Способы аналитической обработки данных для поддержки принятия решений

Л. В. Щавелёв, Leonid@iname.com, http://www.polytech.ivanovo.su/~leonid.

Современный уровень развития аппаратных и программных средств с некоторых пор сделал возможным повсеместное ведение баз данных оперативной информации на всех уровнях управления. В процессе своей деятельности промышленные предприятия, корпорации, ведомственные структуры, органы государственной власти и управления накопили большие объемы данных. Они хранят в себе большие потенциальные возможности по извлечению полезной аналитической информации, на основе которой можно выявлять скрытые тенденции, строить стратегию развития, находить новые решения.

В последние годы в мире оформился ряд новых концепций хранения и анализа корпоративных данных:

1) Хранилища данных[1] (Data Warehouse) [42, 11, 15];
2) Оперативная аналитическая обработка данных (On-Line Analytical Processing, OLAP) [31, 16, 25];
3) Интеллектуальный анализ данных - ИАД (Data Mining) [52, 49, 62, 13].

Обзору этих концепций, а также доказательству их взаимодополняемости в деле поддержки принятия управленческих решений, посвящена настоящая статья.

1. Хранилища (склады) данных

В области информационных технологий всегда сосуществовали два класса систем [16, С. 49]:

1) системы, ориентированные на операционную (транзакционную) обработку данных; в англоязычной литературе они часто называются термином OLTP (On-Line Transaction Processing, оперативная транзакционная обработка), в противовес OLAP - оперативной аналитической обработке [55]; А. А. Сахаров [15, С. 55] определяет их термином "системы обработки данных" (СОД);
2) системы, ориентированные на аналитическую обработку данных - системы поддержки принятия решений (СППР), или Decision Support Systems (DSS).

На первых стадиях информатизации всегда требуется навести порядок именно в процессах повседневной рутинной обработки данных, на что и ориентированы традиционные СОД, поэтому опережающее развитие этого класса систем вполне объяснимо.

Системы второго класса - СППР - являются вторичными по отношению к ним. Часто возникает ситуация, когда данные в организации накапливаются с ряде несвязанных СОД, во многом дублируя друг друга, но не будучи никак согласованы. В таком случае достоверную комплексную информацию получить практически невозможно, несмотря на ее кажущийся избыток.

---

[1] Термин data warehouse чаще всего переводится на русский язык как "хранилище данных" или "информационное хранилище" [25, 26]. Второй перевод противоречит самой идее концепции (концентрация корпоративных данных для обеспечения возможности их анализа с целью извлечения полезной информации), за что и критикуется многими авторами. С. Кузнецов [20] и В. Пржиялковский [23] предлагают использовать дословный перевод - "склад данных" - как наиболее точный. Традиционный термин "хранилище данных" и новаторский - "склад данных" - будут использоваться в дальнейшем как синонимы.

Целью построения корпоративного хранилища данных является интеграция, актуализация и согласование оперативных данных из разнородных источников для формирования единого непротиворечивого взгляда на объект управления в целом. При этом в основе концепции хранилищ данных лежит признание необходимости разделения наборов данных, используемых для транзакционной обработки, и наборов данных, применяемых в системах поддержки принятия решений. Такое разделение возможно путем интеграции разъединенных в СОД и внешних источниках детализированных данных в едином хранилище, их согласования и, возможно, агрегации. W. Inmon, автор концепции хранилищ данных [42], определяет такие хранилища как:

1) "предметно-ориентированные,
2) интегрированные,
3) неизменчивые,
4) поддерживающие хронологию

наборы данных, организованные с целью поддержки управления", призванные выступать в роли "единого и единственного источника истины", обеспечивающего менеджеров и аналитиков достоверной информацией, необходимой для оперативного анализа и поддержки принятия решений.

Концепция хранилищ данных предполагает не просто единый логический взгляд на данные организации, а действительную реализацию единого интегрированного источника данных. Альтернативным по отношению к этой концепции способом формирования единого взгляда на корпоративные данные является создание виртуального источника, опирающегося на распределенные базы данных различных СОД. При этом каждый запрос к такому источнику динамически транслируется в запросы к исходным базам данных, а полученные результаты на лету согласовываются, связываются, агрегируются и возвращаются к пользователю. Однако, при внешней элегантности, такой способ обладает рядом существенных недостатков.

1. Время обработки запросов к распределенному хранилищу значительно превышает соответствующие показатели для централизованного хранилища. Кроме того, структуры баз данных СОД, рассчитанные на интенсивное обновление одиночных записей, в высокой степени нормализованы, поэтому в аналитическом запросе к ним требуется объединение большого числа таблиц, что также приводит к снижению быстродействия.
2. Интегрированный взгляд на распределенное корпоративное хранилище возможен только при выполнении требования постоянной связи всех источников данных в сети. Таким образом, временная недоступность хотя бы одного из источников может либо сделать работу информационно-аналитической системы (ИАС) невозможной, либо привести к ошибочным результатам.
3. Выполнение сложных аналитических запросов над таблицами СОД потребляет большой объем ресурсов сервера БД и приводит к снижению быстродействия СОД, что недопустимо, так как время выполнения операций в СОД часто весьма критично.
4. Различные СОД могут поддерживать разные форматы и кодировки данных, данные в них могут быть несогласованы. Очень часто на один и тот же вопрос может быть получено несколько вариантов ответа, что может быть связано с несинхронностью моментов обновления данных, отличиями в трактовке отдельных событий, понятий и данных, изменением семантики данных в процессе развития предметной области, ошибками при вводе, утерей фрагментов архивов и т. д. В таком случае цель - формирование единого непротиворечивого взгляда на объект управления - может не быть достигнута.
5. Главным же недостатком следует признать практическую невозможность обзора длительных исторических последовательностей, ибо при физическом отсутствии центрально-

го хранилища доступны только те данные, которые на момент запроса есть в реальных БД связанных СОД. Основное назначение СОД - оперативная обработка данных, поэтому они не могут позволить себе роскошь хранить данные за длительный (более нескольких месяцев) период; по мере устаревания данные выгружаются в архив и удаляются из транзакционной БД. Что касается аналитической обработки, для нее как раз наиболее интересен взгляд на объект управления в исторической ретроспективе.

Таким образом, хранилище данных функционирует по следующему сценарию. По заданному регламенту в него собираются данные из различных источников - баз данных систем оперативной обработки. В хранилище поддерживается хронология: наравне с текущими хранятся исторические данные с указанием времени, к которому они относятся. В результате необходимые доступные данные об объекте управления собираются в одном месте, приводятся к единому формату, согласовываются и, в ряде случаев, агрегируются до минимально требуемого уровня обобщения.

Облегченным вариантом корпоративного хранилища данных могут быть витрины данных (Data Mart), то есть тематические БД, содержащие информацию, относящуюся к отдельным аспектам деятельности организации. Концепция витрин данных была предложена Forrester Research в 1991 году [15]. При этом главная идея заключалась в том, что витрины данных содержат тематические подмножества заранее агрегированных данных, по размерам гораздо меньшие, чем общекорпоративное хранилище данных, и, следовательно, требующие менее производительной техники для поддержания. В 1994 году M. Demarest [32] предложил объединить две концепции и использовать хранилище данных в качестве единого интегрированного источника для многочисленных витрин данных. В таком варианте корпоративная информационно-аналитическая система имеет трехуровневую структуру:

1) общекорпоративное централизованное хранилище данных;
2) тематические витрины данных на уровне подразделений;
3) рабочие места конечных пользователей, снабженные аналитическим инструментарием.

Рассмотренная концепция ориентирована исключительно на хранение, а не на обработку корпоративных данных. Она не предопределяет архитектуру целевых аналитических систем, а только создает поле деятельности для их функционирования, концентрируясь на требованиях к данным. Таким образом, она оставляет свободу выбора во всем, что касается:

1) способов представления данных в целевом хранилище (например, реляционный, многомерный);
2) режимов анализа данных хранилища.

2. *Способы аналитической обработки данных*

Для того чтобы существующие хранилища данных способствовали принятию управленческих решений, информация должна быть представлена аналитику в нужной форме, то есть он должен иметь развитые инструменты доступа к данным хранилища и их обработки.

По критерию режима анализа данных информационно-аналитические системы подразделяются на две категории [11, 15]:

1) статические (включающие предопределенный набор сценариев обработки данных и составления отчетов); в эту категорию входят так называемые информационные системы руководителя (ИСР);
2) динамические (поддерживающие построение и выполнение нерегламентированных запросов и формирование отчетов произвольной формы).

Очень часто ИАС, создаваемые в расчете на непосредственное использование лицами, принимающими решения, оказываются чрезвычайно просты в применении, но жестко ограничены в функциональности. Такие статические СППР [15, С. 55], или Информацион-

ные системы руководителя (ИСР) [13, С. 73] - (Executive Information Systems, EIS) [45, С. 4] - содержат в себе предопределенные множества запросов и, будучи достаточными для повседневного обзора, неспособны ответить на все вопросы к имеющимся данным, которые могут возникнуть при принятии решений[2]. Результатом работы такой системы, как правило, являются многостраничные отчеты, после тщательного изучения которых у аналитика появляется новая серия вопросов; однако, каждый новый, непредусмотренный при проектировании такой системы, запрос должен быть сначала формально описан, передан программисту, закодирован и только затем выполнен. Время ожидания в таком случае может составлять часы и дни, что не всегда приемлемо. Таким образом, внешняя простота статических СППР, за которую активно борется большинство заказчиков информационно-аналитических систем, оборачивается катастрофической потерей гибкости.

Динамические СППР, напротив, ориентированы на обработку нерегламентированных, неожиданных (ad hoc) запросов аналитиков к данным. Наиболее глубоко требования к таким системам рассмотрел E. F. Codd в статье [31], положившей начало концепции OLAP. Работа аналитиков с этими системами заключается в интерактивной последовательности формирования запросов и изучения их результатов, каждый из которых может породить потребность новой серии запросов.

Но динамические СППР могут действовать не только в области оперативной аналитической обработки (OLAP); поддержка принятия управленческих решений на основе накопленных данных может выполняться в трех базовых сферах [55].

1. Сфера детализированных данных. Это сфера действия большинства систем, нацеленных на поиск информации. В большинстве случаев реляционные СУБД отлично справляются с возникающими здесь задачами. Общепризнанным стандартом языка манипулирования реляционными данными является SQL. Информационно-поисковые системы, обеспечивающие интерфейс конечного пользователя в задачах поиска детализированной информации, могут использоваться в качестве надстроек как над отдельными системами обработки данных (СОД), так и над хранилищем данных в целом.
2. Сфера агрегированных показателей. Комплексный взгляд на собранную в хранилище данных информацию, ее обобщение и агрегация, гиперкубическое представление и многомерный анализ являются задачами систем оперативной аналитической обработки данных (OLAP) [31, 25, 16]. Здесь можно или ориентироваться на специальные многомерные СУБД [16], или (что, как правило, предпочтительнее) оставаться в рамках реляционных технологий. Во втором случае заранее агрегированные данные могут собираться в БД звездообразного вида [39, 14, 61], либо агрегация информации может производиться на лету в процессе сканирования детализированных таблиц реляционной БД [38].
3. Сфера закономерностей. Интеллектуальная обработка производится методами интеллектуального анализа данных (ИАД, Data Mining) [52, 10], главными задачами которых являются поиск функциональных и логических закономерностей в накопленной информации, построение моделей и правил, которые объясняют найденные аномалии и/или (с определенной вероятностью) прогнозируют развитие некоторых процессов.

Некоторые авторы [55] выделяют в отдельную область анализ отклонений (например, в целях отслеживания колебаний биржевых курсов). В качестве примера может быть приведен статистический анализ рядов динамики [4]. Чаще, однако, этот тип анализа относят к области закономерностей.

---

[2] По определению В. Пржиялковского [23, С. 81], ИСР - это "компьютерная система, позволяющая ... предоставлять информацию в распоряжение старшего управляющего персонала с ограниченным опытом обращения с ЭВМ".

Полная структура ИАС, построенной на основе хранилища данных, показана на рис. 1. В конкретных реализациях отдельные компоненты этой схемы часто отсутствуют. Настоящая статья посвящена системам оперативной аналитической обработки и средствам интеллектуального анализа данных, поэтому в оставшейся ее части будут подробно рассмотрены концепции OLAP и ИАД. Такие вопросы как автоматизация сбора данных в хранилище из внешних источников, их согласование, очистка данных, поддержание целостности хранилища, реализация инструментов поиска детализированных данных, несмотря на их несомненную важность, подробно рассматриваться не будут.

Следует отметить, что средства аналитической обработки - как OLAP, так и ИАД - могут использовать в качестве исходного материала для анализа любые данные, в том числе базы отдельных СОД. Но наибольшего эффекта можно добиться при анализе корпоративного хранилища данных, содержащего максимально полный объем актуальных и исторических сведений обо всех аспектах деятельности объекта управления и ситуации вокруг него.

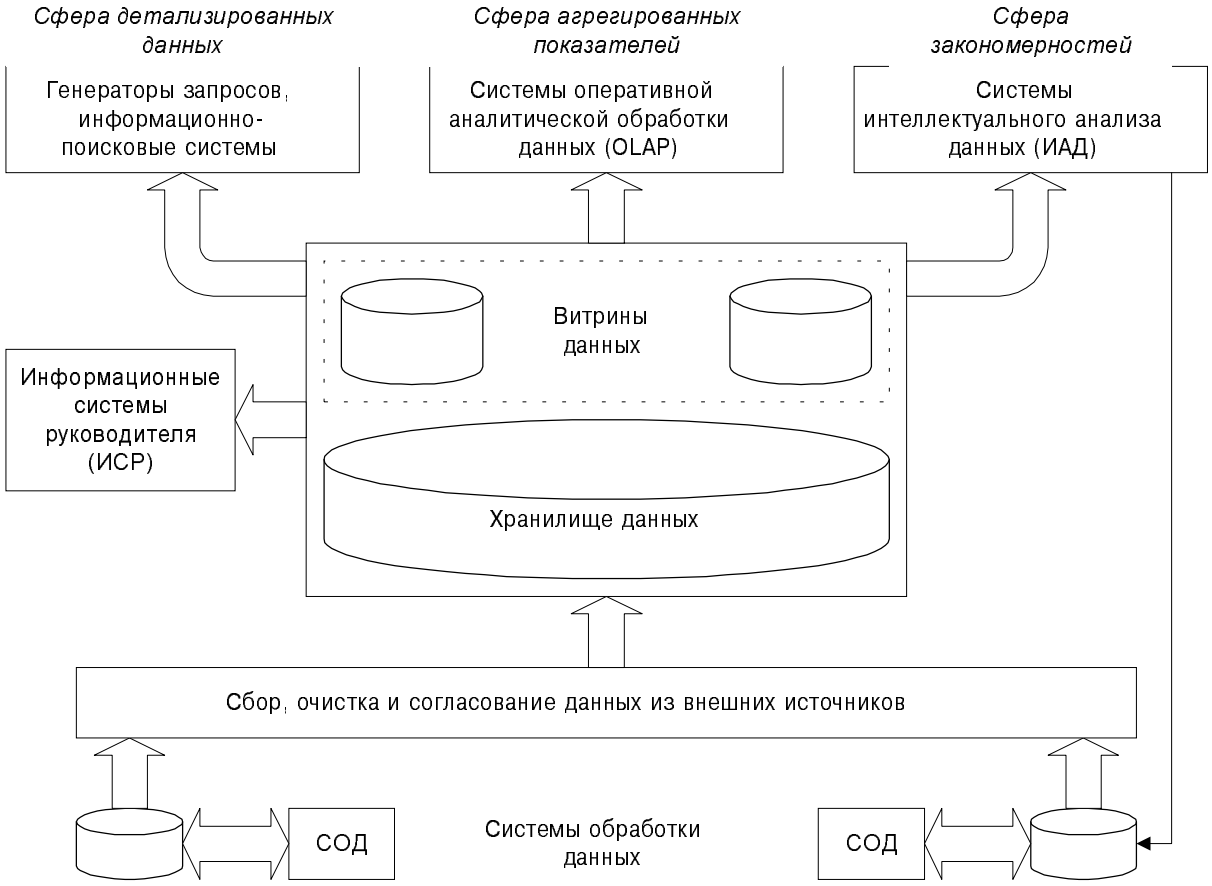

Рис. 1. Полная структура корпоративной ИАС.

3. Оперативная аналитическая обработка данных

В основе концепции оперативной аналитической обработки (OLAP) лежит многомерное представление данных. Термин OLAP ввел E. F. Codd в 1993 году [31]. В своей статье он рассмотрел недостатки реляционной модели, в первую очередь невозможность "объединять, просматривать и анализировать данные с точки зрения множественности измерений, то есть самым понятным для корпоративных аналитиков способом", и определил

общие требования к системам OLAP, расширяющим функциональность реляционных СУБД и включающим многомерный анализ как одну из своих характеристик.

Следует заметить, что Кодд обозначает термином OLAP многомерный способ представления данных исключительно на концептуальном уровне. Используемые им термины - "Многомерное концептуальное представление" ("Multidimensional conceptual view"), "Множественные измерения данных" ("Multiple data dimensions"), "Сервер OLAP" ("OLAP server") - не определяют физического механизма хранения данных (термины "многомерная база данных" и "многомерная СУБД" не встречаются ни разу). Однако в большом числе публикаций аббревиатурой OLAP обозначается не только многомерный взгляд на данные, но и хранение самих данных в многомерной БД [16, 2] (это в принципе неверно, поскольку сам Кодд в [31] отмечает, что "Реляционные БД были, есть и будут наиболее подходящей технологией для хранения корпоративных 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"). Такая путаница приводит к противопоставлениям наподобие "OLAP или ROLAP", что, вообще говоря, некорректно, поскольку ROLAP (реляционный OLAP) на концептуальном уровне поддерживает всю определенную термином OLAP функциональность. Поэтому более предпочтительным кажется использование для OLAP на основе многомерных СУБД специального термина MOLAP, как это и сделано в [14, 24].

По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) является наиболее естественным взглядом управляющего персонала на объект управления. Оно представляет собой множественную перспективу, состоящую из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям данных определяется как многомерный анализ. Каждое измерение включает направления консолидации данных, состоящие из серии последовательных уровней обобщения, где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению. Так, измерение Исполнитель может определяться направлением консолидации, состоящим из уровней обобщения "предприятие - подразделение - отдел - служащий". Измерение Время может даже включать два направления консолидации - "год - квартал - месяц - день" и "неделя - день", поскольку счет времени по месяцам и по неделям несовместим. В этом случае становится возможным произвольный выбор желаемого уровня детализации информации по каждому из измерений. Операция спуска (drilling down) соответствует движению от высших ступеней консолидации к низшим; напротив, операция подъема (rolling up) означает движение от низших уровней к высшим (рис. 2).

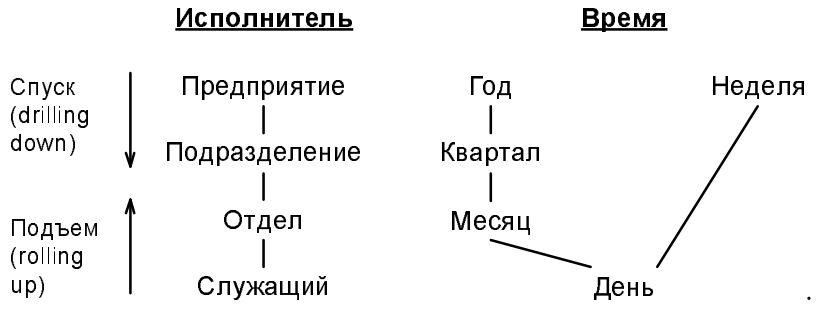

Рис. 2. Измерения и направления консолидации данных.

3.1. Требования к средствам оперативной аналитической обработки

Кодд определил 12 правил, которым должен удовлетворять программный продукт класса OLAP (табл. 1).

Таблица 1

Правила оценки программных продуктов класса OLAP

1.	Многомерное концептуальное представление данных (Multi-Dimensional Conceptual View)	Концептуальное представление модели данных в продукте OLAP должно быть многомерным по своей природе, то есть позволять аналитикам выполнять интуитивные операции "анализа вдоль и поперек"[3] ("slice and dice"), вращения (rotate) и размещения (pivot) направлений консолидации.
2.	Прозрачность (Transparency)	Пользователь не должен знать о том, какие конкретные средства используются для хранения и обработки данных, как данные организованы и откуда берутся.
3.	Доступность (Accessibility)	Аналитик должен иметь возможность выполнять анализ в рамках общей концептуальной схемы, но при этом данные могут оставаться под управлением оставшихся от старого наследства СУБД, будучи при этом привязанными к общей аналитической модели. То есть инструментарий OLAP должен накладывать свою логическую схему на физические массивы данных, выполняя все преобразования, требующиеся для обеспечения единого, согласованного и целостного взгляда пользователя на информацию.
4.	Устойчивая производительность (Consistent Reporting Performance)	С увеличением числа измерений и размеров базы данных аналитики не должны столкнуться с каким бы то ни было уменьшением производительности. Устойчивая производительность необходима для поддержания простоты использования и свободы от усложнений, которые требуются для доведения OLAP до конечного пользователя.
5.	Клиент - серверная архитектура (Client-Server Architecture)	Большая часть данных, требующих оперативной аналитической обработки, хранится в мэйнфреймовых системах, а извлекается с персональных компьютеров. Поэтому одним из требований является способность продуктов OLAP работать в среде клиент-сервер. Главной идеей здесь является то, что серверный компонент инструмента OLAP должен быть достаточно интеллектуальным и обладать способностью строить общую концептуальную схему на основе обобщения и консолидации различных логических и физических схем корпоративных баз данных для обеспечения эффекта прозрачности.
6.	Равноправие измерений (Generic Dimensionality)	Все измерения данных должны быть равноправны. Дополнительные характеристики могут быть предоставлены отдельным измерениям, но поскольку все они симметричны, данная дополнительная функциональность может быть предоставлена любому измерению. Базовая структура данных, формулы и форматы отчетов не должны опираться на какое-то одно измерение.

[3] Перевод С. Д. Кузнецова (выступление на 3-й ежегодной конференции "Корпоративные базы данных '98").

7.	Динамическая обработка разреженных матриц (Dynamic Sparse Matrix Handling)	Инструмент OLAP должен 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 разреженность данных.
8.	Поддержка многопользовательского режима (Multi-User Support)	Зачастую несколько аналитиков имеют необходимость работать одновременно с одной аналитической моделью или создавать различные модели на основе одних корпоративных данных. Инструмент OLAP должен предоставлять им конкурентный доступ, обеспечивать целостность и защиту данных.
9.	Неограниченная поддержка кроссмерных операций (Unrestricted Cross-dimensional Operations)	Вычисления и манипуляция данными по любому числу измерений не должны запрещать или ограничивать любые отношения между ячейками данных. Преобразования, требующие произвольного определения, должны задаваться на функционально полном формульном языке.
10.	Интуитивное манипулирование данными (Intuitive Data Manipulation)	Переориентация направлений консолидации, детализация данных в колонках и строках, агрегация и другие манипуляции, свойственные структуре иерархии направлений консолидации, должны выполняться в максимально удобном, естественном и комфортном пользовательском интерфейсе.
11.	Гибкий механизм генерации отчетов (Flexible Reporting)	Должны поддерживаться различные способы визуализации данных, то есть отчеты должны представляться в любой возможной ориентации.
12.	Неограниченное количество измерений и уровней агрегации (Unlimited Dimensions and Aggregation Levels)	Настоятельно рекомендуется допущение в каждом серьезном OLAP инструменте как минимум пятнадцати, а лучше двадцати, измерений в аналитической модели. Более того, каждое из этих измерений должно допускать практически неограниченное количество определенных пользователем уровней агрегации по любому направлению консолидации.

Набор этих требований, послуживших фактическим определением OLAP, достаточно часто критиковался. Так, в [16] говорится, что в рамках 12 требований смешаны:

1) собственно требования к функциональности (1, 2, 3, 6, 9, 12);
2) неформализованные пожелания (4, 7, 10, 11);
3) требования к архитектуре информационной системы, имеющие к функциональности весьма приблизительное отношение (5, 8); например, согласно требованию 5, система, реализованная на основе UNIX-сервера с терминалами, не может быть продуктом OLAP, так как не работает в клиент-серверной архитектуре; так же, OLAP продукт не может являться настольной однопользовательской системой, так как в этом случае нарушается требование 8.

С другой стороны, по утверждению самого Кодда, ни один из имеющихся в настоящее время на рынке продуктов оперативного анализа данных не удовлетворяет полностью всем выдвинутым им требованиям. Поэтому 12 правил следует рассматривать как рекомендательные, а конкретные продукты оценивать по степени приближения к идеально полному соответствию всем требованиям.

3.2. Классификация продуктов OLAP по способу представления данных

В настоящее время на рынке присутствует около 30 продуктов, которые в той или иной степени обеспечивают функциональность OLAP (по данным обзорного Web-сервера http://www.olapreport.com/ на февраль 1998 года). Обеспечивая многомерное концептуальное представление со стороны пользовательского интерфейса к исходной базе данных, все продукты OLAP делятся на три класса по типу исходной БД.

1. Самые первые системы оперативной аналитической обработки (например, Essbase компании Arbor Software [31], Oracle Express Server компании Oracle [16]) относились к классу MOLAP (Multidimensional OLAP), то есть могли работать только со своими собственными многомерными базами данных. Они основываются на патентованных технологиях для многомерных СУБД и являются наиболее дорогими. Эти системы обеспечивают полный цикл OLAP-обработки и либо включают в себя, помимо серверного компонента, собственный интегрированный клиентский интерфейс, либо используют для связи с пользователем внешние программы работы с электронными таблицами. Для обслуживания таких систем требуется специальный штат сотрудников, занимающихся установкой, сопровождением системы, формированием представлений данных для конечных пользователей. Наиболее мощным (и самым дорогим) представителем данного класса является SAS System компании SAS Institute [30, 45, 56][4]. SAS System состоит из множества подсистем-модулей, которые позволяют проектировать готовые решения - расширенные ИСР, дополненные функциями OLAP и (при использовании специальных модулей) - интеллектуального анализа [18, С. 33]. Благодаря такому подходу достигается компромисс между гибкостью настройки и простотой использования, поскольку разработкой системы поддержки принятия решений занимаются администраторы на этапе проектирования, а аналитики имеют дело с уже адаптированной для их потребностей системой.
2. Возникшие после программной статьи Кодда [31] системы оперативной аналитической обработки реляционных данных (Relational OLAP, ROLAP) позволили представлять данные, хранимые в классической реляционной базе, в многомерной форме [38, 39, 61]. К этому классу относятся DSS/Server и DSS/Agent компании MicroStrategy, MetaCube компании Informix, DecisionSuite компании Information Advantage и другие. ROLAP-системы хорошо приспособлены для работы с крупными хранилищами. Подобно системам первого класса, они требуют значительных затрат на обслуживание специалистами по информационным технологиям и предусматривают многопользовательский режим работы.
3. Наконец, появившиеся около 1997 года первые гибридные системы (Hybrid OLAP, HOLAP) разработаны с целью совмещения достоинств и минимизации недостатков, присущих предыдущим классам. К этому классу относится Media/MR компании Speedware [24]. По утверждению разработчиков, он объединяет аналитическую гибкость и скорость ответа MOLAP с постоянным доступом к реальным данным, свойственным ROLAP. Однако, этот класс систем является новым, и судить о его действительных преимуществах пока рано.

Помимо перечисленных средств существует еще один класс - инструменты генерации запросов и отчетов для настольных ПК, дополненные функциями OLAP и/или интегрированные с внешними средствами, выполняющими такие функции. Эти довольно развитые системы осуществляют выборку данных из исходных источников, преобразуют их и поме-

---

[4] Строго говоря, программное обеспечение компании SAS Institute не может быть отнесено исключительно к классу систем OLAP на основе многомерных СУБД, так как в нем имеются возможности одновременного взаимодействия с несколькими источниками данных через метауровень, что характерно для следующего класса - ROLAP.

щают в динамическую многомерную БД, функционирующую на клиентской станции конечного пользователя. Для работы с небольшими, просто организованными базами эти средства подходят наилучшим образом. Основными представителями этого класса являются BusinessObjects одноименной компании [50], BrioQuery компании Brio Technology [17, С. 34] и PowerPlay компании Cognos [17, С. 34-35].

3.2.1. Многомерный OLAP (MOLAP)

В специализированных СУБД, основанных на многомерном представлении данных, данные организованы не в форме реляционных таблиц, а в виде упорядоченных многомерных массивов:

1) гиперкубов (все хранимые в БД ячейки должны иметь одинаковую мерность, то есть находиться в максимально полном базисе измерений) или
2) поликубов (каждая переменная хранится с собственным набором измерений, и все связанные с этим сложности обработки перекладываются на внутренние механизмы системы).

Использование многомерных БД в системах оперативной аналитической обработки имеет следующие достоинства.

1. В случае использования многомерных СУБД поиск и выборка данных осуществляется значительно быстрее, чем при многомерном концептуальном взгляде на реляционную базу данных. По свидетельству, приведенному в [15, С. 68], "среднее время ответа на нерегламентированный запрос при использовании многомерной СУБД обычно на один-два порядка меньше, чем в случае реляционной СУБД с нормализованной схемой данных".
2. Из-за объективно существующих ограничений SQL в реляционных СУБД невозможно (или, по крайней мере, достаточно сложно) реализовать многие встроенные функции, легко обеспечиваемые в системах, основанных на многомерном представлении данных.

С другой стороны, имеются существенные ограничения.

1. Многомерные СУБД не позволяют работать с большими базами данных. На сегодняшний день их реальный предел - 10-20 гигабайт [15, С. 68]. К тому же за счет денормализации и предварительно выполненной агрегации 20 гигабайт в многомерной базе, как правило, соответствуют (по оценке Кодда [31]) в 2.5-100 раз меньшему объему исходных детализированных данных, то есть в лучшем случае нескольким гигабайтам.
2. Многомерные СУБД по сравнению с реляционными очень неэффективно используют внешнюю память. Ячейки гиперкуба хранятся в них в виде логически упорядоченных массивов (блоков фиксированной длины), причем именно такой блок является минимальной индексируемой единицей. Хотя в многомерных СУБД блоки, не содержащие ни одного определенного значения, не хранятся, это решает проблему только частично. Поскольку данные хранятся в упорядоченном виде, неопределенные значения не всегда удаляются полностью, да и то лишь в том случае, когда за счет выбора порядка сортировки данные удается организовать в максимально большие непрерывные группы. Но порядок сортировки, чаще всего используемый в запросах, может не совпадать с порядком, в котором они должны быть отсортированы в целях максимального устранения несуществующих значений. Таким образом, при проектировании многомерной БД часто приходится жертвовать либо быстродействием (а это одно из первых достоинств и главная причина выбора именно многомерной СУБД), либо внешней памятью (хотя, как отмечалось, максимальный размер многомерных БД ограничен).
3. В настоящее время для многомерных СУБД отсутствуют единые стандарты на интерфейс, языки описания и манипулирования данными.

4. Многомерные СУБД не поддерживают репликацию данных, часто используемую в качестве механизма загрузки.

Следовательно, использование многомерных СУБД оправдано только при следующих условиях.

1. Объем исходных данных для анализа не слишком велик (не более нескольких гигабайт), то есть уровень агрегации данных достаточно высок.
2. Набор информационных измерений стабилен (поскольку любое изменение в их структуре почти всегда требует полной перестройки гиперкуба).
3. Время ответа системы на нерегламентированные запросы является наиболее критичным параметром.
4. Требуется широкое использование сложных встроенных функций для выполнения кроссмерных вычислений над ячейками гиперкуба, в том числе возможность написания пользовательских функций.

3.2.2. Реляционный OLAP (ROLAP)

Непосредственное использование реляционных БД в качестве исходных данных в системах оперативной аналитической обработки имеет следующие достоинства.

1. При оперативной аналитической обработке содержимого хранилищ данных инструменты ROLAP позволяют производить анализ непосредственно над хранилищем (потому что в подавляющем большинстве случаев корпоративные хранилища данных реализуются средствами реляционных СУБД).
2. В случае переменной размерности задачи, когда изменения в структуру измерений приходится вносить достаточно часто, ROLAP системы с динамическим представлением размерности являются оптимальным решением, так как в них такие модификации не требуют физической реорганизации БД.
3. Системы ROLAP могут функционировать на гораздо менее мощных клиентских станциях, чем системы MOLAP, поскольку основная вычислительная нагрузка в них ложится на сервер, где выполняются сложные аналитические SQL-запросы, формируемые системой.
4. Реляционные СУБД обеспечивают значительно более высокий уровень защиты данных и разграничения прав доступа.
5. Реляционные СУБД имеют реальный опыт работы с очень большими базами данных и развитые средства администрирования.

О недостатках ROLAP-систем уже говорилось при перечислении преимуществ использования многомерных баз данных. Это, во-первых, ограниченные возможности с точки зрения расчета значений функционального типа, а во-вторых - меньшая производительность. Для обеспечения сравнимой с MOLAP производительности реляционные системы требуют тщательной проработки схемы БД и специальной настройки индексов. Но в результате этих операций производительность хорошо настроенных реляционных систем при использовании схемы "звезда" вполне сравнима с производительностью систем на основе многомерных баз данных.

Описанию схемы звезды (star schema) и рекомендациям по ее применению полностью посвящены работы [39, 61, 48]. Ее идея заключается в том, что имеются таблицы для каждого измерения, а все факты помещаются в одну таблицу, индексируемую множественным ключом, составленным из ключей отдельных измерений. Каждый луч схемы звезды задает, в терминологии Кодда, направление консолидации данных по соответствующему измерению (например, Магазин - Город/район - Регион).

В общем случае факты имеют разные множества измерений, и тогда их удобно хранить не в одной, а в нескольких таблицах; кроме того, в различных запросах пользователей

может интересовать только часть возможных измерений. Но при таком подходе при большом числе независимых измерений необходимо поддерживать множество таблиц фактов, соответствующих каждому возможному сочетанию выбранных в запросе измерений, что приводит к неэкономному использованию внешней памяти, увеличению времени загрузки данных в БД схемы звезды из внешних источников и сложностям администрирования. Для решения этой проблемы авторы работы [38] предлагают специальное расширение для языка SQL (оператор "GROUP BY CUBE" и ключевое слово "ALL")[5], а авторы [39, 48] рекомендуют создавать таблицы фактов не для всех возможных сочетаний измерений, а только для наиболее полных (тех, значения ячеек которых не могут быть получены с помощью последующей агрегации ячеек других таблиц фактов базы данных).

В сложных задачах с многоуровневыми измерениями имеет смысл обратиться к расширениям схемы звезды - схеме созвездия (fact constellation schema) [61, С. 10-11] и схеме снежинки (snowflake schema) [61, С. 13-15]. В этих случаях отдельные таблицы фактов создаются для возможных сочетаний уровней обобщения различных измерений. Это позволяет добиться наилучшей производительности, но часто приводит к избыточности данных.

В любом случае, если многомерная модель реализуется в виде реляционной базы данных, следует создавать длинные и "узкие" таблицы фактов и сравнительно небольшие и "широкие" таблицы измерений. Таблицы фактов содержат численные значения ячеек гиперкуба, а остальные таблицы определяют содержащий их многомерный базис измерений.

Ориентация на представление многомерной информации с помощью звездообразных реляционных моделей позволяет избавиться от проблемы оптимизации хранения разреженных матриц, остро стоящей перед многомерными СУБД (где проблема разреженности решается специальным выбором схемы). Хотя для хранения каждой ячейки в таблице фактов используется целая запись (которая помимо самих значений включает вторичные ключи - ссылки на таблицы измерений), несуществующие значения могут просто не быть включены в таблицу фактов, то есть наличие в базе пустых ячеек исключается. Индексирование обеспечивает приемлемую скорость доступа к данным в таблицах фактов.

4. Интеллектуальный анализ данных

Сфера закономерностей отличается от двух предыдущих тем, что в ней накопленные сведения автоматически обобщаются до информации, которая может быть охарактеризована как знания. Этот процесс чрезвычайно актуален для пользователей сейчас, и важность его будет со временем только расти, так как, согласно закону, приведенному в [34], "количество информации в мире удваивается каждые 20 месяцев", в то время как "компьютерные технологии, обещавшие фонтан мудрости, пока что только регулируют потоки данных".

Интеллектуальный анализ данных определяется в большинстве публикаций афористически - "извлечение зерен знаний из гор данных" [36], "разработка данных - по аналогии с разработкой полезных ископаемых" [2]. При этом в английском языке существует два термина, переводимые как ИАД, - Knowledge Discovery in Databases (KDD) и Data Mining (DM). В большинстве работ они используются как синонимы [см., например, 52, 36], хотя некоторые авторы [8, 26] рассматривают KDD как более широкое понятие - научное направление, образовавшееся "на пересечении искусственного интеллекта, статистики и теории баз данных" и обеспечивающее процесс извлечения информации из данных и ее использования [8], а DM - как совокупность индуктивных методов этого процесса, то есть то, что ниже будет определено как стадия свободного поиска ИАД.

---

[5] В настоящее время это расширение еще не принято, поэтому данное предложение представляет пока чисто академический интерес.

Остановимся на следующем определении: ИАД - это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей (шаблонов информации) [52, 55]. Следует отметить, что большинство методов ИАД было первоначально разработано в рамках теории искусственного интеллекта (ИИ) в 70-80-х годах, но получили распространение только в последние годы, когда проблема интеллектуализации обработки больших и быстро растущих объемов корпоративных данных потребовала их использования в качестве надстройки над хранилищами данных [49].

Первоначально [21] средства ИАД разрабатывались так, что в качестве исходного материала для анализа принимались данные, организованные в плоские реляционные таблицы. Как будет показано далее, применение ИАД над данными, представленными с помощью систем OLAP в виде реального или виртуального информационного гиперкуба, во многих случаях может оказаться более эффективным, а главное - гораздо более тесно интегрированным в единую информационно-аналитическую систему.

4.1. Классификация задач ИАД по типам извлекаемой информации

Большинство авторов приводит классификацию задач ИАД по типам производимой информации [2, 21, 49]. Следующие пять видов называются всеми без исключений.

1. Классификация. Наиболее распространенная задача ИАД. Она позволяет выявить признаки, характеризующие однотипные группы объектов - классы, - для того чтобы по известным значениям этих характеристик можно было отнести новый объект к тому или иному классу. Ключевым моментом выполнения этой задачи является анализ множества классифицированных объектов. Наиболее типичный пример использования классификации - конкурентная борьба между поставщиками товаров и услуг за определенные группы клиентов. Классификация способна помочь определить характеристики неустойчивых клиентов, склонных перейти к другому поставщику, что позволяет найти оптимальную стратегию их удержания от этого шага (например, посредством предоставления скидок, льгот или даже с помощью индивидуальной работы с представителями "групп риска"). В качестве методов решения задачи классификации могут использоваться алгоритмы типа Lazy-Learning [63], в том числе известные алгоритмы ближайшего соседа (Nearest Neighbor) и k-ближайшего соседа (k-Nearest Neighbor) [22, 23, 28], байесовские сети (Bayesian Networks) [40], индукция деревьев решений [5, 20, 27], индукция символьных правил [37, 54], нейронные сети [19].
2. Кластеризация. Логически продолжает идею классификации на более сложный случай, когда сами классы не предопределены. Результатом использования метода, выполняющего кластеризацию, как раз является определение (посредством свободного поиска) присущего исследуемым данным разбиения на группы. Так, можно выделить родственные группы клиентов или покупателей с тем, чтобы вести в их отношении дифференцированную политику. В приведенном выше примере "группы риска" - категории клиентов, готовых уйти к другому поставщику - средствами кластеризации могут быть определены до начала процесса ухода, что позволит производить профилактику проблемы, а не экстренное исправление положения. В большинстве случаев кластеризация очень субъективна; будучи основанным на измерении "информационного расстояния" между примерами обучающего множества (подобно использованному в методе k-ближайшего соседа), любой вариант разбиения на кластеры напрямую зависит от выбранной меры этого расстояния. В качестве примера используемых методов можно привести обучение "без учителя" особого вида нейронных сетей - сетей Кохонена [19], а также индукцию правил [37].
3. Выявление ассоциаций. В отличие от двух предыдущих типов, ассоциация определяется не на основе значений свойств одного объекта или события, а имеет место между двумя

или несколькими одновременно наступающими событиями. При этом производимые правила указывают на то, что при наступлении одного события с той или иной степенью вероятности наступает другое. Количественно сила ассоциации определяется несколькими величинами; например, в системе MineSet [1] используется три характеристики:

а) предсказуемость (predictability) определяет, как часто события X и Y случаются вместе, в виде доли от общего количества событий X; например, в случае покупки телевизора (X) одновременно покупается видеомагнитофон в 65% случаев (Y);

б) распространенность (prevalence) показывает, как часто происходит одновременное наступление событий X и Y относительно общего числа моментов зафиксированных событий; иными словами, насколько часто производится одновременная покупка телевизора и видеомагнитофона среди всех сделанных покупок;

в) ожидаемая предсказуемость (expected predictability) показывает ту предсказуемость, которая сложилась бы при отсутствии взаимосвязи между событиями; например, как часто покупался бы видеомагнитофон безотносительно к тому, покупался ли телевизор.

Рассмотренный пример является типичной иллюстрацией задачи анализа покупательской корзины (basket analysis). Цель его выполнения - определение пар товаров, при совместной покупке которых покупателю может быть предоставлена скидка ради увеличения значения предсказуемости и, следовательно, повышения объема продаж.

4. Выявление последовательностей. Подобно ассоциациям, последовательности имеют место между событиями, но наступающими не одновременно, а с некоторым определенным разрывом во времени. Таким образом, ассоциация есть частный случай последовательности с нулевым временным лагом. Так, если видеомагнитофон не был куплен вместе с телевизором, то в течение месяца после покупки нового телевизора покупка видеомагнитофона производится в 51% случаев.
5. Прогнозирование. Это особая форма предсказания, которая на основе особенностей поведения текущих и исторических данных оценивает будущие значения определенных численных показателей. Например, может быть сделан прогноз объема продукции, который ожидается в предприятиях текстильной отрасли Ивановского региона в ближайшие месяцы, на основе данных, накопленных в базе социально-экономического положения области. В задачах подобного типа наиболее часто используются традиционные методы математической статистики, а также нейронные сети.

4.2. Классификация стадий ИАД

Согласно [49], ИАД состоит из двух стадий:

1) выявление закономерностей (свободный поиск);
2) использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование).

В дополнение к ним [52] вводит третью стадию - анализ исключений, предназначенный для выявления и толкования аномалий в найденных закономерностях.

Некоторые обзоры явно выделяют промежуточную стадию проверки достоверности найденных закономерностей между их нахождением и использованием (стадия валидации, [26]). Однако более логично считать этот этап частью первой стадии; так, широко распространен прием разделения всего множества известных примеров на обучающее и проверочное подмножества (например, при обучении нейронных сетей [19, 33, 43]), что является частью алгоритмов индукции закономерностей.

Итак, процесс ИАД может быть представлен следующими стадиями и выполняемыми в их рамках действиями (рис. 3) [52]. В этом контексте некоторые задачи ИАД оказываются разбитыми на два или три последовательных этапа, соответствующих стадиям ИАД (см.,

например, задачу "Прогнозирование" и действия "Выявление трендов и колебаний", "Прогнозирование развития процессов" и "Выявление отклонений").

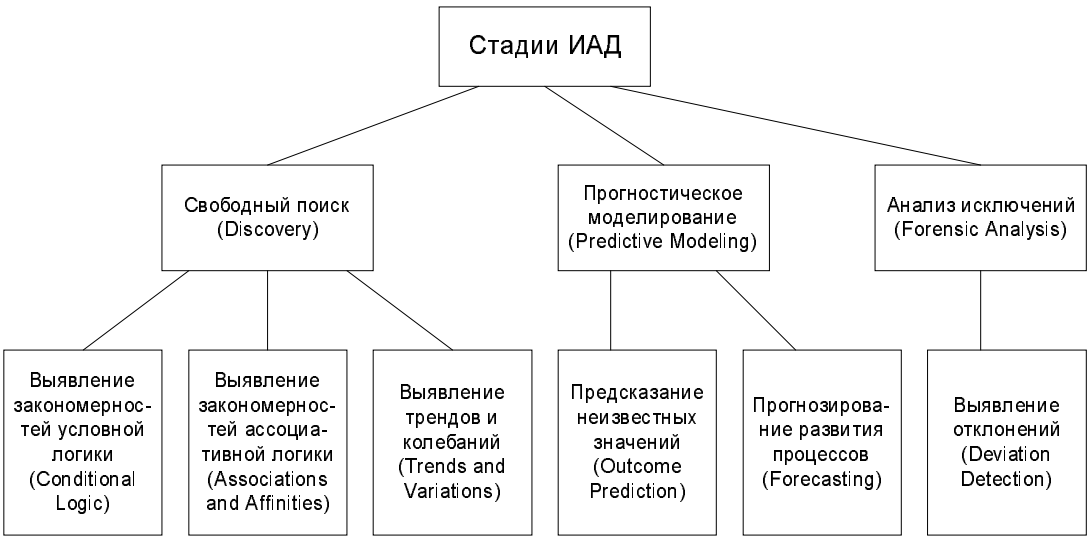

Рис. 3. Стадии ИАД.

4.2.1. Свободный поиск (Discovery)

Свободный поиск определяется как процесс исследования исходной БД на предмет поиска скрытых закономерностей без предварительного определения гипотез относительно вида этих закономерностей. Другими словами, сама программа берет на себя инициативу в деле поиска интересных аномалий, или шаблонов, в данных, освобождая аналитика от необходимости обдумывания и задания соответствующих запросов. Этот подход особенно ценен при исследовании больших баз данных, имеющих значительное количество скрытых закономерностей, большинство из которых было бы упущено при непосредственном поиске путем прямых запросов пользователя к исходным данным.

В качестве примера свободного поиска по инициативе системы рассмотрим исследование реестра физических лиц. Если инициатива принадлежит пользователю, он может построить запрос типа "Каков средний возраст директоров предприятий отрасли промышленности строительных материалов, расположенных в Иванове и находящихся в собственности субъекта Федерации?" и получить ответ - 48. В системе, обеспечивающей стадию свободного поиска, пользователь может поступить иначе и запросить у системы найти что-нибудь интересное относительно того, что влияет на атрибут Возраст. Система начнет действовать так же, как и аналитик-человек, т. е. искать аномалии в распределении значений атрибутов, в результате чего будет произведен список логических правил типа "ЕСЛИ ..., ТО ...", в том числе, например:

- ЕСЛИ Профессия="Программист", ТО Возраст≤30 в 61% случаев;
- ЕСЛИ Профессия="Программист", ТО Возраст≤60 в 98% случаев.

Аналогично, при исследовании реестра юридических лиц аналитика может заинтересовать атрибут Форма_собственности. В результате свободного поиска могут быть получены правила:

- ЕСЛИ Основной_вид_деятельности="Общеобразовательные детские школы", ТО Форма_собственности="Муниципальная собственность" в 84% случаев;
- ЕСЛИ Вид_деятельности="Наука и научное обслуживание", ТО Форма_собственности="Частная собственность" в 73% случаев.

Стадия свободного поиска может выполняться посредством:

1) индукции правил условной логики (как в приведенных примерах) - с их помощью, в частности, могут быть компактно описаны группы похожих обучающих примеров в задачах классификации и кластеризации;
2) индукции правил ассоциативной логики - то есть того, что было определено в рамках классификации задач ИАД по типам извлекаемой информации как выявление ассоциаций и последовательностей;
3) определения трендов и колебаний в динамических процессах, то есть исходного этапа задачи прогнозирования.

Стадия свободного поиска, как правило, должна включать в себя не только генерацию закономерностей, но и проверку их достоверности на множестве данных, не принимавшихся в расчет при их формулировании. Прием разделения исходных данных на обучающее множество (training set) и проверочное множество (test set) хорошо описан в методах обучения нейронных сетей [19]. Некоторые обзоры даже выделяют этап валидации моделей в отдельную стадию [26, 45].

4.2.2. Прогностическое моделирование (Predictive Modeling)

Здесь, на второй стадии ИАД, используются плоды работы первой, то есть найденные в БД закономерности применяются для предсказания неизвестных значений:

1) при классификации нового объекта мы можем с известной уверенностью отнести его к определенной группе результатов рассмотрения известных значений его атрибутов;
2) при прогнозировании динамического процесса результаты определения тренда и периодических колебаний могут быть использованы для вынесения предположений о вероятном развитии некоторого динамического процесса в будущем.

Возвращаясь к рассмотренным примерам, продолжим их на данную стадию. Зная, что некто Иванов - программист, можно быть на 61% уверенным, что его возраст ≤30 годам, и на 98% - что он ≤60 годам. Аналогично, можно сделать заключение о 84% вероятности того, что некоторое новое юридическое лицо будет находиться в муниципальной собственности, если его основной вид деятельности - "Общеобразовательные детские школы".

Следует отметить, что свободный поиск раскрывает общие закономерности, т. е. индуктивен, тогда как любой прогноз выполняет догадки о значениях конкретных неизвестных величин, следовательно, дедуктивен. Кроме того, результирующие конструкции могут быть как прозрачными, т. е. допускающими разумное толкование (как в примере с произведенными логическими правилами), так и нетрактуемыми - "черными ящиками" (например, про построенную и обученную нейронную сеть никто точно не знает, как именно она работает).

4.2.3. Анализ исключений (Forensic Analysis)

Предметом данного анализа являются аномалии в раскрытых закономерностях, то есть необъясненные исключения. Чтобы найти их, следует сначала определить норму (стадия свободного поиска), вслед за чем выделить ее нарушения. Так, определив, что 84% общеобразовательных школ отнесены к муниципальной форме собственности, можно задаться вопросом - что же входит в 16%, составляющих исключение из этого правила? Возможно, им найдется логическое объяснение, которое также может быть оформлено в виде закономерности. Но может также статься, что мы имеем дело с ошибками в исходных данных, и тогда анализ исключений может использоваться в качестве инструмента очистки сведений в хранилище данных [3].

4.3. Классификация технологических методов ИАД

Все методы ИАД подразделяются на две большие группы по принципу работы с исходными обучающими данными.

1. В первом случае исходные данные могут храниться в явном детализированном виде и непосредственно использоваться для прогностического моделирования и/или анализа исключений; это так называемые методы рассуждений на основе анализа прецедентов. Главной проблемой этой группы методов является затрудненность их использования на больших объемах данных, хотя именно при анализе больших хранилищ данных методы ИАД приносят наибольшую пользу.
2. Во втором случае информация вначале извлекается из первичных данных и преобразуется в некоторые формальные конструкции (их вид зависит от конкретного метода). Согласно предыдущей классификации, этот этап выполняется на стадии свободного поиска, которая у методов первой группы в принципе отсутствует. Таким образом, для прогностического моделирования и анализа исключений используются результаты этой стадии, которые гораздо более компактны, чем сами массивы исходных данных. При этом полученные конструкции могут быть либо "прозрачными" (интерпретируемыми), либо "черными ящиками" (нетрактуемыми).

Две эти группы и входящие в них методы представлены на рис. 4.

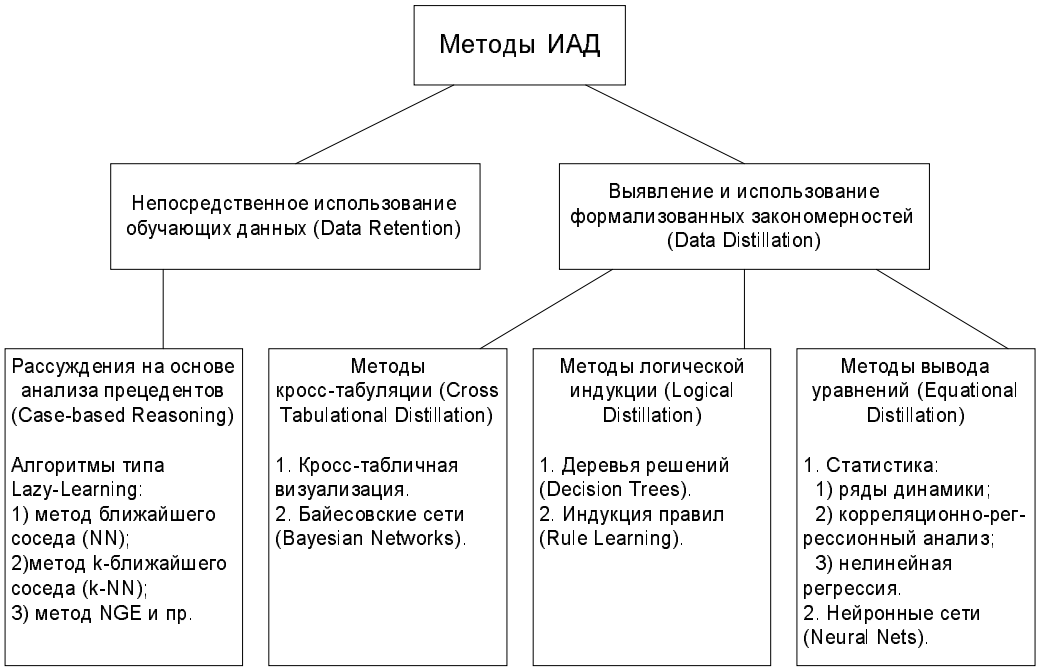

Рис. 4. Классификация технологических методов ИАД.

4.3.1. Непосредственное использование обучающих данных

Обобщенный алгоритм Lazy-Learning, относящийся к рассматриваемой группе, выглядит так (описание алгоритма взято из [63]). На вход классификатора подается пример $\mathbf{q}$, на выходе ожидается предсказание включающего его класса. Каждый пример $\mathbf{x}=\{x_1,x_2,\ldots,x_{|F|}\}$ представляется точкой в многомерном пространстве свойств (атрибутов) $\mathbf{F}$, принадлежащей некоторому классу $x_c \in \mathbf{J}$. Каждый атрибут принимает непрерывные

значения либо дискретные значения из фиксированного набора. Для примера q возвращается его наиболее вероятный класс.

Индивидуальной особенностью алгоритма k-ближайшего соседа является метод определения в нем апостериорной вероятности принадлежности примера классу:

$$p(c_j|q) = \frac{\sum_{x \in K_q} equal(x_c, c_j) \cdot K(d(x,q))}{\sum_{x \in K_q} K(d(x,q))}, \quad (1)$$

где $equal()$ возвращает 1, когда аргументы равны, или 0 в противном случае, $K()$ - функция близости, определяемая как

$$K(d(x,q)) = \frac{1}{d(x,q)}, \quad (2)$$

а K_q - множество k ближайших соседей q во множестве X известных обучающих примеров, близость которых к классифицируемому примеру определяется функцией расстояния $d()$. Метод k-ближайшего соседа рассчитывает расстояние от q до каждого $x \in X$ по формуле:

$$d(x,q) = \left(\sum_{f \in F} w(f) \cdot \delta(x_f, q_f)^r \right)^{\frac{1}{r}}, \quad (3)$$

причем чаще всего принимается $r=2$ (эвклидово пространство), а функция $\delta()$ в зависимости от типа атрибута определяется следующими способами:

$$\delta(x_f, q_f) = \begin{cases} |x_f - q_f|, & \text{если f - непрерывный атрибут} \\ 0, & \text{если f - дискретный атрибут и } x_f = q_f \\ 1, & \text{если f - дискретный атрибут и } x_f \neq q_f \end{cases}. \quad (4)$$

$w(f)$ является функцией веса атрибута f. В чистом алгоритме k-ближайшего соседа:

$$w(f) = w_f, \quad (5)$$

то есть эта функция считается константой.

Метод ближайшего соседа является частным случаем метода k-ближайшего соседа при $k=1$. Более сложные алгоритмы типа Lazy-Learning основываются на том же обобщенном алгоритме [22, 23, 28], но или иначе определяют апостериорные вероятности принадлежности примеров классам, или (как, например, Nested Generalized Exemplars Algoritm [22]) усложняют расчет функции $w(f)$.

Особенность этой группы методов состоит в том, что предсказание неизвестных значений выполняется на основе явного сравнения нового объекта (примера) с известными примерами. В случае большого количества обучающих примеров, чтобы не сканировать последовательно все обучающее множество для классификации каждого нового примера, иногда используется прием выборки относительно небольшого подмножества "типичных представителей" обучающих примеров, на основе сравнения с которыми и выполняется классификация. Однако, этим приемом следует пользоваться с известной осторожностью, так как в выделенном подмножестве могут не быть отражены некоторые существенные закономерности.

Что касается самого известного представителя этой группы - метода k-ближайшего соседа, - он более приспособлен к тем предметным областям, где атрибуты объектов имеют преимущественно численный формат, так как определение расстояния между примерами в этом случае является более естественным, чем для дискретных атрибутов.

4.3.2. Выявление и использование формализованных закономерностей

Методы этой группы извлекают общие зависимости из множества данных и позволяют затем применять их на практике. Они отличаются друг от друга:

1) по типам извлекаемой информации (которые определяются решаемой задачей - см. классификацию задач ИАД выше);
2) по способу представления найденных закономерностей.

Формализм, выбранный для выражения закономерностей, позволяет выделить три различных подхода, каждый из которых уходит своими корнями в соответствующие разделы математики:

1) методы кросс-табуляции;
2) методы логической индукции;
3) методы вывода уравнений.

Логические методы наиболее универсальны в том смысле, что могут работать как с численными, так и с другими типами атрибутов. Построение уравнений требует приведения всех атрибутов к численному виду, тогда как кросс-табуляция, напротив, требует преобразования каждого численного атрибута в дискретное множество интервалов.

Методы кросс-табуляции

Кросс-табуляция является простой формой анализа, широко используемой в генерации отчетов средствами систем оперативной аналитической обработки (OLAP). Двумерная кросс-таблица представляет собой матрицу значений, каждая ячейка которой лежит на пересечении значений атрибутов. Расширение идеи кросс-табличного представления на случай гиперкубической информационной модели является, как уже говорилось, основой многомерного анализа данных, поэтому эта группа методов может рассматриваться как симбиоз многомерного оперативного анализа и интеллектуального анализа данных.

Кросс-табличная визуализация является наиболее простым воплощением идеи поиска информации в данных методом кросс-табуляции. Строго говоря, этот метод не совсем подходит под отмеченное свойство ИАД - переход инициативы к системе в стадии свободного поиска. На самом деле кросс-табличная визуализация является частью функциональности OLAP. Здесь система только предоставляет матрицу показателей, в которой аналитик может увидеть закономерность. Но само предоставление такой кросс-таблицы имеет целью поиск "шаблонов информации" в данных для поддержки принятия решений, то есть удовлетворяет приведенному определению ИАД. Поэтому неслучайно, что множество авторов [1, 8, 21] все же относит кросс-табличную визуализацию к методам ИАД.

К методам ИАД группы кросс-табуляции относится также использование байесовских сетей (Bayesian Networks) [40], в основе которых лежит теорема Байеса теории вероятностей для определения апостериорных вероятностей составляющих полную группу попарно несовместных событий Y_i по их априорным вероятностям:

$$P(Y_i|X) = \frac{P(Y_i) \cdot P(X|Y_i)}{P(X)}. \qquad (6)$$

Байесовские сети активно использовались для формализации знаний экспертов в экспертных системах [41], но с недавних пор стали применяться в ИАД для извлечения знаний из данных.

Heckerman [40] отмечает четыре достоинства байесовских сетей как средства ИАД:

1) поскольку в модели определяются зависимости между всеми переменными, легко обрабатываются ситуации, когда значения некоторых переменных неизвестны;

2) построенные байесовские сети просто интерпретируются и позволяют на этапе прогностического моделирования легко производить анализ по сценарию "что - если";
3) подход позволяет естественным образом совмещать закономерности, выведенные из данных, и фоновые знания, полученные в явном виде (например, от экспертов);
4) использование байесовских сетей позволяет избежать проблемы переподгонки (overfitting), то есть избыточного усложнения модели, чем страдают многие методы (например, деревья решений и индукция правил) при слишком буквальном следовании распределению зашумленных данных.

Так называемый наивно-байесовский подход (naive-bayes approach) [28] является наиболее простым вариантом метода, использующего байесовские сети. Он используется в задачах классификации и производит "прозрачные" модели. Общий вид алгоритма таков. Пусть X - множество атрибутов рассматриваемых объектов $\{X^1, X^2, \dots, X^n\}$, а Y - переменная, по которой проводится классификация объектов. Для зависимой переменной Y вычисляются априорные вероятности каждого ее возможного значения $P(Y_i)$, как отношения числа соответствующих этим значениям примеров обучающего множества к общему числу примеров в нем (события Y_i попарно несовместны). Затем для каждой независимой переменной X^k определяются условные вероятности выпадания каждого ее возможного значения при определенном значении зависимой переменной $P(X_j^k | Y_i)$, в результате чего образуется матрица условных вероятностей, которая графически может быть представлена в виде бинарного графа. При поданном на вход классификатора новом примере $\{x^1, x^2, \dots, x^n\}$ и при условии статистической независимости атрибутов формула Байеса принимает вид:

$$P(Y_i | X^1 = x^1 \cap X^2 = x^2 \cap \dots \cap X^n = x^n) =$$

$$\frac{P(Y_i) \cdot P(X^1 = x^1 | Y_i) \cdot P(X^2 = x^2 | Y_i) \cdot \dots \cdot P(X^n = x^n | Y_i)}{P(X^1 = x^1 \cap X^2 = x^2 \cap \dots \cap X^n = x^n)} \cdot \qquad (7)$$

Знаменатель полученного выражения не будет изменяться при вычислении условных вероятностей возможных значений Y. Поэтому для каждого значения зависимой переменной Y_i вычисляется числитель формулы Байеса как произведение безусловной вероятности $P(Y_i)$ на условные вероятности входящих в пример значений независимых переменных при данном значении зависимой переменной. Результатом классификации является то значение переменной Y, при котором получено максимальное значение этого произведения.

Несмотря на свою простоту, скорость и интерпретируемость результатов, наивно-байесовский алгоритм имеет недостатки:

1) перемножать условные вероятности корректно только тогда, когда все входные переменные действительно статистически независимы; допущение этой независимости и обуславливает приставку наивно- в названии алгоритма, хотя, по приведенным в [28] примерам, он показывает неплохие практические результаты даже при несоблюдении условия статистической независимости; корректно данная ситуация обрабатывается только более сложными методами, основанными на обучении байесовских сетей [35, 40];
2) невозможна непосредственная обработка непрерывных переменных - их требуется разбивать на множество интервалов, чтобы атрибуты были дискретными; такое разбиение в ряде случаев приводит к потере значимых закономерностей [28];

3) наивно-байесовский подход учитывает только индивидуальное влияние входных переменных на результат классификации, не принимая во внимание комбинированного влияния пар или троек значений разных атрибутов [28], что было бы полезно с точки зрения прогностической точности, но значительно увеличило бы количество проверяемых комбинаций.

Несмотря на это, байесовские сети предлагают простой наглядный подход ИАД и широко используются на практике.

Методы логической индукции

Методы данной группы являются, пожалуй, наиболее выразительными, в большинстве случаев оформляя найденные закономерности в максимально "прозрачном" виде. Кроме того, производимые правила, в общем случае, могут включать как непрерывные, так и дискретные атрибуты. Результатами применения логической индукции могут быть построенные деревья решений или произведенные наборы символьных правил. Сравнительному анализу деревьев решений и символьных правил и посвящен настоящий раздел.

Деревья решений

Деревья решений [59, 5, 20] являются упрощенной формой индукции логических правил, рассматриваемой ниже. Основная идея их использования заключается в последовательном разделении обучающего множества на основе значений выбранного атрибута, в результате чего строится дерево, содержащее:

1) терминальные узлы (узлы ответа), задающие имена классов;
2) нетерминальные узлы (узлы решения), включающие тест для определенного атрибута с ответвлением к поддереву решений для каждого значения этого атрибута.

В таком виде дерево решений определяет классификационную процедуру естественным образом: любой объект связывается с единственным терминальным узлом. Эта связь начинается с корня, проходит путь по дугам, которым соответствуют значения атрибутов, и доходит до узла ответа с именем класса. Примем допущение, что примеры относятся к двум классам - положительному и отрицательному (в общем случае классов может быть больше).

Поскольку исходные данные для индукции часто бывают зашумлены, наилучшим решением с точки зрения прогностической точности является не полное дерево решений, объясняющее все примеры обучающего множества, а упрощенное, в котором некоторые поддеревья свернуты в терминальные узлы. Процесс упрощения, или подрезания (pruning), построенного полного дерева имеет целью избежание переподгонки (overfitting), то есть избыточного усложнения, которое может оказаться следствием излишне буквального следования зашумленным данным [5, 37].

После подрезания дерева его различные терминальные узлы оказываются на разных уровнях, то есть путь к ним включает разное количество проверок значений атрибутов; другими словами, для прихода в терминальные узлы, лежащие на высоких уровнях дерева, значения многих атрибутов вообще не рассматриваются. Поэтому при построении деревьев решений порядок тестирования атрибутов в узлах решения имеет решающее значение.

Стратегия, применяемая в алгоритмах индукции деревьев решений, называется стратегией разделения и захвата (divide-and-conquer), в противовес стратегии отделения и захвата (separate-and-conquer), на которой построено большое количество алгоритмов индукции правил. Quinlan [59] описал следующий алгоритм разделения и захвата.

Пусть:

A - множество атрибутов $a_i \in A$, $1 \le i \le |A|$;

V_i - множество возможных значений атрибута a_i, $v_{ij} \in V_i$, $1 \le j \le |V_i|$[6].

Тогда:

1) если все обучающие примеры принадлежат одному классу, то дерево решений есть терминальный узел, содержащий имя этого класса;
2) в противном случае следует:

а) определить атрибут с наименьшей E-оценкой a_{best};

б) для каждого значения $v_{best,i}$ атрибута a_{best} провести ветвь к поддереву решений, рекурсивно строящемуся на основе примеров со значением $v_{best,i}$ атрибута a_{best}.

Quinlan предложил вычислять E-оценку следующим образом. Пусть для текущего узла:

p - число положительных примеров;

n - число отрицательных примеров;

p_{ij} - число положительных примеров со значением v_{ij} для a_i;

n_{ij} - число отрицательных примеров со значением v_{ij} для a_i.

Тогда:

$$E(a_i) = \sum_{j=1}^{|V_i|} \frac{p_{ij} + n_{ij}}{p + n} I(p_{ij}, n_{ij}), \tag{8}$$

где

$$I(x, y) = \begin{cases} 0, \text{ если } x = 0 \\ 0, \text{ если } y = 0 \\ -\frac{x}{x+y}\log_2\frac{x}{x+y} - \frac{y}{x+y}\log_2\frac{y}{x+y}, \text{ если } x > 0 \text{ и } y > 0 \end{cases}. \tag{9}$$

E-оценка - это теоретико-информационная мера, основанная на энтропии. Она показывает меру неопределенности в классификации, возникающей при использовании рассматриваемого атрибута в узле решения. Поэтому считается, что наибольшую классифицирующую силу имеет атрибут с наименьшей E-оценкой. Однако, определенная рассмотренным образом E-оценка имеет и недостатки: она, в частности, предоставляет при построении дерева преимущество атрибутам с большим количеством значений. Поэтому в некоторых работах [5, 37] предложены модификации E-оценки, устраняющие эти недостатки.

Подрезание дерева решений для улучшения прогностической точности при классификации новых примеров обычно производят над построенным полным деревом, то есть выполняют процедуру поступрощения. Двигаясь снизу-вверх, заменяют узлы решения с соответствующими поддеревьями терминальными узлами до тех пор, пока не будет оптимизирована заданная эвристическая мера.

Индукция правил

Популярность деревьев решений проистекает из быстроты их построения и легкости использования при классификации. Более того, деревья решений могут быть легко преобразованы в наборы символьных правил - генерацией одного правила из каждого пути от корня к терминальной вершине. Однако, правила в таком наборе будут неперекрывающимися,

---

[6] Таким образом, области определения непрерывных атрибутов для построения деревьев решений также должны быть разбиты на конечное множество интервалов.

потому что в дереве решений каждый пример может быть отнесен к одному и только к одному терминальному узлу. Более общим (и более реальным) является случай существования теории, состоящей из набора неиерархических перекрывающихся символьных правил. Значительная часть алгоритмов, выполняющих индукцию таких наборов правил, объединяются стратегией отделения и захвата (separate-and-conquer), или покрывания (covering) [37], начало которой положили работы R. Michalski [46, 47]. Термин "отделение и захват" сформулировали Pagallo и Haussler [51], охарактеризовав эту стратегию индукции следующим образом:

1) произвести правило, покрывающее часть обучающего множества;
2) удалить покрытые правилом примеры из обучающего множества (отделение);
3) последовательно обучиться другим правилам, покрывающим группы оставшихся примеров (захват), пока все примеры не будут объяснены.

```
procedure SEPARATEANDCONQUER (Examples)
Theory = ∅
while POSITIVE (Examples) ≠ ∅
        Rule = FINDBESTRULE (Examples)
        Covered = COVER (Rule, Examples)
        if RULESTOPPINGCRITERION (Theory, Rule, Examples)
                exit while
        Examples = Examples \ Covered
        Theory = Theory ∪ Rule
Theory = POSTPROCESS (Theory)
return (Theory)

procedure FINDBESTRULE (Examples)
InitRule = INITIALIZERULE (Examples)
InitVal = EVALUATERULE (InitRule)
BestRule = <InitVal, InitRule>
Rules = {BestRule}
while Rules ≠ ∅
        Candidates = SELECTCANDIDATES (Rules, Examples)
        Rules = Rules \ Candidates
        for Candidate ∈ Candidates
                Refinements = REFINERULE (Candidate, Examples)
                for Refinement ∈ Refinements
                        Evaluation = EVALUATERULE (Refinement, Examples)
                        unless STOPPINGCRITERION
                                (Refinement, Evaluation, Examples)
                                NewRule = <Evaluation, Refinement>
                                Rules = INSERTSORT (NewRule, Rules)
                                if NewRule > BestRule
                                        BestRule = NewRule
        Rules = FILTERRULES (Rules, Examples)
return (BestRule)
```

Рис. 5. Общий алгоритм отделения и захвата для индукции правил.

Рис. 5 показывает общий алгоритм индукции правил методом отделения и захвата [37]. Разные варианты реализации вызываемых в общем алгоритме подпрограмм определяют разнообразие известных методов отделения и захвата.

Алгоритм SEPARATEANDCONQUER начинается с пустой теории. Если в обучающем множестве есть положительные примеры, вызывается подпрограмма FINDBESTRULE для извлечения правила, покрывающего часть положительных примеров. Все покрытые примеры отделяются затем от обучающего множества, произведенное правило включается в теорию, и следующее правило ищется на оставшихся примерах. Правила извлекаются до тех пор, пока не останется положительных примеров или пока не сработает критерий остановки RULESTOPPINGCRITERION. Зачастую полученная теория подвергается постобработке POSTPROCESS.

Процедура FINDBESTRULE ищет в пространстве гипотез правило, которое оптимизирует выбранный критерий качества, описанный в EVALUATERULE. Значение этой эвристической функции, как правило, тем выше, чем больше положительных и меньше отрицательных примеров покрыто правилом-соискателем (candidate rule). FINDBESTRULE обрабатывает *Rules*, упорядоченный список правил-соискателей, порожденных процедурой INITIALIZERULE.

Новые правила всегда вставляются в нужные места (INSERTSORT), так что *Rules* постоянно остается списком, упорядоченным по убыванию эвристических оценок правил. В каждом цикле SELECTCANDIDATES отбирает подмножество правил-соискателей, которое затем очищается в REFINERULE. Каждый результат очистки оценивается и вставляется в отсортированный список *Rules*, если только STOPPINGCRITERION не предотвращает это. Если оценка *NewRule* лучше, чем у лучшего из ранее найденных правил, значение *NewRule* присваивается переменной *BestRule*. FILTERRULES отбирает подмножество упорядоченного списка правил, предназначенное для использования в дальнейших итерациях. Когда все правила-соискатели обработаны, наилучшее правило возвращается.

Основной проблемой, стоящей перед алгоритмами индукции правил, остается избежание переподгонки при использовании зашумленных данных. Средства избежания переподгонки в алгоритмах отделения и захвата могут обрабатывать шум:

1) на этапе генерирования правил - предупрощение (pre-pruning), реализуемое подпрограммами RULESTOPPINGCRITERION и STOPPINGCRITERION обобщенного алгоритма (их основная идея - остановить генерацию правила или теории, которые еще достаточно общи; таким образом, допускается покрытие ими некоторых отрицательных примеров, если их текущий вид уже считается критерием остановки достаточно ценным); и/или
2) на стадии постобработки построенной теории - поступрощение (post-pruning), подпрограмма POSTPROCESS общего алгоритма (эти методы заимствуют идею подрезки созданного дерева решений [60]).

Fürnkranz [37] классифицирует алгоритмы отделения и захвата по типам производимых концептов.

1. Наборы правил-высказываний (Propositional rule sets). Классические алгоритмы отделения и захвата индуцируют наборы правил на основе троек (атрибут, оператор, значение).
2. Списки решений (Decision lists). Более усложненные алгоритмы обобщают первый подход до индукции упорядоченных наборов правил (называемых списками решений). Такие алгоритмы чаще всего применяются для решения многоклассовых задач, где варианты классификации примеров не ограничиваются определением их положительности или отрицательности.
3. Регрессионные правила (Regression rules). Индукция регрессионных правил позволяет решать задачи с непрерывными классовыми переменными [44]; здесь мы имеем дело с

пересечением сфер влияния методов логической индукции и методов вывода уравнений, разделенных на рис. 4 в разных группах.

4. Логические программы (Logic programs). Исследования в области индуктивно-логического программирования породили ряд алгоритмов отделения и захвата, способных решать задачи в терминах более богатого языка логики предикатов первого порядка. На выходе таких алгоритмов могут быть получены программы на Прологе.

Стратегия отделения и захвата не является единственным способом индукции символьных правил. Например, Quinlan предложил алгоритм C4.5 [57, 58] индукции деревьев решений, который позволяет генерировать компактные списки решений из деревьев решений посредством перевода деревьев решений во множество неперекрывающихся правил, упрощения этих правил и упорядочивания полученных перекрывающихся правил в список решений. В качестве средства генерации правил могут применяться и генетические алгоритмы [52, 8], воплощающие идею мутаций и отбора лучших правил. Ряд других альтернативных алгоритмов перечислен в [37].

Сравнение возможностей деревьев решений и индукции правил

Индукция правил и деревья решений, будучи способами решения одной задачи, значительно отличаются по своим возможностям. Несмотря на широкую распространенность деревьев решений, индукция правил по ряду причин, отмеченных в [37, 52, 54], представляется более предпочтительным подходом.

1. Деревья решений часто довольно сложны и тяжелы для понимания. Quinlan [57] заметил, что даже усеченные деревья решений могут быть чересчур громоздкими, усложненными и запутанными, чтобы обеспечить прямой доступ к знаниям, и поэтому изобретал процедуры упрощения деревьев решений путем приведения их к ограниченным наборам порождающих правил (продукций) [58, 57].
2. Непременное требование неперекрываемости правил в алгоритмах обучения деревьев решений навязывает жесткое ограничение на возможность выражения существующих закономерностей. Одна из проблем, вытекающих из этого ограничения - проблема дублированного поддерева [51]. Часто случается, что идентичные поддеревья оказываются в процессе обучения в разных местах дерева решений вследствие фрагментации пространства исходных примеров, обязательной по ограничению на неперекрываемость правил. Индукция отделения и захвата не ставит такого ограничения и, следовательно, менее чувствительна к этой проблеме. Например, можно легко проверить, что записанный в виде двух правил концепт:

   ```
   x :- a = 3, b = 3.
   x :- c = 3, d = 3.
   ```

   имеет минимальное эквивалентное дерево решений, содержащее 10 внутренних узлов и 21 терминальный при предположении, что атрибуты a ... d имеют каждый по три возможных значения [29]. Следовательно, из-за того, что порядок проверок значений различных атрибутов при спуске к терминальной вершине по дереву решений строго последователен, многие простые закономерности в них оказываются по существу потерянными, поскольку разделяются на дублированные поддеревья и затем чаще всего удаляются на стадии подрезания, вследствие чего прогностическая точность классификации таким деревом решений значительно ухудшается по сравнению с набором правил.
3. Построение деревьев решений затруднено при большом количестве исходной информации (что чаще всего имеет место при интеллектуальном анализе хранилищ данных). Для решения этой проблемы часто выделяют относительно небольшое подмножество имеющихся обучающих примеров и на его основе сооружают дерево решений. Как показано в

[54], такой подход во многих случаях приводит к потере информации, скрытой в проигнорированных при индукции примерах.

С другой стороны, индукция правил осуществляется значительно более сложными (и медленными) алгоритмами, чем индукция деревьев решений. Особенно большие трудности возникают с поступрощением построенной теории, в отличие от простоты подрезания деревьев решений, на что обратил внимание Fürnkranz [37]: отсечение ветвей в дереве решений никогда не затронет соседние ветви, тогда как отсечение условий правила оказывает влияние на все перекрывающиеся с ним правила (рис. 6).

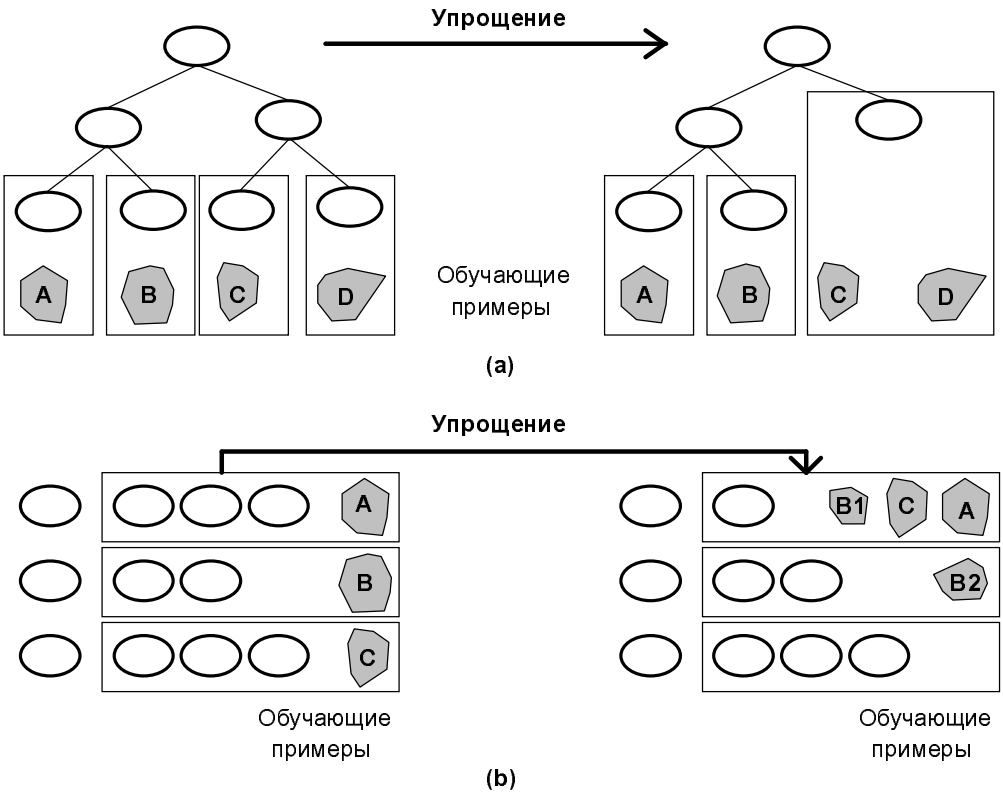

Рис. 6. Поступрощение в обучающих алгоритмах
(a) разделения и захвата и (b) отделения и захвата.

Рис. 6(a) иллюстрирует работу поступрощения в индукции деревьев решений. Правая половина переусложненного дерева покрывается множествами C и D обучающих примеров. Когда упрощающий алгоритм решает отсечь эти две терминальные вершины, порождающий их узел становится терминальным, который теперь покрывается примерами $C \cup D$. Левая ветвь дерева решений не затронута данной операцией.

С другой стороны, отсечение условий от правила означает его обобщение, то есть в новом виде оно будет покрывать больше положительных и больше отрицательных примеров. Следовательно, эти дополнительные положительные и отрицательные примеры должны быть исключены из обучающего множества, дабы не воздействовать на индукцию последующих правил. В случае на рис. 6(b) первое из трех правил упрощается и начинает покрывать не только примеры, покрываемые оригинальной версией, но и все примеры, которые покрывает третье правило, а также часть примеров, которые покрывает второе правило. Если третье правило после этого может быть просто удалено алгоритмом поступрощения,

то ситуация с оставшимся множеством примеров B2 не такая простая. Второе правило, естественно, покрывает все примеры множества B2, потому что оно было произведено для покрытия примеров включающего его множества B. Однако вполне может статься, что другое правило окажется более подходящим для отделения положительных примеров B2 от оставшихся отрицательных примеров. Корректная обработка таких ситуаций требует тесной интеграции процессов предупрощения и поступрощения, значительно усложняющей алгоритм индукции правил и ухудшающей его производительность [37].

Следовательно, исходя из проведенного сравнения, можно заключить, что построение деревьев решений оправдано в несложных задачах при небольшом количестве исходной информации благодаря простоте и быстроте их индукции. Однако при анализе больших объемов данных, накопленных в хранилищах, использование методов индукции правил предпочтительнее, несмотря на их относительную сложность.

Методы вывода уравнений

Методы вывода уравнений пытаются выразить закономерности, скрытые в данных, в форме математических выражений. Поэтому они способны работать только с атрибутами численного типа, тогда как другие атрибуты должны быть искусственно закодированы численными значениями. Отсюда вытекает несколько проблем, ограничивающих использование этих методов на практике. Тем не менее, они широко применяются во многих приложениях.

Статистика

Классические методы статистического анализа применяются в средствах ИАД чаще всего для решения задачи прогнозирования.

1. Выявление тенденций динамических рядов. Тенденцию среднего уровня можно представить в виде графика или аналитической функции, вокруг значения которой варьируют фактические значения уровней исследуемого процесса. Часто тенденции среднего уровня называют детерминированной компонентой процесса, и соответствующий динамический ряд выражается уравнением $x_t = \tilde{y}(t) + \varepsilon_t$, где x_t - уровень ряда в момент времени t, $\tilde{y}(t)$ - детерминированная компонента ряда, ε_t - случайная компонента. Детерминированная компонента обычно представляется достаточно простой аналитической функцией - линейной, параболической, гиперболической, экспоненциальной, - параметры которой подбираются согласно историческим данным для лучшей аппроксимации исторических данных.
2. Гармонический анализ. Во многих случаях сглаживание рядов динамики с помощью определения тренда не дает удовлетворительных результатов, так как в остатках ε_t наблюдается автокорреляция. Причиной автокоррелированности остатков могут быть нередко встречающиеся в рядах динамики заметные периодические колебания относительно выделенной тенденции. В таких случаях следует прибегать к гармоническому анализу, то есть к выделению из динамического ряда периодической составляющей. По результатам выделения из динамического ряда тренда и периодической составляющей может выполняться статистический прогноз процесса по принципу экстраполяции, по предположению, что параметры тренда и колебаний сохранятся для прогнозируемого периода [7, С. 304].
3. Корреляционно-регрессионный анализ. В отличие от функциональной (жестко детерминированной) связи, статистическая (стохастически детерминированная) связь между переменными имеет место тогда, когда с изменением значения одной из них вторая может в

определенных пределах принимать любые значения с некоторыми вероятностями, но ее среднее значение или иные статистические характеристики изменяются по определенному закону [7, С. 191-192]. Частным случаем статистической связи, когда различным значениям одной переменной соответствуют различные средние значения другой, является корреляционная связь. В соответствии с сущностью корреляционной связи ее изучение имеет две цели:

1) измерение параметров уравнения, выражающего связь средних значений зависимых переменных со значениями независимой переменной (зависимость средних значений результативного признака от значений факторных признаков);

2) измерение тесноты связи признаков между собой [7, С. 195-196].

Метод корреляционно-регрессионного анализа хорошо изучен [7, 19, 29] и широко применяется на практике. Однако, он имеет ряд ограничений:

1) для обеспечения достаточной точности и надежности число наблюдений должно быть в десятки или сотни раз больше числа факторов, чтобы закон больших чисел, действуя в полную силу, обеспечил эффективное взаимопогашение случайных отклонений от закономерного характера связи признаков;

2) для надежного выражения закономерности по средней величине требуется достаточно качественная однородность совокупности, чтобы параметры корреляции не были извращены; кроме того, иногда в качестве условия корреляционного анализа выдвигают необходимость подчинения распределения совокупности по результативному и факторным признакам нормальному закону распределения вероятностей (это условие связано с применением метода наименьших квадратов при расчете параметров корреляции - только при нормальном распределении он дает оценку параметров, отвечающую принципам максимального правдоподобия), хотя на практике даже при приближенном выполнении этой предпосылки метод наименьших квадратов дает неплохие результаты [9, С. 14];

3) метод корреляционно-регрессионного анализа не может объяснить роли факторных признаков в создании результативного признака [7, С. 198];

4) интерпретировать корреляционные показатели следует лишь в терминах вариаций результативного и факторного признаков; если же задача состоит в измерении связи между изменениями признаков объекта во времени, то метод корреляционно-регрессионного анализа требует значительных изменений (требуется исследование корреляции рядов динамики) [4; 7, С. 307-313].

Получаемые в результате применения анализа корреляционно-регрессионные модели (КРМ) обычно достаточно хорошо интерпретируемы и могут использоваться в прогностическом моделировании. Но, как отмечается в [8], невозможно применять этот вид анализа, не имея глубоких знаний в области статистики. Теоретическая подготовка аналитика играет здесь особенно важную роль, поэтому немногие существующие средства ИАД предлагают метод корреляционно-регрессионного анализа в качестве одного из инструментов обработки данных.

4. Корреляция рядов динамики. Проблема изучения причинных связей во времени очень сложна, и полное решение всех задач такого изучения до сих пор не разработано [7, С. 307]. Основная сложность состоит в том, что при наличии тренда за достаточно длительный промежуток времени большая часть суммы квадратов отклонений связана с трендом; при этом, если два признака имеют тренды с одинаковым направлением изменения уровней, то это вовсе не будет означать причинной зависимости. Следовательно, чтобы получить реальные показатели корреляции, необходимо абстрагироваться от искажающего влияния трендов - вычислить отклонения от трендов и измерить корреляцию колебаний (подробному рассмотрению этого подхода посвящена полностью работа [4]). Однако, не

всегда допустимо переносить выводы о тесноте связи между колебаниями на связь рядов динамики в целом (согласно приведенному в [7, С. 312] примеру, правомерно рассматривать связь между колебаниями урожайности и колебаниями суммы выпавших за лето осадков, но связь между урожайностью и дозой удобрений нельзя свести только к корреляции колебаний).

Нейронные сети

Искусственные нейронные сети как средство обработки информации моделировались по аналогии с известными принципами функционирования биологических нейронных сетей. Их структура базируется на следующих допущениях [33, С. 3]:

1) обработка информации осуществляется во множестве простых элементов - нейронов;
2) сигналы между нейронами передаются по связям от выходов ко входам;
3) каждая связь характеризуется весом, на который умножается передаваемый по ней сигнал;
4) каждый нейрон имеет активационную функцию (как правило, нелинейную), аргумент которой рассчитывается как сумма взвешенных входных сигналов, а результат считается выходным сигналом.

Таким образом, нейронные сети представляют собой наборы соединенных узлов, каждый из которых имеет вход, выход и активационную функцию (как правило, нелинейную) (рис. 7). Они обладают способностью обучаться на известном наборе примеров обучающего множества. Обученная нейронная сеть представляет собой "черный ящик" (нетрактуемую или очень сложно трактуемую прогностическую модель), которая может быть применена в задачах классификации, кластеризации и прогнозирования [19].

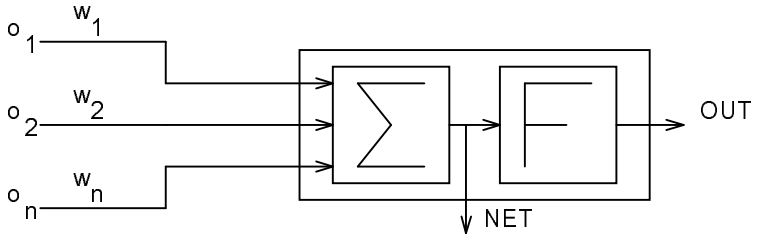

Рис. 7. Нейрон с активационной функцией F; $NET = \sum_{i=1}^{n} o_i w_i;\ OUT = F(NET).$

Обучение нейронной сети заключается в подстройке весовых коэффициентов, связывающих выходы одних нейронов со входами других. Обучение сети может производиться по одному из двух базовых сценариев:

1) обучение с учителем (supervised training) - наиболее типичный случай, когда для каждого вектора значений входных переменных примера обучающего множества известен желаемый вектор значений выходных переменных; такой способ обучения применяется в задачах классификации и прогнозирования;
2) обучение без учителя (unsupervised learning) - механизм настройки весов сети в случае, когда известны только значения входных переменных примеров обучающего множества; обученные таким способом нейронные сети выполняют задачу кластеризации (для этого используются, например, самоорганизующиеся карты Кохонена [33, С. 169-172], сети ART - адаптивной резонансной теории [19, С. 141-165]).

Чаще всего средства ИАД используют специальный тип нейронных сетей, обучаемых "с учителем", - многослойные персептроны [8, С. 42; 19, С. 54-55]. На рис. 8 изображена такая нейронная сеть с двумя слоями нейронов, имеющая три входных и три выходных переменных (в общем случае количество входов, количество выходов, число слоев и число нейронов в каждом внутреннем слое могут быть какими угодно). Выход каждого нейрона предыдущего слоя соединен со входом каждого нейрона последующего слоя.

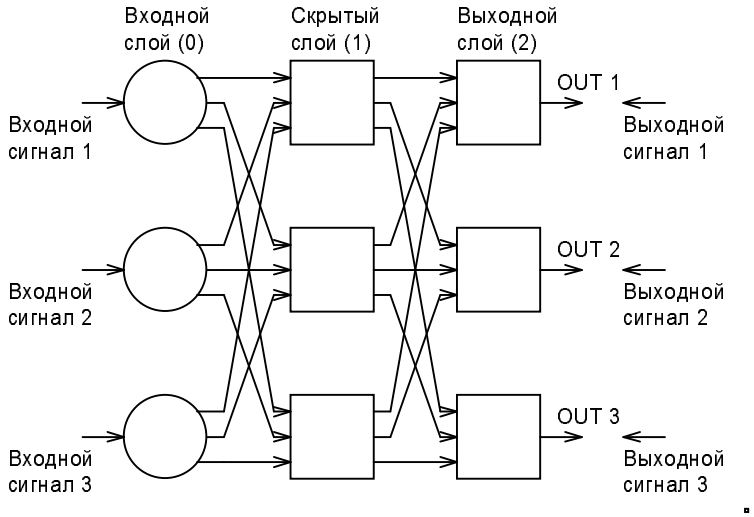

Рис. 8. Многослойный персептрон, обучаемый процедурой обратного распространения ошибки.

Настройка весов многослойного персептрона осуществляется алгоритмом обратного распространения ошибки [19, С. 56-69]. При обучении предполагается, что для каждого входного вектора (множества входов) существует парный ему целевой вектор (множество выходов), и вместе они образуют обучающую пару (пример). Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом, для преодтвращения патологических случаев невозможности обучения. Все множество обучающих пар составляет обучающее множество. Обучение сети требует выполнения следующих операций:

1) выбрать обучающую пару из обучающего множества;
2) подать входной вектор обучающей пары на вход сети;
3) вычислить выход сети;
4) вычислить разность между выходом сети и целевым вектором обучающей пары;
5) скорректировать веса сети, чтобы минимизировать ошибку;
6) повторять шаги 1-5 для каждой пары обучающего множества до тех пор, пока ошибка на всем множестве не достигнет допустимого уровня.

Обучение методом обратного распространения ошибки ведется послойно, начиная от выходного слоя, шагами 4 и 5.

Являясь "универсальными аппроксиматорами", персептроны могут обучиться достаточно сложным закономерностям в отличие от регрессионных моделей, в которых вид аппроксимирующей функции подбирается из ограниченного возможного набора. Но эта гибкость имеет и оборотную сторону - количество степеней свободы создаваемой прогностической модели часто превышает число использовавшихся для обучения примеров. Это означает, что нейросеть может "научиться" даже на массиве сгенерированных случайных чисел. И

действительно, как показывает применение нейросети для решения тестовой задачи по анализу рынка акций, приведенной в [8], она прекрасно объясняет все колебания рынка в прошлом, но не дает обоснованного прогноза на будущее. Улучшение прогностической точности обученной сети может быть достигнуто при использовании для обучения нейронной сети только некоторой части обучающего множества, тогда как оставшаяся часть примеров используется для проверки адекватности созданной модели на неизвестных данных; одновременно следует стараться обучить сеть как можно менее сложной конфигурации для уменьшения числа степеней свободы.

Имеется и ряд других недостатков, ограничивающих использование нейронных сетей в качестве инструмента ИАД.

1. Обученные нейронные сети являются нетрактуемыми моделями - "черными ящиками", поэтому логическая интерпретация описанных ими закономерностей практически невозможна (за исключением простейших случаев).
2. Будучи методом группы вывода уравнений, нейронные сети могут обрабатывать только численные переменные. Следовательно, переменные других типов, как входные, так и выходные, должны быть закодированы числами. При этом недостаточно заменить переменную, принимающую значения из некоторой области определения, одной численной переменной, так как в этом случае могут быть получены некорректные результаты; необходимо ввести по новой переменной для каждого ее возможного значения, чтобы при равенстве исходной переменной определенному значению новая переменная для этого значения устанавливалась, например, в 1, а прочие новые переменные - в 0 [52]. Таким образом, при большом количестве нечисловых переменных с большим количеством возможных значений использование нейронных сетей становится совершенно невозможным.

Главной проблемой обучения нейронных сетей является синтез структуры сети, способной обучиться на заданном обучающем множестве. Нет никакой гарантии, что процесс обучения сети определенной структуры не остановится, не достигнув допустимого порога ошибки, или не попадет в локальный минимум. Хотя многослойные сети широко применяются для классификации и аппроксимации функций, их структурные параметры до сих пор должны определяться путем проб и ошибок. Согласно выводам [6], существующие теоретические результаты дают лишь слабые ориентиры для выбора этих параметров в практических приложениях.

Таким образом, нейронные сети - довольно мощный и гибкий инструмент ИАД - должны применяться с известной осторожностью и подходят не для всех проблем, требующих интеллектуального анализа корпоративных данных.

4.3.3. Выводы

Как видно из сделанного обзора, ни один из рассмотренных методов не способен покрыть все задачи, обеспечивающие поддержку принятия управленческих решений на основе интеллектуального анализа содержимого хранилищ данных. Но большинство существующих на рынке систем интеллектуального анализа реализуют один-три метода (например, Pilot Discovery Server фирмы Pilot Software Inc. и Information Harvester фирмы Information Harvester Corp. - только деревья решений, Idis фирмы Information Discovery Inc. - деревья решений и индукцию правил, Darwin фирмы Thinking Machines - нейронные сети, деревья решений и визуализацию данных [2], MineSet фирмы Silicon Graphics - деревья решений, индукцию ассоциативных правил и визуализацию данных [1]), поэтому в реальных приложениях для того, чтобы не потерять большое количество значимых закономерностей, приходится, как правило, пользоваться несколькими разнородными инструментами. Кроме того, многие инструменты не позволяют напрямую работать с хранилищами данных, требуя

предварительной подготовки исходных данных для анализа в виде плоских файлов фиксированной структуры, что также затрудняет их практическое использование.

5. Взаимодополняемость OLAP и ИАД

Оперативная аналитическая обработка и интеллектуалный анализ данных - две составные части процесса поддержки принятия решений. Но сегодня большинство систем OLAP заостряет внимание только на обеспечении доступа к многомерным данным, а большинство средств ИАД, работающих в сфере закономерностей, имеют дело с одномерными перспективами данных. Эти два вида анализа должны быть тесно объединены, то есть системы OLAP должны фокусироваться не только на доступе, но и на поиске закономерностей.

K. Parsaye [53] вводит составной термин "OLAP Data Mining" (многомерный интеллектуальный анализ) для обозначения такого объединения (рис. 9). Средство многомерного интеллектуального анализа данных должно находить закономерности как в детализированных, так и в агрегированных с различной степенью обобщения данных. Анализ многомерных данных должен строиться над специального вида гиперкубом, ячейки которого содержат не произвольные численные значения (количество событий, объем продаж, сумма собранных налогов), а числа, определяющие вероятность соответствующего сочетания значений атрибутов. Проекции такого гиперкуба (исключающие из рассмотрения отдельные измерения) также должны исследоваться на предмет поиска закономерностей. К сожалению, очень немногие производители предоставляют сегодня средства интеллектуального анализа многомерных данных в рамках систем OLAP - по-видимому, только SAS Institute добился в этом некоторого успеха. Проблема также заключается в том, что многие методы ИАД (байесовские сети, метод k-ближайшего соседа) неприменимы для задач многомерного интеллектуального анализа, так как основаны на определении сходства детализированных примеров и не способны работать с агрегированными данными [53].

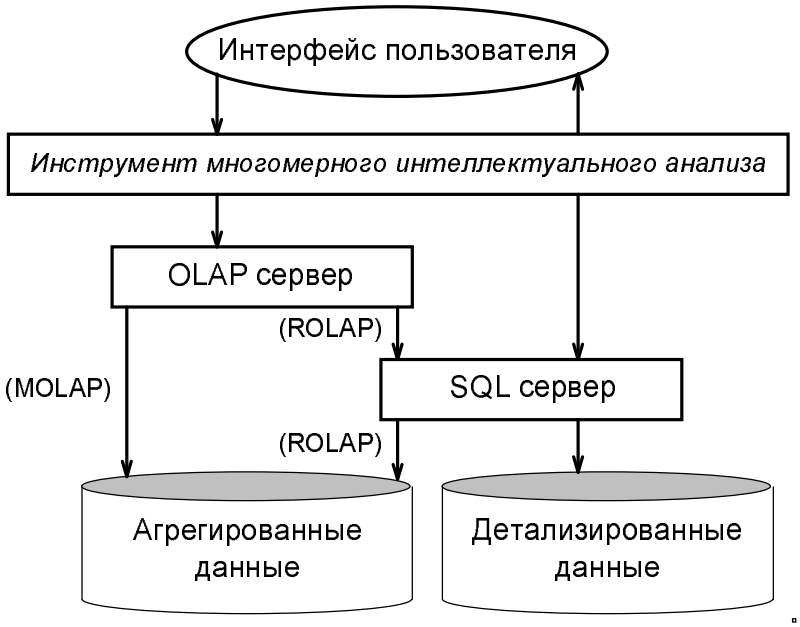

Рис. 9. Архитектура системы многомерного интеллектуального анализа данных.

Идеальной целью построения корпоративной информационно-аналитической системы является создание СППР замкнутого цикла. Как заметил N. Raden, "многие компании

создали ... прекрасные хранилища данных, идеально разложив по полочкам горы неиспользуемой информации, которая сама по себе не обеспечивает ни быстрой, ни достаточно грамотной реакции на рыночные события" [12, С. 39]. В особенно динамичных сферах (например, в розничной торговле), где ситуация меняется ежедневно, своевременное принятие грамотных решений не обеспечивается даже при использовании обычных средств OLAP и ИАД. Они должны быть объединены друг с другом и иметь обратную связь к исходным системам обработки данных, с тем чтобы результаты работы СППР немедленно передавались в виде управляющих воздействий в оперативные системы. Так, крупнейшая американская компания в сфере розничной торговли Wal-Mart занимается разработкой СППР замкнутого цикла [12]. Постоянно пополняемое хранилище данных POS-информации (получаемой от кассовых терминалов о регистрации торговых операций) поставляет исходные данные для работающей на мэйнфрейме СППР, которая автоматически инициирует оперативные действия по восстановлению уровня товарных запасов, а также выдает экспертам детализированные рекомендации по реагированию на состояние рынка на основе оперативного определения рыночных тенденций.

Список литературы

1. Аджиев В. MineSet - визуальный инструмент аналитика // Открытые системы. - 1997. - № 3. - С. 72-77.
2. Альперович М. Технологии хранения и обработки корпоративных данных (Data Warehousing, OLAP, Data Mining). - http://www.sft.ru/reviews/DevCon97/DC2/DC2T12.htm.
3. Арагон Л. Долой грязь! // PC Week/RE. - 1998. - № 6. - С. 53-54.
4. Вайну Я. Я.-Ф. Корреляция рядов динамики. - М.: Статистика, 1977. - 119 с.
5. Гупал А. М., Пономарев А. А., Цветков А. М. Об одном методе индуктивного вывода с подрезанием деревьев решений // Кибернетика и системный анализ. - 1993. - № 5. - С. 174-178.
6. Джейн А. К., Мао Ж., Моиуддин К. М. Введение в искусственные нейронные сети // Открытые системы. - 1997. - № 4. - С. 16-24.
7. Елисеева И. И., Юзбашев М. М. Общая теория статистики / Под ред. чл.-корр. РАН И. И. Елисеевой. - М.: Финансы и статистика, 1996. - 368 с.: ил.
8. Киселев М., Соломатин Е. Средства добычи знаний в бизнесе и финансах // Открытые системы. - 1997. - № 4. - С. 41-44.
9. Крастинь О. П. Разработка и интерпретация моделей корреляционных связей в экономике. - Рига: Зинатне, 1983.
10. Кречетов Н., Иванов П. Продукты для интеллектуального анализа данных // ComputerWeek-Москва. - 1997. - № 14-15. - С. 32-39.
11. Кузнецов С. Д., Артемьев В. Обзор возможностей применения ведущих СУБД для построения хранилищ данных (Data Warehouse) // 3-я ежегодная конференция Корпоративные базы данных '98: Доклады и тезисы / Центр информационных технологий. - М., 1998. - С. 153-161.
12. Моуд Дж. Товары впрок // PC Week/RE. - 1997. - № 33. -С. 39-40.
13. Пржиялковский В. В. Сложный анализ данных большого объема: новые перспективы компьютеризации // СУБД. - 1996. - № 4. - С. 71-83.
14. Раден Н. Данные, данные и только данные // ComputerWeek-Москва. - 1996. - № 8. - С. 28.
15. Сахаров А. А. Концепция построения и реализации информационных систем, ориентированных на анализ данных // СУБД. - 1996. - № 4. - С. 55-70.

16. Сахаров А. А. Принципы проектирования и использования многомерных баз данных (на примере Oracle Express Server) // СУБД. - 1996. - № 3. - С. 44-59.
17. Туо Дж. Инструменты для анализа информации на настольных ПК // ComputerWeek-Москва. - 1996. - № 38. - С 34-35, 46.
18. Туо Дж. Каждому пользователю - свое представление данных // ComputerWeek-Москва. - 1996. - № 38. - С. 1, 32-33.
19. Уоссермен. Ф. Нейрокомпьютерная техника: Теория и практика. - М.: Мир, 1992. - 240 с.
20. Цветков А. М. Разработка алгоритмов индуктивного вывода с использованием деревьев решений // Кибернетика и системный анализ. - 1993. - № 1. - С. 174-178.
21. Эделстейн Г. Интеллектуальные средства анализа, интерпретации и представления данных в информационных хранилищах // ComputerWeek-Москва. - 1996. - № 16. - С. 32-33.
22. Aha D. W. An Implementation and Experiment with the Nested Generalized Exemplars Algorithm (Technical Report AIC-95-003). - Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1995.
23. Aha D. W., Salzberg S. L. Learning to Catch: Applying Nearest Neighbor Algorithms to Dynamic Control Tasks / In P. Cheeseman & R. W. Oldford (Eds.) Selecting Models from Data: Artificial Intelligence and Statistics. - New York, NY: Springer-Verlag, 1993.
24. Alalouf C. Hybrid OLAP. - St. Laurent, Canada: Speedware Corporation Inc., 1997.
25. An Introduction to Multidimensional Database Technology. - Kenan Systems Corporation, 1995.
26. Brand E., Gerritsen R. Data Mining and Knowledge Discovery // DBMS. - 1998. - № 7.
27. Brand E., Gerritsen R. Decision Trees // DBMS. - 1998. - № 7.
28. Brand E., Gerritsen R. Naive-Bayes and Nearest Neighbor // DBMS. - 1998. - № 7.
29. Cendrowska J. PRISM: An algorithm for inducing modular rules // International Journal of Man-Machine Studies. - 1987. - 27. - P. 349-370.
30. CFO Vision. - SAS Institute Inc., 1997.
31. Codd E. F., Codd S. B., Salley C. T. Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. - E. F. Codd & Associates, 1993.
32. Demarest M. Building the Data Mart // DBMS. - 1994. - № 7. - P. 44-50.
33. Fausett L. V. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. - Englewood Cliffs, New Jersey: Prentice Hall, 1994. - 461 p.
34. Frawley W. L., Piatetsky-Shapiro G., Matheus C. J. Knowledge discovery in databases: An overview // AI Magazine. - 1992. - № 13(3). - P. 57-70.
35. Friedman N., Geiger D., Goldszmidt M., etc. Bayesian Network Classifiers // Machine Learning. - 1997. - 29. - P. 131-165.
36. Fuernkranz J. A Brief Introduction to Knowledge Discovery in Databases // OEGAI Journal. - 1995. - № 14(4). - P. 14-17.
37. Fuernkranz J. Separate-and-Conquer Rule Learning. - Vienna: Austrian Research Institute for Artificial Intelligence, Technical Report OEFAI-TR-96-25, 1996.
38. Gray J., Chaudhuri S., Bosworth A., etc. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals // Data Mining and Knowledge Discovery. - 1997. - № 1. - P. 29-53.
39. Harinarayan V., Rajaraman A., Ullman J. D. Implementing Data Cubes Efficiently // SIGMOD Conference. - Montreal, CA. -1996.
40. Heckerman D. Bayesian Networks for Data Mining // Data Mining and Knowledge Discovery. - 1997. - № 1. - P. 79-119.
41. Heckerman D., Geiger D., Chickering D. Learning Bayesian networks: The combination of knowledge and statistical data // Machine Learning. - 1995. - 20. - P. 197-243.
42. Inmon W. H. Building The Data Warehouse (Second Edition). - NY, NY: John Wiley, 1993.

43. Kosko B. Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. - Englewood Cliffs, New Jersey: Prentice Hall, 1992. - 452 p.
44. Kramer S. Structural Regression Trees. - Vienna: Austrian Research Institute for Artificial Intelligence, Technical Report OEFAI-TR-95-35, 1995.
45. Kulkarni J., King R. Business Intelligence Systems and Data Mining. - SAS Institute Inc., 1996.
46. Michalski R. S. A theory and methodology of inductive learning // Artificial Intelligence. - 1983. - 20(2). - P. 111-162.
47. Michalski R. S. On the quasi-minimal solution of the covering problem // In Proceedings of the 5th International Symposium on Information Processing (FCIP-69): Vol. A3 (Switching Circuits). - Bled, Yugoslavia, 1969. - P. 125-128.
48. Mumick I. S., Quass D., Mumick B. S. Maintenance of Data Cubes and Summary Tables in a Warehouse. - Stanford University, Database Group, 1996 (http://www-db.stanford.edu/pub/papers/cube-maint.ps).
49. Newquist H. P. Data Mining: The AI Metamorphosis // Database Programming and Design. - 1996. - № 9.
50. OLAP for the Masses. - Business Objects S. A., 1996 (http://www.businessobjects.com/product/olap/olap.htm).
51. Pagallo G., Haussler D. Boolean feature discovery in empirical learning // Machine Learning. - 1990. - 5. - P. 71-99.
52. Parsaye K. A Characterization of Data Mining Technologies and Processes // The Journal of Data Warehousing. -1998.-№ 1.
53. Parsaye K. OLAP and Data Mining: Bridging the Gap // Database Programming and Design. - 1997. - № 2.
54. Parsaye K. Rules are Much More than Decision Trees // The Journal of Data Warehousing. - 1997. - № 1.
55. Parsaye K. Surveying Decision Support: New Realms of Analysis // Database Programming and Design. - 1996. - № 4.
56. Pyne A. The SAS System and Web Integration. - SAS Institute Inc., 1996.
57. Quinlan J. R. C4.5: Programs for Machine Learning. - San Mateo, CA: Morgan Kaufmann, 1993.
58. Quinlan J. R. Generating production rules from decision trees // In Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87). - Morgan Kaufmann, 1987. - P. 304-307.
59. Quinlan J. R. Induction of decision trees // Machine Learning. - 1986. - 1. - P. 81-106.
60. Quinlan J. R. Simplifying decision trees // International Journal of Man-Machine Studies. - 1987. - 27. - P. 221-234.
61. Raden N. Star Schema. - Santa Barbara, CA: Archer Decision Sciences, Inc., 1995-1996 (http://members.aol.com/nraden/str101.htm).
62. Tukey J. Exploratory Data Analysis. - NY: McMillan, 1973.
63. Wettschereck D., Aha D. W., Mohri T. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms // Artificial Intelligence Review. - 11. - P. 273-314.
64. Решение проблемы комплексного оперативного анализа информации хранилищ данных / Коровкин С. Д., Левенец И. А., Ратманова И. Д., Старых В. А., Щавелев Л. В. // СУБД. - 1997. - № 5-6. - С. 47-51.