Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Пермская государственная сельскохозяйственная академия
имени академика. Д.Н. Прянишникова»

А.Н.Козлов

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Учебник

Рекомендовано Учебно-методическим объединением по образованию
в области прикладной информатики
в качестве учебника для студентов,
обучающихся по направлению и специальности
«Прикладная информатика»

Пермь
ФГБОУ ВПО Пермская ГСХА
2013

УДК 004.491
ББК 32.973.018.2
К 592

Рецензенты:

кафедра прикладной математики и информатики Пермского государственного научно-исследовательского университета(заведующий кафедрой – доктор.физ.-мат. наук, профессор **Русаков С.В.**);

Тарутин А.В., канд. техн. наук, доцент, начальник кафедры АУВ Пермского ВИ МВД РФ.

К592 Козлов, А.Н. Интеллектуальные информационные системы: учебник /А.Н. Козлов; Мин-во с-х. РФ, ФГБОУ ВПО Пермская ГСХА. – Пермь: Изд-во ФГБОУ ВПО Пермская ГСХА, 2013.– 278 с.

ISBN978-5-94279-176-6

Учебникпосвященвопросам изучения и применения интеллектуальных информационных систем (ИИС). Рассматриваются, классификация, архитектура ИИС, вопросы представления знаний в ИИС, структура и этапы проектирования экспертных систем, а также методы интеллектуального анализа данных. Учебник содержит две части. Часть первая содержит лекционный материал. Во второй части представлен практикум в виде руководств по лабораторным (практическим) занятиям.

Предназначено для студентов, обучающихся по специальности 080801 «Прикладная информатика (в экономике)», студентов обучающихся по направлениям 080800 и 230700 «Прикладная информатика», а также для студентов и аспирантов других экономических и технических специальностей.

УДК 004.491
ББК 32.973.018.2

Печатается по решению методической комиссии факультета прикладной информатики.

ISBN978-5-94279-176-6

Оглавление

ВВЕДЕНИЕ

В современном мире прогресс производительности программиста практически достигается только в тех случаях, когда часть интеллектуальной нагрузки берут на себя компьютеры. Одним из способов достигнуть максимального прогресса в этой области, является "искусственный интеллект", когда компьютер берет на себя не только однотипные, многократно повторяющиеся операции, но и сам сможет обучаться. Кроме того, создание полноценного "искусственного интеллекта" открывает перед человечеством новые горизонты развития.

Одним из направлений в области искусственного интеллекта являются интеллектуальные информационные системы. Интеллектуальные информационные системы это естественный результат развития обычных информационных систем. Они сосредоточили в себе наиболее наукоемкие технологии с высоким уровнем автоматизации не только процессов подготовки информации для принятия решений, но и самих процессов выработки вариантов решений, опирающихся на полученные информационной системой данные.

Цель данного учебного пособия — дать обучаемым комплекс ориентирующих знаний по основным понятиям интеллектуальных информационных систем и возможностям их использования в различных предметных областях Владение интеллектуальными информационными технологиями и системами обеспечивает высокий уровеньконрурентоспособности на рынке труда как специалиста в области современных информационных технологий для реализациибизнес-процессов.

Автор в полной мере понимает, что учебное пособие не является «исчерпывающим» трудом в области изучения интеллектуальных информационных систем. Пособие составлено на основе анализа, систематизации, частичной переработки материалов источников [1-12], интернет-источников и прочих материалов. Особо хочется отметить источник [4] , на основе которого были составлены главы посвященные интеллектуальному анализу данных. Данное пособие является учебно-методическим трудом, отражающим видение автора на структуру и наполнение дисциплины «Интеллектуальные информационные системы» в соответствии с ФГОС ВПО по специальности (направлению) «Прикладная информатика в экономике».

Учебное пособие содержит две части. Часть первая – это лекционный материал. Главы являются темами (разделами) в дисциплине, разделы в главе – темы лекций, в тексте которых выделены учебные вопросы. Во второй части представлен практикум по дисциплине в виде руководств по лабораторным (практическим) занятиям. Даны краткие рекомендации преподавателю по проведению занятий. Время выполнения работ указано ориентировочно, на основе опыта проведенных занятий. Материал учебного пособия в течение 5 лет успешно апробирован в учебном процессе по специальности (направлению) «Прикладная информатика в экономике».

Часть 1. ОСНОВЫ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Глава 1. ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1.1. Понятие и классификация ИИС

Основные термины и понятия.

Термин *интеллект* (intelligence) происходит от латинского intellectus — что означает ум, рассудок, разум, мыслительные способности человека.

Интеллект - *это способность мозга решать интеллектуальные задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.*

В этом определении под термином *"знания"* подразумевается не только та информация, которая поступает в мозг через органы чувств. Такого типа знания чрезвычайно важны, но недостаточны для интеллектуальной деятельности. Дело в том, что объекты окружающей нас среды обладают свойством не только воздействовать на органы чувств, но и находиться друг с другом в определенных отношениях. Ясно, что для того, чтобы осуществлять в окружающей среде интеллектуальную деятельность (или хотя бы просто существовать), необходимо иметь в системе знаний модель этого мира. В этой информационной модели окружающей среды реальные объекты, их свойства и отношения между ними не только отображаются и запоминаются, но и, как это отмечено в данном определении интеллекта, могут мысленно "целенаправленно преобразовываться". При этом существенно то, что формирование модели внешней среды

происходит "в процессе обучения на опыте и адаптации к разнообразным обстоятельствам".

Знания – *проверенный практикой и удостоверенный логикой результат познания действительности, отраженный в сознании человека в виде представлений, понятий, рассуждений и теорий. Знания формируются в результате целенаправленного педагогического процесса, самообразования и жизненного опыта.*

Соответственно **искусственный интеллект** (artificialintelligence - AI) обычно толкуется как*свойство автоматических систем брать на себя отдельные функции интеллекта человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий.*

Для того, чтобы пояснить, чем отличается интеллектуальная задача от просто задачи, необходимо ввести термин *"алгоритм"* — один из краеугольных терминов кибернетики.

Алгоритм - *точное предписание о выполнении в определенном порядке системы операций для решения любой задачи из некоторого данного класса (множества) задач.*

Термин *"алгоритм"* происходит от имени узбекского математика Аль-Хорезми, который еще в IX веке предложил простейшие арифметические алгоритмы. В математике и кибернетике класс задач определенного типа считается решенным, когда для ее решения установлен алгоритм. Нахождение алгоритмов является естественной целью человека при решении им разнообразных классов задач.

Интеллектуальные задачи – *это задачи, отыскание алгоритма решения которых связано с тонкими и сложными*

рассуждениями, логическими обобщениями и выводами, требующие большой изобретательности и высокойквалификации.

Принято считать, что подобного рода деятельность требует участия интеллекта человека. Что же касается задач, алгоритмы решения которых уже установлены, то, как отмечает известный специалист в области ИИ М. Минский, "излишне приписывать им такое мистическое свойства, как "интеллектуальность". В самом деле, после того, как такой алгоритм уже найден, процесс решения соответствующих задач становится таким, что его могут в точности выполнить человек, вычислительная машина (должным образом запрограммированная) или робот, не имеющие ни малейшего представления о сущности самой задачи. Требуется только, чтобы лицо, решающее задачу, было способно выполнять те элементарные операции, из которых складывается процесс, и, кроме того, чтобы оно педантично и аккуратно руководствовалось предложенным алгоритмом. Такое лицо, действуя, как говорят в таких случаях, чисто машинально, может успешно решать любую задачу рассматриваемого типа. Поэтому представляется совершенно естественным исключить из класса интеллектуальных такие задачи, для которых существуют стандартные методы (алгоритмы) решения. Примерами таких задач могут служить чисто вычислительные задачи: решение системы линейных алгебраических уравнений, численное интегрирование дифференциальных уравнений и т. д.

Интеллектуальная система - *система или устройство с программным обеспечением, имеющие возможность с помощью встроенного процессора настраивать свои параметры в зависимости от состояния внешней среды.*

Анализ различных источников показал, что на данный момент нет четкого и однозначного определения интеллектуальной информационной системы. Каждый автор вкладывает свой смысл в это понятие с учетом своего понимания и предметной области. Можно использовать следующее определение.

Интеллектуальная информационная система *– это взаимосвязанная совокупность средств, методов и персонала, имеющая возможность хранения, обработки и выдачи информации, а также самостоятельной настройки своих параметров в зависимости от состояния внешней среды (исходных данных) и специфики решаемой задачи.*

По материалам рекрутинговых агентств, представленным в Интернете, существует устойчивый высокий спрос на специалистов, владеющих современными технологиями проектирования и разработки ИИС. Поскольку технические и программные средства изменяются достаточно быстро (их полное обновление происходит в течение 2-3 лет), а принципы работы интеллектуальных систем изменяются относительно медленно (на протяжении 15-20 лет).

Назначение, свойства и особенности ИИС.

Интеллектуальные информационные системы являются естественный результатом развития обычных информационных систем, сосредоточили в себе наиболее наукоемкие технологии с высоким уровнем автоматизации не только процессов подготовки информации для принятия решений, но и самих процессов выработки вариантов решений, опирающихся на полученные информационной системой данные.

ИИС особенно эффективны в применении к слабо структурированным задачам, в которых пока отсутствует строгая формализация, где при принятии решений учитываются наряду с экономическими показателями слабо формализуемые факторы — экономические, политические, социальные.

Предназначение ИИС (в области экономики).

Диагностика состояния предприятия.

Помощь в антикризисном управлении.

Выбор оптимальных решений по стратегии развития предприятия и его инвестиционной деятельности.

Экономический анализ деятельности предприятия.

Стратегическое планирования.

Инвестиционный анализа, оценка рисков.

Формирование портфеля ценных бумаг и т.п.

В основе этих видов деятельности лежит проблема выбора решений, в нашем учебном пособии значительное внимание уделено выбору вариантов решений на основе выявления функциональных или логических зависимостей между наблюдаемыми данными и решениями, а также представлению их в форме байесовских правил, ориентированных графов или правил на языке исчисления предикатов. Этим объясняется то, что диапазон применения ИИС необычайно широк: от управления непрерывными технологическим процессами в реальном времени до оценки последствий от нарушения условий поставки товаров.

Благодаря наличию средств естественно-языкового интерфейса появляется возможность непосредственного

применения ИИС бизнес-пользователем, не владеющим языками программирования, в качестве средств поддержки процессов анализа, оценки и принятия экономических решений.

Для решения вышеперечисленных задач ИИС должна обладать определенными свойствами:

- решать задачи, описанные только в терминах мягких моделей, когда зависимости между основными показателями являются не вполне определенными или даже неизвестными в пределах некоторого класса;
- способность к работе с неопределенными или динамичными данными, изменяющимися в процессе обработки, позволяет использовать ИИС в условиях, когда методы обработки данных могут изменяться и уточняться по мере поступления новых данных;
- способность к развитию системы и извлечению знаний из накопленногоопыта конкретных ситуаций, что увеличивает мобильность и гибкость системы, позволяет ей быстро осваивать новые области применения.
- возможность использования информации, которая явно не хранится, а выводится из имеющихся в базе данных, позволяет уменьшить объемы хранимой фактуальной информации при сохранении богатства доступной пользователю информации.

Для ИИС характерны следующие признаки:

- развитые коммуникативные способности: возможность обработки произвольных запросов в диалоге на языке максимально приближенном к естественному (система естественно-языкового интерфейса — СЕЯИ);

- направленность на решение слабоструктурированных, плохо формализуемых задач (реализация мягких моделей);
- способность работать с неопределенными и динамичными данными;
- способность к развитию системы и извлечению знаний из накопленного опыта конкретных ситуаций;
- возможность получения и использования информации, которая явно не хранится, а выводится из имеющихся в базе данных;
- система имеет не только модель предметной области, но и модель самой себя, что позволяет ей определять границы своей компетентности;
- способность к аддуктивным выводам, т.е. к выводам по аналогии;
- способность объяснять свои действия, неудачи пользователя, предупреждать пользователя о некоторых ситуациях, приводящих к нарушению целостности данных.

Отличительные особенности ИИС по сравнению с обычными ИС состоят в следующем:

- интерфейс с пользователем на естественном языке с использованием бизнес-понятий, характерных для предметной области пользователя;
- представление модели экономического объекта и его окружения в виде базы знаний и средств дедуктивных и правдоподобных выводов в сочетании с возможностью работы с неполной или неточной информацией;
- решения ИИС обладают *"прозрачностью",* т.е. могут быть объяснены пользователю на качественном уровне;

- экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом;
- способность автоматического обнаружения закономерностей бизнеса в ранее накопленных фактах и включения их в базу знаний(т.н. машинное обучение).
- ИИС дает пользователю «готовое» решение, которое по качеству и эффективности не уступает решению эксперта-человека;
- применение специфического компонента – базы знаний.

Классификация и примеры ИИС.

Как уже отмечалось, интеллектуальные системы имеют следующие характерные признаки:

- развитые коммуникативные способности;
- умение решать сложные, плохо формализуемые задачи;
- способность к развитию и самообучению.

Условно каждому из этих признаков соответствует свой класс ИИС, поэтому они могут служить основой классификации ИИС.

По коммуникативным способностям.

- Интеллектуальные базы данных;
- Системы естественно-языкового интерфейса (СЕЯИ);
- Гипертекстовые системы;
- Контекстные системы;
- Системы когнитивной графики.

Интеллектуальные БД – отличаются от обычных возможностью выборки по запросу информации, которая может явно не храниться, а выводиться из имеющейся БД

(например, вывести список товаров, цена которых выше отраслевой).

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на машинный уровень представления знаний. При этом осуществляется распознавание и проверка написанных слов по словарям и синтаксическим правилам. Данный интерфейс облегчает обращение к интеллектуальным БД, а также голосовой ввод команд в системах управления.

Гипертекстовые системы предназначены для поиска текстовой информации по ключевым словам в базах.

Системы контекстной помощи – частный случай гипертекстовых и естественно-языковых систем.

Системы когнитивной графики позволяют осуществлять взаимодействие пользователя ИИС с помощью графических образов.

По типу решаемых задач.

- Экспертные системы;
- Классифицирующие системы;
- Доопределяющие системы;
- Трансформирующие системы;
- Многоагентные системы.

По способности к самообучению.

- Индуктивные системы;
- Нейронные сети;
- Системы, основанные на прецедентах;
- Информационные хранилища.

Архитектура ИИС.

Типовая архитектура ИИС представлена на рис.1.1, в ее состав входят.

База знаний. Служит для представления эвристической и фактологической информации, часто в форме фактов, утверждений и правил вывода. В базе знаний ИИС интегрируются знания, поступающие от экспертов в конкретной предметной области, а также фундаментальные (энциклопедическими знаниями), составляющие суть общеизвестных научных теорий и моделей.

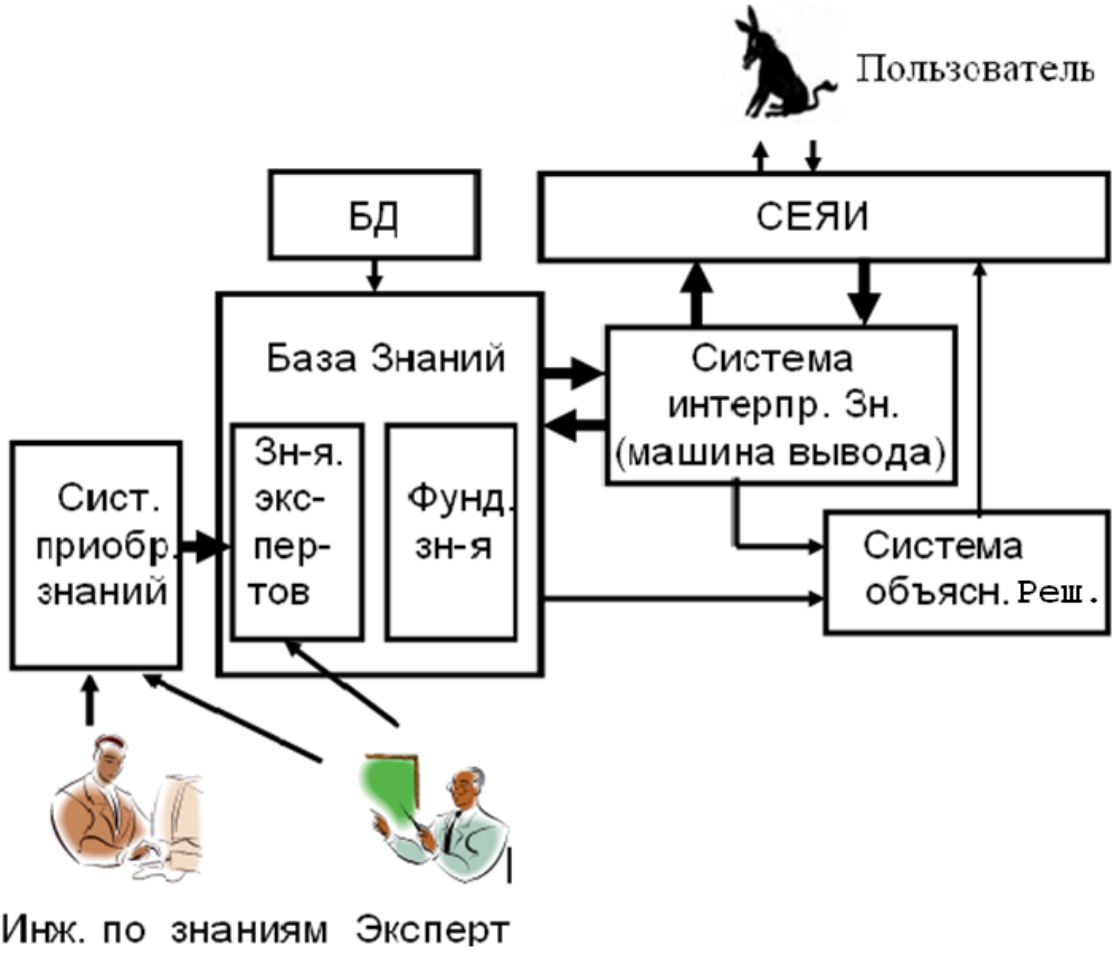


Рис.1.1. Архитектура интеллектуальной информационной системы

База данных хранит то, что называется форматированными данными, то есть конкретные факты и константы, касающиеся предметной области. Для хранения данных в ИИС используются либо реляционные, либо для интегрированного хранения данных и знаний — объектно-ориентированные базы данных.

Система интерпретации знаний (Машина вывода). Это механизм, играющий роль интерпретатора, применяющего знания подходящим образом, чтобы получить результат.

Система приобретения знаний. ***Это механизм формализации знаний эксперта, пополнения базы знаний и обучения (самообучения) ИИС.Это реализуется с помощью инженера по знаниям и эксперта, а также машинного обучения.***

Система естественно-языкового интерфейса (СЕЯИ). Механизм, преобразующий запросы пользователя и выдающий ему решение задачи (результат) на естественном языке. СЕЯИ реализует дружественный к пользователю интерфейс.

Система объяснения решения. Механизм протоколирующий работу системы интерпретации знаний и выдающий его пользователю в виде цепочки логических выводов. Это позволяет «объяснить» пользователю найденное решение.

1.2. Интеллектуальный интерфейс и методы рассуждений в ИИС

Система естественно-языкового интерфейса.

В настоящее время проблема общения с компьютером на естественном языке весьма далека от своего решения.

Современные средства естественно-языкового интерфейса далеки от универсальных возможностей человеческой коммуникации.

Естественно-языковый интерфейс (СЕЯИ) – ***это совокупность программных и аппаратных средств, обеспечивающих общение ИИС с пользователем на ограниченном рамками предметной области естественном языке.***

В состав СЕЯИ входят словари, отражающие словарный состав и лексику языка, а также лингвистический процессор, осуществляющий анализ текстов (морфологический, синтаксический, семантический и прагматический) и синтез ответов пользователю. В идеале СЕЯИ должно удовлетворять двум основным требованиям:

- обеспечивать вход и выход системы на естественном языке;
- обеспечивать обработку входных данных и генерацию выходных, основываясь на знании относительно синтаксических, семантических и прагматических аспектов естественного языка.

Проблемы понимания естественного языка, будь то текст или речь, во многом зависят от знания предметной области. Понимание языка требует знаний о целях говорящего и о контексте. Необходимо также учитывать недосказанность или иносказательность. Например, даже в таком простом предложении «Ваня встретил Машу на поляне с цветами» нам не понятно, кто же был с цветами: Ваня, Маша или поляна? Еще один пример «Врач бегло говорила по-английски». Разбирая это предложение, необходимо в результате разбора зафиксировать, что врач была женщина. Крылатая фраза знаменитого русского лингвиста, академика

Л.В.Щербы «Глокаякуздраштекобудланулабокра и курдячитбокренка» говорит о том, что такая «непонятная» фраза построена по всем правилам русского языка, не вызывает проблем с грамматическим разбором такого предложения, но вызывает проблемы с пониманием. Обозначим лишь некоторые *проблемы понимания естественного языка*.

Проблема «смысл-текст». Об этом только что говорилось и приведем еще один пример по этой проблеме. В предложении «Какой завод заказал оборудование для конвертерного цеха в Бельгии?» неясен смысл: был ли сделан заказ в Бельгии или цех находится в Бельгии.

Проблема «планирования» возникает при необходимости вести диалог, например, на тему «Куда Вы хотите лететь?». В этом случае нужно глубокое знание предметной области (номера рейсов, время прилета-отлета, цены и т.д.).

Проблема «равнозначности». Будут ли равнозначны два предложения «У дома стоит слон» и «У дома стоит существо с хоботом и бивнями»? На первый взгляд нет сомнений в равнозначности этих предложений. А если в базе знаний существо с хоботом и бивнями определено двумя значениями: слон и мамонт, то такие сомнения, наверное, появятся.

Проблемы «моделей» участников общения. У участников общения должны быть сопоставимые модели представления знаний, необходимая глубина понимания, возможность логического вывода, возможность действия.

Проблема «эллиптических конструкций», то есть опущенных элементов диалога. Например, в пословице «Береги платье снову, а честь - смолоду» вторая часть

предложения будет синтаксическим *эллипсисом* (опущен глагол береги).

Проблема «временных противоречий». Например, в предложении «Я хотел завтра пойти в кино» глагол «хотел» в прошедшей форме сочетается с обстоятельством будущего времени «завтра», что противоречит общепринятой логике.

Задача создания естественно-языкового интерфейса может быть рассмотрена как две взаимодействующие подзадачи.

Первая подзадача - это взаимодействие интерфейса с пользователем. Здесь непосредственно должен быть реализован алгоритм понимания смысла того, что сообщает пользователь. То есть должно быть реализовано множество выражений, которые пользователь может сказать, а система таким образом должна понять.

Вторая подзадача - интерфейс должен конвертировать смысл сказанной фразы в какое-либо внутреннее представление. Это может быть уже существующий язык, например, Пролог, VBScript. Это может быть какое-либо свое внутреннее представление. Представление во внутренних кодах нужно для понимания смысла запроса пользователя и трансляции этого смысла в запросы SQL. После формирования запроса во внутреннем представлении необходимо представить информацию на языке, понятном для базы данных. Таким образом, происходит трансляция в SQL. Возврат информации осуществляется в обратном порядке. База данных на запрос отвечает множеством записей из базы данных. Результат транслируется во внутреннее представление, затем во фразы естественного языка и выдается пользователю как ответ. Затем пользователь снова обращается и так далее.

Направления реализации естественно-языкового диалогового интерфейса.

Сейчас многие IT-компании занимаются созданием, так называемых, голосовых пользовательских интерфейсов (sounduserinterface) или SUI, в основе которых лежит решение проблемы понимания смысла пользовательской фразы. Термин “понимание смысла“ интерпретируется как процесс трансляции фразы пользователя в некоторое адекватное действие, совершаемое системой.

Если говорить о системах понимания естественной речи или NLU-ситемах, то можно выделить два типа таких систем:

1. Командные системы (commandcontrol). Этот тип систем интерпретирует единичную фразу пользователя в набор конечных команд, которые выполняются компьютером. С помощью таких систем строится голосовое управление компьютерными приложениями, например, MicrosoftWord. Эти системы полностью содержат в себе команды управления конкретным приложением и транслируют голосовые команды пользователя в команды управления приложением. Следующий пример показывает работу системы commandcontrol.

Пример:

ПОЛЬЗОВАТЕЛЬ: Открыть MicrosoftWord.

КОМПЬЮТЕР: Запускает приложение MicrosoftWord.

ПОЛЬЗОВАТЕЛЬ: Открыть File.

КОМПЬЮТЕР: Открывает диалоговое окно открытия файла.

ПОЛЬЗОВАТЕЛЬ: Изменить путь.

КОМПЬЮТЕР: Изображает дерево изменения директорий.

И т.д.

ПОЛЬЗОВАТЕЛЬ: Выйти.

КОМПЬЮТЕР: Закрывает приложение Microsoft Word.

Система commandcontrol понимает фразы естественного языка только лишь как набор команд. Поэтому в ее задачи не входит понимание смысла фразы пользователя. Система такого типа должна лишь сопоставить фразу пользователя с тем множеством фраз, которые заложены внутри нее, и при совпадении выполнить требуемое действие. Конечно, в этом случае можно говорить лишь об управлении с использованием ограниченного числа команд, представленных на естественном языке.

2. Диалоговые системы (dialogsystem). Эти системы более широко интерпретируют понятие естественного языкового общения и в своей основе содержат диалог пользователя с компьютером для выяснения конечной цели, которую хочет достичь пользователь и получения от него необходимых данных для решения поставленной задачи.

Пример:

ПОЛЬЗОВАТЕЛЬ: Я хотел бы поехать в Киев.

DS: Каким транспортом вы бы хотели воспользоваться?

ПОЛЬЗОВАТЕЛЬ: Конечно, лучше поездом.

DS: Каким транспортом вы бы хотели воспользоваться?

ПОЛЬЗОВАТЕЛЬ: Конечно, лучше поездом.

DS: Когда вы хотите быть в Киеве?

ПОЛЬЗОВАТЕЛЬ: 25 декабря.

DS: Перечисляет возможные рейсы и просит выбрать нужный

ПОЛЬЗОВАТЕЛЬ: Сообщает нужный рейс.

…

DS: После того как получена полная информация, бронирует билет.

Как видно из данного примера, что в течение диалога, DS может работать не с одним, но несколькими приложениями, как базой данных движения поездов, наличием мест, бронированием билетов и т.д.

При работе с системы. DS используется абсолютно иной подход. Здесь система нацелена на понимание смысла сказанного. Это возможно благодаря использованию некоторой логической структуры, на которую отображается речь пользователя.

Совершенно очевидно, что любой тип NLU-системы оперирует со смыслом пользовательских фраз. Существует несколько теорий определения смысла языковых фраз. Это и попытки отображения пользовательской фразы на семантическую модель представления мира, и попытки определения смысла путем разложения фразы на части речи и анализа структуры фразы (так называемая tag/chunk технология, которая определяет части речи фразы –tags, а затем смысловые куски -chunks, но не определяет смысл фразы в нашем понимании и хороша для разработки компьютерных переводчиков) и др.

Для представления естественного языка при разработке данного интерфейса использовалась теория грамматик, основными принципами которой являются.

Всякое множество фраз можно описать с помощью грамматики, которая определяет набор слов, из которых строятся фразы и способы построения возможных фраз. Можно, конечно, иметь полный список возможных фраз, но такая система будет громоздкой и не эффективной, а также довольно трудно модифицируемой. Например, в случае разработки диалога, связанного с продажей напитков, трудно себе представить как можно дополнять такой список фраз в случае поступления новых типов и наименований напитков. Это не только добавление новых слов, но и генерация всех возможных фраз, в которых может встречаться новое слово. С этой задачей успешно справляется грамматика, поскольку оперирует с понятием правил построения фраз, а не только со значениями слов. Далее, при описании грамматики, это станет понятней.

Можно выделять подмножество фраз, которые ожидаются от пользователя в текущий момент. Так называемые правила грамматики, которые содержат подмножества фраз и активизируются в нужный момент, в зависимости от текущего состояния диалога или активизации того или иного приложения.

Можно построить соответствие между каждой фразой пользователя и выходной грамматикой. Выходная грамматика – это фраза, содержащая набор команд (или одну команду), которая будет передана в соответствующее приложение для выполнения. Если в результате парсирования (анализа) входной фразы (фразы пользователя) была получена выходная фраза (тот самый набор команд), мы считаем, что смысл пользовательской фразы понят. Здесь следует отметить, что выходная грамматика может содержать не только команды управления конкретным приложением (что хорошо для систем commandcontrol), но и командные

строки, которые могут передаваться в некую диалоговую систему для отображения текущего состояния диалога и передачи данных в диалоговую систему. Другими словами, фраза пользователя транслируется в некоторую строку, которая в свою очередь передается в нужное приложение (с точки зрения грамматики, DS является тоже приложением) и является для него понятным.

Грамматика – математическая система, определяющая язык. Сердцевину грамматики определяет конечное множество правил образования, которые описывают процесс порождения цепочек языка.

Грамматика состоит из:

- слов (это так называемый алфавит);
- опций (еще одно свойство этой грамматики - опциональность), т.е. выражения, заключенные в квадратные скобки являются опцией и могут произноситься пользователем, а могут и нет. Например, из нашей грамматики: [Can I talkwith] <person>, может быть произнесено полностью фраза, а может быть названо только имя персоны;
- правил (rules). Это выражения, заключенные в угловые скобки <> и написанные маленькими буквами, но вне фигурных скобок {}. Правила содержат в себе те фразы, которые могут быть произнесены в текущий момент диалога, то есть имеется возможность делать активными те или иные правила, что бы таким образом ограничить множество входных фраз и таким образом улучшить понимание фразы;
- выходных грамматик. Т.е. выражений, которые являются результатом парсирования (трансляции, понимания) фразы. В нашем случае эти выходные

грамматики содержат команды управления диалогом. В принципе здесь может быть все, что угодно, главное, чтобы принимающая выходные грамматики программа это понимала;

- переменных выходных грамматик. Выражения, заключенные в <>, написанные большими буквами и находящимися внутри выходных грамматик (внутри {}). Они с точки зрения грамматики бесполезны, но нужны для парсирования.

Методы рассуждения в интеллектуальных информационных системах.

При реализации рассуждений в ИИС различают два подхода: логический метод рассуждения и эвристический метод рассуждения.

Логический метод рассуждений основан на применении логики. Логика это наука о правильных способах рассуждений. В классическом варианте состоит из учения о понятиях, учения о суждениях и учения об умозаключениях. Важно отметить, что логика.есть наука о мышлении в понятиях, а не о познании мира посредством мышления о понятиях.

Различают следующие виды логических рассуждений:

*1) **Дедукция** - рассуждение от сложного к простому, то есть получение частного правила на основе общего правила.*

Например, правила решений квадратных уравнений в общем случае выглядит так: Необходимо определить дискриминант и в зависимости от того, какое значение примет этот дискриминант (>0,=0,<0) делается вывод о том, что квадратное уравнение имеет 2 корня, один корень или не

имеет ни одного, а частные правила для решения квадратного уравнения можно считать по теореме Виета. Она применима только для уравнений, у которых коэффициент при x^2 равен единице.

2) ***Индукция*** - *рассуждение от простого к сложному, то есть когда на основе частных примеров синтезируются общие правила.*

Например, метод математической индукции из школьного курса математики гласит, что если формула верна при n=1,2,…,k, то она будет верна и для n=k+1, то она верна для всех n.

3) ***Аналогия*** - *рассуждение на основе прошлого опыта, т.е. скрытая (неявная) закономерность, которая присуще многим на первый взгляд внешне различающимся объектам.*

Например, было у меня три собаки, одна из них прожила 8 лет, другая - 10 лет и можно сделать вывод, что собака живет от 8 до 15 лет; собаки преданные друзья.

Не монотонность логического вывода означает, что вывод может никогда не закончится и не дать ответа. Другими словами есть не разрешимые проблемы в принципе, либо разрешимы, для некоторого конкретного метода. Не разрешимые проблемы в принципе - это такие проблемы, которые нельзя решить ни одним из существующим методом, но также ни одним из методом, которые когда либо будет создан.

Эвристический метод рассуждения основан на применении правдоподобных рассуждений, которые называют эвристиками.

Эвристика *- это алгоритм или таблица решений, которая базируется на опыте - причем качественном, прошлом и обширном, а не на научных данных или логическом выводе.*

Эвристика отражает особенности того, как такие задачи решает человек, когда он не пользуется строго формальными приемами. Если эти человеческие способы решения удается запрограммировать, то такие программы называются эвристическими. Эвристика часто используются при программировании игр, имитации творческих процессов и т. п. В экспертных системах при формализации профессиональных знаний человека, касающихся способов решения задач в той или иной проблемной области, широко используются те эвристики, которыми руководствуются профессионалы-эксперты.

1.3. Инструментальные средства разработки ИИС.

Средства программирования для ИИ и языки представления знаний

В течение многих лет применяется обширный арсенал языков программирования высокого уровня, ориентированных на удобную и эффективную реализацию различных классов задач, а также широкий спектр трансляторов, обеспечивающих получение качественных исполнительных программ. Все шире используются на современном этапе и методы автоматического синтеза программ. Уже стало обычным применение языково-ориентированных редакторов и специализированных баз данных. И можно сказать, что в рамках технологии программирования уже практически сформировалась

концепция окружения разработки сложных программных продуктов, которая и определяет инструментальные средства, доступные разработчикам.

Необходимость использования средств автоматизации программирования прикладных систем, ориентированных на знания, и в частности ЭС, была осознана разработчиками этого класса программного обеспечения ЭВМ уже давно. По существу, средства поддержки разработки интеллектуальных систем в своем развитии прошли основные стадии, характерные для систем автоматизации программирования.

Оценивая данный процесс с сегодняшних позиций, можно указать в этой области две тенденции. Первая из них как бы повторяет «классический» путь развития средств автоматизации программирования: *автокоды — языки высокого уровня — языки сверхвысокого уровня — языки спецификаций.* Условно эту тенденцию можно назвать *восходящей стратегией* в области создания средств автоматизации разработки интеллектуальных систем. Вторая тенденция, *нисходящая,* связывается со специальными средствами, уже изначально ориентированными на определенные классы задач и методов ИИ. В конце концов, обе эти тенденции, взаимно обогатив друг друга, должны привести к созданию мощного и гибкого инструментария интеллектуального программирования. В этой области характерна концентрация усилий в следующих направлениях:

Разработка систем представления знаний (СПЗ) путем прямого использования широко распространенных языков обработки символьной информации и, все чаще, языков программирования общего назначения.

Расширение базисных языков ИИ до систем представления знаний за счет специализированных библиотек и пакетов.

Создание языков представления знаний (ЯПЗ), специально ориентированных на поддержку определенных формализмов, и реализация соответствующих трансляторов с этих языков.

До недавнего времени наиболее популярным базовым языком реализации ИИС вообще и ЭС, в частности, был ЛИСП. Ниже кратко рассматривается эволюция ЛИСПа, а затем обсуждаются альтернативы этому языку, существовавшие и существующие в области реализации систем, ориентированных на знания. Результаты этого краткого обзора суммированы в виде схемы развития средств автоматизации программирования интеллектуальных систем на рис.1.2.

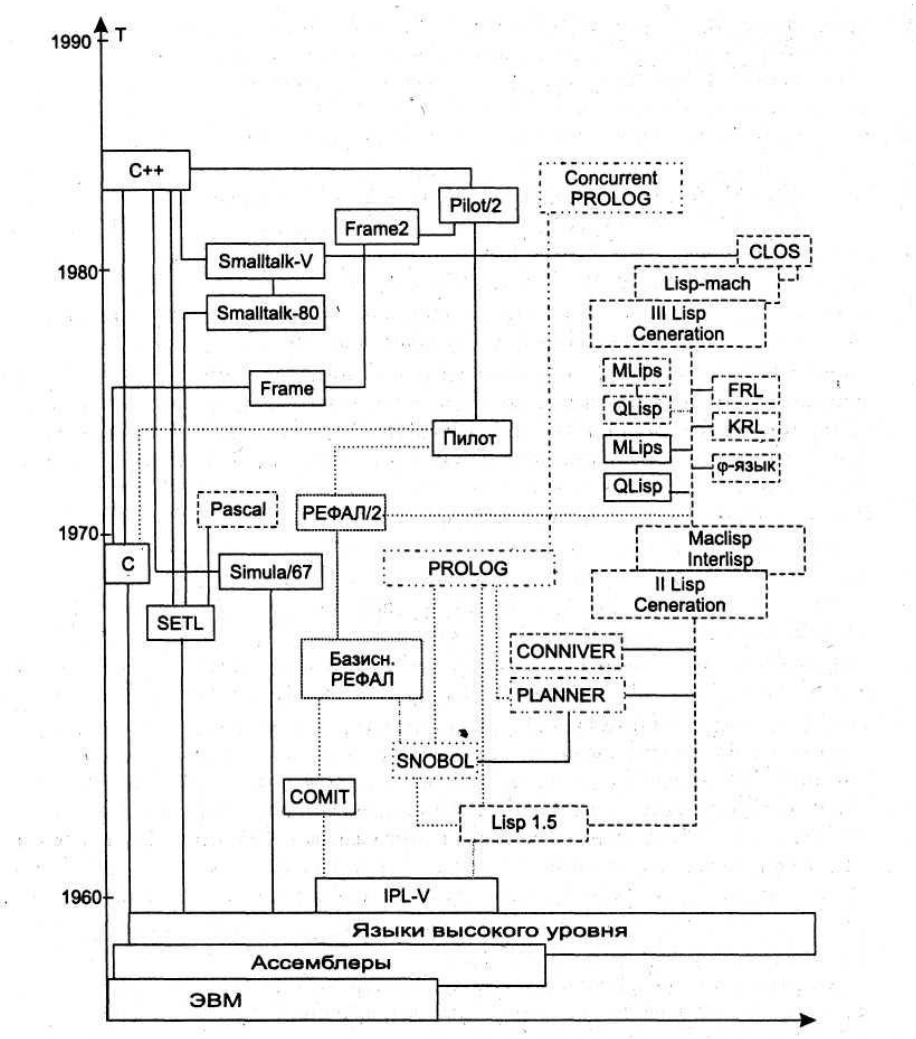


Рис.1.2. Эволюция средств автоматизации программирования интеллектуальных систем

Как известно, язык ЛИСП был разработан в Стэнфорде под руководством Дж. Маккарти в начале 60-х годов и не предназначался вначале для программирования задач ИИ. Это был язык, который должен был стать следующим за ФОРТРАНОМ шагом на пути автоматизации программирования.

По первоначальным замыслам новый язык должен был иметь средства работы с матрицами, указателями и структурами из указателей и т. п. Предполагалось, что первые реализации будут интерпретирующими, но в дальнейшем будут созданы компиляторы. К счастью, для столь амбициозного

проекта не хватило средств. К тому же к моменту создания первых ЛИСП-интерпретаторов в практику работы на ЭВМ стал входить диалоговый режим, и режим интерпретации естественно вписался в общую структуру диалоговой работы.

Примерно тогда же окончательно сформировались и принципы, положенные в основу языка ЛИСП: использование единого спискового представления для программ и данных; применение выражений для определения функций; скобочный синтаксис языка. Процесс разработки языка завершился созданием версии ЛИСП 1.5, которая на многие годы определила пути его развития.

На протяжении всего существования языка было много попыток его усовершенствования за счет введения дополнительных базисных примитивов и/или управляющих структур. Однако, как правило, все эти изменения не прививались в качестве самостоятельных языков, так как их создатели оставались в «лисповской» парадигме, не предлагая нового взгляда на программирование.

После разработки в начале 70-х годов таких мощных ЛИСП-систем, как MacLispи Interlisp, попытки создания языков ИИ, отличных от ЛИСПа, но на той же парадигме, по-видимому, сходят на нет. И дальнейшее развитие языка идет, с одной стороны, по пути его стандартизации (таковы, например, StandartLisp, FranzLisp и CommonLisp), а с другой ~ в направлении создания концептуально новых языков для представления и манипулирования знаниями, погруженных в ЛИСП-среду [Barretal., 1982].

К концу 80-х годов ЛИСП был реализован на всех классах ЭВМ, начиная с персональных компьютеров и кончая высокопроизводительными вычислительными системами. Новый толчок развитию ЛИСПа дало создание ЛИСП-машин,

которые и в настоящее время выпускаются рядом фирм США, Японии и Западной Европы.

Из предыдущего может сложиться впечатление, что язык ЛИСП (а вернее его современные диалекты) — единственный язык ИИ. В действительности, конечно, это не так. Уже в середине 60-х годов, то есть на этапе становления ЛИСПа, разрабатывались языки, предлагающие другие концептуальные основы. Наиболее важными из них в области обработки символьной информации являются, по нашему мнению, СНОБОЛ, разработанный в лабораториях Белла, и язык РЕФАЛ, созданный в ИПМ АН СССР.

Первый из них (СНОБОЛ) — язык обработки строк, в рамках которого впервые появилась и была реализована в достаточно полной мере концепция поиска по образцу. С позиций сегодняшнего дня можно сказать, что язык СНОБОЛ был одной из первых практических реализаций развитой продукционной системы. Наиболее известная и интересная версия этого языка — СНОБОЛ-IV. Здесь, на наш взгляд, техника задания образцов и работа с ними существенно опередили потребности практики. Может быть именно это, а также политика активного внедрения ЛИСПа помешали широкому использованию языка СНОБОЛ в области ИИ.

В основу языка РЕФАЛ положено понятие рекурсивной функции, определенной на множестве произвольных символьных выражений. Базовой структурой данных этого языка являются списки, но не односвязные, как в ЛИСПе, а двунаправленные. Обработка символов ближе, как мы бы сказали сегодня, к продукционному формализму. При этом активно используется концепция поиска по образцу, характерная для СНОБОЛа. Таким образом, РЕФАЛ вобрал в себя лучшие черты наиболее интересных языков обработки

символьной информации 60-х годов. В настоящее время можно говорить о языке РЕФАЛ второго и даже третьего поколения. Реализован РЕФАЛ на всех основных типах ЭВМ и активно используется для автоматизации построения трансляторов, для построения систем аналитических преобразований, а также, подобно ЛИСПу, в качестве инструментальной среды для реализации языков представления знаний.

В начале 70-х годов появился еще один новый язык, способный составить конкуренцию ЛИСПу при реализации систем, ориентированных на знания, — **язык ПРОЛОГ**. Он не дает новых сверхмощных средств программирования по сравнению с ЛИСПом, но поддерживает другую модель организации вычислений. Подобно тому, как ЛИСП скрыл от программиста устройство памяти ЭВМ, ПРОЛОГ позволил ему не заботиться (без необходимости) о потоке управления в программе. ПРОЛОГ предлагает такую парадигму мышления, в рамках которой описание решаемой задачи представляется в виде слабо структурированной совокупности отношений. Это удобно, если число отношений не слишком велико и каждое отношение описывается небольшим числом альтернатив. В противном случае ПРОЛОГ-программа становится весьма сложной для понимания и модификации. Возникают и проблемы эффективности реализации языка, так как в общем случае механизмы вывода, встроенные в ПРОЛОГ, обеспечивают поиск решения на основе перебора возможных альтернатив и декларативного возврата из тупиков. ПРОЛОГ разработан в Марсельском университете в 1971 г.. Однако популярность он стал приобретать лишь в начале 80-х годов, когда благодаря усилиям математиков был обоснован логический базис этого языка, а также в силу того, что в японском проекте вычислительных систем V поколения ПРОЛОГ был выбран в качестве базового для машины вывода. В настоящее

время ПРОЛОГ завоевал признание и на американском континенте, хотя уступает в популярности ЛИСПу и даже специальным продукционным языкам, широко используемым при создании ЭС.

Инструментальные пакеты для ИИС

Развитые среды автоматизации программирования на базе языков символьной обработки являются необходимым технологическим уровнем систем поддержки разработки прикладных интеллектуальных систем. Как правило, такие среды покрывают (и то частично) подсистему автоматизации проектирования и программирования. Вот почему следующим этапом в развитии инструментальных средств стала ориентация на среды поддержки разработок интеллектуальных систем.

Анализ существующих инструментальных систем показывает, что сначала в области ИИ более активно велись работы по созданию интеллектуальных систем автоматизированного синтеза исполнительных программ. И это естественно, если иметь в виду, что инструментарий ИИ является, по существу, эволюционным развитием систем автоматизации программирования. При этом основная доля мощности и интеллектуальности такого инструментария связывалась не с его архитектурой, а с функциональными возможностями отдельных компонентов той или иной технологической среды. Большое значение при разработке инструментария для ИИ уделялось и удобству сопряжения отдельных компонентов. Пожалуй, именно здесь были получены впечатляющие результаты и именно здесь наиболее широко использовались последние достижения теории и практики программирования, такие, например, как синтаксически-ориентированное редактирование и

инкрементная компиляция. Вместе с тем подавляющее большинство современных инструментальных систем «не знают», что проектирует и реализует с их помощью пользователь. И с этой точки зрения можно сказать, что все такие системы являются не более чем «сундучками» с инструментами, успех использования которых определяется искусством работающего с ними мастера. Примерами подобных сред служит подавляющее большинство инструментальных пакетов и систем-оболочек для создания экспертных систем типа EXSYS, GURU и др. Однако не они определяют на сегодняшний день уровень достижений в этой области. К первому эшелону большинство специалистов относит системы ART, KEE и KnowledgeCraft. Заметим, что в последнее время в класс самых мощных и развитых систем вошла и среда G2. Все эти системы, во-первых, отличает то, что это, безусловно, интегрированные среды поддержки разработки интеллектуальных (в первую очередь, экспертных) систем. И вместе с тем для этих систем характерно не эклектичное объединение различных полезных блоков, но тщательно сбалансированный их отбор, что позволило сделать первые шаги от автоматизации программирования систем ИИ к технологическим системам поддержки проектирования сначала экспертных, а затем и других интеллектуальных систем. Остановимся чуть подробнее на одной системе из «большой тройки» — ART.

В середине 80-х годов **система ART** была одной из самых современных интегрированных сред, поддерживающих технологию проектирования систем, основанных на правилах. В ней существует ясное и богатое разнообразие типов правил. Различные типы правил элегантно вводятся с помощью мощного механизма «точек зрения» (Viewpoint). Этот механизм фактически является очень близким к системе,

основанной на истинности предположений (truthmaintanancesystem), которая, по-видимому, является развитием идей KEEWorlds+ в системе KEE. По существу, ART является пакетом разработчика. При этом возможные ограничения в использовании ART вызваны не свойствами самой системы, а философией базового метода представления знаний.

ART часто представляют в качестве лучшей ИС для создания экспертных систем, но следует понимать, что хорошо эта среда отвечает лишь всем требованиям подхода поверхностных знаний. Благодаря компилятору правил система вывода в ART является быстрой по своей природе. Главные стратегии структуры управления поиском решений обеспечиваются, и некоторая гибкость в управлении поиском остается инженеру по знаниям. Чисто декларативные таксономические фреймы языка интегрируются с системой правил, но в ART не существует действительно процедурных фреймов, которые могли бы позволить объединить предметные описания с продукционными. В этом отношении объектно-ориентированное программирование с образцами и возможностями моделирования могло бы быть более полезным. Нельзя сказать, что подход, основанный на моделях, не осуществим в ART. Однако, кажется, что другие ИС более эффективны для этих целей.

Первые версии ART опирались на язык ЛИСП, последние реализованы непосредственно на С. Это увеличивает эффективность периода исполнения ART. Введены в новые версии и некоторые другие усовершенствования. Они касаются в основном выразительной силы формализма, основанного на фреймах, и увеличивают адекватность ART по отношению к методу аргументации,

основанному на модели. Имеется информация, что в ART включен и объектно-ориентированный подход.

Примеры ИИС. Приведем несколько примеров широко известных ИИС.

IntelligentHedger: основанный на знаниях подход в задачах страхования от риска. Фирма: InformationSystemDepartment, NewYorkUniversity. Проблема огромного количества постоянно растущих альтернатив страхования от рисков, быстрое принятие решений менеджерами по рискам в ускоряющемся потоке информации, а также недостаток соответствующей машинной поддержки на ранних стадиях процесса разработки систем страхования от рисков предполагает обширную сферу различных оптимальных решений для менеджеров по риску. В данной системе разработка страхования от риска сформулирована как многоцелевая оптимизационная задача. Данная задача оптимизации включает несколько сложностей, с которыми существующие технические решения не справляются. Краткие характеристики: система использует объектное представление, охватывающее глубокие знания по управлению риском и облегчает эмуляцию первичных рассуждений, управляющих риском, полезных для выводов и их объяснений.

Системарассужденийвпрогнозированииобменавалют. Фирма: Department of Computer Science City Polytechnic University of Hong Kong. Представляет новый подход в прогнозировании обмена валют, основанный на аккумуляции и рассуждениях с поддержкой признаков, присутствующих для фокусирования на наборе гипотез о движении обменных курсов. Представленный в прогнозирующей системе *набор признаков* — это заданный набор экономических значений и

различные наборы изменяющихся во времени параметров, используемых в модели прогнозирования. Краткие характеристики: математическая основа примененного подхода базируется на теории Демпстера—Шейфера.

Nereid: Система поддержки принятия решений для оптимизации работы с валютными опционами. Фирма: NTT Data, The Tokai Bank, Science University of Tokyo. Система облегчает дилерскую поддержку для оптимального ответа как один из возможных представленных вариантов; более практична и дает лучшие решения, чем обычные системы принятия решений. Краткие характеристики: система разработана с использованием фреймовой системы CLP, которая легко интегрирует финансовую область в приложение ИИ. Предложен смешанный тип оптимизации, сочетающий эвристические знания с техникой линейного программирования. Система работает на Sun-станциях.

PMIDSS: Система поддержки принятия решений при управлении портфелем. Разработчики: Финансовая группа Нью-Йоркского университета. Решаемые задачи: выбор портфеля ценных бумаг; долгосрочное планирование инвестиций. Краткие характеристики: смешанная система представления знаний, использование разнообразных механизмов вывода: логика, направленные семантические сети, фреймы, правила.

1.4. Инструментальное средство представления знаний – язык ПРОЛОГ

Общие сведения о языке ПРОЛОГ.

В русском языке слово "пролог" многозначно. В древности на Руси прологом называли специальный вид дидактической литературы. В наше время, под прологом

понимают вступление к литературному произведению. Наконец, существует третье значение этого слова, являющееся аббревиатурой. Оно получено из фразы ПРОграммирование в ЛОГике. На этом значении слова "пролог" мы и остановимся.

Связь между логикой и программированием впервые проявилась в процессе формализации математики. Оказалось, что между вычислениями и доказательствами существует взаимодействие, состоящее в том, что всякое доказательство задает построение или вычисление того объекта, существование которого доказывается. С другой стороны, развитие самого программирования и усложнение реальных программ привели к необходимости формально выразить и доказать их свойства. Для этого используется математическая логика. Концепция логического программирования является следствием сближения логики и программирования. Свое практическое воплощение она получила в языке ПРОЛОГ.

Появившись в начале 70-х годов в качестве экспериментальной разработки лабораторий искусственного интеллекта университетов Марселя и Эдинбурга ПРОЛОГ на протяжении почти десяти лет оставался известным лишь узкому кругу специалистов. Однако, тот факт, что логическое программирование и сам ПРОЛОГ были положены в основу японского проекта ЭВМ пятого поколения, привлек к этому языку всеобщее внимание.

Основной принцип использования языка Пролог состоит в том, что нужно подробно, на логически точном языке, описать условие задачи. Решение ее получается в результате определенного рутинного процесса, который исполняется компьютером. В этом заключается принципиальное отличие Пролога от традиционных языков

программирования, которые требуют описания того как должен быть вычислен результат, или другими словами, требуют описания процедуры решения задачи. Поэтому, кстати, традиционные языки программирования: Ада, Паскаль, Фортран - принято называть процедурными, а Пролог непроцедурным языком. Система Пролог-Д - реализация концепции логического программирования для учебных персональных ЭВМ.

Текст на Прологе-Д содержит сообщения двух типов: факты и правила. Факт - это синтаксическая конструкция, которая позволяет накапливать информацию, а правило - конструкция, с помощью которой можно делать заключение или вывод. Таким образом факты и правила связывают объекты и отношения между ними.

Совокупность фактов и правил образуют базу знаний. В отличие от процедурных языков, порядок фактов и правил не имеет, существенного значения для правильности результата, за исключением нескольких случаев, о которых будет сказано особо.

Для запуска системы необходимо задать вопрос. Вопрос- это факт, которому предшествует символ "?". База знаний и вопрос образуют программу на языке Пролог-Д. Следовательно, программирование на Прологе-Д - это умение создать систему фактов и правил, характеризующих решаемую задачу и умение задать нужный вопрос к этой системе.

Принципы работы языка ПРОЛОГ.

Для пояснения основных концепций языка ПРОЛОГ рассмотрим задачу, решение которой выполним с помощью обратной цепочки рассуждений. Пусть задано правило:

ЕСЛИ *мужчина по имени X со своей матерью по имени M находятся в комнате,*

И *женщина по имени Y со своей матерью по имени M находятся в комнате,*

ТО *мужчина по имени X и женщина по имени Y - брат и сестра.*

Это правило содержит ряд условий, позволяющих определить, действительно ли в комнате находятся брат и сестра. Для решения такой задачи понадобятся некоторые факты. Пусть эти факты представлены в виде двух списков: списка имен всех находящихся в комнате мужчин и их матерей, и списка имен всех находящихся в комнате женщин и их матерей. Воспользовавшись этими списками, можно определить мужчин и женщин имеющих одну и ту же мать, а исходя из приведенного правила сделать вывод об их родстве. Рассмотрим задачу подробнее.Сведем известные факты в таблицы.

Таблица 1		Таблица 2	
Имена мужчин	Имена их матерей	Имена женщин	Имена их матерей
Дэниел Ларри Майкл Пит	Диана Лори Бонни Джейн	Энн Лаура Сьюзи	Лори Ширли Бонни

Просмотрев таблицы, можно предположить, что братом и сестрой могут быть:

Ларри и Энн (мать обоих носит имя Лори) и Майкл и Сьюзи (мать обоих носит имя Бонни).

Попробуем представить логику работы программы. Прежде всего предположим, что часть правила, идущая за словом ТО, верна и действительно существуют мужчина X и женщина Y, приходящиеся друг другу братом и сестрой. Выполним обратную цепочку рассуждений исходя из истинности такого заключения. Подтвердить истинность заключения можно, доказав истинность каждого из условий, входящих в правило. Первое условие гласит: “Мужчина по имени X со своей матерью по имени M находятся в комнате”. Сначала попытаемся отыскать имена для мужчины X и его матери M в табл.1. Это легко сделать, выбрав первую же строчку, т.е. X присвоим значение “Дэниел”, а M - значение “Диана”:

Мужчина по имени Дэниел со своей матерью по имени Диана находятся в комнате.

Таблица 1		Таблица 2	
Имена мужчин	Имена их матерей	Имена женщин	Имена их матерей
Дэниел	Диана	Энн Лаура Сьюзи	Лори Ширли Бонни

Теперь нужно попытаться доказать истинность второго условия правила: «Женщина по имени Y со своей матерью по имени M находятся в комнате». Но значение Муже выбрано-Диана, и, следовательно, нужно найти в табл.2 имя женщины,

чью мать зовут Диана. Но имени Диана нет в правой половине табл. 2. Значит, нужно вернуться назад и для первого условия правила отыскать в табл. 1 другие значения для X и M. Выберем следующую строку с именами Ларри и Лори, то есть присвоим значения:

X = Ларри,

M = Лори.

Таблица 1		Таблица 2	
Имена мужчин	Имена их матерей	Имена женщин	Имена их матерей
Дэниел Ларри	Диана Лори	Энн Лаура Сьюзи	Лори Ширли Бонни

Теперь для подтверждения второго условия правила в табл. 2 нужно найти имя женщины, мать которой носит имя Лори. Из табл. 2 видно, что это Энн:

Y = Энн

Обратившись к части *TO* правила, можно заключить, что Ларри и Энн брат и сестра.

Подытожим выполненные действия.

1. В качестве заключения мы взяли часть TO правила, попытались подтвердить его истинность на основании истинности входящих в правило условий.

2. Подтверждение истинности первого условия.

3. Истинность первого условия определяется значением входящих в него переменных, которые выбираются из первой таблицы.

4. Подтверждение истинности второго условия.

5. Сделать это не удалось, так как во второй таблице не оказалось подходящих данных.

6. Пришлось вернуться назад и попытаться добиться истинности первого условия, а затем и второго с новыми значениями входящих в них переменных.

7. С новыми значениями переменных X и Y последняя цель была успешно достигнута.

8. Заключение в части ТО правила оказалось истинным при новых выбранных значениях переменных.

Представление правил в языке ПРОЛОГ.

Выше была продемонстрирована логика работы программы на языке PROLOG. Программа автоматически выполняет поиск информации и обратную цепочку рассуждений, используя механизм возврата. Для работы программы необходимо преобразовать формат правил, входящих в базу знаний, в формат, принятый в языке PROLOG. Между этими форматами существует взаимооднозначное соответствие.

Правила на языке ПРОЛОГ записывается в виде предложений. Для выше рассматриваемого примера предложение имеет вид (рис.1.3).

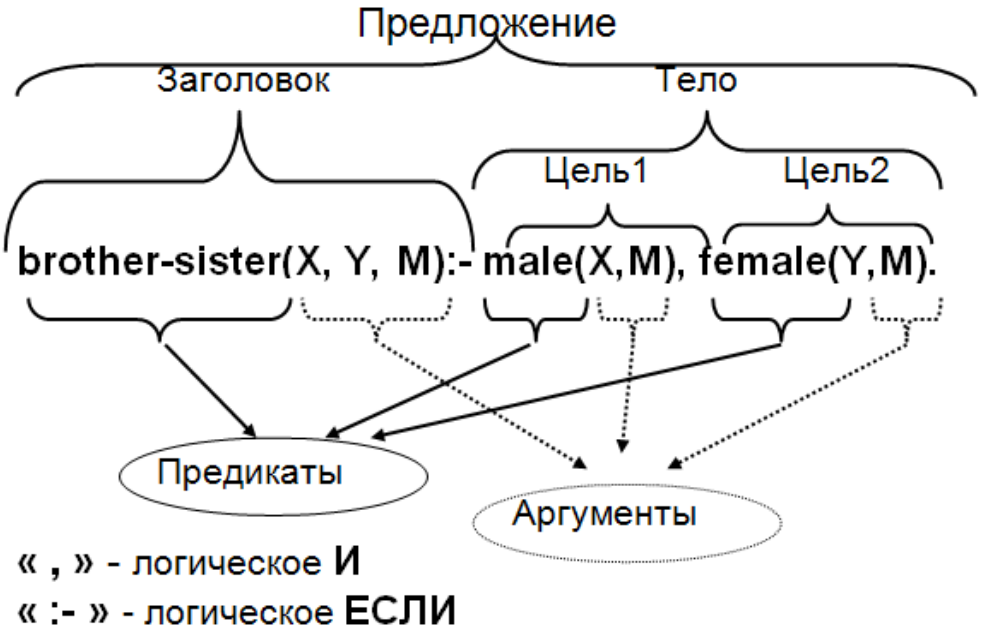


Рис.1.3. Правило в языке ПРОЛОГ.

На выше представленном примере рассмотрим синтаксис и некоторые свойства языка ПРОЛОГ.

1. Полная запись правила на языке ПРОЛОГ называется **предложением.** В языке существует два типа предложений: для записи правил и для записи фактов (предложения для записи фактов будут рассмотрены позднее). Приведенное выше предложение - это предложение для записи правил.

2. Часть предложения языка, соответствующая части ТО правила, записывается слева от знака “:-” и называется **заголовком** правила: brother-sister(X, Y, M)):-

3. Часть ЕСЛИ правила в предложении языка записывается справа от знака “:-”. Предикаты male(X,M), female(Y,M)и называется **телом** предложения.

4. Тело предложения состоит из отдельных частей, разделенных запятыми. Каждая часть в теле предложения

называется **целью.** Запятая, разделяющая цели в предложении, имеет смысл логического**И.**

5. Все предложения языка ПРОЛОГ заканчиваются точкой.

6. Слова, находящиеся до открывающейся левой скобки, т.е. слова brother-sister, male, female, называются **предикатами.** Предикаты всегда начинаются со строчной буквы.

7. Слова, взятые в круглые скобки и идущие следом за именем предиката, называются **аргументами.** Для предиката male аргументом являются переменные X и M.

8. Символы ":-" в языке имеют смысл логического ЕСЛИ.

Используя это правило и факты, собранные в табл.1 и табл.2, напишем программу на языке PROLOG, т.е. в сущности создадим базу знаний:

male(Дэниел,Диана).

male(Ларри,Лори).

male(Майкл,Бонни).

male(Пит,Джейн).

female(Энн,Лори).

female(Лаура,Лирли).

female(Льюзи,Бонни).

brother-sister(X, Y, M):- male(X,M), female(Y,M).

В предикате brother-sister аргументы X, Y, M могут принимать значения имен предполагаемых брата и сестры и их матери соответственно. Предикат male эквивалентен утверждению "X - это имя мужчины, мать которого носит имя M", а предикат female - утверждению " Y - это имя женщины, мать которой носит имя M". Введя эту программу

в компьютер, пользователь может приступить к диалогу с ней. PROLOG выдает на терминал символ приглашения к работе “?”, после которого пользователь может ввести вопрос:

? - brother-sister(X, Y, M).

Программа выведет ответ:

X = Ларри

Y = Энн

M = Лори

Таким образом, будет установлено родство Энн и Ларри, поскольку у них одна и та же мать, носящая имя Лори.

Глава 2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ.

2.1. Данные, знания и представление знаний.

Данные, знания и база знаний.

При изучении интеллектуальных систем традиционно возникает вопрос — что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых ЭВМ. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным.

Данные *— это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.*

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

D1 — данные как результат измерений и наблюдений;

D2 — данные на материальных носителях информации (таблицы, протоколы, справочники);

D3 — модели (структуры) данных в виде диаграмм, графиков, функций;

D4 — данные в компьютере на языке описания данных;

D5 — базы данных на машинных носителях информации.

Знания*— это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.*

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, подученного в результате практической деятельности. С информационной точки зрения можно дать следующее определение знаний.

Знания - совокупность сведений, образующих целостное описание, соответствующее некоторому уровню осведомленности об описываемом вопросе, предмете, проблеме и т.д.

При обработке на ЭВМ знания трансформируются аналогично данным.

Z1 — знания в памяти человека как результат мышления;

Z2 — материальные носители знаний (учебники, методические пособия);

Z3 — *поле знаний* — условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

Z4 — знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы — см. далее);

Z5 — *база знаний на машинных носителях информации.*

Часто используется еще такое определение знаний. ***Знания***— это хорошо структурированные данные, или данные о данных, или метаданные.

Особенности знаний. Знания состоят из понятий, которые имеют имя, структуру и набор признаков, связей между понятиями и утверждений о свойствах понятий. Это

форма представления информации в ЭВМ, имеющая такие особенности, отличающие знания от данных, как:

- внутренняя интерпретируемость (каждая информационная единица имеет уникальное имя, по которому система ее находит и отвечает на запросы, в которых это имя используется);
- структурированность (одни информационные единицы включаются в состав других);
- связность (можно задавать структурные, функциональные, временные, каузальные, пространственные отношения между информационными единицами);
- семантическая метрика (можно задать отношения, определяющие ситуационную близость, ассоциативные связи между информационными единицами);

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. *Интенсионал* понятия — это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому объекту. Это есть определение через данные, или *экстенсионал понятия.*

Например, понятие «персональный компьютер». Его интенсионал: «Персональный компьютер - это дружественная ЭВМ, которую можно поставить на стол и купить менее чем за $2000-3000». Экстенсионал этого понятия: «Персональный компьютер — это Mac, IBMPC, Sinkler...»

Типы и виды знаний.

Существует довольно большое количество признаков, по которым можно разделить знания на типы и виды. С точки зрения интеллектуальных систем знания могут быть классифицированы по следующим категориям:

Поверхностные — знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. Пример, поверхностные знания: «Если нажать на кнопку звонка, раздастся звук. Если болит голова, то следует принять аспирин».

Глубинные знания - этоабстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов. Пример глубинных знаний: *«Принципиальная электрическая схема звонка и проводки. Знания физиологов и врачей высокой квалификации о причинах, видах головных болей и методах их лечения».*

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Процедурные знания описывают последовательности действий, которые могут использоваться при решении задач. Это, например, программы для ЭВМ, словесные записи *алгоритмов,* инструкция по сборке некоторого изделия.

Декларативные знания — это все знания, не являющиеся процедурными, например, статьи в толковых словарях и энциклопедиях, формулировки законов в физике, химии и других науках, собрание исторических фактов и т. п.

В отличие от процедурных знаний, отвечающих на вопрос: "Как сделать *X7",* декларативные знания отвечают, скорее, на вопросы: "Что есть *XI"* или "Какие связи имеются между *X*и У?", "Почему *XI"* и т. д.

Исторически первичными были процедурные знания, то есть знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы. Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знаний сосредоточивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных знаний.

Знания о предметной области - совокупность сведений о предметной области, хранящихся в базе знаний интеллектуальной системы. В эти знания входят факты, относящиеся к предметной области, закономерности, характерные для нее, гипотезы о возможных связях между явлениями, процессами и фактами в ней, процедуры для решения типовых задач в данной проблемной области. Знания о предметной области вводит в базу знаний инженер по знаниям. В процессе функционирования интеллектуальной системы они могут пополняться.

Знания прагматические - знания о способах решения задач в определенной предметной области.

Знания эвристические - знания, накапливаемые интеллектуальной системой в процессе ее функционирования, а также знания, заложенные в ней априорно, но не имеющие статуса абсолютной истинности в данной проблемной области. Часто знания эвристические связаны с отражением в базе знаний человеческого (неформального) опыта решения задач.

Знания экспертные - знания, которыми располагает специалист в некоторой предметной области.

База знаний. Принципиальное отличие систем искусственного интеллекта состоит в том, что для такого рода систем программист не готовит конкретные программы для исполнения. Человек лишь дает машине нужное задание, а программу, выполняющую это задание, система должна построить сама. Для этого нужны знания как о предметной области, к которой относится задание, так и о том, как строятся программы. Все эти знания хранятся в интеллектуальных системах в специальном блоке, называемом базой знаний. База знаний основа любой интеллектуальной системы.

Знания, хранящиеся в базе знаний, записываются в специальной формализованной форме. В базе знаний могут реализоваться процедуры обобщения и корректировки хранимых знаний, а также процедуры, создающие новые знания на основании тех, которые уже там имеются. Знания, хранимые в базе, отличаются от данных, хранящихся в *базе данных,* несколькими особенностями. Во-первых, структура знаний намного сложнее структуры данных. Во-вторых, знания обладают свойством внутренней активности. Чтобы пояснить это принципиальное свойство знаний, вспомним, что при обычной работе с компьютером данные всегда играют пассивную роль.

База знаний - *совокупность программных средств, обеспечивающих поиск, хранение, преобразование и запись в памяти ЭВМ сложно структурированных информационных единиц - знаний.*

База знаний замкнутая - база знаний, содержимое которой в процессе функционирования не изменяется.

Логический вывод в такой базе эквивалентен выводу в формальной системе и обладает свойством монотонности, т.е. ранее выведенные утверждения остаются верными на весь период функционирования базы знаний.

База знаний открытая - база знаний, позволяющая в процессе ее функционирования пополнять содержимое базы и убирать знания из базы. Свойство открытости приводит к тому, что вывод в такой базе является немонотонным, т.е. истинность выведенных в ней утверждений может меняться в процессе работы системы с такой базой.

Требования к моделям знаний.

Представление знаний – важная проблема, т.к. она влияет на свойства и характеристики ИИС.

Представление знаний – это формализация знаний для их ввода в базу знаний.

Действия над знаниями осуществляются программным путём, поэтому знания должны быть представлены формальными моделями, к которым предъявляются два основных требования:

1.ОДНОРОДНОСТЬ (единообразие) - дает возможность упростить механизм управления логическим выводом и знанием;

2.ПОНЯТНОСТЬ для экспертов и пользователей. Иначе затрудняется процесс приобретения знаний и оценка их качества.

Основные типы моделей представления знаний

1. Логическая модель представляет собой формальную систему в виде логического исчисления, как правило,

исчисление предикатов первого порядка. Все знания о предметной области описываются в виде формул этого исчисления или правил вывода. Описание в виде формул дает возможность представить декларативные знания, а правила вывода — процедурные знания.

2. Продукционная модель (модель правил*)* - это модель, основанная на правилах, которая позволяет представить знания в виде предложений типа **«Если** (условие), **то**(действие)». Знания представлены совокупностью правил «если-то».

Различают два противоположных типа продукционной модели:

а) с прямыми выводами – для решения задач диагностического характера;

б) с обратными выводами – для решения задач проектирования и прогнозирования.

3. Сетевые модели**:**

а) *Семантическая сеть* – это модель, в которой структура знаний предметной области формализуется в виде ориентированного графа с помеченными вершинами и дугами. Вершины графа обозначают понятия различных категорий: объекты, события, свойства, операции, а дуги – отношения между ними.

б) *Фреймовая модель* – это модель, в которой структура знаний предметной области формализуется в виде совокупности взаимосвязанных фреймов, описывающих объекты, а свойства этих объектов и факты, относящиеся к ним, описываются в структурных элементах фрейма.

2.2. Модели представления знаний

2.2.1. Логическая модель представления знаний

Логическая модель представляет собой формальную систему — некоторое логическое исчисление как правило, исчисление предикатов первого порядка, когда предметная область или задача описывается в виде набора аксиом.

Все знания о предметной области описываются в виде формул этого исчисления или правил вывода. Описание в виде формул дает возможность представить декларативные знания, а правила вывода — процедурные знания.

Предикат висчислении предикатов специальный знак, отражающий определенное отношение между конечным множеством сущностей - аргументов. Предикат первого порядка имеет два состояния - истина и ложь.

Рассмотрим в качестве примера знание: "Когда температура в печи достигает 120° и прошло менее 30 минут с момента включения печи, давление не может превосходить критическое. Если с момента включения печи прошло более 30 мин, то необходимо открыть вентиль №2".

Логическая модель представления этого знания имеет вид:

$P(p = 120) \wedge T(t<30) \rightarrow (D<D_{Kp});$

$P(p = 120) \wedge T(t>30) => F(№2).$

В этой записи использованы следующие обозначения:

$P(p = 120)$ — предикат, становящийся истинным, когда температура достигает 120°;

$T(t<30)$ — предикат, остающийся истинным в течение 30 мин с начала процесса;

$T(t>30)$ — предикат, становящийся истинным по истечении 30 мин с начала процесса;

$D<D_{кр}$— утверждение о том, что давление ниже критического;

F(№2) — команда открыть вентиль №2.

Кроме того, в этих записях использованы типовые логические связки И ($\wedge$), импликации ($\rightarrow$) и логического следования (=>).

Первая строчка в записи представляет декларативные знания, а вторая — процедурные.

Языки представления знаний логического типа широко использовались на ранних стадиях развития интеллектуальных систем, но вскоре были вытеснены (или, во всяком случае, сильно потеснены) языками других типов. Объясняется это громоздкостью записей, опирающихся на классические логические исчисления. При формировании таких записей легко допустить ошибки, а поиск их очень сложен. Отсутствие наглядности, удобочитаемости (особенно для тех, чья деятельность не связана с точными науками) затрудняло распространение языков такого типа.

Исчисление предикатов 1-го порядка в промышленных экспертных системах практически не используется. Эта логическая модель применима в основном в исследовательских «игрушечных» системах, так как предъявляет очень высокие требования и ограничения к предметной области.

2.2.2. Продукционная модель представления знаний.

Продукционная модель (модель правил) - это модель, основанная на правилах, в которой знания представлены в виде предложений типа **«Если** (условие), **то** (действие)».

Под «условием» *(антецедентом)* понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под «действием» *(консеквентом)* — действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия и терминальными или целевыми, завершающими работу системы).

Основу модели составляют системы продукций. Каждая ***продукция***в наиболее общем виде записывается как стандартное выражение следующего вида:

"Имя продукции":

Имя сферы;

Предусловие;

Условие для ядра;

Ядро: «Если*A,* то *B»;*

Постусловие.

В наиболее простом виде продукция может состоять лишь из имени (например, ее порядкового номера в системе продукций) и ядра.

Имя продукции может выражаться в виде номера или слова (словасочетания). Служит для определения местоположения в системе продукций.

Имя сферы указывает ту предметную область, к которой относятся знания, зафиксированные в данной продукции. В

интеллектуальной системе может храниться совокупность знаний (ее называют *базой знаний)*, относящихся к разным областям (например, знания о различных заболеваниях человека или знания из различных разделов математики). Ясно, что если в данный момент решается задача из области физики твердого тела или из геометрии треугольника, то надо использовать знания, относящиеся именно к этой области. Сферы и выделяют такие подобласти знаний.

Предусловие определяет необходимые предпосылки применения условия для ядра продукции. Предусловия может и не быть вовсе.

Условие для ядра определяет те ситуации, при выполнении которых можно (надо) проверять наличие или истинность*A*в ядре продукции. Следующий пример иллюстрирует употребление условия для ядра: *a, b, c — стороны треугольника; если* $c = \sqrt{a^2 + b^2}$ *, то треугольник является прямоугольным.* Ясно, что при другой интерпретации *a, b*и *c* неимеет никакого смысла использовать данное ядро продукции.

Ядро - основная часть продукции. Ядро имеет вид: "Если*A,* то *B",* где *A* и *B* могут иметь разные значения. Остальные элементы, образующие продукцию, носят вспомогательный характер. Несколько примеров ядра:

"Если сверкает молния, то гремит гром".

"Если в доме вспыхнул пожар, то вызывайте по телефону 01 пожарную команду".

"Если в путеводителе указано, что в городе есть театр, то надо пойти туда".

Первый пример иллюстрирует тот случай, когда ядро продукции описывает причинно-следственную связь явлений*A* и *B*. Во втором примере *A* и *B* представляют собой некоторые действия. В третьем примере*A*— это некоторые знания, а *B* — действие. Возможны и другое варианты ядра продукции. Таким образом, при помощи ядер можно представлять весьма разнообразные знания, как декларативные знания, так и процедурные, хотя сама форма продукций весьма удобна для задания именно процедурных знаний.

Постусловие определяет специфику ядра. Используется редко. Обычно содержит пояснения.

Пример продукции декларативного знания: «Пусть a, b, c — стороны треугольника и при выполнении равенства c = $\sqrt{a^2+b^2}$ треугольник является прямоугольным».

Имя продукции: **№5**

Имя сферы: *Геометрия*;

Предусловие: *Фигуры*;

Условие для ядра: *a, b, c — стороны треугольника*;

Ядро: **Если *c* =$\sqrt{a^2+b^2}$, то треугольник прямоугольный;**

Постусловие: Теорема Пифагора.

Основные достоинства систем продукционных:

Простота создания и понимания, отдельных правил.

Простота понимания и модифицирования знаний.

Простота программной реализации механизма логического вывода.

Слабые стороны систем: 1.Неясность взаимоотношений правил. 2.Сложность оценки целостного образа знаний. 3.Весьма низкая эффективность обработки. 4.Существенные отличия от человеческих систем знаний. 5.Отсутствие гибкости в логическом выводе.

Данная модель широко применяется в экспертных системах, в том числе промышленного (коммерческого) уровня.. Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; «оболочки» или «пустые» ЭС — EXSYSProfessional, Kappa, ЭКСПЕРТ; ЭКО и др.).

2.2.3. Семантическая модель представления знаний

Термин «семантическая»означает «смысловая», а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков.

Семантическая сеть – это модель, в которой структура знаний предметной области формализуется в виде ориентированного графа вершины которого — понятия, а дуги — отношения между ними

В качестве понятий обычно выступают абстрактные или конкретные объекты, события , свойства, операции. Отношения — это связи различного типа. Существуют отношения разных типов:

- логические (дизъюнкция, конъюнкция, отрицание, импликация);

- теоретико-множественные (часть – целое, множество – подмножество, класс – элемент класса, пример элемента класса);
- функциональные (количественные, временные, пространственные и другие характеристики: объект-свойство, свойство-значение);
- квантификационные (логические кванторы общности и существования, нелогические кванторы, например, много, несколько);

Наиболее часто в семантических сетях используются следующие отношения:

- связи типа «часть— целое» («класс— подкласс», «элемент—множество», и т. п.). Отношения “является” и “имеет часть” определяют иерархическую структуру, в которой свойства "высших" понятий автоматически переносятся на "низшие" понятия. Это позволяет избежать дублирования информации в сети.;
- функциональные связи (определяемые обычно глаголами «производит», «влияет»...);
- количественные (больше, меньше, равно...);
- пространственные (далеко от , близко от, за, под, над...);
- временные (раньше, позже, в течение...);
- атрибутивные связи (иметь свойство, иметь значение);

Классификация семантических сетей:

Можно предложить несколько классификаций семантических сетей, связанных с типами отношений между понятиями.

По количеству типов отношений:

Однородные (с единственным типом отношений).

Неоднородные (с различными типами отношений).

По типам отношений:

Бинарные.(в которых отношения связывают два объекта).

N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поставленный запрос к базе.

В качестве примера, рассмотрим текст, который содержит некоторые знания (декларативные): *«Иванов имеет личный автомобиль «Волга» красного цвета с мощностью двигателя 75 л.с.»*

На рис.1.3 изображена семантическая сеть. В качестве вершин тут выступают понятия: «человек», «Иванов», «Волга», «автомобиль», «вид транспорта» и «двигатель».

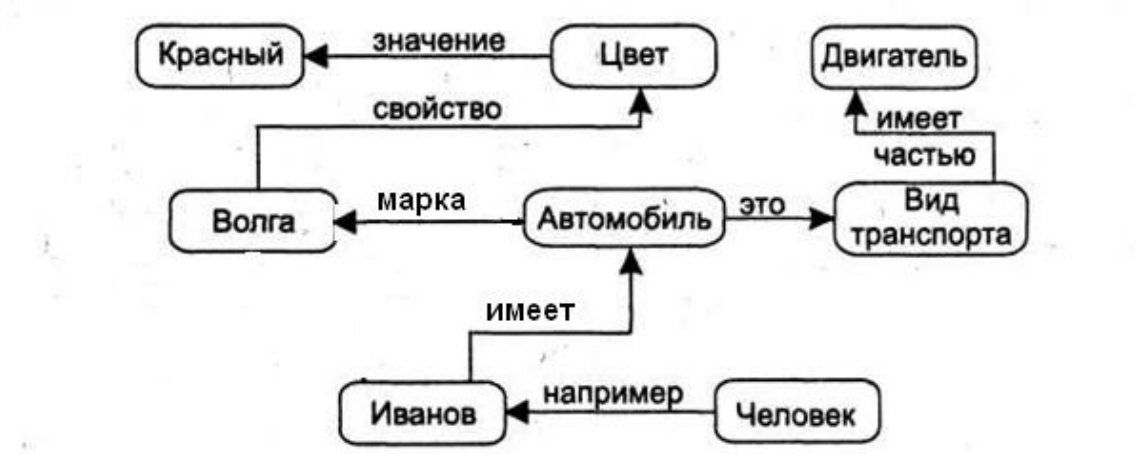


Рис. 1.3. Семантическая сеть.

Основным преимуществом семантической модели является то, что она более других соответствует современным

представлениям об организации долговременной памяти человека.

Недостатком этой модели является сложность организации процедуры поиска вывода на семантической сети.

Для реализации семантических сетей существуют специальные сетевые языки, например NET, язык реализации систем SIMER+MIR [и др. Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний — PROSPECTOR, CASNET, TORUS.

2.2.4. Фреймовая модель представления знаний

Понятие и структура фрейма.

Теория фреймов - это парадигма для представления знаний с целью использования этих знаний компьютером. Теория фреймов (теория представления знаний фреймами) была разработана *Марвином Минским* (США) в 70-е г.г. XX в. В ее основе лежит восприятие фактов посредством сопоставления полученной извне информации с конкретными элементами и значениями, а также с рамками, определенными для каждого объекта в памяти человека.Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Минский разработал такую схему, в которой информация содержится в специальных ячейках, называемых фреймами, объединенными в сеть, называемую системой фреймов. Новый фрейм активизируется с наступлением новой ситуации. Отличительной его чертой является то, что

он одновременно содержит большой объем знаний и в то же время является достаточно гибким для того, чтобы быть использованным как отдельный элемент БД. Термин «фрейм» был наиболее популярен в середине семидесятых годов, когда существовало много его толкований, отличных от интерпретации Минского.

Чтобы лучше понять эту теорию, рассмотрим один из примеров Минского, основанный на связи между ожиданием, ощущением и чувством человека, когда он открывает дверь и входит в комнату. Предположим, что вы собираетесь открыть дверь и зайти в комнату незнакомого вам дома. Находясь в доме, перед тем как открыть дверь, у вас имеются определенные представления о том, что вы увидите, войдя в комнату. Например, если вы увидите какой либо пейзаж или морской берег, поначалу вы с трудом узнаете их. Затем вы будете удивлены, и в конце концов дезориентированы, так как вы не сможете объяснить поступившую информацию и связать ее с теми представлениями, которые у вас имелись до того. Также у вас возникнут затруднения с тем, чтобы предсказать дальнейший ход событий. С аналитической точки зрения это можно объяснить как активизацию фрейма комнаты в момент открывания двери и его ведущую роль в интерпретации поступающей информации. Если бы вы увидели за дверью кровать, то фрейм комнаты приобрел бы более узкую форму и превратился бы во фрейм кровати. Другими словами, вы бы имели доступ к наиболее специфичному фрейму из всех доступных.

Термин *фрейм* (от английского слова frame, что означает «каркас» или «рамка») был предложен М.Минскимдля обозначения структуры знаний для восприятия пространственных сцен.

Фрейм – это структура для представления знаний, которая при заполнении ее соответствующими значениями превращается в описание конкретного факта, события или ситуации.

Фрейм является минимально возможным описание сущности какого-либо явления, события, ситуации, процесса или объекта. (Минимально возможное означает, что при дальнейшем упрощении описания теряется его полнота, оно перестает определять ту единицу знаний, для которой оно предназначено.)

В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6-20 м^2». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки» или «слоты» — это незаполненные значения некоторых атрибутов — например, количество окон,.цвет стен, высота потолка, покрытие пола и др. В теории фреймов такой образ комнаты называется *фреймом комнаты*. Фреймом также называется и формализованная модель для отображения образа.

Структура фрейма. Фрейм имеет почти однородную структуру и состоит из стандартных единиц, называемых слотами. Каждая такая единица — слот — содержит название и свое значение.

Фрейм можно представить:

а) в виде цепочки:

Фрейм:<слот 1>, <слот 2>, … ,<слот N>.

или более подробно, как список свойств:

Имя ФРЕЙМА: <имя 1-го слота: значение 1-го слота>, <имя 2-го слота: значение 2-го слота>, <…>, <имя N-го слота: значение N-го слота>.

б) в виде таблицы:

Имя фрейма	
Имя слота 1	Значение слота 1
Имя слота 2	Значение слота 2
Имя слота N	Значение слота N

В качестве примера, рассмотрим фрейм для понятия "Лекция". Ситуация "лекция" может быть определена как "чтение лектором учебного материала слушателям". Фрейм "лекция" может содержать слоты "предмет" (предмет, по которому проводится лекция), "лектор" (ФИО лектора), "аудитория" (место проведения лекции), "слушатели" (количество слушателей).

Лекция	
Предмет	Физика
Лектор	Иванов И.И.
Аудитория	109

Слушатели	Пи41

В данном случае "ЛЕКЦИЯ" - название фрейма; "ПРЕДМЕТ", "ЛЕКТОР", "АУДИТОРИЯ", "СЛУШАТЕЛИ - слоты; "Физика", "Иванов И.И.", "109", "ПИ41" - значения слотов.

Типы фреймов. Различают фреймы-прототипы, хранящиеся в базе знаний, и фреймы-экземпляры, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных.

Фрейм-прототип *(протофрейм) – это фрейм, содержащий знания о самом понятии.*

Например, фрейм понятия «Битва» можно изобразить следующим образом:

Битва	
Субъект (кто?)	X1
Объект (с кем?)	X2
Место (где?)	X3
Время (когда?)	X4
Результат	X5

В этом фрейме указаны имена слотов (субъект, объект и т. д.), но вместо их значений стоят переменные *(X1, X2*и т. д.).

Фрейм-экземпляр *(экзофрейм) - это фрейм, содержащий конкретное описание понятия или знания.*

В примере, наверное, основными для фрейма "Битва" можно считать слоты с именами "субъект" и "объект".Бутву осуществляет субъект *X1* с объектом *X2* в месте *X3* во время *X4,* при этом получается результат *X5.* Подставляя вместо всех переменных конкретные значения, получим конкретный факт-описание:

Куликовская битва	
Субъект (кто?)	Князь Дмитрий
Объект (с кем?)	Хан Мамай
Место (где?)	Куликово поле
Время (когда?)	Утром в сентябре 1380 года
Результат	Победа князя Дмитрия

Исключение из фрейма любого слота делает его принципиально не полным, а иногда вообще бессмысленным.

Модель фрейма является достаточно универсальной, поскольку в состав фрейма могут входить слоты с именами действий, то фреймы годятся для представления как декларативных, так и процедурных знаний.

Фрейм является *простым*, если он не содержит в себе других фреймов. *Сложный (составной)* фрейм содержит в себе два и более фрейма, и по существу представляет сеть фреймов.

Фрейм позволяет отобразить все многообразие знаний о мире через:

- фреймы – *структуры,* использующиеся для обозначения объектов и понятий (заем, залог, вексель);
- фреймы – *роли* (менеджер, кассир, клиент);

- фреймы – *сценарии* (банкротство, собрание акционеров, празднование именин);
- фреймы – *ситуации* (тревога, авария, рабочий режим устройства) и др.

Фреймовая модель представления знаний *– это модель, в которой структура знаний предметной области формализуется в виде совокупности взаимосвязанных фреймов, описывающих объекты, а свойства этих объектов и факты, относящиеся к ним, описываются в структурных элементах фрейма.*

Чтобы представить семантическую сеть в виде совокупности фреймов, надо уметь представлять отношения между вершинами сети. Для этого также используются слоты фреймов. Эти слоты могут иметь имена вида "Связь Y", где Y есть имя того отношения (его тип), которое устанавливает данный *фрейм-вершина* с другим *фреймом-вершиной*.

Заметим также, что в качестве значения слота может выступать новый фрейм, что позволяет на множестве фреймов осуществлять иерархическую классификацию. Это очень удобное свойство фреймов, так как человеческие знания, как правило, упорядочены по общности.

Важнейшим свойством теории фреймов является заимствование из теории семантических сетей — так называемое *наследование свойств.* Во фреймах наследование происходит по ***АКО-связям*** *(A-Kind-Of= это),* которые связывают фреймы с фреймами, находящимися на уровень выше в иерархии, откуда неявно наследуются (переносятся) значения слотов.Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

ПРИМЕР: Понятие «Лектор».

ЧЕЛОВЕК	
АКО	Млекопитающее
УМЕЕТ	Мыслить

ЛЕКТОР	
АКО	Человек
ОБРАЗОВАНИЕ	Высшее
ВОЗРАСТ	23-60

В данном случае представлено одно звено иерархии (ЧЕЛОВЕК-ЛЕКТОР). Здесь фрейм "ЧЕЛОВЕК" является обощающим для фрейма "ЛЕКТОР". Таким образом, фрейм "ЛЕКТОР" наследует от фрейма "ЧЕЛОВЕК" значение слота "УМЕЕТ" (а также других слотов, не показанных в примере). Цепочка наследования может быть продолжена вплоть до, например, фрейма "ЖИВОЕ СУЩЕСТВО".

Кроме того связь фреймов осуществляется по **значению слота**. Фрагмент сети фреймов представлен на рис.1.4. Такая структура позволяет систематизировать большой объем информации, оставляя ее при этом максимально удобной для использования. Кроме того, система (сеть) фреймов способна отражать концептуальную основу организации памяти человека.

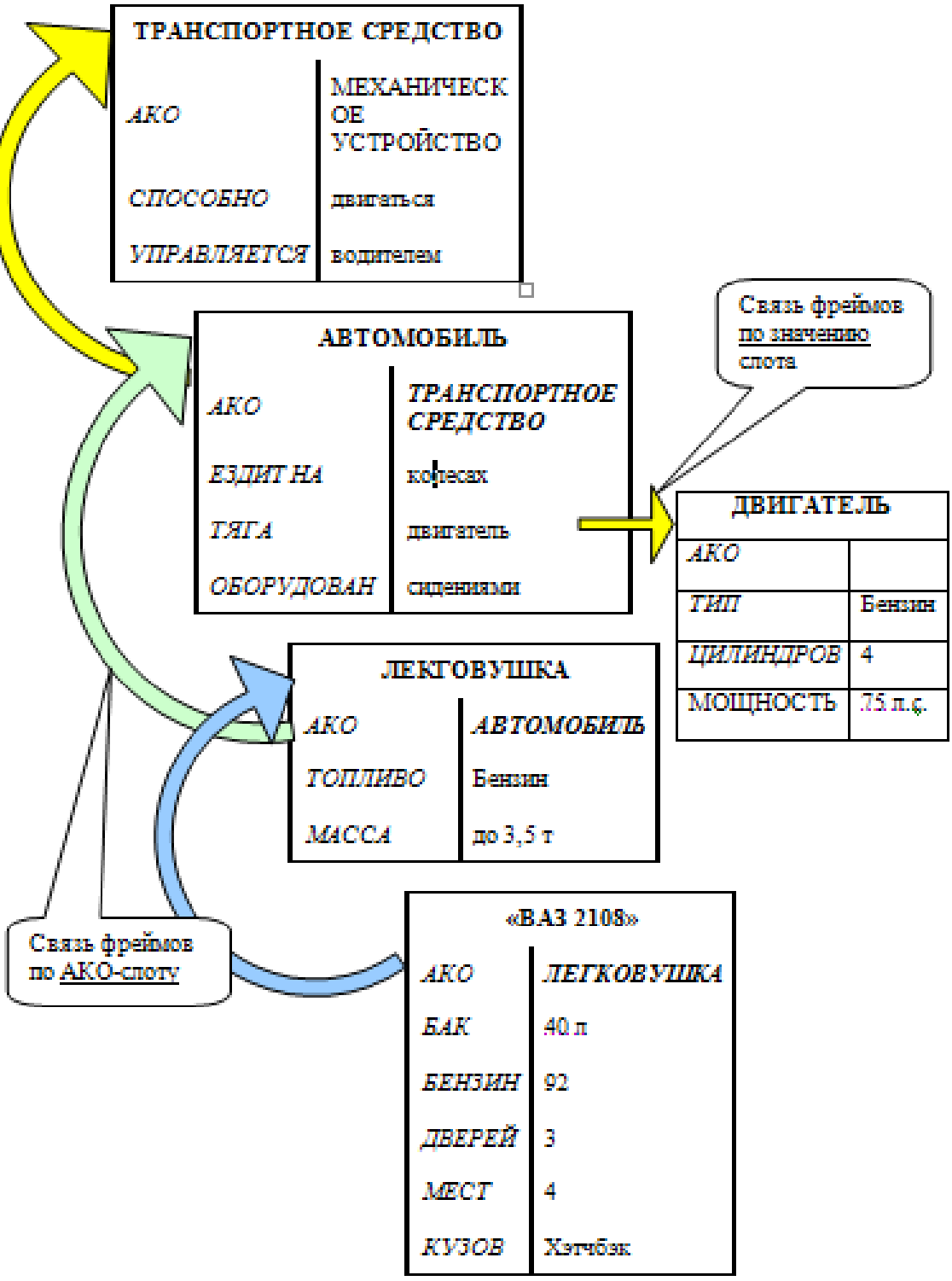


Рис.1.4 Фрагмент сети фреймов.

Глава 3. ЭКСПЕРТНЫЕ СИСТЕМЫ

3.1. Назначение и структура экспертных систем

Назначение экспертных систем.

В начале восьмидесятых годов в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название "экспертные системы" (ЭС). Цель исследований по ЭС состоит в разработке программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Исследователи в области ЭС для названия своей дисциплины часто используют также термин "инженерия знаний", введенный Е.Фейгенбаумом как "привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов".

Важность экспертных систем состоит в следующем:

- технология экспертных систем существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект;
- технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования: длительность и, следовательно, высокая стоимость разработки сложных приложений;
- высокая стоимость сопровождения сложных систем, которая часто в несколько раз превосходит стоимость

их разработки; низкий уровень повторной используемости программ и т.п.;

- объединение технологии ЭС с технологией традиционного программирования добавляет новые качества
- программным продуктам за счет: обеспечения динамичной модификации приложений пользователем, а не программистом; большей "прозрачности" приложения (например, знания хранятся на ограниченном ЕЯ, что не требует комментариев к знаниям, упрощает обучение и сопровождение); лучшей графики; интерфейса и взаимодействия.

По мнению ведущих специалистов, ЭС найдут следующее применение:

- ЭС будут играть ведущую роль во всех фазах проектирования, разработки, производства, распределения, продажи, поддержки и оказания услуг;
- технология ЭС обеспечит революционный прорыв в интеграции приложений из готовых интеллектуально-взаимодействующих модулей.
- решение, так называемых, неформализованных и слабоструктурированных задач.

Отличительные особенности ЭС. Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных следующими основными чертами:

- в них в основном используются символьный (а не числовой) способ представления;
- используется символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма);

- ЭС применяются для решения только трудных практических задач, для решения которых нужны экспертные знания;
- ЭС дает пользователю «готовое» решение, которое по качеству и эффективности не уступает решению эксперта-человека;
- решения экспертных систем обладают *"прозрачностью",* т.е. могут быть объяснены пользователю на качественном уровне. Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях;
- экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом, а также в процессе самообучения (т.н. машинное обучение);
- применение специфического компонента – базы знаний.

Как уже отмечалось, ЭС предназначены, главным образом, для решения практических задач, возникающих в слабо структурированной и трудно формализуемой предметной области. ЭС были первыми системами, которые привлекли внимание потенциальных потребителей продукции искусственного интеллекта.

Однако сэкспертными системами связаны некоторые распространенные заблуждения.Заблуждение первое: ЭС будут делать не более (а скорее даже менее) того, чем может эксперт, создавший данную систему. Для опровержения данного постулата можно построить самообучающуюся ЭС в области, в которой вообще нет экспертов, либо объединить в одной ЭС знания нескольких экспертов, и получить в результате систему, которая может то, чего ни один из ее создателей не может.Заблуждение второе: ЭС никогда не заменит человека-эксперта.

Предметные области для экспертных систем.

Предметная область – это совокупность реальных или абстрактных объектов, связей и отношений между этими объектами, а также процедур преобразования этих объектов для решения возникающих задач.

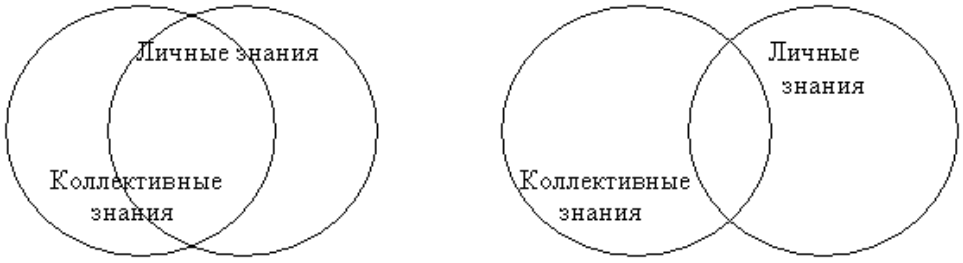
В нашей стране современное состояние разработок в области экспертных систем можно охарактеризовать как стадию всевозрастающего интереса среди широких слоев экономистов, финансистов, преподавателей, инженеров, медиков, психологов, программистов, лингвистов. К сожалению, этот интерес имеет пока достаточно слабое материальное подкрепление - явная нехватка учебников и специальной литературы, отсутствие символьных процессоров и рабочих станций искусственного интеллекта, ограниченное финансирование исследований в этой области, слабый отечественный рынок программных продуктов для разработки экспертных систем.

Поэтому распространяются "подделки" под экспертные системы в виде многочисленных диалоговых систем и интерактивных пакетов прикладных программ, которые дискредитируют в глазах пользователей это чрезвычайно перспективное направление. Процесс создания экспертной системы требует участия высококвалифицированных специалистов в области искусственного интеллекта, которых пока выпускает небольшое количество высших учебных заведений страны.

Традиционно знания существуют в двух видах - коллективныезнания, которыми обладают большинство людей, и личныезнания, которыми обладают специалисты (эксперты).

Если большая часть знаний в предметной области представлена в виде коллективного опыта (например, высшая математика), эта предметная область не нуждается в экспертных системах (рис.3.1а).

Если в предметной области большая часть знаний является личным опытом специалистов высокого уровня (экспертов), если эти знания по каким-либо причинам слабо структурированы, такая предметная область скорее всего нуждается в экспертной системе (рис. 3.1б).



а) б)

Рис.3.1. Соотношение коллективных и личных знаний в предметной области.

Необходимо отметить, что в настоящее время технология экспертных систем используется для решения различных типов задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление) в самых разнообразных проблемных областях, таких, как финансы, нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, промышленность, телекоммуникации и связь и др.Однако степень охвата

решаемых задач существенно отличается для различных предметных областей.

На первый взгляд кажется, что создание единой ЭС (или ИИС), охватывающей все предметные области, возможно. Это - глубокое заблуждение. Прежде всего, следует учитывать, что для решения всех возможных задач во всех предметных областях необходимо бесконечное число фактов и правил. Даже если бы такая система была создана, понадобилось бы длительное время на наполнение ее знаниями. Более того, сегодня еще нет вычислительной машины, способной хранить и обрабатывать такой объем информации, поэтому пока нужно ограничиться только проблемными областями, в которых объём информации не слишком велик, и ее можно обработать в программе.

Таким образом, применение ЭС реально только для узких предметных областей, а если быть более точным, то для решения конкретной задачи в предметной области.

Классификация экспертных систем

Класс "экспертные системы" сегодня объединяет несколько тысяч различных программных комплексов, которые можно классифицировать по различным признакам (рис.3.2).

Рис.3.2. Классификация экспертных систем.

По решаемой задаче

Интерпретация данных. Это одна из традиционных задач для экспертных систем. Под интерпретацией понимается определение смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных. Например, определение основных свойств личности по результатам психодиагностического тестирования в системах АВТАНТЕСТ и МИКРОЛЮШЕР и др.

Диагностика. Под диагностикой понимается обнаружение неисправности в некоторой системе. Неисправность - это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Важной спецификой является необходимость понимания функциональной структуры ("анатомии")

диагностирующей системы. Например, диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ.

Мониторинг. Основная задача мониторинга - непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы - "пропуск" тревожной ситуации и инверсная задача "ложного" срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимость учета временного контекста. Примером может служить система контроля аварийных датчиков на химическом заводе - FALCON и др.

Проектирование. Проектирование состоит в подготовке спецификаций на создание "объектов" с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов чертеж, пояснительная записка и т.д. Основные проблемы здесь - получение четкого структурного описания знаний об объекте и проблема "следа". Для организации эффективного проектирования и, в еще большей степени, перепроектирования необходимо формировать не только сами проектные решения, но и мотивы их принятия. Таким образом, в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ЭС*:* процесс вывода решения и процесс объяснения.

Прогнозирование. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров "подгоняются" под заданную ситуацию. Выводимые из этой модели следствия составляют основу для

прогнозов с вероятностными оценками. Например, предсказание погоды.

Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. Втаких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

Обучение. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом "ученике" и его характерных ошибках, затем в работе способны диагностировать слабости в знаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

По связи с реальным временем

- *Статические ЭС* разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны. Например, диагностика неисправностей в автомобиле.
- *Квазидинамические ЭС*интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени.Например, микробиологические ЭС, в которых снимаются лабораторные измерения с технологического процесса один раз в 4 - 5 (производство лизина, например) и анализируется динамика полученных показателей по отношению к предыдущему измерению.

- *Динамические ЭС*работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступаемых данных. Например, управление гибкими производственными комплексами, мониторинга в реанимационных палатах и т.д.

По типу ЭВМ

На сегодняшний день существуют:

- ЭС для уникальных стратегически важных задач на суперЭВМ (Эльбрус, CONVEX и др.);
- ЭС на ЭВМ средней производительности (типа ЕС ЭВМ, mainframe);
- ЭС на символьных процессорах и рабочих станциях (SUN, APOLLO);
- ЭС на мини- и супермини-ЭВМ (VAX, micro-VAX и др.);
- ЭС на персональных компьютерах (IBM PC, MAC II и подобные).

По степени интеграции с другими программами

- *Автономные ЭС* работают непосредственно в режиме консультаций с пользователем для специфически "экспертных" задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.).
- *Гибридные ЭС* представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может быть интеллектуальная надстройка над ППП или интегрированная среда для решения сложной

задачи с элементами экспертных знаний.Несмотря на внешнюю привлекательность гибридного подхода, следует отметить, что разработка таких систем являет собой задачу, на порядок более сложную, чем разработка автономной ЭС. Стыковка не просто разных пакетов, а разных методологий (что происходит в гибридных системах) порождает целый комплекс теоретических и практических трудностей.

Обобщенная структура экспертной системы.

Необходимо отметить, что в различных источниках встречаются разные определения экспертной системы. Ниже предлагается наиболее чаще встречающееся определение.

Экспертная система *(ЭС) - это сложный программный комплекс, аккумулирующий знания специалистов в конкретных предметных областях н тиражирующий этот эмпирический опыт для консультаций менее квалифицированных пользователей.*

Обобщенная структура экспертной системы представлена на рис.3.2.

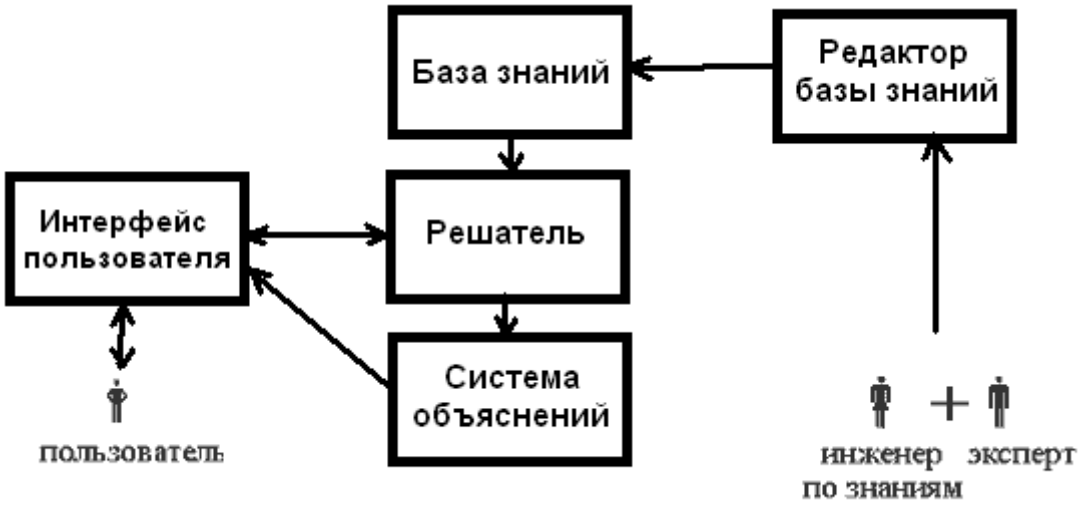


Рис. 16.5 Обобщенная структура экспертной системы.

Следует учесть, что реальные экспертные системы могут иметь более сложную структуру, однако блоки, изображенные на рисунке, непременно присутствуют в любой действительно экспертной системе, поскольку являют собой негласный канон на структуру современной экспертной системы.

Состав и назначение элементов ЭС.

Интерфейс пользователя - комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, получения результатов и «объяснения» решения.

База знаний(БЗ) - ядро ЭС, совокупность формализованных знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и инженеру по знаниям. Параллельно такому представлению существует БЗ во внутреннем "машинном" представлении-

Решатель(*Машина вывода, Интерпретатор)* - программа, моделирующая ход рассуждений эксперта на основании формализованных знаний, имеющихся в БЗ и исходных данных (фактах), получаемых от пользователя.

Подсистема объяснений - программа, протоколирующая работу решателя в виде «цепочки логических выводов». Она позволяет пользователю получить ответы на вопросы; "Как была получена та или иная рекомендация?" и "Почему система приняла такое решение?" Ответ на вопрос "как" - это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, т.е. всех шагов цепи умозаключений. Ответ на вопрос "почему"- ссылка на умозаключение, непосредственно

предшествовавшее полученному решению, т.е. отход на один шаг назад.

Редактор БЗ- программа, представляющая инженеру по знаниям возможность создавать и пополнять БЗ в диалоговом режиме. Осуществляет ввод формализованных знаний. Например, правил продукционной модели представления знаний. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок ("help" - режим) и других сервисных средств, облегчающих работу с базой знаний.

В разработке и эксплуатации ЭС участвуют как минимум четыре человека:

- эксперт;
- инженер по знаниям;
- программист (на схеме не показан);
- пользователь.

Возглавляет коллектив инженер по знаниям, это ключевая фигура при разработке систем, основанных на знаниях.Синонимы: когнитолог, инженер-интерпретатор, аналитик.

Статическая и динамическая ЭС.

Статическая экспертная системаиспользуются в тех случаях, когда можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Структурная схема такой ЭС представлена на рис. 3.3. Первые ЭС, получившие практическое использование, были статическими.

Экспертная система работает в двух режимах:

- в режиме приобретения знаний;

- в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

В режиме приобретения знаний общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний (редактор БЗ), наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт, не владеющий программированием.

В режиме консультации общение с ЭС осуществляет пользователь, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, проконсультироваться, либо возложить на ЭС рутинную работу). В режиме консультации исходные данные (факты) о

задаче от пользователя поступают через интерфейс в решатель. Решатель на основе входных данных, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения, которое выдается из системы объяснений.

Динамическая экспертная системаиспользуются в тех случаях, когда необходимо учитывать изменения окружающего мира, происходящие за время решения задачи. На рис.3.4 показана архитектура динамической экспертной системы. По сравнению со статической ЭС вводятся дополнительных два компонента:

- система сопряжения;
- источники внешних динамических данных (датчики).

Система сопряженияосуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической ЭС (база знаний и машина вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

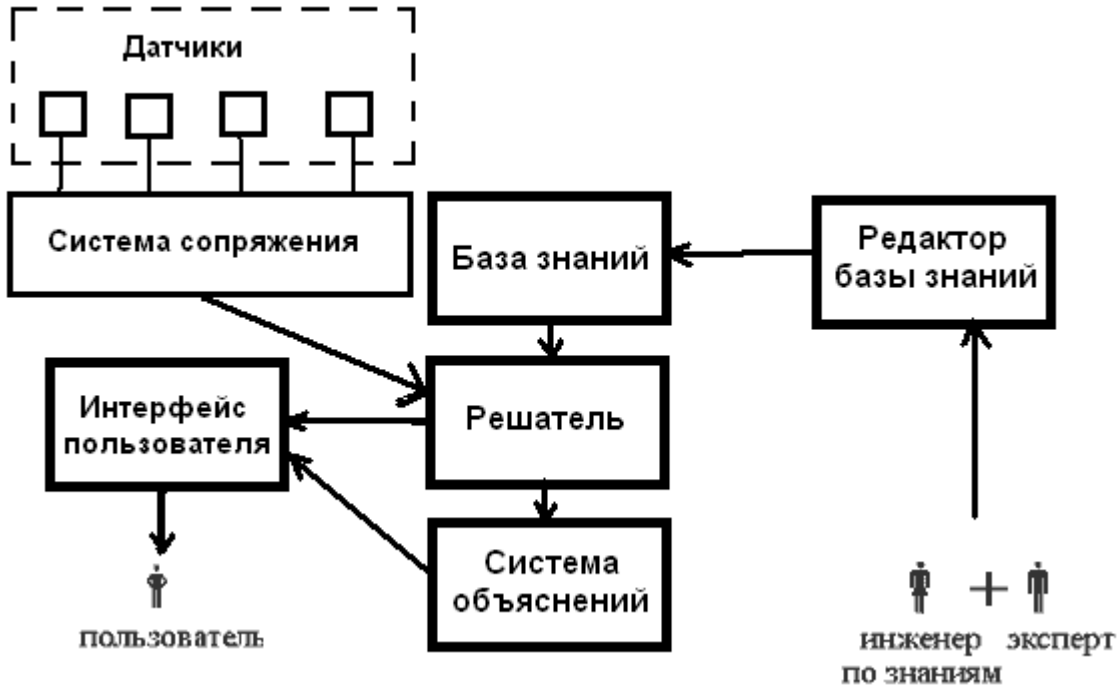


Рис.3.4 Динамическая экспертная система.

Режимы работы динамической ЭС в основном аналогичны вышеописанным режимам. Отличие в том, что пользователь в режиме консультации не вводит исходные данные, лишь отслеживает результат работы. Исходные данные непрерывно поступают с датчиков. Такие системы используют, например, для контроля технологических процессов, контроля работы опасных систем (атомных станций).

Подчеркнем, что структура статической и динамической ЭС, отражает только основные компоненты (функции), и многое остается "за кадром". Кроме основных компонентов существуют дополнительные, которые позволяют создавать интегрированные системы в соответствии с современной технологией использования ЭС.

3.2. Машина вывода экспертных систем

Понятие и работа машины вывода.

Несмотря на все недостатки, наибольшее распространение получила продукционная модель представления знаний. При использовании продукционной модели база знаний состоит из набора правил.

Машина вывода *(интерпретатор) – это программа, управляющая перебором правил в базе знаний.*

Машина вывода(интерпретатор правил) выполняет две функции:

- во-первых, просмотр существующих фактов из рабочей памяти и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов и,
- во-вторых, определение порядка просмотра и применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных.

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой небольшую по объему программу и включает два компонента, реализующих вышеприведенные функции.

Действие *компонента вывода*основано на применении правила, называемого *modusponens:*.

«Если известно, что истинно утверждение A и существует правило вида «ЕСЛИ A, ТО B», тогда утверждение B также истинно».

Правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации.

*Управляющий компонент*определяет порядок применения правил и выполняет четыре функции.

Сопоставление —правила сопоставляются с имеющимися фактами, полученными от пользователя или в результате срабатывания правила..

Выбор — если в конкретной ситуации может быть применено сразу несколькоправил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

Срабатывание — выполнение части «ТО» выбранного правила.

Действие — рабочая память подвергается изменению путем добавления в нее заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие (воздействие), то оно выполняется. Воздействие может понадобиться, например, в системах обеспечения безопасности информации – отключение линии связи, запрет удаления и т.п.

Работа машины вывода (интерпретатора).

Машина вывода (Интерпретатор продукций) работает циклически. Цикл работы интерпретатора схематически представлен на рис.3.5.

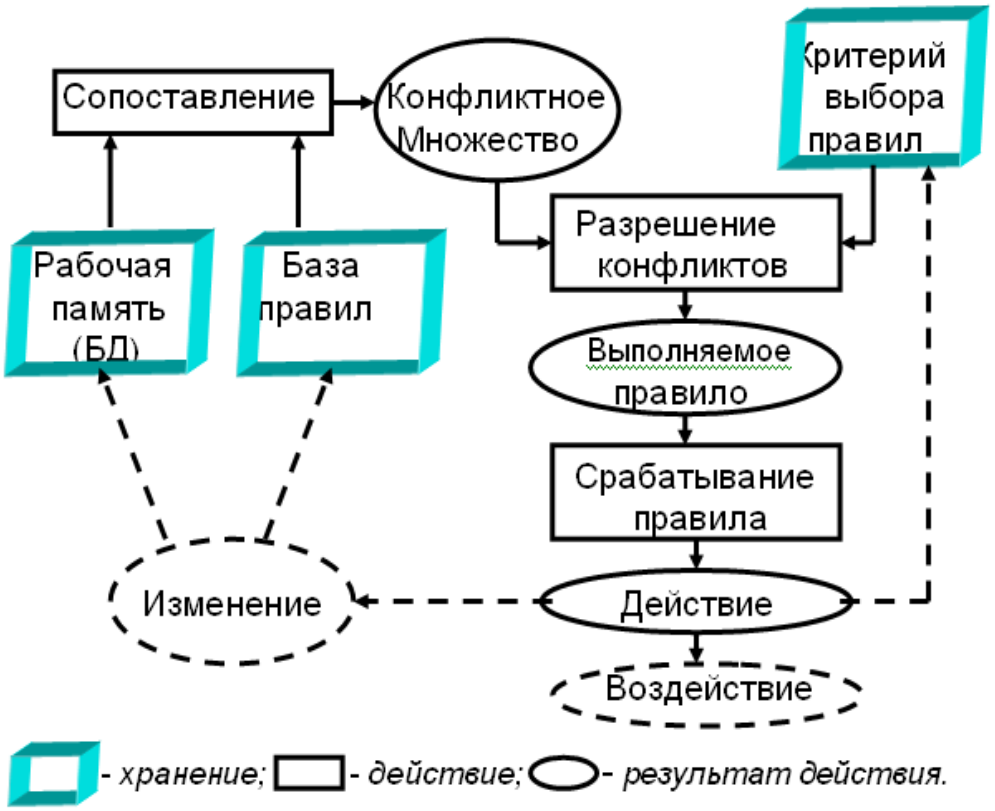


Рис.3.5. Цикл работы машины вывода.

Информация из рабочей памяти (базы данных) последовательно *сопоставляется* с условными частямиправил из базы правил для выявления успешного сопоставления. Совокупность успешно отобранных правил составляет так называемое конфликтное множество. Для *разрешения конфликта* интерпретатор имеет критерий, с помощью которого он выбирает единственное выполняемое правило.

После чего оно *срабатывает* и выполняется *действие*. Действие может выражаться:

- в занесении нового факта (заключения сработавшего правила) в рабочую память;
- в изменении критерия выбора конфликтующих правил,
- воздействии на что- либо (например, отключение аварийного блока).

Затем цикл повторяется снова, до тех пор пока не перестанет образовываться конфликтное множество. Заключение последнего сработавшего правила выдается пользователю в качестве результата решения задачи.

Работа машины вывода зависит только от состояния рабочей памяти и от состава базы знаний.

Таким образом, в каждом цикле просматриваются все правила, но одном цикле может сработать только одно правило

Стратегии управления выводом.

От выбранного метода поиска, то есть стратегии вывода, будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно «зашиты» в механизм вывода, поэтому в большинстве систем инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию.

Вывод - Получение новых информационных единиц из ранее известных. Частным случаем является логический вывод.

При разработке стратегии управления выводом важно определить два вопроса:

Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска — в прямом или обратном направлении.

Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора — глубину, в ширину, по подзадачам или иначе (например, Альфа-бета алгоритм).

Прямой и обратный выводы.

Обратный вывод - вывод, при котором поиск доказательства начинается с целевого утверждения. При обратном порядке вывода вначале выдвигается некоторая гипотеза (выясняются условия), которая доказывает утверждение. Затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу (рис.3.6, правая часть). Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность этой (подчиненной) гипотезы и т.д., пока очередная гипотеза не приведет (укажет) к факту (цели).

Вывод такого типа называется управляемым целями, или управляемым консеквентами. Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

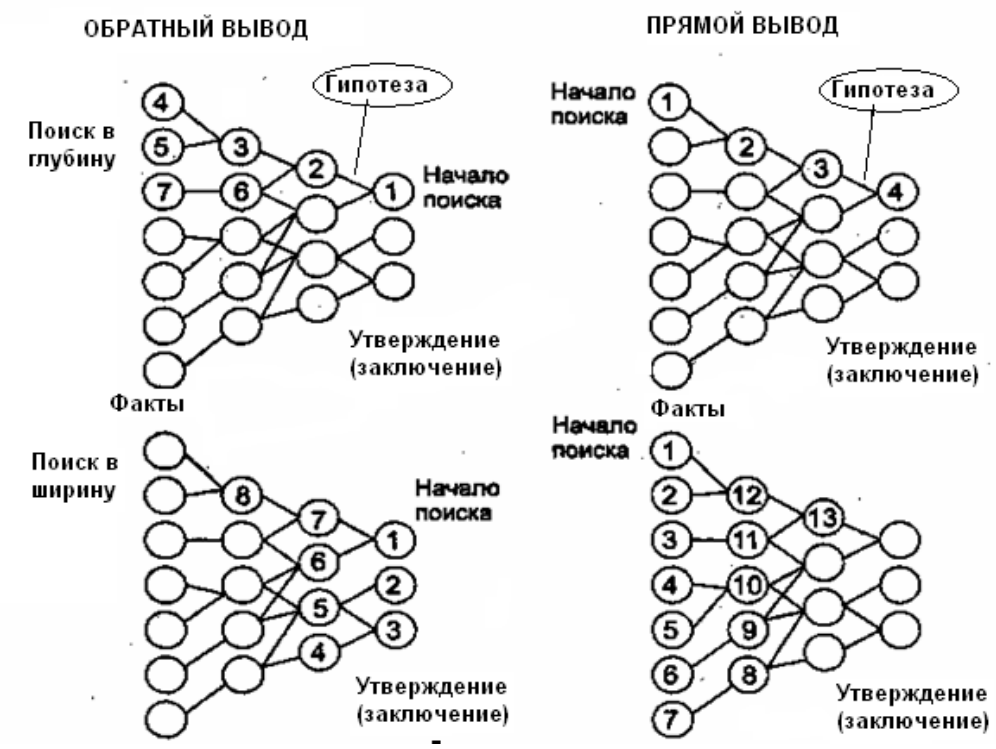


Рис. 3.6. Стратегии вывода

Прямой вывод. В системах с прямым выводом по известным фактам отыскивается утверждение (заключение), которое из этих фактов следует (рис.3.6, левая часть). Если такое заключение удается найти, то оно заносится в рабочую память.

Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами.

Рассмотрим пример**.**Имеется фрагмент базы знаний из двух правил:

Правило 1: Если «отдых — летом» и «человек — активный», то «ехать в горы».

Правило 2: Если «любит солнце», то «отдых летом».

Предположим, в систему ввели факты— «человек активный» и «любит солнце».

Обратный вывод — подтвердить выбранную цель (в данном случае - «ехать в горы») при помощи имеющихся правил и данных.

1-й цикл

Шаг 1.1. Утверждение (Цель) — «ехать в горы»: Проверяем правило *П1*— не хватает данных (факта) «отдых — летом». Поэтому эти данные (факт) («отдых — летом») становятся новым утверждением (целью) и ищется правило, где такое утверждение (цель) есть в левой части.

Шаг 1.2. Утверждение (Цель) «отдых — летом»: Проверяем правило *П2.* Это правило срабатывает и в системе появляется факт - «любит солнце».

2-й цикл.

Шаг 2.1. Проверяем правило*П1*: наличие обоих фактов («отдых — летом» и «человек — активный») подтверждают выбранную цель («ехать в горы»).

Прямой вывод— нужно исходя из фактических данных, получить рекомендацию.

1-й цикл.

Шаг 1.1. Проверяем правило *П1:* не работает, т.к. не хватает данных «отдых — летом».

Шаг 1.2. Проверяем правило *П2:* работает, т.к. есть факт «любит солнце». Следовательно, образуется новый факт «отдых — летом».

2-й цикл.

Шаг 2.1. Проверяем правило *П1,* работает, т.к. присутствуют оба факта - «отдых — летом» и «человек — активный». Следовательно, становится истинным

утверждение (цель) «ехать в горы», которая и выступает как совет, который дает ЭС.

Существуют системы, в которых вывод основывается на сочетании упомянутых выше методов — обратного и ограниченного прямого. Такой комбинированный метод получил название циклического.

3.3. Этапы разработки экспертных систем

В настоящее время сложилась определенная технология разработки ЭС, которая включает следующие шесть этапов (рис.3.7):

- идентификация,
- концептуализация,
- формализация,
- выполнение,
- тестирование
- опытная эксплуатация.

На этапе ***идентификации*** определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.Идентификация задачи заключается в составлении неформального (вербального) описания, в котором указываются: общие характеристики задачи; подзадачи, выделяемые внутри данной задачи; ключевые понятия (объекты), их входные (выходные) данные; предположительный вид решения, а также знания, относящиеся к решаемой задаче.

Результатом данного этапа является ответ на вопрос, что надо сделать и какие ресурсы необходимо задействовать (идентификация задачи, определение участников процесса проектирования и их роли, выявление ресурсов и целей).На выходе этапа – требования к экспертной системе.

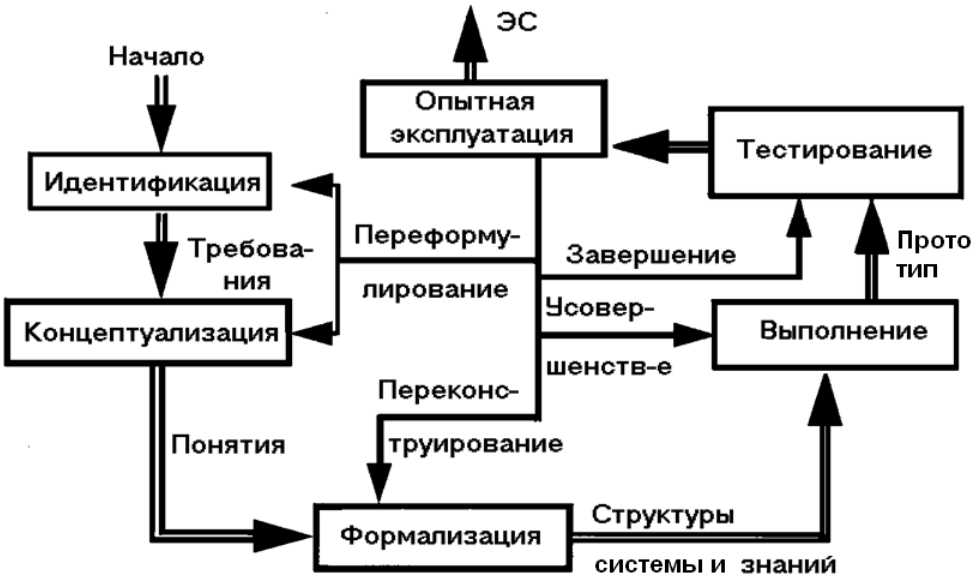


Рис.3.7 Этапы разработки ЭС.

На этапе ***концептуализации*** проводится содержательный анализ проблемной области и решаемой задачи, выявляются используемые понятия и их взаимосвязи, определяются методы решения задачи.

Этот этап завершается созданием модели предметной области, включающей основные концепты и отношения. На этапе концептуализации определяются следующие особенности задачи: типы доступных данных; исходные и выводимые данные, подзадачи общей задачи; используемые стратегии и гипотезы; виды взаимосвязей между объектами, типы используемых отношений (иерархия, причина — следствие, часть — целое и т.п.); процессы, используемые в

ходе решения; состав знаний, используемых при решении задачи; типы ограничений, накладываемых на процессы, используемые в ходе решения; состав знаний, используемых для обоснования решений.На выходе - понятия и их взаимосвязи.

На этапе ***формализации*** выбираются инструментальные средства разработки, определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, разрабатывается программная оболочка (прототип), моделируется работа системы, оценивается адекватность системы.

Выходом этапа формализации является описание того, как рассматриваемая задача может быть представлена в выбранном или разработанном формализме. Сюда относится указание способов представления знаний (фреймы, сценарии, семантические сети и т.д.) и определение способов манипулирования этими знаниями (логический вывод, аналитическая модель, статистическая модель и др.) и интерпретации знаний.В конце этапа имеем структуру системы.

На этапе ***выполнения*** осуществляется создание одного или нескольких прототипов ЭС, решающих требуемые задачи,наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, формализацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа

деятельности эксперта по решению реальных задач. На выходе этапа – программная реализация всех структурных элементов экспертной системы (рис.3.2).

Этап ***тестирования.*** В ходе данного этапа производится оценка выбранного способа представления знаний в ЭС в целом. Для этого инженер по знаниям совместно с экспертом подбирает тестовые примеры, обеспечивающие проверку возможностей разработанной ЭС. При подготовке тестовых примеров следует классифицировать их по подпроблемам предметной области, выделяя стандартные случаи, определяя границы трудных ситуаций и т.п. Степень соответствия решения тестового примера полученного экспертной системой и известного решения характеризует качество разработки системы. Главным в оценке работы системы является полнота и безошибочность работы правил вывода.

На этом этапе ***опытной эксплуатации*** проверяется пригодность ЭС для конечного пользователя. Пригодность ЭС для пользователя определяется в основном удобством работы с ней и ее полезностью. Под полезностью ЭС понимается ее способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неудач в работе, а также удовлетворять указанные потребности пользователя (решать поставленные задачи). В свою очередь, удобство работы с ЭС подразумевает естественность взаимодействия с ней (общение в привычном, не утомляющем пользователя виде), гибкость ЭС (способность системы настраиваться на различных пользователей, а также учитывать изменения в квалификации одного и того же пользователя) и устойчивость системы к ошибкам (способность не выходить из строя при ошибочных действиях неопытного пользователях).

В ходе разработки ЭС почти всегда осуществляется ее модификация. Выделяют следующие виды модификации системы: переформулирование понятий и требований, переконструирование представления знаний в системе и усовершенствование прототипа.

3.4. Коллектив разработчиков ЭС.

Под коллективом разработчиков понимается группа специалистов, ответственных за создание ЭС.В разработке ЭС участвуют представители следующих специальностей.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний. Руководит коллективом.

Программист(ы)осуществляет программную реализацию ЭС, сопрягает все основные компоненты ЭС.

Пользовательосуществляет проверку удобства работы с ЭС на заключительных этапах разработки. Конечно же его нельзя отнести к специалистам, однако его роль достаточно важна.

Таким образом, минимальный состав коллектива включает четыре человека; реально же он разрастается до 8-10 человек. Численное увеличение коллектива разработчиков

происходит по следующим причинам: необходимость учета мнения нескольких пользователей, помощи нескольких экспертов; потребность как в проблемных, так и системных программистах. На Западе в этот коллектив дополнительно традиционно включают менеджера и одного технического помощника.При отсутствии профессионального менеджера руководителем, участвующим во всех стадиях разработки, является инженер по знаниям, поэтому к его квалификации предъявляются самые высокие требования. В целом уровень и численность группы зависят от характеристик поставленной задачи.

Для обеспечения эффективности работы любой творческой группы, в том числе и группы разработчиков ЭС, необходимо возникновение атмосферы взаимопонимания и доверия, которое, в свою очередь, обусловлено психологической совместимостью членов группы; следовательно, при формировании группы должны учитываться психологические свойства участников.

Приведем два аспекта характеристик членов коллектива разработчиков ЭС: 1 — психофизиологический, 2 — профессиональный.

Эксперт.

1.*Психологические качества*. Эксперт — чрезвычайно важная фигура в группе. В конечном счете, его подготовка определяет уровень компетенции базы знаний. Желательные качества:

а) доброжелательность;

б) готовность поделиться своим опытом;

в) умение объяснить (педагогические навыки);

г) заинтересованность (моральная, а лучше еще и материальная) в успешности разработки.

Возраст эксперта обычно почтенный, что необходимо учитывать всем членам группы. Часто встает вопрос о количестве экспертов. Поскольку проблема совмещения подчас противоречивых знаний остается открытой, обычно с каждым из экспертов работают индивидуально, иногда создавая альтернативные базы.

2. *Профессиональные качества*. Помимо безусловно высокого профессионализма в выбранной предметной области, желательно знакомство эксперта с популярной литературой по искусственному интеллекту и экспертным системам для того, чтобы эффективнее прошел этап извлечения знаний.

Инженер по знаниям.

1. *Психологические качества*. Существуют такие профессии и виды деятельности, для которых природные качества личности (направленность, способности, темперамент) могут иметь характер абсолютного показания или противопоказания к занятиям. По-видимому, инженерия знаний принадлежит к таким профессиям. По различным оценкам, это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей. Попытаемся дать наброски к портрету инженера по знаниям (без претензии на полноту и точность определений).

Пол. Психологи утверждают, что мужчины более склонны к широкому охвату явлений и в среднем у них выше аналитичность, чрезвычайно полезная инженеру по знаниям, которому надо иметь развитое логическое мышление и умение оперировать сложными формальными структурами.

Кроме того, при общении с экспертами, которые в большинстве своем настроены скептически по отношению к будущей ЭС, инженер по знаниям – мужчина вызывает более высокую мотивацию успешности со стороны эксперта-женщины. С другой стороны, известно, что у женщин выше наблюдательность к отдельным деталям объектов. Так что пол не является окончательным показанием или противопоказанием к данной профессии.

Интеллект. Это понятие вызывает самые бурные споры психологов; существует до 50 определений интеллекта, но с прагматической точки зрения очевидно, что специалист в области искусственного интеллекта должен стремиться к максимальным оценкам по тестам как вербального, так и невербального интеллекта.

Стиль общения. Инженер по знаниям «задает тон» в общении с экспертом, он ведет диалог, и от него в конечном счете зависит его продуктивность. Можно выделить два стиля общения: деловой (или жесткий) и дружеский (или мягкий, деликатный). Нам кажется, что дружеский будет заведомо более успешным, так как снижает «эффект фасада» у эксперта, раскрепощает его. Деликатность, внимательность, интеллигентность, ненавязчивость, скромность, умение слушать и задавать вопросы, хорошая коммуникабельность и в то же время уверенность в себе — вот рекомендуемый стиль общения. Безусловно, что это дар и искусство одновременно, однако занятия по психологическому тренингу могут дать полезные навыки.

Портрет инженера по знаниям можно было бы дополнить другими характеристиками — широтой взглядов и интересов, артистичностью, чувством юмора, обаянием и т. д.

2. *Профессиональные качества.*При определении профессиональных требований к аналитику следует учитывать, что ему необходимы различные навыки и умения для грамотного и эффективного проведения процессов извлечения, концептуализации и формализации знаний.

Инженер по знаниям имеет дело со всеми формами знаний Z1 (знания в памяти) $\rightarrow$Z2 (знания в книгах) $\rightarrow$Z3 (поле знаний) $\rightarrow$Z4 (модель знаний) $\rightarrow$Z5 (база знаний).

Работа на уровне Z1 требует от инженера по знаниям знакомства с элементами когнитивной психологии и способами репрезентации понятий и процессов в памяти человека, с двумя основными механизмами мышления – логическим и ассоциативным, с такими способами активизации мышления как игры, мозговой штурм и др., с различными моделями рассуждений.

Изучение и анализ текстов на уровне Z2 подразумевает широкую общенаучную подготовку инженера; знакомство с методами реферирования и аннотирования текстов; владение навыками быстрого чтения, а также текстологическими методами извлечения знаний.

Разработка поля знаний на уровне Z3 требует квалифицированного знакомства с методологией представления знаний, системным анализом, теорией познания, аппаратом многомерного шкалирования, кластерным и факторным анализом.

Разработка формализованного описания Z4 предусматривает предварительное изучение аппарата математической логики и современных языков представления знаний. Модель знаний разрабатывается на основании результатов глубокого анализа инструментальных средств

разработки ЭС и имеющихся «оболочек». Кроме того, инженеру по знаниям необходимо владеть методологией разработки ЭС, включая методы быстрого прототипирования.

И наконец, реализация базы знаний Z5, в которой инженер по знаниям участвует вместе с программистом, подразумевает овладение практическими навыками работы на ЭВМ и, возможно, одним из языков программирования. Так как инженеров по знаниям «выращивают» из программистов, уровень Z5 обычно не вызывает затруднения, особенно если разработка ведется на традиционных языках типа С или Паскаль. Специализированные языки искусственного интеллекта Лисп и Пролог требуют некоторой перестройки архаично–алгоритмического мышления.

Программист.

1. *Психологические качества.*Известно, что программисты обладают самой низкой потребностью в общениисреди представителей разных профессий. Однако при разработке ЭС необходим тесный контакт членов группы, поэтому желательны следующие его качества:

а) общительность;

б) способность отказаться от традиционных навыков и освоить новые методы;

в) интерес к разработке.

2.*Профессиональные качества*.Поскольку современные ЭС — сложнейшие и дорогостоящие программные комплексы, программисты должны иметь опыт и навыки разработки программ. Обязательно знакомство с основными структурами представления знаний и механизмами вывода,

состоянием отечественного и мирового рынка программных продуктов для разработки ЭС и диалоговых интерфейсов.

Пользователь.

1. *Психологические качества.* К пользователю предъявляются самые слабые требования, поскольку его не выбирают. Он является в некотором роде заказчиком системы. Желательные качества:

а) дружелюбие;

б) умение объяснить, что же он хочет от системы;

в) отсутствие психологического барьера к применению вычислительной техники;

г) интерес к новому.

От пользователя зависит, будет ли применяться разработанная ЭС. Замечено, что наиболее ярко качества *в*) и *г*) проявляются в молодом возрасте, поэтому иногда такие пользователи охотнее применяют ЭС, не испытывая при этом комплекса неполноценности оттого, что ЭВМ им что-то подсказывает.

2. *Профессиональные качества*. Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС. Кроме того, должна быть полная совместимость в терминологии интерфейса к ЭС с той, которая привычна и удобна для пользователя. Обычно требования к квалификации пользователя не очень велики, иначе он переходит в разряд экспертов и совершенно не нуждается в ЭС.

Таким образом, успешность выбора и подготовки коллектива разработчиков ЭС определяет эффективность и продолжительность всего процесса разработки.

3.5. Основы разработки экспертной системы

Постановка задачи.

Прежде всего, поставим задачу, для решения которой будет разрабатываться экспертная система. Подходящей задачей, при решении которой можно использовать обратную цепочку рассуждений, может быть задача, вытекающая из следующей ситуации: к директору крупной технической фирмы пришел человек, желающий устроиться на работу. Директор располагает сведениями о потребностях фирмы в специалистах и общем положении дел в фирме. Ему нужно решить, какую должность в фирме может занять посетитель. Для этого необходимо задать посетителю такие вопросы, ответы на которые дадут возможность сделать правильный выбор должности.

На первый взгляд задача не очень сложная, но на решение директора влияет много факторов. Допустим, претендент работает в данной области недавно, но уже сделал важное открытие или он закончил учебное заведение с посредственными оценками, но несколько лет работал по специальности. В данной ситуации люди ведут себя по-разному, и хотя для того, чтобы получить работу необходимо, удовлетворять определенным критериям, в биографии претендента могут быть самые различные факты, анализ которых поможет подобрать для него соответствующую должность.Поскольку в задаче надо выбрать один из нескольких возможных вариантов

(должностей), для её решения можно воспользоваться обратной цепочкой рассуждений.

Таким образом, необходимо разработать экспертную систему, которая определит подходящую должность посетителю. Экспертная система будет содержать экспертные знания директора и заменит его при решении описанной задачи. Такой системой может пользоваться как сам посетитель, так и менее квалифицированный (не эксперт - директор) сотрудник фирмы.

Итак, задача поставлена. Теперь нужно наглядно ее представить. Для описания подобных задач обычно используются диаграммы, которые называются деревьями решений. Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений.

Разработка дерева решений.

Дерево решений *– это ориентированный граф, вершинами которого являются условия и выводы, а дугами результат выполнения (проверки) условий.*

Диаграммы называются деревьями решений потому, что, подобно настоящему дереву, имеют ветви. Ветви деревьев решений заканчиваются логическими выводами. Для рассматриваемого примера вывод заключается в том, предложит ли директор должность поступающему на работу, и если да, то какую. Многие задачи сложны, и их непросто представить (или для их решения не собираются использовать экспертную систему). Дерево решений помогает преодолеть эти трудности.

На рис.3.8 показано дерево решений для рассматриваемого примера приема на работу. Видно, что диаграмма состоит из кружков и прямоугольников, которые называются вершинами. Каждой вершине присваивается номер. На вершины можно ссылаться по этим номерам. Номера вершин можно выбрать произвольно, т.к они и служат только для удобства идентификации, за исключением первой вершины. Линии, соединяющие вершины, называются дугами. Совокупность вершин и дуг называется ветвями.

Кружки, содержащие вопросы, называются вершинами условий. Прямоугольники содержат логические выводы. Линии (стрелки) показывают направление диаграммы. Подписи возле линий это ответы на вопрос, содержащийся в вершине условия. Вершины условий могут иметь сразу по нескольку выходящих линий (стрелок), связывающих их с другими вершинами. В этом случае каждая линия (стрелка) должна быть четко определена. Не может быть две линии, у которых подписи одинаковые, например, подпись «Да». Выбор выходящей из вершины ветви определяется проверкой условия (вопроса), содержащегося в вершине. В программе под каждую вершину отводится переменная, а затем ей присваивается значение (ответ посетителя). Можно сказать, что вершины содержат переменные, а пути - это условия, в соответствии с которыми переменным присваиваются значения.

В дереве решений могут быть локальные (частные) выводы или цели. Для рассматриваемого примера локальным выводом может быть содержащийся в прямоугольнике 3 ответ на вопрос,будет ли посетителю предложена должность. Однако эта вершина имеет и исходящие ветви, и, следовательно, через неё может проходить путь к следующему логическому выводу. В последнем случае,

поскольку исходящая ветвь не содержит условия и она только одна, говорят, что вершина содержит локальный вывод для другой цели. Локальный вывод - это также составляющая условной части (ЕСЛИ) правила.

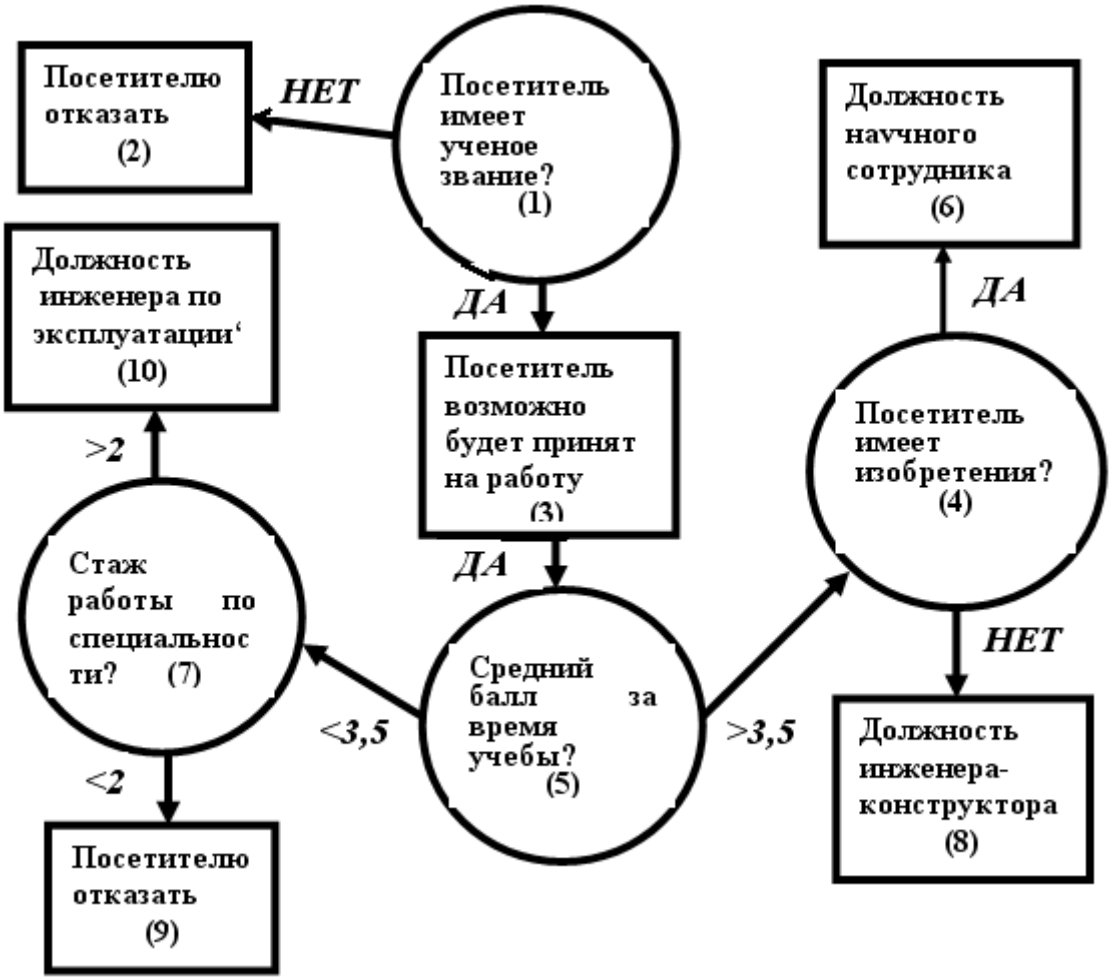


Рис.3.8. Дерево решений для выбора должности

Преобразование дерева решений в правила.

Как уже говорилось, правило «ЕСЛИ-ТО» состоит из двух частей. Часть ЕСЛИ может включать несколько условий, которые связываются между собой логическими операторами И, ИЛИ и НЕ. Часть ТО правила включается в работу только в том случае, если истинны все условия в

условной части. В дереве решений обеим частям правила соответствуют связанные между собой вершина(ы) логического условия(ий) (кружки) и вершина логического вывода (прямоугольник). Условная часть содержит все вершины условия, находящиеся на пути к логическому выводу, т.е. каждая вершина решения на пути к выводу - это одно условие части ЕСЛИ, например, вершины 1 и 4. Вывод же составляет часть ТО правила, в данном примере вершины 6, 8 и т.д.

Порядок формирования правил:

Выбрать из дерева решений вершину вывода (прямоугольник) и зафиксировать её.

В обратном направлении линии (стрелки) найти вершину условия (кружок) и зафиксировать её.

Повторять шаг 2 до тех пор, пока не будут исчерпаны все вершины условия, расположенные в обратном направлении стрелок от зафиксированной вершины вывода, или не встретится вершина локального вывода. Если встретилась вершина локального вывода, то её надо зафиксировать и прекратить выполнение шага 2.

Каждая вершина условия (кружок), составляющая путь, - это одна из переменных части ЕСЛИ правила. Эти вершины объединяются логическим оператором **И.**

Выбранный на шаге 1 логический вывод перенести в часть ТО правила.

Пример создание правила. В качестве примера рассмотрим путь 6, 4, 5, 3. Создание правила начинается с вывода (вершина 6) и дерево решения просматривается в обратную сторону. Просмотр данной ветви (пути)

заканчивается на вершине 3, которая является локальным выводом. Если бы вершины 3 не было в дереве решений, то путь закончился бы на вершине 1.

Применив полученный путь, запишем правило в следующем виде:

ЕСЛИ посетитель, возможно, будет принят на работу = да

И средний балл за время учебы >=3,5

И посетитель имеет изобретения = да,

ТО предложенная должность = научный сотрудник.

Для каждой вершины логического вывода определяется путь и записывается правило. В рассматриваемом дереве решений имеется 6 вершин логического вывода (прямоугольников), в том числе и локальный вывод в вершине 3, значит будет составлено 6 правил.

Длинную фразу “посетитель, возможно, будет принят на работу ” можно заменить переменной, принимающей значения “да” или “нет”. Список имен переменных, текст, который они заменяют, и номера вершин пути сводят в таблицу, (табл.1). Использование переменных вместо полного текста упрощает формирование и запись правил.

Таблица 6.1

Таблица имён переменных

Имя переменной	**Условия**	**Вершина(ы)**
DEGREE	Посетитель имеет ученое звание?	1
QUALIFY	Посетитель, возможно, будет	3

	принят на работу	
PATENT	Посетитель имеет изобретения?	4
EXPERIENCE	Стаж работы по специальности?	7
GRADE	Средний балл за время учебы?	5
POSITION	Предложенная должность	2,6,8,9,10

Используя имена переменных из табл.1, вышеприведенное правило можно запишется в следующем виде:

ЕСЛИ QUALIFY = да **И** GRADE>=3,5 **И** PATENT = да,

ТО POSITION = научный сотрудник

В табл.2 приведены все правила для дерева решений, показанного на рис.6.1. Правила соответствуют всем шести путям, ведущим к шести возможным целям дерева решений. Правила желательно пронумеровать. Совокупность правил является формализованными знаниями (в рассматриваемом примере – знаниями руководителя) и представляет собой *базу знаний*.

Таблица 2

База знаний

№	Правило	Путь
10	**ЕСЛИ** DEGREE = *НЕТ*, **ТО** POSITION=*ОТКАЗАТЬ*	2, 1
20	**ЕСЛИ** DEGREE = *ДА*, **ТО** QUALIFY=*ДА*	3, 1
30	**ЕСЛИ**QUALIFY = *ДА***И**GRADE>=3,5**И**PATENT = *ДА*, **ТО** POSITION = *НАУЧНЫЙ СОТРУДНИК*	6,4,5.3

40	**ЕСЛИ**QUALIFY = *ДА***И**GRADE>=3,5**И**PATENT = *НЕТ*, **ТО** POSITION = *ИНЖЕНЕР КОНСТРУКТОР*	8,4,5,3
50	**ЕСЛИ** QUALIFY = *ДА***И** AVERAGE <= *3,5***И** EXPERIENCE <*2*, **ТО** POSITION=*ОТКАЗАТЬ*	9,7,5,3
60	**ЕСЛИ** QUALIFY = *ДА***И** AVERAGE <= *3,5***И**EXPERIENCE>*2*, **ТО** POSITION = *ИНЖЕНЕРПО ЭКСПЛУАТАЦИИ*	10,7,5, 3

Таким образом, дерево решений позволяет просто и наглядно представить ход рассуждений эксперта при решении задачи и формировать правила для базы знаний, а без базы знаний экспертную систему не построить. Аналогично можно построить базу знаний для своей проблемной области или решаемой задаче.

Структуры данных экспертной системы.

При создании экспертной системы для упрощения ответа на вопросы и решения поставленной задачи в систему включается ряд полезных таблиц или структур данных. Структуры данных нужны для работы с базой знаний. После определения метода решения выбранного круга задач можно приступить к разработке системы.

Список логических выводов - *это структура данных, содержащая упорядоченный список возможных логических выводов.*

Список состоит из номера правила, логического вывода, связанного с этим правилом, и условий, которые формируют вывод. На каждое правило базы знаний в списке приходится одна запись. Создание записи списка поясним на примере правила 10. Часть ТО правила 10 содержит переменную

POSITION, т.е. переменная POSITION связана с логическим выводом правила 10.

Список логических выводов используется исключительно для поиска вывода по номеру правила. Когда условия части ЕСЛИ истинны, вызывается часть ТО правила, ей присваивается значение. Например, если надо узнать, будет ли посетителю предложена работа, в списке ищется переменная POSITION. Она содержится в первой же записи, т.е. в правиле 10:

ЕСЛИ DEGREE=НЕТ,

ТО POSITION=НЕТ

Посетитель не будет принят на работу, если переменная DEGREE имеет значение НЕТ. Если же переменная DEGREE имеет значение ДА, тогда обращаться к части ТО правила нельзя, поскольку не выполняется условие части ЕСЛИ (DEGREE=ДА). Поэтому надо продолжить поиск правила, содержащего в части ТО переменную POSITION (в данном случае правило 30).

На рис.3.9 приведён полностью сформированный список логических выводов для всех правил базы знаний.

10 POSITION
20 QUALIFY
30 POSITION
40 POSITION
50 POSITION
60 POSITION

Рис. 3.9. Список логических выводов.

Список считается сформированным, когда логический вывод каждого правила помещён в запись с номером, совпадающим с номером правила.

Список переменных – это перечень *имен переменных для всех условных частей правил базы знаний и признак их инициализации.*

Признак инициализации показывает, присвоено ли переменной значение. Независимо от того, в скольких условиях встречается переменная, в список переменных она включается всего один раз. В этот список также нельзя включать переменные из списка логических выводов, поскольку их значения определяются с помощью правил. Например, правило 20 использует переменную QUALIFY. Список переменных приведён в таблице3.

Таблица3

Список имен переменных

Имя	**Признак**	**Значение**
DEGREE	I	НЕТ
PATENT	NI	
EXPEREIENCE	NI	
GRADE	NI	

Первоначально предполагается, что переменным значения еще не присвоены и признак инициализации для всех переменных равен NI. По мере того как полученная от посетителя информация передается системе и переменным присваиваются значения, признак инициализации меняется на I. С этого момента, в каком бы правиле в условной части

не встретилась переменная, она будет считаться проинициализированной, имеющей какое либо значение и ее можно использовать для работы с любыми правилами.

Таким образом, до того, как правило включается в работу, все переменные, входящие в его условную часть, должны быть проинициализированы.

Список переменных условия – это перечень всех переменных для всех условных частей всех правил базы знаний.

Условная часть правила (ЕСЛИ) может содержать несколько переменных. Под каждое правило выделяется одинаковое число позиций в списке переменных условия. Минимальное число позиций равно числу переменных условия самого «длинного» правила. Можно добавить еще одну-две позиции «про запас» на случай доработки базы правил.

На рис.3.10 показан список переменных условия для шести правил рассматриваемой базы знаний. Для простоты программирования предполагается, что каждое правило не может содержать больше четырех переменных условия (т.к. самое длинное правило, например №40, содержит три переменных условия). Четвертая позиция добавлена «про запас».

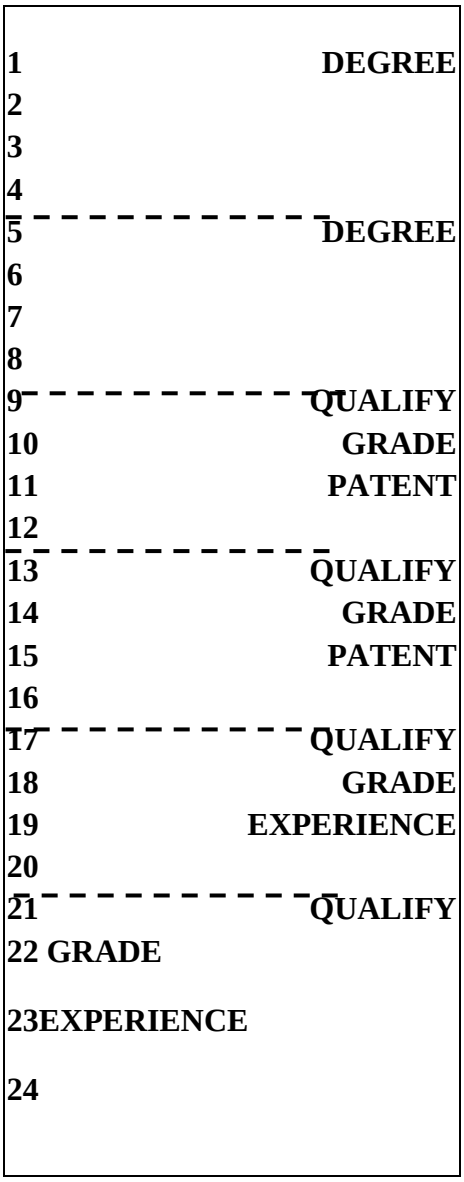


Рис.3.10. Список переменных условия.

Слева от имен переменных даны числа (1-24), указывающие индекс элемента массива (по четыре на правило), в который помещается имя соответствующей переменной. Незанятые элементы массива, отведенные правилу, остаются пустыми. В принципе можно запрограммировать любое число переменных для каждого правила. Однако при отведении места под переменные условия лучше для каждого правила резервировать одинаковое число элементов массива. Это упростит

вычисление индекса первого элемента, отведенного правилу в списке. Его можно вычислить с помощью простой формулы:

№ = 4* (номер правила/10 — 1)+1.

Например, переменные правила 50 будут размещаться, начиная с 17-го элемента массива: 4* (50/10—1)+1=17. №=17.

Теперь посмотрим, каким образом три описанные структуры данных соотносятся с мыслительной деятельностью человека в процессе обратной цепочки рассуждений. Прежде всего, человек просматривает все возможные пути, способные привести к решению задачи (список логических выводов). Затем он выделяет условия, составляющие эти пути (список переменных и список переменных условия). Такие структуры данных позволяют быстро обрабатывать информацию, не повторяя одни и те же шаги по нескольку раз, потому что значения переменных можно использовать в определенной ситуации для различных логических выводов. Если же при разговоре с человеком, устраивающимся на работу, у директора нет не только компьютера, но даже карандаша и бумаги, ему придется много раз переспрашивать, ведь сразу просто невозможно запомнить. Конечно, в конце концов, он примет решение, но затратит много сил и времени.

Глава 4. ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

4.1. Технологии интеллектуального анализа данных

Методы интеллектуального анализа данных.

В настоящее время интенсивно разрабатывались методы автоматического извлечения знаний из накопленных фактов, хранящихся в различных базах данных. Для анализа информации, накопленной в современных базах данных, методы анализа должны быть эффективными, т.е. простыми в использовании, обладать значительным уровнем масштабируемости и определенным автоматизмом. Это концепция, зародившаяся в 1989 г., лежит в основе двух современных технологий анализа данных **Data Mining** и **KDD – Knowledge Discovery in Databases,** которые на русский язык переводятся как «добыча (раскопка) данных» и « «извлечение знаний из баз данных».

Knowledge Discovery in Database(KDD) — процесс получения из данных знаний в виде зависимостей, правил и моделей позволяющих моделирование и прогнозирование различных процессов.

В отечественной литературе применяется термин «**Интеллектуальный анализ данных**» (ИАД). В этой области необходимо отметить пионерские работы отечественных исследователей, в частности, М.М. Бонгарда (программа «Кора»), В.К. Финна (JSM-метод), А.Г. Ивахненко (МГУА), выполнивших свои работы задолго до того, как в этой области на Западе возник настоящий бум.

ИАД*— это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей,*

то есть извлечения информации, которая может быть охарактеризована как знания.

Интеллектуальный анализ данных является кратким обозначением довольно широкого спектра процедур автоматического анализа данных высокоинтеллектуальными технологиями.

В общем случае процесс ИАД состоит из трех стадий:

- выявление закономерностей (свободный поиск);
- использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование);
- анализ исключений, предназначенный для выявления и толкования аномалий в найденных закономерностях.

Причины распространения KDD и Data Mining.

В KDD и Data Mining нет ничего принципиально нового. Специалисты в различных областях человеческого знания решали подобные задачи на протяжении нескольких десятилетий. Однако в последние годы интеллектуальная составляющая бизнеса стала возрастать, и для распространения технологий KDD и Data Mining были созданы все необходимые и достаточные условия.

Развитие технологий автоматизированной обработки информации создало основу для учета сколь угодно большого количества факторов и объема данных.

Возникла острая нехватка высококвалифицированных специалистов в области статистики и анализа данных. Поэтому потребовались технологии обработки и анализа, доступные для специалистов любого профиля за счет

применения методов визуализации и самообучающихся алгоритмов.

Возникла объективная потребность в тиражировании знаний. Полученные в процессе KDD и Data Mining результаты являются формализованным описанием некоего процесса, а следовательно, поддаются автоматической обработке и повторному использованию на новых данных.

На рынке появились программные продукты, поддерживающие технологии KDD и Data Mining, – аналитические платформы. С их помощью можно создавать полноценные аналитические решения и быстро получать первые результаты.

Технология Knowledge Discovery in Databases описывает не конкретный алгоритм или математический аппарат, а последовательность действий, которую необходимо выполнить для построения модели с целю извлечения знания. Она не зависит от предметной области; это набор атомарных операций, комбинируя которые, можно получить нужное решение.

KDD включает в себя этапы подготовки данных, предобработки (очистки) данных, трансформации данных, построения моделей (применения методов Data Mining**)** и интерпретации полученных результатов. Ядром этого процесса являются методы Data Mining**,** позволяющие обнаруживать закономерности и знания (рис.4.1).

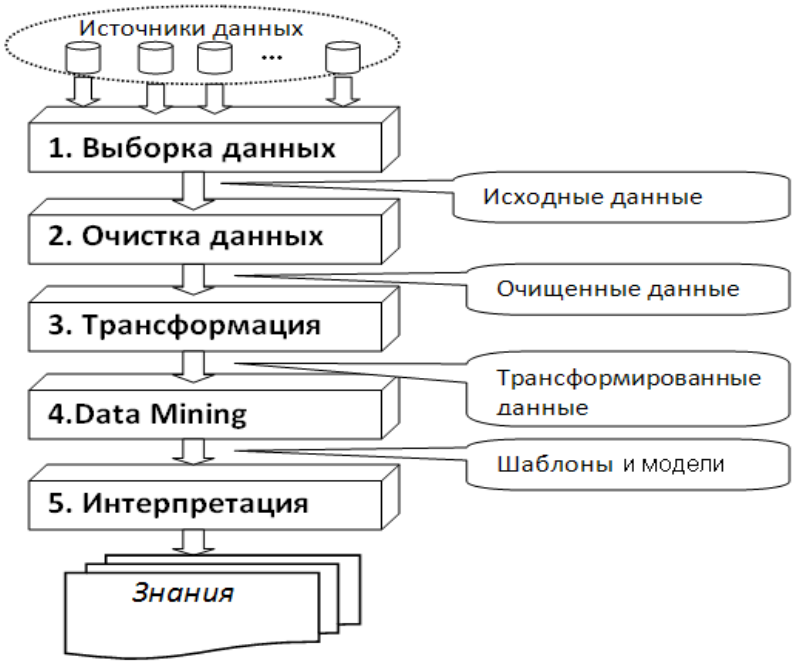


Рис. 4.1. Этапы KDD.

Кратко рассмотрим каждый этап.

1 этап. *Выборка данных*. Этот этап заключается в подготовке набора данных, в том числе из различных источников, выбора значимых параметров и т.д. Для этого должны быть различные инструменты доступа к различным источникам данных – конверторы, запросы, фильтрация данных и т.п. В качестве источника рекомендуется использовать специализированное хранилище данных, агрегирующее всю необходимую для анализа информацию.

2 этап. *Очистка данных*. Реальные данные для анализа редко бывают хорошего качества. Поэтому для эффективного применения методов Data Mining следует обратить серьезное внимание на вопросы предобработки данных. Данные могут содержать пропуски, шумы, аномальные значения и т.д. Кроме того, данные могут быть противоречивы, избыточны, недостаточны, содержать ошибки и т.д.

Для решения каждой из этих проблем есть отработанные методы. Конечно, ошибки можно править и вручную, но при больших объемах данных это становится довольно проблематично. Поэтому рассмотрим варианты решения этих задач в автоматическом режиме при минимальном участии человека.

Противоречивость информации. Для начала нужно решить, что именно считать противоречием. Как ни странно, это задача нетривиальная. Например, пенсионную карточку в России нужно менять в случае изменения фамилии, имени, отчества и пола. Оказывается, в том, что человек родился женщиной, а вышел на пенсию мужчиной, противоречия нет!

После того, как мы определимся с тем, что считать противоречием и найдем их, есть несколько вариантов действий.

При обнаружении нескольких противоречивых записей, удалять их. Метод простой, а потому легко реализуемый. Иногда этого бывает вполне достаточно. Тут важно не переусердствовать, иначе мы можем вместе с водой выплеснуть младенца.

Исправить противоречивые данные. Можно вычислить вероятность появления каждого из противоречивых событий и выбрать наиболее вероятное. Это самый грамотный и корректный метод работы с противоречиями.

Пропуски в данных. Очень серьезная проблема. Это вообще бич для большинства хранилищ данных. Большинство методов прогнозирования исходят из предположения, что данные поступают равномерным постоянным потоком. На практике такое встречается крайне редко. Поэтому одна из самых востребованных областей

применения хранилищ данных – прогнозирование – оказывается реализованной некачественно или со значительными ограничениями. Для борьбы с этим явлением можно воспользоваться следующими методами:

Аппроксимация. Т.е. если нет данных в какой-либо точке, мы берем ее окрестность и вычисляем по известным формулам значение в этой точке, добавляя соответствующую запись в хранилище. Хорошо это работает для упорядоченных данных. Например, сведения об ежедневных продажах продуктов.

Определение наиболее правдоподобного значения. Для этого берется не окрестность точки, а все данные. Этот метод применяется для неупорядоченной информации, т.е. случаем, когда мы не в состоянии определить, что же является окрестностью исследуемой точки.

Аномальные значения. Довольно часто происходят события, которые сильно выбиваются из общей картины. И лучше всего такие значения откорректировать. Это связано с тем, что средства прогнозирования ничего не знают о природе процессов. Поэтому любая аномалия будет восприниматься как совершенно нормальное значение. Из-за этого будет сильно искажаться картина будущего. Какой-то случайный провал или успех будет считаться закономерностью.

Есть метод борьбы и с этой напастью – это робастные оценки. Это методы устойчивые к сильным возмущениям. Мы оцениваем имеющиеся данные ко всему, что выходит за допустимые границы, и применяем одно из следующих действий:

Значение удаляется;

Заменяется на ближайшее граничное значение.

Шум. Почти всегда при анализе мы сталкиваемся с шумами. Шум не несет никакой полезной информации, а лишь мешает четко разглядеть картину. Методов борьбы с этим явлением несколько.

Спектральный анализ. При помощи него мы можем отсечь высокочастотные составляющие данных. Проще говоря, это частые и незначительные колебания около основного сигнала. Причем, изменяя ширину спектра, можно выбирать какого рода шум мы хотим убрать.

Авторегрессионые методы. Этот довольно распространенный метод активно применяется при анализе временных рядов и сводится к нахождению функции, которая описывает процесс плюс шум. Собственно шум после этого можно удалить и оставить основной сигнал.

Ошибки ввода данных. Вообще это тема для отдельного разговора, т.к. количество типов такого рода ошибок слишком велико, например, опечатки, сознательное искажение данных, несоответствие форматов, и это еще не считая типовых ошибок, связанных с особенностями работы приложения по вводу данных. Для борьбы с большинством из них есть отработанные методы. Некоторые вещи очевидны, например, перед внесением данных в хранилище можно провести проверку форматов. Некоторые более изощренные. Например, можно исправлять опечатки на основе различного рода тезаурусов. Но, в любом случае, очищать нужно и от такого рода ошибок.

К задачам очистки данных относятся: заполнение пропусков, подавление аномальных значений, сглаживание, исключение дубликатов и противоречий и пр.

Ошибочно предполагать, что если подать данные на вход системы в существующем виде, то на выходе будут получены полезные знания. Входные данные должны быть качественны и корректны.

3 этап. *Трансформация данных.* Этот шаг необходим для тех методов, которые требуют, чтобы исходные данные были в каком-то определенном виде. Дело в том, что различные алгоритмы анализа требуют специальным образом подготовленные данные. Например, для прогнозирования необходимо преобразовать временной ряд при помощи скользящего окна или вычисление агрегируемых показателей. К задачам трансформации данных относятся: скользящее окно, приведение типов, выделение временных интервалов, преобразование непрерывных значений в дискретные и наоборот, сортировка, группировка и прочее.

4 этап. *Data Mining.* На этом этапе строятся модели, в которых применяются различные алгоритмы для нахождения знаний. Это нейронные сети, деревья решений, алгоритмы кластеризации и установления ассоциаций и т.д.

5 этап. *Интерпретация.* На данном этапе осуществляется применение пользователем полученных моделей (знаний) в бизнес приложениях. Для оценки качества полученной модели нужно использовать как формальные методы, так и знания аналитика. Именно аналитик может сказать, насколько применима полученная модель к реальным данным.

Пример. Имеется сеть магазинов розничной торговли. Требуется получить прогноз объемов продаж на следующий месяц. Первым шагом будет сбор истории продаж в каждом магазине и объединение ее в общую выборку данных. Следующим шагом будет предобработка собранных данных:

их группировка по месяцам, сглаживание кривой продаж, устранение факторов, слабо влияющих на объемы продаж. Далее следует построить модель зависимости объемов продаж от выбранных факторов. Это можно сделать с помощью линейной регрессии или нейронных сетей. Имея такую модель, можно получить прогноз, подав на вход модели историю продаж. Зная прогнозное значение, его можно использовать, например, в приложениях оптимизации для лучшего размещения товара на складе.

Полученные модели являются, по сути, формализованными знаниями эксперта, а следовательно, их можно **тиражировать**. В этом заключается самое главное преимущество KDD. Т.е. построенную одним человеком модель могут применять другие, без необходимости понимания методик, при помощи которой эти модели построены. Найденные знания должны быть использованы на новых данных с некоторой степенью достоверности.

Задачи и методы Data Mining.

Термин Data Mining дословно переводится как «добыча данных» или «раскопка данных» и имеет в англоязычной среде несколько определений.

Data Mining *— обнаружение в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.*

Информация, найденная в процессе применения методов Data Mining, должна быть нетривиальной и ранее неизвестной, например, сведения о средних продажах

таковыми не являются. Знания должны описывать новые связи между свойствами, предсказывать значения одних признаков на основе других.

Нередко KDD отождествляют с Data Mining. Однако правильнее считать Data Mining шагом процесса KDD.

Приведенные ниже примеры из разных областей экономики демонстрируют основное преимущество методов Data Mining – способность обнаружения новых знаний, которые невозможно получить методами статистического, регрессионного анализа или эконометрики.

1. Множество клиентов компании с помощью одного из инструментов Data Mining были объединены в группы, или сегменты со схожими признаками. Это позволило проводить компании различную маркетинговую политику и строить отдельные модели поведения для каждого клиентского сегмента. Наиболее значимыми факторами для разделения на группы оказались следующие: удаленность региона клиента, сфера деятельности, среднегодовые суммы сделок, количество сделок в неделю.

2. Автоматический анализ банковской базы данных кредитных сделок физических лиц выявил правила, по которым потенциальным заемщикам отказывалось в выдаче кредита. В частности, решающими факторами при выдаче кредитов на небольшие суммы, оказались: срок кредита, среднемесячный доход и расход заемщика. В дальнейшем это учитывалось при экспресс-кредитовании наиболее дешевых товаров.

3. При анализе базы данных клиентов страховой компании был установлен социальный портрет человека, страхующего жизнь - это оказался мужчина 35-50 лет,

имеющий 2 и более детей и среднемесячный доход выше $2000.

Задачи, решаемые методами Data Mining:

Классификация – отнесение объектов (наблюдений, событий) к одному из заранее известных классов. Это делается посредством анализа уже классифицированных объектов и формулирования некоторого набора правил.

Классификация используется в случае, если заранее известны классы отнесения объектов. Например, отнесение нового товара к той или иной товарной группе, отнесение клиента к какой-либо категории. При кредитовании это может быть, например, отнесение клиента по каким-то признакам к одной из групп риска.

Кластеризация – это группировка объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность объектов. Причем в отличие от классификации, группы заранее не заданы. Объекты внутри кластера должны быть «похожими» друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем больше похожи объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация. Часто применительно к экономическим задачам вместо кластеризации употребляют термин *сегментация*.

Кластеризация может использоваться для сегментации и построения профилей клиентов (покупателей). При достаточно большом количестве клиентов становится трудно подходить к каждому индивидуально. Поэтому клиентов удобно объединить в группы – сегменты с однородными признаками. Выделять сегменты клиентов можно по нескольким группам признаков. Это могут быть сегменты по

сфере деятельности, по географическому расположению. После сегментации можно узнать, какие именно сегменты являются наиболее активными, какие приносят наибольшую прибыль, выделить характерные для них признаки. Эффективность работы с клиентами повышается за счет учета их персональных предпочтений.

Прогнозирование (регрессия). Это установление зависимости непрерывных выходных переменных от входных. К этому же типу задач относится прогнозирование временного ряда на основе исторических данных.

Регрессия используется для установления зависимостей в факторах. Например, в задаче прогнозирования зависимой величиной является объемы продаж, а факторами, влияющими на эту величину, могут быть предыдущие объемы продаж, изменение курса валют, активность конкурентов и т.д. Или, например, при кредитовании физических лиц вероятность возврата кредита зависит от личных характеристик человека, сферы его деятельности, наличия имущества.

Ассоциация – выявление закономерностей между связанными событиями. Примером такой закономерности служит правило, указывающее, что из события X следует событие Y. Такие правила называются ассоциативными.

Впервые это задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis). *Ассоциации* помогают выявлять совместно приобретаемые товары. Это может быть полезно для более удобного размещения товара на прилавках, стимулирования продаж. Тогда человек,

купивший пачку спагетти, не забудет купить к ним бутылочку соуса.

Последовательность (последовательные шаблоны)– установление закономерностей между связанными во времени событиями.

Последовательные шаблоны могут быть использованы при планировании продаж или предоставлении услуг. Пример последовательного шаблона: если человек приобрел фотопленку, то через неделю он отдаст ее на проявку и закажет печать фотографий.

Методы в Data Mining.

Для решения вышеперечисленных задач используются различные методы и алгоритмы Data Mining. Ввиду того, что Data Mining развивался и развивается на стыке таких дисциплин, как математика, статистика, теория информации, машинное обучение, теория баз данных, вполне закономерно, что большинство алгоритмов и методов Data Mining были разработаны на основе различных методов из этих дисциплин.

На сегодня наибольшее распространение получили самообучающиеся методы и машинное обучение. Рассмотрим кратко наиболее известные алгоритмы и методы, применяющиеся для решения каждой задачи Data Mining.

Деревья решений. Деревья решений (decision trees) предназначены для решения задач классификации. Они создают иерархическую структуру классифицирующих правил типа «ЕСЛИ…ТО…» (if-then), имеющую вид дерева. Дерево решений состоит из узлов (рис.4.2), где производится проверка условия, и листьев – конечных узлов дерева, указывающих на класс (узлов решения).

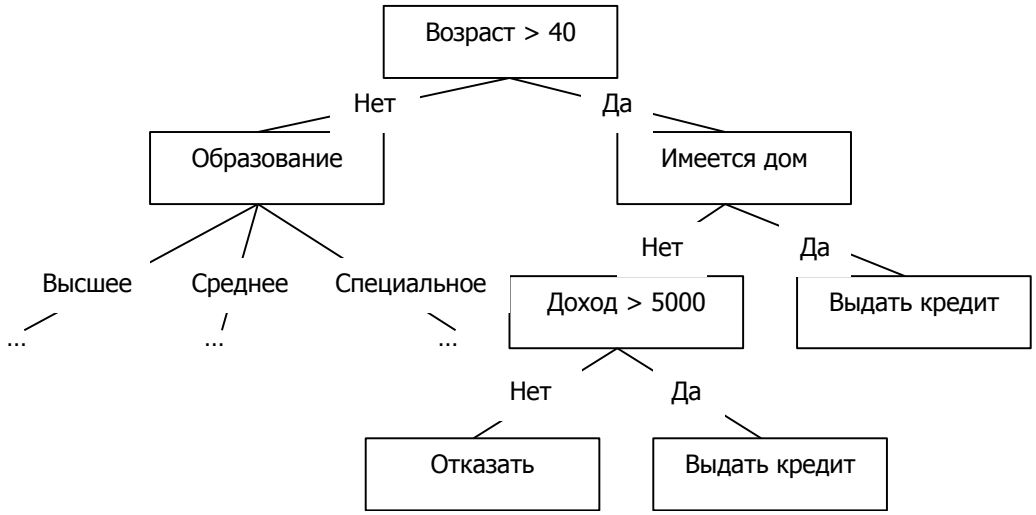


Рис. 2.6. Пример дерева решений.

Дерево решений строится по определенному алгоритму. Наибольшее распространение получили алгоритмы CART и C4.5(C5.0).

Искусственные нейронные сети (ИНС). Искусственные нейронные сети, в частности, многослойный персептрон, решают задачи регрессии и классификации. Однако, в отличие от дерева решений, нейронные сети не способны объяснять выдаваемое решение, поэтому их работа напоминает «черный ящик» со входами и выходами.

Нейронные сети моделируют простые биологические процессы, аналогичные процессам, происходящим в человеческом мозге. ИНС способны к адаптивному обучению путем реакции на положительные и отрицательные воздействия.

В основе их построения лежит элементарный преобразователь, называемый *искусственным нейроном* или просто *нейроном* по аналогии с его биологическим прототипом.

Структуру нейросети – многослойного персептрона (рис.4.3) - можно описать следующим образом. Нейросеть состоит из нескольких слоев: входной, внутренний (скрытый) и выходной слои. Входной слой реализует связь с входными данными, выходной – с выходными. Внутренних слоев может быть от одного и больше. В каждом слое содержится несколько нейронов. Все нейроны соединяются между собой связями, называемые *весами*.

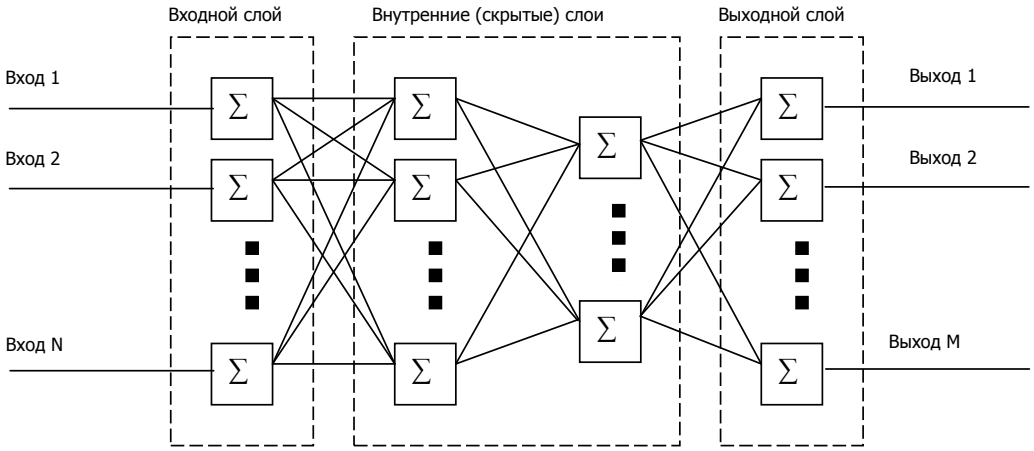


Рис.4.3. Многослойный персептрон.

Перед использованием нейронной сети производится ее обучение, что представляет собой итерационный процесс настройки весовых коэффициентов. Для обучения применяются специальные алгоритмы. Наибольшее распространение получили градиентные методы обучения – алгоритм обратного распространения ошибки (Back Propagation), сопряженных градиентов, RProp и другие. Для проверки адекватности построенной нейронной сети используется специальный прием - тестовое подтверждение.

Основное достоинство нейронных сетей состоит в том, что они моделируют сложные нелинейные зависимости между входными и выходными переменными.

Линейная регрессия. Линейная регрессия, как это следует из названия, решает задачи регрессии. Но она предназначена для поиска линейных зависимостей в данных. Если же зависимости нелинейные, то модель с использованием линейной регрессии может быть не построена вообще. Для этого лучше воспользоваться более универсальным методом нахождения зависимостей, например, искусственной нейронной сетью.

Кластерный анализ. Главное назначение кластерного анализа – разбиение множества исследуемых объектов и признаков на однородные кластеры (группы). Большое достоинство кластерного анализа в том, что он позволяет производить разбиение объектов не по одному параметру, а по целому набору признаков. Кроме того, кластерный анализ в отличие от большинства математико-статистических методов не накладывает никаких ограничений на вид рассматриваемых объектов, и позволяет рассматривать множество исходных данных практически произвольной природы.

Самоорганизующиеся карты. Самоорганизующиеся карты (Self Organizing Maps – SOM), или карты Кохонена, так же как и методы кластерного анализа, используются для решения задач кластеризации и сегментирования. Самоорганизующаяся карта является разновидностью нейронной сети.

Ассоциативные правила. Ассоциативные правила (association rules) позволяют находить закономерности между связанными событиями, т.е . выявление ассоциаций. Примером ассоциативного правила, служит утверждение, что покупатель, приобретающий хлеб, приобретет и молоко с вероятностью 75%. Впервые эта задача была предложена для

поиска ассоциативных правил для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis). Ассоциативные правила эффективно используются в сегментации покупателей по поведению при покупках, анализе предпочтений клиентов, планировании расположения товаров в супермаркетах, адресной рассылке. Классическим алгоритмом нахождения ассоциативных правил считается алгоритм APriori.

Последовательные шаблоны. Последовательные шаблоны (sequential patterns) представляют собой закономерности между связанными во времени событиями. Алгоритмы последовательных шаблонов похожи на алгоритмы ассоциативных правил. Распространение получили AprioriAll и AprioriSome.

4.2. Хранилища данных

Понятие и концепция хранилищ данных.

Предпосылки появления ХД. Со временем в различных информационных системах начали аккумулироваться большие объемы данных – документы, сведения о банковских операциях, информация о клиентах, заключенных сделках, оказанных услугах и т. д.

Постепенно возникло понимание того, что сбор данных не самоцель. Собранная информация может оказаться весьма полезной в процессе управления организацией, поиска путей совершенствования деятельности и получения посредством этого конкурентных преимуществ. Но для этого нужны системы, которые позволяли бы выполнять не только

простейшие действия над данными: подсчитывать суммы, средние, максимальные и минимальные значения. Появилась потребность в информационных системах, которые позволяли бы проводить глубокую аналитическую обработку, для чего необходимо решать такие задачи, как поиск скрытых структур и закономерностей в массивах данных, вывод из них правил, которым подчиняется данная предметная область, стратегическое и оперативное планирование, формирование нерегламентированных запросов, принятие решений и прогнозирование их последствий.

Понимание преимуществ, которые способен дать интеллектуальный анализ, привело к появлению нового класса информационных систем – систем поддержки принятия решений (СППР), ориентированных на аналитическую обработку данных с целью получения знаний, необходимых для разработки решений в области управления. Дополнительным стимулом совершенствования этих систем стали такие факторы, как снижение стоимости высокопроизводительных компьютеров и расходов на хранение больших объемов информации, появление возможности обработки больших массивов данных и развитие соответствующих математических методов. Обобщенная структурная схема системы СППР представлена на рис. 4.4.

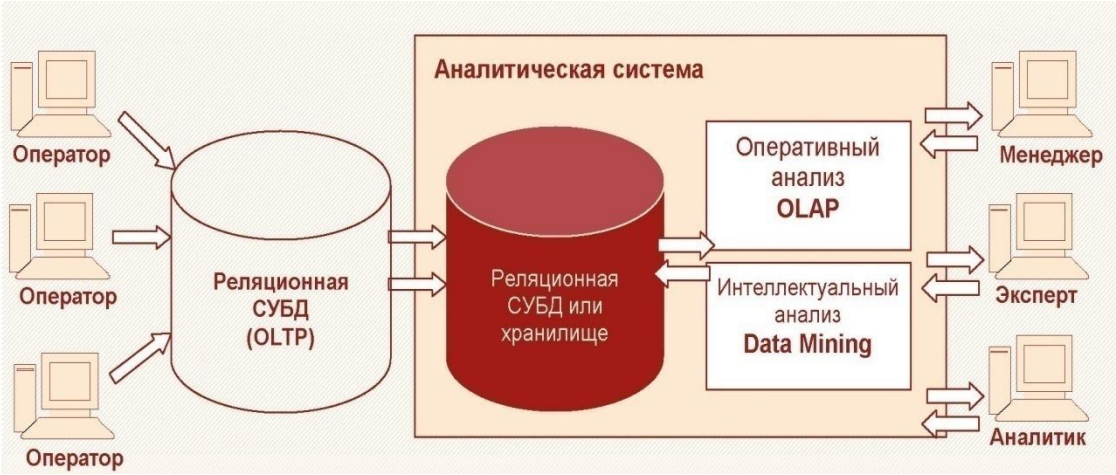


Рис.4.4. Структура СППР.

С СППР используются специализированные базы данных, которые называются хранилищами данных (ХД). Хранилища данных ориентированы на аналитическую обработку и удовлетворяют требованиям, предъявляемым к системам поддержки принятия решений.

В настоящее время однозначного определения ХД не существует, из-за того что разработано большое количество различных архитектур и технологий ХД, а сами хранилища используются для решения самых разнообразных задач. Каждый автор вкладывает в это понятие свое видение вопроса. Обобщая требования, предъявляемые к СППР, можно дать следующее определение ХД, которое не претендует на полноту и однозначность, но позволяет понять основную идею.

Хранилище данных - *разновидность систем хранения, ориентированная на поддержку процесса анализа данных, обеспечивающая целостность, непротиворечивость и хронологию данных, а также высокую скорость выполнения аналитических запросов.*

Применительно к решению бизнес-задач, ***хранилище данных*** *– это специальным образом систематизированная информация из разнородных источников (базы данных учетных систем компании, маркетинговые данные, мнения клиентов, исследования конкурентов и т.п.), необходимая для обработки с целью принятия стратегически важных решений в деятельности компании.*

Для того чтобы получить качественный прогноз, нужно собрать максимум информации об исследуемом процессе, описывающей его с разных сторон. Например, для прогнозирования объемов продаж может потребоваться различная и разнородная следующая информация (рис.2.2).

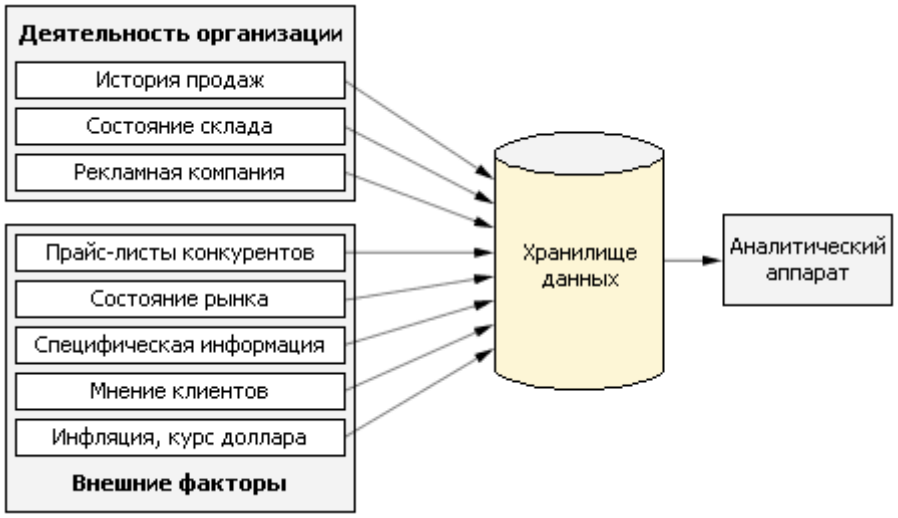


Рис. 4.5. Хранилище данных

Типичное ХД существенно отличается от обычных систем хранения данных (баз данных).

Главным отличием являются цели их создания и использования. База данных играет роль помощника в оперативном управлении организации. Это каждодневные задачи получения актуальной информации: бухгалтерской отчетности, учета договоров и т.д. В свою очередь

хранилище данных консолидирует всю необходимую информацию для осуществления задач стратегического управления в среднесрочном и долгосрочном периоде. Например, продажа товара и выписка счета производятся с использованием базы данных, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, — с помощью хранилища данных.

Другое важное отличие заключается в динамике изменения данных. Базы данных в OLTP-системах характеризуются очень высокой динамикой изменения записей из-за повседневной работы большого числа пользователей (откуда, кстати, велика вероятность появления противоречий, ошибок, нарушения целостности данных и т. д.). Что касается ХД, то данные из него не удаляются, а пополнение происходит в соответствии с определенным регламентом (раз в час, день, неделю, в определенное время).

Важнейшим элементом ХД является **семантический слой** – механизм, позволяющий аналитику оперировать данными посредством бизнес-терминов предметной области. Семантический слой дает пользователю возможность сосредоточиться на анализе и не задумываться о механизмах получения данных.

Основные положения концепции ХД.

Принято считать, что у истоков концепции ХД стоял технический директор компании Prism Solutions Билл Инмон, который в начале 1990-х г. опубликовал ряд работ, ставших основополагающими для последующих исследований в области аналитических систем.

Б. Инмон дал следующее определение ХД*: предметно-ориентированный, интегрированный, неизменяемый и*

поддерживающий хронологию набор данных, предназначенный для обеспечения принятия управленческих решений.

В основе концепции хранилищ данных лежат следующие положения (принципы):

Предметная ориентированность. В данном случае подразумевается, что ХД должно разрабатываться с учетом специфики конкретной предметной области, а не аналитических приложений, с которыми его предполагается использовать. Структура ХД должна отражать представления аналитика об информации, с которой ему приходится работать.

Интегрированность означает, что должна быть обеспечена возможность загрузки в ХД информации из источников, поддерживающих различные форматы данных и созданных в различных приложениях – учетных системах, базах данных, электронных таблицах и других офисных приложениях, поддерживающих структурированность данных (например, текстовые файлы с разделителями). При этом данные, допускающие различный формат (например, числа, дата и время), в процессе загрузки должны быть преобразованы к единому представлению. Кроме того, очень важно проверить загружаемые данные на целостность и непротиворечивость, обеспечить необходимый уровень их обобщения (агрегирования). Объем данных в хранилище должен быть достаточным для эффективного решения аналитических задач, поэтому в ХД может накапливаться информация за несколько лет и даже десятилетий.

Принцип неизменчивости предполагает, что, в отличие от обычных систем оперативной обработки данных, в ХД

данные после загрузки не должны подвергаться каким-либо изменениям, за исключением добавления новых данных.

Поддержка хронологии означает соблюдение порядка следования записей, для чего в структуру ХД вводятся ключевые атрибуты *Дата* и *Время*. Кроме того, если физически упорядочить записи в хронологическом порядке, например в порядке возрастания атрибута *Дата*, можно уменьшить время выполнения аналитических запросов.

Основные требования к ХД.

Чтобы ХД выполняло функции, соответствующие его основной задаче – поддержке процесса анализа данных, – оно должно удовлетворять требованиям, сформулированным одним из авторов концепции ХД – Р. Кимбаллом:

- высокая скорость получения данных из хранилища;
- автоматическая поддержка внутренней непротиворечивости данных;
- возможность получения и сравнения срезов данных;
- наличие удобных средств для просмотра данных в хранилище;
- обеспечение целостности и достоверности хранящихся данных.

Чтобы соблюсти все перечисленные требования, для построения и работы ХД, как правило, используется не одно приложение, а система, в которую входит несколько программных продуктов. Одни из них представляют собой собственно систему хранения данных, другие – средства их просмотра, извлечения, загрузки и т. д.

Архитектуры хранилищ данных.

Разработка и построение корпоративного ХД – это дорогостоящая и трудоемкая задача. Успешность внедрения ХД во многом зависит от уровня информатизации бизнес-процессов в компании, установившихся информационных потоков, объема и структуры используемых данных, требований к скорости выполнения запросов и частоте обновления хранилища, характера решаемых аналитических задач и т. д. Чтобы приблизить ХД к условиям и специфике конкретной организации, в настоящее время разработано несколько архитектур хранилищ:

- многомерные,
- реляционные,
- гибридные
- виртуальные.

Многомерные ХД реализуют многомерное представление данных на физическом уровне в виде многомерных кубов. Данная технология получила название MOLAP – Multidimensional OLAP.

Реляционные ХД используют классическую реляционную модель, характерную для оперативных регистрирующих OLTP-систем. Данные хранятся в реляционных таблицах, но образуют специальные структуры эмулирующие многомерное представление данных. Такая технология обозначается аббревиатурой ROLAP – Relational OLAP.

Гибридные ХД сочетают в себе свойства как реляционной, так и многомерной моделей данных. В гибридных ХД детализированные данные хранятся в реляционных таблицах, а агрегаты – в многомерных кубах. Такая технология построения ХД называется HOLAP – Hybrid OLAP.

Виртуальные ХД не являются хранилищами данных в привычном понимании. В таких системах работа ведется с отдельными источниками данных, но при этом эмулируется работа обычного ХД. Иначе говоря, данные не консолидируются физически, а собираются непосредственно в процессе выполнения запроса.

Кроме того, все ХД можно разделить на одноплатформенные и кросс-платформенные. Одноплатформенные ХД строятся на базе только одной СУБД, а кросс-платформенные могут строиться на базе нескольких СУБД.

Многомерные хранилища данных. Многомерная модель данных, лежащая в основе построения многомерных хранилищ данных (МХД), опирается на концепцию многомерных кубов, или гиперкубов. Они представляют собой упорядоченные многомерные массивы, которые также часто называют OLAP-кубами (аббревиатура OLAP расшифровывается как On-Line Analytical Processing - оперативная аналитическая обработка). Технология OLAP представляет собой методику оперативного извлечения нужной информации из больших массивов данных и формирования соответствующих отчетов.

Не следует пытаться провести геометрическую интерпретацию понятия «многомерный куб», поскольку это просто служебный термин, описывающий метод представления данных.

В основе многомерного представления данных лежит их разделение на две группы – **измерения** и **факты**.

Измерения – это категориальные атрибуты, наименования и свойства объектов, участвующих в

некотором бизнес-процессе. Измерениями являются наименования товаров, названия фирм-поставщиков и покупателей, ФИО людей, названия городов и т. д. Измерения могут быть и числовыми, если какой-либо категории (например, наименованию товара) соответствует числовой код, но в любом случае это данные дискретные, то есть принимающие значения из ограниченного набора. Измерения качественно описывают исследуемый бизнес-процесс.

Факты – это данные, количественно описывающие бизнес-процесс, непрерывные по своему характеру, то есть они могут принимать бесконечное множество значений. Примеры фактов – цена товара или изделия, их количество, сумма продаж или закупок, зарплата сотрудников, сумма кредита, страховое вознаграждение и т. д.

Структура многомерного куба. Многомерный куб можно рассматривать как систему координат, осями которой являются измерения, например, *Дата*, *Товар*, *Город*, а фактами - числовые значения, соответствующие этим измерениям. Принцип организации многомерного куба поясняется на рис.4.6. В такой системе каждому набору значений измерений (например, дата – товар – покупатель) будет соответствовать ячейка, в которой размещаются числовые показатели (то есть факты), связанные с данным набором. Таким образом, между объектами бизнес-процесса и их числовыми характеристиками будет установлена однозначная связь. Например в одной ячейке будут располагаться факты, относящиеся к продаже Товара2 во Владивостоке в декабре месяце.

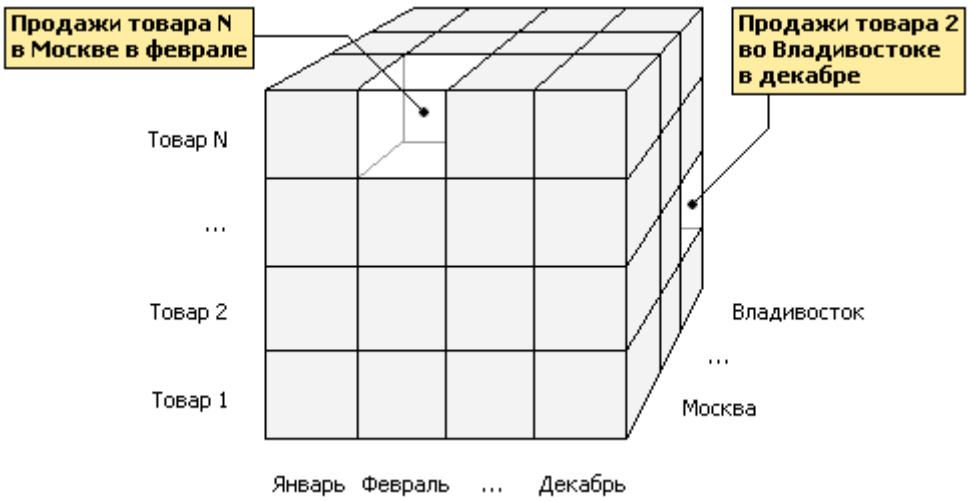


Рис. 4.6. Данные в трехмерном кубе.

Кроме того, куб позволяет реализовать сечения данных. Сечение заключается в выделении подмножества ячеек гиперкуба при фиксировании значения одного или нескольких измерений. В результате сечения получается срез или несколько срезов, каждый из которых содержит информацию, связанную со значением измерения, по которому он был построен. Например, если выполнить сечение по значению *«г. Москва»* измерения *Город*, то полученный в результате срез будет содержать информацию об истории продаж всех товаров, которую можно будет свести в плоскую таблицу. При необходимости ограничить информацию только одним товаром (например, *Товаром 2*) потребуется выполнить еще одно сечение, но теперь уже по значению *Товар 2* измерения *Товар*. Результатом построения двух срезов будет информация о продажах в одном городе одного товара. Манипулируя таким образом сечениями гиперкуба, пользователь всегда может получить информацию в нужном разрезе. Затем на основе построенных срезов может быть сформирована кросс-таблица (рис 4.7), и с ее помощью очень быстро получен необходимый отчет. Данная методика лежит в основе технологии **OLAP**-анализа.

		Товар				
Месяц	ГОРОД	Товар 1	Товар 2	Товар 3	Товар 4	Итого
1	Москва	5 621.20	5 300.21	6 504.24	8 280.13	25 705.78
	С.-Петербург	5 970.10	11 012.78	11 481.84	14 152.99	42 617.71
	Итого	11 591.30	16 312.99	17 986.08	22 433.12	68 323.49
2	Москва	5 607.14	5 190.18	10 175.40	7 207.57	28 180.29
	С.-Петербург	7 948.41	10 186.84	15 558.77	16 869.14	50 563.16
	Итого	13 555.55	15 377.02	25 734.17	24 076.71	78 743.45
3	Москва	4 861.97	5 874.40	6 732.08	3 590.55	21 059.00
	С.-Петербург	9 841.05	8 934.44	12 565.70	14 316.01	45 657.20
	Итого	14 703.02	14 808.84	19 297.78	17 906.56	66 716.20
4	Москва	3 738.74	6 786.68	5 773.56	5 986.63	22 285.61
	С.-Петербург	10 295.08	9 155.90	14 810.26	18 866.58	53 127.82
	Итого	14 033.82	15 942.58	20 583.82	24 853.21	75 413.43
5	Москва	5 173.74	5 447.55	8 196.45	6 682.17	25 499.91
	С.-Петербург	8 380.08	11 119.77	10 390.82	17 555.45	47 446.12
	Итого	13 553.82	16 567.32	18 587.27	24 237.62	72 946.03

Рис. 4.7. Пример многомерного отчета.

Таким образом, информация в многомерном хранилище данных является логически целостной. Это уже целостные структуры типа «кому, что и в каком количестве было продано на данный момент времени».

Преимущества многомерного подхода к организации данных очевидны.

Представление данных в виде многомерных кубов более наглядно, чем совокупность нормализованных таблиц реляционной модели, структуру которой представляет только администратор БД.

Возможность построения более широкого спектра аналитических запросов.

В некоторых случаях использование многомерной модели позволяет значительно уменьшить продолжительность поиска в ХД, обеспечивая выполнение аналитических запросов практически в режиме реального времени. Это связано с тем, что агрегированные данные

вычисляются предварительно и хранятся в многомерных кубах вместе с детализированными, поэтому тратить время на вычисление агрегатов при выполнении запроса уже не нужно.

Использование многомерной модели данных сопряжено с определенными трудностями. Так, для ее реализации требуется больший объем памяти. Это связано с тем, что при реализации физической многомерности используется большое количество технической информации, поэтому объем данных, который может поддерживаться МХД, обычно не превышает нескольких десятков гигабайт. Кроме того, многомерная структура труднее поддается модификации; при необходимости встроить еще одно измерение требуется выполнить физическую перестройку всего многомерного куба.

На основании этого можно сделать вывод, что применение систем хранения, в основе которых лежит многомерное представление данных, целесообразно только в тех случаях, когда объем используемых данных сравнительно невелик, а сама многомерная модель имеет стабильный набор измерений.

Реляционные хранилища данных. Применение реляционной модели при создании ХД в ряде случаев позволяет получить преимущества над многомерной технологией, особенно в части эффективности работы с большими массивами данных и использования памяти компьютера. На основе реляционных хранилищ данных (РХД) строятся ROLAP-системы (Relational OLAP).

В основе технологии РХД лежит принцип, в соответствии с которым измерения хранятся в плоских таблицах так же, как и в обычных реляционных СУБД, а

факты (агрегируемые данные) – в отдельных специальных таблицах этой же базы данных. При этом таблица фактов является основой для связанных с ней таблиц измерений. Она содержит количественные характеристики объектов и событий, совокупность которых предполагается в дальнейшем анализировать.

Схемы построения РХД. На логическом уровне различают две схемы построения РХД – «звезда» и «снежинка».

При использовании схемы «звезда» центральной является таблица фактов, с которой связаны все таблицы измерений. Таким образом, информация о каждом измерении располагается в отдельной таблице, что упрощает их просмотр, а саму схему делает логически прозрачной и понятной пользователю (рис. 4.8).

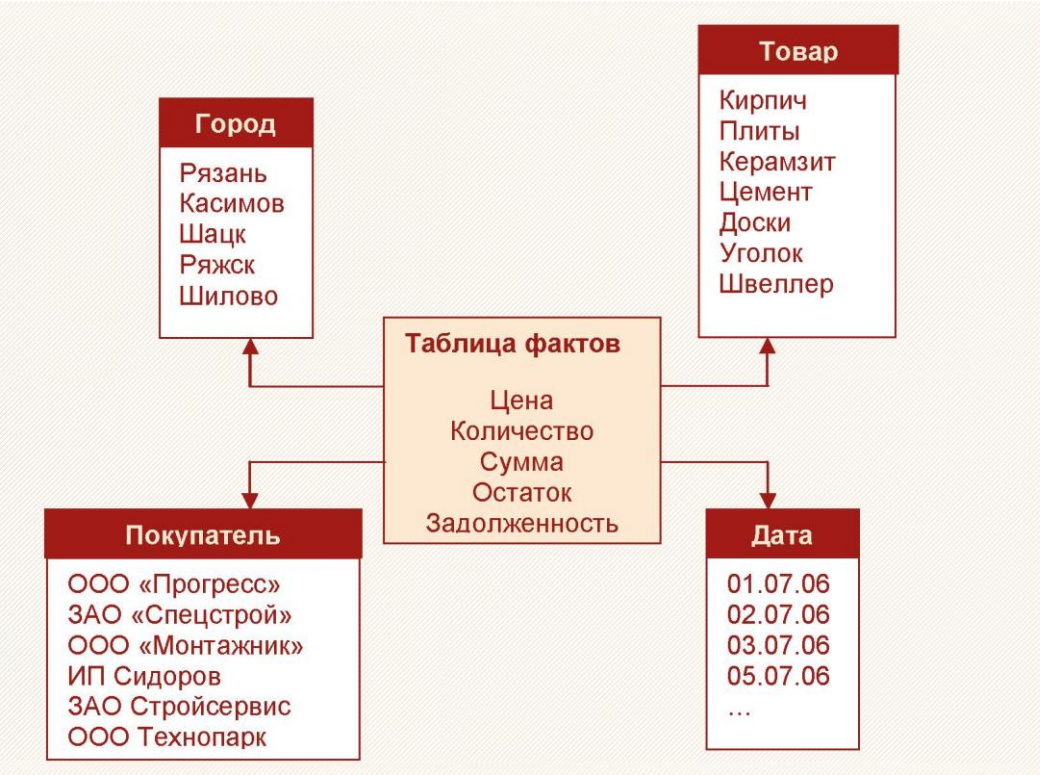


Рис.4.8. Схема построения РХД «звезда».

Однако размещение всей информации об измерении в одной таблице оказывается не всегда оправданным. Например, если продаваемые товары объединены в группы (имеет место иерархия), то придется тем или иным способом показать, к какой группе относится каждый товар, что приведет к многократному повторению названий групп. Это не только вызовет рост избыточности, но и повысит вероятность возникновения противоречий (если, например, один и тот же товар ошибочно отнесут к разным группам).

К преимуществам схемы «звезда» можно отнести:

- простоту и логическую прозрачность модели;
- более простую процедуру пополнения измерений, поскольку приходится работать только с одной таблицей.

Недостатками схемы «звезда» являются:

- медленная обработка измерений, поскольку одни и те же значения измерений могут встречаться несколько раз в одной и той же таблице;
- высокая вероятность возникновения несоответствий в данных (в частности, противоречий), например, из-за ошибок ввода.

Для более эффективной работы с иерархическими измерениями была разработана модификация схемы «звезда», которая получила название «снежинка». Главным отличием схемы «снежинка» является то, что информация об одном измерении может храниться в нескольких связанных таблицах. То есть если хотя бы одна из таблиц измерений имеет одну или несколько связанных с ней других таблиц

измерений, в этом случае будет применяться схема «снежинка» (рис. 4.9).

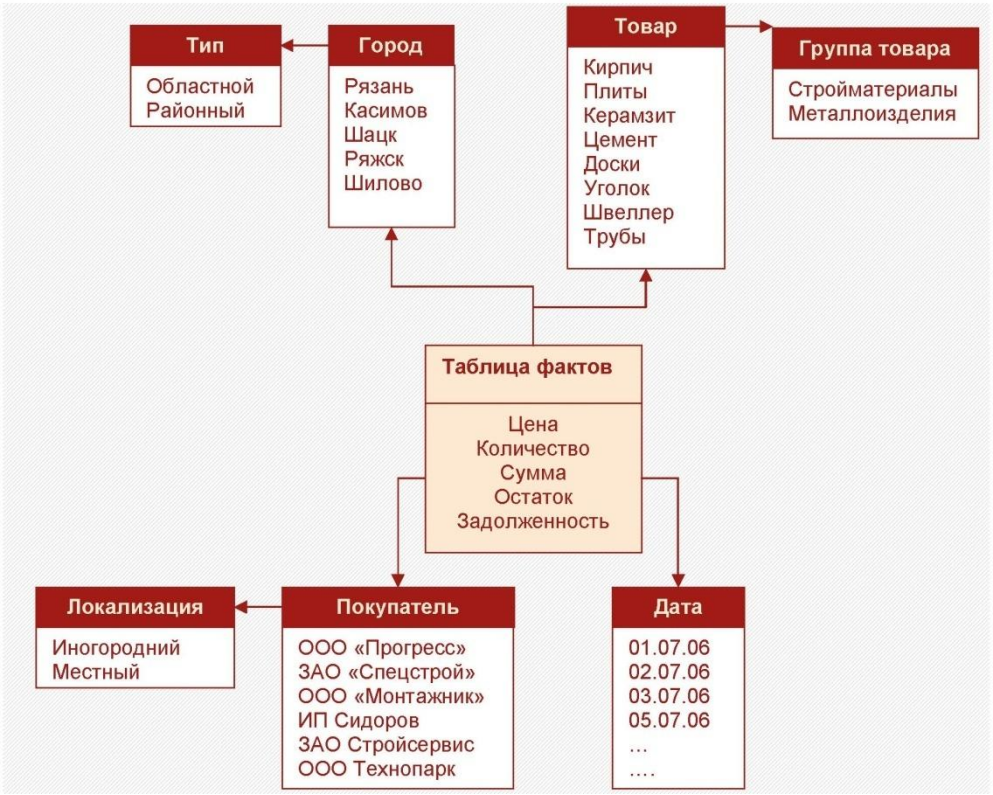


Рис.4.9. Схема построения РХД «снежинка».

Основное функциональное отличие схемы «снежинка» от схемы «звезда» – это возможность работы с иерархическими уровнями, определяющими степень детализации данных. В приведенном примере схема «снежинка» позволяет работать с данными на уровне максимальной детализации, например, с каждым товаром отдельно, или использовать обобщенное представление по группам товаров с соответствующей агрегацией фактов.

Выбор схемы для построения РХД зависит от используемых механизмов сбора и обработки данных. Каждая из схем имеет свои преимущества и недостатки, которые, однако, могут проявляться в большей или меньшей степени в зависимости от особенностей функционирования ХД в целом.

Преимуществами схемы «снежинка» являются:

- она ближе к представлению данных в многомерной модели;
- процедура загрузки из РХД в многомерные структуры более эффективна и проста, поскольку загрузка производится из отдельных таблиц;
- намного ниже вероятность появления ошибок, несоответствия данных;
- большая, по сравнению со схемой «звезда», компактность представления данных, поскольку все значения измерений упоминаются только один раз.

Недостатки схемы «снежинка»:

- достаточно сложная для реализации и понимания структура данных;
- усложненная процедура добавления значений измерений.
- Кроме того, существует ряд технических особенностей, которые могут определить предпочтения разработчиков РХД при выборе схемы его построения.

Основные преимущества РХД:

- практически неограниченный объем хранимых данных;
- поскольку реляционные СУБД лежат в основе построения многих систем оперативной обработки (OLTP), которые обычно являются главными источниками данных для ХД, использование реляционной модели позволяет упростить процедуру загрузки и интеграции данных в хранилище;
- при добавлении новых измерений данных нет необходимости выполнять сложную физическую

реорганизацию хранилища в отличие, например, от многомерных ХД;

- обеспечиваются высокий уровень защиты данных и широкие возможности разграничения прав доступа.

Главный недостаток РХД заключается в том, что при использовании высокого уровня обобщения данных и иерархичности измерений в таких хранилищах начинают «размножаться» таблицы агрегатов. В результате скорость выполнения запросов реляционным хранилищем замедляется. В то же время в многомерных хранилищах, где данные хранятся в виде многомерных кубов, эта проблема практически не возникает, и в большинстве случае удается достичь более высокой скорости выполнения запросов.

Таким образом, выбор реляционной модели при построении ХД целесообразен в следующих случаях.

Значителен объем хранимых данных (многомерные ХД становятся неэффективными).

Иерархия измерений несложная (то есть немного агрегированных данных).

Требуется частое изменение размерности данных. При использовании реляционной модели можно ограничиться добавлением новых таблиц, а для многомерной модели придется выполнять сложную перестройку физической структуры хранилища.

Гибридные хранилища данных. Многомерная и реляционная модели хранилищ данных имеют свои преимущества и недостатки. Например, многомерная модель позволяет быстрее получить ответ на запрос, но не дает возможности эффективно управлять такимиже большими объемами данных, как реляционная модель. Логично было бы

использовать такую модель ХД, которая представляла бы собой комбинацию реляционной и многомерной моделей и позволяла бы сочетать высокую производительность, характерную для многомерной модели, и возможность хранить сколь угодно большие массивы данных, присущую реляционной модели. Такая модель, сочетающая в себе принципы реляционной и многомерной моделей, получила название **гибридной**, или HOLAP (Hybrid OLAP). Хранилища данных, построенные на основе HOLAP, называются гибридными хранилищами данных (ГХД) (рис.4.10).

Рис.4.10 Гибридное ХД.

Главным принципом построения ГХД является то, что детализированные данные хранятся в реляционной структуре (ROLAP), которая позволяет хранить большие объемы

данных, а агрегированные данные – в многомерной (MOLAP), которая позволяет увеличить скорость выполнения запросов (поскольку при выполнении аналитических запросов уже не требуется вычислять агрегаты).

Пример. В супермаркете, ежедневно обслуживающем десятки тысяч покупателей, установлена регистрирующая OLTP-система. При этом максимальному уровню детализации регистрируемых данных соответствует покупка по одному чеку, в котором указываются общая сумма покупки, наименования или коды приобретенных товаров и стоимость каждого товара. Оперативная информация, состоящая из детализированных данных, консолидируется в реляционной структуре ХД. С точки зрения анализа представляют интерес обобщенные данные, например, по группам товаров, отделам или некоторым интервалам дат. Поэтому исходные детализированные данные агрегируются, и вычисленные агрегаты сохраняются в многомерной структуре гибридного ХД.

Если данные из OLTP-системы имеют большой объем (несколько десятков тысяч записей в день и более) и высокую степень детализации, а для анализа используются в основном обобщенные данные, гибридная архитектура хранилища оказывается наиболее подходящей.

Преимущества гибридного хранилища данных.

- Хранение данных в реляционной структуре делает их в большей степени системно-независимыми, что особенно важно при использовании в управлении предприятием экономической информации (показателей).
- Реляционная структура формирует устойчивые и непротиворечивые опорные точки для многомерного хранилища.

- Поскольку реляционное хранилище поддерживает актуальность и корректность данных, оно обеспечивает очень надежный транспортный уровень для доставки информации в многомерное хранилище.

Недостатком гибридной модели является усложнение администрирования ХД из-за более сложного регламента его пополнения, поскольку при этом необходимо согласовывать изменения в реляционной и многомерной структурах.

Виртуальные хранилища данных. Неизбежной проблемой при использовании хранилищ данных в корпоративных аналитических системах является избыточность. Она снижает эффективность использования дискового пространства и оперативной памяти компьютерной системы, а при очень больших объемах хранящейся и обрабатываемой информации может вызвать снижение производительности, возрастание времени ожидания отклика на запрос и даже привести к полной неработоспособности системы. Избыточность в той или иной степени характерна как для реляционных, так и для многомерных хранилищ.

Ситуация усугубляется еще и тем, что ХД хранят историческую информацию и реализуют принцип неизменчивости данных. То есть в отличие от обычных систем оперативной обработки (OLTP-систем), где хранятся лишь актуальные данные, а данные, утратившие актуальность, уничтожаются, ХД могут только пополняться новыми данными, а удаление исторических данных не производится. Кроме того, часто требуется хранить большие объемы агрегированных данных. В совокупности эти факторы могут привести к «взрывному» росту объемов ХД.

Преодолеть проблему избыточности и даже свести ее к нулю можно путем использования виртуальных хранилищ данных (ВХД). В основе концепции виртуального ХД лежит принцип, в соответствии с которым данные из локальных источников, внешнего окружения, баз данных и учетных систем не консолидируются в единое ХД физически, а извлекаются, преобразуются и интегрируются непосредственно при выполнении запроса в оперативной памяти ПК. Фактически запросы адресуются непосредственно к источникам данных.

Виртуальным хранилищем данных – это система, которая работает с разрозненными источниками данных и эмулирует работу обычного хранилища данных, извлекая, преобразуя и интегрируя данные непосредственно в процессе выполнения запроса.

При работе с ВХД пользователь, можно сказать, имеет дело с «иллюзией» хранилища данных (рис.4.11). Виртуальность предполагает, что ВХД существует только до тех пор, пока работает соответствующее приложение. Как только оно завершает работу, виртуальное хранилище прекращает существование.

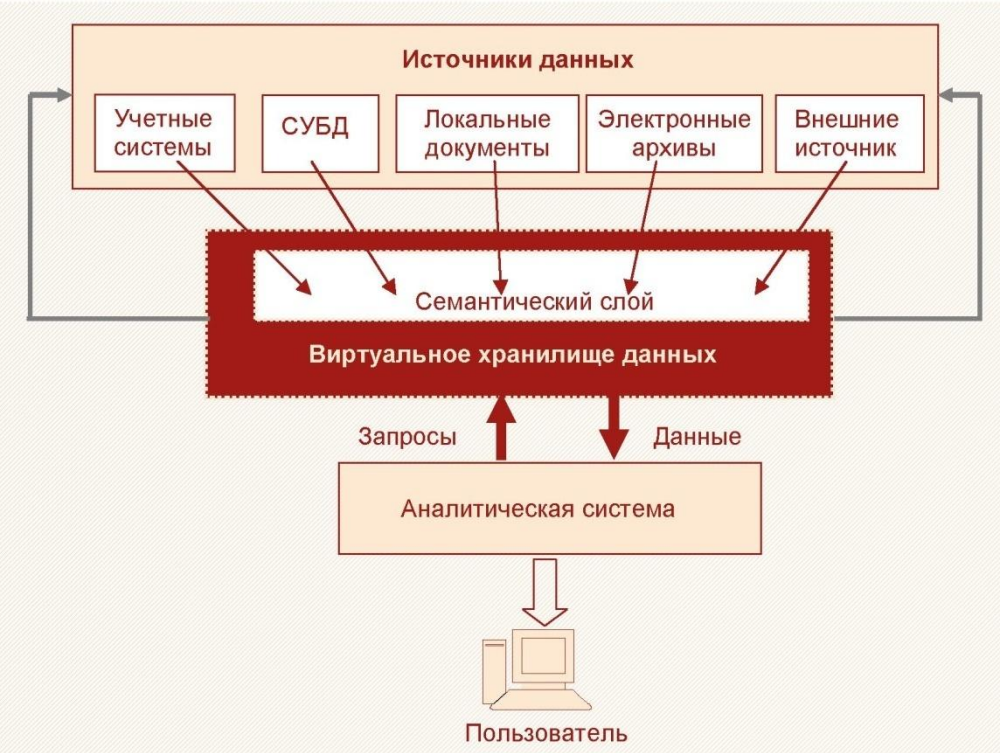


Рис.4.11 Виртуальное ХД.

Преимущества виртуального хранилища данных.

Минимизируется объем требуемой дисковой и оперативной памяти, поскольку отсутствует необходимость хранения исторических данных и многочисленных агрегированных данных для различных уровней обобщения информации.

Наличие в ВХД развитого семантического слоя позволяет аналитику полностью абстрагироваться от проблем, связанных с процессом извлечения данных из разнообразных источников, и сосредоточиться на решении задач анализа данных.

Появляется возможность анализа данных в OLTP-системе сразу после их поступления без ожидания загрузки в хранилище.

Однако концепция ВХД имеет ряд недостатков по сравнению с ХД, где информация консолидируется физически.

Источники данных, информация из которых запрашивается в ВХД, могут оказаться недоступными, если доступ к ним осуществляется по сети или если изменилось место их локализации. Временная недоступность хотя бы одного из источников может привести к невозможности выполнения запроса или к искажению представленной по нему информации.

Отсутствует автоматическая поддержка целостности и непротиворечивости данных, могут быть утеряны отдельные фрагменты документов и т. д.

Данные в источниках хранятся в различных форматах и кодировках, что может привести к ошибкам при их обработке и к искажению информации, полученной в ответ на запрос. Например, если в текстовых файлах с разделителями используются неоднотипные разделители или в файле Excel данные в одном столбце не являются типизированными, это, скорее всего, приведет к неправильной работе аналитических алгоритмов.

Из-за возможной несогласованности моментов пополнения источников данных и из-за отсутствия поддержки в них хронологии по одному и тому же запросу в различные моменты времени могут быть получены отличающиеся данные.

Практически невозможна работа с историческими данными, поскольку в ВХД доступны только те данные, которые находятся в источниках в конкретный момент времени.

Поскольку некоторые типы источников данных не оптимизированы по скорости доступа к ним, извлечение данных из них занимает определенное время, что снижает скорость выполнения запросов виртуальными хранилищами.

Пример. Устоявшейся практикой при использовании ХД является ночная загрузка собранных за день данных из OLTP-систем. Такой регламент позволяет уменьшить нагрузку на OLTP-систему в период ее активного использования. Однако подобная практика не обеспечивает возможности анализировать информацию в течение рабочего дня. Использование ВХД снимает эту проблему, поскольку такое хранилище не требует загрузки данных, а может предоставить актуальную информацию по первому требованию.

Таким образом, применение ВХД оказывается полезным для предприятий, которые не имеют технических средств и квалифицированного персонала для поддержки физических ХД. Особенно велики преимущества ВХД при необходимости анализировать самую свежую информацию. В ВХД отсутствует этап загрузки данных, поэтому временной интервал между появлением информации в OLTP-системе и ее готовностью к анализу данных минимален. При этом следует учитывать, что, поскольку ВХД поддерживает историческую информацию только за период актуальности OLTP-систем, применение такого хранилища оправданно лишь тогда, когда исторические данные для анализа не требуются.

4.3. Средства реализации интеллектуального анализа данных

Программное обеспечение в области интеллектуального анализа данных.

Даже самые мощные технологии извлечения закономерностей и машинного обучения, такие как KDD и Data Mining, не представляют особой ценности без инструментальной поддержки в виде соответствующего *программного обеспечения*. Рынок программных средств продолжает формироваться по сей день, однако в этой области уже можно выделить некоторые стандарты де-факто.

Рынок программного обеспечения KDD и Data Mining делится на несколько сегментов (рис.4.12).

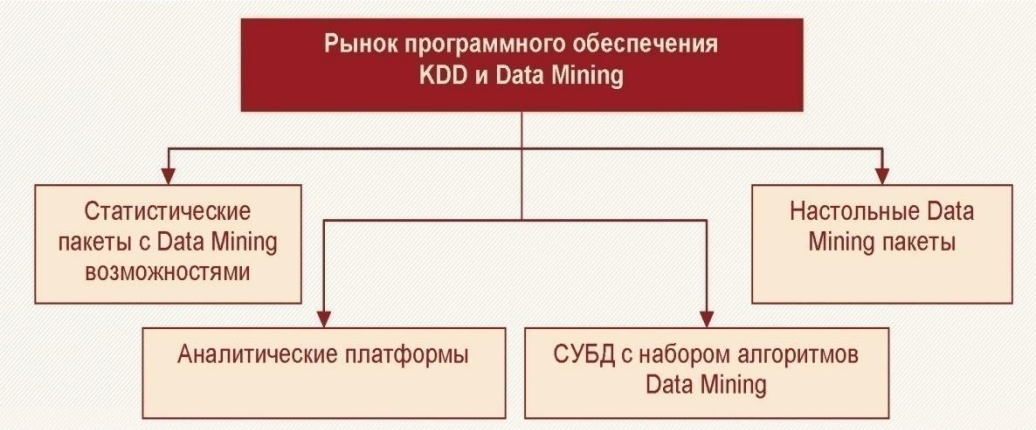


Рис.4.12 Классификация ПО в области Data Mining и KDD.

Статистические пакеты с возможностями Data Mining и *настольные Data Mining пакеты* ориентированы в основном на профессиональных пользователей. Их отличительные особенности:

- слабая интеграция с промышленными источниками данных;
- бедные средства очистки, предобработки и трансформации данных;
- отсутствие гибких возможностей консолидации информации, например, в специализированном хранилище данных;
- конвейерная (поточная) обработка новых данных затруднительна или реализуется встроенными языками программирования и требует высокой квалификации;
- из-за использования пакетов на локальных рабочих станциях обработка больших объемов данных затруднена.

Настольные Data Mining пакеты могут быть ориентированы на решение всех классов задач Data Mining или какого-либо одного, например кластеризации или классификации. Вместе с тем эти пакеты предоставляют богатые возможности в плане алгоритмов, что достаточно для решения исследовательских задач.

Недостатком пакетов является невозможность создания прикладных решений промышленного уровня.

СУБД с элементами Data Mining. Практически все крупные производители систем управления базами данных (СУБД) включают в состав своих продуктов средства для анализа данных, OLAP, а также поддержку хранилищ данных. Эти инструменты как бы «встраиваются» в СУБД. Отличительные особенности СУБД с элементами Data Mining:

- высокая производительность;

- алгоритмы анализа данных по максимуму используют преимущества СУБД;
- жесткая привязка всех технологий анализа к одной СУБД;
- сложность в создании прикладных решений, поскольку работа с СУБД ориентирована на программистов и администраторов баз данных.

Аналитические платформы. В отличие от СУБД с набором алгоритмов Data Mining, аналитические платформы изначально ориентированы на анализ данных и предназначены для создания готовых решений промышленного уровня. Они позволяют наиболее полно реализовать все этапы KDD.

Аналитическая платформа *— специализированное программное решение (или набор решений), которое содержит в себе все инструменты для извлечения закономерностей из «сырых» данных: средства консолидации информации в едином источнике (хранилище данных), извлечения, преобразования, трансформации данных, алгоритмы Data Mining, средства визуализации и распространения результатов среди пользователей, а также возможности «конвейерной» обработки новых данных.*

Отличительные особенности аналитических платформ:

- в аналитической платформе, как правило, всегда присутствуют гибкие и развитые средства консолидации (создание ХД);
- наличие средств импорта данных из широкого спектра различных источников;
- наличие средств интеграции с промышленными источниками данных;

- обязательное наличие инструментов очистки и преобразования структурированных данных;
- хранение данных в едином источнике — в хранилище данных;
- наличие репозитария моделей, описывающих выявленные закономерности, правила и прогнозы;
- широкий спектр алгоритмов Data Mining;
- развитый инструментарий визуализации данных и результатов анализа (моделирования).

На рис. 2 изображена типовая схема системы ИАД на базе аналитической платформы. Вообще говоря, приведенную на рис.4.13 систему можно построить с использованием нескольких программных решений, например, разделить функции извлечения/загрузки, OLAP-отчетности, хранилища данных, Data Mining между различным программным обеспечением. Но чтобы эти отдельные компоненты превратились в полноценную аналитическую систему, необходимо произвести интеграцию между ними на уровне обмена данными, а еще лучше — метаданными.

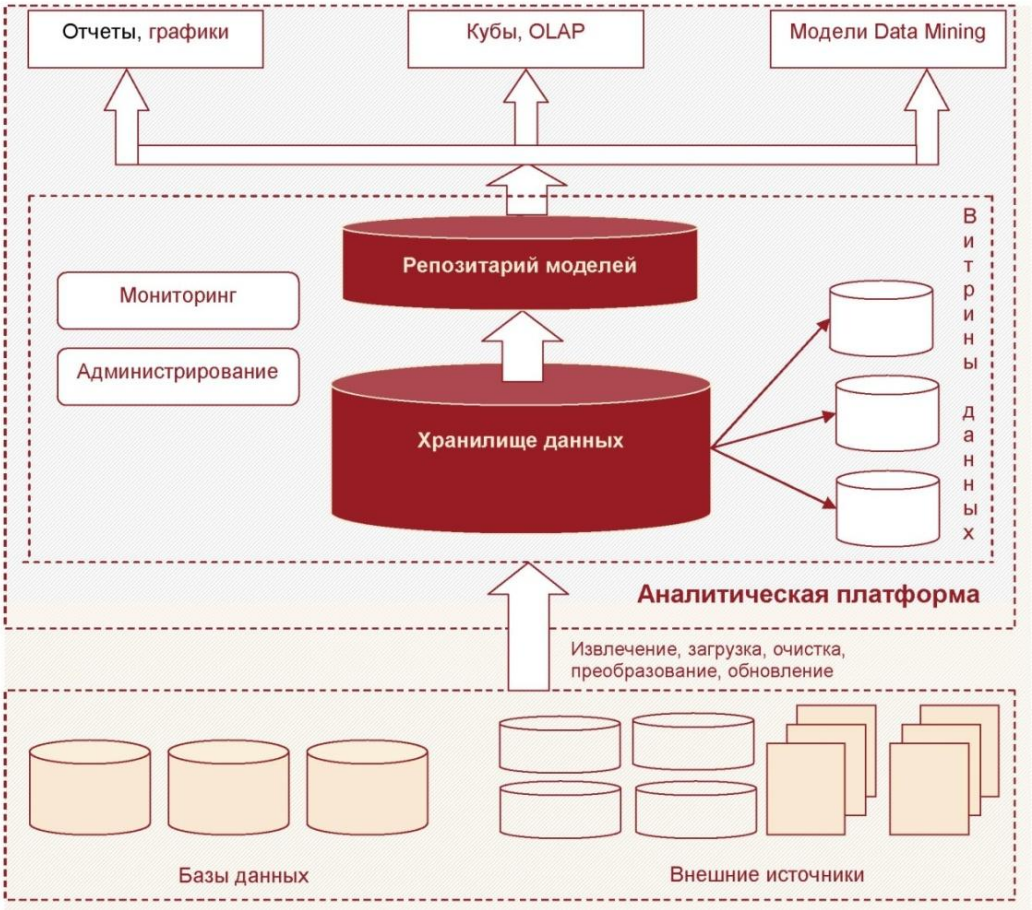


Рис.4.13 Типовая схема системы ИАД на базе аналитической платформы.

Аналитическая платформа Deductor.

Аналитическая платформа Deductor является одним из программных продуктов, реализующих методы ИАД. Аналитическая платформа Deductor разработана отечественной фирмой BaseGroup Labs г. Рязань (www.basegpoup.ru).

Первая версия **Deductor** увидела свет в 2000 г. и с тех пор идет непрерывное развитие платформы. В 2007 г. выпущена пятая по счету версия системы, в 2009 г. – версия 5.2.

Сегодня **Deductor** – это яркий представитель как настольной, так и корпоративной системы анализа данных последнего поколения.

Аналитическая платформа **Deductor** состоит из пяти частей:

Deductor Warehouse – многомерное хранилище данных, аккумулирующее всю необходимую для анализа предметной области информацию. Использование единого хранилища позволяет обеспечить непротиворечивость данных, их централизованное хранение и автоматически обеспечивает всю необходимую поддержку процесса анализа данных. Deductor Warehouse оптимизирован для решения именно аналитических задач, что положительно сказывается на скорости доступа к данным. Предназначен для аналитика.

Deductor Studio – программа, реализующая функции импорта, обработки, визуализации и экспорта данных. Deductor Studio может функционировать и без хранилища данных, получая информацию из любых других источников, но наиболее оптимальным является их совместное использование. В Deductor Studio включен полный набор механизмов, позволяющий получить информацию из произвольного источника данных, провести весь цикл обработки (очистку, трансформацию данных, построение моделей), отобразить полученные результаты наиболее удобным образом (OLAP, диаграммы, деревья…) и экспортировать результаты на сторону. Это полностью соответствует концепции извлечения знаний из баз данных (KDD).

Позволяет пройти все этапы построения прикладного решения. Предназначен для аналитика.

Deductor Viewer – рабочее место конечного пользователя. Позволяет отделить процесс построения моделей от использования уже готовых моделей. Все сложные операции по подготовке моделей выполняются аналитиками-экспертами при помощи Deductor Studio, а Deductor Viewer обеспечивает пользователям простой способ работы с готовыми результатами, скрывает от них все сложности построения моделей и не предъявляет высоких требований к квалификации сотрудников. Является средством тиражирования знаний, т.е. когда построенные аналитиком модели используют пользователи, не владеющие технологиями анализа данных.

Deductor Server – служба, обеспечивающая удаленную аналитическую обработку данных через компьютерную сеть.

Deductor Client – клиент доступа к **Deductor Server**. Обеспечивает доступ к серверу из сторонних приложений и управление его работой.

Существует три типа варианта поставки платформы **Deductor**:

- Enterprise;
- Professional;
- Academic.

В зависимости от типа поставки набор доступных компонентов может различаться. Версия **Enterprise** предназначена для корпоративного использования. В ней присутствуют: серверные компоненты **Deductor Server** и **Deductor Client**, интерфейс доступа к **Deductor** через механизм OLE Automation, традиционное хранилище данных **Deductor Warehouse** на трех СУБД: Firebird, MS SQL, Oracle

и виртуальное хранилище данных **Deductor Virtual Warehouse**.

Версия **Professional** предназначена для небольших компаний и однопользовательской работы. В ней отсутствуют серверные компоненты, поддержка OLE, виртуальное хранилище, а традиционное хранилище данных можно создавать только на СУБД FireBird. Автоматизация выполнения сценариев обработки данных осуществляется только через пакетный режим.

Версии **Professional** и **Enterprise** требуют установки драйверов Guardant для работы с лицензионным ключом.

Версия **Academic** предназначена для образовательных и обучающих целей. Ее функционал аналогичен версии **Professional** за исключением: отсутствует пакетный запуск сценариев, т.е. работа в программе может вестись только в *интерактивном режиме*; отсутствует импорт из промышленных источников данных: 1С, СУБД, файлы MS Excel, Deductor Data File; некоторые другие возможности.

Архитектура системы построена таким образом, что вся работа по анализу данных в Deductor Studio базируется на выполнении следующих действий:

- импорт данных;
- обработка данных;
- визуализация;
- экспорт данных.

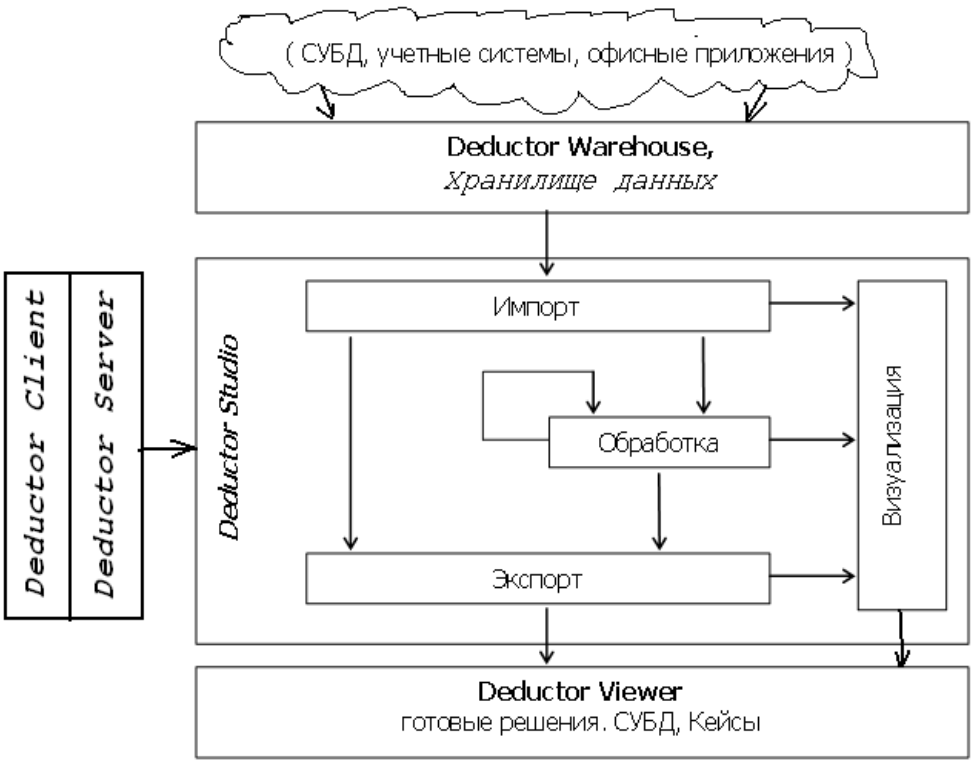


Рис.4.14 Архитектура Deductor.

Процесс построения моделей в Deductor основывается на следующих трех принципах:

1. Использование обработчиков;

2. Использование визуализаторов;

3. Создание сценариев.

Реализованные в Deductor обработчики покрывают основную потребность в анализе данных и создания законченных аналитических решений на базе алгоритмов Data Mining.

Любой набор данных можно *визуализировать* каким-либо доступным способом или несколькими способами, поскольку визуализация помогает интерпретировать построенные модели.

В Deductor предусмотрены следующие способы визуализации данных.

OLAP. Многомерное представление данных. Любые данные, используемые в программе, можно посмотреть в виде кросс-таблицы и кросс-диаграммы.

Таблица. Стандартное табличное представлении с возможностью фильтрации данных.

Диаграмма. График изменения любого показателя.

Гистограмма. График разброса показателей.

Статистика. Статистические показатели набора данных.

Диаграмма рассеяния. График отклонения прогнозируемых при помощи модели значений от реальных. Может быть построена только для непрерывных величин и только после использования механизмов построения модели, например, нейросети или линейной регрессии. Используется для визуальной оценки качества построенной модели.

Таблица сопряженности. Предназначена для оценки результатов классификации вне зависимости от используемой модели. Таблица сопряженности отображает результаты сравнения категориальных значений исходного выходного столбца и категориальных значений рассчитанного выходного столбца. Используется для оценки качества классификации.

«Что-если». Таблица и диаграмма. Позволяют «прогонять» через построенную модель любые интересующие пользователя данные и оценить влияние того или иного фактора на результат.

Обучающая выборка. Набор данных, используемый для построения модели.

Диаграмма прогноза. Применяется после использования метода обработки – Прогнозирование. Прогнозные значения выделяются цветом.

Граф нейросети. Визуальное отображение обученной нейросети. Отображается структура нейронной сети и значения весов;

Дерево решений. Отображение дерева решений, полученного при помощи соответствующего алгоритма.

Дерево правил. Отображение в иерархическом виде (в виде дерева) ассоциативных правил.

Правила. Отображает в текстовом виде правила, полученные при помощи алгоритма построения деревьев решений или поиска ассоциаций.

Карта Кохонена. Отображение карт, построенных при помощи соответствующего алгоритма.

Описание. Текстовое описание параметров импорта/обработки/экспорта в дереве сценариев обработки.

Сценарий представляет собой иерархическую последовательность обработки и визуализации наборов данных. Сценарий всегда начинается с импорта набора данных из произвольного источника. После импорта может следовать произвольное число обработчиков любой степени глубины и вложенности. Каждой операции обработки соответствует отдельный узел дерева, или объект сценария. Любой объект можно визуализировать тем или иным доступным средством. Набор данных служит механизмом, соединяющим все объекты сценария. Можно сказать, что

сценарий – наиболее естественный с точки зрения аналитика способ представления этапов построения модели. Это позволяет быстро создавать модели, обладающие большой гибкостью и расширяемостью, сравнивать несколько моделей. На рис. 1.4 изображен пример сценария.

Интерфейс Deductor Studio состоит из главного окна (рис.4.15), внутри которого располагаются панели сценариев, отчетов, источников данных и результаты моделирования (таблицы, графики, кросс-диаграммы, правила и т.д.).

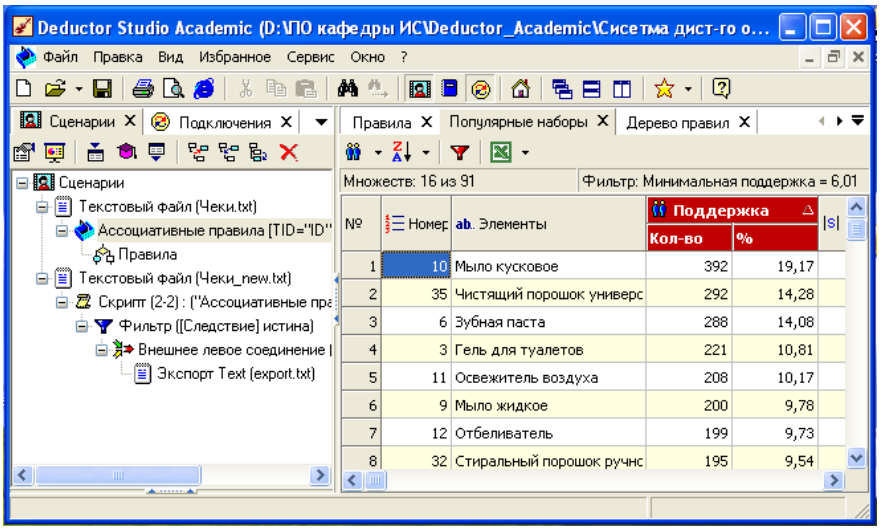


Рис.4.15. Главное окно **Deductor Studio.**

Все сценарии создаются на основе запуска мастеров. В распоряжение аналитика имеется 4 мастера: импорт, экспорт, обработка, отображение.

Мастер импорта предназначен для автоматизации получения данных из любого источника, предусмотренного в системе. На первом шаге мастера импорта открывается список всех предусмотренных в системе типов источников данных. Число шагов мастера импорта, а также набор настраиваемых параметров отличается для разных типов источников.

Мастер обработки предназначен для настройки всех параметров выбранного алгоритма.

Мастер отображений позволяет в пошаговом режиме выбрать и настроить наиболее удобный способ представления данных. В зависимости от обработчика, в результате которого была получена ветвь сценария, список доступных для него видов отображений будет различным. Например, после построения деревьев решений их можно отобразить с помощью визуализаторов «Деревья решений» и «Правила». Эти способы отображения не доступны для других обработчиков.

Мастер экспорта позволяет в пошаговом режиме выполнить экспорт данных в файлы наиболее распространенных форматов.

Глава 5. ИНЖЕНЕРИЯ ЗНАНИЙ

5.1. Основы инженерии знаний

Инженерия знания — достаточно молодое направление искусственного интеллекта, появившееся тогда, когда практические разработчики столкнулись с весьма нетривиальными проблемами трудности «добычи» и формализации знаний. В первых работах по искусственному интеллекту эти факты обычно только постулировались, в дальнейшем начались серьезные исследования по выявлению оптимальных стратегий выявления знаний.

Инженерия знаний - раздел искусственного интеллекта, в рамках которого решаются проблемы, связанные с извлечением знаний, приобретением знаний, представлением знаний и манипулированием знаниями. И.З.

служит основой для создания экспертных систем и других интеллектуальных систем.

Центральным понятием на стадиях получения и структурирования знаний является так называемое *поле знаний.* Поле знаний формируется на третьем этапе разработки ЭС — этапе структурирования.

Поле знаний — это условное неформальное описание основных понятий и взаимосвязей между понятиями предметной области, выявленных из системы знаний эксперта, в виде графа, диаграммы, таблицы или текста.

Стратегии получения знаний.

Существует несколько стратегий получения знаний. Наиболее распространенные (рис.5.1):

- извлечение знаний
- приобретение знаний
- формирование знаний

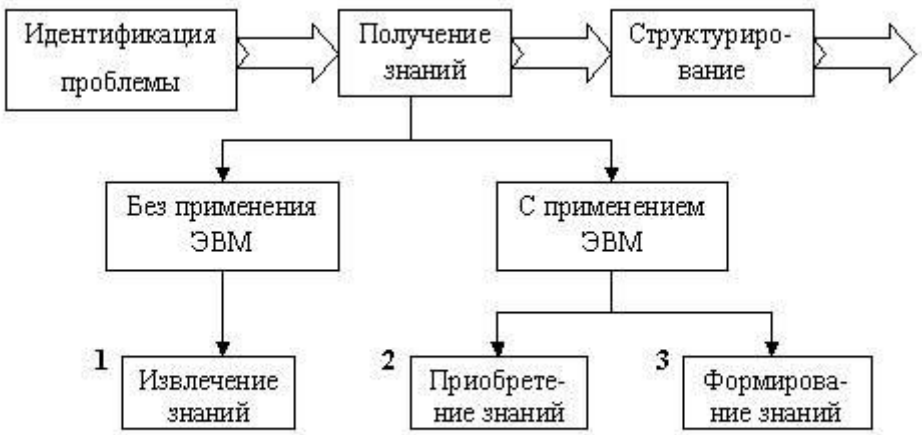


Рис.5.1. Три стратегии получения знаний.

На современном этапе разработки экспертных систем стратегия ***извлечения знаний***, по-видимому, является

наиболее актуальной, поскольку промышленных системприобретения и формированиязнаний наотечественном рынке программных средств практически нет.

Извлечение знаний *- это процедура взаимодействия инженера по знаниям с источником знаний (как правило, это эксперт), в результате которой становятся явными процесс рассуждений экспертовпри принятии решения и структура их представлений о предметной области.*

С точки зрения разработки ЭС извлечение знаний – это получение информации о предметной области от специалистов и выражение ее на языке представления знаний.

Термин *извлечение знаний* касается непосредственногоживого контакта инженера познаниям и источниказнаний. Этот термин как болееемкий и более точно выражающий смысл процедуры переноса компетентности экспертачерез инженера по знаниям в базузнаний экспертной системы.

В настоящее время большинство разработчиков экспертных систем отмечают, что процесс извлечениязнаний остается самым"узким" местом при построении промышленных систем.

Проблемы и особенности извлечения знаний.

Процесс извлечениязнаний - этодлительная и трудоемкая процедура, в которой инженеру по знаниям,вооруженному специальнымизнаниями по когнитивной психологии, системному анализу, математической логике и пр., необходимо воссоздать модель предметной области, которой пользуются эксперты для

принятия решения. Часто начинающие разработчики экспертных систем, желая избежать этой мучительной процедуры, задают вопрос: может ли эксперт сам извлечь из себя знания? Помногим причинамэтонежелательно.

Большая частьзнаний эксперта - эторезультат многочисленных наслоений, ступеней опыта. И часто зная, чтоиз*A* следует *B,* экспертне дает себе отчета, что цепочка его рассуждении была гораздо длиннее.

Как было известно еще древним (вспомним "Диалоги" Платона), мышление диалогично. И поэтому диалог инженера по знаниям и эксперта - наиболее естественная форма "раскручивания" лабиринтовпамяти эксперта, в которых хранятся знания, частью носящие невербальный характер, т.е. выраженныене в форме слов, в форме наглядных образов,например. Именно в процессе объяснения инженеру по знаниям эксперт на эти размытые ассоциативные образы надевает четкие словесные ярлыки,т.е. вербализует знания.

Эксперту гораздо труднее создать модель предметной области вследствие той глубины и необозримости информации, которой он обладает. Многочисленные причинно-следственные связи реальной предметной области образуют сложную систему,из которой выделить "скелет", или главнуюструктуру, иногда доступнее инженеру по знаниям (аналитику), владеющему к тому же системной методологией. Любая модель - это упрощение, а упрощать легче с меньшим знанием деталей.

Приобретением знаний - *процесс (способ) автоматизированного построения базы знаний посредством диалога эксперта и специальной программы (при этом структура знаний заранее закладывается в программу).*

Эта стратегия требует существенной предварительной проработки предметной области. Системы приобретения знаний действительноприобретают готовые фрагментызнаний в соответствии со структурами, заложенными разработчиками систем. Большинство этих инструментальных средств специально ориентировано на конкретные экспертные системы с жестко обозначенной предметной областьюи моделью представления знаний, т.е.не являютсяуниверсальными. Например, система TEIRESIAS, ставшая прародительницей всех инструментариев для приобретения знаний, предназначена для пополнения базы знаний системы MYCIN или ее дочерних ветвей,построенных на "оболочке" EMYCIN в областимедицинской диагностики с использованием продукционной модели представления знаний.

Формирование знаний *– процесс анализа данных и выявление скрытых закономерностей с использованием специального математического аппарата и программных средств.*

Термин формирование знаний традиционно закрепился за чрезвычайно перспективнойи активно развивающейся областью инженериизнаний – интеллектуальным анализом данных. Данное направление (см. Главу 4) занимаетсяразработкой моделей, методов и алгоритмов анализа для получения знаний из накопленных данных.

Теоретические аспекты получения знаний.

Чтобы разобраться в природе извлечениязнаний, выделим три основных аспектаэтойпроцедуры (рис.5.2): психологический, лингвистический, гносеологический.

A = {Al, A2, A3} = {психологический, лингвистический, гносеологический}.

Рис.5.2. Теоретические аспекты инженерии знаний

Психологический аспект. Из трех аспектов извлечения знаний психологический является ведущим, поскольку он определяет успешность и эффективность взаимодействия инженера по знаниям (аналитика) с основным источником знаний — экспертом-профессионалом. Психологический аспект выделяется еще и потому, что извлечение знаний происходит чаще всего в процессе непосредственного общения разработчиков системы. А в общении психология является доминантной.

Общение, или коммуникация (от лат. communicatio — связь), — это междисциплинарное понятие, обозначающее все формы непосредственных контактов между людьми — отдружеских до деловых. Оно широко исследуется в психологии, философии, социологии, этологии, лингвистике, семиотике и других науках. Существует несколько десятков теорий общения, и единственное, в чем сходятся все авторы, — это сложность, многоплановость процедуры общения. Подчеркивается, что общение — не просто однонаправленный

процесс передачи сообщений и не двухтактный обмен порциями сведений, а нерасчлененный процесс циркуляции информации, то есть совместный поиск истины.

Таким образом, общение есть процесс выработки новой информации, общей для общающихся людей и рождающей их общность. И хотя общение — первый вид деятельности, которым овладевает человек в онтогенезе, по-настоящему владеют культурой и наукой общения единицы.

Можно выделить четыре основных уровня общения.

Уровень манипулирования, когда один субъект рассматривает другого как средство или помеху по отношению к проекту своей деятельности.

Уровень «рефлексивной игры», когда в процессе своей деятельности человек учитывает «контрпроект» другого субъекта, но не признает за ним самоценность и стремится к «выигрышу», к реализации своего проекта.

Уровень правового общения, когда субъекты признают право на существование проектов деятельности друг друга и пытаются согласовать их хотя бы внешне.

Уровень нравственного общения, когда субъекты внутренне принимают общий проект взаимной деятельности.

Стремление и умение общаться на высшем, четвертом, уровне может характеризовать степень профессионализма инженера по знаниям. Извлечение знаний — это особый вид общения, который можно отнести к духовно-информационному типу. Известно, что общение делится на материально-практическое; духовно-информационное; практически-духовное. При этом информационный аспект

общения для инженера по знаниям с прагматической точки зрения важнейший.

Известно, что потери информации при разговорном общении велики. В связи с этим рассмотрим проблему увеличения информативности общения аналитика и эксперта за счет использования психологических знаний. Можно выделить такие структурные компоненты модели общения при извлечении знаний:

- участники общения (партнеры);
- средства общения (процедура);
- предмет общения (знания).

В соответствии с этой структурой выделяют три «слоя» психологических проблем, возникающих при извлечении знаний (рис.5.3):

A1 ={S11, S12, S13 } = {контактный, процедурный, когнитивный}.

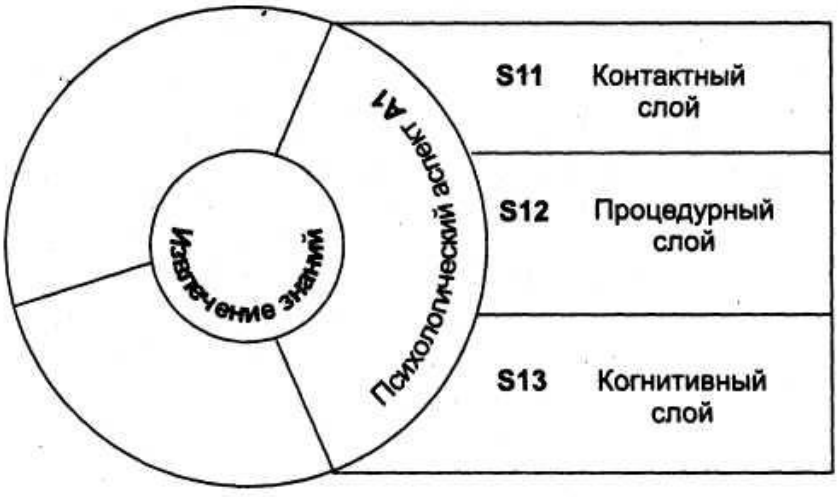


Рис. 5.3. Психологический аспект извлечения знаний

Контактный слой. Он связан с атмосферой и уровнем общения в коллективе разработчиков ЭС. Практически все психологи отмечают, что на любой коллективный процесс

влияет атмосфера, возникающая в группе участников. Существуют эксперименты, результаты которых неоспоримо говорят, что дружеская атмосфера в коллективе больше влияет на результат, чем индивидуальные способности отдельных членов группы. Особенно важно, чтобы в коллективе разработчиков складывались кооперативные, а не конкурентные отношения. Для кооперации характерна атмосфера сотрудничества, взаимопомощи, заинтересованности в успехах друг друга, т.е. уровень нравственного общения, а для отношений конкурентного типа - атмосфера индивидуализма и межличностного соперничества(более низкий уровень общения).

К сожалению, прогнозировать совместимость в общении со 100%-ной гарантией невозможно. Однако можно выделить ряд черт личности, характера и других особенностей участников общения, несомненно, оказывающих влияние на эффективность процедуры. Знание этих психологических закономерностей составляет часть багажа психологической культуры, которым должен обладать инженер по знаниям для успешного проведения стадии извлечения знаний:

- доброжелательность и дружелюбие;
- чувство юмора;
- хорошая память и внимание;
- наблюдательность;
- воображение и впечатлительность;
- большая собранность и настойчивость;
- общительность и находчивость;
- аналитичность;
- располагающая внешность и манера одеваться;
- уверенность в себе.

*Процедурный слой*касается проведения самой процедуры извлечения знаний. Инженер по знаниям, успешно овладевший наукой доверия и взаимопонимания с экспертом (контактныйслой), должен еще уметь воспользоватьсяблагоприятным воздействием этой атмосферы. Здесь мало проницательности и обаяния, полезного для решения проблемы контакта, тут необходимы профессиональные знания.

Остановимся на общих закономерностях проведения процедуры.

Беседу с экспертом лучше всего проводить в небольшом помещении tete-a-tete. Освещение, тепло, уют влияют непосредственно на настроение. Чай или кофе создадут дружескую атмосферу. Американский психолог И.Атватер считает, что для делового общения наиболее благоприятная дистанция от 1,2 до 3 м. Минимальным "комфортным" расстоянием можно считать 0,7 - 0,8 м.

Реконструкция собственных рассуждений - нелегкий труд, и поэтому длительность одного сеанса обычно не превышает 1,5 - 2ч. Эти два часа лучше выбрать в первой половине дня (например, с 10 до 12 ч). Известно, что взаимная утомляемость партнеров при беседе наступает обычно через 20 - 25 мин, поэтому в сеансе нужны паузы.

Любой инженер по знаниям имеет свою уникальную манеру разговора. Одни говорят быстро, другие медленно; одни громко, другие тихо и т.д. Стиль разговора изменить практически невозможно - он закладывается в человеке в раннем детстве. Однако извлечение знаний - это профессиональный разговор, и на его успешность влияет также длина фраз, которые произносит инженер по знаниям.

Этот факт был установлен американскими учеными - лингвистом Ингве и психологом Миллером. Оказалось, что человек лучше всего воспринимает предложения глубиной (или длиной) 7 плюс-минус 2 слова. Это число (7|2) получило название число Ингве-Миллера. Можно считать его мерой "разговорности" речи.

Необходимость фиксации процедуры извлечения знаний ни у кого не вызывает сомнений. Встает вопрос: в какой форме это делать? Можно предложить три способа протоколирования результатов:

• запись, на бумагу непосредственно по ходу беседы (недостатки - это часто мешает беседе, кроме того, трудно успеть записать все, даже при наличии навыков стенографии);

• магнитофонная запись, помогающая аналитику проанализировать весь ход сеанса и свои ошибки (недостаток - может сковывать эксперта);

• запоминание с последующей записью после беседы (недостаток - годится только для аналитиков с блестящей памятью).

Когнитивный слой(англ. cognition - познание)связан со знанием механизмов, при помощи которых человек познает окружающий мир.

С позиций когнитивной психологии при извлечении знаний желательно:

• не навязывать эксперту ту модель представления, которая ему (аналитику) более понятна и естественна;

• использовать различные методы работы с экспертом исходя из условия, что метод должен подходить к эксперту, как "ключ к замку";

• четко осознавать цель процедуры извлечения или ее главную стратегию, которая может быть определена как выявление основных понятий предметной области и связывающих их отношений;

• чаще рисовать схемы, отображающие рассуждения эксперта. Это связано с образной репрезентацией информации в памяти человека.

Материал, изложенный выше, тесно связан с азами психологической культуры, которая включает понимание и знание себя и других людей; адекватную самооценку и оценку других людей; саморегулирование психического состояния. Овладеть этой культурой легче с помощью специалистов - психологов, психотерапевтов, но можно самостоятельно с помощьюкниг. Кроме этого успешному преодолению психологических неудач способствует овладение основами актерского мастерства и участие в специальных занятиях по социально-психологическому видеотренингу.

Проблемы и трудности инженера по знаниям при извлечении знаний:

- отсутствие контакта между экспертом и инженером по знаниям (из-за психологических особенностей того или другого; ошибок в процедуре; возникновения эффекта "фасада", т.е. желания эксперта "показать себя");
- отсутствие понимания (из-за эффекта "проекции",т.е. переноса взгляда аналитика на взгляды эксперта; или эффекта "порядка", т.е. концентрации внимания в

первую очередь на том, что высказывается вначале, и др.);

- низкая эффективность бесед (слабая мотивация эксперта, т.е. отсутствие у него интереса; или неудачныйтемп беседы; или неподходящая форма вопросов;или неудовлетворительные ответы эксперта).

Лингвистический аспект. Лингвистический (А2) аспект касается исследований языковых проблем, так как язык — это основное средство общения в процессе извлечения знаний. Сразу же следует оговорить, что поскольку тема данной книги ограничена изложением теории и технологии инженерии знаний, то область разработки естественно-языковых интерфейсов и весь спектр проблем, связанных с ней — лексических, синтаксических, семантических, прагматических и т. д. — не рассматривается.

В инженерии знаний можно выделить три слоя лингвистических проблем (рис.5.4):

А2 = {S21, $22, S23} = {«общий код», понятийная структура, словарь пользователя}.

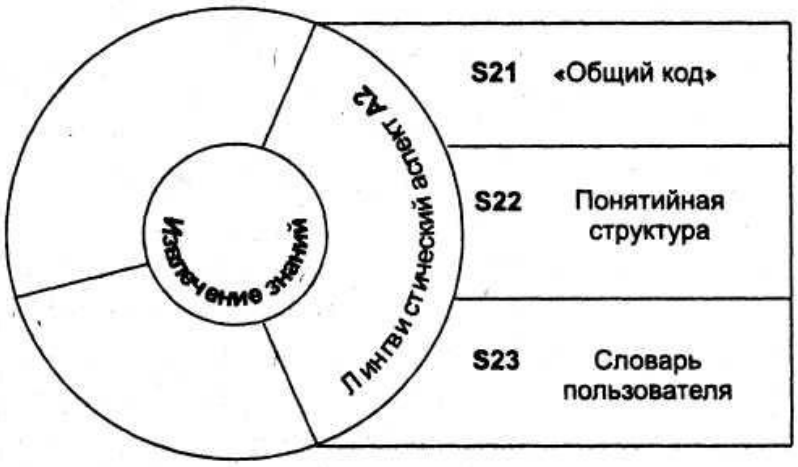


Рис. 5.4. Лингвистический аспект извлечения знаний.

*Проблема «Общего кода»***.** Большинство психологов и лингвистов считают, что язык - это основное средство мышления наряду с другими знаковыми системами "внутреннего пользования". Языки, на которых говорят и размышляют аналитик и эксперт, могут существенно отличаться. Различие языков и обусловливает «языковый барьер» или «языковые ножницы» в общении инженера по знаниям и эксперта.

*Язык инженера по знаниям (аналитика)*состоит из трех компонентов:

- общенаучной терминологии из его «теоретического багажа»;
- терминов предметной области, которые он почерпнул из специальной литературы в период подготовки;
- бытового разговорного языка, которым пользуется аналитик.

Язык эксперта включает:

- общенаучную терминологию;
- специальную терминологию, принятую в предметной области;
- бытовой язык;
- неологизмы, созданные экспертом за время работы, то есть его профессиональный жаргон.

Если считать, что бытовой и общенаучный языки у двух участников общения примерно совпадают, то некоторый общий язык, или код, который необходимо выработать партнерам для успешного взаимодействия, будет включать еще специальные терминологии инженера по знаниям и эксперта. В дальнейшем этот общий код преобразуется в

некоторую понятийную (семантическую) сеть, которая является прообразом поля знаний предметной области.

Выработка общего кода начинается с выписывания аналитиком всех терминов, употребляемых экспертом, и уточнения их смысла. Фактически это составление словаря предметной области. Затем следуют группировка терминов и выбор синонимов (слов, означающих одно и то же). Разработка общего кода заканчивается составлением словаря терминов предметной области с предварительной группировкой их по смыслу, т.е. по понятийной близости (это уже первый шаг структурирования знаний).

Важной является проблема неоднозначности интерпретации терминов двумя специалистами. В семиотике, науке о знаковых системах, проблема интерпретации является одной из центральных. Интерпретация связывает "знак" и "означаемый предмет". Только в интерпретации знак получает смысл. Так, например, термин "прибор Х" для эксперта означают некоторую конкретную схему, которая соответствует схеме оригинала прибора, а в голове начинающего аналитика слова "прибор Х" вызывают пустой образ или некоторый "черный ящик" с ручками.

*Понятийная структура.*Большинство специалистов по искусственному интеллекту и когнитивной психологии считают, что основная особенность естественного интеллекта и памяти в частности - это связанность всех понятий в некоторую сеть. Поэтому для разработки базы знаний нужен не словарь, а энциклопедия, в которой все термины объяснены в словарных статьях со ссылками на другие термины.

Таким образом, лингвистическая работа инженера по знаниям на данном слое проблем заключается в построении

таких связанных фрагментов с помощью "сшивания" терминов. При тщательной работе аналитика и эксперта в понятийных структурах начинает проглядывать иерархия понятий, что в общем согласуется с результатами когнитивной психологии.

Иерархия понятий - это глобальная схема, которая может быть в основе концептуального анализа структуры знаний любой предметной области.

Следует подчеркнуть, что работа по составлению словаря и понятийной структуры требует лингвистического "чутья", легкости манипулирования терминами и богатого словарного запаса инженера по знаниям, так как зачастую аналитик вынужден самостоятельно разрабатывать словарь признаков Чем богаче и выразительнее общий код, тем полнее база знаний.

Аналитик вынужден все время помнить о трудности передачи образов и представлений в вербальной форме. Часто инженеру по знаниям приходится подсказывать слова и выражения эксперту.

*Словарь пользователя.*Лингвистические результаты, соотнесенные со слоями общего кода и понятийной структуры, направлены на создание адекватной базы знаний. Однако не следует забывать, что профессиональный уровень конечного пользователя может не позволить ему применить специальный язык предметной области в полном объеме. Для разработки пользовательского интерфейса необходима дополнительная доработка словаря общего кода с поправкой на доступность и "прозрачность" системы.

В заключение перечислим характерные лингвистические неудачи, подстерегающие начинающего инженера по знаниям:

- разговор на разных языках (из-за слабой подготовки инженера по знаниям);
- несоотнесение с контекстом и неадекватная интерпретация терминов (из-за отсутствия обратной связи, т.е. слишком независимой работы инженера по знаниям);
- отсутствие отличий между общим кодом и языком пользователя (не учтены различия в уровне знаний эксперта и пользователя).

Гносеологический аспект связан с теорией познания, или теорией отражения действительности в сознании человека.

Инженерия знаний как наука, если можно так выразиться, дважды гносеологична - действительность сначала отражается в сознании эксперта, а затем деятельность и опыт эксперта интерпретируются сознанием инженера по знаниям, что служит уже основой для построения третьей интерпретации - поля знаний экспертной системы. Процесс познания в сущности направлен на создание внутреннего представления окружающего мира в сознании человека.

В процессе извлечения знаний аналитика в основном интересует компонент знания, связанный с неканоническими индивидуальными знаниями экспертов, поскольку предметные области именно с таким типом знаний считаются наиболее восприимчивыми к внедрению экспертных систем. Эти области обычно называют эмпирическими, так как в них накоплен большой объем отдельных эмпирических фактов и наблюдений, в то время как их теоретическое обобщение - вопрос будущего.

Познание всегда связано с созданием новых понятий и теории. Интересно, что часто эксперт как бы "на ходу" порождает новые знания, прямо в контексте беседы с аналитиком. Такая генерация знаний может быть полезна и самому эксперту, который до того момента мог не осознавать ряд соотношений и закономерностей предметной области. Аналитику, который является "повитухой" при рождении нового знания, может помочь тут и инструментарий системной методологии, позволяющий использовать известные принципы логики научных исследований, понятийной иерархии науки. Эта методология заставляет его за частным увидеть общее, т.е. строить *гносеологические цепочки*:

ФАКТ=> ОБОБЩЕННЫЙ ФАКТ=> ЭМПИРИЧЕСКИЙ ЗАКОН=> ТЕОРЕТИЧЕСКИЙ ЗАКОН.

Не всегда инженер по знаниям дойдет до последнего звена этой цепочки, но уже само стремление к движению бывает чрезвычайно плодотворным. Такой подход полностью согласуется со структурой самого знания, которое имеет два уровня:

- эмпирический (наблюдения, явления);
- теоретический (законы, абстракции, обобщения).

Методологическая структура познания может быть представлена как последовательность этапов (рис.5.5), которые рассмотрим с позиций инженера по знаниям.

Описание и обобщение фактов. Это как бы "сухой остаток" бесед аналитика с экспертом. Тщательность и полнота ведения протоколов во время процесса извлечения и пунктуальная "домашняя работа" над ними - вот залог продуктивного первого этапа познания.

На практике оказывается трудным придерживаться принципов объективности и системности, описанных выше. Чаще всего на этом этапе факты просто собирают и как бы бросают в "общий мешок"; опытный инженер по знаниям часто сразу пытается найти "полочку" или "ящичек" для каждого факта, тем самым подспудно готовясь к этапу концептуализации.

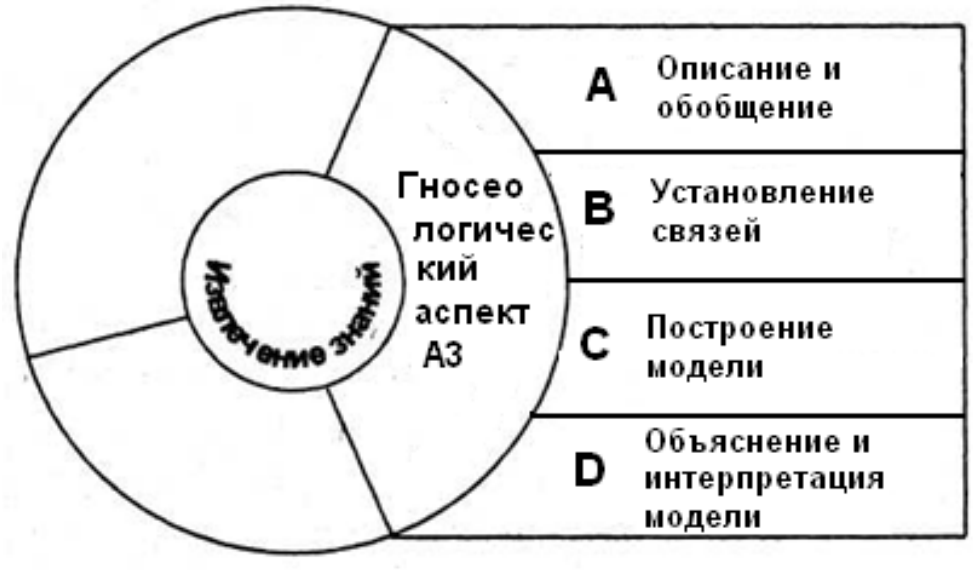


Рис. 5.5. Структура познания.

Установление связей и закономерностей. В голове эксперта связи установлены, хотя часто и неявно; задача инженера - выявить каркас умозаключений эксперта. Реконструируя рассуждения эксперта, инженер по знаниям может опираться на две наиболее популярные теории мышления - логическую и ассоциативную. При этом, если логическая теория благодаря горячим поклонникам в лице математиков широко цитируется и всячески эксплуатируется в работах по искусственному интеллекту, то вторая, ассоциативная, менее известна и популярна, хотя имеет также древние корни. Красота и стройность логической

теории не должны заслонять печального факта, что человек редко мыслит в категориях математической логики .

Ассоциативная теория представляет мышление как цепочку идей, связанных общими понятиями. Основными операциями такого мышления являются ассоциации, приобретенные на основе различных связей; припоминание прошлого опыта; пробы и ошибки со случайными успехами; привычные ("автоматические") реакции и пр.

Построение модели. Для построения модели, отражающей представление субъекта о предметной области, необходим специализированный язык, с помощью которого можно описывать и конструировать те идеализированные модели мира, которые возникают в процессе мышления. Язык этот создается постепенно с помощью понятийного аппарата, принятого в соответствующей предметной области, а также формально-знаковых средств математики и логики. Для эмпирических, предметных областей такой язык пока не разработан, и поле знаний, которое полуформализованным способом опишет аналитик, может быть первым шагом к созданию такого языка.

Объяснение и интерпретация моделей. Этот завершающий этап структуры познания является одновременно и частичным критерием истинности полученного знания. Если выявленная система знаний эксперта полна и объективна, то на ее основании можно делать прогнозы и объяснять любые явления из данной предметной области. Обычно базы знаний экспертных систем страдают фрагментарностью и модульностью (несвязанностью) компонентов. Все это не позволяет создавать действительно интеллектуальные системы, которые, равняясь на человека, могли бы предсказывать

новые закономерности и объяснять случаи, не указанные в явном виде в базе. Исключением тут являются системы формирования знаний, которые ориентированы на генерацию новых знаний и "предсказание".

В заключение перечислим наиболее часто встречающиеся неудачи, связанные с гносеологическими проблемами инженерии знаний :

- обрывочность, фрагментарность знаний (из-за нарушений принципа системности или ошибок в выборе фокуса внимания);
- противоречивость знаний (из-за естественной противоречивости природы и общества неполноты извлеченных знаний, некомпетентности эксперта);
- ошибочная классификация (из-за неправильного определения числа классов или неточного описания класса);
- ошибочный уровень обобщения (из-за чрезмерной детализации или обобщенности классов объектов).

5.2. Классификация методов извлечения знаний.

Изложим классификацию методов извлечения знаний (рис. 10) , что позволит инженерам по знаниям, в зависимости от конкретной задачи и ситуации, выбрать конкретный метод. Из предложенной схемы классификации видно, что основной принцип деления связан с источником знаний. *Коммуникативные методы* охватывают все виды контактов с живым источником знаний - экспертом, а *текстологические касаются* методов извлечения знаний из документов (методик, пособий, руководств) и специальной литературы (статей, монографий, учебников).

Разделение этих групп методов на верхнем уровне классификации не означает их антагонистичности, обычно инженер по знаниям комбинирует различные методы, например, сначала изучает литературу, затем беседует с экспертами, или наоборот.

Рис.5.6. Классификация методов извлечения знаний.

В свою очередь, коммуникативные методы можно также разделить на две группы: активные и пассивные. Пассивные методы подразумевают, что ведущая роль в процедуре извлечения знаний как бы передается эксперту, а инженер по знаниям только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным самостоятельно рассказать в форме лекции. В активных методах, напротив, инициатива полностью в руках инженера по знаниям, который активно контактирует с экспертом различными

способами - в играх, диалогах, беседах за "круглым столом" и т.д.

Пассивные методы на первый взгляд достаточно просты, но на самом деле требуют от инженера по знаниям умения четко анализировать "поток мыслей" эксперта и выявлять в нем значимые фрагменты знаний. Отсутствие обратной связи (пассивность инженера по знаниям) значительно ослабляет эффективность этих методов, чем и объясняется их обычно вспомогательная роль при активных методах.

Активные методы можно разделить на две группы в зависимости от числа экспертов, отдающих свои знания. Если их число больше одного, то целесообразно помимо серии индивидуальных контактов с каждым применять и методы групповых обсуждений предметной области. Такие групповые методы обычно активизируют мышление участников дискуссий и позволяют выявлять весьма нетривиальные аспекты их знаний. В свою очередь, индивидуальные методы на сегодняшний день остаются ведущими, поскольку столь деликатная процедура, как "отъем знаний", не терпит лишних свидетелей.

Отдельно следует сказать об играх. Игровые методы сейчас широко используются и социологии, экономике, менеджменте, педагогике для подготовки руководителей, учителей, врачей и других специалистов. Игра - это особая форма деятельности и творчества, где человек раскрепощается и чувствует себя намного свободнее, чем в обычной трудовой деятельности.

5.3. Коммуникативные методы извлечения знаний.

Пассивные коммуникативные методы.

Название "пассивные" не должно вызывать иллюзий, поскольку этот термин введен как противовес к "активным" методам. В реальности же пассивные методы требуют от инженера по знаниям не меньшей отдачи, чем такие активные методы, как игры и диалог. Согласно классификации к этой группе относятся.

Наблюдения. В процессе наблюдений инженер по знаниям находится непосредственно рядом с экспертом во время его профессиональной деятельности или имитации этой деятельности; При подготовке к сеансу извлечения эксперту необходимо объяснить цель наблюдений и попросить максимально комментировать свои действия.

Во время сеанса аналитик записывает все действия эксперта, его реплики и объяснения. Может быть сделана и видеозапись в реальном масштабе времени. Непременное условие этого метода - невмешательство аналитика в работу эксперта хотя бы на первых порах.

Существуют две разновидности проведения наблюдений:наблюдение за реальным процессом и наблюдение за имитацией процесса.

Обычно используют обе разновидности. Сеансы наблюдений могут потребовать от инженера по знаниям:

- овладения техникой стенографии для фиксации действий эксперта в реальном масштабе времени;

- ознакомления с методиками хронометрирования для четкого структурирования производственного процесса по времени,
- развития навыков "чтения по глазам", наблюдательности к жестам, мимике и другим невербальным компонентам общения;
- серьезного предварительного знакомства с предметной областью, так как из-за отсутствия "обратной связи" иногда многое непонятно в действиях экспертов. Протоколы наблюдений после сеансов в ходе домашней работы тщательно расшифровываются, а затем обсуждаются с экспертом. Таким образом, наблюдения - один из наиболее распространенных методов извлечения знаний на начальных этапах разработки. Обычно он применяется не самостоятельно, а в совокупности с другими методами.

Анализ протоколов "мыслей вслух". Протоколирование "мыслей вслух" отличается от наблюдений тем, что эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено, т.е. продемонстрировать всю цепочку своих рассуждений. Во время рассуждения эксперта все его слова протоколируются инженером по знаниям: при этом полезно отмечать даже паузы и междометия. Вопрос об использовании для этой цели магнитофонов и диктофонов является дискуссионным, поскольку магнитофон иногда действует на эксперта парализующе, разрушая атмосферу доверительности, которая может и должна возникать при непосредственном общении.

Основной трудностью при протоколировании "мыслей вслух" является принципиальная сложность для любого человека объяснить, как он думает. При этом существуют экспериментальные психологические доказательства, что

люди не всегда в состоянии достоверно описать мыслительные процессы. Кроме того, часть знаний, хранящихся в невербальной форме (например, различные процедурные знания типа "как завязывать шнурки"), вообще слабо кореллируют с их словесным описанием.

Автор теории фреймов М.Минский считает, что "только как исключение, а не как правило человек может объяснить то, что он думает".

Расшифровка полученных протоколов производится инженером по знаниям самостоятельно с коррекциями на следующих сеансах извлечения знаний. Удачно проведенное протоколирование "мыслей вслух" является одним из наиболее эффективных методов извлечения, поскольку в нем эксперт может проявить себя максимально ярко, он ничем не скован, он как бы свободно парит в потоке своих умозаключений и рассуждений. Для большого числа экспертов это самый приятный и лестный способ извлечения знаний.

Лекции. Лекция - самый старый способ передачи знаний. Лекторское искусство издревле высоко ценилось во всех областях науки и культуры. Но нас интересует не столько способность к подготовке и чтению лекций, сколько способность эту лекцию слушать, конспектировать и усваивать. Уже говорилось, что экспертов чаще всего не выбирают, и поэтому инженер по знаниям учить эксперта чтению лекций не сможет. Но если у эксперта опыт преподавателя (например, профессора клиники или опытного руководителя производства), то можно воспользоваться таким концентрированным фрагментом знаний, как лекция.

В лекции эксперту предоставлено много степеней свободы для самовыражения; при этом необходимо сформулировать

эксперту тему и задачу лекции. При такой постановке опытный лектор может заранее структурировать свои знания, ход рассуждении. От инженера по знаниям в этой ситуации требуется лишь грамотно законспектировать лекцию и в конце задать необходимые вопросы.

Продолжительность лекции стандартная - от 40 до 50 мин и через 5-10 мин - еще столько же. Курс - от двух до пяти лекций.

Метод извлечения знаний в форме лекций, как и все пассивные методы, используют в начале разработки как эффективный способ быстрого погружения инженера по знаниям в предметную область.

Известны основные моменты, позволяющие слушателю - инженеру по знаниям качественно использовать данный метод для извлечения знаний:

- к лекции подготовьтесь, т.е. познакомьтесь с предметной областью;
- слушайте с максимальным вниманием. Для этого: устраните мешающие факторы (скрип двери, шорохи и т.д.); удобно устройтесь; поменьше двигайтесь;
- учитесь отдыхать во время слушания (например, когда лектор приводит цифры, которые найдете в справочнике);
- слушайте одновременно и лектора, и самого себя (параллельно рассуждениям лектора по ассоциации возникают и собственные мысли);
- слушайте и одновременно записывайте, но записывайте текст сокращенно, используя условные значки (для всего этого достаточно только установить для себя ряд условных значков и ими неизменно пользоваться);
- расшифруйте записи лекции в тот же день;

- не спорьте с лектором во время слушания.

Активные коммуникативные методы.

Различают активные групповые методы и активные индивидуальные методы.

***Активные групповые методы*.** Основное достоинство групповых методов - это возможность одновременного поглощения знаний от нескольких экспертов, взаимодействие которых вносит элемент принципиальной новизны от наложения разных взглядов и позиций.

К групповым методам извлечения знаний относятся:

Мозговой штурм *или* ***мозговая атака***- один из наиболее распространенных методов раскрепощения и активизации творческого мышления. Впервые этот метод был использован в США как способ получения новых идей в условиях запрещения критики. Замечено, что боязнь критики мешает творческому мышлению, поэтому основная идея штурма - это отделение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей.

Как правило, штурм длится около 40 мин. Участникам (до 10 человек) предлагается высказывать любые идеи (шутливые, фантастические, ошибочные) на заданную тему (критика запрещена). Обычно высказывается более 50 идей. Регламент до двух минут на выступление. Самый интересный момент штурма - это наступление пика (ажиотажа), когда идеи начинают "фонтанировать",т.е. происходит непроизвольная (бессознательная) генерация гипотез участниками. При последующем анализе всего лишь 10 - 15% идей оказываются разумными, но среди них бывают весьма

оригинальные. Оценивает результаты обычно группа экспертов, не участвовавшая в генерации.

Ведущий мозгового штурма - инженер по знаниям - должен свободно владеть аудиторией, подобрать активную группу экспертов - "генераторов", не зажимать плохие идеи - они могут служить катализатором хороших. Искусство ведущего - это искусство задавать вопросы аудитории, "подогревая" генерацию. Вопросы служат "крючком", которым извлекаются идеи. Вопросы также могут останавливать многословных экспертов и служить способом развития идей других.

Основной девиз штурма - "чем больше идей, тем лучше". Фиксация сеанса - традиционная (протокол или магнитофон).

Метод ***круглого стола*** (термин заимствован из журналистики) предусматривает обсуждение какой-либо проблемы из выбранной предметной области, в котором принимают участие с равными правами несколько экспертов. Обычно вначале участники высказываются в определенном порядке, а затем переходят к живой свободной дискуссии. Число участников дискуссии колеблется от трех до пяти-семи. Существует и специфика, связанная с поведением человека в группе.

Во-первых, от инженера по знаниям подготовка круглого стола потребует дополнительных усилий, как организационных (место, время, обстановка, минеральная вода, чай, кворум и т.д.), так и психологических (умение вставлять уместные реплики, чувство юмора, память на имена и отчества, способность гасить конфликтные ситуации и т.д.).

Во-вторых, большинство участников будут говорить под воздействием "эффекта фасада" совсем не то, что они сказали бы в другой обстановке, т.е. желание произвести впечатление на других экспертов будет существенно "подсвечивать" их высказывания.

Задача дискуссии - коллективно, с разных точек зрения, под разными углами исследовать спорные гипотезы предметной области. Обычно эмпирические области богаты таким дискуссионным материалом. Для остроты на круглый стол приглашают представителей разных научных направлений и разных поколений, это также уменьшает опасность получения односторонних знаний.

Несколько практических рекомендаций инженеру по знаниям по процедурным вопросам круглого стола перед началом дискуссии:

- убедиться, что все правильно понимают задачу (т.е. происходит сеанс извлечения знаний);
- установить регламент и четко сформулировать тему.

По ходу дискуссии проследить, чтобы слишком эмоциональные и разговорчивые эксперты не подменили тему и критика позиций друг друга была обоснованной.

Ролевые групповые игры предусматривают участие в игре нескольких экспертов. К такой игре обычно заранее составляется сценарий, распределяются роли, к каждой роли готовится портрет-описание и разрабатывается система оценивания игроков.

Существует несколько способов проведения ролевых игр. В одних играх игроки придумывают себе новые имена и играют под ними; в других - игроки переходят на "ты"; в третьих роли выбирают игроки, в четвертых роли

вытягивают по жребию. Роль - это комплекс образцов поведения. Роль связана с другими ролями. "Короля играет свита". Поскольку в нашем случае режиссером и сценаристом является инженер по знаниям, то ему и предоставляется полная свобода в выборе формы проведения игры.

Создание игровой обстановки потребует немало фантазии и творческой выдумки от инженера по знаниям. Ролевая игра, как правило, требует некоторых простейших заготовок (например, табличек "Директор", "Бухгалтерия", "Плановый отдел", специально напечатанных инструкций с правилами игры). Но главное, конечно, чтобы эксперты в игре действительно "заиграли", раскрепостились и "раскрыли свои карты".

Активные индивидуальные методы. Активные индивидуальные методы извлечения знаний на сегодняшний день наиболее распространенные. В той или иной степени к ним прибегают при разработке практически любой экспертной системы.В этих методах активную функцию выполняет инженер по знаниям, который пишет сценарий и режиссирует сеансы извлечения знаний.

Анкетирование - наиболее «жесткий» метод, т.е. наиболее стандартизованный. Инженер по знаниям заранее составляет вопросник или анкету, размножает ее и использует для опроса нескольких экспертов. Это основное преимущество анкетирования.

Сама процедура может проводиться двумя способами:

- аналитик вслух задает вопросы, и сам заполняет анкету по ответам эксперта;

- эксперт самостоятельно заполняет анкету после предварительного инструктирования.

Выбор способа зависит от конкретных условий. Второй способ нам кажется предпочтительнее, так как у эксперта появляется неограниченное время на обдумывание ответов.

Вопросник (анкета) заслуживает отдельного разговора. Существует несколько общих рекомендаций при составлении анкет. Эти рекомендации универсальны, т.е. не зависят от предметной области. Наибольший опыт работы с анкетами накоплен в социологии и психологии.

Во-первых, анкета не должна быть монотонной и однообразной, т.е. вызывать скуку или усталость. Это достигается вариациями вопросов, сменой тематики, вставкой вопросов-шуток и игровых вопросов. Во-вторых, анкета должна быть приспособлена к языку экспертов. В-третьих, следует учитывать, что вопросы влияют друг на друга, и поэтому последовательность вопросов должна быть строго продумана. В-четвертых, желательно стремиться к оптимальной избыточности. Известно, что в анкете всегда много лишних вопросов, часть из них необходима - это так называемые контрольные вопросы, а другую часть нужно минимизировать. И, наконец, в-пятых, у анкеты должны быть "хорошие мане-ры", т.е. ее язык ясен, понятен, предельно вежлив. Методическим мастерством составления анкеты овладевают только на практике.

Интервью. Под интервью понимаем специфическую форму общения инженера по знаниям и эксперта, в которой инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области. Наибольший опыт в проведении интервью накоплен также в журналистике и социологии .

Интервью очень близко к анкетированию, когда аналитик сам заполняет анкету, занося туда ответы эксперта. Основное отличие в том, что интервью позволяет аналитику опускать ряд вопросов в зависимости от ситуации, вставлять новые вопросы в анкету, изменять темп, разнообразить ситуацию общения. Кроме того, у аналитика появляется возможность "взять в плен" эксперта своим обаянием, заинтересовать его самой процедурой и тем самым увеличить эффективность сеанса извлечения.

Теперь подробнее о центральном звене активных индивидуальных методов - о вопросах. Инженеры по знаниям редко задумываются: умеют ли они задавать вопросы? В философии и математике эта проблема обсуждается с давних пор. Существует даже специальная ветвь математической логики - логика вопросов.

Вопрос в интервью - это не просто средство общения, но и способ передачи мыслей и позиции аналитика. Отсюда необходимость фиксировать в протоколах не только ответы, но и вопросы, предварительно отработав их форму и содержание.

Очевидно, что любой вопрос имеет смысл только в контексте. Поэтому вопросы может готовить инженер по знаниям, уже овладевший ключевым набором знаний.

Вопросы для эксперта имеют диагностическое значение - несколько откровенно "глупых" вопросов полностью разочаруют эксперта и отобьют у него охоту к дальнейшему сотрудничеству.

Диалог - это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в которой нет жестко регламентированного плана и вопросника.

Это определение не означает, что к свободному диалогу не надо готовиться. Напротив, внешне свободная и легкая форма этого метода требует высочайшей профессиональной и психологической подготовки. Подготовка к свободному диалогу практически может совпадать с предлагаемой в подготовкой к журналистскому интервью.

Квалифицированная подготовка к диалогу помогает аналитику стать истинным драматургом или сценаристом будущих сеансов, т.е. запланировать гладкое течение процедуры извлечения - от приятного впечатления в начале беседы к профессиональному контакту, - пробудив интерес и завоевав доверие эксперта. Для обеспечения желания эксперта продолжить беседу необходимо проводить подбадривать эксперта и подтверждать всячески его уверенность в собственной компетентности.

Так, в одном из исследований по технике журналистских диалогов экспериментально доказано, что одобрительное и поощрительное "хмыканье" интервьюера увеличивает длину ответов респондента. Чтобы разговорить собеседника, аналитику следует рассказать о себе, о работе, т.е. поговорить самому.

В свободном диалоге важно выбрать правильный темп или ритм беседы: без больших пауз, так как эксперт может отвлечься, но и "без гонки", иначе быстро утомляются оба участника и нарастает напряженность; кроме того, некоторые люди говорят и думают очень медленно. Умение чередовать разные темпы; напряжение и разрядку существенно влияет на результат беседы.

Подготовка к диалогу так же, как и к другим активным методам извлечения знаний, включает план сеанса

извлечения, в котором необходимо предусмотреть следующие стадии:

- начало беседы (знакомство, создание у эксперта "образа" аналитика, объяснение целей и задач работы);
- диалог по извлечению знаний;
- заключительная стадия (благодарность эксперту, подведение итогов, договор о последующих встречах).

Экспертные игры. Игрой называют такой вид деятельности, который отражает (воссоздает) другие ее виды [7]. Понятие экспертной игры, или игры с экспертами, в целях извлечения знаний восходит к трем источникам - это понятие деловой игры, широко используемое при подготовке специалистов и моделировании, и понятие диагностической игры, а также компьютерные игры, все чаще применяемые в обучении.

Под *деловой игрой* чаще всего понимают эксперимент, где участникам предлагается производственная ситуация, а они на основе своего жизненного опыта, общих и специальных знаний и представлений принимают решения. Решения анализируются, и вскрываются закономерности мышления участников эксперимента. Именно эта анализирующая часть деловой игры полезна для получения знаний. И если участниками такой игры становятся эксперты, то игра из деловой превращается в экспертную. Из трех основных типов деловых игр (учебных, планово-производственных и исследовательских) к экспертным ближе всего исследовательские, которые используются для анализа систем, проверки правил принятия решений.

Диагностическая игра - это та же деловая игра, но применяемая конкретно для диагностики методов принятия решения в медицине (диагностика методов диагностики). Эти

игры возникли при исследовании способов передачи опыта от квалифицированных врачей новичкам. В нашем понимании диагностическая игра - это игра, безусловно, экспертная без всяких оговорок, только с жестко закрепленной предметной областью - медициной.

Компьютерные экспертные игры. Идея использовать компьютеры в деловых играх известна давно. Но только когда компьютерные игры взяли в плен практически всех пользователей персональных ЭВМ от мала до велика, стала очевидной особая притягательность игр такого рода.

В соответствии с приведенной классификацией экспертные игры делятся на:

- индивидуальные;
- групповые.

Кроме того можно ввести другие критерии (рис.5.7):

- по числу участников;
- использование специального оборудования;
- применение компьютерной техники.

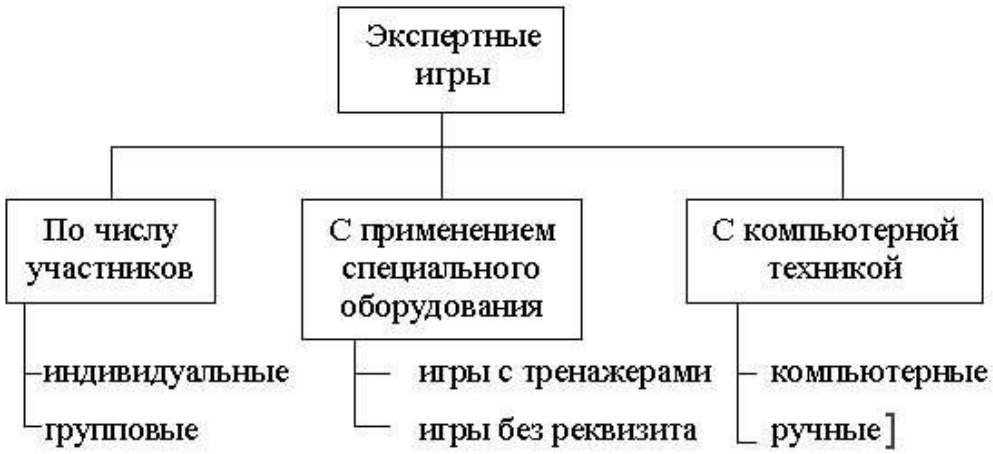


Рис.5.7. Классификация экспертных игр.

Плодотворность моделирования реальных ситуаций в играх подтверждается сегодня практически во всех областях науки и техники. Они развивают логическое мышление, способности быстро принимать решения, вызывают интерес у экспертов.

Индивидуальные игры с экспертом.В этом случае с экспертом играет инженер по знаниям, который берет на себя какую-нибудь роль в моделируемой ситуации. Например, игра "Учитель и ученик", в которой инженер по знаниям берет на себя роль ученика и на глазах эксперта выполняет его работу, а эксперт поправляет ошибки "ученика". Эта игра - удобный способ разговорить застенчивого эксперта.

В другой игре инженер по знаниям берет на себя роль врача, который хорошо знает больного, а эксперт - роль консультанта. Консультант задает вопросы, делает прогноз о целесообразности применения того или иного вида лечения. Такая игра "двух врачей" позволила, например, выявить, что эксперту понадобилось всего 30 вопросов для успешного прогноза, в то время как первоначальный вопросник, составленный медиками для этой же цели, содержал 170.

*Ролевые игры в группе.*Групповые игры предусматривают участие в игре нескольких экспертов. К такой игре обычно заранее составляется сценарий, распределяются роли, к каждой роли готовится портрет-описание и разрабатывается система оценивания игроков.

Существует несколько способов проведения ролевых игр. В одних играх игроки придумывают себе новые имена и играют под ними; в других - игроки переходят на "ты"; в третьих роли выбирают игроки, в четвертых роли вытягивают по жребию. Роль - это комплекс образцов поведения. Роль связана с другими ролями. "Короля играет

свита". Поскольку в нашем случае режиссером и сценаристом является инженер по знаниям, то ему и предоставляется полная свобода в выборе формы проведения игры.

Создание игровой обстановки потребует немало фантазии и творческой выдумки от инженера по знаниям. Ролевая игра, как правило, требует некоторых простейших заготовок (например, табличек "Директор", "Бухгалтерия", "Плановый отдел", специально напечатанных инструкций с правилами игры). Но главное, конечно, чтобы эксперты в игре действительно "заиграли", раскрепостились и "раскрыли свои карты".

Игры с тренажерами.Игры с тренажерами в значительной степени ближе не к играм, а к имитационным упражнениям в ситуации, приближенной к действительности.

Наличие тренажера позволяет воссоздать почти производственную ситуацию и понаблюдать за экспертом. Тренажеры широко применяют для обучения (например, летчиков или операторов атомных станций). Очевидно, что применение тренажеров для извлечения знаний позволит зафиксировать фрагменты "летучих" знаний, возникающих во время и на месте реальных ситуаций и выпадающих из памяти при выходе за пределы ситуации.

Компьютерные экспертные игры. Идея использовать компьютеры в деловых играх известна давно. Но только когда компьютерные игры взяли в плен практически всех пользователей персональных ЭВМ от мала до велика, стала очевидной особая притягательность игр такого рода.

5.4. Текстологические методы извлечения знаний.

Группа текстологических методов объединяет методы извлечения знаний, основанные на изучении специальных текстов из учебников, монографий, статей, методик и других носителей профессиональных знаний.

Задачу извлечения знаний из текстов можно сформулировать как задачу понимания и выделения смысла текста. Сам текст на естественном языке является лишь проводником смысла, а замысел и знания автора лежат во вторичной структуре (смысловой структуре или макроструктуре текста), настраиваемой над естественным текстом.

При этом можно выделить две такие смысловые структуры: M1 смысл, который пытался заложить автор, это его модель мира, и M2 смысл, который постигает читатель, в данном случае инженер по знаниям (рис. 13) в процессе интерпретации I. При этом T - это словесное одеяние M1, т.е. результат вербализации V.

Сложность процесса заключается в принципиальной невозможности совпадения знаний, образующих M1 и M2, из-за того, что M1 образуется за счет совокупности представлений, потребностей, интересов и опыта автора, лишь малая часть которых находит отражение в тексте T. Соответственно и M2 образуется в процессе интерпретации текста T за счет привлечения всей совокупности научного и человеческого багажа читателя.

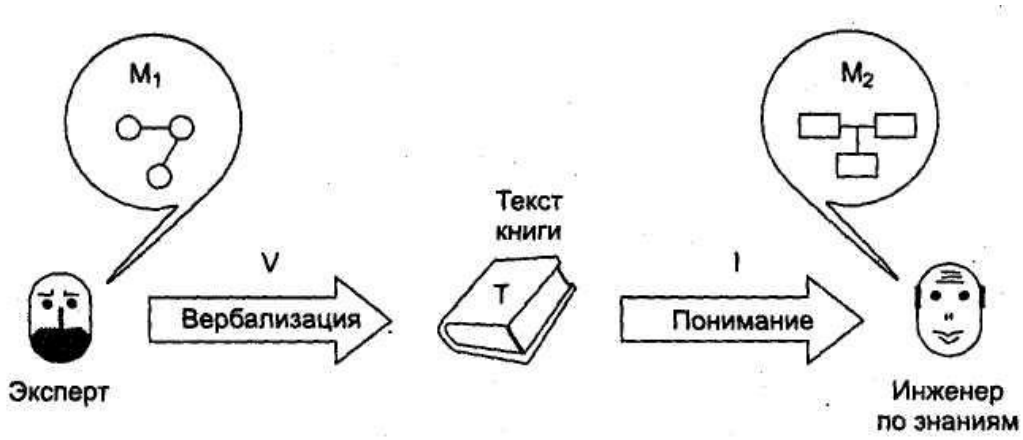


Рис.5.8. Извлечение знаний из текстов.

Встает задача выяснить, за счет чего можно достичь максимальной адекватности М1 и М2, помня о том, что понимание всегда относительно. Рассмотрим подробнее, какие источники питают модель М1 и создают текст Т. Существуют два компонента любого научного текста. Это первичный материал наблюдений и система научных понятий в момент создания текста. В дополнение к этому, на наш взгляд, помимо объективных данных экспериментов и наблюдений, в тексте обязательно присутствуют субъективные взгляды автора, результат его личного опыта, а также некоторые "общие места", или "вода". Кроме того, любой научный текст содержит заимствования из других источников (статей, монографий) и.т.д.

Основными моментами понимания текста являются:

- выдвижение предварительной гипотезы о смысле всего текста (предугадывание);
- определение значения непонятных слов (т.е. специальной терминологии);
- возникновение общей гипотезы о содержании текста (о знаниях);

- уточнение значения терминов и интерпретация отдельных фрагментов текста под влиянием общей гипотезы (от целого к частям);
- формирование некоторой смысловой структуры текста за счет установления внутренних связей между отдельными важными (ключевыми) словами и фрагментами, а также за счет образования абстрактных понятий, обобщающих конкретные фрагменты знаний;
- корректировка общей гипотезы относительно содержащихся в тексте фрагментов знаний (от частей к целому);
- принятие основной гипотезы, т.е. формирование М2.

Следует отметить наличие как дедуктивной (от целого к частям), так и индуктивной (от частей к целому) составляющей процесса понимания.

Выделяют три вида текстологических методов:

- анализ специальной литературы;
- анализ учебников;
- анализ методик.

Перечисленные три метода существенно отличаются, во-вторых, по степени концентрированности специальных знаний, и, во-вторых, по соотношению специальных и фоновых знаний. Наиболее простым методом является анализ учебников, в которых логика изложения обычно соответствует логике предмета и поэтому макроструктура такого текста будет, наверное, более значима, чем структура текста какой-нибудь специальной статьи. Анализ методик затруднен как раз сжатостью изложения и практическим отсутствием комментариев, то есть фоновых знаний, облегчающих понимание для неспециалистов. Поэтому можно

рекомендовать для практической работы комбинацию перечисленных методов.

Алгоритм извлечения знаний из текста.

Составление "базового" списка литературы для ознакомления с предметной областью и чтения по списку.

Выбор текста для извлечения знаний.

Первое знакомство с текстом (беглое прочтение). Для определения значения незнакомых слов - консультации со специалистами или привлечение справочной литературы.

Формирование первой гипотезы о структуре текста.

Внимательное прочтение текста с выписыванием ключевых слов и выражений, т.е. выделение "смысловых вех" (компрессия текста).

Определение связей между ключевыми словами, разработка макроструктуры текста в форме графа или "сжатого" текста (реферата).

Формирование поля знаний на основании структуры текста.

5.5. Понятие машинного обучения.

Изучение алгоритмов, которые позволяют компьютерам развивать их представления и базы знаний называется машинным обучением. Исследования по этой теме начались еще в середине 1950-х годов. Первые значительные успехи принадлежат программе Артура Самьюэла, которая играла в шашки на уровне профессионала. Она выбирала наиболее подходящий ход исходя из сложившейся ситуации и использовала математические функции, 16 параметров

которых описывали позиции на доске. Программа развивала их представление через выяснение значимости ходов для выбора наиболее удачных ходов.

Машинное обучение появилось как отдельная область исследований, приблизительно в 1980 и с тех пор была темой семинаров, конференций и журнала "Машинное Обучение", который появился в 1986 году. Многочисленные технологии развивались чтобы позволить компьютерам учиться разными путями. Хотя не было единой теории обучения, различные подходы доказывались вычислительно сильно и когнитивно интересно.

Машинное обучение *(англ. MachineLearning) — обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.*

Различают два типа обучения. *Обучение по прецедентам (индуктивное обучение)* основано на выявлении закономерностей в эмпирических данных. *Дедуктивное обучение* предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний. Дедуктивное обучение принято относить к области экспертных систем, поэтому термины *машинное обучение* и *обучение по прецедентам* можно считать синонимами.

Машинное обучение находится на стыке математической статистики, методов оптимизации и дискретной математики, но имеет также и собственную специфику, связанную с проблемами вычислительной эффективности и переобучения. Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы

тесно связаны с интеллектуальным анализом данных (*DataMining*).

Различаюттриосновныхнаправления в машинном обучении:

- обучение на примерах,
- искусственные нейронные сети,
- генетические алгоритмы.

Хотя вместе эти подходы не предлагают полной теории того, как люди и машины учатся, но они представили строгие описания обучения, которые были проверены на компьютерную силу и психологическое соответствие (релевантность).

Каждый из этих подходов может быть охарактеризован в терминах ввода, вывода, алгоритмов и приложений. Разные подходы к машинному обучению получили разные приложения к практическим проблемам и когнитивному моделированию. Кратко охарактеризуем каждый подход.

Обучение на примерах (англ. *LearningfromExamples*) - вид обучения, при котором интеллектуальной системе предъявляется набор положительных и отрицательных примеров, связанных с какой-либо заранее неизвестной закономерностью. В интеллектуальных системах вырабатываются решающие правила, с помощью которых происходит разделение множества примеров на положительные и отрицательные. Качество разделения, как правило, проверяется экзаменационной выборкой примеров.

Искусственные нейронные сети (ИНС) — математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это

понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Маккалока и Питтса. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

Нейронные сети не программируются в привычном смысле этого слова, они **обучаются**. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

Генетический алгоритм (англ. *geneticalgorithm*) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическуюэволюцию. Является разновидностью эволюционных вычислений. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе. «Отцом-основателем» генетических алгоритмов считается Джон Холланд (*англ.*), книга которого «Адаптация в естественных и искусственных системах» (1975) является основополагающим трудом в этой области исследований.

Вопросы для самостоятельного изучения (ознакомления).

1. Предметное (фактуальное) и проблемное (операционное) знания.
2. Понятие нечетких знаний и нечеткого вывода.
3. Немонотонность вывода.
4. Структура нейронных сетей.
5. Бизнес-приложения методов ИАД.
6. Программные продукты в области ИАД на рынке программного обеспечения.
7. Машинное обучение на примерах.
8. Искусственный интеллект в робототехнике.

Часть 2. ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

ПРАКТИКУМ

ЛАБОРАТОРНАЯ РАБОТА №1. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Тема: **ПРЕДСТАВЛЕНИЕ ЗНАНИЙ.**

Время: 4 часа.

Место: учебная аудитория.

Материально-техническое обеспечение:

1. Конспект лекций (или данное учебное пособие).
2. Рабочая тетрадь.

1. Задание.

Представить декларативное знание о понятии «Квартира» четырьмя моделями представления знаний:

1. в виде семантической сети.
2. в виде фреймов.
3. в виде логической модели.
4. в виде продукционной модели.

Квартира состоит из:

1. Кухня.
2. Гостиная.
3. Прихожая.
4. Спальня.
5. Детская.
6. Санитарный узел (туалет).
7. Ванная комната.
8. Кладовка.
9. Гардеробная.
10. Комната отдыха (игровая комната).
11. Спортивная комната (тренажерная).

12. Бытовая комната.

Дополнительные задания:

1. Компьютерный класс.
2. Компьютерный клуб.

2. Порядок выполнения и результаты.

Методические рекомендации по проведению занятия.

- Студенты по вариантам (вариант - часть квартиры) самостоятельно выполняют задание в рабочей тетради (лучше сначала на черновиках). Используется конспект лекций (или материал п.2.2 Части 1данного учебного пособия).
- Семантическая сеть должна содержать не менее 20 вершин с разными типами связей. Фреймовая модель должна содержать не менее 6 фреймов, связанных двумя типами связи. В продукции отразить все составляющие.
- В процессе работы каждый студент предоставляет преподавателю составленные модели. Преподаватель со студентом обсуждают и уточняют модели, при необходимости модели дорабатываются.
- Затем обучаемые составляют общую семантическую сеть и сеть фреймов понятия «Квартира» на доске и зарисовывают ее в тетрадь.

*Результат:*четыре модели (согласно варианта) и две общие модели в рабочей тетради.

ЛАБОРАТОРНАЯ РАБОТА №2. ОСНОВЫ ПРОГРАММИРОВАНИЯ В «ПРОЛОГ – Д»

Тема: **ОСНОВЫ ПРОГРАММИРОВАНИЯ В «ПРОЛОГ – Д».***Учебные вопросы:*

1. Интерфейс и синтаксис.
2. Арифметика и сравнение.
3. Графические возможности.
4. Создание базы знаний.

Время: 4 часа.

*Место:*компьютерный класс.

Материально-техническое обеспечение:

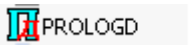
3. ПЭВМ.
4. Система Пролог-Д для Windows. Архив с системой можно скачать с сайта http://pgsha.ru в разделе учебно-методической работы кафедры информационных систем (прямая ссылка - http://pgsha.ru/web/generalinfo/facultiesPGSHA/AppliedInformatics/about/structure/cathedras/is/method/).

Методические рекомендации по проведению занятия.

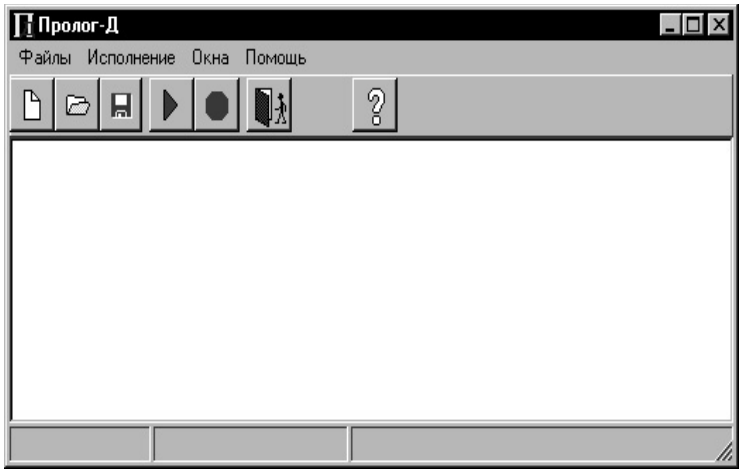
- Перед началом занятия необходимо распаковать архив с системой Пролог-Д (можно прямо на рабочий стол компьютера). Система работает без инсталляции прямо из директории.
- Целесообразно каждое задание делать в отдельном окне. Это облегчит проверку заданий.

1. Интерфейс и синтаксис.

Интерфейс логического языка программирования «Пролог-Д» аналогичен интерфейсу и приемам работы в операционной системе WINDOWS.

Для начала работы с «Пролог-Д» необходимо зайти в папку под названием «Пролог-Д», расположение которой укажет преподаватель. Затем выбрать ярлык Пролога-Д 

или с помощью программ просмотра диска найти файл с программой prologw.exe и двойным нажатием левой клавиши мыши инициировать его исполнение. На экране появится заставка системы Пролог-Д Windows.



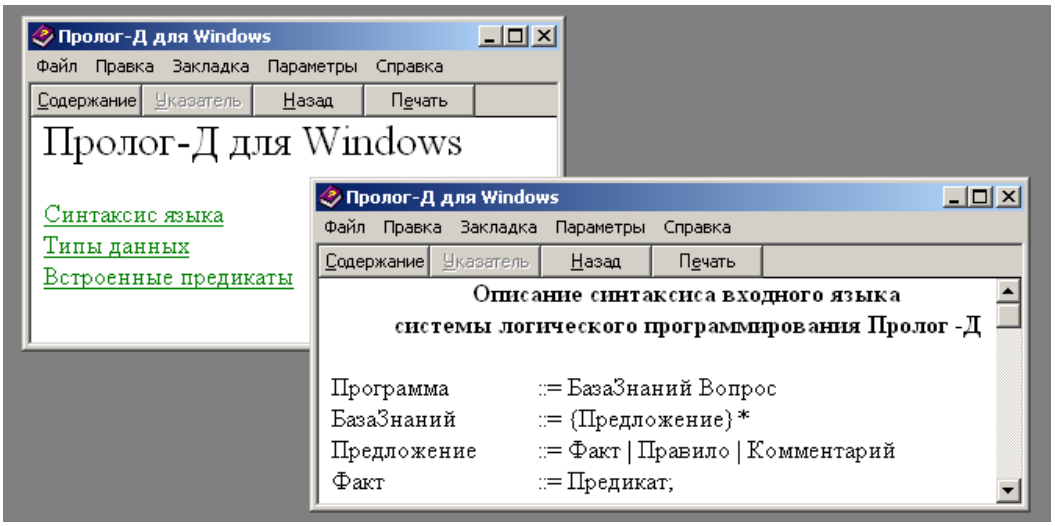
Элемент меню **Файлы.**Система позволяет работать одновременно снескольким файлами аналогично работе в ОС WINDOWS.

Элемент меню**Исполнение.**Содержит элементы*Исполнить*, *Прервать*и*Настройки*.Отмечая или снимая отметки в соответствующих квадратиках панели настроек можно включить и выключить трассировку, вывод вопроса, поручить системе сохранять текст программы при

каждом отладочном запуске, а также выбрать, куда направлять вывод в ходе исполнения программы.

Элемент меню**Окна.**При выборе этого элемента появляется падающее меню. Выбор элементов меню *Каскад, Мозаика, Упорядочить* и *Свернуть все* определяет взаимное расположение окон на экране.

Элемент меню**Помощь.** При выборе данного элемента меню пользователю предоставляется возможность выбор различных видов помощи, указываемых в падающем меню: помощь о программе и помощь по разделам. Все виды помощи построены по гипертекстовому принципу с использование стандартной подсистемы организации помощи MSWindows.



Синтаксис языка «Пролог-Д» подробно описан в файле помощи - кнопка  или окно *Помощь*, *Язык*, *Синтаксис*. Откройте его и просмотрите.

В данной версии Пролога-Д длина имени предикатного символа не ограничена. Аргументом может быть любой терм.

Аргументов может и не быть. Факты описывают объекты и отношения между ними.

2. Арифметика и сравнение.

В языке пролог имеются встроенные предикаты, в частности предикаты арифметики и сравнения. Их описание находится в файле помощи – кнопка  или окно *Помощь*, *Язык*, *Встроенные предикаты*, *Арифметика*.

Для освоения принципов работы со встроенными арифметическими предикатами и встроенными предикатами сравнения необходимо выполнитьнижеприведенные задачи.Например, выражение Z = X*Y запишется на Прологе в следующем виде:

?УМНОЖЕНИЕ(X,Y,Z).

Задача 1. *На Прологе-Д необходимо описать вычисление объема параллелепипеда высотой h, в основании которого прямоугольник, имеющий стороны длиной a и b*.

Известна формула определения объема параллелепипеда:

$$V_{пар}=a*b*h.$$

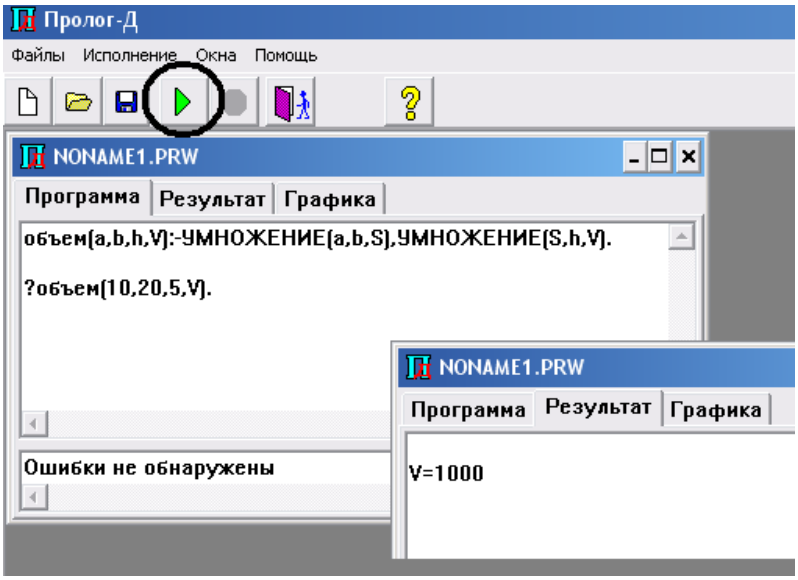
Предикат, который будет выполнен, если будет вычислен объем параллелепипеда, должен иметь четыре аргумента - длины сторон a и b, высоту h и величину объема. Целесообразно, чтобы Имя предиката отражало его назначение - этому критерию удовлетворит имя «объем». Данный предикат будет составным и имеет вид:

объем(a,b,h,V):-УМНОЖЕНИЕ(a,b,S),УМНОЖЕНИЕ(S,h,V).

К данному предикату (базе знаний) можно задать вопросы:

?объем(10,20,5,V).

Нажав кнопку  получим ответ системы Пролог-Д:V=1000.



Предикат «объем» обратим, это означает, что используя это описание можно вычислить не только объем по заданным сторонам и высоте, но и любую (одну) сторону или высоту по заданным высоте, стороне и объему.

Задайте вопрос:

?объем(a,20,5,1000).

Получим ответ : a = 10.

Задача2: Если *необходимо вычислить число x=2*3+1*, то для этого достаточно набрать на клавиатуре вопрос к другому встроенному предикату:

?УМНОЖЕНИЕ(2,3,1,x).

Ответ системы:

x=7

Реализация **деления** осуществляется через встроенный предикат умножения. Например, для выражения x=z/y имеем:

ДЕЛЕНИЕ(z,y,x):-УМНОЖЕНИЕ(x,y,z).

Реализация **вычитания.**Для выражения x=y-z запишем:

ВЫЧИТАНИЕ(y,z,x):-СЛОЖЕНИЕ(x,z,y).

В данных случаях во встроенных предикатах переменные как бы перевернуты на оборот.

Задание1:

1. Напишите правило(предикат) и вопрос для вычисления площади круга.
2. Вычислите выражение «x=8/2».
3. Вычислите выражение «x=10 - 7».
4. Вычислите выражение «x=5*2*3+2».

Результат представить преподавателю.

Рассмотрим несложный пример, иллюстрирующий применение операторов сравнения **БОЛЬШЕ** и **НЕ.**

Задача2: *Опишите на языке Пролог-Д вычисление функции Хевисайда, определяемой формулой*:

$$h(x)= \begin{cases} 0, \text{ если } x < 0. \\ 0, \text{ если } x = 0. \\ 1, \text{ если } x > 0. \end{cases}$$

База знаний должна содержать описание предиката меньше и равно, который выше уже был описаны, предикат, выполняющийся при вычислении функции Хевисайда, будет называться ХЕВИСАЙД. Этот предикат будет иметь два аргумента, первый это аргумент функции, а второй ее

значение. Предикат ХЕВИСАЙД определяется через два альтернативных описания для всех значений X.

М(X,Y):-НЕ(БОЛЬШЕ(X,Y).

ХЕВИСАЙД(X,0):-М(X,0).

ХЕВИСАЙД(X,1):-БОЛЬШЕ(X,0).

К этой базе знаний можно задать различные вопросы. Например:

?ХЕВИСАЙД(20,X).

Ответ системы Пролог-Д:

X=1.

Задание2: задать базе знаний вопросы для получения всех вариантов ответов для формулы Хевисайда.

Попробуйте набрать вопрос:

?УМНОЖЕНИЕ(x,3,1,7).

Объясните результат и запишите исходный текст (задание) упражнения.

Результаты работы представить преподавателю.

3. Графические возможности.

Они предназначены для выполнения вывода графики и других подобных операций. Встроенные предикаты, если они записаны в вопросе, должны выполняться одинаково, независимо от того, записана в память машины база знаний или нет, и какая это база знаний. В определенном смысле это напоминает непосредственный режим работы в языке БЕЙСИК. Например, если необходимо построить на экране

отрезок, соединяющий две точки с координатами (10,10) и (200, 200), то достаточно задать вопрос:

?ЗАПИСЬ_В(“grp:”),ЛИНИЯ(10,10,100,100,1

Графические возможности «Пролог-Д» более подробно описаны в файле помощи – кнопка  или окно *Помощь*, *Язык*, *Встроенные предикаты*, *Графика* .

Задание3: введите вышеуказанное выражение и запустите его. Требуемый отрезок появится на экране. Дополнительно ниже начертите три параллельных линии разного цвета, а также нарисуйте крест (две пересеченные линии разного цвета).

Попробуйте набрать вопрос:

?ЛИНИЯ(x, 10, 100, 100, 1).

Объясните результат и запишите исходный текст (задание) упражнения.

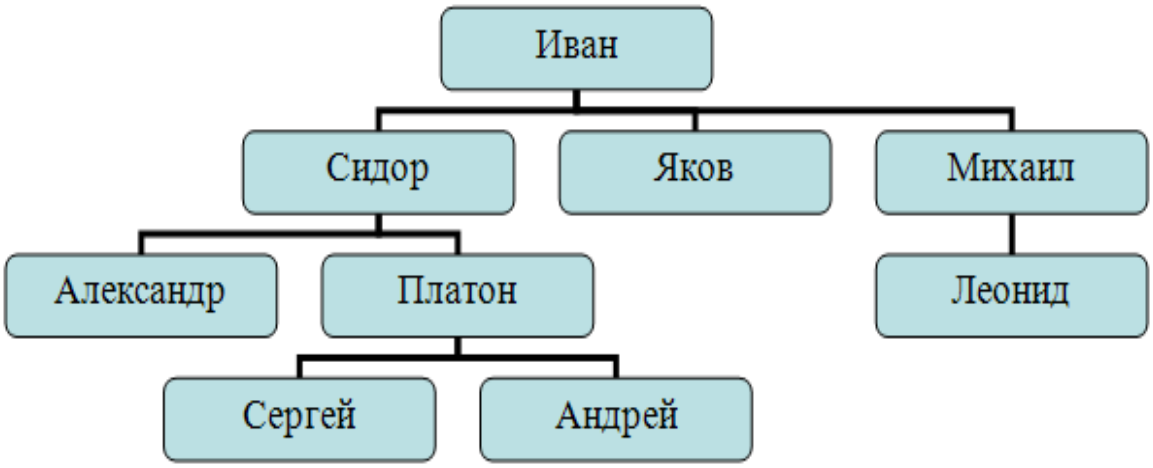
Задние4: *построить изображения небольшого дома с окном*.

Результат представить преподавателю.

4. Создание базы знаний.

Задание 5.Для изучения основного предназначения логического языка создайте базу знаний и проверьте ее. Для этого наберите текст программы и вопрос, которые рассмотрены в п.1.4 Части 1 данного пособия. Отладьте программу и задайте пару других вопросов.

Задание 6:Создать на языке пролог-Д базу знаний, описывающую семейное древо семьи.



Составьте запросы к базе знаний, позволяющие выяснить:

1) Сына, задав имя отца;
2) Всех братьев;
3) Всех внуков и дедов;
4) Всех племянников и дядей.
5) Племянников, задав имя дяди.

ЛАБОРАТОРНАЯ РАБОТА №3. РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ

Тема: **РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ.**

*Время:*8 часа.

Место: компьютерный класс.

Материально-техническое обеспечение:

1. ПЭВМ.
2. Высокоуровневые языки программирования.
3. Конспект лекций (или данное учебное пособие).
4. Рабочая тетрадь.

1. Задание.

Отработать этапы разработки экспертной системы для решения задачи (проблемы) выбора. Осуществить программную реализацию экспертной системы на любом языке программирования.

Разрабатываемая экспертная система относится к классу поверхностных демонстрационных (учебных) систем. *Поверхностные ЭС* представляют знания в виде правил (условие – действие).

Создание экспертной системы в рамках данного занятия проекта позволяет изучить и реализовать все этапы разработки ЭС:

1. идентификация,
2. концептуализация,
3. формализация,
4. выполнение,
5. тестирование,
6. опытная эксплуатация.

Особенностью работы является то, что студент выполняет функционал всех членов коллектива разработчиков ЭС – эксперта, инженера по знаниям, программиста и пользователя.

Примерные задачи (проблемы) для экспертной системы.

1. Разработка экспертной системы «Выбор сотового телефона».
2. Разработка экспертной системы «Выбор квартиры».
3. Разработка экспертной системы «Выбор игрушек для девочек».
4. Разработка экспертной системы «Выбор персонального компьютера».
5. Разработка экспертной системы «Выбор домашнего животного».
6. Разработка экспертной системы «Выбор ноутбука».
7. Разработка экспертной системы «Выбор свадебного платья».
8. Разработка экспертной системы «Выбор оружия самозащиты».
9. Разработка экспертной системы «Выбор места отдыха».
10. Разработка экспертной системы «Выбор специальности».
11. Разработка экспертной системы «Выбор принтера».
12. Разработка экспертной системы «Выбор мотоцикла».
13. Разработка экспертной системы «Выбор прически».
14. Разработка экспертной системы «Выбор компьютерной техники».
15. Разработка экспертной системы «Выбор страны отдыха».
16. Разработка экспертной системы «Выбор спортивной секции для ребенка».
17. Разработка экспертной системы «Выбор КПК».
18. Разработка экспертной системы «Выбор автомобиля».

19. Разработка экспертной системы «Выбор антивирусной программы».
20. Разработка экспертной системы «Выбор шампуня для волос».
21. Разработка экспертной системы «Выбор сабвуфера».
22. Разработка экспертной системы «Выбор вида отдыха».
23. Разработка экспертной системы «Формирование кадрового резерва».
24. Разработка экспертной системы «Брачное агентство».
25. Разработка экспертной системы «Свадебное платье».

2. Порядок выполнения и результаты.

Методические рекомендации по проведению занятия.

- Студенты самостоятельновыбирают решаемую задачу из представленного выше перечня или придумывают другую задачу.
- **ВАЖНО**. Студент должен разбираться в выбранной задаче, то есть быть экспертом. Именно эти знания будут заложены в экспертную систему, которая предназначена для замены эксперта.
- Задание выполняют в рабочей тетради (лучше сначала на черновиках). Используется конспект лекций (или материал Части 1 п.3.5 данного учебного пособия).
- Количество правил в базе знаний: на оценку «отлично» - не менее 30, на «хорошо» - не менее 25, на «удовлетворительно» - не менее 20.
- Дерево решений составлять исходя из требуемого функционала, например, цель (т.е. что нужно делать) приобретения компьютера (телефона и т.д.), а не сумма денежных средств для этого.
- Вопросы в дереве решений могут повторяться не более двух раз.

- Рекомендуемая длина ветви дерева решений 3-6 вершин.
- Промежуточные вершины вывода делать не рекомендуется.
- Программная реализация экспертной системы возможна на любом языке программировании. Предпочтительно на объектно-ориентированных языках (Delphi, C#) или Пролог.
- Программная реализация экспертной системы должна иметь дружественный интерфейс, содержать картинки и иллюстрации, эстетично оформлена.
- Желательно отражать в окне программы вопросы из дерева решений и ответы на них, т.е. отобразить «цепочку логических выводов».
- В процессе работы каждый студент сообщает преподавателю выбранную задачу, предоставляет дерево решений. Преподаватель со студентом обсуждают и уточняют тему и дерево решений, при необходимости они дорабатываются.

Результатыработы:

1. Сформулированная проблема (задача).
2. Дерево решений для выбранной проблемы (задачи).
3. Таблица переменных.
4. База знаний (правила).
5. Таблицы структур данных.
6. Блок схема алгоритма программной реализации.
7. Программная реализация ЭС.

Постановка задачи, дерево решений и его описание, таблицы, база знаний, структуры данных выполняются и алгоритм в рабочей тетради или на ПЭВМ и распечатываются на листах формата А4.

Программная реализация экспертной системы записывается на носитель (флэшка или CD-диск) и предоставляется на проверку в конце занятия. Программа должна работать в автономном режиме (exe – файл).

ЛАБОРАТОРНАЯ РАБОТА №4. АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. ХРАНИЛИЩЕ ДАННЫХ

Тема: **АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. ХРАНИЛИЩЕ ДАННЫХ.**

Учебные вопросы:

5. Архитектура хранилища данных в DeductorWarehouse.
6. Создание хранилища данных в DeductorWarehouse.
7. Наполнение хранилища данных.
8. Извлечение информации из хранилища данных.

Время: 4 часа.

Место: компьютерный класс.

Материально-техническое обеспечение:

5. ПЭВМ.
6. Аналитическая платформа **Deductor** версии Academic, которую можно скачать с официального сайта фирмы-разработчика BaseGroupLabs (www.basegroup.ru) или с диска, прилагающегося к источнику [4].
7. Файлы с исходными данными в формате txt. Архив файлами можно скачать с сайта фирмы разработчика (http://www.basegroup.ru/download/demoprg/practicum/) или с диска, прилагающегося к источнику [4].

Методические рекомендации по проведению занятия.

- Перед началом занятий необходимо распаковать архив с файлами исходных данных в папку «Мои документы\Лаб4».
- Первый учебный вопрос рекомендуется изучить и законспектировать в рабочую тетрадь.
- Ответы на вопросы для проверки в конце каждого

учебного вопроса могут быть письменными или устными.

1. Архитектура хранилища данных в DeductorWarehouse.

Хранилище данных (ХД)**DeductorWarehouse**- это специально организованная база данных, ориентированная на решение задач анализа данных и поддержки принятия решений, обеспечивающая максимально быстрый и удобный доступ к информации. **DeductorWarehouse 6** соответствует модели ROLAP (схема «снежинка»).

Хранилище данных **DeductorWarehouse**включает в себя потоки данных, поступающие из различных источников, и специальный семантический слой, содержащий так называемые метаданные (данные о данных). Семантический слой и сами данные хранятся в одной СУБД. Все данные в хранилище DeductorWarehouse хранятся в структурах типа «снежинка», где в центре расположены таблицы фактов, а «лучами» являются измерения, причем каждое измерение может ссылаться на другое измерение. Именно эта схема чаще всего встречается в хранилищах данных (рис.4.9.).

Объекты хранилища данных DeductorWarehouseследующие.

Измерение - это последовательность значений одного из анализируемых параметров. Например, для параметра «время» это последовательность календарных дней, для параметра «регион» - список городов. Каждое значение измерения может быть представлено координатой в многомерном пространстве процесса, например, Товар, Клиент, Дата.

Атрибут - это свойство измерения (т.е. точки в пространстве). Атрибут как бы скрыт внутри другого

измерения и помогает пользователю полнее описать исследуемое измерение. Например, для измерения Товар атрибутами могут выступать Цвет, Вес, Габариты.

Факт - значение, соответствующее измерению. Факты - это данные, отражающие сущность события. Как правило, фактами являются численные значения, например, сумма и количество отгруженного товара, скидка.

Ссылка на измерение - это установленная связь между двумя и более измерениями. Дело в том, что некоторые бизнес-понятия (соответствующие измерениям в хранилище данных) могут образовывать иерархии, например, Товары могут включать Продукты питания и Лекарственные препараты, которые, в свою очередь, подразделяются на группы продуктов и лекарств и т. д. В этом случае первое измерение содержит ссылку на второе, второе - на третье и т.д.

Процесс - совокупность измерений, фактов и атрибутов. По сути, процесс и есть «снежинка». Процесс описывает определенное действие, например, продажи товара, отгрузки, поступления денежных средств и прочее.

Атрибут процесса - свойство процесса. Атрибут процесса в отличие от измерения не определяет координату в многомерном пространстве. Это справочное значение, относящееся к процессу, например, № накладной, Валюта документа и так далее. Значение атрибута процесса в отличие от измерения может быть не всегда определено.

В DeductorWarehouseможет одновременно храниться множество процессов, имеющих общие измерения, например, измерение *Товар*, фигурирующее в процессах *Поступления*и *Отгрузка*.

Все загружаемые в ХД данные обязательно должны быть определены как измерение, атрибут либо факт.Принадлежность данных к типу (измерение, ссылка на измерение, атрибут или факт) содержится в семантическом слое хранилища. Обратим внимание на то, что:

■ таблицы *измерений* содержат только справочную информацию (коды, наименования и т.п.) и ссылки на другие измерения при необходимости;

■ таблица *процесса* содержит только факты и коды измерений (без их атрибутов).

Проектирования структуры ХД. Рассмотрим проектирование структуры хранилища данных на несложном примере.

Имеется история продаж различных товаров по дням в нескольких торговых объектах. Товары объединены в группы. Требуется спроектировать структуру хранилища данных. Все данные представлены в 4 таблицах: Товарные группы (groups.txt), Товары (produces.txt), Отделы (stores.txt), Продажи (sales.txt). Архитектура хранилища данных и фрагменты этих таблиц приведены ниже (рис.1).

-

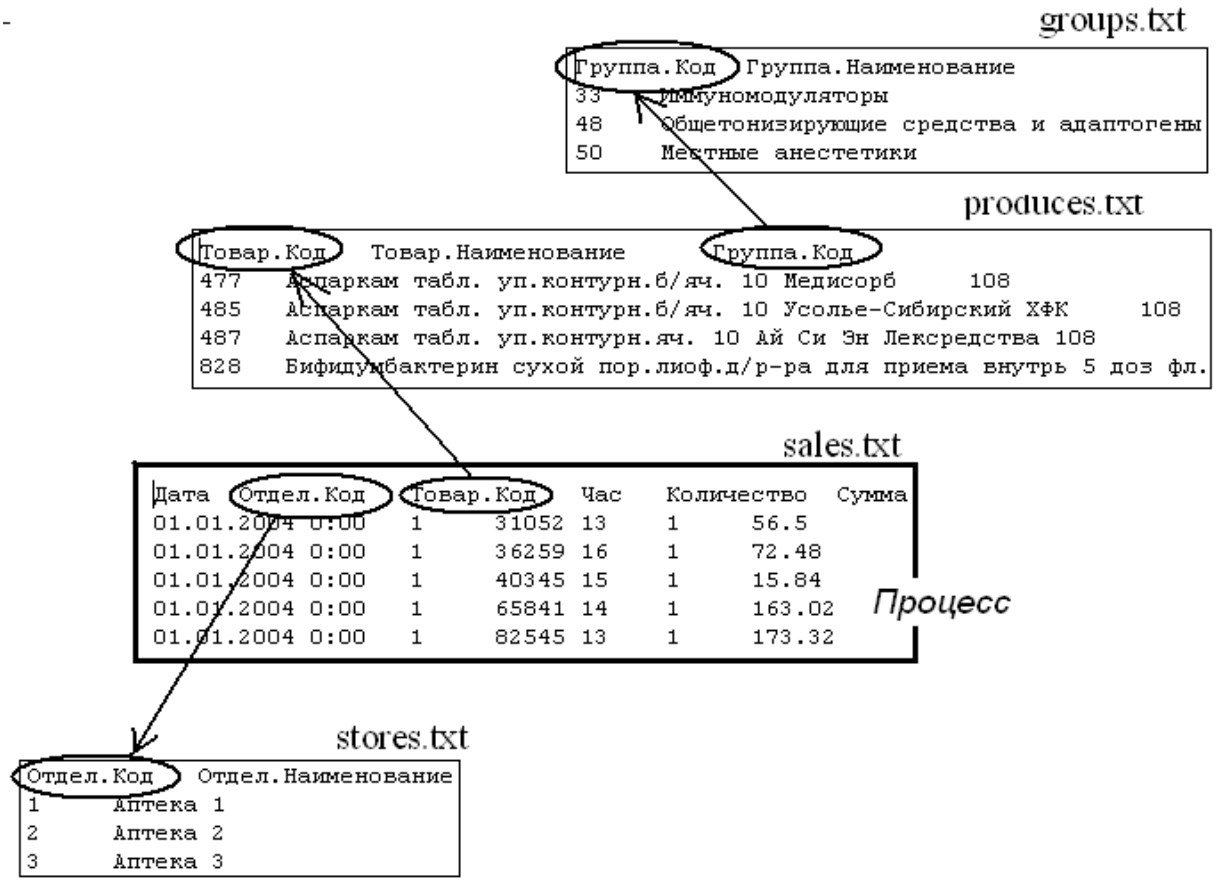


Рис. 1. Пример схемы ХД «снежинка».

В таблице groups.txt*Код группы* является измерением, а *Наименование группы* - его атрибутом.

В таблице produces.txt*Код товара* является измерением, а *Наименование товара*- его атрибутом, а *Код группы* -ссылкой на одноименное измерение.

В таблице stores.txt*Код отдела* является измерением, а *Наименование отдела* - его атрибутом.

В таблице sales.txt*Дата* является измерением, *Отдел, Код товара* и *Код группы* как было сказано выше – измерения.*Час покупки* - измерение, *Количество* и *Сумма*- факты, т.е. таблица sales.txt является описанием процесса продаж в трех аптеках.

Вопросы для проверки (самопроверки):

1. Какая схема реляционного ХД используется в DeductorWarehouse?
2. Перечислите объекты хранилища DeductorWarehouseи дайте их определения.
3. Чем отличается атрибут процесса от измерения?
4. Как должна выглядеть структура таблицы-справочника, если имеются иерархии?
5. Что нужно сделать, если в приведенном примере выбранный уровень детализации продаж по дням не устраивает - необходимо сохранить максимальную детализацию исходных данных?
6. Почему в приведенном примере поле Час покупки не может быть фактом?

2. Создание хранилища данных в DeductorWarehouse.

Откройте программу DeductorStudio, используя ярлык на рабочем столе  Deductor Studio или через кнопку Пуск.

Для создания нового хранилища данных или подключения к существующему в DeductorStudio необходимо перейти на закладку **Подключения** и запустить **Мастер подключений**.

На экране появится первый шаг **Мастера**, в котором следует выбрать тип источника (приемника), к которому нужно подключиться. Выберите **DeductorWarehouse**и нажмите кнопку**Далее.**

На следующем шаге из единственно доступного в списке типа базы данных выберем **Firebird**и перейдем на третий шаг мастера. В нем зададим параметры базы данных, в

которой будет создана физическая и логическая структура хранилища данных (рисунок 2):

- база данных - D:\Мои документы\farma.gdb (или любой другой путь на диске);
- логин - 1, пароль - 1;
- поднятый флажок**Сохранять пароль**.

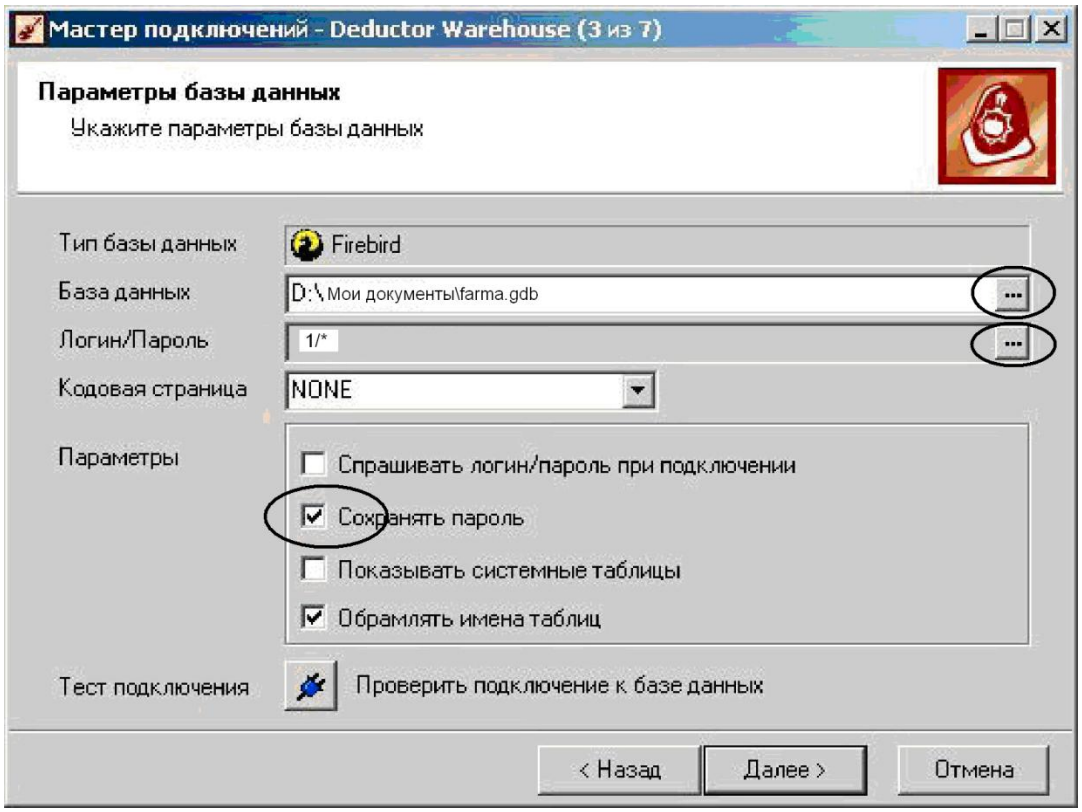


Рис.2. Установка параметров базы данных

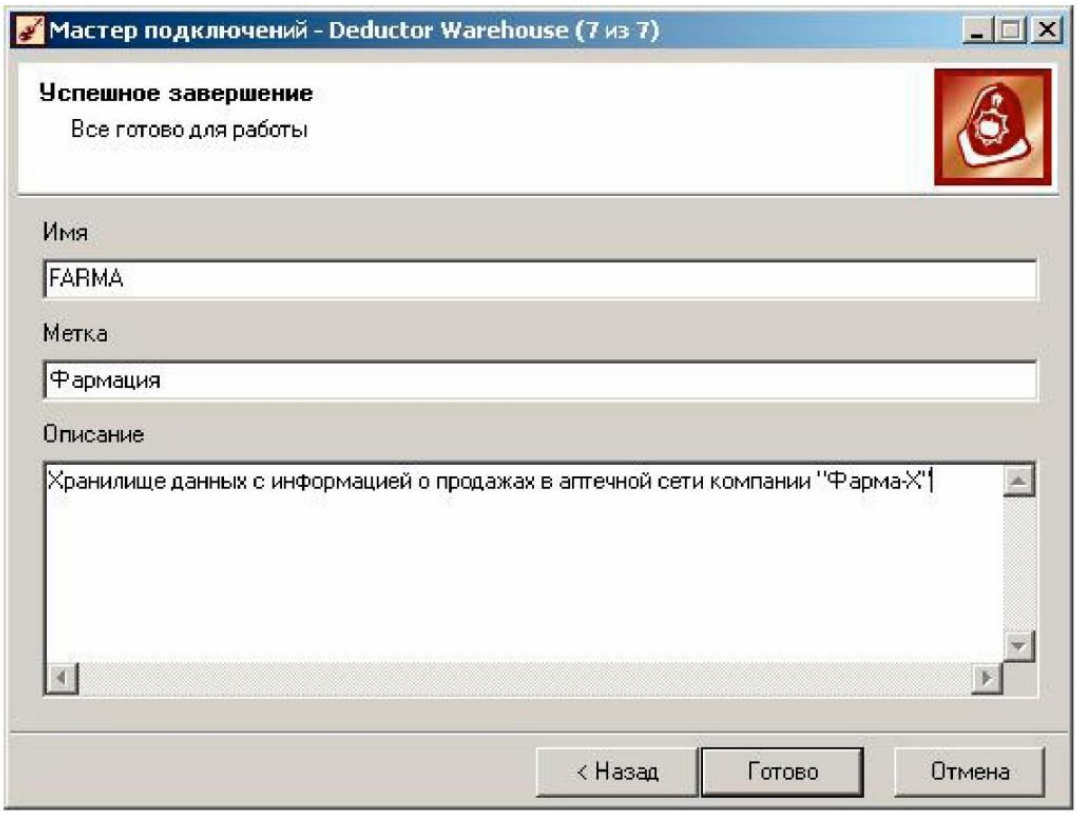
НажмитеДалее.

На следующей вкладке выберем последнюю версию для работы с ХД **DeductorWarehouse6** (предыдущие версии необходимы для совместимости с ранними версиями хранилищ).

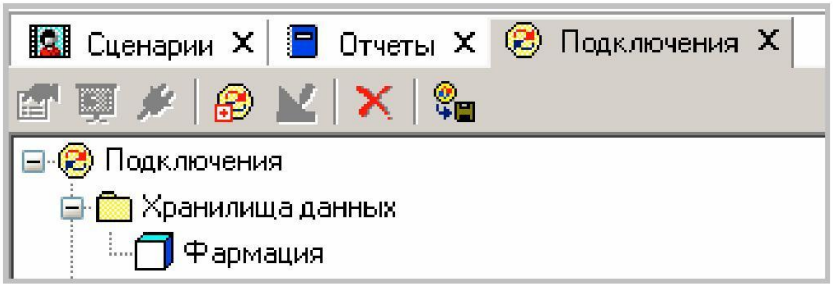
На следующем шаге при нажатии на кнопку

По указанному ранее пути будет создан файл **farma.gdb**(появится сообщение об успешном создании). Это и есть пустое хранилище данных, готовое к работе.

На последних двух шагах осталось выбрать визуализатор для подключения (здесь это **Сведения** и **Метаданные**) и задать имя, метку и описание для нового хранилища.



После нажатия на кнопку **Готово** на дереве узлов подключений появится метка хранилища.



Если соединение по какой-либо причине установить не удалось, то будет выдано сообщение о соответствующей ошибке. В этом случае нужно проверить параметры подключения хранилища данных и при необходимости внести в них изменения (используйте для этого кнопку **Настроить подключение**.

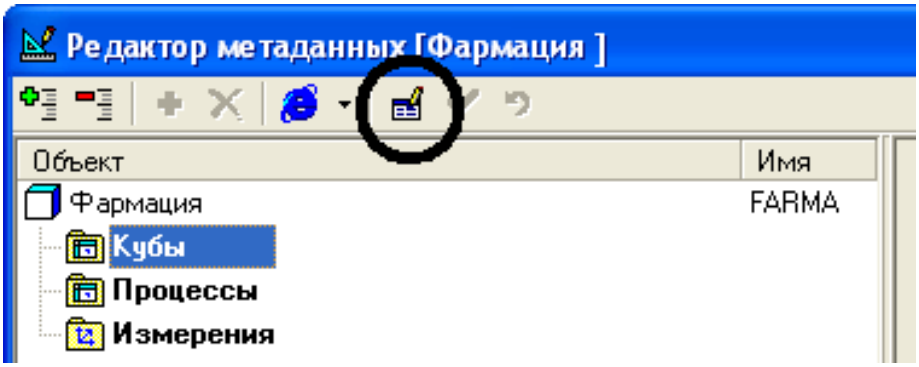
Для проверки доступа к новому хранилищу данных воспользуйтесь кнопкой 

Если спустя некоторое время появится сообщение «Тестирование соединения прошло успешно», то хранилище готово к работе.

Сохраните настройки подключений, нажав на кнопку сохранения 
.

После создания хранилища необходимо спроектировать его структуру, т.к. в пустом хранилище нет ни одного объекта (процессов, измерений, фактов). Для этого предназначен «Редактор метаданных», который вызывается кнопкой  на вкладке **Подключения**. Нажмите ее.

Для перехода в режим внесения изменений в структуру хранилища нажмем кнопку**Разрешить редактировать.**



Появится диалоговое окно с предупреждением. Нажмем**Да** и в открывшемся окне редактора метаданных,

встав на узле **Измерения**, при помощью кнопки **Добавить** добавим в метаданные первое измерение Код группы со следующими параметрами:

- имя – GR_ID;
- метка - Группа.Код;
- тип данных-целый.

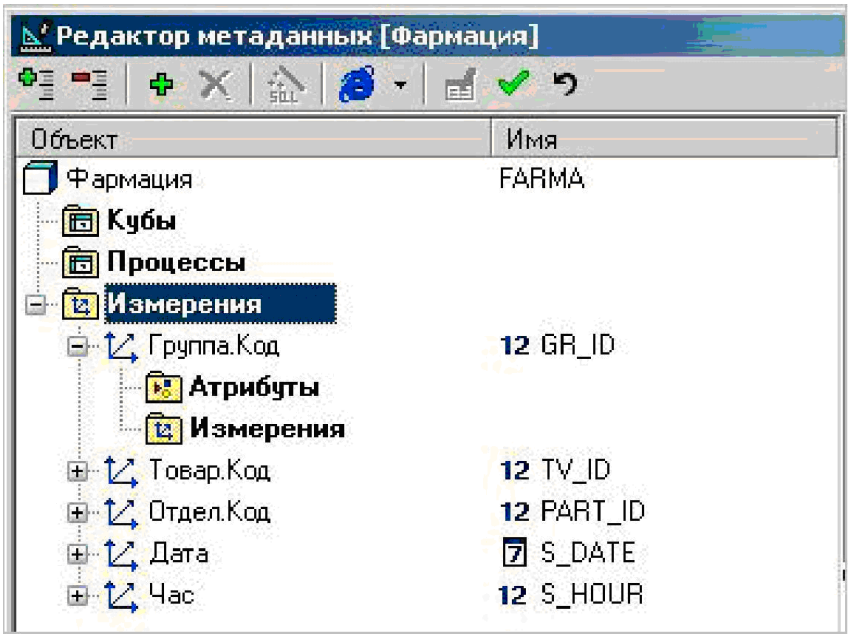
Имя - это семантическое название объекта хранилища данных, которое увидит пользователь, работающий с ХД. (Эти параметры для таблицы «Товарные группы»).

Выполните аналогичные действия для создания всех остальных измерений, взяв параметры из таблицы 1.

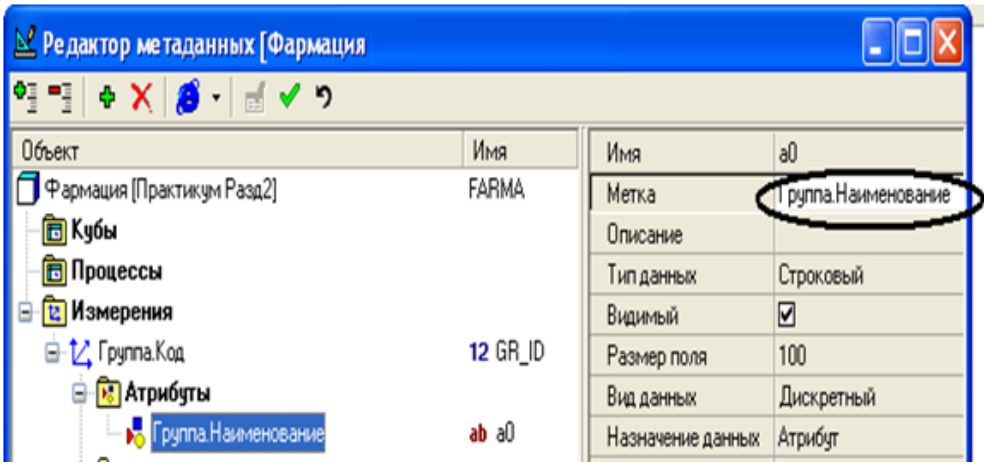
Таблица 1. - Параметры измерений

Измерение	**Имя**	**Метка**	**Тип данных**
Код группы	GR-ID	Группа.Код	целый
Код товара	TV_ID	Товар.Код	целый
Код отдела	PART_ID	Отдел.Код	целый
Дата	S_DATE	Дата	дата/время
Час покупки	S_HOUR	Час	целый

В результате структура метаданных нашего хранилища будет содержать 5 измерений.



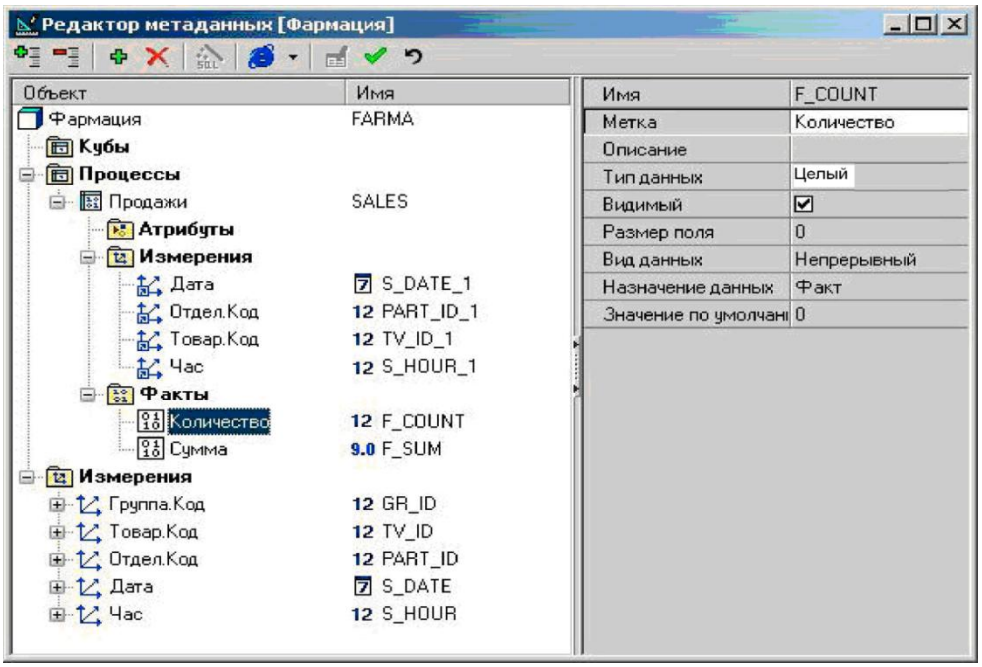
К каждому измерению, кроме Дата и Час, теперь добавим по текстовому атрибуту. Для этого в измерении «Группа.Код» правой кнопкой мыши откроем **Атрибуты** и справа в поле «Метка» введем название атрибута - Группа.Наименование. Тип данных оставим строковым. Размер поля в строковых атрибутах предлагается равным 100, оставим это без изменений.



Аналогично введите названия атрибутов :для измерения Товар.Код - Товар.Наименование, для измерения Отдел.Код - Отдел.Наименование..

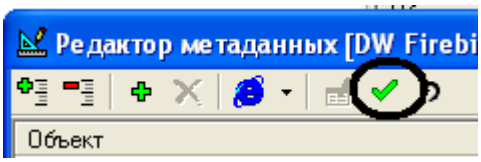
Каждое измерение может ссылаться на другое измерение, реализуя тем самым иерархию измерений (схема «снежинка»). В нашем случае измерение Товар.Код ссылается на Группа.Код (см. табл. 1 и табл. 2). Эту ссылку и установим путем добавления объекта к измерению, для этого в измерении «Товар.Код» правой кнопкой мыши откроем Измерение и выберем пункт Добавить. Имя ссылки зададим GR_ID_1, а метку - Группа.Код. Ссылка на измерение отображается иконкой .

После того как все измерения и ссылки на измерения созданы, приступают к формированию процесса. Назовем его *Продажи* и «соберем» его из 4 существующих измерений: Дата, Отдел.Код, Товар.Код, Час (кнопка). Кроме них в нашем процессе присутствуют два факта: Количество и Сумма, причем первый - целочисленный, второй – вещественный. Результат представлен на рисунке 3.



Рису. 3. Создание метаданных процесса.

На этом проектирование структуры и метаданных ХД закончено. Для того чтобы принять все изменения, нужно нажать кнопку**Принять изменения**.



После этого закройте окно редактора. Структура хранилища данных готова.

Вопросы для проверки (самопроверки):

1. Что такое «Редактор метаданных» в DeductorStudio?
2. Как создать новое пустое хранилище данных? Как сделать иерархию измерений?
3. Какие типы данных могут быть у объектов хранилища

DeductorWarehouse 6?

3. Наполнение хранилища данных.

После создания структуры хранилища данных оно представляет с собой «пустое» ХД **DeductorWarehouse 6** с настроенным семантическим слоем. В таком виде оно готово к загрузке в него данных из внешних структурированных источников. Для этого необходимо написать соответствующий сценарий в DeductorStudio.

Сценарий загрузки должен выполнять следующие функции.

1. Импорт данных в DeductorStudioиз базы данных, учетной системы или предопределенных файлов.
2. Опциональная предобработка данных, например, очистка или преобразование формата.
3. Загрузка данных в измерения и процессы хранилища DeductorWarehouse.

В рассматриваемой задаче исходными данными для ХД служат 4 текстовых файла: **groups.txt**(товарные группы), **produces.txt**(товары), **stores.txt**(отделы), **sales.txt**(продажи по дням). Поэтому сценарий загрузки должен быть настроен на использование в качестве источников данных на эти файлы.

ВАЖНО!При создании сценария необходимо строго придерживаться следующих правил.

1. Первыми загружаются все измерения, имеющие атрибуты. Только после загрузки всех измерений загружаются данные в процесс(ы).
2. Среди измерений также имеется правило на порядок загрузки: загружать измерения нужно, начиная с самого верхнего уровня иерархии и спускаться по иерархии ниже.

Это крайне важно, в противном случае иерархия не будет создана.

Поясним второе правило (рис. 4). Измерение *Группа* находится выше измерения *Товар*, поэтому последовательность загрузки измерений будет следующая: сначала *Группа*, а затем *Товар*.

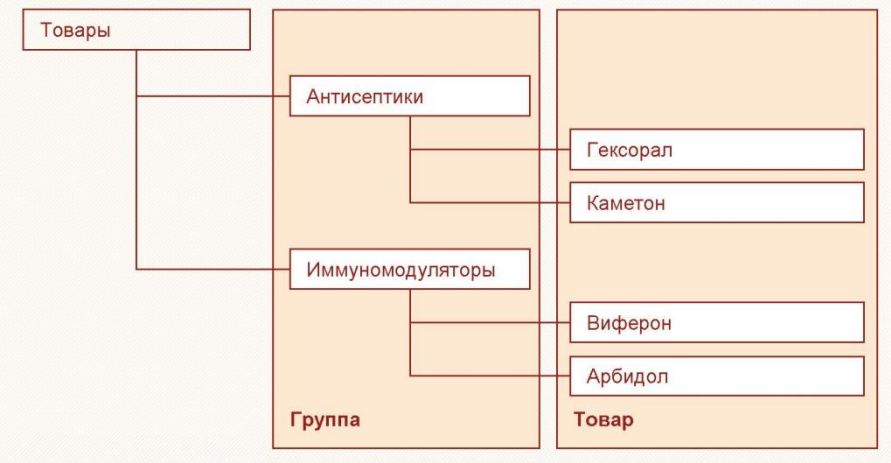


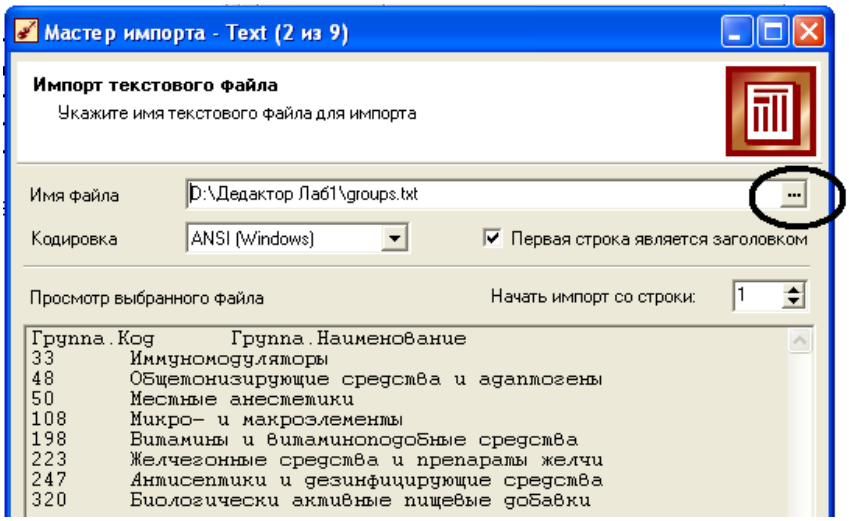
Рис.4. Иерархия измерений.

3. Допускается не загружать отдельно измерения, не имеющие атрибутов и не состоящие в иерархии измерений. Значения таких измерений можно при использовании специальной опции создавать во время загрузки в процесс.

На закладке «Сценарии» правой кнопкой откройте **Сценарии** и выберите **Мастер импорта.**

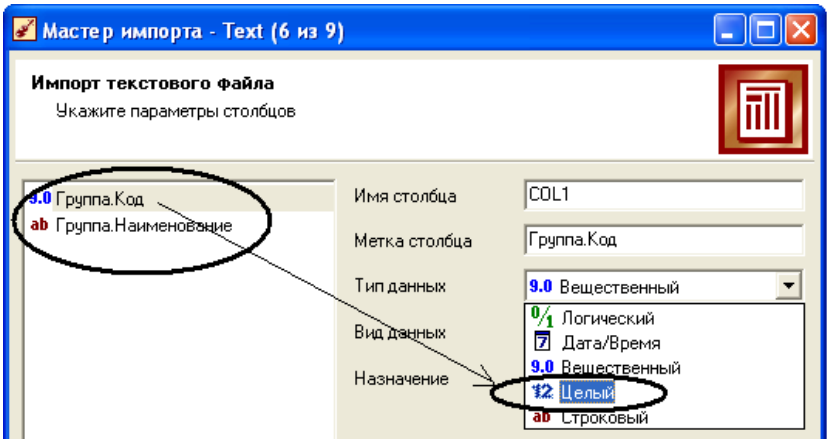
В Файлах данных выберите Text и нажмите**Далее.**

В поле «Имя файла» найдите файл **groups.txt** с данными в папке «Мои документы\Лаб4» и откройте его.



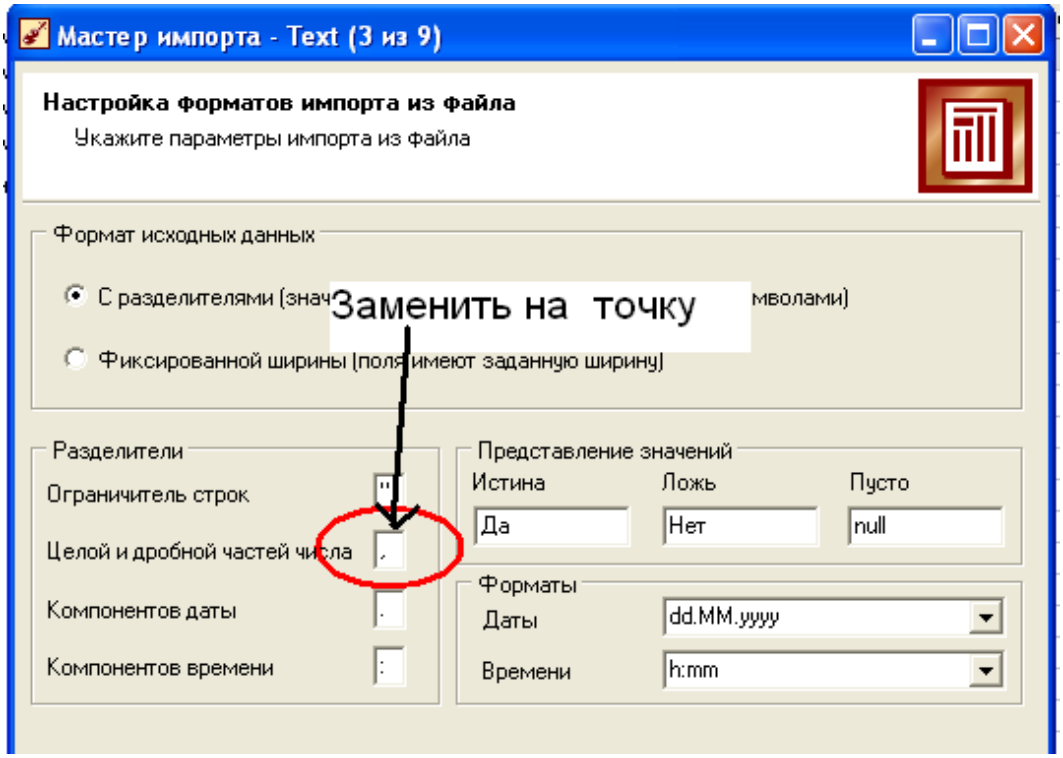
Нажмите**Далее**, **Далее**, **Далее**.

На шестом шаге проверьте и, если необходимо, измените Тип данных столбцов. Для столбца Группа.Код он должен быть целым, а для столбца Группа.Наименование – строковым.



Аналогичным образом последовательно импортируем все 4 текстовых файла в Deductor в следующей последовательности: **groups.txt**,**produces.txt**, **stores.txt**, **sales.txt**. Внимательно установите типы данных у полей Группа.Код,Товар.Код, Отдел.Код, Час.Код- они должны быть целыми.

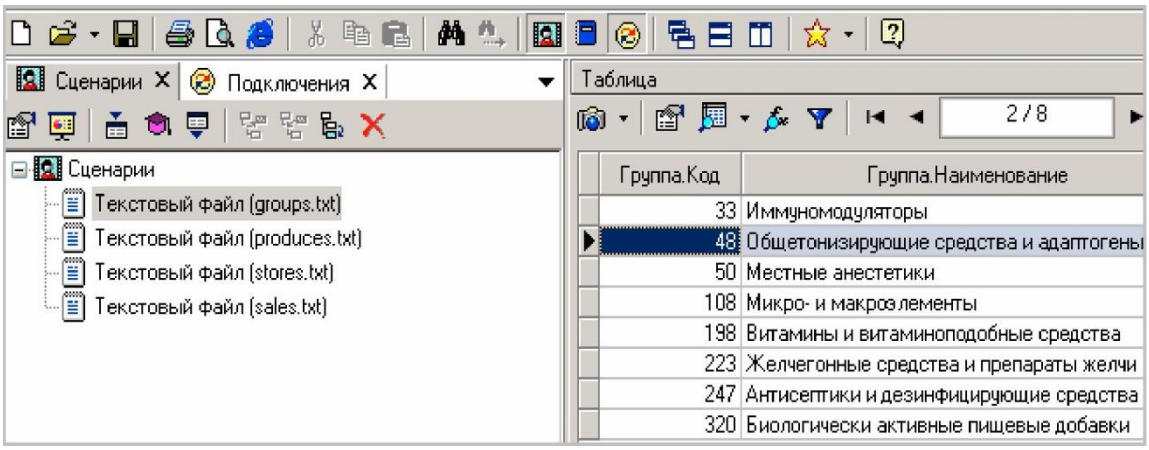
При импорте данных файла **sales.txt** , на третьем шаге в поле «Разделители» в окне «Целой и дробной частей числа» необходимо заменить *запятую* на *точку*. Поскольку в исходном текстовом файле sales.txt в столбце сумма разделителем является именно *точка*. Вернитесь кнопкой*Назад* на предыдущий наг и удостоверьтесь в этом.



На шестом шаге для столбцов Отдел.Код, Товар.Код, Час и Количество укажите тип данных – целый.

Нажмите**Далее, Пуск, Далее, Готово**.

В результате получим сценарий, состоящий из 4 узлов импорта текстовых файлов в DeductorStudio.

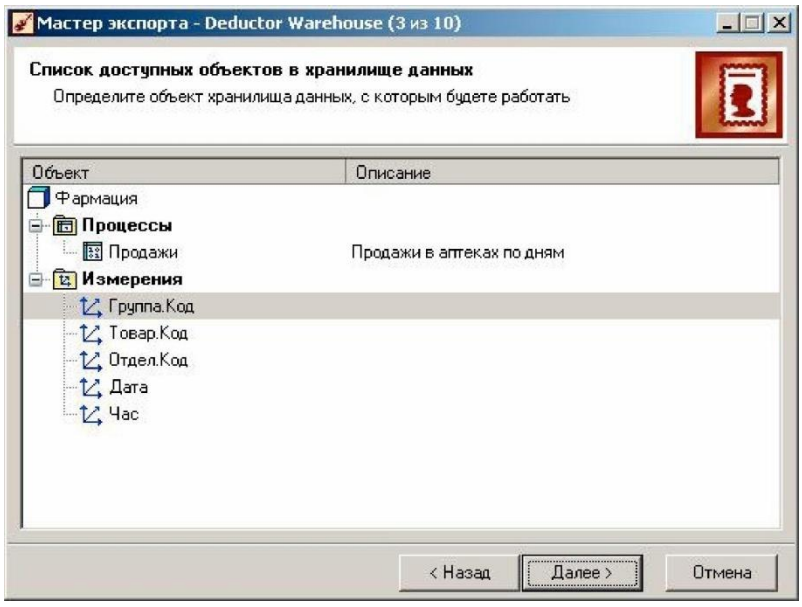


Как уже говорилось выше, первыми следуют таблицы измерений - groups.txt,produces.txt, stores.txt, и только в конце – таблица процесса - sales.txt.

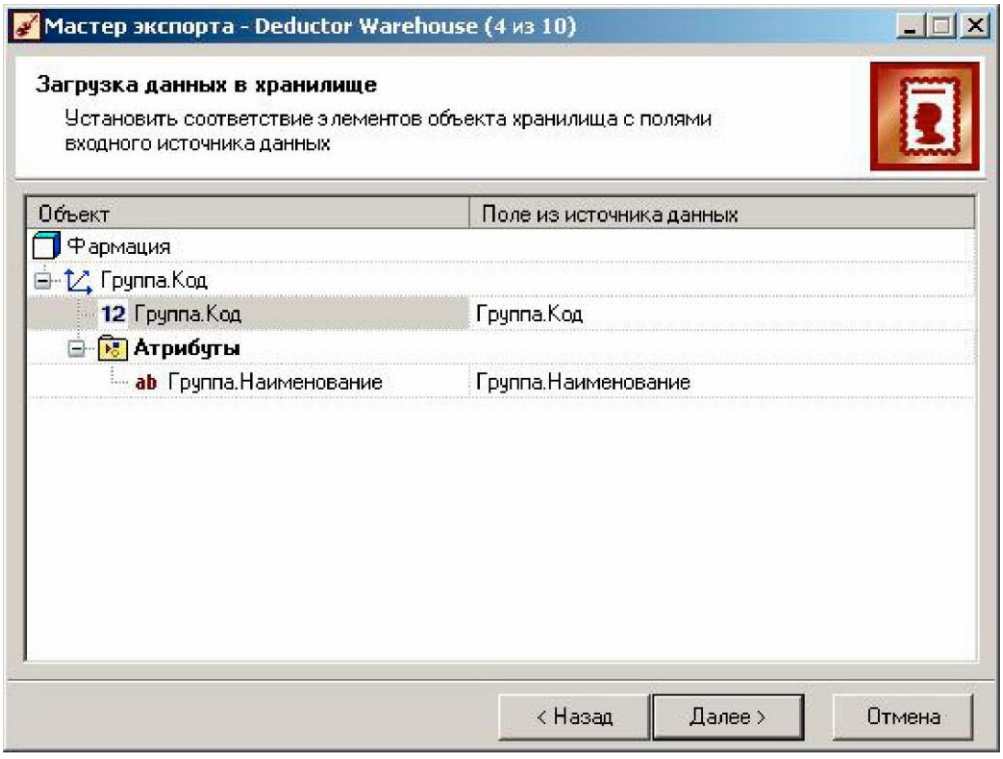
Загрузите эти импортированные данные в измерения.Для этого встав на первом узле сценария, правой кнопкой мыши откройте панель и выберите пункт **Мастер экспорта**. Из списка приемников выберите **DeductorWarehouse**и нажмите **Далее.**

На следующей вкладке из списка доступных хранилищ укажите нужное нам ХД под названием **Фармация** и нажмите **Далее**.

Далее требуется указать, в какое именно измерение будет загружаться информация. Это Группа.Код.

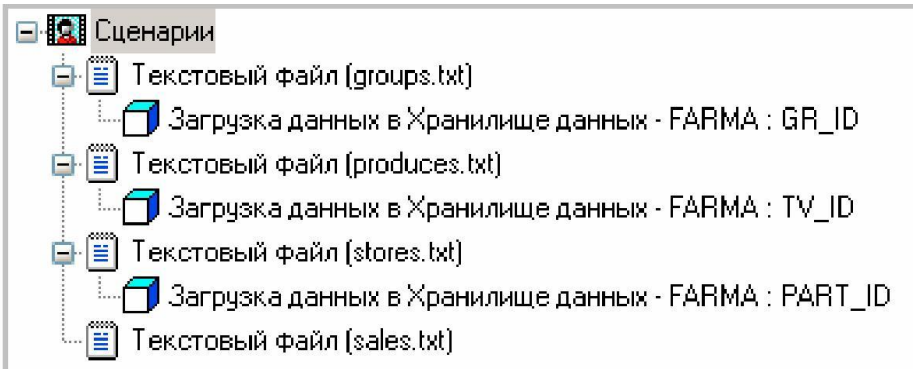


Последнее, что осталось, это установить соответствие элементов объекта в хранилище данных с полями входного источника данных (т.е. таблицы **groups.txt**). В случае, когда имена полей и метки в семантическом слое хранилища данных совпадают, делать ничего не нужно.



Нажмите кнопки**Далее,Пуск** - данные будут загружены в измерение. При этом «старые» данные, если они были, будут обновлены.

Аналогичным образом загрузите данные еще для двух измерений - Отдел.Код, Товар.Код, получим следующий сценарий.

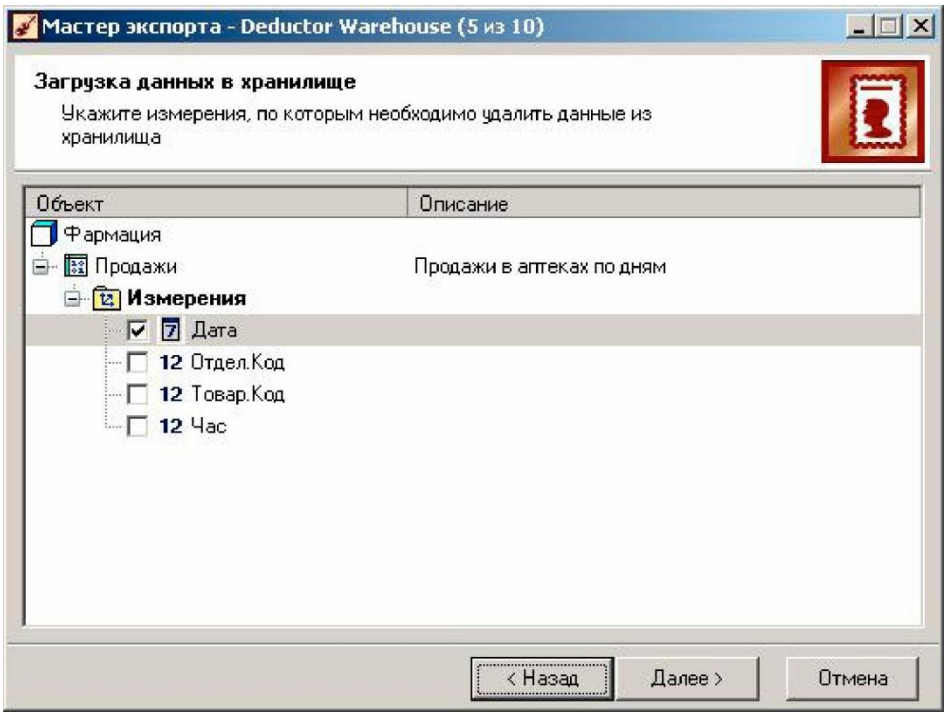


Загрузка измерений на этом заканчивается, несмотря на то, что еще остались два измерения Дата и Час. Но они без атрибутов и не участвуют в иерархии, поэтому их значения можно загрузить на этапе экспорта в процесс.

Сейчас, когда все измерения загружены (т.е. определены все координаты в многомерном пространстве), аналогично загрузите данные в процесс **Продажи.**

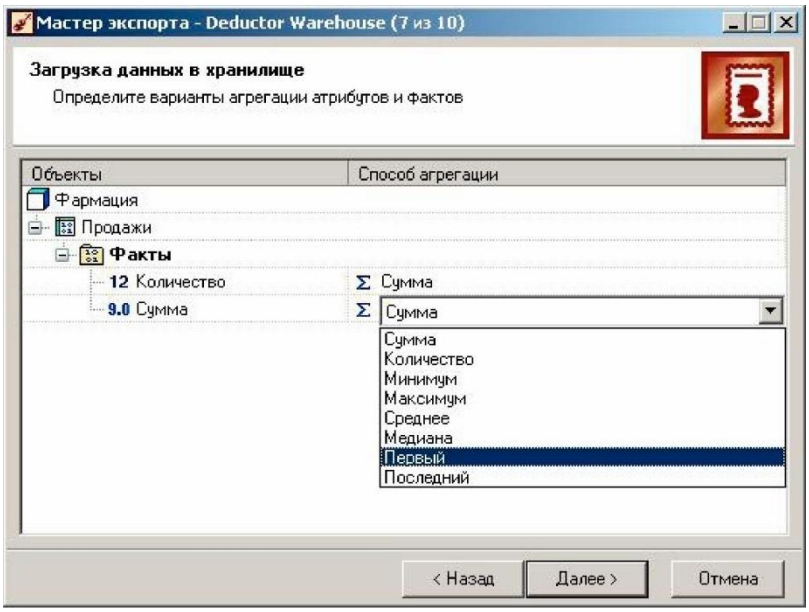
В отличие от загрузки измерений в мастере экспорта появляются два специфических шага.

На одном из них нужно задать параметры контроля непротиворечивости данных в хранилище - указать измерения, по которым следует удалять данные из хранилища. В данном случае это измерение *Дата*. Поставьте галочку напротив этого измерения.



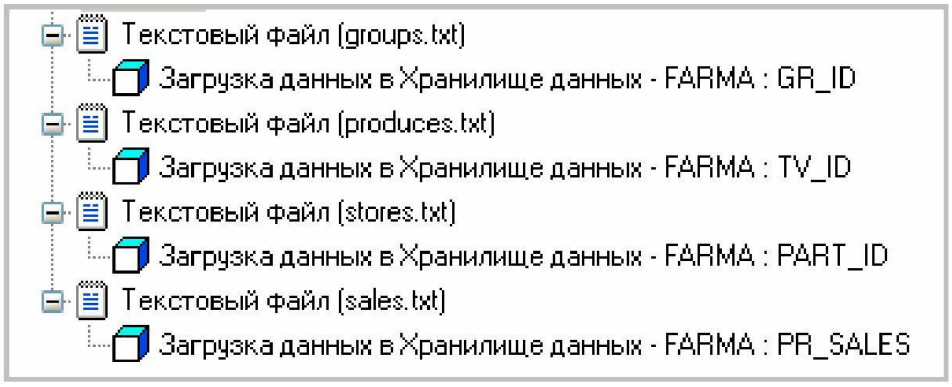
Это действие целесообразно в ситуации, когда в процесс загружается информация, которая совпадает по значениям из нескольких измерений. Вариантов может быть два: удалить «старые» данные и загрузить «новые» или запретить удаление и оставить то, что уже было загружено ранее. Подобный способ загрузки удобен еще и тем, что позволяет избежать коллизий, например, когда в хранилище имеются некорректные данные за какой-то период. В таком случае лучше все данные за этот период удалить, а после загрузить новые корректные сведения.

На втором специфическом шаге (в данном случае его делать не нужно) можно в мастере экспорта можно задать и любой другой вариант агрегации данных.



В случае, когда есть уверенность в том, что совокупность измерений процесса обеспечивает уникальность точки в многомерном пространстве, группировку можно не производить - это сэкономит время загрузки.

В конечном итоге получим окончательный сценарий загрузки данных в хранилище **Фармация**.



Сохраните файл сценария под именем **farma.ded**в папке Мои документы.

Таким образом, врезультате всех вышеописанных действий будет:

1. создано и наполнено хранилище данных;

2. создан сценарий загрузки информации из источников в ХД;
3. продуман контроль непротиворечивости данных при пополнении ХД.

Обратим внимание на то, что сценарий загрузки ***не привязан*** непосредственно к данным. Он привязан к их структуре, т.е. в нем смоделирована последовательность действий, которые нужно выполнить для загрузки информации в ХД: имена файлов-источников, соответствие полей и т.д. Один раз созданный сценарий впоследствии используется повторно для пополнения хранилища данных. Для этого нужно выгрузить новую информацию о продажах и измерениях в текстовые файлы. Как правило, эти процедуры проводятся по регламенту в нерабочее время (например, ночью) с использованием пакетного режима, который недоступен в DeductorAcademic.

Вопросы для проверки (самопроверки):

1. Какова последовательность загрузки информации в хранилище?
2. В каких случаях измерение можно не загружать отдельным узлом экспорта?
3. Какие предусмотрены способы контроля непротиворечивости данных в DeductorWarehouse?

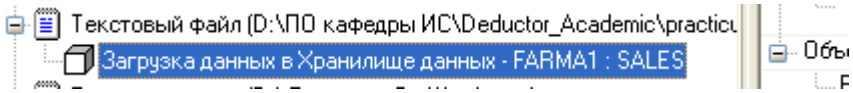
4. Извлечение информации из хранилища данных.

Процесс получения данных из хранилища осуществляется при помощи Мастера импорта. Можно извлекать информацию, так называемые срезы данных, из процесса и (или) измерений.

Извлечение информации из процесса.

Осуществим извлечение данных (сделаем срез) из процесса *Продажи*, например, за последние 3 месяца от имеющихся данных. Для этого проделаем следующие действия.

Выберите загруженные данные процесса Продажи в сценариях.



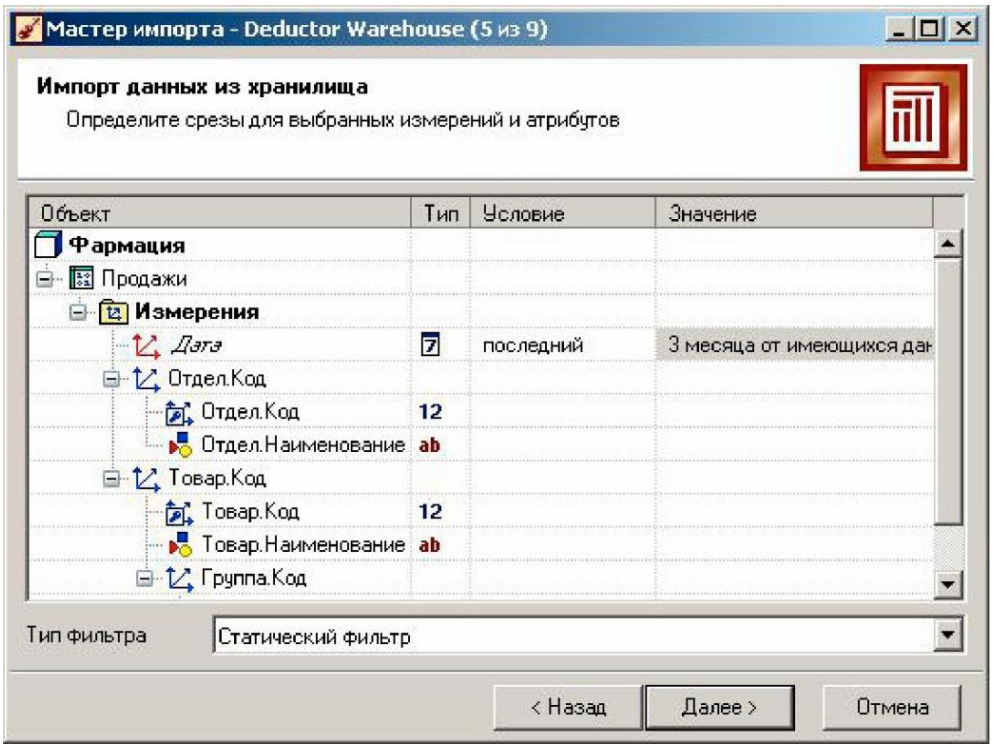
Правой кнопкой мыши откройте меню и выберите **Мастер импорта.**

Выберите тип источника данных- **Deductor Warehouse**, на следующем шаге - ХД **Фармация**, а затем - процесс **Продажи**. Далее задайте, какие измерения и атрибуты из выбранного на предыдущем шаге процесса должны быть импортированы (рис. 7). Заметим, что внутри измерения Товар. Код появилась возможность доступа к измерению Группа. Код.

Рис. 7. Выбор импортируемых измерений и атрибутов.

Определите импортируемые факты и виды их агрегаций В большинстве случаев требуется агрегация в виде суммы.

Определите условие и значение среза «Все продажи за последние 3 месяца от имеющихся данных».



Изменение цвета пиктограммы у объекта хранилища на красный говорит о том, что по нему настроен срез.

Нажмите **Далее**. Убедитесь в правильности среза. Покажите результат преподавателю.

Чтобы задать другие параметры среза для данного измерения или атрибута, необходимо выделить его в дереве объектов, выбрать условие и щелкнуть по кнопке выбора значений-  . В результате откроется диалоговое окно, в котором нужно задать параметры фильтрации.

Список условий содержит несколько значений, основные из которых следующие:

- <*пусто*>- фильтрация по указанному объекту не производится;
- = - указывается значение, которое *нужно* импортировать;
- <>- указывается значение, которое *не нужно* импортировать.
- в *списке* - указываются значения, которые *входят* в список импортируемых;
- *вне списка* — указываются значения, которые *не входят* в список импортируемых.
- *содержит* - указывается подстрока, которая *должна содержаться* в импортируемых значениях измерений;
- *не содержит* - указывается подстрока, которая *не должна содержаться* в импортируемых значениях измерений;
- *начинается на*- указывается подстрока, с которой *должны начинаться* импортируемые значения измерений;
- *начинается на*- указывается подстрока, с которой *не должны начинаться* импортируемые значения измерений;
- *заканчивается на*- указывается подстрока, на которую

должны заканчиваться импортируемые значения измерений;

- *не заканчивается на*- указывается подстрока, на которую *не должны заканчиваться* импортируемые значения измерений.

Для измерений типа *вещественное и целое число* дополнительно доступны следующие основные фильтры:

- <, <=, >, >= - соответствующие операции сравнения чисел;
- в *интервале* - выбираются значения, *попадающие* в заданный числовой интервал;
- *вне интервала* - выбираются значения *за пределами* заданного числового интервала;

Для измерений типа *дата/время* дополнительно доступны следующие основные фильтры:

- в *интервале* - выбираются записи, для которых измерение *лежит в заданном диапазоне* дат;
- *вне интервала* - выбираются записи, для которых измерение *не входит в заданный диапазон дат,*
- *последний* - выбираются записи, для которых измерение лежит в *указанном промежутке времени* (промежуток задается), предшествующем указанной дате;
- *не последний* — выбираются все записи, *кроме тех, для которых измерение лежит в указанном промежутке времени,* предшествующем указанной дате.

Извлечение информации из измерений.

Импорт измерений ничем не отличается от импорта из процесса. Аналогично в мастере импорта выбирается интересующее измерение из списка доступных объектов хранилища. Дальше нужно выбрать атрибуты текущего

измерения, которые будут включены в выходной набор данных и связанные измерения (если существует иерархия измерений). Для этого требуемые атрибуты или связанные измерения помечаются флагом, расположенным слева от имени атрибута.

Неотмеченные объекты не будут включены в итоговый набор.

Задания для проверки (самопроверки):

1. Создать срез «Продажи в аптеке №2 за летние месяцы».
2. Создать срез «Продажи в аптеке №1 с 8 до14 часов».
3. Создать срез «Продажи в аптеке №3 за 4 квартал».

и.т.п.

ЛАБОРАТОРНАЯ РАБОТА №5. АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. АССОЦИАТИВНЫЕ ПРАВИЛА

Тема: **АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. АССОЦИАТИВНЫЕ ПРАВИЛА.**

Учебные вопросы:

9. Введение в ассоциативные правила.
10. Генерация ассоциативных правил.
11. Интерпретация ассоциативных правил.
12. Визуализатор «Что-если» в ассоциативных правилах.

Время: 4 часа.

Место: компьютерный класс.

Материально-техническое обеспечение:

8. ПЭВМ.
9. Аналитическая платформа **Deductor**версии Academic, которую можно скачать с официального сайта фирмы-разработчика BaseGroupLabs (www.basegroup.ru) или с диска, прилагающегося к источнику [4].
10. Файлы с исходными данными в формате txt. Архив файлами можно скачать с сайта фирмы разработчика (http://www.basegroup.ru/download/demoprg/practicum/) или с диска, прилагающегося к источнику [4].

Методические рекомендации по проведению занятия.

- Перед началом занятий необходимо распаковать архив с файлом исходных данных в папку «Мои документы\Лаб5».
- Первый учебный вопрос рекомендуется изучить и законспектировать в рабочую тетрадь.
- Ответы на вопросы для проверки в конце каждого

учебного вопроса могут быть письменными или устными.

1. Введение в ассоциативные правила.

Одним из распространенных аналитических методов является аффинитивный анализ (англ: *affinityanalysis*). Название метода происходит от английского слова affinity – близость, сходство. Целью данного метода является исследование взаимной связи между событиями, которые происходят совместно. Одной из разновидностей аффинитивного анализа является *анализ рыночной корзины* (англ: *marketbasketanalysis*), цель которого – обнаружить ассоциации между различными событиями, т.е. найти правила для количественного описания взаимной связи между двумя или более событиями. Такие правила называются **ассоциативными правилами** (англ.: *associationrules*)[4].

Примерами приложения ассоциативных правил могут быть следующие задачи:

1. Обнаружение наборов товаров, которые в супермаркетах часто покупаются вместе или никогда не покупаются вместе.
2. Определение доли клиентов, положительно относящихся к нововведениям в их обслуживании.
3. Определение профиля посетителя веб-ресурса.
4. Определение доли случаев, в которых новое лекарство показывает опасный побочный эффект.

Базовым понятием в теории ассоциативных правил является транзакция.

Транзакция *– некоторое множество событий, происходящих совместно.*

Типичной транзакцией является приобретение клиентом некоторого товара в супермаркете. В табл. 1 представлен простой пример набора транзакций. В каждой строке содержится комбинация продуктов, приобретенных за одну покупку[4].

Таблица 1. Пример набора транзакций.

№ транзакции	Товары
1	сливы, салат, помидоры
2	сельдерей, конфеты
3	конфеты
4	яблоки, морковь, помидоры, картофель, конфеты
5	яблоки, апельсины, салат.конфеты, помидоры
6	персики, апельсины, сельдерей, помидоры
7	фасоль, салат, помидоры
8	апельсины, салат.помидоры
9	яблоки, сливы, морковь, помидоры, лук, конфеты
10	яблоки, картофель

Хотя на практике приходится иметь дело с миллионами транзакций, в которых участвуют десятки и сотни различных продуктов, данный пример ограничен 10 транзакциями, содержащими 13 видов продуктов, что достаточно для иллюстрации методики обнаружения ассоциативных правил.

В подавляющем большинстве случаев клиент приобретает не один товар, а некоторый набор товаров, который и называется рыночной корзиной. При этом возникает вопрос: является ли покупка одного товара в

корзине следствием или причиной покупки другого товара, т.е. являются ли данные события связанными? Эту связь и устанавливают ассоциативные правила. Например, может быть обнаружено ассоциативное правило, утверждающее, что покупатель, купивший молоко, с вероятностью 75% купит и хлеб.

Анализ рыночной корзины – это анализ наборов данных для определения комбинаций товаров, связанных между собой. Иными словами, производится поиск товаров, присутствие которых в транзакции влияет на вероятность наличия других товаров или комбинаций товаров[4].

Современные кассовые аппараты в супермаркетах позволяют собирать информацию о покупках, которая может храниться в базе данных. Накопленные данные затем могут использоваться для построения систем поиска ассоциативных правил.

Визуальный анализ примера (табл.1) показывает, что все четыре транзакции, в которых фигурирует салат, также включают и помидоры, и что четыре из семи транзакций, содержащих помидоры, также содержат и салат. Салат и помидоры в большинстве случаев покупаются вместе. Ассоциативные правила позволяют обнаруживать и количественно описывать такие совпадения.

Ассоциативное правило состоит из двух наборов предметов, называемых *условие* (англ: *antecedent*) и следствие (англ: *consequent*), записываемых в виде $X \rightarrow Y$, что читается «из X следует Y». Таким образом, ассоциативное правило формулируется в виде «Если условие, то следствие».

Условие часто ограничивают содержанием только одного предмета. Правила обычно отображаются с помощью

стрелок, направленных от условия к следствию, например, (*помидоры*) → (*салат*). Условие и следствие часто называются соответственно левосторонним (LHS – left-handside) и правосторонним (RHS – right-handside) компонентом ассоциативного правила.

Ассоциативные правила описывают связь между наборами предметов, соответствующим условию и следствию. Эта связь характеризуется двумя показателями – поддержкой и достоверностью.

Обозначим *D*как базу данных транзакций, а *N*как число транзакций в этой базе. Каждая транзакция D_iпредставляет собой некоторый набор предметов. Зададим, что *S*(англ.: *support*) – поддержка, *C* (англ.: *confidence*) – достоверность.

Поддержка ассоциативного правила *– это число транзакций, которые содержат как условие, так и следствие.*

Например, для ассоциации *A*→*B*можно записать:

$$S(A \to B) = P(A \cap B) = \frac{\text{количество транзакций, содержащих A и B}}{\text{общее количество транзакций}}.$$

Достоверность ассоциативного правила- *это мера точности правила, которая определяется как отношение количества транзакций, содержащих как условие, так и следствие, к количеству транзакций, содержащих только условие.*

Например, для ассоциации *A*→*B*можно записать:

$$C(A \to B) = P(A \mid B) = P(A \cap B)/P(A) = \frac{\text{количество транзакций, содержащих A и B}}{\text{количество транзакций, содержащих только A}}.$$

Если поддержка и достоверность достаточно высоки, то это позволяет с большой вероятностью утверждать, что любая будущая транзакция, которая включает условие, будет также содержать и следствие.

Рассмотрим пример для вычисления поддержки и достоверности для ассоциаций из табл.1. Возьмем ассоциацию *(салат) →(помидоры)*. Поскольку количество транзакций, содержащее как *(салат),* так и *(помидоры),* равно 4, а общее число транзакций 10, то поддержка данной ассоциации будет:

$S((салат) \rightarrow (помидоры)) = 4/10 = 0{,}4$.

Поскольку количество транзакций, содержащее только *(салат)* как условие, равно 4, то достоверность данной ассоциации будет:

$C((салат) \rightarrow (помидоры)) = 4/4 = 1.$

Иными словами, все наблюдения, содержащие *салат,* также содержат и *помидоры,* что позволяет сделать вывод о том, что данная ассоциация может рассматриваться как правило. С точки зрения интуитивного поведения такое правило вполне объяснимо, поскольку оба продукта широко используются для приготовления растительных блюд и часто покупаются вместе.

Теперь рассмотрим ассоциацию *(конфеты) →(помидоры),* в которой содержатся, в общем-то, слабо совместимые в гастрономическом плане продукты (тот, кто планирует сделать растительное блюдо, вряд ли станет покупать конфеты, а покупатель, желающий приобрести что-нибудь к чаю, скорее всего, не станет покупать помидоры). Поддержка данной ассоциации $S = 3/10 = 0{,}3$, а достоверность$\boldsymbol{C}= 3/7 = 0{,}43$. Таким образом, сравнительно

невысокая достоверность данной ассоциации дает повод усомниться в том, что она является правилом.

Аналитики могут отдавать предпочтение правилам, которые имеют только высокую поддержку или только высокую достоверность, либо, что является наиболее частым, оба эти показателя. Правила, для которых значения поддержки или достоверности превышают некоторый, заданный пользователем порог, называются *сильными правилами* (strongrules). Например, аналитика может интересовать, какие товары в супермаркете, покупаемые вместе, образуют ассоциации с минимальной поддержкой 20% и минимальной достоверностью 70 %. С другой стороны, при анализе с целью обнаружения мошенничеств, аналитику может потребоваться уменьшение поддержки до 1%, поскольку сравнительно небольшое число транзакций являются связанными с мошенничеством.

Значимость ассоциативных правил

Методики поиска ассоциативных правил обнаруживают все ассоциации, которые удовлетворяют ограничениям на поддержку и достоверность, наложенные пользователем. Это часто приводит к необходимости рассмотреть десятки и сотни тысяч ассоциаций, что делает невозможным «ручную» обработку такого большого количества данных. Очевидно, что желательно уменьшить число правил таким образом, чтобы проанализировать только наиболее значимые правила. Часто значимость связана с разностью между поддержкой правила в целом и произведением поддержки только условия и поддержки только следствия.

Можно выделить объективные и субъективные меры значимости правил. Объективными являются такие меры, как поддержка и достоверность, которые могут применяться

независимо от конкретного приложения. Субъективные меры связаны со специальной информацией, определяемой пользователем в контексте решаемой задачи. Такими субъективными мерами являются **лифт** (англ: *lift)* и **левередж**(от англ. *leverage*- плечо, рычаг).

Лифт *- это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом.*

Лифт (оригинальное название - *интерес)* определяется следующим образом:

$$L(A \rightarrow B) = C(A \rightarrow B)/S(B).$$

Значения лифта б*о*льшие, чем единица показывают, что условие более часто появляется в транзакциях, содержащих и следствие, чем в остальных. Можно сказать, что лифт является обобщенной мерой связи двух предметных наборов: при значениях лифта >1 связь положительная, при 1 она отсутствует, а при значениях <1 -отрицательная.

Другой мерой значимости правила является **левередж**(англ: *leverage;* предложена Г. Пятецким-Шапиро):

Левередж*- это разность между наблюдаемой частотой, с которой условие и следствие появляются совместно (т.е., поддержкой ассоциации), и произведением частот появления (поддержек) условия и следствия по отдельности.*

$$\boldsymbol{L}(A \rightarrow B) = S(A \rightarrow B) - S(A)S(B).$$

Такие меры, как *лифт* и *левереджм*огут использоваться для последующего ограничения набора рассматриваемых

ассоциаций путем установки порога значимости, ниже которого ассоциации отбрасываются.

2. Генерация ассоциативных правил.

В DeductorStudio для решения задач ассоциации используется обработчик **Ассоциативные правила**. В нем реализован алгоритм apriori. Обработчик требует на входе два поля: идентификатор транзакции и элемент транзакции. Например, идентификатор транзакции - это номер чека или код клиента. А элемент - это наименование товара в чеке или услуга, заказанная клиентом.

Оба поля (идентификатор и элемент транзакции) должны быть дискретного вида.

После работы обработчика по умолчанию предлагается визуализатор *Правила*(его структура подробнее будет рассматриваться далее). Вся остальная дополнительная информация, располагается в специализированных визуализаторах *Популярные наборы*, *Дерево правил*, *Что-если*.

Рассмотрим пример решения конкретной задачи ассоциации из области розничной торговли.

Розничная сеть по продаже домашних товаров поставила задачу анализа покупательских корзин для оптимизации их размещения на витринах и проведения кросс-продаж. Отдел маркетинга предоставил несколько тысяч чеков, в которых отражены покупки сделанные предыдущими клиентами магазинов. Стоит следующая задача:

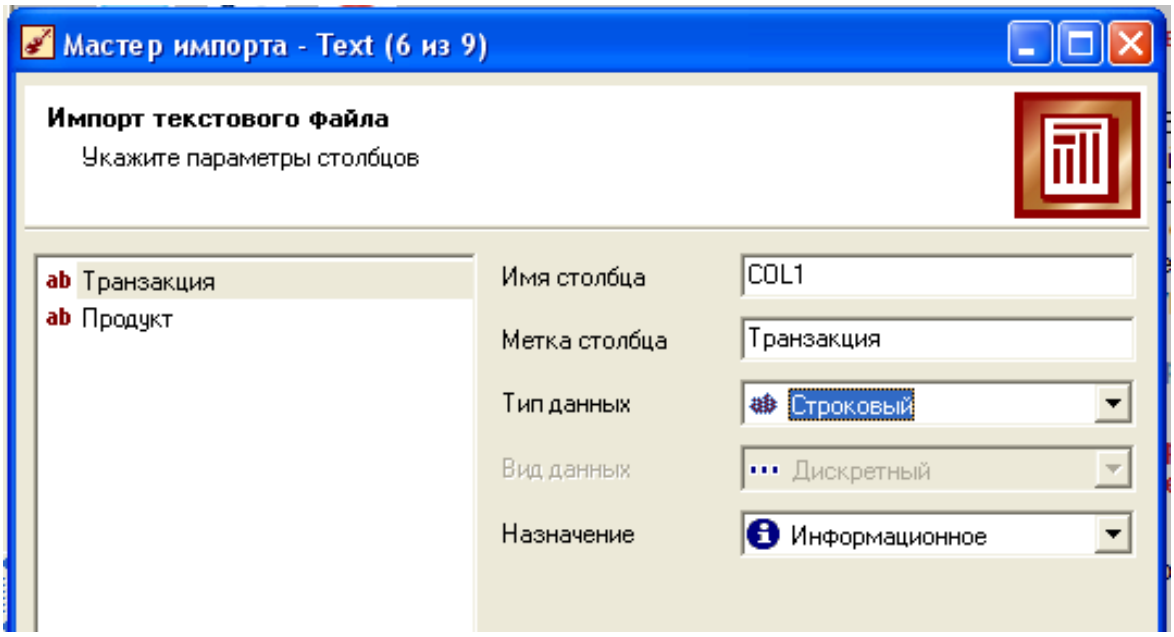
- предсказать то, какие товары покупатели могут выбрать в зависимости от того, что уже есть в их корзинах;

- выявить наиболее популярные товарные наборы, состоящие из более, чем 1 предмета;
- предложить рекламные акции типа «Каждому купившему товары A и B, товар C в подарок».

Откройте программу DeductorStudio, используя ярлык на рабочем столе Deductor Studio или через кнопку Пуск.

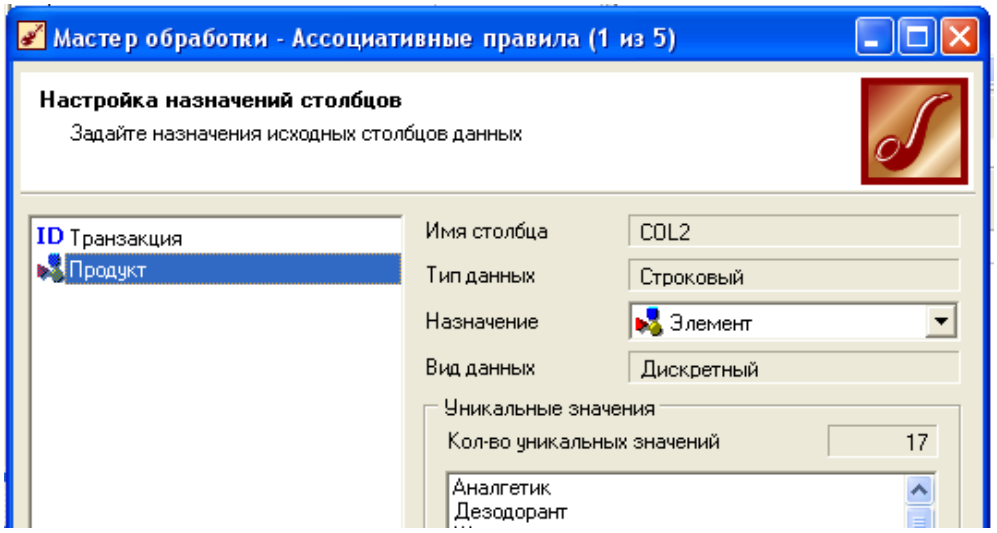
Данныепо транзакциямнаходятся в текстовом файле **transactions.txt**[4]в папке «Мои документы\Лаб5».

Импортируйте данные из текстового файла transactions.txt в DeductorStudio. В файле данных имеются два столбца*Транзакция* и *Продукт*, для которых нужно Тип

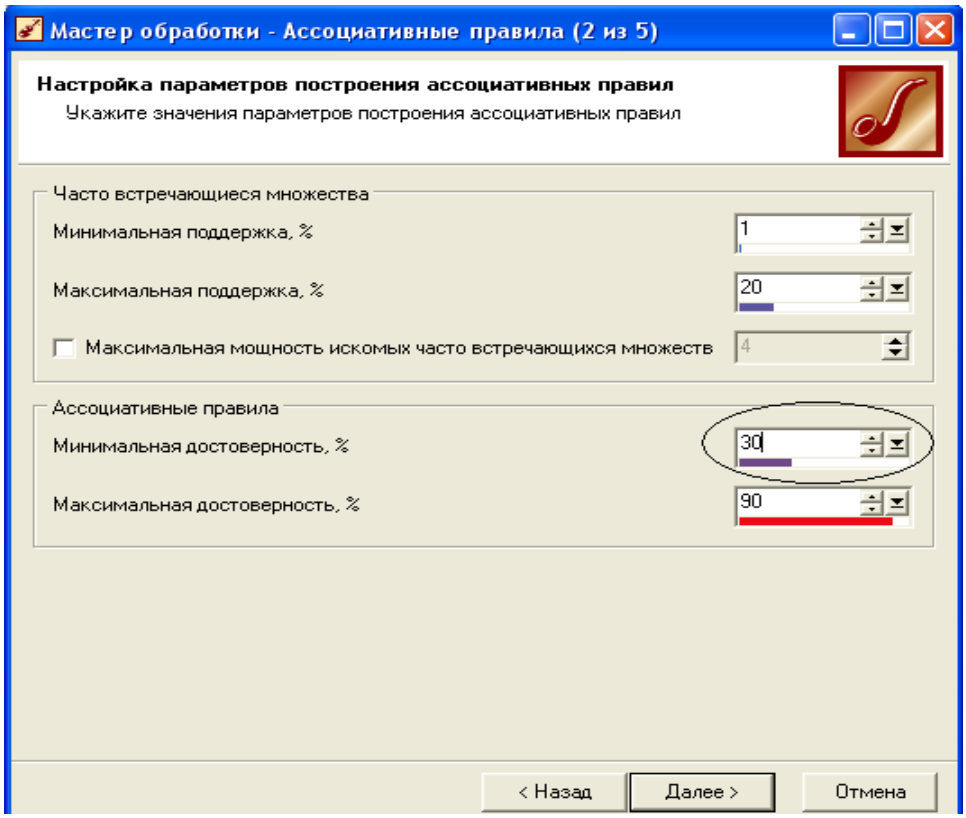


поля нужно установить строковый.

После импорта к данному загруженному файлу применим обработчик**Ассоциативные правила**. Столбец *Транзакция* сделаем идентификатором транзакции, а столбец*Продукт* – ее элементом:



На следующем шаге мастера настроим параметры построения ассоциативных правил, что, по сути, есть



параметры алгоритма apriori:

Здесь для изменения доступны следующие параметры.

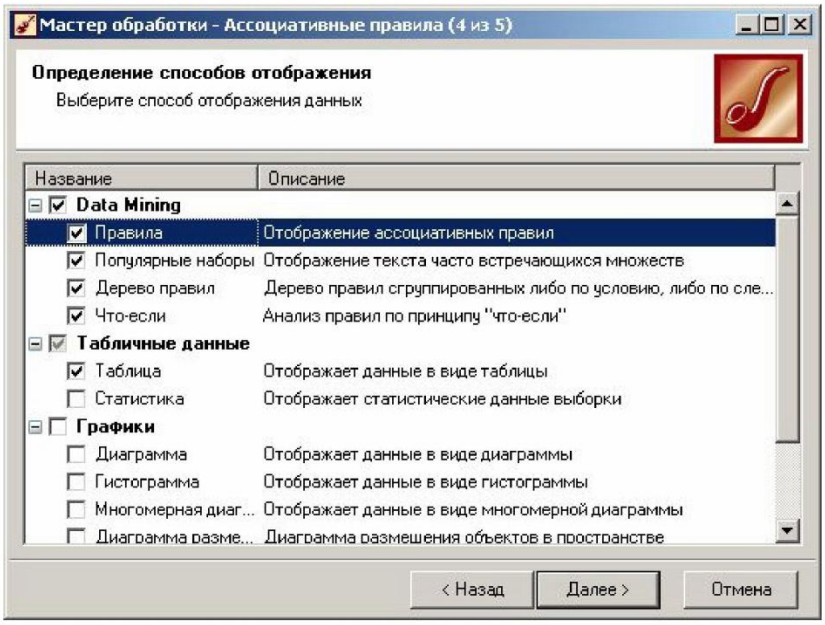
Минимальная и максимальная поддержка в % – ограничивают пространство поиска часто встречающихся предметных наборов. Эти границы определяют множество популярных наборов, из которых и будут создаваться ассоциативные правила.

Минимальная и максимальная достоверность в % – в результирующий набор попадут только те ассоциативные правила, которые удовлетворяют условиям минимальной и максимальной достоверности.

Максимальная мощность искомых часто встречающихся множеств – параметр ограничивает длину *k*-предметного набора. Например, при установке значения 4 шаг генерации популярных наборов будет остановлен после получения множества 4-предметных наборов. В конечном итоге это позволяет избежать появления длинных ассоциативных правил, которые трудно интерпретируются.

Нажмите на кнопку **Пуск,** что приведетк работе алгоритма поиска ассоциативных правил. По окончании его работы справа в полях появится следующая информация:

Далее выбираем все доступные специализированные визуализаторы DataMiningи визуализатор Таблица:



Все эти визуализаторы, кроме **Что-если**, отображают результаты работы алгоритма в различных формах. Рассмотрим их подробнее.

На вкладке **Правила** помимо самих ассоциативных правил приводятся их основные расчетные характеристики:

Правила | Популярные наборы | Дерево правил | Что-если | Таблица

Правил: 6 из 6 | Фильтр: Без фильтрации

№	Номер правила	Условие	Следствие	Поддержка		Достоверность	Лифт
				Кол-во	%		
1	1	Зубная щетка	Парфюм	446	2,23	35,20	3,952
2	2	Зубная паста Карандаши	Конфеты	218	1,09	45,04	2,618
3	3	Карандаши Конфеты	Зубная паста	218	1,09	33,90	2,125
4	4	Зубная паста Конфеты	Поздравительная открыт	272	1,36	34,61	2,288
5	5	Зубная паста Поздравительная открыт	Конфеты	272	1,36	40,60	2,360
6	6	Конфеты Поздравительная открыт	Зубная паста	272	1,36	30,63	1,920

поддержка, достоверность и лифт:

На вкладке **Популярные наборы**отображается множество найденных популярных предметных наборов в виде списка. Кнопка  предлагает на выбор несколько

вариантов сортировки списка, а кнопка  * вызывает окно настройки фильтра множеств. Например, задав в фильтре минимальное значение поддержки 3% и отсортировав их по убыванию поддержки, получим 17 популярных наборов (на картинке изображено только 12):

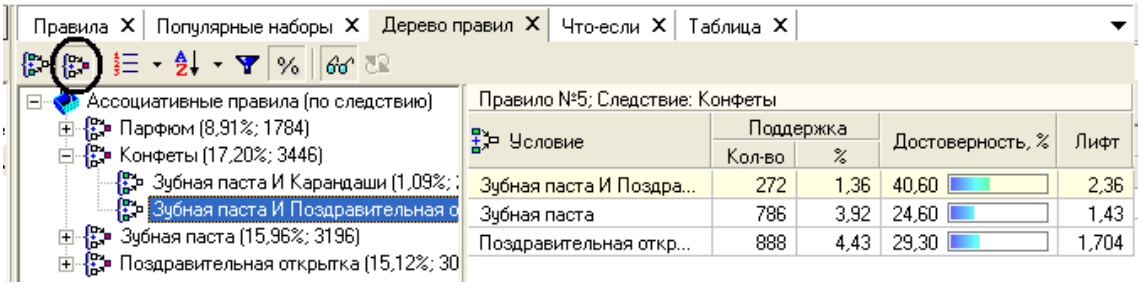
Правила | Популярные наборы | Дерево правил | Что-если | Таблица

Множеств: 17 из 26 | Фильтр: Минимальная поддержка = 3,00

№	Номер множества	Элементы	Поддержка		Мощность
			Кол-во	%	
1	6	Конфеты	3446	17,20	1
2	2	Зубная паста	3196	15,96	1
3	13	Поздравительная открытка	3029	15,12	1
4	10	Набор ручек	2922	14,59	1
5	4	Карандаши	2714	13,55	1
6	12	Парфюм	1784	8,91	1
7	3	Зубная щетка	1267	6,33	1
8	5	Карта флеш-памяти	1198	5,98	1
9	7	Лекало	1093	5,46	1
10	11	Оберточная бумага	1025	5,12	1
11	24	Конфеты Поздравительная открытка	888	4,43	2
12	9	Мыло	832	4,15	1

На вкладке **Дерево правил** предлагается еще один удобный способ отображения множества ассоциативных правил, которое строится либо *по условию*, либо *по следствию*. При построении дерева правил по условию, на первом (верхнем) уровне находятся узлы с условиями, а на втором уровне – узлы со следствием. В дереве, построенном по следствию, наоборот, на первом уровне располагаются узлы со следствием.

Справа от дерева расположен список правил, построенный по выбранному узлу дерева, например по правилу №5:



Для каждого правила отображаются поддержка и достоверность и лифт. Если дерево построено по условию, то вверху списка отображается условие правила, а список состоит из его следствий. Тогда правила отвечают на вопрос, что будет при таком условии. Если же дерево построено по следствию, то вверху списка отображается следствие правила, а список состоит из его условий. Эти правила отвечают на вопросы, что нужно, чтобы было заданное следствие или какие товары нужно продать для того, чтобы продать товар из следствия.

Сохраните сценарий под именем **assosiation.ded**.

Поэкспериментируйте с настройкой фильтра (кнопка ) в визуализаторах **Правила**, **Популярные наборы** и **Дерево правил**.

Вопросы для проверки:

1. Какой алгоритм генерации ассоциативных правил имеется в Deductor?
2. Какие входные поля набора данных необходимы для запуска обработчика Ассоциативные правила в Deductor?
3. Какие специализированные визуализаторы предлагаются к узлу-обработчику Ассоциативные правила?

3. Интерпретация ассоциативных правил.

Теперь остановимся на наиболее важном этапе – интерпретации ассоциативных правил. Дело в том, что ассоциативные правила сами по себе, как результат работы некоторого алгоритма, еще не готовы к использованию. Их нужно интерпретировать, т.е. понять, какие из ассоциативных правил представляют интерес, действительно ли правила отражают закономерности или наоборот являются артефактом. Это требует тщательной работы аналитика и понимания предметной области, в которой решается задача ассоциации[4].

Все множество ассоциативных правил можно разделить на три вида:

Полезные правила– содержат действительную информацию, которая ранее была неизвестна, но имеет логичное объяснение. Такие правила могут быть использованы для принятия решений, приносящих выгоду.

Тривиальные правила– содержат действительную и легко объяснимую информацию, которая уже известна. Такие правила, хотя и объяснимы, но не могут принести какой-либо пользы, т.к. отражают или известные законы в исследуемой области, или результаты прошлой деятельности. При анализе рыночных корзин в правилах с самой высокой поддержкой и достоверностью окажутся товары-лидеры продаж. Практическая ценность таких правил крайне низка.

Непонятные правила– содержат информацию, которая не может быть объяснена. Такие правила могут быть получены или на основе аномальных значений, или глубоко скрытых знаний. Напрямую такие правила нельзя использовать для принятия решений, т.к. их необъяснимость

может привести к непредсказуемым результатам. Для лучшего понимания требуется дополнительный анализ.

Варьируя верхним и нижним пределами поддержки и достоверности, можно избавиться от очевидных и неинтересных закономерностей. Как следствие, правила, генерируемые алгоритмом, принимают приближенный к реальности вид. Понятия «верхний» и «нижний» предел очень сильно зависят от предметной области, поэтому не существует четкого алгоритма их выбора. Но есть ряд общих рекомендаций.

Полезные советы при интерпретации правил

- Большая величина параметра Максимальная поддержка означает, что алгоритм будет находить хорошо известные правила, или они будут настолько очевидными, что в них нет никакого смысла. Поэтому ставить порог максимальная поддержка очень высоким (более 20%) не рекомендуется.
- Большинство интересных правил находится именно при низком значении порога поддержки, хотя слишком низкое значение поддержки ведет к генерации статистически необоснованных правил. Поэтому правила, которые кажутся интересными, но имеют низкую поддержку, дополнительно анализируйте по лифту, а при необходимости рассчитывайте для них левередж и улучшение.
- Ограничивайте мощность часто встречающихся множеств – правила с большим числом предметов в условии трудно интерпретируются и воспринимаются.
- Уменьшение порога достоверности приводит к увеличению количества правил. Значение минимальной достоверности не должно быть слишком маленьким, так как ценность правила с достоверностью 5% чаще всего

настолько мала, что это и правилом считать нельзя.

- Правило с очень большой достоверностью (>85-90%) практической ценности в контексте решаемой задачи не имеет, т.к. товары, входящие в следствие, покупатель покупает всегда.

Например, первое правило «**Зубная паста → Парфюм»** имеет *S*= 2,23%; *C*= 35,2% и *L* = 3,95.

Это означает следующее:

- ожидаемая вероятность покупки набора «Зубная паста + Парфюм» равна 2,23%;
- если клиент положил в корзину товар «Зубная паста», то с вероятностью 35,2% он купит и товар «Парфюм»;
- клиент, купивший «Зубная паста», в 3,9 раз чаще выберет «Парфюм», нежели любой другой товар.

Анализ полученных правил позволяет придти к выводу, что многие из них тривиальны, поэтому имеют высокую достоверность.

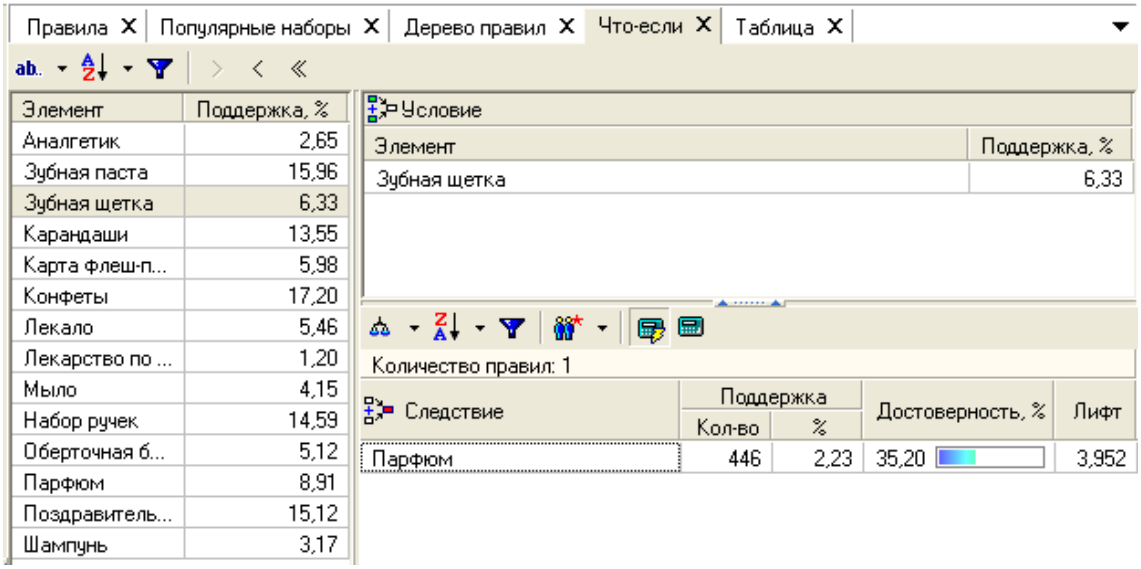
Вопросы для проверки:

1. Какова вероятность того, что клиент, купивший Карандаши и конфеты, купит и Зубную пасту?
2. Приведите пример, когда после анализа ассоциативного правила, товары размещают на большом расстоянии друг от друга или их не советуют приобретать вместе.

4. Визуализатор «Что-если» в ассоциативных правилах.

Кроме уже изученных визуализаторов *Правила, Популярные наборы* и*Дерево правил*, в DeductorStudio к узлу ***Ассоциативные правила***доступен визуализатор **Что-если**. Он позволяет ответить на вопрос, что мы получим в качестве

следствия, если выберем данные условия, например какие товары, приобретаются совместно с выбранными товарами:



В окне слева расположен список всех элементов транзакций. Справа от каждого элемента указана поддержка: сколько раз данный элемент встречается в транзакциях.

В правом верхнем углу расположен список элементов, входящих в условие, выбирается он спомощью нажатия мышки или вспомогательных кнопок. Это, например, списоктоваров, которые приобрел покупатель. Для них можно найти следствие, нажав на кнопку  **Вычислить правила**. Причем в условие могут входить несколько элементов, или товаров в данном случае. Тогда в следствие попадут все товары, условия которых удовлетворяют списку ассоциативных правил. Информация о купленных товарах вносится продавцом в отчет и тут же формируется набор предложений, который озвучивается клиенту.

С помощью встроенных возможностей можно настроить интерактивный отчет возможных предложений-напоминаний клиенту. Для этого используются следующие кнопки:

 **Автоматически вычислить правила** – позволяет рассчитывать новый набор следствий после каждого добавления товара в набор условий;

 **Порядок сортировки**,  **Направление сортировки** – позволяютупорядочить «список предложений» по одному из выбранных параметров (поддержка, достоверность, лифт);

 **Фильтрация правил** – задаются ограничения на формируемый «список предложений»;

 **Тип определения лучшего правила** – при повторении наименований товаров в «списке предложений» позволяет отображать, один из них, в соответствии с лучшей характеристикой.

*Задание:*определите, что приобретет клиент купивший: зубную пасту, зубную щетку и конфеты с величиной лифта больше 3.

Вопросы для проверки (самопроверки):

1. Какой визуализатор используется для формирования предложений клиенту?
2. С помощью, каких кнопок можно отсортировать правилапо убыванию лифта?

ЛАБОРАТОРНАЯ РАБОТА №6. АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. ПРОГНОЗИРОВАНИЕ

Тема: **АНАЛИТИЧЕСКАЯ ПЛАТФОРМА DEDUCTOR. ПРОГНОЗИРОВАНИЕ.**

Учебные вопросы:

13. DataMining в задачах прогнозирования.
14. Создание сценария прогнозирования объема продаж.
15. Прогнозирование суммы продаж.
16. Прогнозирование количества проданного товара.

Время: 4 часа.

Место: компьютерный класс.

Материально-техническое обеспечение:

11. ПЭВМ.
12. Аналитическая платформа **Deductor**версии Academic, которую можно скачать с официального сайта фирмы-разработчика BaseGroupLabs (www.basegroup.ru) или с диска, прилагающегося к источнику [4].
13. Файлы с исходными данными в формате txt. Архив с файлами можно скачать с сайта http://pgsha.ru в разделе учебно-методической работы кафедры информационных систем.

Методические рекомендации по проведению занятия.

- Перед началом занятий необходимо распаковать архив с файлами исходных данных в папку «Мои документы\Лаб6».
- Первый учебный вопрос рекомендуется изучить и

законспектировать в рабочую тетрадь.

- Ответы на вопросы для проверки в конце каждого учебного вопроса могут быть письменными или устными.

1. DataMining в задачах прогнозирования.

Для решения задач регрессии, одной из которых является задача прогнозирования, в Deductor используются различные обработчики[4].

Обработчик «Автокорреляция». Целью автокорреляционного анализа является выяснение степени статистической зависимости между различными значениями (отсчетами) случайной последовательности, которую образует поле выборки данных. В процессе автокорреляционного анализа рассчитываются коэффициенты корреляции (мера взаимной зависимости) для двух значений выборки, отстоящих друг от друга на определенное количество отсчетов, называемые также лагом.

Применительно к анализу временных рядов автокорреляция позволяет выделить месячную и годовую сезонность в данных (рис.1). Видно, что пик зависимости на данных приходится на 12 месяц, что свидетельствует о годовой сезонности. Поэтому величину продаж годовой давности необходимо обязательно учитывать при построении модели (если используется нейронная сеть – то подавать на вход).

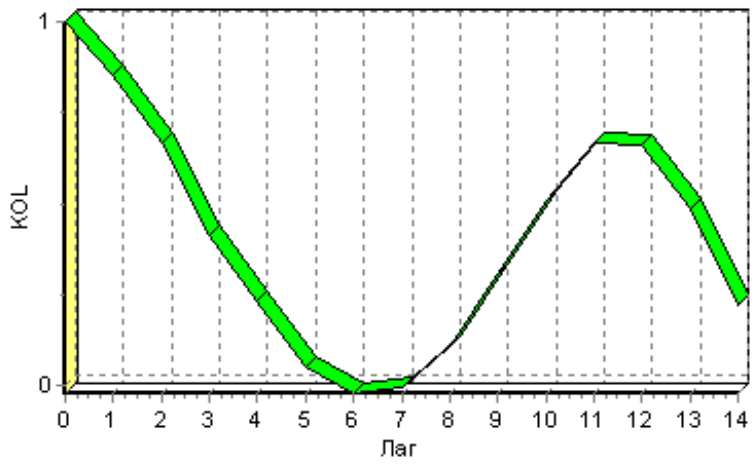


Рис. 1. Автокорреляция.

Обработчик «Парциальная предобработка». Данный обработчик позволяет удалить аномалии и шумы в исходных данных. Шумы в данных (в данном случае временной ряд истории продаж) не только скрывают общую тенденцию, но и проявляют себя при построении модели прогноза. Из-за них модель может получиться плохого качества. Аналогичная ситуация с аномалиями, т.е. резкими отклонениями величины от ее ожидаемого значения.

Однако следует понимать, что решение о сглаживании, удалении шумов и аномалий целиком зависит от специфики решаемой задачи, бизнес-правил предметной области и т.д. Аномалии, или случайные всплески в отгрузке товаров могут быть вызваны такими факторами, как задержка транспорта в пути и другими форс-мажорами. В любом случае, необходимо строить несколько моделей.

В обработчике «Парциальная обработка» предусмотрен выбор степени подавления аномалий и вычитания шумов: малая, средняя, большая. Кроме того, в обработчике предусмотрена возможность заполнения пропусков, а также сглаживание данных, в том числе вейвлет-преобразованием.

Выбор того или иного алгоритма определяется спецификой конкретной задачи.

На рис.2. приведен пример временного ряда после удаления аномалий и шумов.

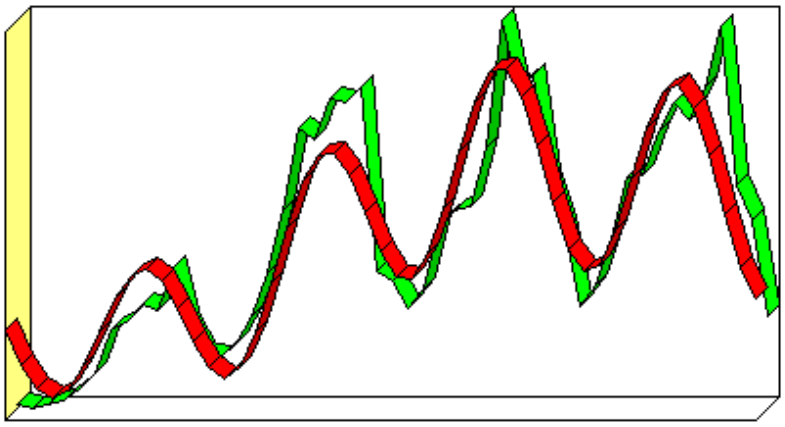


Рис. 2. Исходные и сглаженные данные.

Обработчик «Скользящее окно». При прогнозировании временных рядов при помощи обучающихся алгоритмов (в том числе искусственной нейронной сети), требуется подавать на вход анализатора значения нескольких, смежных, отсчетов из исходного набора данных. Например, в том случае, когда необходимо подавать на вход значения сумм продаж за последние 3 месяца и сумму продаж год назад. При этом эффективность реализации заметно повышается, если не выбирать данные каждый раз из нескольких последовательных записей, а последовательно расположить данные, относящиеся к конкретной позиции окна, в одной записи.

Значения в одном из полей записи будут относиться к текущему отсчету, а в других – смещены от текущего отсчета «в будущее» или «в прошлое». Следовательно, преобразование скользящего окна имеет два параметра: «глубина погружения» - количество «прошлых» отсчетов

(включая текущий отсчет), попадающих в окно, и «горизонт прогнозирования» – количество «будущих» отсчетов. Такой метод отбора данных называется скользящим окном (поскольку это окно «перемещается» по всему набору).

Обработчик «Нейросеть». Обработчик предназначен для решения задач регрессии и прогнозирования. В данном случае нейросеть строится для прогнозирования будущих значений временного ряда. Для проверки обобщающей способности нейросети рекомендуется разбить имеющееся множество данных на две части: обучающее и тестовое. Как правило, при прогнозировании временных рядов, доля тестового множества составляет не более 10-20%.

С помощью визуализатора «Диаграмма» оценивается способность построенной нейросетевой модели к обобщению. Для этого в одном окне выводятся графики исходного и спрогнозированного временных рядов. На рис.3 изображен пример таких графиков. Видно, что построенная модель обеспечивает приемлемую точность.

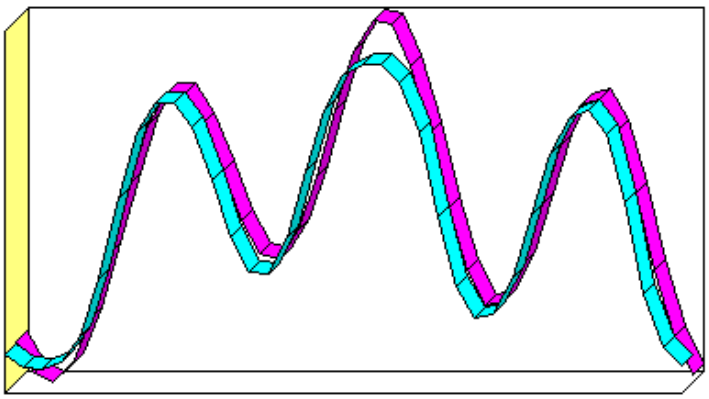


Рис. 3. Оценка качества построенной модели.

Обработчик «Прогнозирование» и визуализатор «Диаграмма прогноза». Прогнозирование позволяет

получать предсказание значений временного ряда на число отсчетов, соответствующее заданному горизонту прогнозирования. Алгоритм прогнозирования работает следующим образом. Пусть в результате преобразования методом скользящего окна была получена последовательность временных отсчетов:

X(-n),...,X(-2),X-(-1),X0,X(+1)

где X+1 прогнозируемое значение, полученное с помощью предыдущего этапа обработки (например нейронной сети) на основе *n* предыдущих значений. Тогда, чтобы построить прогноз для значения X(+2) нужно сдвинуть всю последовательность на один отсчет влево, чтобы ранее сделанный прогноз X(+1) тоже вошел в число исходных значений. Затем снова будет запущен алгоритм расчета прогнозируемого значения - X(+2) будет рассчитан с учетом X(+1) и так далее в соответствии с заданным горизонтом прогноза.

Диаграмма прогноза становится доступной в списке способов представления только для тех ветвей сценария, которые содержат прогноз временного ряда. Основное отличие диаграммы прогноза от обычной диаграммы в том, что на ней, кроме исходных данных отображаются результаты прогноза, при этом исходные данные и прогноз отличаются по цвету (рис. 4).

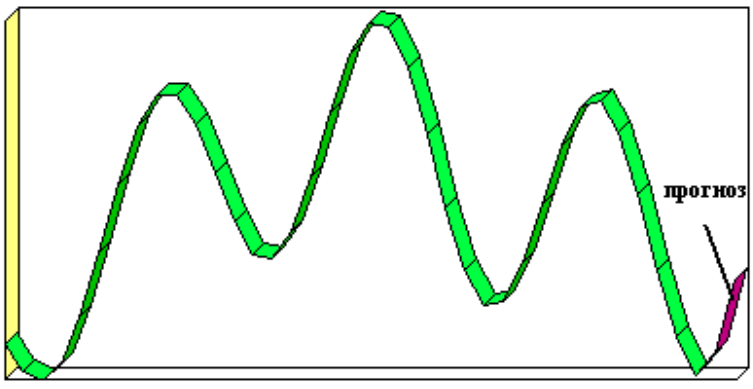


Рис. 4 Диаграмма прогноза.

Обработчик «Разгруппировка». Применительно к задаче прогнозирования объема продаж разгруппировка позволяет распределить прогнозные значения, рассчитанные моделью (например, нейросетью) для определенной группы товара, по каждой товарной позиции в данной группе. Основой для такого попозиционного распределения служит гипотеза о том, что каждый товар в группе

Просмотр результатов попозиционного прогноза удобно реализовать в многомерном виде, особенно если прогноз осуществляется более чем на один период (рис.5)

	Шаг прогноза		
TOVAR	1	2	Итого
1818	20 026	32 515	52 540
2473	20 590	33 431	54 021
2844	8 141	13 218	21 358
3061	8 776	14 249	23 025
3068	1 021	1 657	2 678
4397	6 176	10 027	16 203
4530	2 033	3 302	5 335
4531	1 844	2 994	4 838
4718	5 308	8 618	13 926
Итого	73 915	120 011	193 926

Рис. 5. Прогноз по каждой товарной позиции после разгруппировки.

Таким образом, типичный сценарий построения модели, описывающей объема продаж, в Deductor выглядит следующим образом (рис.6):

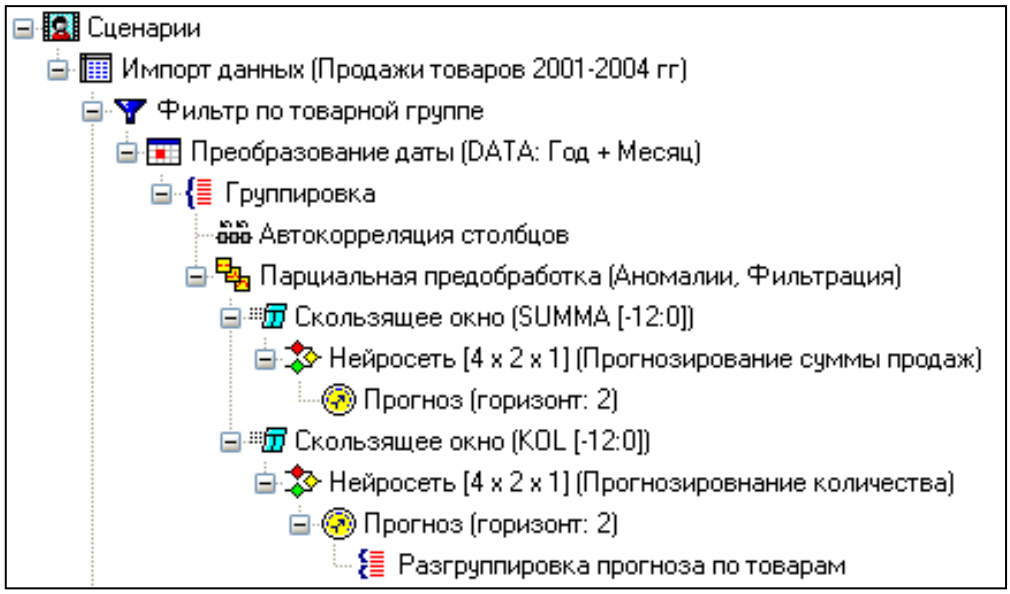


Рис.6. Сценарий для прогнозирования объема продаж.

2. Создание сценария прогнозирования объема продаж.

В папке «Мои документы\Лаб6» расположен файл sales_kraska.txt(данные, содержащие историю продаж за некоторый период), имеющий следующую структуру:

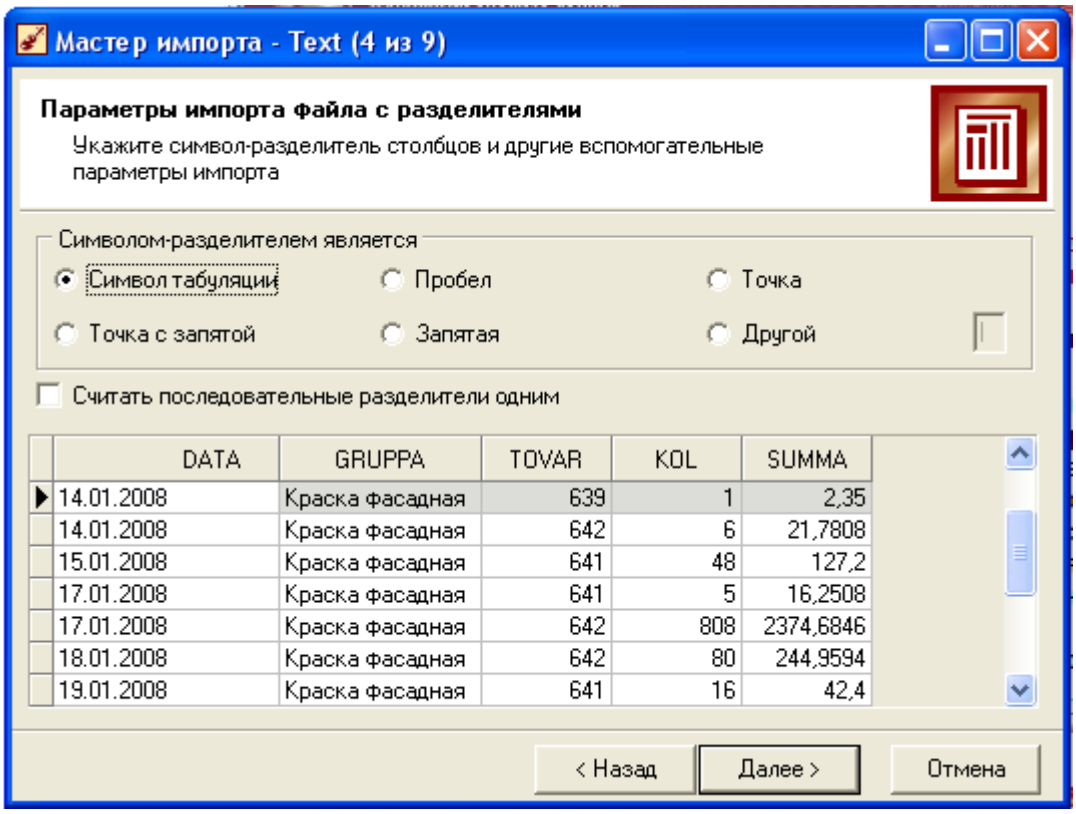
DATA	GRUPPA	TOVAR	KOL	SUMMA	
10.01.2008		Краска фасадная	642	768	2419,2000
12.01.2008		Краска фасадная	642	64	211,1773
13.01.2008		Краска фасадная	641	346	1042,3400
14.01.2008		Краска фасадная	639	1	2,3500
14.01.2008		Краска фасадная	642	6	21,7808
15.01.2008		Краска фасадная	641	48	127,2000
17.01.2008		Краска фасадная	641	5	16,2508
17.01.2008		Краска фасадная	642	808	2374,6846
18.01.2008		Краска фасадная	642	80	244,9594
19.01.2008		Краска фасадная	641	16	42,4000
20.01.2008		Краска фасадная	642	48	144,1154
...		...	...	...	...

Требуется на основе исторических данных построить прогноз количества и сумм продаж на будущие два периода (период - месяц) по каждой товарной позиции. Сравнить полученный прогноз с фактическим результатом.

Сценарий представлен на рис.6. Именно его необходимо создать.

Откройте программу Deductor Studio, используя ярлык на рабочем столе Deductor Studio или через кнопку Пуск.

*Импорт данных.*В панели сценарии выберите «Мастер импорта», а в нем выбрать text (Текстовый файл Direct). Укажите путь к файлу «sales.txt» и импортируйте данные из файла.



На шаге 9 в поле «Метка» введите заголовок - Импорт данных (Продажи товаров).

Фильтр по товарной группе. Выбрать Импорт данных (Продажи товаров) и в «Мастер обработки» нажать «Фильтрация».Настроить условие фильтрации:

Операция	Поле	Условие	Значение
	ab GRUPPA	=	Краска фасадная
И	DATA	<=	31.12.2012

Введите заголовок для окна - Фильтр по товарной группе.

Преобразование даты. Выбрать Фильтр по товарной группе и в «Мастер обработки» нажать «Дата и время».Data – используемое, остальные – непригодные. В столбце «Дата» поставить галочку напротив «Год + Месяц».Заголовок для окна - Преобразование даты (DATA: Год + Месяц).

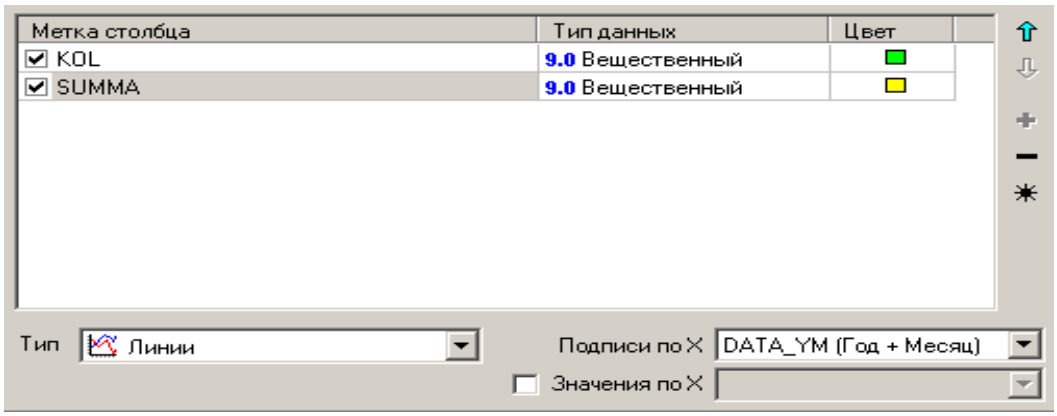
Группировка. Выбрать - Преобразование даты (DATA: Год + Месяц) и в «Мастер обработки» нажать «Группировка».

Неиспользуемые поля : Data, Gruppa, Tovar.

Факты: Kol, Summa.

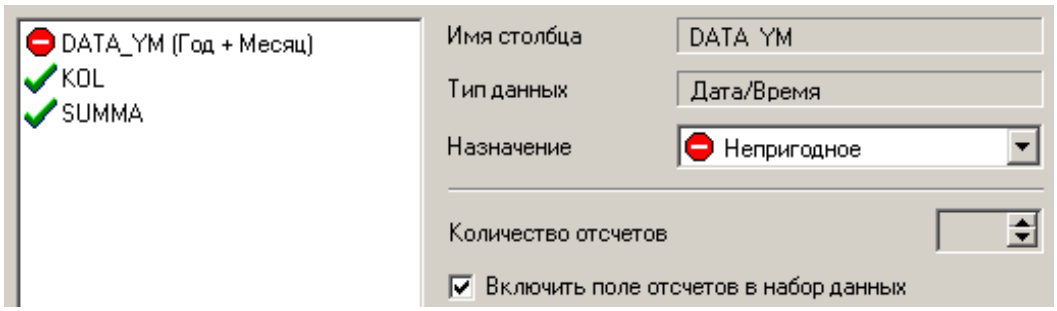
Измерение: Data (Год+Месяц).

Способ отображения данных выбрать «Таблица и Диаграмма».Настройте столбцы диаграммы:



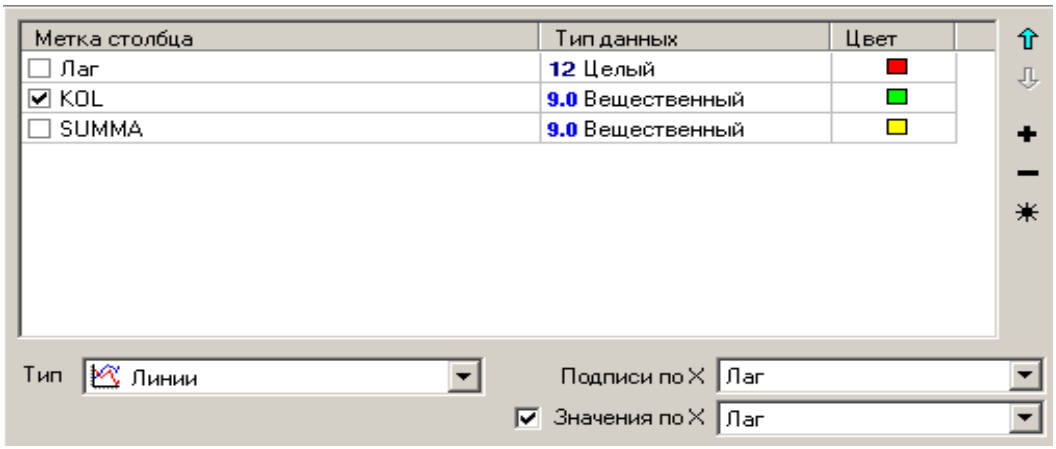
Автокорреляция столбцов. Выбрать - Группировка и в «Мастер обработки» нажать «Автокорреляция».

Параметры «Автокорреляции»:



Способ отображения «Таблица и Диаграмма».

Настройка столбцов диаграммы:

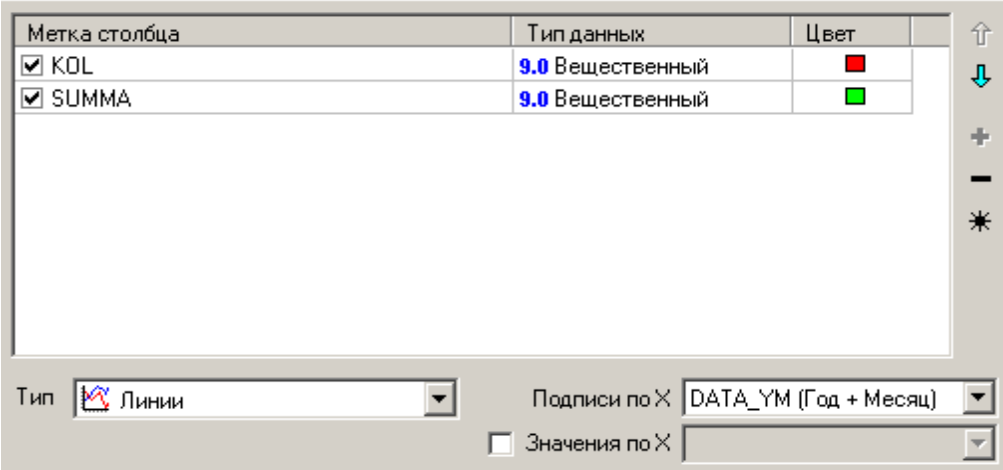


Заголовок для окна - Автокорреляция столбцов KOL, SUMMA.

Парциальная предобработка. Выбрать - Группировка и в «Мастер обработки» нажать «Парциальная обработка».Восстановление пропущенных данных оставить без изменения.Редактирование аномальных значений: поставить галочку в KOL и в SUMMA. Степень подавления – Малая.

Спектральная обработка: поставить в KOL и в SUMMA вычитание шума, степень подавлениямалая. Затем «Далее», «Пуск». Способ отображения Таблица и Диаграмма.

Настройка столбцов диаграммы:



Заголовок для окна - Парциальная предобработка (Аномалии, Фильтрация).

На этом этапы подготовки набора данных для прогнозирования выполнены.

3. Прогнозирование суммы продаж.

Скользящее окно. Выбрать - Парциальная предобработка (Аномалии, Фильтрация) и в «Мастер обработки» нажать «Скользящее окно».

Информационные: Data (Год+Месяц), KOL.

Используемое: SUMMA, глубина погружения – 12, горизонт прогнозирования – 0.

Заголовок для окна - Скользящее окно (SUMMA [-12:0]).

Нейросеть. Выбрать - Скользящее окно (SUMMA [-12:0]) и в «Мастер обработки» нажать «Нейросеть».

Входные: SUMMA -1, SUMMA -2, SUMMA -3, SUMMA -12.

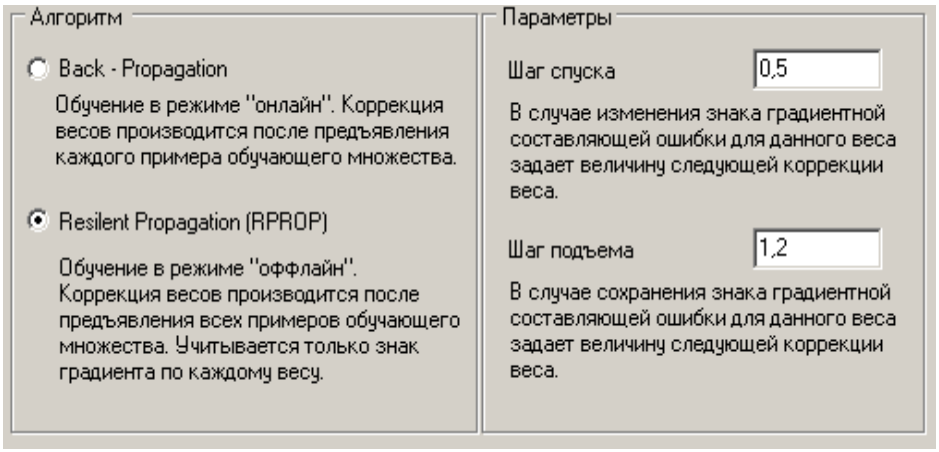
Выходное: SUMMA

Остальные: неиспользуемые.

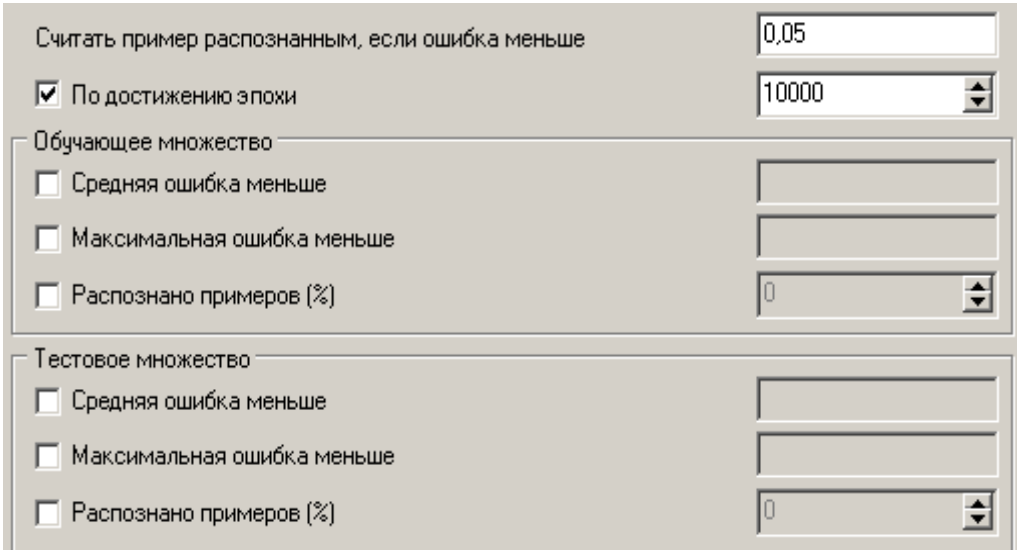
Далее установите (проверьте) следующие параметры:

Шаг 4:

Шаг 5:



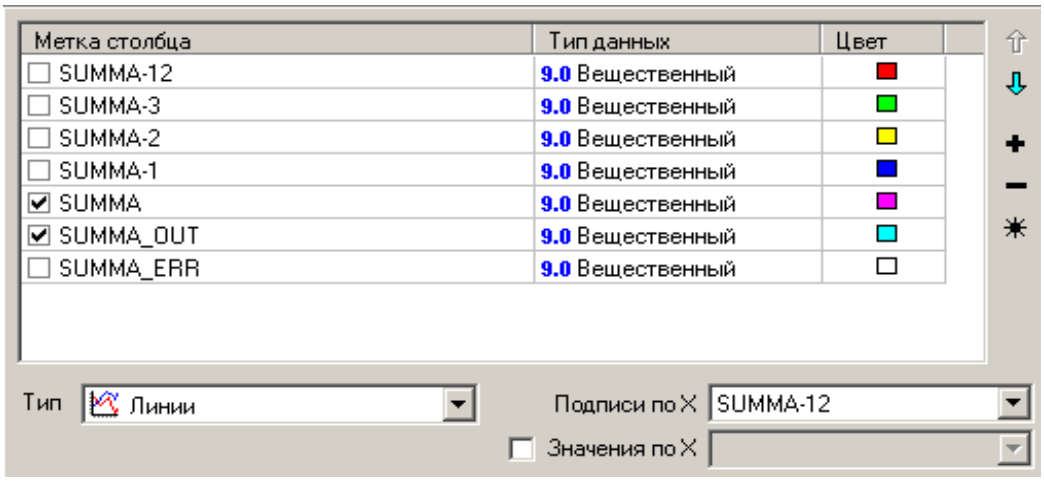
Шаг 6:



Шаг 8:

Поставить галочки: Граф нейросети, Обучающий набор, Таблица, Диаграмма.

Шаг 9:



Заголовок для окна - Нейросеть [4x2x1] (Прогнозирование суммы продаж).

Прогноз. Выбрать - Нейросеть [4x2x1] (Прогнозирование суммы продаж) и в «Мастер обработки» нажать «Прогнозирование». В шаге 2 данные остаются без изменения.Горизонт прогноза: 2

Установить галочки: Добавлять горизонт прогноза и Исходные данные.Способ отображения данных: Диаграмма прогноза и Таблица.

Поставить галочку SUMMA, с остальных полей убрать. Установить Тип – Линии.

Заголовок для окна - Прогноз (горизонт: 2).

4. Прогнозирование количества проданного товара.

Скользящее окно. Выбрать –«Парциальная предобработка (Аномалии, Фильтрация)» и в «Мастер обработки» нажать «Скользящее окно».

Информационные: Data_YM (Год+Месяц), SUMMA.

Используемое: KOL, глубина погружения – 12, горизонт прогнозирования – 0.

Заголовок для окна - Скользящее окно (KOL [-12:0]).

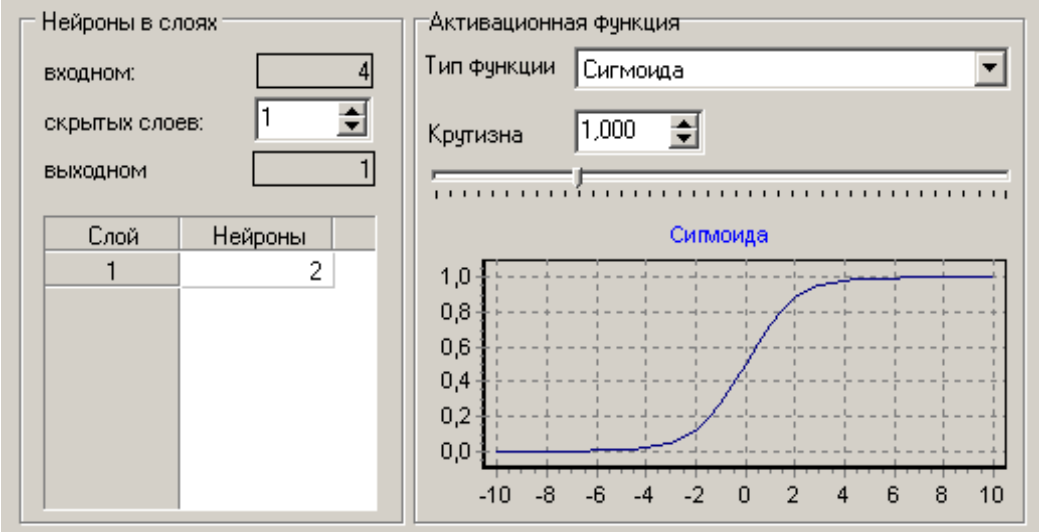
Нейросеть. Выбрать - Скользящее окно (KOL [-12:0]) и в «Мастер обработки» нажать «Нейросеть».

Входные: KOL - 1, KOL - 2, KOL - 3, KOL - 12.

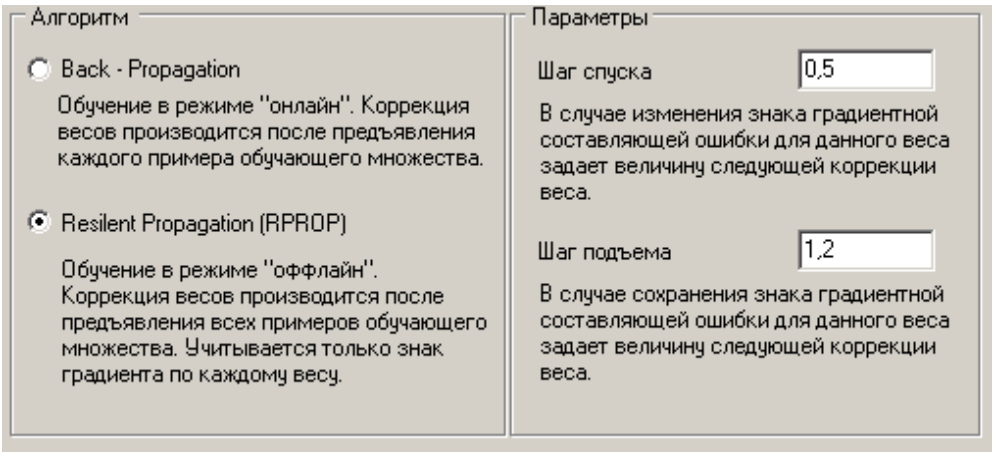
Выходное: KOL

Остальные – неиспользуемые.

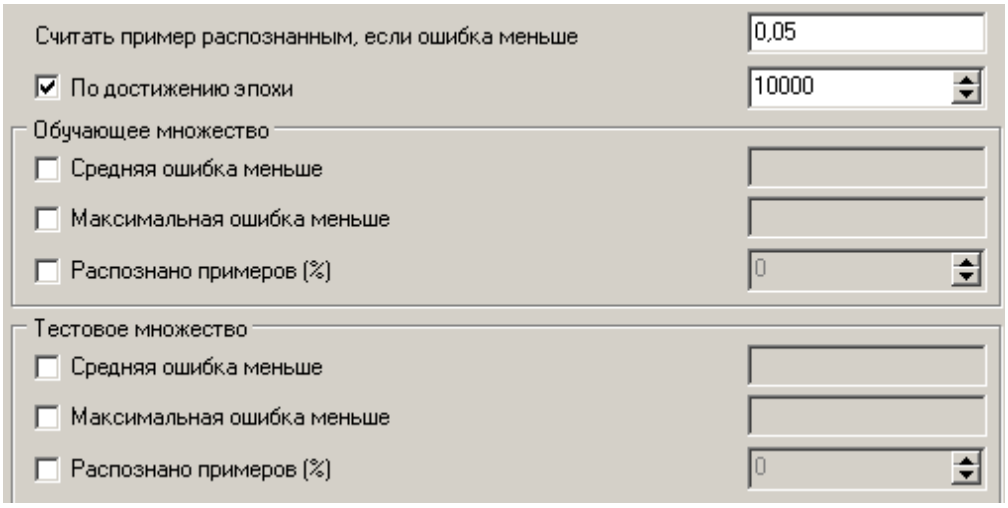
Шаг 4:



Шаг 5:

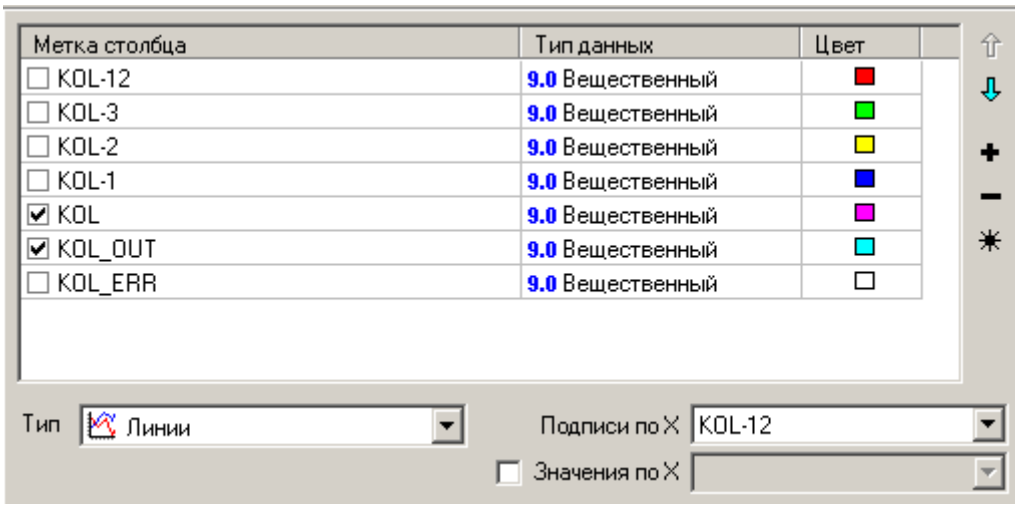


Шаг 6:



Шаг 8:Поставить галочки: Граф нейросети, Диаграмма рассеяния, Таблица, Диаграмма.

Шаг 9:



Заголовок для окна - Нейросеть [4x2x1] (Прогнозирование количества).

Прогноз. Выбрать - Нейросеть [4x2x1] (Прогнозирование суммы продаж) и в «Мастер обработки» нажать «Прогнозирование».

В шаге 2 данные остаются без изменения.Горизонт прогноза: 2

Установить галочки: Добавлять горизонт прогноза и Исходные данные.

Способ отображения данных: Диаграмма прогноза и Таблица.

Поставить галочку KOL, с остальных полей убрать. Установить Тип – Линии.

Заголовок для окна - Прогноз (горизонт: 2).

*Задание.*Сравните полученный прогноз с фактическим результатом, который находится в файле

fact_kraska.txt.Проанализируйте полученные прогнозы и сформулируйте рекомендации по ним.

ЗАКЛЮЧЕНИЕ

Изучив материал данного учебного пособия, обучаемый получит комплекс ориентирующих знаний в области разработки и применения интеллектуальных информационных систем и программных продуктов. Эти знания могут служить отправной точкой в дальнейшем изучении обширной и актуальной области искусственного интеллекта.

Практикум позволяет получить навыки формализации знаний. Понять основы этапов разработки простых экспертных систем, причем особенностью является то, что при этом обучаемый выполняет функционал всех членов коллектива разработчиков ЭС – эксперта, инженера по знаниям, программиста и пользователя. Кроме того формируются навыки работы бизнес-аналитика с использованием средств интеллектуального анализа данных.

СПИСОК ИСПОЛЬЗОВАНИЙ ИСТОЧНИКОВ

1. Романов, В.П. Интеллектуальные информационные системы в экономике: Учебное пособие.-.: «Экзамен», 2003. – 496 с.
2. Гаврилова, Т.А., Хорошевский, С.В. Базы знаний интеллектуальных систем: учебное пособие. – СПб.: Питер,2006. -382 с.
3. Гаскаров, Д.В. Интеллектуальные информационные системы:Учебник для вузов. -М.: ВШ, 2005. – 432
4. Паклин, Н.Б., Орешков, В.И. Бизнес аналитика: от данных к знаниям: Учеб. пособие .2-е изд. – СПб.: Питер, 2010. – 704 с. , прил. CD
5. Тельнов, Ю.Ф. Интеллектуальные информационные системы в экономике. Учебное пособие. -М.: СИНТЕГ. 2001. – 316 с.
6. **Роберт Левин, Диана Дранг. Практическое введение в технологию искусственного интеллекта и экспертных систем с примерами.** /Пер. с англ. М.Л.Сальникова, Ю.В.Сальниковой .-М.: Мир, 2000. - 160 с.
7. Программирование на языке Пролог для искусственного интеллекта: /Пер. с англ. И.Братко.-М.: Мир, 1990. - 560 с.
8. Х. Уэно, М. Исидзука. Представление и использование знаний. М: Мир, 1989. -220 с.
9. Змиртович А.И. Интеллектуальные информационные системы. – Мн.: НТООО «ТетраСистемс». 1997. – 256 с.

10. Попов Э.В. Системы общения и экспертные системы. Искусственный интеллект. В 3-х кн. Кн. 1. Справочник / Под ред. Э.В. Попова – М.: Радио и связь. 1990.
11. Turban, McLean, Wetherbe Information Technology for Management: Making Connection for Strategic Advantage 2nd Edition, John Willey & Songs, Inc., New York, 1999.
12. J. Giarratano Expert Systems: Principles and Programming. PWS Publishing Company, Boston, 1998.

Учебное издание

Козлов Алексей Николаевич

Интеллектуальные информационные системы

Учебник

Напечатано в авторской редакции
Подписано в печать 29.07. 2013. Формат 60×84 1/16
Усл. печ. л. 17,38. Тираж 70 экз. Заказ № 121

ИПЦ «ПрокростЪ»
Пермской государственной сельскохозяйственной академии
имени академика Д.Н.Прянишникова,
614990, Россия, г. Пермь, ул. Петропавловская, 23
тел. 210-35-34