделить шесть основных этапов. Независимо от того, как заказчик ХД понимает процесс его создания, эти этапы должны быть пройдены и заложены в бюджет проекта. Они обычно выполняются в последовательном порядке, хотя некоторые из них могут разрабатываться параллельно.

Выше была рассмотрена классическая схема организации проекта разработки программного обеспечения в приложении к ХД. Руководитель проекта может придерживаться иной схемы разбиения на этапы.

Контрольные вопросы

1. Какие факторы влияют на структуру проекта создания хранилища данных?

2. В чем заключается главная цель создания ХД?
3. Для кого создаются системы бизнес-аналитики?
4. Что понимается под жизненным циклом продукта?
5. Из каких основных стадий состоит процесс создания и разработки ХД?
6. Какие задачи решаются на этапе планирования жизненного цикла создания ХД?
7. Какие стадии включает в себя этап разработки требований к ХД?

5. OLAP-системы

5.1. Многомерная модель данных

Как было сказано выше, в концепции ХД нет постановки вопросов, связанных с организацией эффективного анализа данных и предоставления доступа к ним. Эти задачи решаются подсистемами анализа. Посмотрим, какой способ работы с данными наиболее подходит пользователю СППР — аналитику.

В процессе принятия решений пользователь генерирует некоторые гипотезы. Для превращения их в законченные решения эти гипотезы должны быть проверены. Проверка гипотез осуществляется на основании информации об анализируемой предметной области. Как правило, наиболее удобным способом представления такой информации для человека является зависимость между некоторыми параметрами. Например, зависимость объемов продаж от региона, времени, категории товара и т. п. Другим примером может служить зависимость количества выздоравливающих пациентов от применяемых средств лечения, возраста и т. п.

В процессе анализа данных, поиска решений часто возникает необходимость в построении зависимостей между различными параметрами. Кроме того, число таких параметров может варьироваться в широких пределах. Как уже отмечалось ранее, традиционные средства анализа, оперирующие данными, которые представлены в виде таблиц реляционной БД, не могут в полной мере удовлетворять таким требованиям. В 1993 г. Е. Кодд — основоположник реляционной модели БД — рассмотрел ее недостатки, указав в первую очередь на невозможность "объединять, просматривать и анализировать данные с точки зрения множественности измерений, т. е. самым понятным для аналитиков способом".

Измерение — это последовательность значений одного из анализируемых параметров. Например, для параметра "время" это последовательность кален-

дарных дней, для параметра "регион" это может быть список городов. Множественность измерений предполагает представление данных в виде многомерной модели. По измерениям в многомерной модели откладывают параметры, относящиеся к анализируемой предметной области.

По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) — это множественная перспектива, состоящая из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям определяется как многомерный анализ.

Каждое измерение может быть представлено в виде иерархической структуры. Например, измерение "Исполнитель" может иметь следующие иерархические уровни: "предприятие — подразделение — отдел — служащий". Более того, некоторые измерения могут иметь несколько видов иерархического представления. Например, измерение "Время" может включать две иерархии со следующими уровнями: "год — квартал — месяц — день" и "неделя — день".

На пересечениях осей измерений (Dimensions) располагаются данные, количественно характеризующие анализируемые факты, — меры (Measures). Это могут быть объемы продаж, выраженные в единицах продукции или в денежном выражении, остатки на складе, издержки и т. п.

Таким образом, многомерную модель данных можно представить как гиперкуб (рис. 5.1) (конечно, название не очень удачное, поскольку под кубом обычно понимают фигуру с равными ребрами, что в данном случае далеко не так). Ребрами такого гиперкуба являются измерения, а ячейками — меры.

Над таким гиперкубом могут выполняться следующие операции.

Срез (Slice) (рис. 5.2) — формирование подмножества многомерного массива данных, соответствующего единственному значению одного или нескольких элементов измерений, не входящих в это подмножество.

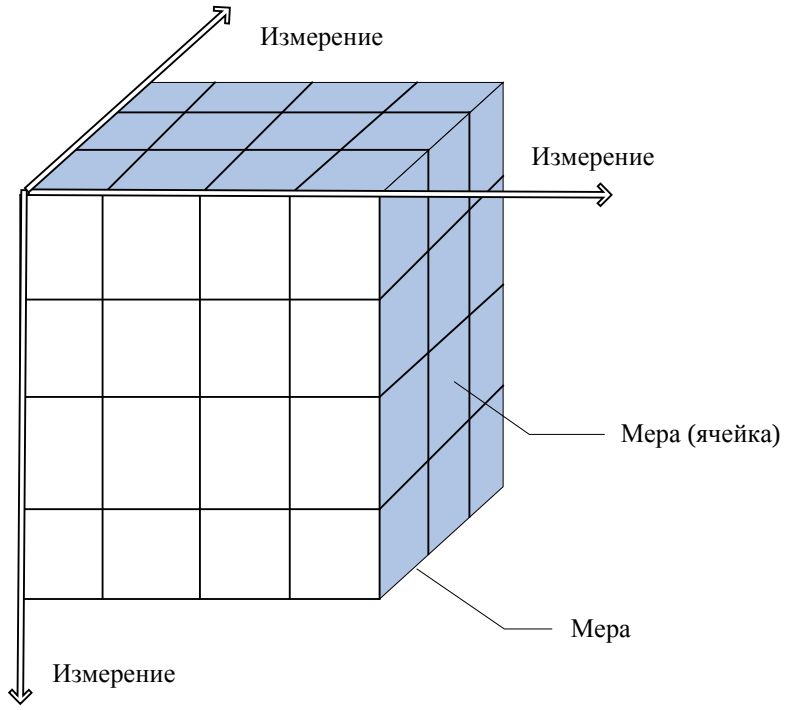

Рис. 5.1. Представление данных в виде гиперкуба

Данные, что не вошли в сформированный срез, связаны с теми элементами измерения, которые не были указаны в качестве определяющих. Если рассматривать термин "срез" с позиции конечного пользователя, то наиболее часто его роль играет двумерная проекция куба.

Вращение (Rotate) (рис. 5.3) — изменение расположения измерений, представленных в отчете или на отображаемой странице. Например, операция вращения может заключаться в перестановке местами строк и столбцов таблицы или перемещении интересующих измерений в столбцы или строки создаваемого отчета, что позволяет придавать ему желаемый вид. Кроме того, вращением куба данных является перемещение внетабличных измерений на место измерений, представленных на отображаемой странице, и наоборот (при этом внетабличное измерение становится новым измерением строки или измерением столбца).

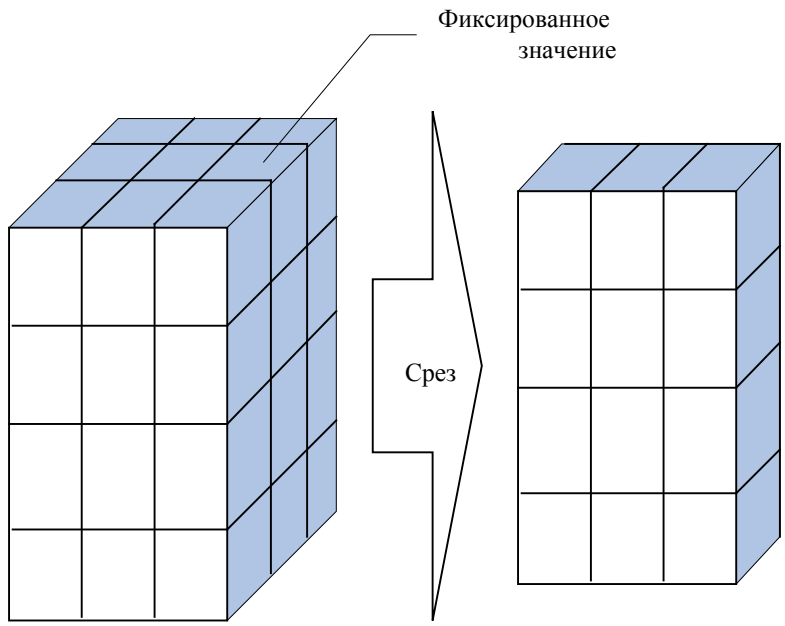

Рис. 5.2. Операция среза

В качестве примера первого случая может служить отчет, для которого элементы измерения "Время" располагаются поперек экрана (являются заголовками столбцов таблицы), а элементы измерения "Продукция" — вдоль экрана (заголовки строк таблицы). После применения операции вращения отчет будет иметь следующий вид: элементы измерения "Продукция" будут расположены по горизонтали, а элементы измерения "Время" — по вертикали. Примером второго случая может служить преобразование отчета с измерениями "Меры" и "Продукция", расположенными по вертикали, и измерением "Время", расположенным по горизонтали, в отчет, у которого измерение "Меры" располагается по вертикали, а измерения "Время" и "Продукция" — по горизонтали. При этом элементы измерения "Время" располагаются над элементами измерения "Продукция". Для третьего случая применения операции вращения можно привести пример преобразования отчета с расположенными по горизонтали измерением "Время" и по вертикали измерением "Продукция" в отчет, у которого по горизонтали представлено измерение "Время", а по вертикали — измерение "География".

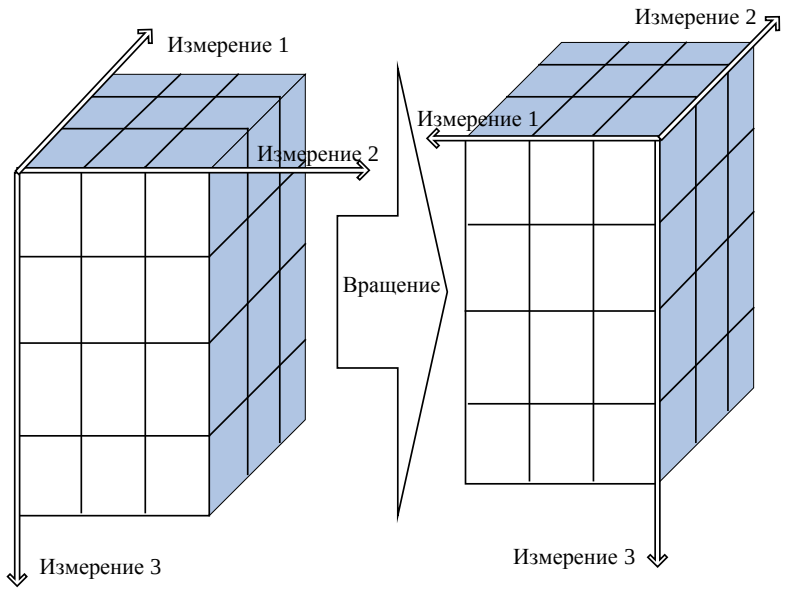

Рис. 5.3. Операция вращения.

Консолидация (Drill Up) и детализация (Drill Down) (рис. 5.4) — операции, которые определяют переход вверх по направлению от детального (down) представления данных к агрегированному (up) и наоборот, соответственно. Направление детализации (обобщения) может быть задано как по иерархии отдельных измерений, так и согласно прочим отношениям, установленным в рамках измерений или между измерениями. Например, если при анализе данных об объемах продаж в Северной Америке выполнить операцию Drill Down для измерения "Регион", то на экране будут отображены такие его элементы, как "Канада", "Восточные Штаты Америки" и "Западные Штаты Америки". В результате дальнейшей детализации элемента "Канада" будут отображены элементы "Торонто", "Ванкувер", "Монреаль" и т. д.

С концепцией многомерного анализа данных тесно связывают оперативный анализ, который выполняется средствами OLAP-систем.

OLAP (On-Line Analytical Processing) — технология оперативной аналитической обработки данных, использующая методы и средства для сбора, хранения и анализа многомерных данных в целях поддержки процессов принятия решений.

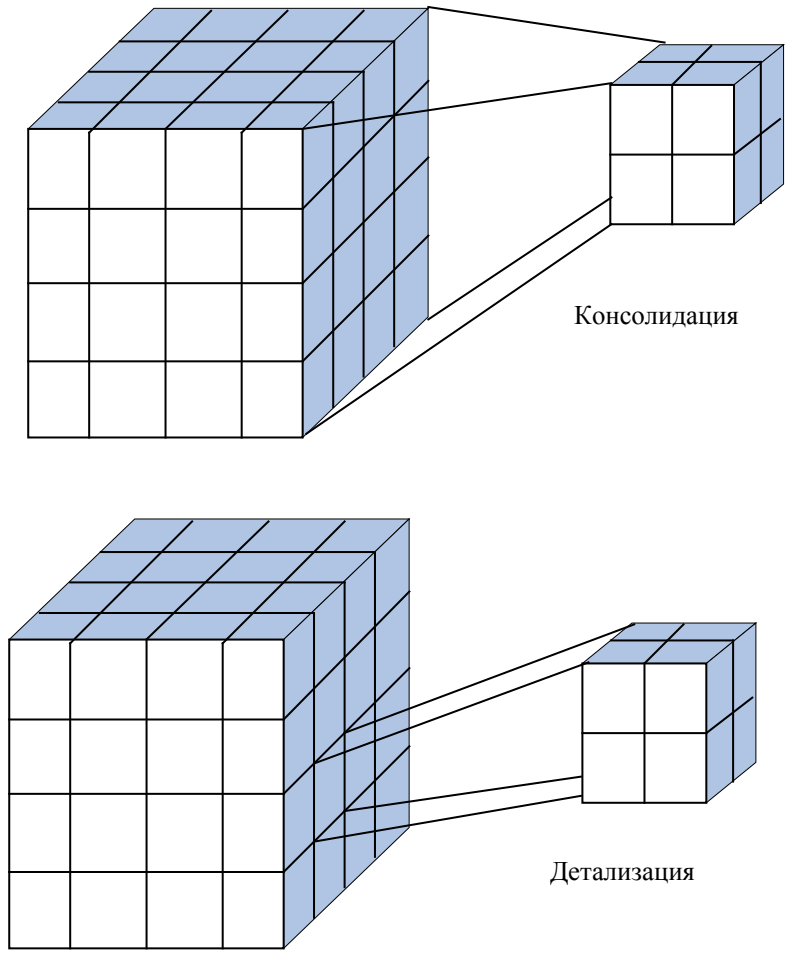

Рис. 5.4. Операции консолидации и детализации.

Основное назначение OLAP-систем — поддержка аналитической деятельности, произвольных (часто используется термин ad-hoc) запросов пользователей-аналитиков. Цель OLAP-анализа — проверка возникающих гипотез.

У истоков технологии OLAP стоит основоположник реляционного подхода Э. Кодд. В 1993 г. он опубликовал статью под названием "OLAP для пользователей-аналитиков: каким он должен быть". В данной работе были изложены основные концепции оперативной аналитической обработки и определены 12 требований, которым должны удовлетворять продукты, позволяющие выполнять оперативную аналитическую обработку.

5.2 Двенадцать правил Кодда

Рассмотрим 12 правил, изложенных Коддом и определяющих OLAP:

1. Многомерность. OLAP-система на концептуальном уровне должна представлять данные в виде многомерной модели, что упрощает процессы анализа и восприятия информации.

2. Прозрачность. OLAP-система должна скрывать от пользователя реальную реализацию многомерной модели, способ организации, источники, средства обработки и хранения.

3. Доступность. OLAP-система должна предоставлять пользователю единую, согласованную и целостную модель данных, обеспечивая доступ к данным независимо оттого, как и где они хранятся.

4. Постоянная производительность при разработке отчетов. Производительность OLAP-систем не должна значительно уменьшаться при увеличении количества измерений, по которым выполняется анализ.

5. Клиент-серверная архитектура. OLAP-система должна быть способна работать в среде "клиент-сервер", т. к. большинство данных, которые сегодня требуется подвергать оперативной аналитической обработке, хранятся распределенно. Главной идеей здесь является то, что серверный компонент инструмента OLAP должен быть достаточно интеллектуальным и позволять строить общую концептуальную схему на основе обобщения и консолидации различных логических и физических схем корпоративных БД для обеспечения эффекта прозрачности.

6. Равноправие измерений. OLAP-система должна поддерживать многомерную модель, в которой все измерения равноправны. При необходимости дополнительные характеристики могут быть предоставлены отдельным измерениям, но такая возможность должна быть у любого измерения.

7. Динамическое управление разреженными матрицами. OLAP-система должна обеспечивать оптимальную обработку разреженных матриц. Скорость доступа должна сохраняться вне зависимости от расположения ячеек данных и

быть постоянной величиной для моделей, имеющих разное число измерений и различную степень разреженности данных.

8. Поддержка многопользовательского режима. OLAP-система должна предоставлять возможность нескольким пользователям работать совместно с одной аналитической моделью или должна создавать для них различные модели из единых данных. При этом возможны как чтение, так и запись данных, поэтому система должна обеспечивать их целостность и безопасность.

9. Неограниченные перекрестные операции. OLAP-система должна обеспечивать сохранение функциональных отношений, описанных с помощью определенного формального языка между ячейками гиперкуба при выполнении любых операций среза, вращения, консолидации или детализации. Система должна самостоятельно (автоматически) выполнять преобразование установленных отношений, не требуя от пользователя их переопределения.

10. Интуитивная манипуляция данными. OLAP-система должна предоставлять способ выполнения операций среза, вращения, консолидации и детализации над гиперкубом без необходимости пользователю совершать множество действий с интерфейсом. Измерения, определенные в аналитической модели, должны содержать всю необходимую информацию для выполнения вышеуказанных операций.

11. Гибкие возможности получения отчетов. OLAP-система должна поддерживать различные способы визуализации данных, т. е. средства формирования отчетов должны представлять синтезируемые данные или информацию, следующую из модели данных, в ее любой возможной ориентации. Это означает, что строки, столбцы или страницы должны показывать одновременно от 0 до N измерений, где N— число измерений всей аналитической модели. Кроме того, каждое измерение содержимого, показанное в одной записи, колонке или странице, должно позволять показывать любое подмножество элементов (значений), содержащихся в измерении, в любом порядке.

12. Неограниченная размерность и число уровней агрегации. Исследование о возможном числе необходимых измерений, требующихся в аналитической модели, показало, что одновременно могут использоваться до 19 измерений. Отсюда вытекает настоятельная рекомендация, чтобы аналитический инструмент мог одновременно предоставить хотя бы 15, а предпочтительнее и 20 измерений. Более того, каждое из общих измерений не должно быть ограничено по числу определяемых пользователем-аналитиком уровней агрегации и путей консолидации.

5.3 Дополнительные правила Кодда

Набор правил Кодда, послуживших де-факто определением OLAP, достаточно часто вызывает различные нарекания, например, правила 1, 2, 3, 6 являются требованиями, а правила 10, 11— неформализованными пожеланиями. Таким образом, перечисленные 12 правил Кодда не позволяют точно определить OLAP [1].

В 1995 г. Кодд к приведенному перечню добавил следующие шесть правил:

13. Пакетное извлечение против интерпретации. OLAP-система должна в равной степени эффективно обеспечивать доступ как к собственным, так и к внешним данным.

14. Поддержка всех моделей OLAP-анализа. OLAP-система должна поддерживать все четыре модели анализа данных, определенные Коддом: категориальную, толковательную, умозрительную и стереотипную.

15. Обработка ненормализованных данных. OLAP-система должна быть интегрирована с ненормализованными источниками данных. Модификации данных, выполненные в среде OLAP, не должны приводить к изменениям данных, хранимых в исходных внешних системах.

16. Сохранение результатов OLAP: хранение их отдельно от исходных данных. OLAP-система, работающая в режиме чтения-записи, после модификации исходных данных должна сохранять результаты отдельно. Иными словами, должна обеспечиваться безопасность исходных данных.

17. Исключение отсутствующих значений. OLAP-система, представляя данные пользователю, должна отбрасывать все отсутствующие значения. Другими словами, отсутствующие значения должны отличаться от нулевых значений.

18. Обработка отсутствующих значений. OLAP-система должна игнорировать все отсутствующие значения без учета их источника. Эта особенность связана с 17-м правилом.

Кроме того, Е. Кодд разбил все 18 правил на следующие четыре группы, назвав их особенностями. Эти группы получили названия B, S, R. и D.

Основные особенности (B) включают следующие правила:

- многомерное концептуальное представление данных (правило 1);
- интуитивное манипулирование данными (правило 10);
- доступность (правило 3);
- пакетное извлечение против интерпретации (правило 13);
- поддержка всех моделей OLAP-анализа (правило 14);
- архитектура "клиент-сервер" (правило 5);
- прозрачность (правило 2);
- многопользовательская поддержка (правило 8).

Специальные особенности (S):

- обработка ненормализованных данных (правило 15);
- сохранение результатов OLAP: хранение их отдельно от исходных данных (правило 16);
- исключение отсутствующих значений (правило 17);
- обработка отсутствующих значений (правило 18).

Особенности представления отчетов (R):

- гибкость формирования отчетов (правило 11);
- постоянная производительность отчетов (правило 4);
- автоматическая настройка физического уровня (измененное оригинальное правило 7).

Управление измерениями (D):

- универсальность измерений (правило 6);
- неограниченное число измерений и уровней агрегации (правило 12);
- неограниченные операции между размерностями (правило 9).

5.4 Тест FASMI

Помимо рассмотренных ранее особенностей известен также тест FASMI (Fast of Analysis Shared Multidimensional Information), созданный в 1995 г. Н. Пендсом и Р. Критом на основе анализа правил Кодда. В данном контексте акцент сделан на скорость обработки, многопользовательский доступ, релевантность информации, наличие средств статистического анализа и многомерность, т. е. на представление анализируемых фактов как функций от большого числа их характеризующих параметров. Таким образом, Пендс и Крит определили OLAP следующими пятью ключевыми словами: FAST (Быстрый), ANALYSIS (Анализ), SHARED (Разделяемой), MULTIDIMENSIONAL (Многомерной), INFORMATION (Информации). Изложим эти пять ключевых представлений более подробно [1].

FAST (Быстрый). OLAP-система должна обеспечивать выдачу, большинства ответов пользователям в пределах приблизительно 5 секунд. При этом самые простые запросы обрабатываются в течение 1 секунды, и очень немногие — более 20 секунд. Недавнее исследование в Нидерландах показало, что конечные пользователи воспринимают процесс неудачным, если результаты не получены по истечении 30 секунд. Они могут нажать комбинацию клавиш <Alt>+<Ctrl>+<Del>, если система не предупредит их, что обработка данных требует большего времени. Даже если система предупредит, что процесс будет длиться существенно дольше, пользователи могут отвлечься и потерять мысль, при этом качество анализа страдает. Такой скорости нелегко достигнуть с большим количеством данных, особенно если требуются специальные вычисления "на лету". Для достижения данной цели используются

разные методы, включая применение аппаратных платформ с большей производительностью.

ANALYSIS (Анализ). OLAP-система должна справляться с любым логическим и статистическим анализом, характерным для данного приложения, и обеспечивать его сохранение в виде, доступном для конечного пользователя. Естественно, система должна позволять пользователю определять новые специальные вычисления как часть анализа и формировать отчеты любым желаемым способом без необходимости программирования. Все требуемые функциональные возможности анализа должны обеспечиваться понятным для конечных пользователей способом.

SHARED (Разделяемой). OLAP-система должна выполнять все требования защиты конфиденциальности (возможно, до уровня ячейки хранения данных). Если для записи необходим множественный доступ, обеспечивается блокировка модификаций на соответствующем уровне. Обработка множественных модификаций должна выполняться своевременно и безопасным способом.

MULTIDIMENSIONAL (Многомерной). OLAP-система должна обеспечить многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий, обеспечивающих наиболее логичный способ анализа. Это требование не устанавливает минимальное число измерений, которые должны быть обработаны, поскольку этот показатель зависит от приложения. Оно также не определяет используемую технологию БД, если пользователь действительно получает многомерное концептуальное представление информации.

INFORMATION (Информации). OLAP-система должна обеспечивать получение необходимой информации в условиях реального приложения. Мощность различных систем измеряется не объемом хранимой информации, а количеством входных данных, которые они могут обработать. В этом смысле мощность продуктов весьма различна. Большие OLAP-системы могут оперировать по крайней мере в 1 000 раз большим количеством данных по

сравнению с простыми версиями OLAP-систем. При этом следует учитывать множество факторов, включая дублирование данных, требуемую оперативную память, использование дискового пространства, эксплуатационные показатели, интеграцию с информационными хранилищами и т. п.

5.5. Архитектура OLAP-систем

OLAP-система включает в себя два основных компонента:

- OLAP-сервер — обеспечивает хранение данных, выполнение над ними необходимых операций и формирование многомерной модели на концептуальном уровне. В настоящее время OLAP-серверы объединяют с ХД или ВД;
- OLAP-клиент — представляет пользователю интерфейс к многомерной модели данных, обеспечивая его возможностью удобно манипулировать данными для выполнения задач анализа.

OLAP-серверы скрывают от конечного пользователя способ реализации многомерной модели. Они формируют гиперкуб, с которым пользователи посредством OLAP-клиента выполняют все необходимые манипуляции, анализируя данные. Между тем способ реализации очень важен, т. к. от него зависят такие характеристики, как производительность и занимаемые ресурсы. Выделяют три основных способа реализации:

MOLAP — многомерный (multivariate) OLAP. Для реализации многомерной модели используют многомерные БД;

ROLAP — реляционный (relational) OLAP. Для реализации многомерной модели используют реляционные БД;

HOLAP — гибридный (hybrid) OLAP. Для реализации многомерной модели используют и многомерные, и реляционные БД.

Достаточно часто в литературе по OLAP-системам можно встретить аббревиатуры DOLAP и JOLAP:

DOLAP — настольный (desktop) OLAP. Является недорогой и простой в использовании OLAP-системой, предназначенной для локального анализа и

представления данных, которые загружаются из реляционной или многомерной БД на машину клиента;

JOLAP — новая, основанная на Java коллективная OLAP API-инициатива, предназначенная для создания и управления данными и метаданными на серверах OLAP. Основной разработчик – Hyperion Solutions. Другими членами группы, определяющей предложенный API, являются компании IBM, Oracle и некоторые другие.

MOLAP. MOLAP-серверы используют для хранения и управления данными многомерных БД. При этом данные хранятся в виде упорядоченных многомерных массивов. Такие массивы подразделяются на гиперкубы и поликубы. В гиперкубе все хранимые в БД ячейки имеют одинаковую мерность, т. е. находятся в максимально полном базисе измерений. В поликубе каждая ячейка хранится с собственным набором измерений, и все связанные с этим сложности обработки перекладываются на внутренние механизмы системы.

Очевидно, что физически данные, представленные в многомерном виде, хранятся в "плоских" файлах. При этом куб представляется в виде одной плоской таблицы, в которую построчно вписываются все комбинации членов всех измерений с соответствующими им значениями мер.

Можно выделить следующие преимущества использования многомерных БД в OLAP-системах:

- поиск и выборка данных осуществляются значительно быстрее, чем при многомерном концептуальном взгляде на реляционную БД, т. к. многомерная база данных денормализована и содержит заранее агрегированные показатели, обеспечивая оптимизированный доступ к запрашиваемым ячейкам и не требуя дополнительных преобразований при переходе от множества связанных таблиц к многомерной модели;
- многомерные БД легко справляются с задачами включения в информационную модель разнообразных встроенных функций, тогда как объективно существующие ограничения языка SQL делают

выполнение этих задач на основе реляционных БД достаточно сложным, а иногда и невозможным.

С другой стороны, имеются также существенные недостатки многомерных БД:

- за счет денормализации и предварительно выполненной агрегации объем данных в многомерной БД, как правило, соответствует (по оценке Кодда) в 2,5 - 100 раз меньшему объему исходных детализированных данных;
- в подавляющем большинстве случаев информационный гиперкуб является сильно разреженным, а поскольку данные хранятся в упорядоченном виде, неопределенные значения удается удалить только за счет выбора оптимального порядка сортировки, позволяющего организовать данные в максимально большие непрерывные группы. Но даже в этом случае проблема решается только частично. Кроме того, оптимальный с точки зрения хранения разреженных данных порядок сортировки, скорее всего, не будет совпадать с порядком, который чаще других используется в запросах. Поэтому в реальных системах приходится искать компромисс между быстродействием и избыточностью дискового пространства, занятого базой данных;
- многомерные БД чувствительны к изменениям в многомерной модели. Так при добавлении нового измерения приходится изменять структуру всей БД, что влечет за собой большие затраты времени.

На основании анализа достоинств и недостатков многомерных БД можно выделить следующие условия, при которых их использование является эффективным:

- объем исходных данных для анализа не слишком велик (не более нескольких гигабайт), т. е. уровень агрегации данных достаточно высок;
- набор информационных измерений стабилен;

- время ответа системы на нерегламентированные запросы является наиболее критичным параметром;
- требуется широкое использование сложных встроенных функций для выполнения кроссмерных вычислений над ячейками гиперкуба, в том числе необходима возможность написания пользовательских функций.

ROLAP. ROLAP-серверы используют реляционные БД. По словам Кодда, "реляционные БД были, есть и будут наиболее подходящей технологией для хранения данных. Необходимость существует не в новой технологии БД, а скорее в средствах анализа, дополняющих функции существующих СУБД, и достаточно гибких, чтобы предусмотреть и автоматизировать разные виды интеллектуального анализа, присущие OLAP".

В настоящее время распространены две основные схемы реализации многомерного представления данных с помощью реляционных таблиц: схема "звезда" (рис. 5.5) и схема "снежинка" (рис. 5.6).

Основными составляющими схемы "звезда" (Star Schema) являются денормализованная таблица фактов (Fact Table) и множество таблиц измерений (Dimension Tables).

Таблица фактов (фактологическая таблица), как правило, 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 является телефонный звонок или снятие денег со счета с помощью банкомата);
- факты, связанные с "моментальными снимками" (Snapshot facts). Они основаны на состоянии объекта (например, банковского счета) в определенные моменты времени (например, на конец дня или месяца). Типичными примерами таких фактов является объем продаж за день или дневная выручка;

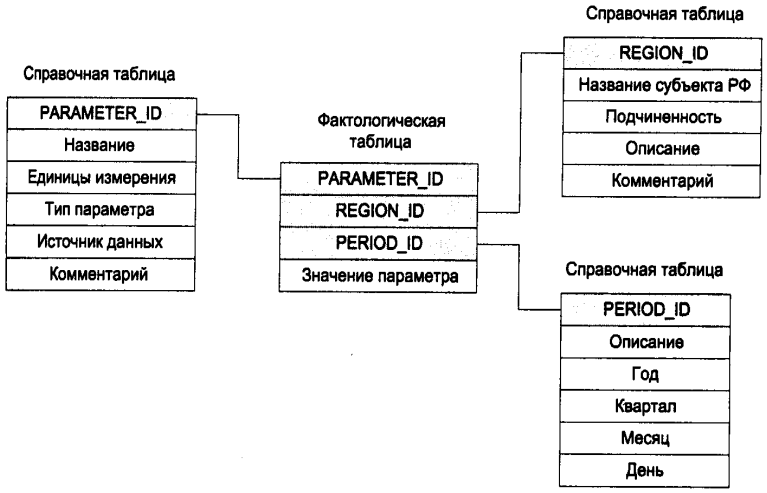

Рис. 5.5. Пример базы данных со схемой "звезда".

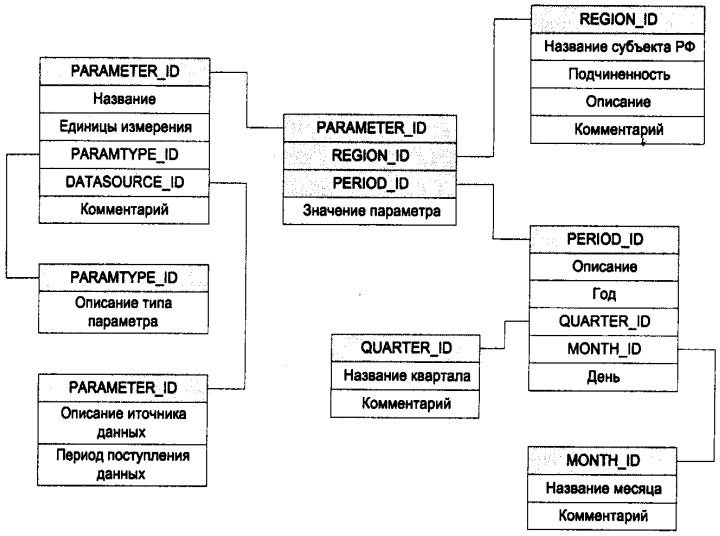

Рис. 5.6. Пример базы данных со схемой "снежинка".

- факты, связанные с элементами документа (Line-item facts). Они основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, о количестве, цене, проценте скидки);
- факты, связанные с событиями или состоянием объекта (Event or state facts). Они представляют возникновение события без подробностей о нем (например, просто факт продажи или факт отсутствия таковой без иных подробностей).

Таблица фактов, как правило, содержит уникальный составной ключ, объединяющий первичные ключи таблиц измерений. При этом как ключевые, так и некоторые не ключевые поля должны соответствовать измерениям гиперкуба. Помимо этого таблица фактов содержит одно или несколько числовых полей, на основании которых в дальнейшем будут получены агрегатные данные.

Для многомерного анализа пригодны таблицы фактов, содержащие как можно более подробные данные, т. е. соответствующие членам нижних уровней иерархии соответствующих измерений. 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, используемые в дальнейшем для построения иерархий в измерениях куба, содержатся в таблицах измерений.

Таблицы измерений содержат неизменяемые или 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 измерение, соответствующее таблице, содержит иерархию, то такая таблица также может содержать поля, указывающие на "родителя" данного члена в этой иерархии. Каждая таблица измерений должна находиться в отношении "один-ко-многим" с таблицей фактов.

Скорость роста таблиц измерений должна быть незначительной по сравнению со скоростью роста таблицы фактов. Например, новая запись в таблицу измерений, характеризующую товары, добавляется только при появлении нового товара, не продававшегося ранее.

В сложных задачах с иерархическими измерениями имеет смысл обратиться к расширенной схеме "снежинка" (Snowflake Schema). В этих случаях от-

дельные таблицы фактов создаются для возможных сочетаний уровней обобщения различных измерений (см. рис. 5.6). Это позволяет добиться лучшей производительности, но часто приводит к избыточности данных и к значительным усложнениям в структуре базы данных, в которой оказывается огромное количество таблиц фактов.

Увеличение числа таблиц фактов в базе данных определяется не только множественностью уровней различных измерений, но и тем обстоятельством, что в общем случае факты имеют разные множества измерений. При абстрагировании от отдельных измерений пользователь должен получать проекцию максимально полного гиперкуба, причем далеко не всегда значения показателей в ней должны являться результатом элементарного суммирования. Таким образом, при большом числе независимых измерений необходимо поддерживать множество таблиц фактов, соответствующих каждому возможному сочетанию выбранных в запросе измерений, что также приводит к неэкономному использованию внешней памяти, увеличению времени загрузки данных в БД схемы "звезды" из внешних источников и сложностям администрирования.

Использование реляционных БД в OLAP-системах имеет следующие достоинства:

- в большинстве случаев корпоративные хранилища данных реализуются средствами реляционных СУБД, и инструменты ROLAP позволяют производить анализ непосредственно над ними. При этом размер хранилища не является таким критичным параметром, как в случае MOLAP;
- в случае переменной размерности задачи, когда изменения в структуру измерений приходится вносить достаточно часто, ROLAP-системы с динамическим представлением размерности являются оптимальным решением, т. к. в них такие модификации не требуют физической реорганизации БД;

- реляционные СУБД обеспечивают значительно более высокий уровень защиты данных и хорошие возможности разграничения прав доступа.

Главный недостаток ROLAP по сравнению с многомерными СУБД — меньшая производительность. Для обеспечения производительности, сравнимой с MOLAP, реляционные системы требуют тщательной проработки схемы базы данных и настройки индексов, т.е. больших усилий со стороны администраторов БД. Только при использовании схем типа "звезда" производительность; хорошо настроенных реляционных систем может быть приближена к производительности систем на основе многомерных баз данных.

HOLAP. HOLAP-серверы используют гибридную архитектуру, которая объединяет технологии ROLAP и MOLAP. В отличие от MOLAP, которая работает лучше, когда данные более-менее плотные, серверы ROLAP показывают лучшие параметры в тех случаях, когда данные сильно разрежены. Серверы HOLAP применяют подход ROLAP для разреженных областей многомерного пространства и подход MOLAP для плотных областей. Серверы HOLAP разделяют запрос на несколько подзапросов, направляют их к соответствующим фрагментам данных, комбинируют результаты, а затем предоставляют результат пользователю.

Контрольные вопросы

8. На какие недостатки реляционной модели данных для проведения анализа указал Е. Кодд?.
9. Что такое «многомерное концептуальное представление»?
10. Приведите пример реализации операции «срез».
11. Раскройте сущность процесса управления транзакциями.
12. Что такое OLAP?
13. Какое из 12 правил Кодда, определяющих OLAP, является самым значимым?
14. Какова цель теста FASMI?

15. Какие основные операции присущи многомерной базе данных?
16. Какие основные схемы реализации многомерного представления данных используются в ROLAP?.
17. В чем заключается главный недостаток ROLAP по сравнению с многомерными СУБД?

6. Интеллектуальный анализ данных

6.1. Добыча данных — Data Mining

OLAP-системы предоставляют аналитику средства проверки гипотез при анализе данных. При этом основной задачей аналитика является генерация гипотез. Он решает ее, основываясь на своих знаниях и опыте. Однако знания есть не только у человека, но и в накопленных данных, которые подвергаются анализу. Такие знания часто называют "скрытыми", т. к. они содержатся в гигабайтах и терабайтах информации, которые человек не в состоянии исследовать самостоятельно. В связи с этим существует высокая вероятность пропустить гипотезы, которые могут принести значительную выгоду.

Очевидно, что для обнаружения скрытых знаний необходимо применять специальные методы автоматического анализа, при помощи которых приходится практически добывать знания из "завалов" информации. За этим направлением прочно закрепился термин добыча данных или Data Mining. Классическое определение этого термина дал в 1996 г. один из основателей этого направления — Григорий Пятецкий-Шапиро [1].

Data Mining — исследование и обнаружение "машиной" (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны, доступны для интерпретации человеком.

Рассмотрим свойства обнаруживаемых знаний, данные в определении, более подробно.

Знания должны быть новые, ранее неизвестные. Затраченные усилия на открытие знаний, которые уже известны пользователю, не окупаются. Поэтому ценность представляют именно новые, ранее неизвестные знания.

Знания должны быть нетривиальны. Результаты анализа должны отражать неочевидные, неожиданные закономерности в данных, составляющие так называемые скрытые знания. Результаты, которые могли бы быть получены

более простыми способами (например, визуальным просмотром), не оправдывают привлечение мощных методов Data Mining.

Знания должны быть практически полезны. Найденные знания должны быть применимы, в том числе и на новых данных, с достаточно высокой степенью достоверности. Полезность заключается в том, чтобы эти знания могли принести определенную выгоду при их применении.

Знания должны быть доступны для понимания человеку. Найденные закономерности должны быть логически объяснимы, в противном случае существует вероятность, что они являются случайными. Кроме того, обнаруженные знания должны быть представлены в понятном для человека виде.

В Data Mining для представления полученных знаний служат модели. Виды моделей зависят от методов их создания. Наиболее распространенными являются: правила, деревья решений, кластеры и математические функции.

6.2. Задачи Data Mining

Методы Data Mining помогают решить многие задачи, с которыми сталкивается аналитик. Из них основными являются: классификация, регрессия, поиск ассоциативных правил и кластеризация. Далее приведено краткое описание основных задач анализа данных.

Задача классификации сводится к определению класса объекта по его характеристикам. В этой задаче множество классов, к которым может быть отнесен объект, известно заранее.

Задача регрессии подобно задаче классификации позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значением параметра является не конечное множество классов, а множество действительных чисел.

Поиск ассоциативных правил имеет целью является нахождение частых зависимостей (или ассоциаций) между объектами или событиями. Найденные зависимости представляются в виде правил и могут быть использованы как для

лучшего понимания природы анализируемых данных, так и для предсказания появления событий.

Задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик во всем множестве анализируемых данных. Решение этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет сократить их число, а следовательно, и облегчить анализ.

Перечисленные задачи по назначению делятся на описательные и предсказательные.

Описательные (descriptive) задачи уделяют внимание улучшению понимания анализируемых данных. Ключевой момент в таких моделях — легкость и прозрачность результатов для восприятия человеком. Возможно, обнаруженные закономерности будут специфической чертой именно конкретных исследуемых данных и больше нигде не встретятся, но это все равно может быть полезно и потому должно быть известно. К такому виду задач относятся кластеризация и поиск ассоциативных правил.

Решение ***предсказательных (predictive) задач*** разбивается на два этапа. На первом этапе, на основании набора данных с известными результатами строится модель. На втором этапе она используется для предсказания результатов на основании новых наборов данных. При этом, естественно, требуется, чтобы построенные модели работали максимально точно. К данному виду задач относят задачи классификации и регрессии. Сюда можно отнести и задачу поиска ассоциативных правил, если результаты ее решения могут быть использованы для предсказания появления некоторых событий.

По способам решения задачи разделяют на *supervised learning* (обучение с учителем) и *unsupervised learning* (обучение без учителя). Такое название произошло от термина Machine Learning (машинное обучение), часто используемого в англоязычной литературе и обозначающего все технологии Data Mining [1].

В случае *supervised learning* задача анализа данных решается в несколько этапов. Сначала с помощью какого-либо алгоритма Data Mining строится модель анализируемых данных — классификатор. Затем классификатор подвергается обучению. Другими словами, проверяется качество его работы, и, если оно неудовлетворительное, происходит дополнительное обучение классификатора. Так продолжается до тех пор, пока не будет достигнут требуемый уровень качества или не станет ясно, что выбранный алгоритм не работает корректно с данными, либо же сами данные не имеют структуры, которую можно выявить. К этому типу задач относят задачи классификации и регрессии.

Unsupervised learning объединяет задачи, выявляющие описательные модели, например закономерности в покупках, совершаемых клиентами большого магазина. Очевидно, что если эти закономерности есть, то модель должна их представить и неуместно говорить об ее обучении. Достоинством таких задач является возможность их решения без каких-либо предварительных знаний об анализируемых данных. К этим задачам относятся кластеризация и поиск ассоциативных правил.

Задача классификации и регрессии. При анализе часто требуется определить, к какому из известных классов относятся исследуемые объекты - классифицировать их. Например, когда человек обращается в банк за предоставлением ему кредита, банковский служащий должен принять решение: кредитоспособен ли потенциальный клиент или нет. Очевидно, что такое решение принимается на основании данных об исследуемом объекте (в данном случае — о человеке): его место работы, размер заработной платы, возраст, состав семьи и т. п. В результате анализа этой информации банковский служащий должен отнести человека к одному из двух известных классов: "кредитоспособен" и "некредитоспособен".

Другим примером задачи классификации является фильтрация электронной почты. В этом случае программа фильтрации должна классифицировать вхо-

дящее сообщение как спам (зрат — нежелательная электронная почта) или как письмо. Данное решение принимается на основании частоты появления в сообщении определенных слов (например, имени получателя, безличного обращения, слов и словосочетаний: "приобрести", "заработать", "выгодное предложение" и т. п.).

В общем случае количество классов в задачах классификации может быть более двух. Например, в задаче распознавания образа цифр таких классов может быть 10 (по количеству цифр в десятичной системе счисления). В такой задаче объектом классификации является матрица пикселов, представляющая образ распознаваемой цифры. При этом цвет каждого пиксела является характеристикой анализируемого объекта.

В Data Mining задачу классификации рассматривают как задачу определения значения одного из параметров анализируемого объекта на основании значений других параметров. Определяемый параметр часто называют зависимой переменной, а параметры, участвующие в его определении, — независимыми переменными. В рассмотренных примерах независимыми переменными являлись:

зарплата, возраст, количество детей и т. д.;

частота появления определенных слов;

значения цвета пикселов матрицы.

Зависимыми переменными в этих же примерах являлись соответственно:

кредитоспособность клиента (возможные значения этой переменной — "да" и "нет");

тип сообщения (возможные значения этой переменной — "spam" и "mail");

цифра образа (возможные значения этой переменной — 0, 1, ..., 9). Необходимо обратить внимание, что во всех рассмотренных примерах независимая переменная принимала значение из конечного множества значений: ("да", "нет"}, ("spam", "mail"}, (0, 1,..., 9}. Если значениями независимых и зависимой переменных являются действительные числа, то задача называется ***задачей регрессии***. Примером задачи регрессии может служить задача опре-

деления суммы кредита, которая может быть выдана банком клиенту. Задача классификации и регрессии решается в два этапа. На первом выделяется обучающая выборка. В нее входят объекты, для которых известны значения как независимых, так и зависимых переменных. В описанных ранее примерах такими обучающими выборками могут быть:

- информация о клиентах, которым ранее выдавались кредиты на разные суммы, и информация об их погашении;
- сообщения, классифицированные вручную как спам или как письмо;
- распознанные ранее матрицы образов цифр.

На основании обучающей выборки строится модель определения значения зависимой переменной. Ее часто называют функцией классификации или регрессии. Для получения максимально точной функции к обучающей выборке предъявляются следующие основные требования:

- количество объектов, входящих в выборку, должно быть достаточно большим. Чем больше объектов, тем точнее будет построенная на ее основе функция классификации или регрессии;
- в выборку должны входить объекты, представляющие все возможные классы в случае задачи классификации или всю область значений в случае задачи регрессии;
- для каждого класса в задаче классификации или для каждого интервала области значений в задаче регрессии выборка должна содержать достаточное количество объектов.

На втором этапе построенную модель применяют к анализируемым объектам (к объектам с неопределенным значением зависимой переменной). Задача классификации и регрессии имеет геометрическую интерпретацию. Рассмотрим ее на примере с двумя независимыми переменными, что позволит представить ее в двумерном пространстве (рис. 6.1).

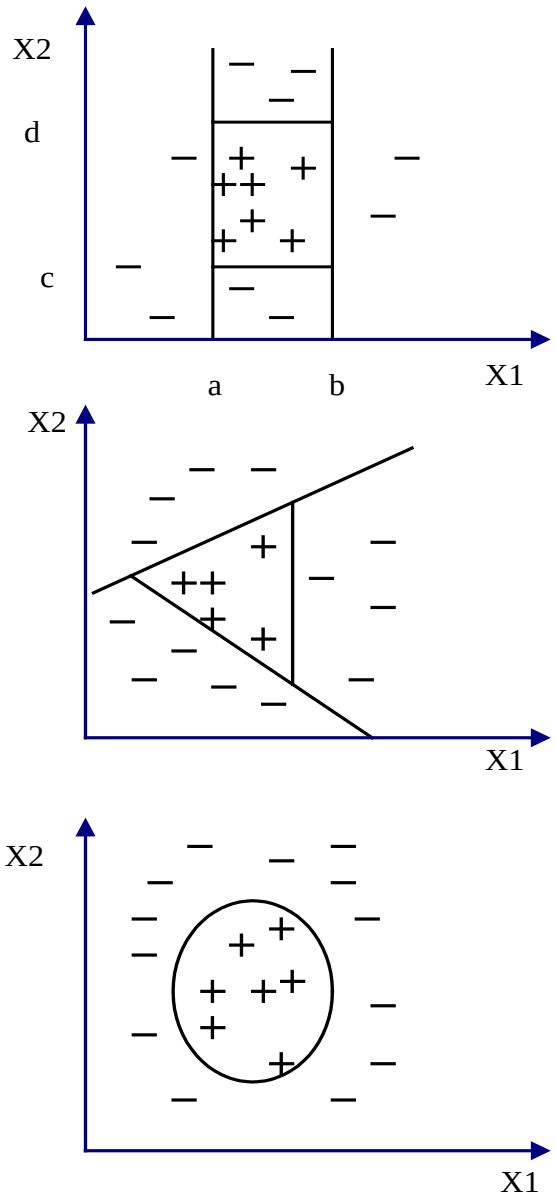

Рис. 6.1. Классификация в двумерном пространстве.

Каждому объекту ставится в соответствие точка на плоскости. Символы "+" и "-" обозначают принадлежность объекта к одному из двух классов. Очевидно, что данные имеют четко выраженную структуру: все точки класса "+" сосредоточены в центральной области. Построение классификационной функции сводится к построению поверхности, которая обводит центральную область. Она определяется как функция, имеющая значения "+" внутри обведенной области и "-" — вне ее.

Как видно из рисунка, есть несколько возможностей для построения обводящей области. Вид функции зависит от применяемого алгоритма.

Основные проблемы, с которыми сталкиваются при решении задач классификации и регрессии — это неудовлетворительное качество исходных данных, в которых встречаются как ошибочные данные, так и пропущенные значения, различные типы атрибутов — числовые и категорические, разная значимость атрибутов, а также так называемые проблемы overfitting и underfitting. Суть первой из них, заключается в том, что классификационная функция при построении "слишком хорошо" адаптируется к данным и встречающиеся в них ошибки и аномальные значения пытается интерпретировать как часть внутренней структуры данных. Очевидно, что в дальнейшем такая модель будет некорректно работать с другими данными, где характер ошибок будет несколько иной. Термином underfitting обозначают ситуацию, когда слишком велико количество ошибок при проверке классификатора на обучающем множестве. Это означает, что особых закономерностей в данных не было обнаружено, и либо их нет вообще, либо необходимо выбрать иной метод их обнаружения.

Задача поиска ассоциативных правил Поиск ассоциативных правил является одним из самых популярных приложений Data Mining. Суть задачи заключается в определении часто встречающихся наборов объектов в большом множестве таких наборов. Данная задача является частным случаем задачи классификации. Первоначально она решалась при анализе тенденций в поведении покупателей в супермаркетах. Анализу подвергались данные о совершаемых ими покупках, которые покупатели складывают в тележку (корзину). Это послужило причиной второго часто встречающегося названия — анализ рыночных корзин (Basket Analysis). При анализе этих данных интерес прежде всего представляет информация о том, какие товары покупаются вместе, в какой последовательности, какие категории потребителей какие товары предпочитают, в какие периоды времени и т. п. Такая информация позволяет более эффективно планировать закупку товаров, проведение рекламной кампании и т. д.

Например, из набора покупок, совершаемых в магазине, можно выделить следующие наборы товаров, которые покупаются вместе: {чипсы, пиво}; {вода, орехи}. Следовательно, можно сделать вывод, что если покупаются чипсы или орехи, то, как правило, покупаются и пиво или вода соответственно. Обладая такими знаниями, можно разместить эти товары рядом, объединить их в один пакет со скидкой или предпринять другие действия, стимулирующие покупателя приобрести товар.

Задача поиска ассоциативных правил актуальна не только в сфере торговли. Например, в сфере обслуживания интерес представляет информация о том, какими услугами клиенты предпочитают пользоваться в совокупности. Для получения этой информации задача решается применительно к данным об услугах, которыми пользуется один клиент в течение определенного времени (месяца, года). Это помогает определить, например, как наиболее выгодно составить пакеты услуг, предлагаемых клиенту.

В медицине анализу могут подвергаться симптомы и болезни, наблюдаемые у пациентов. В этом случае знания о том, какие сочетания болезней и симптомов встречаются наиболее часто, помогают в будущем правильно ставить диагноз.

При анализе часто вызывает интерес последовательность происходящих событий. При обнаружении закономерностей в таких последовательностях можно с некоторой долей вероятности предсказывать появление событий в будущем, что позволяет принимать более правильные решения. Такая задача является разновидностью задачи поиска ассоциативных правил и называется *сиквенциалъным анализом*.

Основным отличием задачи сиквенциального анализа от поиска ассоциативных правил является установление отношения порядка между исследуемыми наборами. Данное отношение может быть определено разными способами. При анализе последовательности событий, происходящих во времени, объектами таких наборов являются события, а отношение порядка соответствует хронологии их появления.

Сиквенциальный анализ широко используется, например, в телекоммуникационных компаниях, для анализа данных об авариях на различных узлах сети. Информация о последовательности совершения аварий может помочь в обнаружении неполадок и предупреждении новых аварий. Например, если известна последовательность сбоев: {e5, e2, e7, e13, e6, e1, …}, где ei — код сбоя, то на основании факта появления сбоя e2 можно сделать вывод о скором появлении сбоя e7. Зная это, можно предпринять профилактические меры, устраняющие причины возникновения сбоя. Если дополнительно обладать и знаниями о времени между сбоями, то можно предсказать не только факт его появления, но и время, что часто не менее важно.

Задача кластеризации Задача кластеризации состоит в разделении исследуемого множества объектов на группы "похожих" объектов, называемых *кластерами* (cluster). Слово cluster переводится с английского как сгусток, пучок, группа. Родственные понятия, используемые в литературе, — класс, таксон, сгущение. Часто решение задачи разбиения множества элементов на кластеры называют кластерным анализом.

Кластеризация может применяться практически в любой области, где необходимо исследование экспериментальных или статистических данных. Рассмотрим пример из области маркетинга, в котором данная задача называется сегментацией.

Концептуально сегментирование основано на предпосылке, что все потребители разные. У них разные потребности, разные требования к товару, они ведут себя по-разному: в процессе выбора товара, в процессе приобретения товара, в процессе использования товара, в процессе формирования реакции на товар. В связи с этим необходимо по-разному подходить к работе с потребителями: предлагать им различные по своим характеристикам товары, по-разному продвигать и продавать товары. Для того чтобы определить, чем отличаются потребители друг от друга и как эти отличия отражаются на требованиях к товару, и производится сегментирование потребителей.

В маркетинге критериями (характеристиками) сегментации являются: географическое местоположение, социально-демографические характеристики, мотивы совершения покупки и т. п.

На основании результатов сегментации маркетолог может определить, например, такие характеристики сегментов рынка, как реальная и потенциальная емкость сегмента, группы потребителей, чьи потребности не удовлетворяются в полной мере ни одним производителем, работающим на данном сегменте рынка, и т. п. На основании этих параметров маркетолог может сделать вывод о привлекательности работы фирмы в каждом из выделенных сегментов рынка.

Для научных исследований изучение результатов кластеризации, а именно выяснение причин, по которым объекты объединяются в группы, способно открыть новые перспективные направления. Традиционным примером, который обычно приводят для этого случая, является периодическая таблица элементов. В 1869 г. Дмитрий Менделеев разделил 60 известных в то время элементов на кластеры или периоды. Элементы, попавшие в одну группу, обладали схожими характеристиками. Изучение причин, по которым элементы разбивались на явно выраженные кластеры, в значительной степени определило приоритеты научных изысканий на годы вперед. Но лишь спустя 50 лет квантовая физика дала убедительные объяснения периодической системы. Кластеризация отличается от классификации тем, что для проведения анализа не требуется иметь выделенную зависимую переменную, поэтому она относится к классу unsupervised learning. Эта задача решается на начальных этапах исследования, когда о данных мало что известно. Ее решение помогает лучше понять данные, и с этой точки зрения задача кластеризации является описательной.

Для задачи кластеризации характерно отсутствие каких-либо различий как между переменными, так и между объектами. Напротив, ищутся группы наиболее близких, похожих объектов. Методы автоматического разбиения на кластеры редко используются сами по себе, а только для получения групп схожих объектов. После определения кластеров используются другие методы

Data Mining, чтобы попытаться установить, что означает такое разбиение, чем оно вызвано.

Кластерный анализ позволяет рассматривать достаточно большой объем' информации и резко сокращать, сжимать большие массивы информации, делать их компактными и наглядными.

Отметим ряд особенностей, присущих задаче кластеризации. Во-первых, решение сильно зависит от природы объектов данных (и их атрибутов). Так, с одной стороны, это могут быть однозначно определенные, количественно очерченные объекты, а с другой — объекты, имеющие вероятностное или нечеткое описание.

Во-вторых, решение в значительной степени зависит и от представления кластеров и предполагаемых отношений объектов данных и кластеров. Так, необходимо учитывать такие свойства, как возможность/невозможность принадлежности объектов к нескольким кластерам. Необходимо определение самого понятия принадлежности кластеру: однозначная (принадлежит/не принадлежит), вероятностная (вероятность принадлежности), нечеткая (степень принадлежности).

6.3. Практическое применение Data Mining

Интернет-технологии. В системах электронного бизнеса, где особую важность имеют вопросы привлечения и удержания клиентов, технологии Data Mining часто применяются для построения рекомендательных систем интернет-магазинов и для решения проблемы персонализации посетителей Web-сайтов. Рекомендации товаров и услуг, построенные на основе закономерностей в покупках клиентов, обладают огромной убеждающей силой. Статистика показывает, что почти каждый посетитель магазина Amazon не упускает возможности посмотреть на то, что же купили "Customers who bought this book also boaght: …" ("Те, кто купил эту книгу, также купили ..."). Персонализация клиентов или, другими словами, автоматическое распознание принадлежности клиента к определенной целевой аудитории позволяет компании проводить

более гибкую маркетинговую политику. Поскольку в электронной коммерции деньги и платежные системы тоже электронные, то важной задачей становится обеспечение безопасности при операциях с пластиковыми карточками. Data Mining позволяет обнаруживать случаи мошенничества (fraud detection). В области электронной коммерции также остаются справедливыми все методологии Data Mining, разработанные для обычного маркетинга. С другой стороны, эта область тесно связана с понятием Web Mining.

Специфика Web Mining заключается в применении традиционных технологий Data Mining для анализа крайне неоднородной, распределенной и значительной по объему информации, содержащейся на Web-узлах. Здесь можно выделить два направления: Web Content Mining и Web Usage Mining. В первом случае речь идет об автоматическом поиске и извлечении качественной информации из перегруженных "информационным шумом" источников Интернета, а также о всевозможных средствах автоматической классификации и аннотировании документов. Данное направление также называют Text Mining. Web Usage Mining направлен на обнаружение закономерностей в поведении пользователей конкретного Web-узла (группы узлов), в частности на то, какие страницы в какой временной последовательности и какими группами пользователей запрашиваются.

Торговля. Для успешного продвижения товаров всегда важно знать, что и как продается, а также кто является потребителем. Исчерпывающий ответ на первый вопрос дают такие средства Data Mining как анализ рыночных корзин и сиквенциальный анализ. Зная связи между покупками и временные закономерности, можно оптимальным образом регулировать предложение. С другой стороны, маркетинг имеет возможность непосредственно управлять спросом, но для этого необходимо знать как можно больше о потребителях — целевой аудитории маркетинга. Data Mining позволяет решать задачи выделения групп потребителей со схожими стереотипами поведения, т. е. сегментировать рынок. Для этого можно применять такие технологии Data Mining как кластеризация и классификация.

Сиквенциальный анализ помогает торговым предприятиям принимать решения о создании товарных запасов. Он дает ответы на вопросьг типа "Если сегодня покупатель приобрел видеокамеру, то через какое время он вероятнее всего купит новые батарейки и пленку?".

Медицина. В медицинских и биологических исследованиях, равно как и в практической медицине, спектр решаемых задач настолько широк, что возможно использование любых методологий Data Mining. Примером может служить построение диагностической системы или исследование эффективности хирургического вмешательства.

Известно много экспертных систем для постановки медицинских диагнозов. Они построены главным образом на основе правил, описывающих сочетания различных симптомов отдельных заболеваний. С помощью таких правил узнают не только, чем болен пациент, но и как нужно его лечить. Правила помогают выбирать средства медикаментозного воздействия, определять показания/противопоказания, ориентироваться в лечебных процедурах, создавать условия наиболее эффективного лечения, предсказывать исходы назначенного курса лечения и т. п. Технологии Data Mining позволяют обнаруживать в медицинских данных шаблоны, составляющие основу указанных правил.

Одним из наиболее передовых направлений медицины является биоинформатика — область науки, разрабатывающая и применяющая вычислительные алгоритмы для анализа и систематизации генетической информации с целью выяснения структуры и функции макромолекул, последующего использования этих знаний для объяснения различных биологических явлений и создания новых лекарственных препаратов (Drug Design). Объектом исследования биоинформатики являются огромные объемы информации о последовательностях ДНК и первичной структуре белков, появившиеся в результате изучения структуры геномов микроорганизмов, млекопитающих и человека. Абстрагируясь от конкретного содержания этой информации, ее можно рассматривать как набор генетических текстов, состоящих из протяженных символьных последовательностей. Выявление

структурных закономерностей в таких последовательностях входит в число задач, эффективно решаемых средствами Data Mining, например, с помощью сиквенциального и ассоциативного анализа. Основная область практического применения биоинформатики — разработка лекарств нового поколения, которые полностью преобразят современную медицину.

Банковское дело. Классическим примером применения Data Mining на практике является решение проблемы о возможной некредитоспособности клиентов банка. Этот вопрос, тревожащий любого сотрудника кредитного отдела банка, можно разрешить и интуитивно. Если образ клиента в сознании банковского служащего соответствует его представлению о кредитоспособном клиенте, то кредит выдавать можно, иначе — отказать. По схожей схеме, но более продуктивно и полностью автоматически, работают установленные в тысячах американских банков системы поддержки принятия решений (Decision System Support) со встроенной функциональностью Data Mining. Лишенные субъективной предвзятости, они опираются в своей работе только на историческую базу данных банка, где записывается детальная информация о каждом клиенте и, в конечном итоге, факт его кредитоспособности. Классификационные алгоритмы Data Mining обрабатывают эти данные, а полученные результаты используются далее для принятия решений.

Анализ кредитного риска заключается, прежде всего, в оценке кредитоспособности заемщика. Эта задача решается на основе анализа накопленной информации, т. е. кредитной истории "старых" клиентов. С помощью инструментов Data Mining (деревья решений, кластерный анализ, нейронные сети и др.) банк может получить профили добросовестных и неблагонадежных заемщиков. Кроме того, можно классифицировать заемщика по группам риска, а значит, не только решить вопрос о возможности кредитования, но и установить лимит кредита, проценты по нему и срок возврата.

Мошенничество с кредитными карточками представляет собой серьезную проблему, т. к. убытки от него измеряются миллионами долларов ежегодно, а рост количества мошеннических операций составляет, по оценкам экспертов, от

15 до 25 % ежегодно. В борьбе с мошенничеством технология Data Mining использует стереотипы подозрительных операций, созданные в результате анализа огромного количества транзакций —- как законных, так и неправомерных. Исследуется не только отдельно взятая операция, но и совокупность последовательных во времени транзакций. Кроме того, алгоритмы и модели (например, нейронные сети), имеющиеся в составе продуктов Data Mining. способны тестироваться и самообучаться. При попытке совершения подозрительной операции средства интеллектуального анализа данных оперативно выдают предупреждение об этом, что позволяет банку предотвратить незаконные действия, а не устранять их последствия. Использование технологии Data Mining позволяет сократить число нарушений на 20—30 %.

Data Mining может применяться практически везде, где возникает задача автоматического анализа данных. В качестве примера приведем такие популярные направления, как анализ и последующая фильтрация спама, а также разработка так называемых виртуальных собеседников. Последние сейчас являются не более чем экзотическим дополнением к интерфейсу некоторых сайтов, но предполагается, что в будущем они могут заменить собой call-центры компаний.

6.4. Модели Data Mining

Цель технологии Data Mining — нахождение в данных таких моделей, которые не могут быть найдены обычными методами. Существуют два вида моделей: предсказательные и описательные.

Предсказательные (predictive) модели строятся на основании набора данных с известными результатами. Они используются для предсказания результатов на основании других наборов данных. При этом, естественно, требуется, чтобы модель работала максимально точно, была статистически значима и оправданна и т. д. К таким моделям относятся следующие:

- модели классификации — описывают правила или набор правил, в соответствии с которыми можно отнести описание любого нового объекта к одному из классов. Такие правила строятся на основании информации о существующих объектах путем разбиения их на классы;
- модели последовательностей — описывают функции, позволяющие прогнозировать изменение непрерывных числовых параметров. Они строятся на основании данных об изменении некоторого параметра за прошедший период времени.

Описательные (descriptive) модели уделяют внимание сути зависимостей в наборе данных, взаимному влиянию различных факторов, т. е. построению эмпирических моделей различных систем. Ключевой момент в таких моделях — легкость и прозрачность для восприятия человеком. Возможно, обнаруженные закономерности будут специфической чертой именно конкретных исследуемых данных и больше нигде не встретятся, но это все равно может быть полезно, и потому должно быть известно. К таким моделям относятся следующие виды:

регрессионные модели — описывают функциональные зависимости между зависимыми и независимыми показателями и переменными в понятной человеку форме. Необходимо заметить, что такие модели описывают функциональную зависимость не только между непрерывными числовыми параметрами, но и между категориальными параметрами;

модели кластеров — описывают группы (кластеры), на которые можно разделить объекты, данные о которых подвергаются анализу. Группируются объекты (наблюдения, события) на основе данных (свойств), описывающих сущность объектов. Объекты внутри кластера должны быть "похожими" друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем сильнее "похожи" объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация;

модели исключений — описывают исключительные ситуации в записях (например, отдельных пациентов), которые резко отличаются чем-либо от

основного множества записей (группы больных). Знание исключений может быть использовано двояким образом. Возможно, эти записи представляют собой случайный сбой, например ошибки операторов, вводивших данные в компьютер. Характерный случай: если оператор, ошибаясь, ставит десятичную точку не в том месте, то такая ошибка сразу дает резкий "всплеск" на порядок. Подобную "шумовую" случайную составляющую имеет смысл отбросить, исключить из дальнейших исследований, поскольку большинство методов, которые будут рассмотрены в данной главе, очень чувствительно к наличию "выбросов" — резко отличающихся точек, редких, нетипичных случаев. С другой стороны, отдельные, исключительные записи могут представлять самостоятельный интерес для исследования, т. к. они могут указывать на некоторые редкие, но важные аномальные заболевания. Даже сама идентификация этих записей, не говоря об их последующем анализе и детальном рассмотрении, может оказаться очень полезной для понимания сущности изучаемых объектов или явлений;

итоговые модели — выявление ограничений на данные анализируемого массива. Например, при изучении выборки данных по пациентам не старше 30 лет, перенесшим инфаркт миокарда, обнаруживается, что все пациенты, описанные в этой выборке, либо курят более 5 пачек сигарет в день, либо имеют вес не ниже 95 кг. Подобные ограничения важны для понимания данных массива, по сути дела это новое знание, извлеченное в результате анализа. Таким образом, построение итоговых моделей заключается в нахождении каких-либо фактов, которые верны для всех или почти всех записей в изучаемой выборке данных, но которые достаточно редко встречались бы во всем мыслимом многообразии записей такого же формата и, например, характеризовались бы теми же распределениями значений полей. Если взять для сравнения информацию по всем пациентам, то процент либо сильно курящих, либо чрезмерно тучных людей будет весьма невелик. Можно сказать, что решается как бы неявная задача классификации, хотя фактически задан только один класс, представленный имеющимися данными;

ассоциативные модели — выявление закономерностей между связанными событиями. Примером такой закономерности служит правило, указывающее, что из события X следует событие У. Такие правила называются ассоциативными.

Для построения рассмотренных моделей используются различные методы и алгоритмы Data Mining. Ввиду того, что технология Data Mining развивалась и развивается на стыке таких дисциплин, как статистика, теория информации, машинное обучение и теория баз данных, вполне закономерно, что большинство алгоритмов и методов Data Mining были разработаны на основе различных технологий и концепций. Далее рассмотрим технологии, наиболее часто реализуемые методами Data Mining.

6.5. Методы Data Mining

Базовые методы. К базовым методам Data Mining принято относить, прежде всего, алгоритмы, основанные на переборе. Простой перебор всех исследуемых объектов требует $O(2^N)$ операций, где N — количество объектов. Следовательно, с увеличением количества данных объем вычислений растет экспоненциально, что при большом объеме делает решение любой задачи таким методом практически невозможным.

Для сокращения вычислительной сложности в таких алгоритмах, как правило, используют разного вида эвристики, приводящие к сокращению перебора. Оптимизация подобных алгоритмов сводится к приведению зависимости количества операций от количества исследуемых данных к функции линейного вида. В то же время, зависимость от количества атрибутов, как правило, остается экспоненциальной. При условии, что их немного (в подавляющем большинстве случаев их значительно меньше, чем данных), такая зависимость является приемлемой.

Основным достоинством данных алгоритмов является их простота, как с точки зрения понимания, так и реализации. К недостаткам можно отнести отсутствие формальной теории, на основании которой строятся такие

алгоритмы, а следовательно, и сложности, связанные с их исследованием и развитием. К базовым методам Data Mining можно отнести также и подходы, использующие элементы теории статистики. В связи с тем, что Data Mining является развитием статистики, таких методов достаточно много. Их основная идея сводится к корреляционному, регрессионному, и другим видам статистического анализа. Главным недостатком является усреднение значений, что приводит к потере информативности данных. Это в свою очередь приводит к уменьшению количества добываемых знаний.

Нечеткая логика. Основным способом исследования задач анализа данных является их отображение на формализованный язык и последующий анализ полученной модели. Неопределенность по объему отсутствующей информации у системного аналитика можно разделить на три большие группы:

Неизвестность.

Неполнота (недостаточность, неадекватность).

Недостоверность.

Недостоверность бывает *физической* (источником ее является внешняя среда) и *лингвистической* (возникает в результате словесного обобщения и обусловливается необходимостью описания бесконечного числа ситуаций ограниченным числом слов за ограниченное время).

Выделяют два вида физической неопределенности:

1. Неточность (неточность измерений значений определенной величины, выполняемых физическими приборами).
2. Случайность (или наличие во внешней среде нескольких возможностей, каждая из которых случайным образом может стать действительностью; предполагается знание соответствующего закона распределения вероятностей).

Выделяют два вида лингвистической неопределенности:

1. Неопределенность значений слов (многозначность, расплывчатость, неясность, нечеткость). Она возникает в случае, если отображаемые одним и тем же словом объекты задачи управления различны.

2. Неоднозначность смысла фраз (выделяют синтаксическую и семантическую).

Для обработки физических неопределенностей успешно используются методы теории вероятностей и классическая теория множеств. Однако с развитием систем, использующих методы теории искусственного интеллекта, в которых требуется обрабатывать понятия и отношения естественного языка, возникла необходимость расширения множества формальных методов с целью учета лингвистической неопределенности задач.

Основной сферой применения нечеткой логики было и во многом остается управление. Не случайно основоположником теории нечетких множеств стал известный специалист в области управления Л. Заде. В исходную идею о нечеткой логике очень хорошо укладывались представления об управлении и процессах принятия решений. А поскольку подобные задачи возникают почти во всех технологических процессах, потребности в развитии данной теории и возможности ее приложения достаточно широки. С увеличением размеров и сложности системы существенно усложняется ее моделирование с помощью известных математических выражений. Это связано с увеличением числа переменных и параметров, повышением сложности измерения отдельных переменных. В результате, создание адекватной модели становится практически невозможным. Вместо этого Л. Заде предложил лингвистическую модель, которая использует не математические выражения, а слова, отражающие качество. Применение словесной модели не обеспечивает точность, аналогичную математическому моделированию, однако создание хорошей, качественной модели возможно. В этом случае предметом обсуждения становится нечеткость слов языка описания системы.

Человеку в процессе управления сложными объектами свойственно оперировать понятиями и отношениями с расплывчатыми границами. Источником расплывчатости является существование классов объектов, степень принадлежности к которым — величина, непрерывно изменяющаяся от полной принадлежности к нему до полной непринадлежности. Обычное

математическое понятие множества, основанное на бинарной характеристической функции, не позволяет формализовать такое описание.

Введение Л. Заде двух основных исходных понятий — нечеткого множества и лингвистической переменной — существенно расширило возможности формализации описаний подобных сложных систем. Такие модели стали называться лингвистическими.

Рассмотрим основные достоинства нечеткой логики, наиболее ярко проявляющиеся на примере общей задачи нечеткого управления. Если говорить кратко, то нечеткая логика позволяет удачно представить мышление человека. Очевидно, что в повседневной деятельности человек никогда не пользуется формальным моделированием на основе математических выражений, не ищет одного универсального закона, описывающего все окружающее. Он использует нечеткий естественный язык. В процессе принятия решения человек легко овладевает ситуацией, разделяя ее на события, находит решение сложных проблем, применяя для отдельных событий соответствующие, по опыту, правила принятия решений, используя при этом большое количество иногда даже противоречивых качественных критериев. Таким образом, перед человеком возникает ряд локальных моделей, описывающих свойства фрагментов объектов в определенных условиях. Крайне важным является то, что все модели обладают некой общностью и очень просты для понимания на качественном уровне. Ярким примером каркаса подобной словесной модели является конструкция "если ..., то ... ".

Теперь определим три основные особенности нечеткой логики:

1. Правила принятия решений являются условными высказываниями типа "если ... , то ... " и реализуются с помощью механизма логического вывода.
2. Вместо одного четкого обобщенного правила нечеткая логика оперирует с множеством частных правил. При этом для каждой локальной области распределенного информационного пространства, для каждой регулируемой величины, для каждой цели управления

задаются свои правила. Это позволяет отказываться от трудоемкого процесса свертки целей и получения обобщенного целевого критерия, что, в свою очередь, дает возможность оперировать даже с противоположными целями.

3. Правила в виде "если ..., то ..." позволяют решать задачи классификации в режиме диалога с оператором, что способствует повышению качества классификатора уже в процессе эксплуатации.

Таким образом, нетрудно заметить существенные общие черты нечеткой логики и мышления человека, поэтому методы управления на основе нечеткой логики можно считать во многом эвристическими. Эвристические приемы решения задач основаны не на строгих математических моделях и алгоритмах, а на соображениях "здравого смысла".

Развитием эвристических алгоритмов обработки нечетких данных можно считать самоорганизующиеся системы. В любом случае исходным ядром последних является обработка нечеткостей, а следовательно, используются принципы мышления человека. Однако самоорганизующиеся системы идут дальше и начинают развиваться, настраиваться на объект, в определенном смысле, самостоятельно, используя получаемую в процессе работы информацию об объекте управления. В общем случае можно предложить следующую схему реализации процесса управления: распознавание —> предсказание —> идентификация —> принятие решения —> управление.

Генетические алгоритмы (ГА) относятся к числу универсальных методов оптимизации, позволяющих решать задачи различных типов (комбинаторные, общие задачи с ограничениями и без ограничений) и различной степени сложности. При этом ГА характеризуются возможностью как однокритериального, так и многокритериального поиска в большом пространстве, ландшафт которого является негладким. В последние годы резко возросло число работ, прежде всего зарубежных ученых, посвященных развитию теории ГА и вопросам их практического использования. Результаты

данных исследований показывают, в частности, что ГА могут получить более широкое распространение при интеграции с другими методами и технологиями. Появились работы, в которых доказывается эффективность интеграции ГА и методов теории нечеткости, а также нейронных вычислений и систем.

Эффективность такой интеграции нашла практическое подтверждение в разработке соответствующих инструментальных средств (ИС). Так, фирма Attar Software включила ГА-компонент, ориентированный на решение задач оптимизации, в свои ИС, предназначенные для разработки экспертной системы.

Интеграция ГА и нейронных сетей позволяет решать проблемы поиска оптимальных значений весов входов нейронов, а интеграция ГА и нечеткой логики позволяет оптимизировать систему продукционных правил, которые могут быть использованы для управления операторами ГА (двунаправленная интеграция).

Одним из наиболее востребованных приложений ГА в области Data Mining является поиск наиболее оптимальной модели (поиск алгоритма, соответствующего специфике конкретной области).

Нейронные сети — это класс моделей, основанных на биологической аналогии с мозгом человека и предназначенных после прохождения этапа так называемого обучения на имеющихся данных для решения разнообразных задач анализа данных. При применении этих методов, прежде всего, встает вопрос выбора конкретной архитектуры сети (числа "слоев" и количества "нейронов" в каждом из них). Размер и структура сети должны соответствовать (например, в смысле формальной вычислительной сложности) существу исследуемого явления. Поскольку на начальном этапе анализа природа явления обычно известна плохо, выбор архитектуры является непростой задачей и часто связан с длительным процессом "проб и ошибок" (однако в последнее время стали появляться нейронно-сетевые программы, в которых для решения трудоемкой задачи поиска наилучшей архитектуры сети применяются методы искусственного интеллекта).

Затем построенная сеть подвергается процессу так называемого обучения. На этом этапе нейроны сети итеративно обрабатывают входные данные и корректируют свои веса так, чтобы сеть наилучшим образом прогнозировала (в традиционных терминах следовало бы сказать "осуществляла подгонку") данные, на которых выполняется "обучение". После обучения на имеющихся данных сеть готова к работе и может использоваться для построения прогнозов.

Нейронная сеть, полученная в результате "обучения", выражает закономерности, присутствующие в данных. При таком подходе она оказывается функциональным эквивалентом некоторой модели зависимостей между переменными, подобной тем, которые строятся в традиционном моделировании. Однако, в отличие от традиционных моделей, в случае нейронных сетей эти зависимости не могут быть записаны в явном виде, подобно тому, как это делается в статистике (например, "A положительно коррелированно с B для наблюдений, у которых величина C мала, а O велика"). Иногда нейронные сети выдают прогноз очень высокого качества, однако они представляют собой типичный пример нетеоретического подхода к исследованию (иногда это называют "черным ящиком"). При таком подходе сосредотачиваются исключительно на практическом результате, в данном случае на точности прогнозов и их прикладной ценности, а не на сути механизмов, лежащих в основе явления, или на соответствии полученных результатов какой-либо имеющейся теории.

Следует, однако, отметить, что методы нейронных сетей могут применяться и в исследованиях, направленных на построение объясняющей модели явления, поскольку нейронные сети помогают изучать данные с целью поиска значимых переменных или групп таких переменных, и полученные результаты могут облегчить процесс последующего построения модели. Более того, сейчас имеются нейросетевые программы, которые с помощью сложных алгоритмов могут находить наиболее важные входные переменные, что уже непосредственно помогает строить модель.

Одно из главных преимуществ нейронных сетей состоит в том, что они, по крайней мере теоретически, могут аппроксимировать любую непрерывную

функцию, и поэтому исследователю нет необходимости заранее принимать какие-либо гипотезы относительно модели и даже в ряде случаев о том, какие переменные действительно важны. Однако существенным недостатком нейронных сетей является то обстоятельство, что окончательное решение зависит от начальных установок сети и, как уже отмечалось, его практически невозможно интерпретировать в традиционных аналитических терминах, которые обычно применяются при построении теории явления.

Нейронная сеть — это процессор с массивным распараллеливанием операций, обладающий естественной способностью сохранять экспериментальные знания и делать их доступными для последующего использования. Он похож на мозг в двух отношениях: сеть приобретает знания в результате процесса обучения; для хранения информации используются величины интенсивности межнейронных соединений, которые называются синоптическими весами.

6.6. Процесс обнаружения знаний

Для обнаружения знаний в данных недостаточно просто применить методы Data Mining, хотя, безусловно, этот этап является основным в процессе интеллектуального анализа. Весь процесс состоит из нескольких этапов. Рассмотрим основные из них, чтобы продемонстрировать, что без специальной подготовки аналитика методы Data Mining сами по себе не решают существующих проблем. Итак, весь процесс можно разбить на следующие этапы (рис. 6.2):

- понимание и формулировка задачи анализа;
- подготовка данных для автоматизированного анализа (препроцессинг);
- применение методов Data Mining и построение моделей;
- проверка построенных моделей;

интерпретация моделей человеком.

На первом этапе выполняется осмысление поставленной задачи и уточнение целей, которые должны быть достигнуты методами Data Mining. Важно правильно сформулировать цели и выбрать необходимые для их достижения методы, т. к. от этого зависит дальнейшая эффективность всего процесса.

Рис. 6.2. Этапы интеллектуального анализа данных

Второй этап состоит в приведении данных к форме, пригодной для применения конкретных методов Data Mining. Данный процесс далее будет описан более подробно, здесь заметим только, что вид преобразований, совершаемых над данными, во многом зависит от используемых методов, выбранных на предыдущем этапе.

Третий этап— это собственно применение методов Data Mining. Сценарии этого применения могут быть самыми различными и могут включать сложную

комбинацию разных методов, особенно если используемые методы позволяют проанализировать данные с разных точек зрения. Следующий этап — проверка построенных моделей. Очень простой и часто используемый способ заключается в том, что все имеющиеся данные, которые необходимо анализировать, разбиваются на две группы. Как правило, одна из них большего размера, другая — меньшего. На большей группе, применяя те или иные методы Data Mining получают модели, а на меньшей — проверяют их. По разнице в точности между тестовой и обучающей группами можно судить об адекватности построенной модели.

Последний этап — интерпретация полученных моделей человеком в целях их использования для принятия решений, добавление получившихся правил и зависимостей в базы знаний и т. д. Этот этап часто подразумевает использование методов, находящихся на стыке технологии Data Mining и технологии экспертных систем. От того, насколько эффективным он будет, в значительной степени зависит успех решения поставленной задачи. Этим этапом завершается цикл Data Mining. Окончательная оценка ценности добытого нового знания выходит за рамки анализа, автоматизированного или традиционного, и может быть проведена только после претворения в жизнь решения, принятого на основе добытого знания, после проверки нового знания практикой. Исследование достигнутых практических результатов завершает оценку ценности добытого средствами Data Mining нового знания.

Подготовка исходных данных. Как уже отмечалось ранее, для применения того или иного метода Data Mining к данным их необходимо подготовить к этому. Например, поставлена задача построить фильтр электронной почты, не пропускающий спам. Письма представляют собой тексты в электронном виде. Практически ни один из существующих методов Data Mining не может работать непосредственно с текстами. Чтобы работать с ними, необходимо из исходной текстовой информации предварительно получить некие производные параметры, например: частоту встречаемости ключевых слов, среднюю длину предложений, параметры, характеризующие сочетаемость тех или иных слов в

предложении, и т. д. Другими словами, необходимо выработать некий четкий набор числовых или нечисловых параметров, характеризующих письмо. Эта задача наименее автоматизирована в том смысле, что выбор системы данных параметров производится человеком, хотя, конечно, их значения могут вычисляться автоматически. После выбора описывающих параметров изучаемые данные могут быть представлены в виде прямоугольной таблицы, где каждая строка представляет собой отдельный случай, объект или состояние изучаемого объекта, а каждая колонка — параметры, свойства или признаки всех исследуемых объектов. Большинство методов Data Mining работают только с подобными прямоугольными таблицами.

Полученная прямоугольная таблица пока еще является слишком сырым материалом для применения методов Data Mining и входящие в нее данные необходимо предварительно обработать. Во-первых, таблица может содержать параметры, имеющие одинаковые значения для всей колонки. Если бы исследуемые объекты характеризовались только такими признаками, они были бы абсолютно идентичны, значит, эти признаки никак не индивидуализируют исследуемые объекты. Следовательно, их надо исключить из анализа. Во-вторых, таблица может содержать некоторый категориальный признак, значения которого во всех записях различны. Ясно, что мы никак не можем использовать это поле для анализа данных и его надо исключить. Наконец, просто этих полей может быть очень много, и если все их включить в исследование, то это существенно увеличит время вычислений, поскольку практически для всех методов Data Mining характерна сильная зависимость времени от количества параметров (квадратичная, а нередко и экспоненциальная). В то же время зависимость времени от количества исследуемых объектов линейна или близка к линейной. Поэтому в качестве предварительной обработки данных необходимо, во-первых, выделить то множество признаков, которые наиболее важны в контексте данного исследования, отбросить явно неприменимые из-за константности или чрезмерной вариабельности и выделить те, которые наиболее вероятно войдут в искомую зависимость. Для этого, как правило,

используются статистические методы, основанные на применении корреляционного анализа, линейных регрессий и т. д. Такие методы позволяют быстро, хотя и приближенно оценить влияние одного параметра на другой.

Мы обсудили очистку данных по столбцам таблицы (признакам). Точно так же бывает необходимо провести предварительную очистку данных по строкам таблицы (записям). Любая реальная база данных обычно содержит ошибки, очень приблизительно определенные значения, записи, соответствующие каким-то редким, исключительным ситуациям, и другие дефекты, которые могут резко понизить эффективность методов Data Mining, применяемых на следующих этапах анализа. Такие записи необходимо отбросить. Даже если подобные "выбросы" не являются ошибками, а представляют собой редкие исключительные ситуации, они все равно вряд ли могут быть использованы, поскольку по нескольким точкам статистически значимо судить об искомой зависимости невозможно. Эта предварительная обработка или препроцессинг данных и составляет второй этап процесса обнаружения знаний.

Средства Data Mining. В настоящее время технология Data Mining представлена целым рядом коммерческих и свободно распространяемых программных продуктов. Достаточно полный и регулярно обновляемый список этих продуктов можно найти на сайте www.kdnuggets.com, посвященном Data Mining. Классифицировать программные продукты Data Mining можно по тем же принципам, что положены в основу классификации самой технологии. Однако подобная классификация не будет иметь практической ценности. Вследствие высокой конкуренции на рынке и стремления к полноте технических решений многие из продуктов Data Mining охватывают буквально все аспекты применения аналитических технологий. Поэтому целесообразнее классифицировать продукты Data Mining по тому, каким образом они реализованы и, соответственно, какой потенциал для интеграции они предоставляют. Очевидно, что и это условность, поскольку такой критерий не позволяет очертить четкие границы между продуктами. Однако у подобной классификации есть одно несомненное преимущество. Она позволяет быстро

принять решение о выборе того или иного готового решения при инициализации проектов в области анализа данных, разработки систем поддержки принятия решений, создания хранилищ данных и т. д.

Итак, продукты Data Mining условно можно разделить на три больших категории: 1) входящие, как неотъемлемая часть, в системы управления базами данных; 2) библиотеки алгоритмов Data Mining с сопутствующей инфраструктурой; 3) коробочные или настольные решения ("черные ящики").

Продукты первых двух категорий предоставляют наибольшие возможности для интеграции и позволяют реализовать аналитический потенциал практически в любом приложении в любой области. Коробочные приложения, в свою очередь, могут предоставлять некоторые уникальные достижения в области Data Mining или быть специализированными для какой-либо конкретной сферы применения. Однако в большинстве случаев их проблематично интегрировать в более широкие решения.

Включение аналитических возможностей в состав коммерческих систем управления базами данных является закономерной и имеющей огромный потенциал тенденцией. Действительно, где, как ни в местах концентрации данных, имеет наибольший смысл размещать средства их обработки. Исходя из этого принципа, функциональность Data Mining в настоящий момент реализована в следующих коммерческих базах данных: Oracle, Microsoft SQL Server, IBM DB2.

Каждая из названных СУБД позволяет решать основные задачи, связанные с анализом данных, и имеет хорошие возможности для интеграции. Однако только Oracle может считаться действительно аналитической платформой. Помимо реализации функциональности Data Mining, Oracle имеет мощные средства для анализа неструктурированной текстовой информации (Oracle Text), информации, имеющей сетевую модель организации (Oracle Network Data Models), и информации, имеющей пространственные атрибуты (Oracle Spatial Topology). Таким образом, используя СУБД Oracle как платформу для построения аналитической системы, можно решить практически любую по-

ставленную задачу: от построения рекомендательных систем для интернет-магазинов до многофункциональных систем поддержки принятия решений для различных государственных и силовых ведомств. Далее приводится обзор основных аналитических возможностей Oracle.

Orac1e Data Mining включает в себя четыре наиболее мощных и зарекомендовавших себя алгоритма классификации (supervised learning):

Naive Bayes (NB) — использует теорию вероятности для классификации объектов;

Decision Trees — классифицирует объекты путем построения деревьев решений;

Adaptive Bayes Networks (ABN) — расширенный вариант алгоритма NB;

Support Vector Machines — использует теорию вычисления близости векторов для классификации объектов.

Для решения задач кластеризации и ассоциативного анализа (unsupervised learning) в Oracle применяется пять алгоритмов:

Enhanced k-Means Clustering — для обнаружения групп схожих объектов;

Orthogonal Partitioning Clustering — для кластеризации методом ортогонального деления;

Anomaly Detection — для обнаружения редких вызывающих подозрение событий (аномалий);

Association Rules — для обнаружения шаблонов в событиях;

Nonnegative Matrix Factorization (NMF) — для уменьшения количества атрибутов.

Oracle также включает в себя алгоритм Minimum Description Length (MDL) для решения проблемы важности атрибутов. С его помощью можно определить те атрибуты, которые имеют наибольшее влияние на зависимые поля или атрибуты.

В последние годы особую важность приобрели задачи, связанные с обработкой больших массивов генетической информации. Для их решения OracleI предлагает алгоритм BLAST (Basic Local Alignment Search Technique),

позволяющий в геномных данных отыскивать последовательности, которые наиболее точно соответствуют определенным последовательностям.

Доступ к функциональности Data Mining в Oracle осуществляется либо посредством расширения языка SQL, либо с помощью прикладного программного интерфейса на Java. Отметим, что данный интерфейс совместим со спецификацией JDM.

Контрольные вопросы

1. Что такое Data Mining.
2. Какие методы Data Mining являются основными?
3. Что отличает описательные и предсказательные задачи анализа данных.
4. Чем характеризуется задача классификации и регрессии?
5. При решении каких задач применяется двухстадийная фиксация транзакций?
6. В чем суть задачи поиска ассоциативных правил?
7. В чем состоит задача кластеризации?
8. Какова цель технологии Data Mining?
9. Приведите основные характеристики процесса обнаружения знаний в хранилище данных.
10. Какие существуют основные методы анализа данных?

Заключение

Представленный в данном пособии материал является базой для успешного изучения особенностей построения и использования Хранилищ Данных, являющихся центральным звеном систем поддержки принятия решений, все шире входящих в информационный процесс управления на всех уровнях иерархии общества.

Дальнейшее изучение представленных в списке литературы источников позволит заинтересованным читателям развить представление о сфере использования и создания Хранилищ Данных, сделать правильный выбор необходимых технологий для решения вставших перед ними масштабных задач анализа данных.

Литература

1. Барсегян А.А., Куприянов М.С., Холод И.И., Тесс М.Д.. Анализ данных и процессов: учеб. пособие – СПб.: БХВ-Петербург, 2009.– 512 с.
2. Туманов В.Е., Маклаков С.В. Проектирование реляционных хранилищ данных. – М.: Диалог-МИФИ, 2007.– 333 с
3. Туманов В.Е. Проектирование хранилищ данных для систем бизнес-аналитики: учебное пособие. М.: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2010. — 615 с

Глоссарий

CIF – (Corporate Information Factory) корпоративная информационная фабрика.

CWM – (Common Warehouse Metamodel) общая метамодель хранилища данных, которая является одним из стандартов, использующих XML-технологии.

CWMI – (Common Warehouse Metadata Interchange) обмен общими метаданными Хранилища данных.

Data Mining — исследование и обнаружение "машиной" (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны, доступны для интерпретации человеком.

DSS – (Decision Support System) система поддержки принятия решений.

EIF – (Enterprise Information Factory) виртуальное предприятие.

EIS – (Executive Information Systems) информационная система руководства.

ERP – (Enterprise Resource Planing) система планирования масштаба предприятия.

ETL – (extraction, transformation, loading) извлечение, трансформация и загрузка данных в хранилище.

FASMI – (Fast of Analysis Shared Multidimensional Information) тест для определения принадлежности информационной системы к классу OLAP систем.

MRP – (Manufacturing Resource Planing) планирование материальных затрат.

OLAP – (On-Line Analysis Processing) автоматизированные системы оперативной аналитической обработки данных.

OLTP – (On-Line Transaction Processing) автоматизированные системы оперативной обработки транзакций.

ROLAP — реляционный (relational) OLAP.

SADT – метод структурного анализа и проектирования.

UML – (Unified Modeling Language) язык объектно-ориентированного моделирования.

VDW – (Virtual Data Warehouse) виртуальное хранилище данных.

XML – (eXtensible Markup Language) расширяемый язык разметки.

Архитектура ХД – совокупность программно-аппаратных компонент и технологических и организационных решений, предпринимаемых для создания, разработки и функционирования ХД.

Ассоциативные модели — выявление закономерностей между связанными событиями.

Диаграммы потоков данных – (ДПД или DFD), технология описания асинхронного процесса преобразования информации от ее ввода в систему до выдачи пользователю.

Жизненный цикл продукта – набор определенным образом расположенных во времени фаз (этапов), которые проходит продукт от момента его создания до момента его утилизации.

Измерение — последовательность значений одного из анализируемых параметров.

Информационная технология – совокупность методов и способов получения, обработки, представления информации, направленных на изменение ее состояния, свойств, формы, содержания и осуществляемых в интересах пользователей.

Киоск данных **–** облегченный вариант ХД тематической направленности.

Монитор транзакций – (TPM - transaction processing monitor). Мониторы транзакций выполняют две основные функции: динамическое распределение запросов в системе (выравнивание нагрузки) и оптимизация числа выполняющихся серверных приложений.

Нейронные сети — класс моделей, основанных на биологической аналогии с мозгом человека.

ПО – программное обеспечение.

Предсказательные (predictive) *модели* – строятся на основании набора данных с известными результатами.

Сиквенциалъный анализ – разновидность задачи поиска ассоциативных правил. Основным отличием задачи сиквенциального анализа от поиска ассоциативных

правил является установление отношения порядка между исследуемыми наборами.

СППР – система поддержки принятия решений.

Технология – наука о законах производства материальных благ, содержащая три такие основные части как идеология (принципы производства), орудия труда (станки, машины, агрегаты) и кадры, владеющие профессиональными навыками

Транзакция – неделимая с позиции воздействия на БД последовательность операции манипулирования данными.

Фактографические системы – автоматизированные информационные системы, оперирующие фактическими сведениями, представленными в виде специальным образом организованных совокупностей формализованных записей данных.