Министерство сельского хозяйства Российской Федерации

**Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Смоленская государственная сельскохозяйственная академия»**

И.Н. МИШИН

Основы информационных систем и баз данных

Учебное пособие для вузов

Электронное издание

Смоленск, 2023

ДАЛЕЕ

УДК 004.9:004.65(075.8)
ББК 16.3я75

Рецензент: Белокопытов А.В., доктор экономических наук, профессор ФГБОУ ВО Смоленская ГСХА

Мишин, И.Н. Основы информационных систем и баз данных: Учебное пособие для вузов. [Электронный ресурс] /И.Н. Мишин. – Электрон. текстовые дан. (1 файл: 1,16 Мб). – Смоленск: ФГОУ ВО Смоленская ГСХА, 2023. – 1 электрон, опт. диск. – Систем. требования: Intel PIII и выше от 0,5 ГГц; Windows (XP и выше), Android (4 и выше), Acrobat Reader, Google Chrome. – Загл. с титул. экрана.

Учебное пособие направленно на получение студентами базовых знаний и компетенций в области современных информационных систем и баз данных, В пособии рассматриваются основные понятия и определения информационных систем и баз данных, теоретические основы баз данных, основы их проектирования, принципы построения и использования распределенных баз данных и интеллектуальных информационных систем, определяющих современный уровень подготовки специалистов и позволяющих применять полученные теоретические знания и практические навыки в образовательной и профессиональной деятельности.

Учебное пособие предназначено для студентов высших учебных заведений, обучающихся по направлениям подготовки 19.03.04 Технология продукции и организация общественного питания, 38.03.01 Менеджмент.

Текстовое электронное издание.

Рекомендовано к изданию научно-методическим советом ФГБОУ ВО Смоленская ГСХА от 26.04.2023 протокол № 4.

Минимальные системные требования:
Компьютер: Intel Pentium III и выше с тактовой частотой от 0,5 ГГц; ОЗУ 512 Мб и выше; 32 Мб и выше на жестком диске; видеосистема с разрешением 1280×720 пикселей и выше; привод CD/DVD-ROM;
Операционная система: Windows (XP и выше), Android (4 и выше);
Программное обеспечение: Acrobat Reader, Google Chrome.

ТИТУЛ
ДАЛЕЕ

ТИТУЛ

ОГЛАВЛЕНИЕ

В ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сегодня отрасль агропромышленного комплекса (АПК), в том числе производство и переработка продукции животноводства и растениеводства – один из самых динамично развивающихся секторов российской экономики. Все отрасли АПК требуют автоматизации, причем на каждом уровне использования технологий, производства и переработки продукции. Информационные технологии и системы крайне необходимы в сельском хозяйстве – начиная от дистанционного управления оборудованием и заканчивая автоматизацией крупных перерабатывающих предприятий.

В современных условиях одним из основных направлений в решении задач приоритетного развития АПК страны и регионов, решении продовольственных вопросов, повышения конкурентоспособности является интенсификация агропромышленного производства на базе автоматизации, комплексной механизации и развития информационных технологий и систем. Информационные системы и базы данных применяются в различных сферах сельского хозяйства: размещении сельскохозяйственных культур в зональных системах севооборота, рационе кормления животных, расчете доз удобрений, проведении комплекса землеустроительных работ и управлении земельными ресурсами, ведении государственного земельного кадастра и разработке технологических карт возделывания сельскохозяйственных культур, регулировании режима питания растений и микроклимата в теплицах и т. д.

Спектр использования информационных систем (ИС) просто огромный. Сегодня товаропроизводители в АПК понимают, что без использования качественных информационных технологий они не смогут снизить издержки производства, кроме этого без ИС, без использования современных методов анализа информации выжить на конкурентном рынке невозможно.

Специалисты АПК уже осознали все выгоды внедрения ИС в своем секторе: стоимость информационных технологий и систем не столь велика, чтобы отпугивать даже самых осторожных и консервативных хозяйственников. Достаточно крупные информационные проекты окупаются в секторе АПК за два, максимум за три года.

Данное учебное пособие предназначено для студентов высших учебных заведений, обучающихся по направлениям подготовки 19.03.04 Технология продукции и организация общественного питания, 38.03.02 Менеджмент. Учебное пособие также может быть использовано студентами, обучающихся по направлениям подготовки 35.00.00 Сельское, лесное и рыбное хозяйство и 36.00.00 Ветеринария и зоотехния. В данном пособии рассматриваются основные понятия и определения информационных систем и баз данных, теоретические основы и проектирование баз данных, принципы построения и использование распределенных баз данных и интеллектуальных информационных систем в различных областях профессиональной деятельности специалиста АПК.

В ОГЛАВЛЕНИЕ

Глава 1. Информационные системы

1.1. Информация и данные

Восприятие реального мира можно соотнести с последовательностью разных, хотя иногда и взаимосвязанных, явлений. С давних времен люди пытались описать эти явления (даже тогда, когда не могли их понять). Такое описание называют ***данными.*** Традиционно фиксация данных осуществляется с помощью конкретного средства общения (например, с помощью естественного языка или изображений на конкретном носителе на камне, бумаге и т.д.).

Данные могут рассматриваться как признаки или записанные наблюдения, которые по каким-то причинам не используются, а только хранятся. В том случае, если появляется возможность использовать эти данные для уменьшения неопределенности о чем-либо, данные превращаются в информацию. Поэтому можно утверждать, что информацией являются используемые данные.

Наряду с понятием «данные» в литературе используется такое понятие как «***информация***», причём чаще всего эти два понятия отождествляются. Если подойти к этому вопросу с позиции теории информации, то следуя определению основоположника этой теории *Клода Шеннона* под «информацией» понимаются не любые сообщения (данные), а лишь те, которые уменьшают неопределённость у получателя информации. Таким образом, эти два понятия, строго говоря не тождественны.

Термин ***информация*** происходит от латинского informatio, что означает разъяснение, осведомление, изложение. С позиции материалистической философии информация есть отражение реального мира с помощью сведений (сообщений). Сообщение — это форма представления информации в виде речи, текста, изображения, цифровых данных, графиков, таблиц и т.п. В широком смысле информация — это общенаучное понятие, включающее в себя обмен сведениями между людьми, обмен сигналами между живой и неживой природой, людьми и устройствами.

Информация — ***сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределенности, неполноты знаний***.

Знание определяется как проведенный практикой опыт познания окружающего мира, отражение действительности в мышлении человека. Таким образом, с точки зрения понятия «знание» - **Информация - это отчужденное знание, выраженное на определенном языке в виде знаков алфавита, записанное на материальный носитель, доступное для воспроизведения без участия автора и переданное в каналы общественной коммуникации.**

Информатика рассматривает информацию как концептуально связанные между собой сведения, данные, понятия, изменяющие наши представления о явлении или объекте окружающего мира.

С середины XX века информация является общенаучным понятием, включающим в себя: обмен сведениями между людьми, человеком и автоматом, автоматом и автоматом; обмен сигналами в животном и растительном мире; передачу признаков от клетки к клетке, от организма к организму и т.д.

Поэтому, в зависимости от сферы использования, информация может быть экономической, технической, генетической и т.п.

Информация в управлении и экономике.

В частности, под экономической информацией понимается информация, характеризующая производственные отношения в обществе.

К ней относятся сведения, которые циркулируют в экономической системе, о процессах производства, материальных ресурсах, процессах управления производством, финансовых процессах, а также сведения экономического характера, которыми обмениваются между собой различные системы управления.

Ее отличительная черта — связь с процессами управления коллективами людей, организацией. Экономическая информация сопровождает процессы производства, распределения, обмена и потребления материальных благ и услуг. Значительная часть ее связана с общественным производством и может быть названа производственной информацией.

Экономическая информация – совокупность сведений, отражающих социально-экономические процессы и служащих для управления этими процессами и коллективами людей в производственной и непроизводственной сфере.

В соответствии с общей теорией управления, процесс управления можно представить как взаимодействие двух систем: управляющей и управляемой. Система управления предприятием функционирует на базе информации о состоянии объекта в соответствии с поставленной целью. Управление осуществляется путем подачи управленческого воздействия с учетом обратной связи - информации о текущем состоянии управляемой системы и внешней среды. Назначение управляющей системы - формировать такие воздействия на управляемую систему, которые побуждали бы последнюю принять состояние, определяемое целью управления.

Экономическую информацию принято разделять по следующим основным признакам:

– по функциям управления;

– по месту возникновения (уровню управления).

По функциям управления экономическая информация разделяется на:

– планово-учетную информацию (директивные и фактические показатели деятельности предприятия за некоторый период);

– нормативно-справочную информацию (справочные и нормативные данные, связанные с производственными процессами и отношениями);

– отчетно-статистическую информацию (результаты фактической деятельности для вышестоящих и государственных органов).

Классификация экономической информации по уровням управления (месту возникновения) включает в себя входную и выходную информацию.

Входная информация - это информация, поступающая извне и используемая как первичная информация для реализации экономических и управленческих функций и задач управления.

Выходная информация - это информация, поступающая из одной системы управления в другую. Одна и та же информация может являться входной для одного структурного подразделения и выходной - для другого.

Форма представления экономической информации может быть текстовой и графической, а физическим носителем ин формации может быть бумага, магнитный диск, изображение на экране дисплея и т.п.

В развитии экономики определяющую роль играет ее информационная инфраструктура. Информационные процессы накопления знаний и превращение их в информационный ресурс общества становится важнейшим фактором социально-экономического развития страны и национальной экономики. Многочисленные международные, региональные и национальные информационные системы обладают громадными потенциальными возможностями для поиска идей по развитию экономики.

В производстве, распределении, обмене и потреблении возникают, распространяются и развиваются три основных информационных потока:

1. Информация, которая существует в виде овеществленных знаний в наукоемкой продукции. Современная наукоемкая продукция создается на основе использования результатов сложных фундаментальных и прикладных исследований. В информационных продуктах сконцентрировано новейшее знание по проблемам научных разработок, технологии производства и управления.

2. Информация, отражающая профессиональные знания, частично фиксируемые в виде изобретений, патентов, лицензий, но главным образом передаваемые в виде производственных навыков и приемов.

3. Информация по методам и технологии практического решения задач управления производством. Эта информация используется для менеджмента (управления предприятием, персоналом и производством), маркетинга (управления разработкой продукции и рынком сбыта) и таргетинга (долгосрочных программ завоевания рынков сбыта).

Эти информационные потоки создаются в результате интеллектуальной составляющей труда наиболее квалифицированной части трудоспособного населения. Современное производство - высокоорганизованный процесс, в котором человек выполняет функции регулирования и контроля. Выполнение

этих функций возможно только на основе знания, которое рождается и потребляется в процессе сбора, накопления и анализа производственной информации.

Иерархия информации в управлении.

В информационных системах основной элемент информации представлен в виде данных. Данные играют существенную роль в деятельности любого предприятия. Данные фиксируются в определенной форме, пригодной для последующей обработки, хранения и передачи (на бумаге, магнитных носителях, перфоленте и пр.).

Из данных извлекается необходимая информация. Данные можно рассматривать как сырье (ресурс) для производства информации. В результате обработки данные приобретают смысл, т.е. становятся информацией.

На предприятии необходима информация для управления трудовыми, финансовыми (денежными), материальными (сырьевыми) ресурсами.

Для управления денежными ресурсами предприятия (эффективного расходования денег) обычно необходимо иметь следующую информацию: какие средства доступны, сколько и на что израсходовано, откуда поступают средства, сколько осталось.

Для управления материальными ресурсами (эффективного использования) необходимо знать: какие материалы имеются в наличии, откуда поступают, куда направляются различные виды сырья, сроки получения заказов на сырье, количество заказанного, использованного, оставшегося сырья, наиболее экономичные размеры заказов и требуемое время их доставки на производство.

Для управления трудовыми ресурсами (решение кадровых вопросов) необходимо иметь информацию о числе сотрудников, их профессии, зарплате, местонахождении рабочего места, должности, прошлых достижениях сотрудников (их ценности), сегодняшнем положении, возможности продвижения по службе и пр.

Информацию, предназначенную для управления предприятием, можно условно разбить на три уровня: оперативная информация, тактическая информация, стратегическая информация (рис.1.1).

Оперативная информация нужна на нижнем уровне управления предприятием (сотрудникам различных отделов) в повседневной работе. Она представляет собой часто обновляемую, первичную, рутинную информацию. Оперативная информация - основа в информационной иерархии ИС, поэтому ее обработка автоматизируется в первую очередь. Оперативная информация должна быть доставлена и обработана в минимально возможные сроки. К данной категории относятся также различные сообщения в контуре оперативного управления, а также служебные и технологические потоки данных, связанные с контролем функционирования информационной системы.

Рис. 1.1 – Уровни информации.

Тактическая информация получается путем обобщения информации оперативного уровня и предназначена для руководителей среднего звена (начальники отделов и пр.). Автоматизация существенно ускоряет подготовку тактической информации, которая, выдается в виде отчета, различных вариантов решения на основании соответствующего информационного запроса.

К тактической информации также относится:

– внутренняя регламентная информация – различные виды отчетности о текущей бухгалтерской, производственной, организационной деятельности. Отличительной особенностью данной категории является периодический характер формирования и необходимость получения необходимых данных к определенному нормативными документами сроку;

— внешняя регламентная информация, используемая для формирования официальных статистических, налоговых отчетов, бюллетеней и сборников. Информация данной категории должна быть максимально достоверной и полной, при этом на оперативность ее формирования, содержание отчетов и правильность составления строго регламентируется;

–нормативно-справочная информация, которая регламентирует деятельность организации, персонала и информационной системы. Информация имеет срок действия и периодически обновляется.

Стратегическая информация (информация для руководства) получается в результате обработки оперативной и тактической информации. Она содержит краткие, но содержательные сводки, отчеты, прогнозы. На ее основе осуществляются долгосрочное планирование и разработка политики предприятия в целом.

Обработка оперативной информации фактически представляет собой конторскую работу, связанную со сбором и первичной обработкой данных, формированием новых документов, справок, отчетов. Автоматизация этой деятельности привела к появлению концепции электронного офиса.

Примечательно, что около 70% всего парка ПК связано с автоматизацией учрежденческой деятельности.

Автоматизация обработки тактической и стратегической информации требует реализации функций принятия решений, в связи с чем требует более гибкого и мощного программного обеспечения с элементами "искусственного интеллекта.

Адекватность информации.

При работе с информацией всегда имеется ее источник и потребитель (получатель). Пути и процессы, обеспечивающие передачу сообщений от источника информации к ее потребителю, называются ***информационными коммуникациями***.

Для потребителя информации очень важной характеристикой является ее адекватность.

Адекватность информации — это определенный уровень соответствия создаваемого с помощью полученной информации образа реальному объекту, процессу, явлению и т.п.

В реальной жизни вряд ли возможна ситуация, когда вы сможете рассчитывать на полную адекватность информации. Всегда присутствует некоторая степень неопределенности. От степени адекватности информации реальному состоянию объекта или процесса зависит правильность принятия решений человеком.

Адекватность информации может выражаться в трех формах: *семантической, синтаксической, прагматической.*

Синтаксическая адекватность. Она отображает формально-структурные характеристики информации и не затрагивает ее смыслового содержания. На синтаксическом уровне учитываются тип носителя и способ представления информации, скорость передачи и обработки, размеры кодов представления информации, надежность и точность преобразования этих кодов и т.п. Информацию, рассматриваемую только с синтаксических позиций, обычно называют данными, так как при этом не имеет значения смысловая сторона. Эта форма способствует восприятию внешних структурных характеристик, т.е. синтаксической стороны информации.

Семантическая (смысловая) адекватность. Эта форма определяет степень соответствия образа объекта и самого объекта. Семантический аспект предполагает учет смыслового содержания информации. На этом уровне анализируются те сведения, которые отражает информация, рассматриваются смысловые связи. В информатике устанавливаются смысловые связи между кодами представления информации. Эта форма служит для формирования понятий и представлений, выявления смысла, содержания информации и ее обобщения.

Прагматическая (потребительская) адекватность. Она отражает отношение информации и ее потребителя, соответствие информации цели управления, которая на ее основе реализуется. Проявляются прагматические

свойства информации только при наличии единства информации (объекта), пользователя и цели управления. Прагматический аспект рассмотрения связан с ценностью, полезностью использования информации при выработке потребителем решения для достижения своей цели. С этой точки зрения анализируются потребительские свойства информации. Эта форма адекватности непосредственно связана с практическим использованием информации, с соответствием ее целевой функции деятельности системы.

1.2. Понятие и принципы построения информационной системы

Производственные, торговые предприятия, предприятия общественного питания, фирмы, корпорации, банки, органы территориального управления представляют собой сложные организационно-экономические системы. Они состоят из большого числа элементов, реализующих производственные и управленческие функции, функции управления поставками сырья, обработки продукции, управления сбытом и взаимодействия с клиентами и многие другие. Такие экономические системы имеют многоуровневую структуру, а также обширные внешние и внутренние связи. Для обеспечения нормального функционирования сложных систем, где взаимодействуют разнообразные материальные, производственные, финансовые, информационные ресурсы и большие коллективы людей, современное развитие технологий и экономики требует внедрения и использования автоматизированных информационных систем и баз данных.

Система (в широком значении) – это образующая единое целое совокупность материальных и/или нематериальных объектов, объединенная некоторыми общими признаками, свойствами, назначением или условиями существования, жизнедеятельности, функционирования.

Любая система одновременно рассматривается и как единое целое, и как совокупность подсистем, отдельных объектов и элементов, объединенных в интересах достижения поставленных целей. Системы значительно отличаются между собой как по составу, так и по главным целям.

Приведем несколько систем, состоящих из разных элементов и направленных на реализацию разных целей (табл. 1.1).

В информатике понятие "система" широко распространено и имеет множество смысловых значений. Чаще всего оно используется применительно к набору технических средств и программ. Системой может называться аппаратная часть компьютера. Системой может также считаться множество программ для решения конкретных прикладных задач, дополненных процедурами ведения документации и управления расчетами.

Добавление к понятию "система" слова "информационная" отражает цель ее создания и функционирования. Информационные системы обеспечивают сбор, хранение, обработку, поиск, выдачу информации, необходимой в

процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые продукты.

Таблица 1.1 – Системы и их назначение

Система	Элементы системы	Главная цель системы
Организация	Люди, оборудование, материалы, здания и др.	Производство товаров
Компьютер	Электронные и программные элементы, линии связи и др.	Обработка данных
Телекоммуникационная система	Компьютеры, модемы, кабели, сетевое программное обеспечение и др.	Передача информации
Информационная система	Компьютеры, компьютерные сети, люди, информационное и программное обеспечение	Обработка, хранение и т.д. профессиональной информации

Общая теория систем была предложена Л. Берталанфи в 30-е годы XX в. Теория систем – специально-научная и логико-методологическая концепция исследования объектов, представляющих собой системы.

Термин «система» происходит от греческого слова systema, что означает «целое, составленное из частей или множества элементов, связанных друг с другом и образующих определенную целостность, единство». Под системой понимается совокупность связанных между собой и с внешней средой элементов или частей, функционирование которых направлено на получение конкретного полезного результата.

Систему определяют:

– структура – множество элементов системы и их взаимосвязей. Математической моделью структуры является граф;

– входы и выходы – материальные потоки или потоки сооб^-щений, поступающие в систему и выводимые ею;

– поведение системы, описываемое неким законом. Основные свойства системы:

– сложность (зависит от множества входящих в нее компонентов, их структурного взаимодействия);

– делимость (означает, что она состоит из ряда подсистем или элементов, выделенных по определенному признаку);

– целостность (означает, что функционирование множества элементов системы подчинено одной цели);

– многообразие элементов и различие их природы (связано с их функциональной специфичностью и автономностью);

– структурированность (определяет наличие установленных связей и отношений между элементами внутри системы, распределение элементов по уровням иерархии).

В основе решения многих задач лежит обработка информации. Для автоматизации информационных процессов, в том числе для обработки, хранения, передачи информации создаются и используются информационные системы.

Существует несколько взаимосвязанных определений понятия информационная система.

Информационная система (ИС) – это система сбора, хранения, накопления, поиска и передачи информации, применяемой в процессе управления или принятия решений. Она обычно включает информационно-справочный фонд (документы, базы данных, информационные хранилища), язык обработки информации и общения с системой, носители информации, а также комплекс моделей, обеспечивающих функционирование системы.

Информационная система (ИС) – это взаимосвязанная совокупность технических и программных средств, методов и обслуживающего персонала, используемых для автоматизированного сбора, обработки и манипулирования данными.

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели**.**

Экономическая информационная система (ЭИС) представляет собой совокупность внутренних и внешних потоков информации экономического объекта, методов, средств, специалистов, участвующих в процессах сбора, хранения, обработки, поиска и выдачи необходимой информации, предназначенной для выполнения функций управления.

Автоматизированная информационная система (АИС) представляет собой совокупность информации, экономико-математических методов и моделей, аппаратно-программных, организационных, технологических средств и специалистов. АИС предназначена для эффективной эксплуатации ЭИС, включая принятие оптимальных управленческих решений.

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации персонального компьютера. В крупных организациях наряду с персональным компьютером в состав технической базы информационной системы используются специализированные высокопроизводительные серверные системы. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление.

Необходимо понимать разницу между компьютерами и информационными системами. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система

немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

Отметим, что любая информационная система объединяет следующие базовые составляющие:

–языковые средства и правила, используемые для отбора и подготовки информации ко вводу в компьютерную систему, для отображения картины реального мира в модель данных, для работы пользователя с системой, для представления пользователю выдаваемой системой информации;

–информационный фонд системы;

–способы и методы организации информационных массивов и всех процессов обработки информации в системе;

–алгоритмы функционирования системы, т. е. алгоритмы всех процедур по созданию, ведению и обработке информационных массивов и т. д.

–программное обеспечение системы, в состав которого входят программы, реализующие все алгоритмы функционирования системы;

–комплекс технических средств, функционирующих в системе;

–персонал, обслуживающий ИС.

Можно выделить ряд базовых свойств, которые являются общими для информационных систем:

–информационные системы предназначены для сбора, хранения и обработки информации, поэтому в основе любой из них лежит среда хранения и доступа к данным;

–информационные системы ориентированы на конечного пользователя, не обладающего высокой квалификацией в области вычислительной техники. Поэтому клиентские приложения информационной системы должны обладать простым, удобным, легко осваиваемым интерфейсом, который предоставляет конечному пользователю все необходимые для работы функции и в то же время не дает ему возможности выполнять какие-либо лишние действия;

–любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;

– информационная система является динамичной и развивающейся;

–при построении информационной системы необходимо использовать системный подход;

–выходной продукцией информационной системы является информация, на основе которой принимаются решения;

–информационную систему следует воспринимать как человеко-компьютерную систему обработки информации.

Технология работы в информационной системе доступна для понимания специалистом некомпьютерной области и может быть успешно использована для контроля процессов профессиональной деятельности и управления ими.

Создание, развитие и эксплуатация информационных систем способствует значительному повышению качества управления за счет эффективного использования информационных ресурсов предприятия.

Информационные системы могут обслуживать как предприятие в целом, так и отдельные функции или задачи управления. Современная индустрия информатизации предлагает множество различных информационно-технологических решений, которые могут использоваться для построения информационных систем на самых разнообразных организационно-экономических объектах и уровнях управления экономикой.

Внедрение информационных систем может способствовать:

–обеспечению планирования и контроля затрат, анализа хозяйственной деятельности;

–получению более рациональных вариантов решения управленческих задач за счет внедрения математических методов и интеллектуальных систем;

–последовательному финансовому планированию, облегчению анализа факторов, влияющих на бюджет, анализу различных сценариев;

–освобождению работников от рутинной работы за счет ее автоматизации;

–обеспечению достоверности информации;

–замене бумажных носителей данных на электронный документооборот, совершенствованию структуры потоков информации и системы документооборота в организации;

–улучшению контроля затрат, повышению точности и детальности данных;

–уменьшению затрат на обработку данных, на внутренний аудит, на производство продуктов и услуг, на материалы и улучшение снабжения;

–предоставлению потребителям уникальных услуг;

–отысканию новых рыночных ниш;

–повышению эффективности взаимодействия между работниками организации и с внешними контрагентами.

Основные принципы построения и функционирования информационных систем.

Принцип системности является главнейшим на всех этапах создания и функционирования информационных систем. Он позволяет подойти к исследуемому объекту как к единому целому, выявить на этой основе многочисленные и многообразные связи между структурными элементами, обеспечивающими целостность системы, установить цели производственно-хозяйственной деятельности и реализуемые для достижения этих целей функции. Системный подход предполагает проведение анализа в двух аспектах, получивших название макро- и микроподходов. При макроанализе система рассматривается как часть системы более высокого порядка, что позволяет выявить имеющиеся информационные связи и выявить наиболее предпочтительные, реализующие поставленные цели. При микроанализе изучается структура объекта, анализируются его элементы, их свойства, взаимоотношения, способы функционирования. Здесь широко используются различные методы моделирования изучаемых процессов для анализа работы

системы. Это позволяет находить такой вариант структуры системы, который обеспечивает наибольшую эффективность ее функционирования.

Системный подход при формировании ИС требует выполнения следующих этапов (см. глава 3):

– определение целей системы и критериев оценки их достижения;

–определение требований к системе, т.е. каким образом она должна функционировать;

–структуризация системы, т.е. определение функциональных подсистем, их состава и перечня задач управления;

–выявление и анализ связей между подсистемами, комплексами и задачами;

– установление связей с внешней средой.

Системный принцип создания ИС позволяет внедрять отдельные подсистемы и задачи последовательно, независимо одну от другой и при этом не нарушать функциональных связей между ними.

Принцип непрерывного развития заключается в том, что информационная система создается с учетом постоянного расширения и изменения ее возможностей. Благодаря этому принципу обновляются функции системы, применяются более совершенные ИТ и инструментальные средства, наращиваются информационно-вычислительные мощности. Происходит своевременное реагирование на изменения в организационно-экономической сфере. Это требование объясняется тем, что по мере развития как экономики в целом, так и организационного управления отдельными предприятиями возникают совершенно новые задачи управления, видоизменяются старые. Поэтому ЭИС должны быстро реагировать на все подобные изменения.

Принцип единства функционирования с другими информационными системами (принцип совместимости) заключается в реализации способности информационных систем различных видов и уровней совместно функционировать, обеспечивая эффективную работоспособность экономических объектов.

Принцип стандартизации и унификации заключается в возможности применения типовых программных и информационно-технологических решений, которые предлагает современный ИТ-рынок, что позволяет сократить затраты (временные, трудовые, стоимостные) на создание и внедрение ИС.

Принцип экономической целесообразности (эффективности) заключается в достижении рационального соотношения между затратами на создание и эксплуатацию информационной системы и экономическим эффектом, полученным объектом от ее функционирования.

Помимо основополагающих принципов, необходимо иметь в виду и целый ряд частных принципов, соблюдение которых также оказывает определенное влияние на экономический эффект, получаемый от эксплуатации системы. Это:

–*принцип первого руководителя* состоит в том, что разработку и внедрение информационной системы следует производить под

непосредственным руководством первого лица. Данный принцип закрепляет за пользователем прямую заинтересованность в конечных результатах функционирования системы;

– *принцип новых задач* означает, что в составе информационной системы должны решаться качественно новые задачи, а не автоматически переноситься существующие приемы управления. На практике этот принцип осуществляется путем решения многовариантных оптимизационных управленческих задач;

–*принцип участия управленческого персонала в разработке и внедрении системы* означает, что в рабочей группе по созданию информационной системы участвуют пользователи проектируемой системы. Они должны сотрудничать с разработчиками системы в части постановки целей предприятия и путей их достижения, в формировании специфических требований к информационной системы в определении приоритетов задач и особенностей алгоритмов их решения и т.п.

– *принцип всесторонней подготовки (обучения) пользователей* предусматривает обязательное обучение аппарата управления методам и технике работы в условиях функционирования информационной системы.

Проблемы создания информационной системы тесно связаны с общими основами развития экономики и конкретного экономического объекта (предприятия, банка, фирмы) с одной стороны, а с другой – со спецификой и тенденциями быстро развивающихся ИТ. Поэтому рассмотренные базовые принципы дополняются не менее важными информационно-технологическими, без ориентации на которые невозможна разработка современных информационных систем:

– *принцип декомпозиции*, используемый для разделения системы на части (элементы) и обеспечивающий изучение свойств элементов, подсистем и системы в целом;

– *принцип автоматизации документооборота* (безбумажная технология). Суть его заключается в том, что все или, по крайней мере, подавляющее большинство информационных потоков проходит непосредственно через компьютерную сеть предприятия. Обмен информацией между отдельными органами управления, а также между ними и объектами управления должен производиться по каналам связи. Возможности безбумажной технологии обеспечивают однократность ввода и хранения информации в ЭВМ, оперативность, достоверность и многократность ее использования, а также защиту от несанкционированного доступа;

– *принцип надежности и живучести информационной системы* требует, чтобы информационная система обладала способностью компенсировать последствия нарушений и повреждений отдельных элементов системы.

Кроме всего прочего, должны соблюдаться общие принципы современных информационной технологии: интегрированность, гибкость, интерактивность.

Этапы развития информационных систем.

Первые информационные системы появились в 50-х гг. В эти годы они были предназначены для обработки счетов и расчета зарплаты, а реализовывались на электромеханических бухгалтерских счетных машинах. Это приводило к некоторому сокращению затрат и времени на подготовку бумажных документов.

60-е гг. знаменуются изменением отношения к информационным системам. Информация, полученная из них, стала применяться для периодической отчетности по многим параметрам. Для этого организациям требовалось компьютерное оборудование широкого назначения, способное обслуживать множество функций, а не только обрабатывать счета и считать зарплату, как было ранее.

В 70-х – начале 80-х гг. информационные системы начинают широко использоваться в качестве средства управленческого контроля, поддерживающего и ускоряющего процесс принятия решений.

К концу 80-х гг. концепция использования информационных систем вновь изменяется. Они становятся стратегическим источником информации и используются на всех уровнях организации любого профиля. Информационные системы этого периода, предоставляя вовремя нужную информацию, помогают организации достичь успеха в своей деятельности, создавать новые товары и услуги, находить новые рынки сбыта, обеспечивать себе достойных партнеров, организовывать выпуск продукции по низкой цене и многое другое.

В 90-х–2000-х годах характерно развитие телекоммуникационных средств, которое привело к созданию гибких локальных и глобальных вычислительных сетей, предопределивших возможность разработки и внедрения корпоративных информационных систем, объединяют возможности систем комплексной автоматизации управления 70-х годов и локальной автоматизации 80 - годов. Наличие гибких средств связывания управленческих работников в процессе хозяйственной деятельности, возможность коллективной работы, как непосредственных исполнителей хозяйственных операций, так и менеджеров, принимающих управленческие решения, позволяют во многом пересмотреть принципы управления предприятиями или проводить кардинальный реинжиниринг бизнес-процессов.

С 2000 годов по настоящее время – характерна системная автоматизация, которая учитывает автоматизацию функций принятия решения на всех уровнях управления предприятием на основе использования систем поддержки принятия решения. Активно развиваются эксплуатационные возможности технических средств, производительность процессоров, объем оперативной памяти возросли в десятки раз и используются их все более совершенные виды для создания компьютеров: при обработке информации различного назначения; при сборе и регистрации информации – от отдельных приборов к автоматическим системам; при передаче данных – от кабельной к беспроводной связи; были созданы глобальные распределенные информационные системы и базы данных на основе компьютерной сети Интернет, инфокоммуникационные

сети различных форм, функционирующие в режиме «клиент — сервер». Программная основа, так же как логическая, зависит от возможностей технических средств и развивается в направлении совершенствования системных и прикладных программных средств. Системные программные средства развиваются как в управляющей, так и в обрабатывающей частях. В управляющей части они связаны с совершенствованием операционных систем, особенно сетевых. В обрабатывающей – средств диагностирования работы устройств компьютера, архивации, обеспечения защиты информации от несанкционированного доступа и различных вирусов. Развиваются прикладные программные средства как общего, так и функционального назначения; при этом средства функционального назначения совершенствуются в направлении разработки программных средств на основе удаленного доступа и облачных технологий, позволяющих достигнуть максимальной степени интеграции данных и пользователей информационных систем.

1.3 Классификация информационных систем

Классификация информационных систем осуществляется по целому ряду классификационных признаков. Однако, следует отметить, что на практике какая-либо современная информационная система характеризуется множеством взаимосвязанных признаков, которые тесно пересекаются межу собой.

Классификация по масштабу.

По масштабу ИС подразделяются на следующие группы: одиночные, групповые, корпоративные.

Одиночные ИС – реализуются, как правило, на автономном персональном компьютере (сеть не используется). Такая система может содержать несколько простых приложений, связанных общим информационным фордом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место. Такие приложения создаются с помощью локальных систем управления базами данных (Clarion, Clipper, FoxPro, Paradox, dBase, Access), (см. главу 2).

Групповые ИС – ориентированы на коллективное использование информации членами рабочей группы строятся на базе локальной сети или на базе сети Интернет. При разработке таких приложений используются серверы баз данных (SQL-серверы) для рабочих групп (Oracle, DB2, InterBase, Sybase, Informix, Microsoft SQL Server).

Корпоративные ИС – являются развитием систем для рабочих групп, они ориентированы на крупные компании и могут поддерживать территориально разнесенные узлы или сети (см. раздел 1.5). В основном они имеют иерархическую структуру из нескольких уровней. Для таких ИС характерна архитектура клиент-сервер со специализацией серверов или же многоуровневая архитектура. При разработке таких ИС могут использоваться те же серверы баз данных, что и при разработке групповых ИС. Однако в крупных ИС наибольшее распространение получили серверы Oracle, DB2, Microsoft SQL

Server.

Для групповых и корпоративных ИС существенно повышаются требования к надежности функционирования и сохранности данных.

Классификация информационных систем по признаку структурированности задач.

Понятие структурированности задач. При создании или при классификации информационных систем неизбежно возникают проблемы, связанные с формальным – математическим и алгоритмическим описанием решаемых задач. От степени формализации во многом зависят эффективность работы всей системы, а также уровень автоматизации, определяемый степенью участия человека при принятии решения на основе получаемой информации.

Чем точнее математическое описание задачи, тем выше возможности компьютерной обработки данных и тем меньше степень участия человека в процессе ее решения. Это и определяет степень автоматизации задачи.

Различают три типа задач, для которых создаются информационные системы: структурированные (формализуемые), неструктурированные (неформализуемые) и частично структурированные.

Структурированная (формализуемая) задача – задача, где известны все ее элементы и взаимосвязи между ними.

Неструктурированная (неформализуемая) задача – задача, в которой невозможно выделить элементы и установить между ними связи.

В структурированной задаче удается выразить ее содержание в форме математической модели, имеющей точный алгоритм решения. Подобные задачи обычно приходится решать многократно, и они носят рутинный характер. Целью использования информационной системы для решения структурированных задач является полная автоматизация их решения, т. е. сведение роли человека к нулю.

Решение неструктурированных задач из-за невозможности создания математического описания и разработки алгоритма связано с большими трудностями. Возможности использования здесь информационной системы невелики. Решение в таких случаях принимается человеком из эвристических соображений на основе своего опыта и, возможно, косвенной информации из разных источников.

В практике работы любой организации существует сравнительно немного полностью структурированных или совершенно неструктурированных задач. О большинстве задач можно сказать, что известна лишь часть их элементов и связей между ними. Такие задачи называются частично структурированными. В этих условиях можно создать информационную систему. Получаемая в ней информация анализируется человеком, который будет играть определяющую роль. Такие информационные системы являются *автоматизированными*, так как в их функционировании принимает участие человек.

Типы информационных систем, используемые для решения частично структурированных задач.

Информационные системы, используемые для решения частично

структурированных задач, подразделяются на два вида (рис. 1.2):

–создающие управленческие отчеты и ориентированные главным образом на обработку данных (поиск, сортировку, агрегирование, фильтрацию). Используя сведения, содержащиеся в этих отчетах, управляющий принимает решение;

–разрабатывающие возможные альтернативы решения. Принятие решения при этом сводится к выбору одной из предложенных альтернатив.

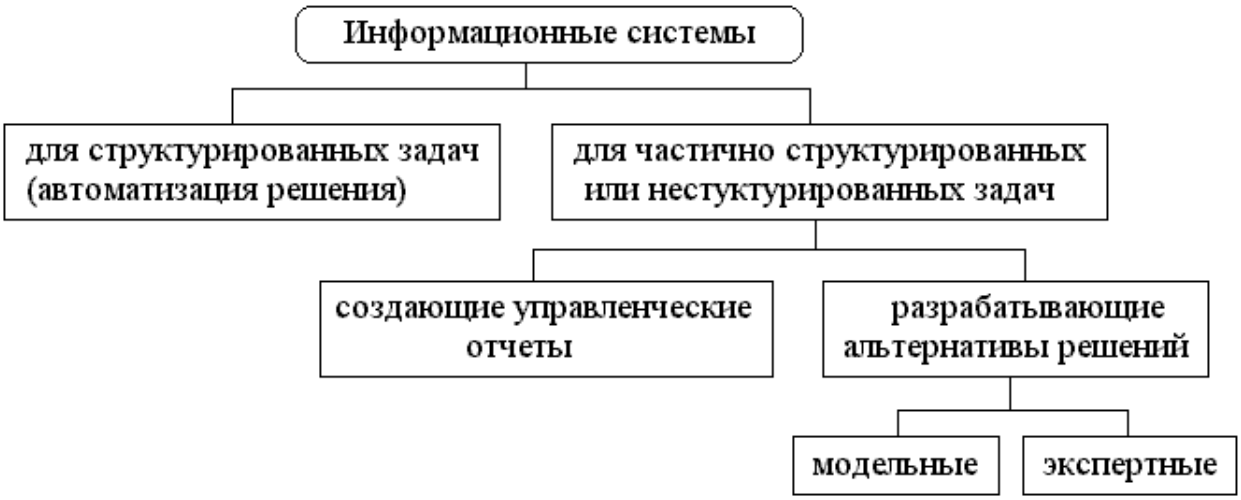


Рис. 1.2 – Классификация информационных систем по признаку структурированности решаемых задач.

Информационные системы, создающие управленческие отчеты обеспечивают информационную поддержку пользователя, т.е. предоставляют доступ к информации в базе данных и ее частичную обработку. Процедуры манипулирования данными в информационной системе должны обеспечивать следующие возможности:

–составление комбинаций данных, получаемых из различных источников;

–быстрое добавление или исключение того или иного источника данных и автоматическое переключение источников при поиске данных;

–управление данными с использованием возможностей систем управления базами данных;

–логическую независимость данных этого типа от других баз данных, входящих в подсистему информационного обеспечения;

–автоматическое отслеживание потока информации для наполнения баз данных.

Информационные системы, разрабатывающие альтернативы решений, могут быть модельными или экспертными.

Модельные информационные системы предоставляют пользователю математические, статистические, финансовые и другие модели, использование которых облегчает выработку и оценку альтернатив решения. Пользователь может получить недостающую ему для принятия решения информацию путем

установления диалога с моделью в процессе ее исследования.

Основными функциями модельной информационной системы являются:

–возможность работы в среде типовых математических моделей, включая решение основных задач моделирования типа "как сделать, чтобы?", "что будет, если?", анализ чувствительности и др.;

–достаточно быстрая и адекватная интерпретация результатов моделирования;

–оперативная подготовка и корректировка входных параметров и ограничений модели;

–возможность графического отображения динамики модели;

–возможность объяснения пользователю необходимых шагов формирования и работы модели.

Экспертные информационные системы обеспечивают выработку и оценку возможных альтернатив пользователем за счет создания экспертных систем, связанных с обработкой знаний (см. глава 5). Экспертная поддержка принимаемых пользователем решений реализуется на двух уровнях.

Работа первого уровня экспертной поддержки исходит из концепции "типовых управленческих решений", в соответствии с которой часто возникающие в процессе управления проблемные ситуации можно свести к некоторым однородным классам управленческих решений, т.е. к некоторому типовому набору альтернатив. Для реализации экспертной поддержки на этом уровне создается информационный фонд хранения и анализа типовых альтернатив.

Если возникшая проблемная ситуация не ассоциируется с имеющимися классами типовых альтернатив, в работу должен вступать второй уровень экспертной поддержки управленческих решений. Этот уровень генерирует альтернативы на базе имеющихся в информационном фонде данных, правил преобразования и процедур оценки синтезированных альтернатив.

Классификация по степени автоматизации.

В зависимости от степени автоматизации информационных процессов в системе управления фирмой информационные системы определяются как ручные, автоматические, автоматизированные.

Автоматические ИС выполняют все операции по переработке информации без участия человека.

Автоматические ИС предполагают участие в процессе обработки информации и человека, и технических средств, причем главная роль отводится компьютеру. В современном толковании в термин "информационная система" вкладывается обязательно понятие автоматизируемой системы.

Автоматизированные ИС, учитывая их широкое использование в организации процессов управления, имеют различные модификации и могут быть классифицированы, например, по характеру использования информации и по сфере применения.

Классификация по характеру использования информации.

Информационно-поисковые системы производят ввод, систематизацию,

хранение, выдачу информации по запросу пользователя без сложных преобразований данных. Например, информационно-поисковая система Google, поиск и продажа железнодорожных и авиа билетов. Наибольшее развитие такие информационные системы получили в сети Интернет.

Информационно-решающие системы осуществляют все операции переработки информации по определенному алгоритму. Среди них можно провести классификацию по степени воздействия выработанной результатной информации на процесс принятия решений и выделить два класса: управляющие и советующие.

Управляющие ИС вырабатывают информацию, на основании которой человек принимает решение. Для этих систем характерны тип задач расчетного характера и обработка больших объемов данных. Примером могут служить система оперативного планирования выпуска продукции, система бухгалтерского учета.

Советующие ИС вырабатывают информацию, которая принимается человеком к сведению и не превращается немедленно в серию конкретных действий. Эти системы обладают более высокой степенью интеллекта, так как для них характерна обработка знаний, а не данных.

Классификация информационных систем по режиму работы.

Пакетные информационные системы работают в пакетном режиме: вначале данные накапливаются, и формируется пакет данных, а затем пакет последовательно обрабатывается рядом программ. Недостаток этого режима - низкая оперативность принятия решений и обособленность пользователя от системы.

Диалоговые информационные системы работают в режиме обмена сообщениями между пользователями и системой (например, система продажи авиабилетов). Этот режим особенно удобен, когда пользователь может выбирать перспективные варианты из числа предлагаемых системой.

Классификация по сфере применения.

С учетом истории развития классификаций ИС и того, что в последнее время разрабатываются разнообразные системы специального назначения, такие как корпоративные информационные системы, бухгалтерские, банковские информационные системы, информационные системы фондового рынка, страховые информационные системы, налоговые информационные системы, статистические информационные системы и другие, используемых в различных сферах деятельности. Удобен обобщенный перечень разновидностей (типов) этих систем, отличающихся степенью сложности по совокупности признаков.

Информационные системы организационного управления предназначены для автоматизации функций управленческого персонала. Учитывая наиболее широкое применение и разнообразие этого класса систем, часто любые информационные системы понимают именно в данном толковании. К этому классу относятся информационные системы управления как промышленными фирмами, так и непромышленными объектами: гостиницами, банками,

торговыми фирмами и др.

Основными функциями подобных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом и снабжением и другие экономические и организационные задачи.

ИС управления предприятием служат для автоматизации базовых функций управления предприятием, в частности на основе отдельных специализированных стандартов и методов управления. Они широко используются при организации деятельности различных предприятий и организаций. Как правило ИС управления имеют модульную структуру. Каждый модуль предназначен для автоматизаций отдельных видов деятельности: бухгалтерский ует, кадровый учет, торговля и склад и так далее.

ИС автоматизированного проектирования (САПР) предназначены для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники или технологии. Основными функциями подобных систем являются: инженерные расчеты, создание графической документации (чертежей, схем, планов), создание проектной документации, моделирование проектируемых объектов.

Корпоративные (интегрированные) ИС (см. раздел 1.5).

Классификация по методам обработки данных.

Офисные информационные системы используются для автоматизации делопроизводства и управление документооборотом (электронный документооборот).

Системы обработки транзакций, в свою очередь, по оперативности обработки данных, разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций – OLTP (On-Line Transaction Processing), для отражения актуального состояния предметной области в любой момент времени, а пакетная обработка занимает весьма ограниченную часть. Для систем OLTP характерен регулярный (возможно, интенсивный) поток довольно простых транзакций, играющих роль заказов, платежей, запросов и т. п. (см. главы 2 и 3). Важными требованиями для них являются: высокая производительность обработки транзакций; гарантированная доставка информации при удаленном доступе к БД по компьютерным сетям.

Системы поддержки принятия решений – DSS (Decision Support System) –тип информационных систем, в которых с помощью относительно сложных запросов производится отбор и анализ данных в различных разрезах: временных, географических и по другим показателям. Такие системы используют технологии и режим OLAP (On-Line Analitical Processing (OLAP) - оперативная аналитическая обработка данных) и предназначены для нахождения зависимостей между данными (например, можно попытаться определить, как связан объем продаж товаров с характеристиками потенциальных покупателей), для проведения анализа "что если…". OLAP-

приложения оперируют с большими массивами данных, уже накопленными в OLTP-приложениях, взятыми их электронных таблиц или из других источников данных (см. главу 5).

Интеллектуальные информационные системы, или системы, основанные на знаниях (Knowledge Based System) - поддерживают задачи принятия решения в сложных системах на основе обработки больших массивов данных (Big Date) и интеллектуального анализа данных (Date Miting), где необходимо использование знаний в достаточно широком диапазоне, особенно в плохо формализуемых и плохо структурируемых системах, нечетких системах и при нечетких критериях принятия решения (см. главу 5).

1.4 Структура информационных систем

Структуру информационной системы составляет совокупность отдельных ее частей, называемых *подсистемами*. Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. В этом случае говорят *о структурном признаке* классификации, а структура любой информационной системы может быть представлена совокупностью функциональных и обеспечивающих подсистем (рис. 1.3).

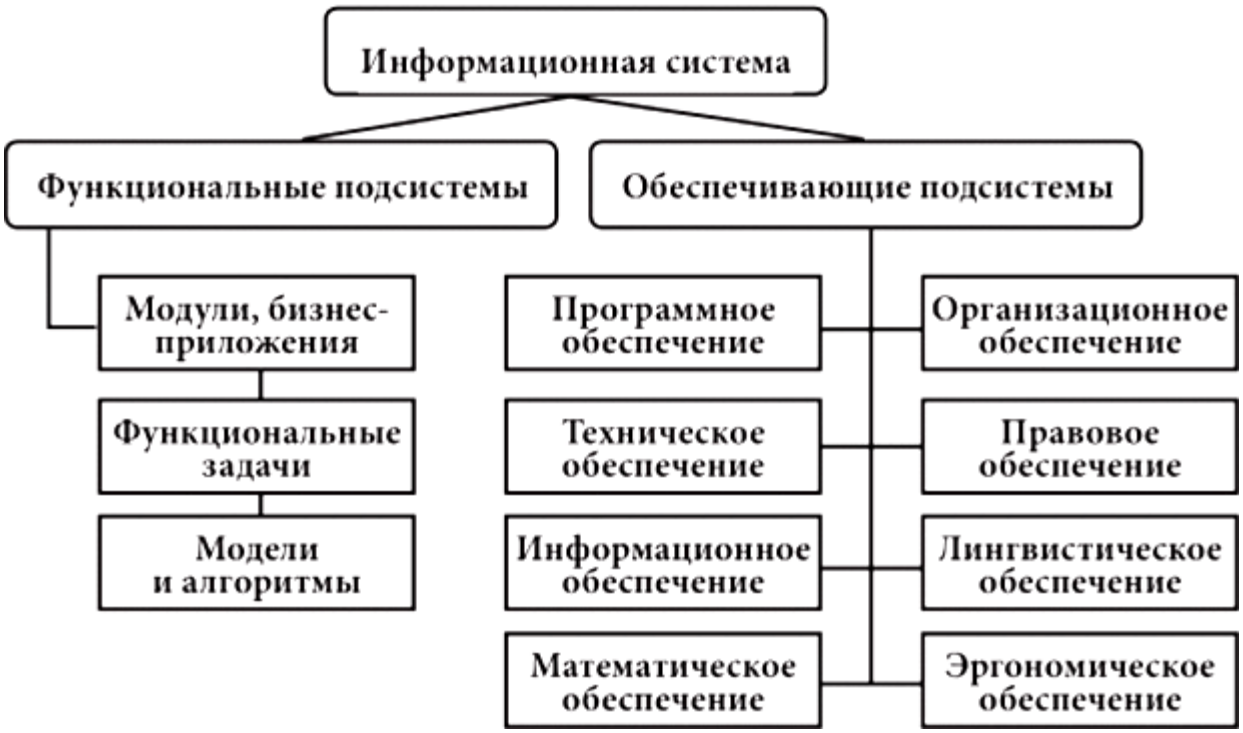


Рис. 1.3 – Структура информационной системы.

Функциональная подсистема - это подсистема, реализующая одну или несколько взаимосвязанных функций. Назначение подсистемы, ее основные задачи, цели и функции определяются видами деятельности производственных

и хозяйственных объектов: производственная, кадровая, финансовая, бухгалтерская, маркетинговая. Указанные направления деятельности и определяют типовой набор функциональных подсистем ИС (см. раздел 1.5).

Обеспечивающая подсистема - это среда, в которой используются средства для преобразования информации независимо от сферы применения. Интеграция функциональных подсистем в единую систему достигается за счет создания и функционирования обеспечивающих подсистем, таких как программная, техническая, организационная, правовая, информационная, эргономическая, лингвистическая и математическая подсистемы

Обеспечивающие подсистемы.

Программное обеспечение - это комплекс программ, реализующих функции ИС и обеспечивающий взаимодействие всех частей информационной системы при ее функционировании и решении задач пользователя.

Программное обеспечение делится на два комплекса (рис.1.4):

–базовое (общесистемное) программное обеспечение: операционные системы, сетевые операционные системы, управляющие работой серверов и сетью, программные среды для разработки прикладных программ, сетевые программы, антивирусные программы, тестовые и диагностические программы);

– прикладное программное обеспечение (совокупность прикладных программ, разработанных для конкретных задач в рамках функциональных подсистем).

Рис. 1.4 – Классификация программных средств ИС.

Техническое обеспечение - это комплекс технических средств, предназначенных для обработки данных в ИС; методические и руководящие материалы, техническая документация; обслуживающий эти технические средства персонал. В состав технического обеспечения входят компьютеры, средства сбора и регистрации информации, средства передачи данных по каналам связи, средства накопления и хранения данных и выдачи результатной информации, вспомогательное оборудование и организационная техника.

1. Средства сбора и регистрации информации:

–персональные компьютеры для ввода информации, данных, документов и их запись в базы данных. При вводе информации применяются аппаратные и программные методы контроля достоверности, в том числе контроль на диапазон значений, контроль формата значений и другие;

–сканеры для автоматического считывания информации документов в виде графических символов, распознавания графических образов и преобразования в текст;

–автоматические датчики информации для формирования сигналов наступления контролируемых событий и их преобразования в цифровое представление;

–получение и обработка данных из сопряженных функциональных и обеспечивающих подсистем и внешних информационных систем.

2. *Комплекс средств передачи информации* (технические и программные средства компьютерных сетей):

–локальные компьютерные сети ограниченного масштаба, с большими скоростями передачи данных, ограничением количества и местоположения пользователей;

–глобальные компьютерные сети, в том числе сеть Интернет, для всемирных коммуникаций и создания информационных сообществ с неограниченным кругом пользователей;

–корпоративные компьютерные сети расширенного масштаба на основе технологий VPN (Virtual Private Network – виртуальная частная сеть), специализированного назначения, с относительно высокими скоростями передачи данных, доступных только корпоративным пользователям сети;

3. *Средства хранения данных*. Базы данных ИС в виде совокупности файлов хранятся на серверах БД, файловых серверах, локальных компьютерах. В качестве носителей информации используются: специализированные серверы и устройства хранения данных большой емкости.

4. *Средства обработки данных*. Обработка информации в ИС выполняется с помощью компьютеров, различной производительности и форм фактора от смартфона и персонального компьютера, все ресурсы которых полностью направлены на обеспечение деятельности одного работника до суперкомпьютеров (для крупных корпораций).

5. *Средства вывода информации*. Для отображения и вывода информации используются видеомониторы, принтеры, графопостроители.

Информационное обеспечение представляет собой совокупность

проектных решений по объемам, размещению, формам организации информации, циркулирующей в ИС (информационные потоки), а также методология построения баз данных. Она включает в себя совокупность показателей, справочных данных, классификаторов и кодификаторов информации, унифицированные системы документации, специально организованные для обслуживания, массивы информации на соответствующих носителях.

Назначение информационного обеспечения состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений.

Центральным компонентом информационного обеспечения является база данных, через которую осуществляется обмен данными различных задач. База данных обеспечивает интегрированное использование различных информационных объектов в функциональных подсистемах.

Каждая информационная система формирует свою собственную информационную среду, в которой циркулируют потоки информации (рис.1.5).

Информационную среда включает компоненты внешнего информационного обеспечения (классификаторы технико-экономической информации, кодификаторов информации, справочные данные, унифицированные системы документации) и компоненты внутреннего информационного обеспечения информационной системы (данные, электронные документы, справочник, экранные формы для ввода/вывода информации, структура информационной базы). В нее также входит персонал, обеспечивающий надежность хранения, своевременность и качество технологии обработки информации.

Схемы информационных потоков отражают маршруты движения информации и ее объемы, места возникновения первичной информации и использования результатной информации.

Рис. 1.5 – Информационные потоки информационной системы предприятия.

В информационной среде формируются следующие потоки:

–информационный поток из внешней среды в систему управления (нормативная, правовая информация, информация о конъюнктуре рынка –

спросе на товары, конкурентах, поставщиках и т.д.);

– информационный поток из системы управления во внешнюю среду (информация о финансовом, производственно-хозяйственном состоянии предприятия государственным органам, инвесторам, кредиторам, потребителям и т.д.);

– информационный поток из системы управления на объект управления (совокупность распорядительной информации для осуществления производственных, финансовых и хозяйственных процессов: планы, приказы, задания и т.п.);

–информационный поток от объекта управления в систему управления (учетная информация о состоянии объекта управления, о наличии и качестве всех видов произведенной продукции и имеющихся ресурсов).

Организационное обеспечение – совокупность методов и средств, регламентирующих взаимодействие работников с техническими средс-вами и между собой в процессе разработки и эксплуатации информационной системы.

Организационное обеспечение является одной из важнейших подсистем ИС, от которой зависит успешная реализация целей и функций системы.

Организационное обеспечение реализует следующие основные функции:

–анализ существующей системы управления организацией, где будет использоваться ИС, и выявление задач, подлежащих автоматизации;

–подготовку задач к решению на компьютере, включая техническое задание на проектирование ИС и технико-экономическое обоснование ее эффективности;

·–разработку управленческих решений по составу и структуре организации, методологии решения задач, направленных на повышение эффективности системы управления.

В составе организационного обеспечения можно выделить четыре группы компонентов.

Первая группа включает важнейшие методические материалы, регламентирующие процесс создания и функционирования системы:

- общеотраслевые руководящие методические материалы по созданию ИС;
- типовые проектные решения;
- методические материалы по организации и проведению пред- проектного обследования на предприятии;
- методические материалы по вопросам создания и внедрения проектной документации.

Вторым компонентом в структуре организационного обеспечения ИС является совокупность средств, необходимых для эффективного проектирования и функционирования ИС (типовые пакеты прикладных программ, типовые структуры управления предприятием, унифицированные системы документов, общесистемные и отраслевые классификаторы и т.п.).

Третьим компонентом подсистемы организационного обеспечения является техническая документация, получаемая в процессе обследования,

проектирования и внедрения системы: технико-экономическое обоснование, техническое задание, технический и рабочий проекты и документы, оформляющие поэтапную сдачу системы в эксплуатацию.

Четвертым компонентом подсистемы организационного обеспечения является персонал, где представлена организационно-штатная структура проекта, определяющая, в частности, состав главных конструкторов системы и специалистов по функциональным подсистемам управления.

Правовое обеспечение – совокупность правовых норм, определяющих создание, юридический статус, эксплуатацию и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации.

В состав правового обеспечения входят законы, указы, постановления государственных органов власти, приказы, инструкции и другие нормативные документы министерств, ведомств, организаций, местных органов власти. В правовом обеспечении можно выделить общую часть, регулирующую функционирование любой информационной системы, и локальную часть, регулирующую функционирование конкретной системы. Правовое обеспечение также включает совокупность юридических документов с констатацией регламентных отношений по формированию, хранению, обработке промежуточной и результатной информации системы.

К правовым документам, действующим на этапе создания системы, относятся: договор между разработчиком и заказчиком; документы, регламентирующие отношения между участниками процесса создания системы, нормативные акты, связанные с договорными отношениями разработчика и заказчика и правовым регулированием отклонений от договора.

К правовым документам, создаваемым на этапе внедрения, относятся: характеристика статуса создаваемой системы; правовые полномочия подразделений ИС; правовые полномочия отдельных видов процессов обработки информации; правовые отношения пользователей в применении технических средств.

Правовое обеспечение этапов функционирования информационной системы включает: статус информационной системы; права, обязанности и ответственность персонала; правовые положения отдельных видов процесса управления; порядок создания и использования информации и др.

Лингвистическое обеспечение. В состав лингвистического (языкового) обеспечения входит множество языков, используемых в базе данных и СУБД (см. глава 2), совокупность научно-технических терминов, применяемых в процессе разработки и функционирования ИС, а также набор словарей, образующих словарный запас информационной системы.

Языковые средства делятся на две группы: традиционные языки (естественные, математические, алгоритмические языки, языки моделирования) и языки, предназначенные для диалога с компьютером (информационно-поисковые языки, языки СУБД, языки операционных сред, входные языки пакетов прикладных программ).

В лингвистическом обеспечении предусмотрены: способы организации диалога пользователей с программными средствами ИС в виде меню; средства исправления ошибок при взаимодействии пользователей с техническими средствами; непроцедурный язык запросов к базам данных и к услугам компьютерной сети; средства сообщения об ошибках и обработки типовых ошибок.

Эргономическое обеспечение - это совокупность методов и средств, используемых на различных этапах разработки и функционирования ИС, предназначенная для создания оптимальных условий высокоэффективной деятельности человека (персонала) в ИС, для ее быстрого освоения. Она содержит комплекс различной документации, регламентирующей эргономические требования к рабочим местам, информационным моделям, условиям деятельности персонала, а также способы реализации этих требований и осуществление эргономической экспертизы уровня их реализации.

Математическое обеспечение представляет собой совокупность математических моделей и алгоритмов для решения задач и обработки информации с применением компьютерной техники. В нее входит также комплекс средств и методов, используемых для решения экономических задач и в процессе проектирования информационных систем; техническая документация (описание задач, заданий по алгоритмизации экономико-математической модели, задач и конкретных примеров их решения); персонал (специалисты по вычислительным методам, проектировщики ИС, постановщики задач управления и т.д.). К средствам математического обеспечения также относятся: средства моделирования процессов управления; типовые задачи управления; методы математического программирования, математической статистики, теории массового обслуживания и др.

Все обеспечивающие подсистемы связаны между собой и с функциональными подсистемами. Подсистема «Организационное обеспечение» определяет порядок разработки и внедрения ИС, организационную структуру ИС и состав работников, правовые инструкции для которых содержатся в подсистеме правовое обеспечение. Функциональные подсистемы определяют составы задач и постановки задач, математические модели и алгоритмы, решения которых разрабатываются в составе подсистемы Математическое обеспечение и которые, в свою очередь, служат базой для разработки прикладных программ, входящих в состав подсистемы Программное обеспечение.

Функциональные подсистемы, компоненты математического и программного обеспечения определяют принципы организации и состав классификаторов документов, состав информационной базы. Разработка структуры и состава информационной базы позволяет интегрировать все задачи функциональных подсистем в единую информационную систему, функционирующую по принципам, сформулированным в документах

организационного и правового обеспечения.

Объемные данные потоков информации вместе с расчетными данными относительно степени сложности разрабатываемых алгоритмов и программ позволяют выбрать компоненты технического обеспечения. Выбранный комплекс технических средств дает возможность определить тип операционной системы, а разработанное программное, информационное обеспечение позволяет организовать технологию обработки информации для решения задач, входящих в соответствующие функциональные подсистемы

1.5 Корпоративные информационные системы

Любое современное предприятие (организация) осуществляет свою деятельность на основе управленческой информации. Постоянно изменяющиеся потоки всех видов ресурсов: денежных, материально-технических, трудовых требуют воздействия (управления) и без информационного отражения этих потоков решить задачи менеджмента невозможно, причем, чем полнее информационное отражение потоков ресурсов, тем выше эффективность принимаемых управленческих решений. Современные предприятия имеют сложную функциональную структуру, обусловленную территориальной распределенностью, многопрофильностью, большим числом партнеров, заказчиков (клиентов). В связи с этим информационные потоки по-разному структурированы, сложны для восприятия и не всегда доступны. Появление современных ИТ-решений позволяет изменить производственную и организационную структуру предприятия, перейти от функционально-ориентированного управления к процессо-ориентированной организации деятельности предприятия. В основе процессного подхода к управлению лежит деловой процесс (или бизнес-процесс).

Бизнес-процесс – это логически завершенный набор этапов работ, поддерживающий деятельность предприятия и реализующий его политику, направленную на достижение поставленных целей.

Совокупность бизнес-процессов, поддерживающих управление предприятием, формирует информационную логистическую систему предприятия, а управление бизнес-процессами нацелено на качественное обслуживание потребителей

Корпоративные (интегрированные) информационные системы это многоуровневая процессо-ориентированная система управления бизнес-процессами предприятия, информационными потоками, процессами принятия решений, планирования по единым правилам (стандартам) на основе интеграции методов управления и технических, программных, средств и специалистов,

Корпоративные системы позволяют решить следующие базовые задачи:

–обеспечивать большинство бизнес-процессов предприятия (нескольких

предприятий);

–осуществлять сбор, обработку и анализ информации о предприятии и внешней среде с целью решения задач управления предприятием как по вертикали (от первичной информации до поддержки принятия решений высшим руководством), так и по горизонтали (все направления деятельности и технологические операции);

–в гарантировать требуемое качество управления предприятием;

–повысить оперативность и эффективность взаимодействия между подразделениями;

–обеспечить управляемость качеством выпускаемой продукции;

–увеличить экономическую эффективность деятельности предприятия;

–создать систему статистического учета на предприятии;

–осуществлять прогноз развития предприятия;

–создать систему стратегического и оперативного планирования, систему прогнозирования.

1.5.1 Классификация корпоративных информационных систем

Корпоративные информационные системы можно разделить на два базовых класса: финансово-управленческие и производственные.

1. *Финансово-управленческие системы* включают подкласс малых интегрированных систем. Такие системы предназначены для ведения учета по одному или нескольким направлениям (бухгалтерия, сбыт, склад, кадры и т.д.). Системами этой группы может воспользоваться практически любое предприятие.

Системы этого класса обычно универсальны, цикл их внедрения невелик, иногда можно воспользоваться «коробочным» вариантом, купив программу и самостоятельно установив ее на ПК.

Финансово-управленческие системы (особенно системы российских разработчиков) значительно более гибкие в адаптации к нуждам конкретного предприятия. Часто предлагаются «конструкторы», с помощью которых можно практически полностью перестроить исходную систему, самостоятельно или с помощью поставщика установив связи между таблицами БД или отдельными модулями.

2. *Производственные системы* (также называемые системами производственного управления) включают подклассы средних и крупных интегрированных систем. Они предназначены в первую очередь для управления и планирования производственного процесса. Учетные функции, хотя и глубоко проработаны, играют вспомогательную роль, и порой невозможно выделить модуль бухгалтерского учета, так как информация в бухгалтерию поступает автоматически из других модулей.

Эти системы функционально различны: в одной может быть хорошо развит производственный модуль, в другой - финансовый. Для внедрения

системы нужна целая команда из финансовых, управленческих и технических экспертов. Производственные системы значительно более сложны в установке (цикл внедрения может занимать от 6 - 9 месяцев до полутора лет и более). Это обусловлено тем, что система покрывает потребности всего предприятия, и это требует значительных совместных усилий сотрудников предприятия и поставщиков программ.

Производственные системы часто ориентированы на одну или несколько отраслей и/или типов производства: серийное сборочное (электроника, машиностроение), мелкосерийное и опытное (авиация, тяжелое машиностроение), дискретное (металлургия, химия, упаковка), непрерывное (нефтедобыча, газодобыча).

Специализация отражается как в наборе функций системы, так и в существовании бизнес - моделей данного типа производства. Наличие встроенных моделей для определенного типа производства отличает производственные системы друг от друга. У каждой из них есть глубоко проработанные направления и функции, разработка которых только начинается или вообще не ведется.

Производственные системы по многим параметрам значительно более жестки, чем финансово-управленческие. Основное внимание уделяется планированию и оптимальному управлению производством. Эффект от внедрения производственных систем проявляется на верхних эшелонах управления предприятием, когда становится видна вся картина его работы, включая планирование, закупки, производство, сбыт, запасы, финансовые потоки и другие аспекты.

Корпоративные информационные системы условно можно разделить на две большие категории по используемым методам управления:

1. *Учетные системы*, построенные на основе журнала хозяйственных операций, которые подразделяются;

− на локальные для ведения учета на отдельном рабочем месте, без организации компьютерной сети. Имеют ограниченное число функций. Используются для малых предприятий, ИП;

– распределенные, с обязательной организацией сетевого взаимодействия между функциональными и организационными подсистемами, самые развитые из которых способны поддерживать решения большого класса учетных задач крупного предприятия, вплоть до международной корпорации, осуществляющей многовалютный учет и сложную консолидацию данных.

2. Системы управления производством, в той или иной степени реализующие один или несколько стандартов управления (см. ниже.), которые подразделяются:

– на «тяжелые» системы, имеющие максимально возможную функциональность (например, SAP R/3, Oracle Application, BAAN, Microsoft Dynamics и др.);

− на специализированные «легкие» системы, обязанные своим

появлением «закону 80 – 20»: 80 % всех предприятий использует в целом около 20 % возможностей «тяжелой» системы. Как правило, такие системы поддерживают ограниченное количество типов производств (обычно одно), легче и быстрее внедряются и стоят дешевле (например, JD Edwards, Syteline ERP и др.).

Также различают следующие виды корпоративных информационных систем:

Заказные КИС. Под заказными КИС обычно понимают системы, создаваемые для конкретного предприятия, не имеющего аналогов и не подлежащие в дальнейшем тиражированию. Подобные системы используются либо для автоматизации деятельности предприятий с уникальными характеристиками либо для решения крайне ограниченного круга специальных задач. Заказные системы, как правило, либо вообще не имеют прототипов, либо использование прототипов требует значительных его изменений, имеющих качественный характер. Разработка заказной КИС характеризуется повышенным риском в плане получения требуемых результатов.

Тиражируемые (адаптируемые) КИС. Суть проблемы адаптации тиражируемых КИС, то есть в приспособлении к условиям работы на конкретном предприятии в том, что в конечном итоге каждая КИС уникальна, но вместе с тем ей присущи и общие, типовые свойства. Требования к адаптации и сложность их реализации существенно зависят от проблемной области, масштабов системы. Даже первые программы, решавшие отдельные задачи автоматизации, создавались с учетом необходимости их настройки по параметрам.

Простые (“коробочные”) КИС реализуют небольшое число бизнес-процессов организации. Типичным примером систем подобного типа являются бухгалтерские, складские и небольшие торговые системы наиболее широко представленные на российском рынке. Например, системы таких фирм как 1С, Инфин и т.д.

Отличительной особенностью таких продуктов является относительная легкость в усвоении, что в сочетании с низкой ценой, соответствием российскому законодательству и возможностью выбрать систему “на свой вкус” приносит им широкую популярность.

Системы среднего класса отличаются большей глубиной и широтой охвата функций. Данные системы предлагают российские и зарубежные компании. Как правило, это системы, которые позволяют вести учет деятельности предприятия по многим или нескольким направлениям: финансы, логистика, кадры, сбыт.

Эти системы больше всего подходят для средних и некоторых крупных предприятий в силу своей функциональности и более высокой, по сравнению с первым классом, стоимости. Из российских систем данного класса можно выделить, например, продукцию компаний Галактика, ТБ.СОФТ

К *высшему классу* относятся системы, которые отличаются высоким уровнем детализации хозяйственной деятельности предприятия. Современные

версии таких систем обеспечивают планирование и управление всеми ресурсами организации (ERP-системы). Как правило, при внедрении таких систем производится моделирование существующих на предприятии бизнес-процессов и настройка параметров системы под требования бизнеса.

На российском рынке имеется большой выбор КИС высшего класса, и их число растет. Признанными мировыми лидерами являются, например, R/3 фирмы SAP, Oracle Application компании Oracle.

Классификация корпоративных информационных систем по уровням управления. Корпоративные информационные системы высшего класса («тяжёлые» системы) в организации можно разделить по уровням (рис. 1.6), базовым функциям управления и процессам обработки информации.

Рис.1.6 – Классификация КИС по уровням управления.

Информационная инфраструктура корпоративной информационной системы представляет собой совокупность технически и программных средств, линий связи, компьютерных сетей, персонала, общая структура которых деятельность осуществляется на основе стандартов и методологий управления ИТ-инфраструктурой. В частности к ним относится методология ITIL (IT Infrastructure Library (библиотека инфраструктуры информационных технологий).

В библиотеке ITIL (Information Techno-logy Infrastructure Library) разъясняется, что надо сделать для организации управления ИТ-услугами (Information Technology Service Management, ITSM) В ITIL описывается, как

должна быть организована деятельность ИТ-структур, Использование ITIL предполагает эффективное предоставление ИТ-услуг конечному пользователю. Другими словами, взаимодействие основной организации и ИТ-департамента строится на договорной основе, и, кроме того, четко определены качественные и количественные требования к ИТ-услугам. Такой подход предполагает дифференциацию функций и децентрализацию управления в организации. В частности, ИТ-отдел должен обладать как широкими полномочиями в своей области, так и ответственностью за предоставление качественных услуг. Таким образом, стандарт ITIL регулирует взаимодействие ИТ-отдела и потребителей ИТ-услуг остальных структурных подразделений организации использующих ИТ –услуги в виде корпоративной информационной системы.

Система автоматизации документооборота и делопроизводства (система электронного документооборота) - организационно-техническая система, обеспечивающая процесс создания, управления доступом и распространения электронных документов в компьютерных сетях, а также обеспечивающая контроль над потоками документов в организации. Задача подобных информационных систем — интеграция новых сведений в организацию и помощь в обработке бумажных документов. Эта система помогают специалистам, работающим с данными, повышают продуктивность и производительность работы инженеров и специалистов. Система электронного документооборота вследствие своей относительной простоты и многопрофильности активно используются работниками любого организационного уровня. Наиболее часто их применяют работники средней квалификации: бухгалтеры, секретари, специалисты руководители начального и среднего уровня. Основная цель – обработка данных, документов, повышение эффективности их работы и упрощение канцелярского труда.

Системы оперативного управления, обработки данных ориентирована на выполнение и обработку данных оперативного учета и регулирования хозяйственных операций (бухгалтерский, учет, учет материальных, производственных и финансовых средств), подготовки первичных документов.

К задачам ИС на этом уровне следует отнести запросы о текущем состоянии учета, отслеживание потока всех видов операций, что соответствует оперативному управлению. Чтобы с этим справляться, информационная система должна быть легкодоступной, непрерывно действующей и предоставлять точную информацию. Задачи, цели и источники информации на операционном уровне заранее определены и в высокой степени структурированы. Решение запрограммировано в соответствии с заданным алгоритмом. Эти задачи имеют итеративный, регулярный характер, выполняются непосредственными исполнителями хозяйственных процессов (специалистами, кладовщиками, администраторами и т.д.) и связаны с обработкой, оформлением и пересылкой данных, документов в соответствии с четко определенными алгоритмами. Результаты выполнения хозяйственных операций через экранные формы вводятся в базу данных. Горизонт оперативного управления хозяйственными процессами составляет от одного до

несколько дней и реализует регистрацию и обработку событий, например оформление и мониторинг выполнения заказов, приход и расход материальных ценностей на складе, ведение табеля учета рабочего времени.

Информационная система оперативного уровня является связующим звеном между организацией и внешней средой. Если система работает плохо, то организация либо не получает информации извне, либо не выдает информацию. Кроме того, система – это основной поставщик информации для остальных типов информационных систем в организации, так как содержит и оперативную, и архивную информацию.

Система аналитического управления ориентирована на тактический уровень управления: среднесрочное планирование, анализ и организацию работ в течение нескольких недель (месяцев), например анализ и планирование поставок, сбыта, составление производственных программ. Для данного класса задач характерны регламентированность (периодическая повторяемость) формирования отчетных документов и четко определенный алгоритм решения задач, например свод заказов для формирования производственной программы и определение потребности в комплектующих деталях и материалах на основе спецификации изделий. Решение подобных задач предназначено для руководителей различных служб предприятий (отделов материально-технического снабжения и сбыта, цехов).

Системы поддержки принятия решений используют в основном на верхнем уровне управления (руководства фирм, предприятий, организаций), имеющего стратегическое долгосрочное значение в течение года или нескольких лет. К таким задачам относятся формирование стратегических целей, планирование привлечения ресурсов, источников финансирования, выбор места размещения предприятий и т.д. Реже задачи класса СППР решаются на тактическом уровне, например при выборе поставщиков или заключении контрактов с клиентами. Задачи СППР имеют, как правило, нерегулярный характер.

Для задач СППР свойственны недостаточность имеющейся информации, ее противоречивость и нечеткость, преобладание качественных оценок целей и ограничений, слабая формализуемость алгоритмов решения. В качестве инструментов обобщения чаще всего используются средства составления аналитических отчетов произвольной формы, методы статистического анализа, экспертных оценок и систем, математического и имитационного моделирования. При этом применяются базы обобщенной информации, информационные хранилища, базы знаний о правилах и моделях принятия решений.

Стратегические информационные системы корпоративного типа (Enterprise Strategic System — ESS) обеспечивают поддержку принятия решений по реализации стратегических перспективных целей развития организации и предназначены для оказания помощи высшему руководству компании (Top Managers) в процессе поддержки принятия стратегических решений. Эти системы учитывают долгосрочные изменения, происходящие в

окружающей среде и деловом окружении предприятия, интегрируют в себе знания и данные всех информационных систем предприятия и строятся, как правило, на базе систем искусственного интеллекта (экспертных систем). Их назначение – приводить в соответствие изменения в условиях эксплуатации с существующей организационной возможностью.

Для функционирования ESS необходимо:

–создание единого информационного пространства и эффективной развитой коммуникационной инфраструктуры;

–внедрение новых форм и методов управления на основе современных информационных технологий и концепции управления качеством;

–кардинальное сокращение времени, необходимого на прохождение информации, требующейся для принятия решения;

–введение единого стандарта работы с электронными документами, учитывающего существующую нормативную базу и обеспечивающего защищенность, управляемость и доступность документов;

–автоматизация и повышение эффективности работы сотрудников и подразделений путем внедрения специализированных приложений и средств поддержки групповой работы;

–создание инфраструктуры управления корпоративными отраслевыми знаниями.

Информационные системы стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, подобные описанным выше, осуществлять долгосрочное планирование. Основная задача – сравнение происходящих во внешнем окружении изменений с существующим потенциалом организации. Они призваны создать общую среду компьютерной и телекоммуникационной поддержки решений в неожиданно возникающих ситуациях. Используя самые совершенные программы, эти системы способны в любой момент предоставить информацию из многих источников. Для некоторых стратегических систем характерны ограниченные аналитические возможности.

1.5.2 Стандарты управления корпоративных информационных систем

Информационные технологии управления неуклонно развиваются в соответствии с требованиями системы, применяемыми методами управления, прогрессом в области информатики и вычислительной техники.

В системах управления предприятиями применяют различные методы управления, основанные на конкретных алгоритмах подготовки и принятия управленческих решений с использованием информационных систем и технологий. Информационные системы управления неуклонно развиваются в соответствии с требованиями системы, применяемыми методами и стандартами управления, прогрессом в области компьютерной технии и развития компьютерных сетей.

В корпоративных информационных системах управления предприятиями применяют различные методы (стандарты) управления, основанные на конкретных алгоритмах подготовки и принятия управленческих решений с использованием информационных технологий.

1.***Планирование потребности в материалах (Material Requirement Planning - MRP I).*** Главной задачей MRP-систем является обеспечение наличия на складе необходимого количества требуемых материалов (комплектующих изделий) в любой момент времени в рамках периода планирования. Программные системы, реализованные на базе MRP-методологии, позволили оптимально регулировать поставки комплектующих изделий для производства продукции, контролировать складские запасы и саму технологию производства. Кроме того, использование MRP-систем позволило уменьшить объем постоянных складских запасов.

2. ***Планирование ресурсов производства (Manufacturing Resource Planning - MRP II).*** Планирование ресурсов производства (MRPII) является усовершенствованным методом планирования всех видов ресурсов предприятия, продолжением и расширением замкнутого цикла MRP. Важнейшая установка стандарта MRP II - обеспечение руководящего персонала необходимой информацией для принятия управленческих решений.

Система MRP II обеспечивает поддержку следующих функций управления предприятием:

–бизнес-планирование;

–планирование продаж и операций;

–планирование производства;

–формирование главного календарного плана производства;

–планирование потребности в материалах;

–планирование потребности в мощностях;

–система поддержки исполнения планов для производственных мощностей и материалов.

ИС, реализованная на базе MRP II (рис.1.7), предназначена для эффективного планирования всех ресурсов предприятия (включая финансовые и кадровые). Основная суть MRP II-концепции состоит в том, что прогнозирование, планирование и контроль производства осуществляется по всему жизненному циклу продукции, начиная от закупки сырья и заканчивая отгрузкой продукции потребителю.

Задачей информационных систем класса MRP II является оптимальное формирование потока материалов (сырья), полуфабрикатов (комплектующих) и готовых изделий. Система имеет целью интеграцию основных процессов, реализуемых предприятием: планирование и контроль выполнения плана, затраты, снабжение, производство, продажа, управление запасами, загрузка основных средств и т. д.

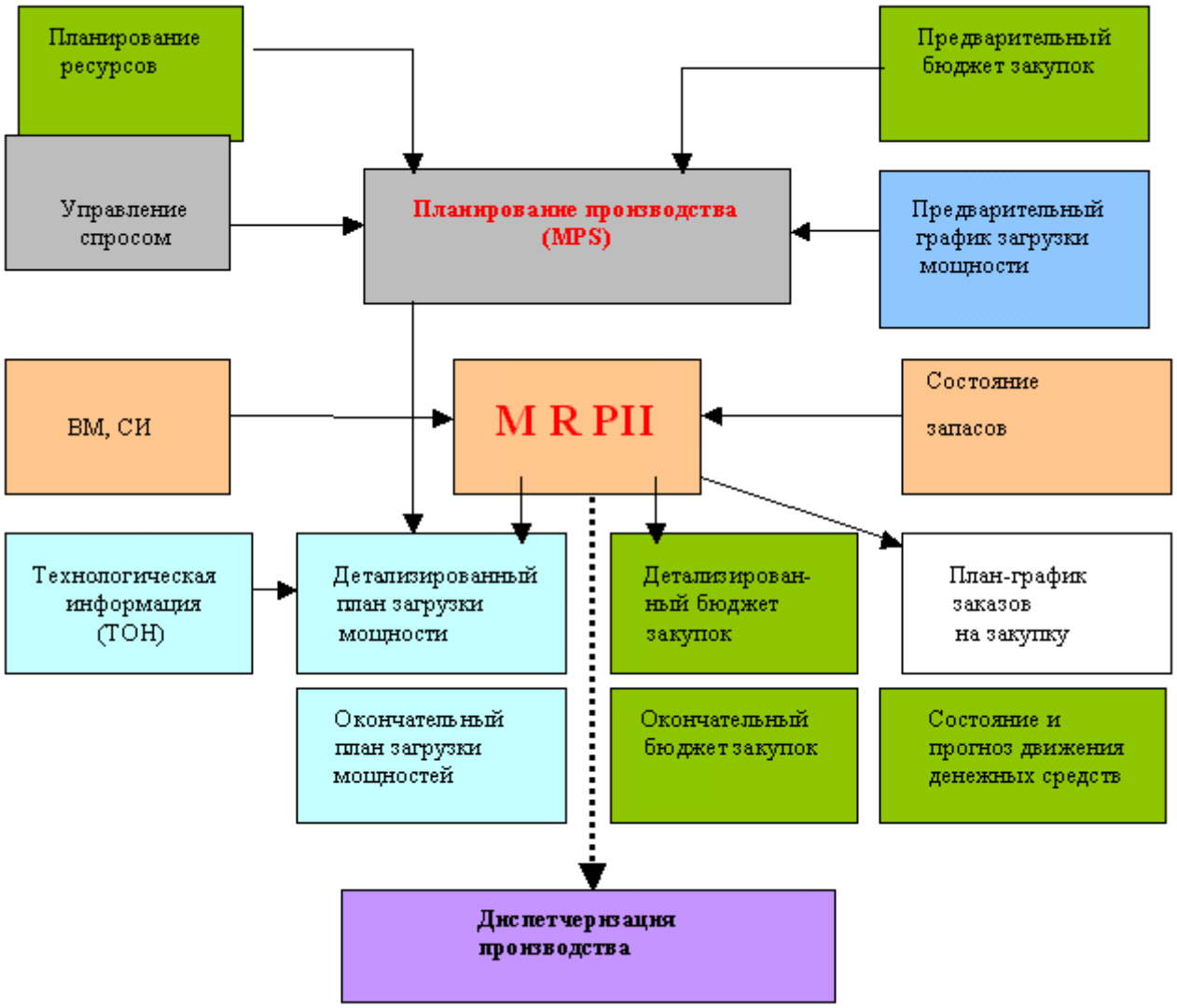


Рис. 1.7 – Структура системы MRP II.

В MRPII-системе интегрировано большое число модулей, результат работы которых анализируются MRPII-системой в целом, что и обеспечивает ее гибкость по отношению к различным внешним факторам, например, текущему спросу на продукцию. В результате применения MRPII-систем реализуются:

–оперативное получение информации о текущих результатах деятельности предприятия как в целом, так и с полной детализацией по отдельным заказам, видам ресурсов, выполнению планов;

– долгосрочное, оперативное и детальное планирование деятельности предприятия с возможностью корректировки плановых данных на основе оперативной информации;

–оптимизация производственных и материальных потоков со значительным сокращением непроизводственных затрат и реальным сокращением материальных ресурсов на складах;

–отражение финансовой деятельности предприятия в целом.

3. ***Планирование ресурсов предприятия (MRP II & FRP (Finance Resource Planning), Enterprise Resource Planning — ERP I***) – организационная стратегия интеграции производства и операций, управления трудовыми

ресурсами, финансового менеджмента и управления активами, ориентированная на непрерывную балансировку и оптимизацию ресурсов предприятия посредством специализированного интегрированного пакета прикладного программного обеспечения, обеспечивающего общую модель данных и процессов для всех сфер деятельности.

ERP-система – это пакет интегрированных приложений, реализующий стратегию ERP, позволяющий создать интегрированную информационную среду (ИИС) для автоматизации планирования, учета, контроля и анализа всех основных бизнес-операций предприятия.

В основе ERP-систем лежит принцип создания единого хранилища (репозитария) данных, содержащего всю корпоративную бизнес-информацию: финансовую информацию, производственные данные, данные по персоналу и т.д. Наличие такого корпоративного репозитария устраняет необходимость в передаче данных от одной подсистемы к другой (например, от производственной подсистемы к финансовой), а также обеспечивает одновременную доступность к информации любого числа сотрудников предприятия, обладающих соответствующими полномочиями. Следует отметить, что ряд зарубежных аналитиков считает, что целью ERP-систем является не столько улучшение производственной деятельности предприятия, сколько уменьшение усилий на поддержку его внутренних информационных потоков. ERP-системы предназначены для управления всей финансовой и хозяйственной деятельностью предприятия. Они используются для оперативного предоставления руководству предприятия информации, необходимой для принятия управленческих решений, а также для создания инфраструктуры электронного обмена данными предприятия с поставщиками и потребителями.

4.***Оптимизации управления ресурсами (ERP II***) - это стратегия разработки и внедрения приложения, которая распространяется за пределы ERP-функций, чтобы обеспечить интеграцию ключевой для предприятия специфики, внутреннего и внешнего сотрудничества, операционных и финансовых процессов. Таким образом, ERP II начинается, прежде всего, как стратегия разработки приложения, которая нацелена на интеграцию в рамках предприятия всех бизнес-процессов, ориентированных на коммерцию. А как стратегия внедрения, ERP II позволяет пользователям ориентироваться на одного производителя лишь в той степени, в которой через интеграционные возможности собственно ERP II обеспечиваются обязательные для выполнения требования к процессам предприятий, при этом возможно подключение отдельных, лучших в своем классе, компонент от сторонних производителей.

Как правило, ERP II-системы создаются для отраслей и отдельных направлений бизнеса, модель открытого взаимодействия обеспечивает интеграцию с другими приложениями, поддержку многочисленных стандартов и протоколов межплатформенного взаимодействия (языки Java, XML, ASP, технологии Corba, COM, система электронной документации EDI и т.д.).

В ERP II-системы включены функциональные компоненты электронного бизнеса, реализованные как веб-приложения:

1) SRM (Supplier Relationship Management) – система управления взаимоотношениями с поставщиками (снабжение) для закупок ресурсов;

2)CRM (Customer Relationship Management) – система управления связями с клиентами (сбыт) для сбыта и реализации продукции;

3)SCM (Supply Chain Management) – система управления виртуальными логистическими цепочками для доставки ресурсов или продукции;

4) BI (Business Intelligence) — система бизнес-аналитики для формирования аналитических отчетов и оценки бизнес-процессов;

5) PLM (Product Lifecycle Management) – система управления жизненным циклом продукта;

6) HRM (Human Resource Management) — система управления человеческими ресурсами;

и целый ряд других систем.

Применение ERP II-систем на сегодняшний день фактически стало обязательным для крупных и средний предприятий. ERP II –направлена на обеспечение сотрудничества с другими корпорациями в том числе и в непроизводственной сфере. Они позволяют достичь максимальной производительности предприятия.

Особенности ERP 2-систем

–полная автоматизация функций системы управления в режиме реального времени;

–открытость системы, поддержка взаимодействия с внешними информационными системами на базе стандартных технологий и программных интерфейсов;

–единое информационное пространство для принятия управленческих решений, высокий уровень качества информации для реализации функций управления, современные информационные технологии обработки данных;

–независимость от масштаба предприятия и их расположения;

–высокая надежность функционирования КИС, защита данных от несанкционированного доступа, других угроз целостности и сохранности данных, дружественный пользовательский интерфейс и др.

5. ***Планирование потребности в производственных мощностях (Capacity Resource Planning – CRP***). Система выполняет планирование и балансировку загрузки рабочих центров, производственных мощностей рабочих центров (оборудования, поточных линий, бригад рабочих и т.п.) с учетом ресурсных ограничений и планов выпуска готовой продукции. Планирование потребности в производственных мощностях осуществляется по каждому виду продукции, включенного в главный календарный план. При планировании учитывается последовательность выполнения технологических операций изготовления продукции на рабочих центрах.

CRP являются основой для построения систем MRP и MRP 2.

Информационные системы классов CRP/MRP обеспечивают реализацию функций управления в направлении «сверху вниз» без учета обратной связи, а также решение функциональных задач планирования потребностей в материалах и производственных мощностях.

Существует целый ряд других методологий (стандартов) управления, используемых для организации корпоративных информационных систем.

1.5.3 Общие функции и свойства корпоративных информационных систем

Корпоративные информационные системы имеют следующие общие функции и свойства:

1. *Поддержка стандартов управления*: MRPII (Manufacturing Resource Planning) – планирование производственных ресурсов (материальных, трудовых, ресурсов оборудования); ERP (Enterprise Resource Planning) – полнофункциональное управление всеми видами ресурсов (материальными, трудовыми, финансовыми, ресурсами оборудования); ERP II (Enterprise Resource Planning) – полнофункциональное управление всеми видами ресурсов (материальными, трудовыми, финансовыми, ресурсами оборудования), реализация бизнес-процессов в среде Интернет; ISO-9000 — международный стандарт качества; и др.

2. *Комплексность*, взаимосвязь автоматизируемых бизнес-процессов планирования, контроля, учета и анализа деятельности предприятия.

3. *Многофункциональность (многозадачность).* Определяется модульностью построения корпоративной информационной системы и уровнями управления (см. разделы 1.5.1 и 1.5.4). Следует выделить следующие базовые функции КИС:

3.1. Функция сбора и регистрации информационных ресурсов. Сбор информации о предметной области необходим для поддержания информационной модели в адекватном состоянии.

3.2. Функция хранения информационных ресурсов. Данная функция связана, прежде всего, с необходимостью управления ресурсами хранимых данных и ресурсами памяти.

3.3. Функция актуализации информационных ресурсов. Актуализация информационных ресурсов заключается в приведении их в соответствие текущему состоянию предметной области системы.

3.4. Функция обработки информационных ресурсов. Одним из важнейших качеств КИС является возможность производства новых данных и знаний на основе уже существующих.

3.5. Функция предоставления информационных ресурсов пользователям. Цель создания КИС – это, прежде всего, удовлетворение информационных потребностей пользователей, поэтому функции обеспечения интерфейса системы с пользователем являются одной из важнейших составляющих информационной системы.

3.6. Функция планирования. Данная функция состоит в разработке и реализации планов по выполнению поставленных задач на различные сроки (год, квартал, месяц, день), например, план производства, план маркетинговых исследований, финансовый план и т.д.

3.7. Учетная функция. Эта функция заключается в разработке или использовании уже готовых форм и методов учета показателей деятельности фирмы: бухгалтерский учет, финансовый учет, управленческий учет и т.п. Другими словами, учет состоит в получении, регистрации, накоплении, обработке и предоставлении информации о реальных хозяйственных процессах.

3.8. Аналитическая функция. Анализ заключается в изучении итогов выполнения планов и заказов, выявлении резервов и тенденций развития и т.д.

3.9. Контрольная функция. Это контроль за выполнением планов, расходованием материальных ресурсов, использованием финансовых средств и т.д.

4. *Открытость и гибкость* компонентной архитектуры, специализация серверов обеспечивают:

–модульный принцип построения, позволяющий легко конфигурировать системы под конкретный заказ с последующим наращиванием;

–функциональную настройку и конфигурирование функциональных модулей;

– способность взаимодействовать с различными внешними системами (системы телекоммуникации, финансового анализа и др.), обеспечивать выбор программно-технической платформы и переносимость ее на другие аппаратные средства; обеспечивать смену аппаратной платформы, системного программного обеспечения (модели СУБД, типа операционной системы);

–высокую производительность обработки транзакций, возможность оперативной замены серверов, оптимизацию расхода вычислительных ресурсов;

–гибкость настройки модулей КИС и адаптация их к потребностям и условиям конкретной корпорации;

–возможность выбора языка интерфейса, соответствие требованиям национальных стандартов, законодательным актам; наличие лицензий и сертификатов на программное обеспечение.

5. *Масштабируемость*, предусматривающая расширение и усложнение функциональных модулей системы по мере развития бизнес-процессов (например, поддержка работы филиалов и отделений банка, углубление анализа и т.д.). К созданию КИС приступают, как правило, крупные предприятия и организации, для которых необходимо обеспечить «управляемость». Рост масштаба объекта управления в связи с возрастанием числа внутренних пользователей, увеличением интенсивности информационных потоков, ростом объемов хранимых данных, увеличением количества и размерности решаемых задач выражается в изменении требований к информационным технологиям.

6. *Поддержка корпоративных сетевых коммуникаций*. Все многообразие компьютерных сетей: локальные (ЛВС), ассоциация ЛВС, Интернет, VPN, облачные технологии обеспечивает поддержку совместной работы территориально распределенных пользователей, взаимодействие с удаленными информационными источниками, совместное использование сетевого оборудования, данных и программ.

7. *Многоплатформенностъ технологий*. Информационные технологии КИС ориентированы на использование компьютерной техники различных классов и разнородных операционных систем. В ряде случаев это многообразие является объективной основой эффективной реализации информационных технологий. Корпоративные информационные системы создаются как открытые системы, которые допускают замену и дополнение программно-технических компонентов.

8. *Поддержка специальных корпоративных информационных технологий*.

8.1. Бизнес-моделирование КИС. Бизнес-процессы КИС обладают масштабом выполняемых функций, сложной организацией взаимодействия компонентов – процедур управления (действий). Для обеспечения эффективности бизнес-процессов осуществляется их реинжиниринг (Business Process Reengineering – BPR), который основан на описании, анализе, моделировании и проектировании.

8.2. Корпоративные сети. Сеть объединяет несколько рабочих станций и различные типы серверов: сервер БД, сервер приложений (бизнес-логики), сервер представлений (презентации), сервер печати, прокси-сервер, шлюз межкорпоративных связей и др.

Специализация серверов и открытость архитектуры КИС обеспечивают высокую производительность обработки транзакций, возможность оперативной замены серверов, оптимизацию расхода вычислительных ресурсов и т.п.

8.3. Сервис-ориентированная архитектура приложений. Приложения функционируют как распределенные в сети Интернет /VPN.

9. *Интеграция предприятий с внешней средой*. Процессы в КИС реализованы в виде потоков бизнес-операций обработки бизнес-объектов, содержащих: ядро – данные (свойства) объекта; бизнес-логику объекта – набор правил и ограничений (методы обработки объекта); интерфейс – независимое от платформы описание бизнес-объекта для его применения во внешних информационных Системах. Для бизнес-объекта применяются разнообразные технологии доступа: компонентная модель объектов – COM (Component Object Model), распределенная компонентная модель объектов – DCOM (Distributed COM), удаленный вызов процедур (функций и методов обработки объекта– RFC (Remote Function Call) и др. Интерфейс программирования бизнес-приложений B API (Business Application Program Interface) обеспечивает обработку бизнес-объектов, создание библиотек классов объектов и связанных с ними методов обработки.

10. *Обеспечение высокого качества информации для принятий*

управленческих решений. Отличительной особенностью КИС является комплексность, взаимосвязь автоматизируемых бизнес-процессов планирования, контроля, учета и анализа деятельности предприятия. Система обладает открытостью и гибкостью компонентной архитектуры, состоит из ряда интегрированных модулей, объединенных в контуры (подсистемы) управления.

11. *Надежность и защищенность*, которые обеспечивают:

–надежность системы определяется возможностью информационной системы сохранять работоспособность и обеспечить высокую устойчивость ее работы при отказах отдельных модулей КИС, при возникновении системных и программных ошибок, ошибочных действий персонала, ошибок в условиях дестабилизирующих факторов (выход из строя оборудования, линий связи, компьютерных сетей);

–многопользовательский доступ к данным в реальном времени (Системы типа on-line) и реализация функций в едином информационном пространстве;

– санкционированный доступ пользователей к информационным ресурсам (базам данных, документам);

– разделение прав доступа к различным функциям и данным при многопользовательской работе, идентификацию и аутентификацию пользователей;

– создание профилей пользователей (параметры рабочего сеанса пользователя: размеры и цвета экранных окон; параметры экранных шрифтов; структура и доступные пункты генерального меню); и др;

– наличие средств защиты данных от действия пассивных и активных угроз информационной безопасности, компьютерных вирусов;

–наличие средств восстановления при сбоях. В КИС должны быть предусмотрены средства для прогноза, фиксации и локализации различных нештатных ситуаций и отказов оборудования, таких как: выявления угроз информационной безопасности, повреждений и перегрузок каналов связи, перегрузок устройств внешней и оперативной памяти, нарушения бесперебойного энергоснабжения, нарушения целостности БД и др.

1.5.4 Базовые функциональные модули корпоративных информационных систем

Функциональные модули корпоративных информационных систем строятся на основе стандартов управления MRP/ERP-систем**.** Модульный принцип организации позволяет внедрять корпоративные информационные системы поэтапно, последовательно переводя в эксплуатацию один или несколько функциональных модулей, а также выбирать только те из них, которые актуальны для организации. Кроме того, модульность корпоративных информационных систем позволяет строить решения на основе нескольких

MRP/ERP-систем, выбирая из каждой лучшие в своём классе модули (англ. best-of-breed). Разбивка по модулям и их группировка различная, но у большинства основных поставщиков выделяются группы модулей: финансы, персонал, инжиниринг, операции и ряд других.

Так в корпоративной информационной системе, реализованной на базе стандарта MRP II, должны быть реализованы следующие основные функциональные модули различных уровней управления:

– Планирование продаж и производства (Sales and Operation Planning);

–Управление спросом (Demand Management);

– Составление плана производства (Master Production Scheduling);

– Управление складом (Inventory Transaction Subsystem);

– Плановые поставки (Scheduled Receipts Subsystem);

– Управление на уровне производственного цеха (Shop Flow Control);

–Планирование производственных мощностей (Capacity Requirement Planning);

– Материально-техническое снабжение (Purchasing);

–Планирование распределения ресурсов (Distribution Resourse Planning);

– Планирование и контроль производственных операций (Tooling Planning and Control);

– Финансовое планирование (Financial Planning);

– Моделирование (Simulation);

– Оценка результатов деятельности (Performance Measurement).

Указанные модули различных уровней управления, как правило являются частью более крупных модулей, иерархически и функционально связанных между собой.

Финансы. Финансовые инструменты обеспечивают мониторинг финансовых событий в реальном масштабе времени, ведение бухгалтерского и финансового учета в российских и международных стандартах, контроль и управление на всех уровнях организации для поддержки принятия решений. Ядро этого контура составляют правила, создаваемые на основе учетной политики, бухгалтерский учет. Основным учетным регистром является журнал хозяйственных операций, а также регистры, справочники, различные специализированные инструменты бухгалтерского и финансового учета.

Среди финансовых модулей фигурирует множество различных функциональных блоков, в разных системах и разных версиях выделяются различные их компоновки, среди наиболее часто встречающихся (по организационным подразделениям):

–бухгалтерские: главная книга, счета к получению (дебиторы), счета к оплате (кредиторы), консолидация;

–учётно-управленческие, контроллинговые: учёт затрат и доходов по местам возникновения, по продуктам, по проектам, калькуляция себестоимости;

–казначейские: управление ликвидностью, управление движением денежных средств (включая банковские счета и кассу), взаимодействие с

банками, управление долгом и заимствованиями;

–финансово-управленческие: управление основными средствами, инвестиционный менеджмент, финансовый контроль и управление рисками.

Также иногда в состав финансовых модулей включены финансовое планирование и управление ключевыми показателями эффективности, но основные разработчики поставляют для этих функций отдельные специализированные программные продукты.

***Персонал* (*зарплата и кадры*).** В состав модуля управления персоналом входит: кадровый учёт, учёт рабочего времени (табельный учёт), управление нарядами на работы, командировками, расчёт производительности трудовых ресурсов, управление оплатой труда, премиями, компенсациями и расчёт заработной платы, пенсионный учёт, оценка персонала, управление квалификацией (профессиональными навыками, обучением), подбор персонала, возможность оперативного планирования и управления операциями с учётом информации о доступности персонала, возможности точно рассчитывать затраты по местам возникновения и продуктам в согласовании с информацией о компенсации задействованного персонала.

Модуль обеспечивает управление персоналом как человеческим капиталом организации, в части возможности ведения информации о профессиональных навыках, планирования обучения, карьеры сотрудников и обеспечив применимость информации, обрабатываемой в этих модулях, для целей стратегического управления организацией, расчёта ключевых показателей эффективности, финансового менеджмента.

Инжиниринг (проектные работы). На предприятии выполняются проектные и опытно-конструкторские работы для выпуска новой продукции. С помощью инжиниринга осуществляется управление проектированием и созданием новых видов продукции, поддержка технологических процессов изготовления изделий, учет и техническое обслуживание производственных ресурсов. К основным функциям инжиниринга относятся:

–в ведение БД конструкторских изделии для проектирования, конфигурационного управления и отслеживания технологии изготовления изделий;

–ведение БД «Основные фонды» (технологическое оборудование);

–ведение БД «Маршрутные карты технологических процессов изготовления продукции»;

–в ведение БД «Технологическая оснастка» для конструирования оснастки;

–управление жизненным циклом изделия и др.

Логистика. Логистические цепочки представляют собой последовательную реализацию следующих функций: логистические снабжение, управление взаимоотношениями с поставщиками, управление цепочками поставок и транспортировкой, управление запасами, складами, инвентаризацией сбыт, закупка, планирование потребностей в материалах, техническое обслуживание и ремонт. Иногда логистические системы разделены

на логистику закупок, производства, сбыта и хранения.

Производство. В большинстве КИС реализуется стандарт класса MRP II, ориентированный на базовые модели управления: производство на склад, сборка, изготовление и конструирование под заказ. Предприятия могут иметь различные типы производства: поточное (массовое), серийное, заказное, единичное, партионное (порционное), производство с непрерывным циклом. К основным функциям управления модуля относятся:

–поддержка полного жизненного цикла продукции (Product Lifecycle Management – PLM) – для готовой продукции рассматриваются технологические маршруты, конструкторская спецификация, производственное оборудование, квалификация персонала, управление техническим обслуживанием и ремонтами оборудования, планирование мощностей, управление транспортом и т.п.;

–планирование потребности в материалах MRP – выполняется с учетом производственной программы, складских запасов, запланированных | поступлений материалов и отгрузки готовой продукции;

–планирование потребностей в производственных мощностях, определение реальных сроков выполнения производственной программы с учетом производственных мощностей; расчет загрузки производственных ресурсов;

–формирование производственной логистики на уровне цехов предприятия с дискретным и процессным производством (детальное планирование, контроль выполнения, контроль качества и отслеживание единиц произведенной продукции); и др.

Контроллинг затрат на продукт – расчет себестоимости продукции различными методами.

Показатели качества продукции отслеживаются по всей цепочке, начиная от поставок сырья и материалов, комплектующих изделий на стадиях снабжения и до планирования и контроля выполнения производственных заказов, сбыта и распределения продукции.

Продажи. Функциям системы управления продажами КИС основаны на системе ***управления взаимоотношениями с клиентами*** – CRM (Customer Relationship Management). Основной функцией CRM-системы является обеспечение организации и ее внутренних сотрудников входящей и исходящей информацией о клиенте. Под входящей информацией понимается вся совокупность данных, известных организации о клиенте, под исходящей же – рекомендации, либо конкретные поручения сотрудникам организации о дальнейших действиях или алгоритмах поведения, связанных с конкретным клиентом.

CRM – это стратегия, определяющая взаимодействие с клиентами во всех организационных аспектах: она касается рекламы, продажи, доставки и обслуживания клиентов, дизайна и производства новых продуктов, выставления счетов и т. п. Эта стратегия основана на выполнении следующих условий:

–наличие единого хранилища информации и системы, куда помещаются и где в любой момент доступны все сведения обо всех случаях взаимодействия с клиентами;

–синхронизированность управления множественными каналами взаимодействия (т. е. существуют организационные процедуры, которые регламентируют использование этой системы и информации в каждом подразделении компании);

–постоянный анализ собранной информации о клиентах и принятие соответствующих организационных решений, например, о ранжировании клиентов исходя из их значимости для компании, выработке индивидуального подхода к клиентам согласно их специфическим потребностям и запросам.

CRM повышают эффективность торговых и маркетинговых операций и в частности позволяет:

–привлекать всё новых клиентов, управлять маркетинговыми кампаниями и вести продажи;

–выявлять потенциальных клиентов и возможные сделки и в результате увеличивать объемы продаж;

–систематизировать стратегию продаж путём централизации и координации бизнес-процессов;

–достичь нужного уровня обслуживания клиентов и формировать по каждому клиенту единое представление с учётом всей имеющейся о нем информации;

–быстро реагировать на запросы клиентов, отвечая на них в режиме реального времени, углубить отношения с клиентами за счет постоянного доступа к расширенным функциям управления данными о них и о возможностях продаж;

– вывести управление информацией на новый уровень благодаря такой возможности, как подписка мобильных пользователей на конкретные данные CRM в зависимости от их деловых приоритетов;

–задействовать централизованные средства планирования и управления для контроля ресурсов и координации обслуживания.

–упорядочивать маркетинговую деятельность с помощью интеллектуальных средств организации списков и сегментного анализа;

–пользуясь аналитическими инструментами, организовывать выверенные маркетинговые кампании;

–выявлять фактические потребности клиентов, определять эффективность деятельности сотрудников и эффективность взаимодействия клиентов по множеству показателей с помощью специализированных средств анализа и создания отчетов.

Компоненты общего назначения. К компонентам общего назначения КИС относятся: управление электронным документооборотом; управление проектами (Project Management); оценка эффективности бизнеса; информационно-технологический компонент и др.

В ОГЛАВЛЕНИЕ

Глава 2. Основы теории баз данных

2.1 Классификация данных

Основой любой информационной системы являются данные, которые должны быть определенным образом организованы с целью адекватного отображения непрерывно меняющегося реального мира и эффективного удовлетворения информационных нужд пользователей данной системы.

Данные, согласно общепринятой терминологии, представляют собой некоторый факт, на котором основан вывод или любая интеллектуальная система.

Данные – это информация, фиксированная в определенной форме пригодной для последующей обработки, хранения и передачи. Данные это набор конкретных значений, параметров, характеризующих информационный объект. Данные не обладают определенной структурой, данные становятся информацией тогда, когда пользователь задает им определенную структуру, то есть осознает их смысловое содержание.

Первичными компонентами данных являются цифры и символы естественного языка или их кодированное представление в виде строки двоичных битов. Наименьшей семантически значимой поименованной единицей данных является ***элемент данных.***

Поименованная совокупность элементов данных, рассматриваемая в программе обработки данных как единое целое, носит название ***агрегата данных.***

Агрегатами данных могут являться, например, дата, фамилия-имя-отчество. Упорядоченная совокупность значений взаимосвязанных элементов данных называется логической записью.

Данные в своей совокупности характеризуют некую предметную область.

Предметная область – набор объектов, совокупность конкретных и абстрактных понятий, между которыми существуют определенные связи.

Объектом предметной области могут быть либо человек, либо событие, либо предмет, который характеризуется конечным набором качественных и количественных параметров, которые представляются элементами данных.

Элементам данных присущи следующие два свойства: свойство избыточности и свойство полноты. *Свойство избыточности* заключается в том, что один элемент данных может входить в разные записи. *Свойство полноты* состоит в том, что любой элемент данных может характеризовать полностью или частично предмет, явление, событие.

Данные могут быть отнесены к одному из двух типов основному или составному (рис.1).

Основной (простой) тип данных – это символы, числа и другие элементы, дальнейшее дробление которых не имеет смысла. Символьные данные, в свою очередь разделены на неструктурированные, частично

структурированные и структурированные (семантические сети, обычный текст и построение по модели).

–числовые. Примеры значений данных: 0.43, 328, 2E+5;

–символьные (алфавитно-цифровые). Примеры значений данных: "пятница", "строка", "программист";

–даты, задаваемые с помощью специального типа "Дата" или как обычные символьные данные. Примеры значений данных: 1.12.07, 23/2/07;

–временные и дата-временные, предназначенные для хранения информации о времени и/или дате. Примеры значений данных: 31.01.85 (дата), 9:10:03 (время), 6.03.1960 12:00 (дата и время);

–символьные переменной длины, предназначенные для хранения текстовой информации большой длины, например, документа.

Рис. 2.1 – Классификация типов данных.

Составной (структурный) тип данных, конструируемый пользователем для решения конкретных задач.

Статические структуры – структуры данных, которые занимают в памяти ПК постоянный объем, к ним относятся массив, запись, множество:

- Массив (функция с конечной областью определения) - простая совокупность элементов данных одного типа, средство оперирования группой данных одного типа. Отдельный элемент массива задается индексом. Массив может быть одномерным, двумерным и т.д. Разновидностями одномерных массивов переменной длины являются структуры типа кольцо, стек, очередь и двухсторонняя очередь.
- Запись (декартово произведение) - совокупность элементов данных разного типа. В простейшем случае запись содержит постоянное количество элементов, которые называют полями. Совокупность записей одинаковой структуры называется файлом. (Файлом называют также набор данных во внешней памяти, например, на магнитном диске). Для того, чтобы иметь возможность извлекать из файла отдельные записи, каждой записи присваивают уникальное имя или номер, которое служит

ее идентификатором и располагается в отдельном поле. Этот идентификатор называют ключом.

- двоичные, предназначенные для хранения графических объектов, аудио- и видеоинформации, пространственной, хронологической и другой специальной информации. в формате BMP (Bitmap) и автоматически их отображать при работе с БД;

Динамические структуры – структура данных, которые могут изменять свою длину, к ним относятся дерево, список, ссылка.

- гиперссылки (hyperlinks), предназначенные для хранения ссылок на различные ресурсы (узлы, файлы, документы и т. д.), находящиеся вне базы данных, например, в сети Internet, корпоративной сети intranet или на жестком диске компьютера. Примеры значений данных: http:\\www.chat.ru, ftp:\\chance4u.teens.com.
- дерево используется для размещения элементов которой требуется нелинейное адресное пространство. Существует большое количество структур данных, которые могут быть представлены как деревья. Это, например, классификационные, иерархические, рекурсивные и др. структуры.

Структурно данные могут быть представлены тремя уровнями (рис.2): концептуальным, внешним и внутренним. Концептуальный уровень отражает объективные свойства данных, описывающих предметную область. Внешний уровень, напротив, отражает субъективные взгляды приложений на данные. В практических случаях внешний уровень является подмножеством концептуального представления. Внутренний уровень представления данных определяет машинно-ориентированное, физическое представление данных. В структуре концептуальный уровень находится между внешним и внутренним.

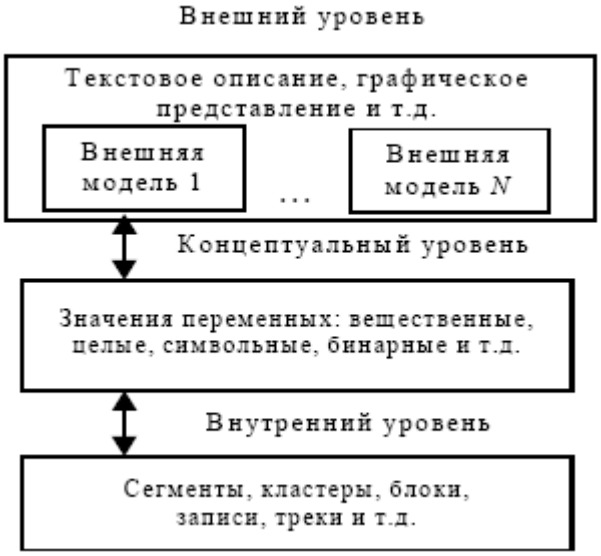


Рис. 2.2 – Уровни представления данных.

Рассмотрим в качестве примера уровни представления списка или списков деталей, входящих в состав изделия. Как видно из схемы (рис. 2), на

концептуальном уровне могут быть представлены не обязательно данные из списка деталей − это может быть и бухгалтерский документ. Одновременно внешний уровень отражает наше восприятие. Здесь решающим является квалификация, например инженера-конструктора. Внутренний уровень не играет в решении данной задачи главной роли и может меняться в зависимости от типа и уровня развития компьютерной техники.

2.2 Банки и базы данных, требования к ним, структура и классификация

Структура и общие функции банка данных.

В основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются информационные системы (ИС). Автоматизированными информационными системами называют системы, в которых применяют технические средства, в частности компьютерные системы. Большинство существующих ИС являются автоматизированными.

Современной формой информационных систем являются банки данных – это система специально организованных данных, программных, языковых, организационных и технических средств, предназначенных для централизованного накопления, хранения и многоцелевого использования данных.

Основными функциями банков данных являются:

–хранение данных и их защита;

–изменение (обновление, добавление и удаление) хранимых данных;

–поиск и отбор данных по запросам пользователей;

–обработка данных и вывод результатов.

Термин "банк данных" не является общепризнанным. Наиболее близким к нему в англоязычной литературе является термин "система баз данных" (data base system). Система баз данных включает базу данных, СУБД, соответствующее оборудование и персонал. Понятие "система баз данных" уже, чем банк данных, так как "банк" обозначает то, что хранится в нем и всю инфраструктуру, но по сути они одинаковы.

Основные требования к банкам данных.

Адекватность информации состоянию предметной области. Банки данных являются информационной моделью предметной области. Хранимая в нем информация должна полно и точно отображать ее объекты, их свойства и отношения между ними. Отступление от принципа адекватности делает систему бесполезной и даже опасной, недопустимой для использования. В свою очередь, требование адекватности порождает ряд новых требований к системе, таких, как необходимость постоянного внесения изменений в данные и периодического изменения организации данных.

Надежность функционирования – одно из важнейших требований, предъявляемых к любой системе.

Быстродействие и производительность. Первое определяется временем ответа (реакции) системы на запрос, исчисляемым с момента ввода запроса до момента начала выдачи найденных данных. Это время зависит не только от быстродействия ЭВМ, но и от способов физической организации данных, методов доступа, способов поиска, сложности запроса и других факторов. Второе требование определяется количеством запросов, выполняемых в единицу времени.

Быстрота и удобство использования. Это требование предъявляется со стороны всех категорий пользователей, особенно конечных. Сложность запросов, отсутствие сервиса формируют в психологии пользователя нежелание работать с информационной системой.

Массовость использования. Необходимо обеспечить коллективный доступ пользователей, при котором они могут одновременно и независимо обращаться к базам данных для получения необходимых сведений.

Защита информации от случайных искажений, уничтожения, а также от преднамеренных, несанкционированных действий пользователя.

Возможность расширения. Архитектура системы должна допускать расширение ее возможностей путем модификации или замены существующих программных модулей либо добавления новых компонентов, а также путем реорганизации информационных массивов.

Компоненты банка данных (рис. 2.3)**:**

–информационная база (одна или несколько баз данных);
–лингвистические (языковые) средства (ЯОД, ЯМД, SQL);
–программные средства (СУБД, Прикладные программы);
–технические средства (вычислительная система, серверы);
–организационное и нормативное обеспечение;
–администратор банка данных.

Рис. 2.3 – Компоненты банков данных.

Информационной базой называют данные, отражающие состояние определенной предметной области и используемые информационной системой**.** Информационная база состоит из ***собственно данных*** и описания данных – ***метаданных***

Основная функция информационной базы хранение (представление) данных (информации).

Хранение информации можно рассматривать как передачу информации во времени. Различают оперативное и долговременное хранение информации. Носители информации можно разделить на оперативные и долговременные запоминающие устройства.

Фаза хранения информации может быть представлена на концептуальном, логическом и физическом уровнях.

Концептуальный уровень отражает содержательно информацию и способы реализации ее хранения (Разработчик ИС). Логический (внешний) уровень определяет порядок представления информации, организацию информационных массивов (Прикладной программист). Физический (внутренний) уровень означает реализацию хранения информации на конкретных физических носителях (СУБД).

Автоматизация информационной базы фазы хранения информации осуществляется за счет процессов кодирования.

Информационная база состоит из информационного обеспечения и баз данных.

Информационное обеспечение – совокупность системы классификации и кодирования информации, входных документов и вспомогательных информационных массивов, процедур организации информационных массивов, алгоритмизации процессов ввода, поиска, вывода и обновления информации.

База данных – это поименованная совокупность данных, организованных по определенным правилам (структурированная), отображающая состояние объектов и их взаимосвязей в рассматриваемой предметной области, предусматривающая общие принципы описания, хранения, манипулирования данными, независимая от прикладных программ.

Также используется следующее определение:

База данных представляет собой совокупность структурированных данных, предназначенных для машинной обработки и использования многими пользователями.

Государственным комитетом по науке и технике (ГКНТ) в 1982 г. был принят ряд документов, определяющих **базу данных** как именованную совокупность данных, отражающую состояние объектов и их отношений в рассматриваемой предметной области.

На уровне информационных систем база данных определяется как совокупность файлов операционной системы, содержащих данные о предметной области.

Термин база данных начал применяться с 1963 г. и записывался на английском языке как data base. По мере развития вычислительной техники, эти два слова были объединены в одно (database). Основной смысл, вкладываемый в термин "база данных" – это база информационной системы. Однако единого мнения по поводу определения термина "база данных" пока не существует.

Организация БД отличается от организации обычного файла тем, что описание полей (метаданные) хранится вместе с данными; и данные структурированы.

Структурирование – это введение соглашений о способах представления данных. Неструктурированными называют данные, записанные, например, в текстовом файле.

Лингвистические средства используются для описания данных на каждом уровне, описания БД и её частей, программирования задач для обращения к БД (рис.2.4).

Рис.2.4 – Лингвистические средства базы данных.

В состав лингвистического обеспечения входит ***множество языков,*** используемых в СУБД, а также ***набор словарей***, образующих словарный запас информационной системы.

Языковые средства БД обеспечивают интерфейс пользователей разных категорий с банком данных и базируются на языковых средствах СУБД.

Типы языков:

Язык описания данных (ЯОД) – высокоуровневый язык декларативного типа, внутрисистемного определения данных, предназначенный для описания логической структуры данных (модель данных и их отношений) (DDL (SDL) Data (Schema) Definition Language);

Язык манипулирования данными – совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку данных по запросам (DML – Data Manipulation Language).

SQL – структурированный язык запросов, представляющий собой стандартизованное средство описания запросов к базам данных (Structured Query Language), организации и обработки данных.

Словарь данных представляет собой специальную систему в составе банка данных, предназначенную для хранения единообразной и централизованной информации обо всех ресурсах данных конкретного банка данных. В словаре данных содержатся сведения: об объектах, их свойствах и отношениях для данной предметной области; о данных, хранимых в базе данных (их наименования, смысловое описание, структура, связи с другими данными); о возможных значениях и форматах представления данных; об источника возникновения данных; о кодах защиты и разграничениях доступа к данным со стороны пользователей.

Программные средства – сложный комплекс, обеспечивающий взаимодействие всех частей информационной системы при ее функционировании и настраиваемые на задачи пользователя (рис.2.5).

Это программы, которые используются для обработки данных, управлением данными и обработкой, а также взаимодействие с операционной системой, в среде которых функционирует банк данных и набор сервисных программ, необходимых для выполнения вспомогательных операций и решения пользовательских задач. Основное программное средство СУБД – система управления базами данных (см. раздел 4).

Рис.2.5 – Программные средства базы данных.

В состав программного обеспечения входят общесистемные и специальные программные продукты, а также техническая документация.

К общесистемному программному обеспечению относятся комплексы программ, ориентированных на пользователей и предназначенных для решения типовых задач обработки информации. Они служат для расширения функциональных возможностей компьютеров, контроля и управления процессом обработки данных.

Специальное программное обеспечение представляет собой совокупность программ, разработанных при создании конкретного банка данных. В его состав входят пакеты прикладных программ (ППП), реализующие

разработанные модели разной степени адекватности, отражающие функционирование реального объекта.

Техническое обеспечение – это все те аппаратные средства, которые обеспечивают функционирование банка данных и работу пользователей.

Комплекс технических средств составляют:

–компьютеры разных моделей;

–устройства сбора, накопления, обработки, передачи и вывода информации;

–устройства передачи данных и линий связи;

–оргтехника и устройства автоматического съема информации;

–эксплуатационные материалы и др.

Организационно-нормативное обеспечение представляет собой комплекс мероприятий и руководящих документов, определяющих организацию повседневной эксплуатации банка данных и эффективное информационное обслуживание пользователей.

Нормативное (правовое) обеспечение — совокупность правовых норм, определяющих создание, юридический статус и функционирование банка данных информационных систем, регламентирующих порядок получения, преобразования и использования информации.

К организационным средствам банка данных относятся различные инструкции, методические и регламентирующие документы, для пользователей различных категорий.

Администратором банка данных называется группа специалистов, обеспечивающих разработку, функционирование, управление и развитие систем баз данных.

Общая классификация банков и баз данных.

Банки или системы баз данных являются сложными системами и их классификация может быть проведена по различным признакам, относящимся как в целом к банку данных, так и к его компонентам. Большинство классификационных признаков относится к центральной компоненте банка данных – базе данных. Признаки классификации баз данных тесно пересекаются с признаками классификации информационных систем.

Классификация по сфере применения:

– системы оперативности обработки данных обработки транзакций (OLTP - OnLineTransactionProcessing), для отражения актуального состояния предметной области в любой момент времени. Транзакция неделимая последовательность операций с данными в СУБД Все действия, составляющие транзакцию, должны либо выполниться полностью, либо полностью не выполниться;

– системы аналитической обработки информации (принятия решений); OLAP(OnLineAnalyticalProcessing); средства доступа, отбора, просмотра и анализа информации и прогнозирования её будущего состояния;

–информационно-справочные системы основанные на гипертекстовых документах и мультимедиа предназначенные для накопления и поиска по различным критериям документов, записанных на естественном языке;

–офисные информационные системы – автоматизацию делопроизводства и управление документооборотом.

По форме представления и типу хранимой информации – фактографические, документальные, мультимедийные, XML – базы данных, гипертекстовые, геоинформационные, в том числе картографические (символьные, цифровые данные).

По топологии хранения и технологии обработки данных.

Централизованная (локальная) база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе

Распределенная база данных – это совокупность множества взаимосвязанных БД, распределенная по сети географически удаленных узлов и обеспечивающая распределение ресурсов компьютеров, одновременный и независимый доступ к данным.

Системы технологии обработки данных с сетевым доступом предполагают различные *архитектуры* подобных систем: (файл-сервер, клиент-сервер, сервер приложений) .

По типу используемой модели данных – иерархические, сетевые, реляционные, объектно-ориентированные, постреляционные.

По способу доступа к данным базы данных разделяются на базы данных с *локальным доступом* и базы данных с *удаленным (сетевым)* доступом.

По характеру использования хранимой информации на специализированные и интегрированные (разнородные данные и разнотипные базы).

По функциональному назначению (характеру решаемых с помощью БД задач) – операционные (управление процессами) и справочно-информационные.

Классификация баз и банков данных с точки зрения информационных процессов:

Системы управления базами данных и программирования АИС– к ним относят СУБД в чистом виде, СУБД с элементами программирования АИС (ORACLE) и системы программирования АИС с элементами СУБД

***Автоматизированные информационно*-поисковые системы** (АИПС)– собственно АИПС (фактографические, документальные) и пакеты прикладных программ для разработки АИПС.

Основные типы баз данных.

Фактографические БД – оперируют фактическими данными, представленными в виде хорошо структурированных и форматизированных (точных по содержанию) данных (сетевые, иерархические, реляционные). Такие

базы данных задают четкую схему соответствий и связей (отношений) между данными, изменение которых связаны с изменением структуры базы.

Состоят из СУБД, хранилища данных, системой отбора данных по запросам.

Фактографические базы данных используются для реализации разнообразных справочных функций и для решения задач обработки данных (АСУ, СУБД). Ввод, хранение, сортировка, отбор по запросам, генерация отчетов.

Документальные базы данных служат для работы со слабоструктурированными данными, как правило, с текстами, документами на естественном языке. Документальные базы данных не требуют четкого знания типов отношений, изменение которых не связаны со структурными преобразованиями базы.

Наиболее распространенным видом документальных систем являются информационно-поисковые системы (ИПС), предназначенные для накопления и поиска по различным критериям документов, записанных на естественном языке. Основными компонентами таких систем служат:

–программные средства, предназначенные для организации ввода и хранения информации, поддержки общения пользователя с системой, обработки запросов на поиск документов и выдачи результатов поиска;

–поисковый массив документов обычно называют базой данных. Он чаще всего не содержит непосредственно текстов документов. Наиболее широко распространенные ИПС – библиографические системы – оперируют, как правило, только библиографическими описаниями и значительно реже рефератами или аннотациями документов.

Объектно-ориентированные базы данных (ООБД) –системы баз данных, построенных на основе объектно-ориентированного программирования, В ООБД данные и правила для их обработки организованы в объекты, которые имеют свойства и набор методов работы с объектом..

В данном типе баз данных нет ограничений на типы и структуру данных. Объекты с одинаковыми свойствами объединяются в классы. Разные классы могут иметь одинаковые методы. В ООБД пользователь просто объявляет связь, и СУБД автоматически генерирует методы управления, динамически создавая, удаляя и пересекая связи.

Гипертекстовая база данных – нелинейная сетевая неструктурированная форма (с возможной иерархической структурой) организации данных, разделенных на фрагменты (объекты – текст, документ, графика, видео, аудио). Каждый фрагмент, имеет связи (переход) с другими фрагментами, позволяет уточнить данные или двигаться в одном или нескольких направлениям по выбранной связи. Типы связей семантические. Система запросов отсутствует. Поиск по темам, тезарусу, словарю.

Гипертекст содержит не только информацию, но и аппарат ее эффективного поиска. По степени формализации информации гипертекстовые БД занимают промежуточное положение между документальными и фактографическими БД.

Структурно гипертекстовая БД состоит из информационного материала (заголовок и текст), тезауруса гипертекста (отображает семантические связи и предназначен для поиска слов по их смысловому содержанию), списка главных тем (заголовков) и алфавитного словаря информационных статей.

Основополагающим принципом, на базе которого формируются гипертекстовые БД, является принцип общезначимости. Согласно этому принципу в информационный материал включают лишь суждения, справедливые относительно всех объектов, соответствующих заголовку статьи. Общие суждения, неспецифические для данного заголовка, должны помещаться в статье по более широкой родовой теме.

Распределенная база данных – это совокупность множества взаимосвязанных баз данных, распределенная по сети географически удаленных узлов и обеспечивающая распределение ресурсов компьютеров, одновременный и независимый доступ к данным. **Распределенная база данных** - совокупность узлов, каждый из которых оперирует своей локальной базой данных, способная обрабатывать транзакции, затрагивающие данные нескольких узлов.

XML–базы данных используется для хранения, поиска, обработки, модификации, передачи XML документов (данных) и компонентов XML-модели и не является реальной самостоятельными базами данных в привычном смысле этого слова. XML–базы данных, как правило, применяются для хранения дата-ориентированных и документ-ориентированных данных (например XHTML, или DocBook), данных, которые имеют очень сложную структуру с глубокой вложенностью, но также используются и для хранения слабоструктурированных данных.

Мультимедийные БД - это совокупность связанных друг с другом упорядоченным образом мультимедийных данных. Мультимедийные данные включают один или несколько основных типов медиаданных, таких как текст, изображения, графические объекты (включая чертежи, эскизы и иллюстрации), анимационные последовательности, аудио и видео. Мультимедийная база строится на основе объектно-ориентированных или объектно-реляционных СУБД и семантических связях. Специфика мультимедийных данных требует использования устройств хранения большой емкости.

Мультимедийные БД используются для хранения и доступа к видеоконтенту, для мультимедийного обучения, в экспертных системах, в системах распознавания и т.д.

Основные принципы построения баз и банков данных.

Принцип интеграции данных состоит в объединении отдельных, взаимно несвязанных данных в единое целое, в роли которого выступает база данных, в результате чего, пользователю и его прикладным программам, все данные представляются единым информационным массивом. Интеграцию данных необходимо рассматривать на двух уровнях – логическом и физическом. На логическом уровне множество структур данных отображается в единую структуру данных, на физическом уровне автономные файлы объединяются в базу данных.

Принцип централизации управления состоит в передаче всех функций управления данными единому комплексу управляющих программ – СУБД. Все операции, связанные с доступом к БД, выполняются не прикладными программами, а централизованно – ядром СУБД - на основании информации, получаемой из этих программ.

Остальные принципы в той или иной степени связаны с принципами интеграции и централизации.

Принцип целостности данных отражает требование адекватности хранимой в БД информации состоянию предметной области: в любой момент времени – данные должны в точности соответствовать свойствам и характеристикам объектов. Нарушение целостности возникает вследствие искажения или даже разрушения всех или части данных, а также как результат записи в базу данных неверной информации. Поддержание целостности достигается контролем входной информации, периодической проверкой хранимых в БД данных, применением специальной системы восстановления данных и т.д.

Принцип независимости данных – независимость прикладных программ от хранимых данных, при которой любые изменения в организации данных не требуют коррекции этих программ. Одним из путей достижения независимости является введение дополнительных уровней абстрагирования данных (принцип многоуровневости). Вместо двух традиционных уровней, предусмотренных базовым программным обеспечением и стандартными языками программирования, - логического и физического – в архитектуре банка используется принцип трехуровневой организации данных: логический уровень делится на два – внешний (уровень пользователя) и концептуальный (общий системный уровень данных).

Принцип неизбыточности данных - состояние данных, когда каждое из них присутствует в базе данных в единственном экземпляре. Избыточность может иметь место как на логическом уровне, когда в структуре данных повторяются одни и те же типы данных, так и на физическом уровне, когда данные хранятся в двух или более экземплярах. Принцип интеграции позволяет свести избыточность к минимуму.

Принцип непротиворечивости данных – смысловое соответствие между данными; это состояние БД, при котором хранимые в ней данные не противоречат друг другу. Различают два аспекта непротиворечивости:

смысловое соответствие разнотипных данных и идентичность (равенство) дублирующих данных.

Принцип связности данных заключается в том, что данные в банке данных взаимосвязаны, и связи отражают отношения между объектами предметной области. Множество связей и множество типов данных образуют **логическую структуру данных**. Наличие связей между записями в базе данных позволяет уменьшить избыточность, упростить и ускорить поиск данных.

2.3 Системы управления базами данных

В информационных системах, в среде которых функционирует банк данных, специальных средств для создания и обработки баз данных не предусмотрено. Поэтому для обеспечения автоматизации всех операций, связанных с решением этих задач используется специальный комплекс программ – **Система управления базами данных** (СУБД) – это совокупность языковых и программных средств, предназначенная для создания, ведения и совместного использования баз данных многими пользователями.

СУБД представляет собой пакет прикладных программ, расширяющих возможности операционной системы по обработке БД.

Обычно современная СУБД содержит следующие компоненты (рис.2.6):

- Ядро – управляющая программа, предназначенная для автоматизации всех процессов, связанных с обращениям к БД. Ядро отвечает за управление данными во внешней и оперативной памяти. После запуска СУБД ее ядро постоянно находится в памяти и организует обработку поступающих запросов, управляет очередностью их выполнения, взаимодействует с прикладными программами и операционной системой, контролирует завершение операций доступа к БД, выдает сообщения. Важнейшей функцией ядра является организация параллельного выполнения запросов и протоколирование изменений (журнализация).
- Процессор языка базы данных – набор обрабатывающих программ: трансляторов с языков описания данных, языков запросов и языков программирования, редакторов, отладчиков. Обеспечивает обработку и оптимизацию запросов (трансляцию и компиляцию) на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода.
- Подсистема поддержки программных вызовов, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД.
- Сервисные программы (внешние утилиты), обеспечивающие настройку СУБД, восстановление после сбоев и др.

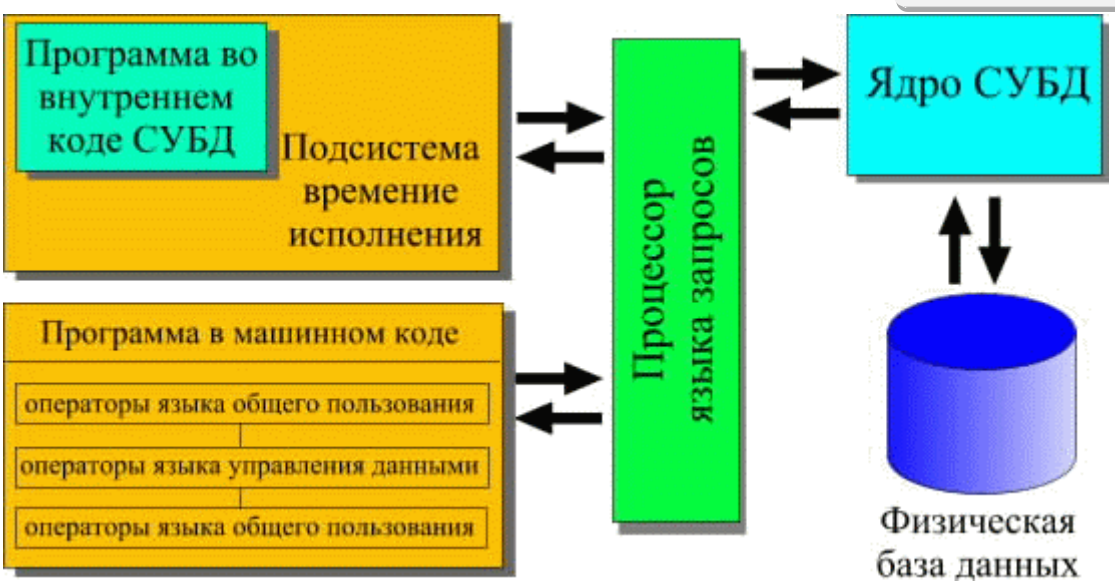


Рис.2.6 – Компоненты СУБД.

СУБД, являясь инструментальным средством и средством организации доступа к БД, не решает никаких прикладных расчетных задач. Обработка найденных системой данных, вычисления, формирование выходных документов по заданной форме выполняются с помощью прикладных программ.

СУБД классифицируются по следующим основным признакам**:**

1) по выполняемым функциям:

–операционные;

–информационные;

2) по форме применения:

–универсальные

–проблемно ориентированные;

3) по используемому языку общения:

–замкнутые, имеющие собственные, самостоятельные языки общения пользователя с базой данных;

–открытые, в которых используется язык программирования расширенный операторами языка манипулирования данными;

4) по числу поддерживаемых уровней моделей данных:

–одноуровневые системы;

–двухуровневые системы;

–трехуровневые системы;

5) по способу установления связей между данными:

–реляционные;

–иерархические;

–сетевые;

6) по способу организации хранения данных и выполнен функций обработки баз данных:

–централизованные;

–распределенные.

Существуют и другие способы классификации СУБД.

Основные функции и характеристики СУБД:

–физическое размещение данных в памяти и их описаний;

–поддержание баз данных в актуальном состоянии;

–определение данных. СУБД должна допускать определение данных в исходной форме и преобразовывать эти определения в форму соответствующих объектов;

–обработка данных. СУБД должна уметь обрабатывать запросы пользователя по поиску, выбору, изменение или удаление существующих данных в базе данных или добавление новых данных. Программное обеспечение должно быть построено таким образом, чтобы реализовать планируемые и непланируемые запросы. К планируемым относятся запросы, необходимость которых предусмотрена заранее, непланируемым (специальные) – наоборот. Планируемые запросы обычно осуществляются из написанных заранее приложений, а непланируемые запросы по определению производятся интерактивно;

–обеспечение безопасности и целостности данных. Достигается шифрованием прикладных программ, шифрованием данных, защитой данных паролем, ограничением доступа к базе данных. снижением избыточности данных, исключением ввода неверных данных, восстановлением после отказов, контролем пользовательских запросов и пресечением попыток нарушения правил безопасности и целостности данных, определяемых администратором БД;

–поддержка словаря данных. СУБД должна обеспечивать функцию словаря данных. Сам словарь данных является системной БД, содержащей данные о данных, например, исходные и объектные схемы внешнего и концептуального уровня, перекрестные ссылки программ или частей БД, отчеты для различных пользователей и т.д;

–поддержка системы управления передачей данных, которая применяется при обработке запросов пользователей, физически удаленных от СУБД;

–управление данными во внешней памяти и управление буферами оперативной памяти. Включает обеспечение необходимых структур внешней памяти как для хранения непосредственных данных, так и служебных целей, например, для хранения индексов;

– управление транзакциями. При помощи транзакций поддерживается логическая целостность БД за счет объединения элементарных операций над разными файлами в одну транзакцию. Поддержание механизма транзакций – обязательное условие однопользовательских, а тем более многопользовательских СУБД;

– восстановление и дублирования данных. Соблюдение надежности системы достигается избыточностью хранимых данных или ведением журнала изменений БД;

–поддержание языков БД (язык определения данных, язык манипулирования данными, язык SQL);

–разграничение доступа (многопользовательский режим);

–возможность экспорта и импорта данных;

–обеспечение доступа к данным с помощью SQL;

–наличие инструментальных средств разработки прикладных программ.

–обеспечение производительности, когда все перечисленные функции должны выполняться с максимально возможной эффективностью. Производительность СУБД оценивается:

–временем выполнения запросов;

–скоростью поиска информации;

–временем импортирования базы данных из других форматов;

–скоростью обновления операций (обновление, вставка, удаление);

–временем генерации отчета и другими показателями.

В зависимости от того, что является объектом управления, в СУБД предусмотрены три уровня управления (манипулирования):

–управление файлами, осуществляемое в процессе их генерации и эксплуатации. Основными операциями являются открытие и закрытие, копирование, переименование, реструктурирование, реорганизация, восстановление БД, снятие отчетов по БД;

–управление записями (кортежами), которое включает чтение, добавление, удаление и упорядочивание записей;

–управление полями записей (атрибутами).

Такие операции, как ввод данных с клавиатуры, вычисления, организация циклов, вывод данных на экран и принтер и некоторые другие, не являются сферой деятельности СУБД, а определяются в прикладных программах. Для разработки прикладных программ в СУБД предусматривается специальный язык программирования.

В соответствии с указанным набором функций в состав СУБД входят программы трех типов:

–управляющие;

–обрабатывающие (транслятор);

–сервисные.

Программы функционально взаимосвязаны и взаимодействуют друг с другом и с ОС. При запуске СУБД в основную память загружается большая часть управляющих программ (ядро). Остальные модули вызываются по мере необходимости.

2.4 Трехуровневая архитектура баз данных

Методы доступа к данным развивались на протяжении нескольких последних десятилетий от громоздких, физически ориентированных методов начального периода обработки файлов к различным видам обработки баз

данных. Одним из наиболее важных аспектов реляционной «революции» стала идея отделения логической структуры, как она понимается конечным пользователем, от физического представления, требуемого компьютерным оборудованием. Наиболее удачным способом реализации баз данных стала предложенная американским комитетом по стандартизации ANSI (American National Standards Institute) трехуровневая система организации базы данных, представленная в модели ANSI/SPFRC, которая определяет не только методы доступа к данным, но и структуру данных, их хранение, обработку (см.раздел 1), а также методы проектирования базы данных (см. раздел 6).

Трехуровневая архитектура базы данных – это стандартная структура базы данных, состоящая из концептуального, внешнего и внутреннего уровней.

Концептуальный уровень – структурный уровень базы данных, определяющий логическую схему базы данных.

Концептуальный уровень – определяет концептуальное проектирование базы данных и концептуальную схему базы данных – единое логическое описание всех элементов данных и отношений между ними. Он включает анализ информационных потребностей пользователей и определение нужных им элементов данных.

Концептуальный уровень – центральное управляющее звено, здесь база данных представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной базой данных. Фактически концептуальный уровень отражает обобщенную, логическую модель предметной области (объектов реального мира), для которой создавалась база данных. Как любая модель, концептуальная модель отражает только существенные, с точки зрения обработки, особенности объектов реального мира.

Внешний уровень – структурный уровень базы данных, определяющий пользовательские представления данных.

Внешний уровень составляют пользовательские представления данных БД. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, и наоборот, именно эти сведения используются в подсистеме отдела кадров .Каждая пользовательская группа имеет свое представление данных в базе данных. Каждое такое представление имеет ориентированное на пользователя описание элементов данных и отношений между ними. Его можно напрямую вывести из концептуальной схемы. Совокупность всех таких пользовательских представлений данных и есть внешний уровень.

Внутренний уровень – структурный уровень БД, определяющий физический вид БД.

Внутренний (физический) уровень обеспечивает физический взгляд на базу данных. Физический уровень – собственно данные, расположенные в

файлах или в страничных структурах, расположенных на внешних носителях информации: дисководы, физические адреса, индексы, указатели и т.д. За этот уровень отвечают проектировщики физической базы данных. Ни один пользователь не касается этого уровня.

Трехзвенная архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных.

Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных.

Управление доступом к базе данных происходит следующим образом (рис.2.7):

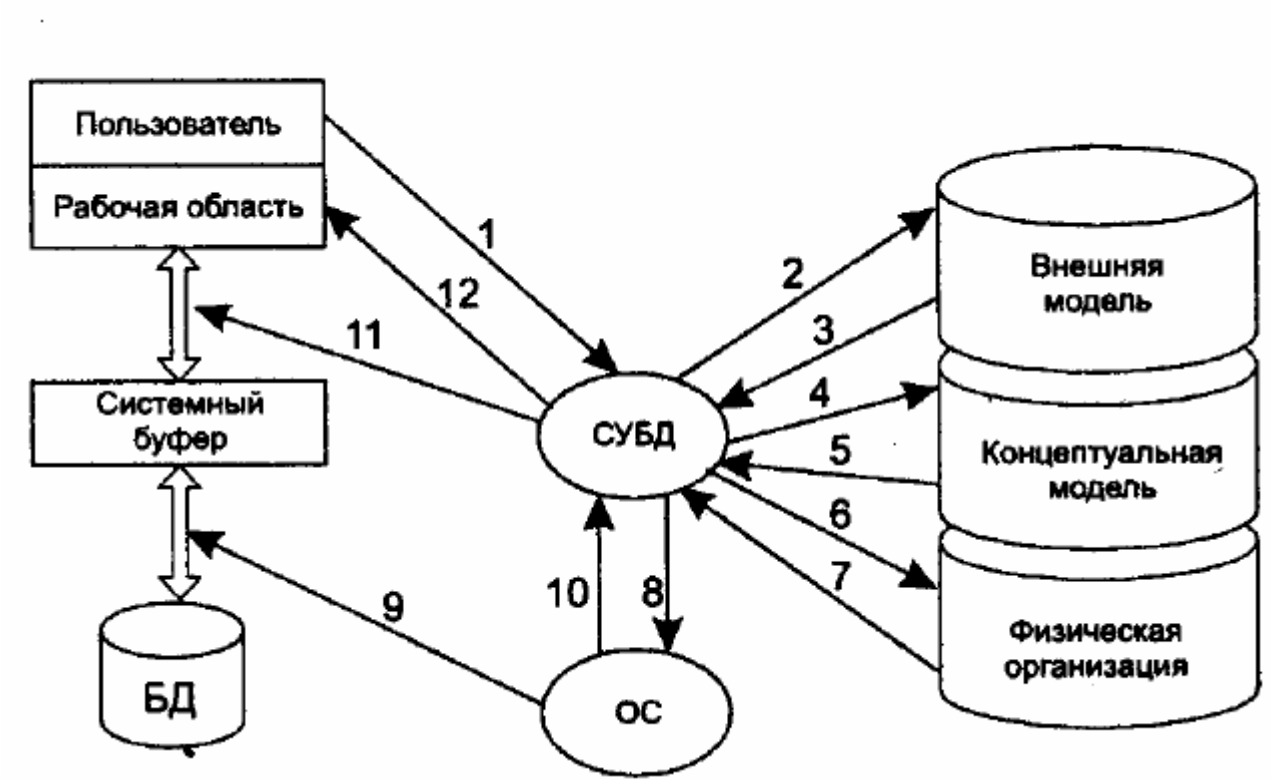


Рис.2.7– Взаимодействие пользователя, СУБД и операционной системы (ОС).

1. Пользователь посылает СУБД запрос на получение данных из БД, применяя определенный подъязык данных (обычно SQL).

2. СУБД перехватывает этот запрос и анализирует его.

3. Анализ прав пользователя и внешней модели данных, соответствующей данному пользователю, подтверждает или запрещает доступ данного пользователя к запрошенным данным.

4. В случае запрета на доступ к данным СУБД сообщает пользователю об этом (стрелка 12) и прекращает дальнейший процесс обработки данных, в противном случае СУБД определяет часть концептуальной модели, которая затрагивается запросом пользователя (стрелка 4)

5. Затем СУБД просматривает внешнюю схему для этого пользователя, концептуальную, внутреннюю схему и определяет структуру хранения.

6. СУБД получает информацию о запрошенной части концептуальной модели.

7. СУБД запрашивает информацию о местоположении данных на физическом уровне (файлы или физические адреса).

8. В СУБД возвращается информация о местоположении данных в терминах операционной системы.

9. СУБД просит операционную систему предоставить необходимые данные, используя средства операционной системы.

10. Операционная система осуществляет перекачку информации из устройств хранения и пересылает ее в системный буфер.

11. Операционная система оповещает СУБД об окончании пересылки.

12. СУБД выбирает из доставленной информации, находящейся в системном буфере, только то, что нужно пользователю, и пересылает эти данные в рабочую область пользователя.

2.5 Жизненный цикл базы данных

Под жизненным циклом базы данных понимаются этапы развития базы данных, начиная от анализа предметной области, и заканчивая сопровождением базы данных.

Среди основных этапов жизненного цикла наибольшую важность имеют разработка, эксплуатация и сопровождение. Каждый этап характеризуется определенными задачами и методами их решения, исходными данными; полученными на предыдущем этапе, и результатами.

Разработка базы данных.

Разработка базы данных, наиболее сложный и трудоемкий этап, как правило, включает в себя следующие основные этапы:

1) Стратегическое планирование.
2) Проверка осуществимости.
3) Определение требований.
4) Концептуальное проектирование.
5) Реализация (программирование).

Стратегическое планирование.

Стратегическое планирование конкретной системы осуществляется в процессе разработки стратегического плана. Проводится системный анализ и словесное описание информационных объектов предметной области. Когда

начинается разработка проекта реализации, общая информационная модель, созданная в процессе планирования, пересматривается и, если нужно, совершенствуется. В процессе планирования фирма собирает информацию для определения потребностей пользователей.

Проверка осуществимости.

Это часть жизненного цикла, определяет технологическую, операционную и экономическую осуществимость плана создания БД. Подготавливаются отчеты по следующим вопросам:

а) Технологическая осуществимость. Наличие технологий и оборудования, необходимых для реализации, запланированной БД.

б) Операционная осуществимость. Наличие специалистов и средств, необходимых для успешного осуществления плана создания БД.

в) Экономическая целесообразность.

Определение требований.

Определение требований включает выбор целей базы данных, выяснение информационных потребностей различных отделов и руководителей и требований к оборудованию и программному обеспечению. Информационные потребности выясняются с помощью анкет, опросов менеджеров и работников компании, а также форм и отчетов, которыми компания пользуется в текущий момент.

Концептуальное проектирование.

Этап концептуального проектирования включает создание концептуальной схемы БД, инфологическое и датологическое проектирование (см. раздел 7). На этом этапе последовательно проектируют инфологическую модель предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели, и даталогическую (логическую) модель базы данных.

То есть на этом этапе создаются подробные модели пользовательских представлений данных; затем они интегрируются в концептуальную модель, фиксирующую все элементы корпоративных данных, которые будет содержать база данных.

Реализация.

Реализация – шаги, которые надо выполнить для превращения концептуальной модели в функционирующую БД. На этом этапе создается физическая проектирование БД, то есть выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.

В процессе реализации БД выбирается и приобретается СУБД. Затем подробная концептуальная модель превращается в проект реализации БД; создается словарь данных, БД наполняется данными, создаются прикладные программы и обучаются пользователи.

Построение словаря данных (СД) является ключевым шагом в реализации БД, поскольку словарь данных является центральным хранилищем определений структуры данных базы данных. Словарь данных содержит информацию о

полномочиях доступа, правила защиты данных и связанного сними контроля данных. Словарь данных действует как управляющий центр системы

На этапе реализация создается проектная и эксплуатационная документация. Готовятся материалы и программные средства необходимые для проведения тестирования базы данных. Разрабатываются материалы, необходимые для организации обучения персонала.

Эксплуатация базы данных.

Эксплуатационные работы можно подразделить на подготовительные и основные. К подготовительным относятся:

–внедрение базы данных

–конфигурирование базы данных и рабочих мест пользователей;

– обеспечение пользователей эксплуатационной документацией;

– обучение персонала.

Основные эксплуатационные работы включают;

– непосредственно эксплуатацию;

– локализацию проблем и устранение причин их возникновения;

– модификацию программного обеспечения;

– подготовку предложений по совершенствованию системы;

– развитие и модернизацию системы.

Внедрение занимает, как правило, длительное время, от нескольких месяцев до нескольких лет. Осуществляется первоначальная загрузка нормативно-справочной информации, ввод в схему документооборота новых форм документов, обучение пользователей. Внедрение базы данных разбивается на опытную и промышленную стадии эксплуатации, которая начинается после приемки базы данных.

Сопровождение и развитие базы данных.

Этот этап является наиболее длительным в жизненном цикле базы данных. В процессе эксплуатации базы данных осуществляется регистрация ошибок, проводится экспертиза проектных решений, формулируются требования к модификации базы данных в связи с изменениями объекта и функций управления, появлением новых информационных технологий.

Службы технической поддержки играют весьма заметную роль в жизни любой корпоративной информационной системы. Наличие квалифицированного технического обслуживания на этапе эксплуатации базы данных является необходимым условием для решения поставленных перед ней задач.

Сопровождение включает опрос пользователей с целью выяснения, какие информационные потребности остались неучтенными. В случае необходимости вносятся изменения. Обеспечивается поддержка системы путем внесения изменения и добавления новых программ и элементов данных по мере расширения и изменения потребностей бизнеса.

Основным нормативным документом, регламентирующим жизненный цикл программного обеспечения, является международный стандарт ISO/IEC 12207. Разработка отечественных программных средств ориентирована на ГОСТ ЕСПД (Единая система программной документации), ОРММ (Общеотраслевые руководящие методические материалы) по созданию автоматизированных систем управления).

Наиболее типичными моделями жизненного цикла информационных систем являются:

–каскадная модель. Последовательное выполнение всех этапов проектирования и реализации, полная определенность требований к компонентам баз данных. Любые изменения на ранних этапах приводят к повторному выполнению последующих этапов работ (принцип «от начала и до конца»);

–спиральная модель. Особый акцент делается на начальных этапах жизненного цикла: анализе и проектировании. Реализуемость технических решений проверяется путем создания прототипов баз данных или отдельных частей. На основании полученных разработок уточняются требования к базам данных, выполняется корректировка спецификаций, создается новая версия. Если результаты удовлетворительные, выполняется переход на следующий этап с параллельным завершением работ предыдущих этапов.

В настоящее время, средний жизненный цикл базы данных (СУБД) составляет 5-6 лет, из которых 1-2 года тратится на тестирование, внедрение и устранение ошибок. Учитывая, что за 5-6 лет происходит существенное обновление элементной базы компьютеров и системного программного обеспечения, более длительный жизненный цикл базы данных оказывается экономически невыгодным. Современные технологии создания автоматизированных информационных систем, использование спиральной модели, предполагают постоянное обновление программного продукта или создание новых версий, примерно через каждые 2-3 года.

2.6 Модели данных

Общая классификация моделей организации данных.

Одними из основополагающих категорий в концепции баз данных являются понятия «данные» и «модель данных». Сами данные не обладают определенной структурой, данные становятся информацией тогда, когда пользователь задает им определенную структуру, то есть осознает их смысловое содержание. Поэтому центральным понятием в области баз данных является понятие модели. Не существует однозначного определения этого термина, у разных авторов эта абстракция определяется с некоторыми различиями, но, тем не менее, можно определить это понятие следующим образом:

Модель данных – совокупность структур данных и операций их обработки.

Понятие модель данных опирается на понятие предметная область.

Предметная область – набор объектов (представляющих интерес актуальным или предполагаемым пользователям), совокупность конкретных и абстрактных понятий, между которыми существуют (фиксируются) определенные связи. Предметная область представляется множеством *фрагментов*, например, предприятие - цехами, дирекцией, бухгалтерией и т.д. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, характеризуемых различными взглядами на предметную область.

Каждая модель (база) данных содержит (отражает) информацию о некоторой предметной области.

Описание предметной области представляется в следующем виде:

- множества сущностей (реальных объектов, явлений, процессов, состояний, ситуаций);
- множества свойств (атрибутов, реквизитов) сущностей;
- множества связей (отношений) между сущностями;

Предметная область в свою очередь тесно связана с понятием:

Информационный объект - это описание какой-либо сущности в виде совокупности логически связанных атрибутов, т.е. совокупности ее свойств.

Или **информационный объект (сущность)** - это класс реальных объектов, обладающих одинаковыми характеристиками, которому присвоено уникальное имя, и который имеет множество описаний конкретных объектов ПрО, относящихся к данному классу, в виде совокупности конкретных значений атрибутов.

Например, в предметной области связанной с учебным процессом в ВУЗе, сущность (информационный объект) СТУДЕНТ может быть описана как совокупность атрибутов – Номер студента (предполагаем, что каждому студенту присвоен уникальный, т.е. неповторяющийся номер) и ФИО (студента). Понятно, что такое описание относится не к конкретному студенту, а к классу (типу) студентов, т.е.

Информационно-логическая модель предметной области – это представление предметной области в виде совокупности информационных объектов и связей между ними.

Информационно-логическая модель реализуется посредством структурирования данных на основе определенных соглашений и правил, совокупность которых определяет ту или иную модель данных.

Общая классификация моделей данных представлена на рис. 2.8.

Рис.2.8 – Общая классификация моделей данных.

В соответствии с рассмотренной выше трехуровневой архитектурой базы данных понятие модель данных следует рассматривать по отношению к каждому уровню. И действительно, физическая модель данных оперирует категориями, касающимися организации внешней памяти и структур хранения, используемых в данной операционной среде. В настоящий момент в качестве физических моделей используются различные методы размещения данных, основанные на файловых структурах: это организация файлов прямого и последовательного доступа, индексных файлов и инвертированных файлов, файлов, использующих различные методы хеширования, взаимосвязанных файлов. Кроме того, современные СУБД широко используют страничную организацию данных. Физические модели данных, основанные на страничной организации, являются наиболее перспективными.

Наибольший интерес вызывают модели данных, используемые на концептуальном уровне.

Модели концептуального уровня выражают информацию о предметной области в виде, независимом от используемой СУБД. Эти модели называются инфологическими, или семантическими, и отражают в естественной и удобной для разработчиков и других пользователей форме информационно-логический уровень абстрагирования, связанный с фиксацией и описанием объектов предметной области, их свойств и их взаимосвязей. По отношению к ним внешние модели называются подсхемами и используют те же абстрактные категории, что и концептуальные модели данных.

Инфологические модели данных используются на ранних стадиях проектирования для описания структур данных в процессе разработки приложения, а даталогические модели уже поддерживаются конкретной СУБД.

Документальные модели данных соответствуют представлению о слабоструктурированной информации, ориентированной в основном на свободные форматы документов, текстов на естественном языке.

Модели, основанные на языках разметки документов, связаны, прежде всего, со стандартными языками разметки — SGML,. HTML, XML. Эти языки определяют допустимый набор тегов (ссылок) или объектов, их атрибуты и внутреннюю структуру документа.

Тезаурусные модели основаны на принципе организации словарей, содержат определенные языковые конструкции и принципы их взаимодействия в заданной грамматике. Эти модели эффективно используются в системах-переводчиках, особенно многоязыковых переводчиках.

Дескрипторные модели — самые простые из документальных моделей, они широко использовались на ранних стадиях использования документальных баз данных. В этих моделях каждому документу соответствовал дескриптор — описатель. Этот дескриптор имел жесткую структуру и описывал документ в соответствии с теми характеристиками, которые требуются для работы с документами в разрабатываемой документальной БД.

Фактографические модели данных, наиболее широко используются в настоящее время, и соответствуют представлению информации в виде определенных структур данных (дерево, сеть, таблица). К этим моделям относятся иерархические, сетевые и реляционные модели данных,

Иерархические и сетевые модели данных исторически предшествовали реляционным, и внутренняя организация реляционных моделей во многом основана на использовании методов ранних моделей.

Иерархическая модель данных.

Иерархическая модель данных состоит из упорядоченного набора деревьев, более точно, из упорядоченного набора нескольких экземпляров одного типа дерева (рис.2.9). То есть такой структуры, в которой все элементы связаны отношениями подчиненности, и при этом любой элемент может подчиняться только одному какому-нибудь другому элементу. Такую форму зависимости удобно изображать с помощью древовидного графа (схемы, состоящей из точек и стрелок, которая связна и не имеет циклов). Тип дерева состоит из одного "корневого" типа записи (наивысший уровень) и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор одних или нескольких типов сегментов (записей), каждый из которых может иметь свой формат и свою длину.

Сегмент - это поименованная единица данных фиксированной длины, которая может содержать одно или несколько полей. Понятие "сегмент", в некотором смысле, аналогично понятию "запись" и является важной частью

логической структуры базы данных. Сегменты различаются по типу, а каждый тип характеризуется фиксированной длиной и конкретным разбиением на поля данных.

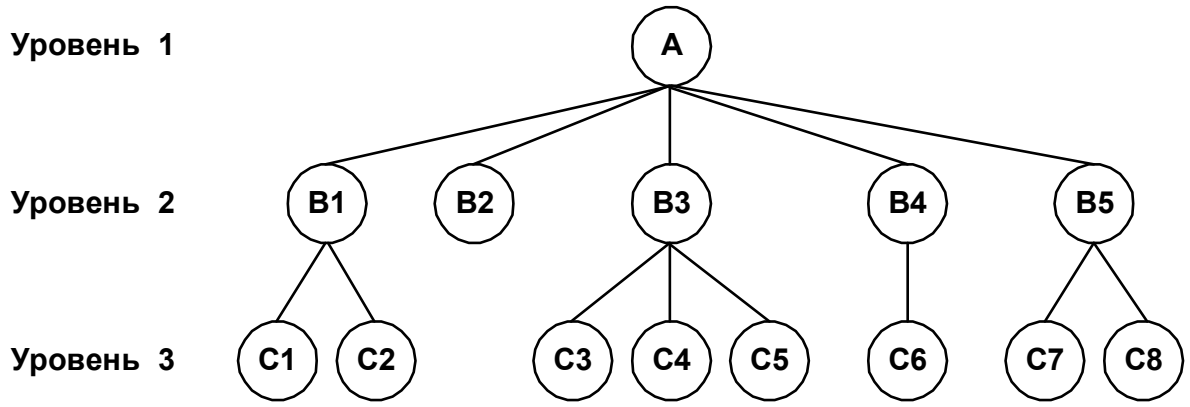


Рис. 2.9 – Иерархическое представление данных.

На схеме иерархического дерева, два связанных сегмента, расположенные на смежных уровнях (рис.2.9), называются ***исходными или предками*** для более высокого уровня (A, B1, B3,B4, B5) и ***порожденными или потомками*** для более низкого уровня (все кроме A). Соответственно каждый порожденный узел на более низком уровне связан только с одним исходным узлом, находящимся на более высоком уровне. Иерархическая запись есть система иерархически взаимосвязанных сегментов, в которой каждый порожденный сегмент представлен столько раз, сколько необходимо для полного раскрытия исходного сегмента.

Иерархическое дерево имеет только одну вершину (корень дерева) – единственный сегмент, который не зависит ни от какого другого сегмента. Зависимые узлы находятся на втором, третьем и т.д. уровнях. В корневом сегменте обычно располагается идентификатор объекта, свойства которого раскрываются в сегментах второго и более глубоких у ровней иерархии.

Все сегменты одного типа, которые порождены одним и тем же исходным сегментом, называются ***подобными или близнецами*** (например сегменты C1 и C2).

К каждому сегменты базы данных существует только один (иерархический) путь от корневого сегмента. Например, как видно из рис. 9, для сегмента C4 путь проходит через сегменты A и B3.

Иерархической базой данных является каталог папок Windows. Верхний уровень занимает папка Рабочий стол. На втором уровне находятся папки Мой компьютер, Мои документы, Сетевое окружение и Корзина, которые являются порожденными папки Рабочий стол, а между собой является подобными. В свою очередь, папка Мой компьютер является исходной по отношению к папкам третьего уровня.

В иерархической модели для обеспечения целостности автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя. В то же время ссылки между записями различных деревьев не поддерживаются.

К достоинствам иерархической модели данных относятся эффективное использование памяти ПК и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

К недостаткам иерархической модели относятся ее громоздкость для обработки информации с достаточно сложными логическими связями, сложность понимания для обычного пользователя, затруднения при выполнении операций добавления и удаления данных, а также сложность реализации отображения связи M:M.

Сетевая модель данных.

Сетевую модель представления данных можно представить как граф с записями в виде узлов и наборами в виде ребер.

Узлами сети являются отдельные экземпляры записи, которые являются единицей доступа. Сеть является более общей структурой в сравнении с иерархией, так как отдельный узел может иметь произвольное количество непосредственно старших узлов (владельцев, предков), также как и произвольное количество непосредственно подчиненных узлов (членов, потомков). Это обеспечивает прямое представление отображения связей типа M:M.

Сетевая модель состоит из набора записей и набора связей между ними, а точнее из набора экземпляров каждого типа из заданного в схеме базы данных набора типов записи и набора экземпляров каждого типа из заданного набора типов связи.

Рассмотрим сетевую модель данных об исполнителях, выполняющих определенные работы. Узлами сети являются следующие три отдельных экземпляра записей – исполнитель, количество работ и характеристика вида работы. Тогда графически сетевая модель будет иметь вид представленный на рис. 2.10.

Исполнитель

Шифр подразделения	**Название подразделения**	**ФИО** исполнителя	**Телефон**

Количество работ

Количество работ

Характеристики вида работы

Код работ	Продолжительность	Трудоемкость

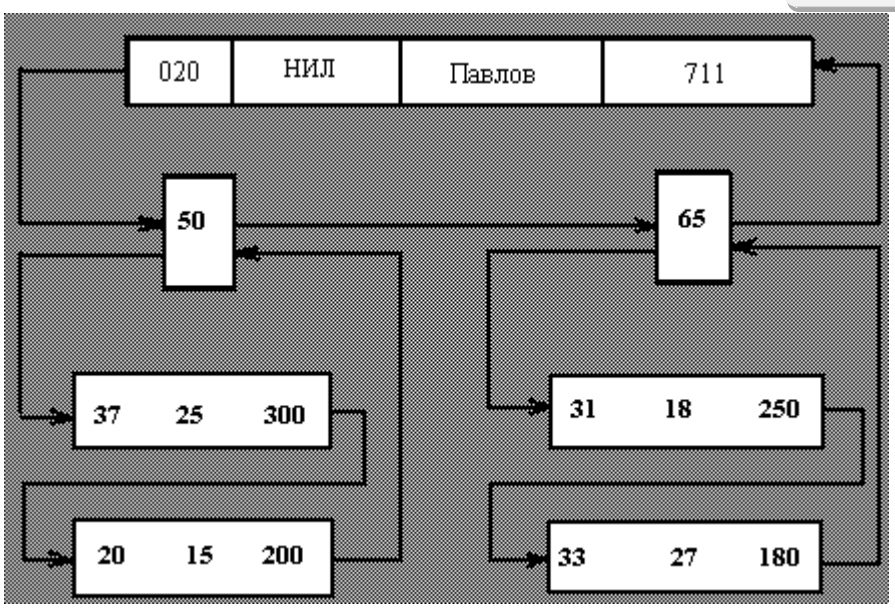


Рис.2.10 – Сетевое представление данных.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка, например исполнителей и виды работ.

Для обеспечения связи (типа M:M) между предками и потомками а в сетевой модели используется третий тип записи – ***связующая запись*** (рис.2.10). Экземпляр связующей записи представляет связь между одним исполнителем и одним видом работ и содержит данные, описывающие эту связь. В данном примере это количество работ, выполняемых исполнителем. Все экземпляры связующей записи, соответствующие одному исполнителю, помещается в цепочку, начинающуюся и возвращающуюся к этому исполнителю. Аналогичным образом устанавливаются связи для каждой отдельной работы.

В сетевых моделях, если на нее не накладывается никаких ограничений, в принципе любой файл может быть точкой входа в систему, каждый из файлов может быть связан с произвольными числами других файлов, и между записями связанных файлов могут быть любые отношения 1 : 1, 1 : M, M : M.

Каждый экземпляр записи-набора представляет иерархические связи между экземпляром записи- владельца и соответствующими экземплярами записей -членов. Это является следствием того ограничения, что ни один экземпляр записи-члена набора не может принадлежать в каждый момент времени более чем одному экземпляру набора. В данном примере такая связь существует между исполнителями и видом работ. Одним из способов организации таких связей (но не единственным) является установление цепочки указателей, выходящих из экземпляра записи-владельца, проходящих через все экземпляры записей-членов и возвращающихся обратно к экземпляру записи-владельца.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В

сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы базы данных, построенной на ее основе, а также сложность для понимания и выполнения обработки информации обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями. Основными недостатками сетевой модели данных являются сложная структура памяти, а также необходимость понижать сложность сетевой модели, а именно исключать циклические связи.

Реляционная модель данных.

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение. Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица. Подробно теоретическая основа реляционной модели данных рассматривается в разделе 2.7.

Таблица имеет строки (записи) и столбцы (колонки). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам – атрибуты отношения.

С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно: деление одного объекта (явления, сущности, системы), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование.

Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют связывание таблиц.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ПК. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Основными недостатками реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Примерами зарубежных реляционных СУБД являются следующие:

dBaseIII Plus и dBase IY (фирма Ashton-Tate), DB2 (IBM), R:BASE (Microrim), FoxBase (Fox Software), Paradox и dBASE for Windows (Borland), FoxPro, Visual FoxPro и Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) и Oracle (Oracle).

Последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционными. Примером такой системы можно считать продукты Oracle 8.x. Системы предыдущих версий вплоть до Oracle 7.x считаются «чисто» реляционными.

Постреляционная модель.

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

В постреляционной модели, по сравнению с реляционной моделью, данные хранятся более эффективно, а при обработке не требуется выполнять операцию соединения данных из двух таблиц.

Помимо обеспечения вложенности полей постреляционная модель поддерживает ассоциированные многозначные поля (множественные группы). Совокупность ассоциированных полей называется ассоциацией. При этом в строке первое значение одного столбца ассоциации соответствует первым значениям всех других столбцов ассоциации. Аналогичным образом связаны все вторые значения столбцов и т. д.

На длину полей и количество полей в записях таблицы не накладывается требование постоянства. Это означает, что структура данных и таблиц имеют большую гибкость.

Для описания функций контроля значений в полях имеется возможность создавать процедуры (коды конверсии и коды корреляции), автоматически вызываемые до или после обращения к данным. Коды корреляции выполняются сразу после чтения данных, перед их обработкой. Коды конверсии, наоборот, выполняются после обработки данных.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

К числу СУБД, основанных на постреляционной модели данных, относятся системы uniVers, Bubba и Dasdb.

Объектно-ориентированная модель.

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Стандартизованная объектно-ориентированной модель описана в рекомендациях стандарта ODMG-93 (Object Database Management Group — группа управления объектно-ориентированными базами данных). Реализовать в полном объеме рекомендации ODMG-93 пока не удается. Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, усиленные объектно-ориентированными механизмами инкапсуляции, наследования и полиморфизма. Ограниченно могут применяться операции, подобные командам SQL.

Создание и модификация БД сопровождается автоматическим формированием и последующей корректировкой индексов (индексных таблиц), содержащих информацию для быстрого поиска данных.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных связях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

В 90-е годы существовали экспериментальные прототипы объектно-ориентированных систем управления базами данных. В настоящее время такие системы получили широкое распространение, в частности, к ним относятся следующие СУБД:

POET (POET Software), Jasmine (Computer Associates), Versant (Versant Technologies), 02 (Ardent Software), ODB-Jupiter (научно-производственный центр "Интелтек Плюс"), а также Iris, Orion и Postgres.

2.7 Основы реляционных баз данных

2.7.1 Реляционная модель данных. Основные понятия

Реляционная модель данных (от английского relation – отношение) используется для представления (структурирования) данных в виде отношений – двумерных таблиц. Наименьшая единица данных реляционной модели – это отдельное атомарное (неразложимое) для данной модели значение данных.

Основные понятия и ограничения реляционной модели впервые были сформулированы сотрудником компании IBM, известного американского специалиста в области систем баз данных Е.Ф.Коддом в 1970 г. Он доказал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение.

Реляционная модель связана с тремя аспектами данных: объектами данных (структурой данных), целостностью данных и обработкой данных.

Рассмотрим наиболее важные термины, используемые в структурной части реляционной модели

Сущность – есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении (см. раздел 7).

Основной структурной частью (объектом) реляционной модели является **отношение** – это таблица содержащая некоторые данные. Отношение состоит из атрибутов и кортежей (рис.2.11).

Атрибуты отношения (поля, столбцы таблицы) имеют имя и предоставляют собой свойства данных и характеризуют сущность.

Кортежи отношения (записи, строки таблицы) – не имеют имени и содержат значения данных, заданных атрибутами.

Степень отношения – это число его атрибутов. Отношение степени один называют унарным, степени два – бинарным, степени три – тернарным, а степени n – n-арным. Степень отношения "Расписание" (рис. 2.11) – 7.

Кардинальное число или мощность отношения – это число его кортежей. Мощность отношения "Расписание" равна 4. Кардинальное число отношения изменяется во времени в отличие от его степени.

Доменом называется множество атомарных значений одного и того же типа. По сути домен – тип данных атрибута. Под атомарным значением понимается “наименьшая семантическая единица данных”, то есть это значение, не имеющее внутренней структуры при рассмотрении в реляционной модели. Домены являются общими совокупностями значений, из которых берутся конкретные значения атрибутов. То есть каждый атрибут должен быть определен на основе одного домена; это значит, что значения атрибута должны браться из этого домена. Например, домен пунктов отправления (назначения) – множество названий населенных пунктов, а домен номеров рейса – множество целых положительных чисел. Эти домены атомарны.

Значение доменов заключается в том, что домены ограничивают сравнения. То есть если два атрибута определены на одном и том домене, то их

можно сравнивать, применяя операции сравнения допустимые для данного домена. Например, атрибуты Время отправления и Время прибытия определены на одном домене Время; для этого домена допустимы операции сравнения: =, ≠, <, ≤, >, ≥. Поэтому данные атрибуты можно сравнивать, используя все указанные операции сравнения.

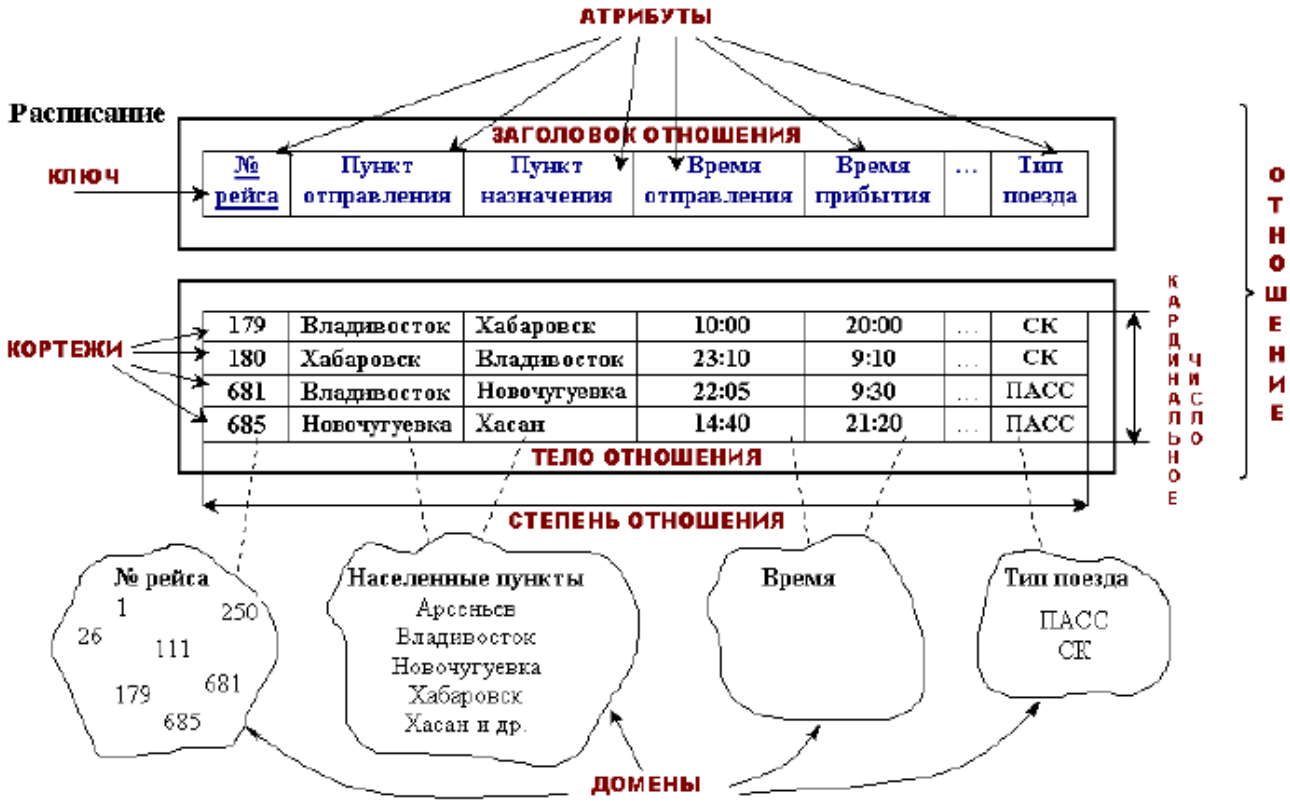


Рис.2.11 – Отношение «Расписание».

Количество доменов может быть меньше количества атрибутов, когда некоторые атрибуты определены на одном и том же домене. Так в отношении «Расписание» атрибуты Пункт отправления и Пункт назначения определены на домене Населенные пункты, а атрибуты Время отправления и Время прибытия – на домене Время. То есть атрибуты Пункты отправления и Пункт назначения (и соответственно Время отправления и Время прибытия) можно сравнивать.

Отношение представляют в виде таблицы, столбцы которой соответствуют вхождениям доменов в отношение, а строки – наборам из n значений, взятых их исходных доменов, и расположенным в соответствии с заголовком отношения.

Математически отношение можно описать следующим образом. Пусть даны n множеств D_1, D_2, D_3,..., D_n, тогда отношение R есть множество упорядоченных кортежей $(d_1, d_2, d_3, ..., d_n)$, где $d_k \in D_k$, d_k – атрибут, а D_k – домен отношения R.

Отношение содержит две части: заголовок и тело.

Схема отношения (заголовок отношения) представляет собой список имен атрибутов. Например, для отношения «Расписание» схема отношения

имеет вид № рейса, Пункт отправления, Пункт назначения, Время отправления, Время прибытия, Тип поезда.

Схема отношения содержит фиксированное множество атрибутов или, точнее, пар <имя-атрибута : имя-домена:

{<A_1:D_1>, <A_2:D_2>, …, <A_n:D_n>},

причем каждый атрибут A_j соответствует только одному из лежащих в основе доменов D_j *(j =1, 2, …, n)*. Все имена атрибутов A_1, A_2, …, A_n разные.

Схемы двух отношений называются **эквивалентными**, если они имеют одинаковую степень и возможно такое упорядочение имен атрибутов в схемах, что на одинаковых местах будут находиться сравнимые атрибуты, т.е. атрибуты, принимающие значения из одного домена.

Схема базы данных (в структурном смысле) - это набор именованных схем отношений. Тогда реляционная база данных – это набор отношений, имена которых совпадают с именами схем отношений в схеме базы данных.

Содержимым отношения (телом отношения) называют множество кортежей отношения.

Каждый кортеж, в свою очередь, содержит множество пар <имя-атрибута: значение-атрибута>:

{<A_1:v_{i1}>, <A2:v_{i2}>, …, <A_n:v_{in}>},

(i = 1, 2, …, m, где m – количество кортежей в этом множестве). В каждом таком кортеже есть одна такая пара <имя-атрибута: значение-атрибута>, то есть. <A_j:v_{ij}>, для каждого атрибута A_j в заголовке. Для любой такой пары <A_j:v_{ij}> v_{ij} является значением из уникального домена D_j, связанного с атрибутом A_j.

То есть можно сказать, что отношение – это множество кортежей, соответствующих одной схеме отношения.

Первичным ключом (ключом отношения, ключевым атрибутом) называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей. Например, в отношении «Расписание» ключевым является атрибут "№ рейса". Значение ключа всегда уникально и, как правило, не повторяется.

Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется тем, что отношение – это множество, которое не содержит одинаковых элементов – кортежей. То есть. в отношении нет повторяющихся кортежей, а это значит, что, по крайней мере, вся совокупность атрибутов обладает свойством однозначной идентификации кортежей отношения. Во многих СУБД допускается создавать отношения, не определяя ключи.

В отношении атрибут, входящий в состав первичного ключа, называется ключевым, не входящий - неключевым.

Ключ называется простым, если он состоит только из одного элемента. Последний является атомарным, а его значения - уникальными.

Составной ключ содержит два или более элементов данных, то есть состоит из нескольких атрибутов, каждый из которых необходим для

однозначной идентификации конкретного кортежа отношения. В случае реляционной модели полезно различать два варианта составного ключа.

Если кортежи идентифицируются только сцеплением значений нескольких атрибутов, то говорят, что отношение имеет составной ключ.

Отношение может содержать несколько ключей. Всегда один из ключей объявляется первичным, его значения не могут обновляться. Все остальные ключи отношения называются возможными (потенциальными или альтернативными) ключами.

Если выбранный первичный ключ состоит из минимально необходимого набора атрибутов, говорят, что он является не избыточным.

Полностью составной ключ содержит несколько атрибутов, которые не зависят друг от друга: ни один из них не является дополнительным квалификационным признаком другого атрибута.

Полусоставной ключ содержит несколько атрибутов, которые можно считать упорядоченными в том смысле, что каждый следующий атрибут в ключе является дополнительным квалификационным признаком предшествующих атрибутов.

Пусть в отношении R1 имеется не ключевой атрибут *A*, значения которого являются значениями ключевого атрибута *B* другого отношения R2. Тогда говорят, что атрибут *A* отношения R1 есть **внешний ключ.**

Чтобы связать две реляционные таблицы, необходимо ключевой атрибут первой таблицы ввести в состав ключа второй таблицы (возможно совпадение атрибутов), именно этот «связующий» атрибут является внешним ключом. Эти же атрибуты могут являться первичными ключами исходных отношений.

Ключи обычно используют для следующих целей:

– исключение дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);

– упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов, а также смешанное упорядочение (по одним — возрастание, а по другим — убывание);

–ускорения доступа к кортежами отношения;

– организации связей между отношениями.

Свойства отношений:

– в таблице не может быть двух одинаковых кортежей, записи должны отличаться хотя бы одним значением. Это свойство определяет различия отношения и таблицы, так таблица может содержать одинаковые строки, а отношение не может содержать одинаковые кортежи;

– каждый элемент данных отношения (ячейка таблицы) является атомарным (неделимым), т.е. содержит только один элемент данных;

– все элементы атрибута являются однородными по типу данных (например, или только текстовые, или только числовые);

– атрибуты имеют уникальные имена;

–атрибуты и кортежи размещаются в произвольном порядке (отсутствие упорядоченности)

–кортежи имеют фиксированное число полей (столбцов) и значений. Иначе говоря, в каждой позиции таблицы на пересечении строки и столбца всегда имеется в точности одно значение или ничего.

2.7.2 Реляционная база данных

При создании реляционной базы данных совокупность отношений позволяет хранить данные об объектах предметной области и моделировать связи между ними. Элементы реляционной модели данных и формы их представления в реляционной базе данных приведены в таблице 2.1.

Таблица 2.1 – Элементы реляционной модели

Элемент реляционной модели	**Форма представления в базе данных**
Сущность	Описание свойств объекта
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Запись (строка) таблицы
Атрибут	Поле (столбец) таблицы
Домен	Тип данных полей таблицы
Значение атрибута	Значение поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы
Степень отношения	Количество полей в таблице
Кардинальное число (мощность отношения)	Количество записей в таблице

Реляционная база данных должна отвечать следующим 12 правилами КОДДА

1. Правило информации. Вся информация в базе данных на логическом уровне представляется только значениями в таблицах, без указателей и индексов.

2. Правило гарантированного доступа. Каждый элемент таблицы (ячейка) доступен через комбинации из имени таблицы, имени поля и ключа.

3.Правило поддержки недействительных значений. Для представления отсутствующих данных с любым типом используется Null – значение.

4.Правило динамического каталога на основе реляционной модели. Метаданные формируются теми же языками, что и данные. То есть база данных должна содержать набор системных таблиц для описания самой базы данных.

5. Правило исчерпывающего подъязыка данных. В базе можно использовать несколько языков, но один (SQL) должен быть главным, который поддерживает все основные функции СУБД.

6. Правило обновления представлений (показ различным пользователям фрагментов структуры базы данных). Все теоретически обновляемые представления, должны обновляться.

7. Правило добавления, обновления и удаления. Возможность работать с отношением как с множеством и выполнять операции над множеством строк и запретить операции только над одной строкой.

8. Правило физической независимости данных.

9. Правило независимости логических данных. Правила 8 и 9 означают отделение пользователя и прикладной программы от реляционной базы данных. То есть изменение способов хранения базы данных, методов доступа, физической структуры базы данных, не должны влиять на прикладные программы для работы с данными.

10.Правило независимости целостности. База данных должна поддерживать ограничивающие условия, налагаемые на вводимые данные и действия над ними. Например, ключ не должен иметь нулевые значения.

11. Правило независимости распределения. В распределенной базе данных расположение данных независимо. То есть распределенная база данных для пользователя, выступает как централизованная.

12. Правило единственности (Правило соблюдения правил). Отсутствие возможности использовать какой-либо язык для снятия ограничений (не соблюдения правил) введенных с помощью другого языка SQL

Наиболее часто таблица с отношением размещается в отдельном файле. В некоторых СУБД одна отдельная таблица (отношение) считается базой данных. В других СУБД база данных может содержать несколько таблиц.

В общем случае можно считать, что база данных включает одну или несколько таблиц, объединенных смысловым содержанием, а также процедурами контроля целостности и обработки информации в интересах решения некоторой прикладной задачи. Например, при использовании СУБД Microsoft Access в файле базы данных наряду с таблицами хранятся и другие объекты базы: запросы, отчеты, формы, макросы и модули.

Таблица данных обычно хранится на носителях в отдельном файле операционной системы, поэтому по ее именованию могут существовать ограничения. Имена полей хранятся внутри таблиц. Правила их формирования определяются СУБД, которые, как правило, на длину полей и используемый алфавит серьезных ограничений не накладывают.

Если задаваемое таблицей отношение имеет ключ, то считается, что таблица тоже имеет ключ, и ее называют **ключевой** или таблицей с ключевыми полями.

У большинства СУБД файл таблицы включает управляющую часть (описание типов полей, имена полей и другая информация) и область размещения записей.

К отношениям можно применять систему операций, позволяющую получать одни отношения из других. Например, результатом запроса к реляционной БД может быть новое отношение, вычисленное на основе имеющихся отношений. Поэтому можно разделить обрабатываемые данные на хранимую и вычисляемую части.

Основной единицей обработки данных в реляционных базах данных является отношение, а не отдельные его кортежи (записи).

2.7.3 Типы связей в реляционной базе данных

При создании реальных баз данных информацию обычно размещают в нескольких таблицах. Таблицы при этом связаны семантикой информации. В реляционных СУБД для указания связей таблиц производят операцию их связывания.

СУБД при связывании таблиц автоматически выполняют контроль целостности вводимых в базу данных в соответствии с установленными связями. В конечном итоге это повышает достоверность хранимой в базе данных информации.

Кроме того, установление связи между таблицами облегчает доступ к данным. Связывание таблиц при выполнении таких операций как поиск, просмотр, редактирование, выборка и подготовка отчетов обычно обеспечивает возможность обращения к произвольным полям связанных записей. Это уменьшает количество явных обращений к таблицам данных и число манипуляций в каждой из них.

Основные типы связи таблиц.

Между таблицами могут устанавливаться бинарные (между двумя таблицами), тернарные (между тремя таблицами) и, в общем случае, n-арные связи. Рассмотрим наиболее часто встречающиеся бинарные связи.

При связывании двух таблиц выделяют основную и дополнительную (подчиненную) таблицы. Логическое связывание таблиц производится с помощью ключевых полей.

Ключ связи, по аналогии с обычным ключом таблицы, состоит из одного или нескольких полей, которые в данном случае называют полями связи.

Суть связывания состоит в установлении соответствия полей связи основной и дополнительной таблиц. Поля связи основной таблицы могут быть обычными и ключевыми. В качестве полей связи подчиненной таблицы чаще всего используют ключевые поля.

В зависимости от того, как определены поля связи основной и дополнительной таблиц (как соотносятся ключевые поля с полями связи),

между двумя таблицами в общем случае могут устанавливаться следующие четыре основные вида связи (таблица 2.2):

–один–к– одному (1:1);
–один –ко– многим (1:M);
–многие–к– одному (M:1);
–многие–ко– многим (M:M или M:N).

Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми, при этом одной связанной записи в таблице R соответствует только одна связанная запись в таблице S. Поскольку значения в ключевых полях обеих таблиц не повторяются, обеспечивается взаимнооднозначное соответствие записей из этих таблиц. Сами таблицы, по сути, здесь становятся равноправными.

На практике связи вида 1:1 используются сравнительно редко, так как хранимую в двух таблицах информацию легко объединить в одну таблицу, которая занимает гораздо меньше места в памяти ПК.

Связь 1:M имеет место в случае, когда одной связанной записи основной таблицы R соответствует несколько связанных записей вспомогательной таблицы S. При этом одной связанной записи в таблице S соответствует только одна связанная запись в таблице R.

Связь M:1 имеет место в случае, когда одной или нескольким связанным записям основной таблицы ставится в соответствие только одна связанная запись дополнительной таблицы.

Связь M:M возникает в случаях, когда нескольким связанным записям основной таблицы соответствует несколько связанных записей дополнительной таблицы.

Таблица 2.2 –Характеристика типов связей таблиц

Характеристика полей связи по типам	**1:1**	**1:M**	**M:1**	**M:M**
Поля связи основной таблицы	являются ключом	являются ключом	не являются ключом	не являются ключом
Поля связи дополнительной таблицы	являются ключом	не являются ключом	являются ключом	не являются ключом

Очевидно, аналогично связи 1:1, связь M:M, не устанавливает подчиненности таблиц. Для проверки этого можно основную и дополнительную таблицу поменять местами и выполнить объединение информации путем связывания. Результирующие таблицы будут отличаться только порядком следования полей, а также порядком расположения записей.

На практике в связь обычно вовлекается сразу несколько таблиц. При этом одна из таблиц может иметь различного рода связи с несколькими

таблицами. В случаях, когда связанные таблицы, в свою очередь, имеют связи с другими таблицами, образуется иерархия или дерево связей.

2.7.4 Обеспечение целостности реляционной базы данных

В отношении базы данных важно хорошо уяснить себе следующее. База данных – это не просто архив, хранилище полезной информации. База данных – главным образом информационная модель предметной области, для которой она создана. Поэтому важнейшим условием ее успешной эксплуатации является адекватность хранимых в ней данных актуальному состоянию предметной области.

Важным аспектом реляционной модели данных является поддержка целостности (см. 12 правил Кодда).

Целостность данных – правильность данных в любой момент времени при манипулировании данными. Поддержание целостности базы данных может рассматриваться как защита данных от неверных изменений или разрушений.

Целостность базы данных – это синоним её системности. Это значит, что структура базы данных определяется характером связей между ними. Связи соответствуют зависимостям между компонентами предметной области. Всякие зависимости представляют собой ограничения на возможные отношения элементов системы. В теории баз данных эти ограничения называются ограничениями целостности данных.

Поддержка целостности включает:

–структурную целостность;

–языковую целостность;

– семантическую целостность;

– ссылочную целостность;

Структурная целостность.

Структурная целостность подразумевает, ограничения целостности отношений (таблиц), то есть реляционная СУБД может работать только с реляционными отношениями. А реляционное отношение, в свою очередь, должно удовлетворять ограничениям, накладываемым на него в классической теории реляционных баз данных (отсутствие одинаковых кортежей и, следовательно, наличие первичного ключа, отсутствие упорядоченности атрибутов и кортежей).

Требование структурной целостности осуществляется с помощью двух ограничений:

–при добавлении кортежей в отношение проверяется уникальность их первичных ключей;

–не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.

Здесь возникает необходимость рассмотреть проблему неопределенных значений (Null-значений). Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени.

При сравнении неопределенных значений: одно Null-значение никогда не считается равным другому Null-значению (не действуют стандартные правила сравнения).

Для выявления равенства значения некоторого атрибута неопределенному атрибуту применяют стандартные предикаты:

–<Имя атрибута> Is Null (Null-значение для заданного атрибута может быть разрешено или запрещено).

–<Имя атрибута> Is Not Null (Значение Not Null (обязательное поле) является ограничением для ключевых полей и означает, что ни при каких условиях ключевые атрибуты не могут принимать неопределенные значения).

Таблица 2.3 содержит пример проверки атрибута *Адрес* на неопределенное значение.

Таблица 2.3 – Проверка атрибута *Адрес* на неопределенное значение

Адрес	**Адрес Is Null**	**Адрес Is Not Null**
Null	True	False
ул.Герцена, 3	False	True

Ограничения целостности отношений задаются при спецификации структуры каждой таблицы при определении ее ключевых или индексированных полей. Ограничения целостности отношений также являются безотлагательными, они проверяются всегда и притом немедленно. Любая попытка пропуска значений или ввода дублирующих значений ключевых полей пресекается СУБД немедленно.

Языковая целостность.

Языковая целостность состоит в том, что реляционная СУБД должна обеспечивать языки описания и манипулирования данными не ниже стандарта SQL. Не должны быть доступны иные низкоуровневые средства манипулирования данными, не соответствующие стандарту.

Семантические ограничения целостности определяют ограничения целостности атрибутов (доменов) – определение множества элементов данных – домена, на котором определен атрибут. Ограничения такого рода задаются при определении структуры таблицы в виде свойств образующих ее полей, таких как тип данных, размер поля, маска ввода, значение по умолчанию, условие на значение и др.;

Например, понятие «оценка» предполагает, что это данное «числового» и никакого другого типа со значениями в диапазоне от 2 до 5 или что «фамилия» – это строка символов определенной длины, содержащая фамилию и инициалы и т. д. Ограничение целостности атрибутов проверяется СУБД всегда и притом немедленно. Любая попытка введения некорректного значения атрибута будет отвергнута.

В случае, когда база данных моделируется не одним, а несколькими (хотя бы и двумя) связанными отношениями, возникает проблема ограничения целостности базы данных. Ограничение целостности базы данных в реляционном представлении определяется как ограничение ссылочной

целостности, или ограничение, основанное на связях между отношениями-объектами. Связи между отношениями могут быть типа «один к одному» (1:1) или «один ко многим» (1:N).

Ссылочная целостность.

При установлении связи между отношениями возникает необходимость поддержания целостности по ссылкам. Отношение со стороны «один» будем называть – основным отношением, а отношение со стороны «многие» – подчиненным.

Требование ссылочной целостности состоит в следующем: для каждого значения внешнего ключа, появляющегося в подчиненном отношении, в основном отношении должен существовать кортеж с таким же значением первичного ключа.

У первичного и внешнего ключей, образующих связь, должен быть одинаковый тип данных.

То есть значение внешнего ключа должно либо:

–быть равным значению первичного ключа;

–быть полностью неопределенным, т.е. каждое значение поля, участвующего во внешнем ключе должно быть неопределенным.

Ссылочная целостность проявляется в том, что они налагают ограничения на выполнение корректирующих операций (добавление, удаление и обновление), выполняемых над строками-кортежами связанных таблиц. Корректирующие операции должны выполняться таким образом, чтобы связи между таблицами всегда оставались согласованными и соответствовали семантике связей между объектами предметной области.

Для каждого внешнего ключа в процессе проектирования необходимо решить три вопроса:

1. Может ли данный внешний ключ принимать неопределенные значения?

2. Что произойдет при попытке УДАЛЕНИЯ записи из основного отношения, на которую ссылается внешний ключ подчиненного отношения?

Например, удалить поставщика, для которого имеется, по крайней мере, одна поставка.

В общем случае существует три ситуации:

Каскадирование удаления, при котором удаляются все записи из подчиненного отношения, соответствующие удаляемому первичному ключу основного отношения (будет удален поставщик и все его поставки).

Ограничение удаления, при котором удаляется запись из основного отношения только в том случае, если в подчиненном отношении нет соответствующих значений внешнего ключа, иначе удаление отменяется (удаление поставщика невозможно, пока существует хотя бы одна его поставка).

Установка неопределенных значений, при которой внешний ключ подчиненного отношения устанавливается в неопределенное значение (Null-

значение), а соответствующая запись из основного отношения удаляется (все значения внешнего ключа в поставках принимают Null-значение, а поставщик удаляется).

Данное свойство поддерживается не всеми СУБД. Если необходимо применить эту ситуацию, то в подчиненном отношении сначала нужно удалить все значения внешнего ключа соответствующие первичному, и только после этого удалять запись из основного отношения с соответствующим первичным ключом

3. Что произойдет при попытке ОБНОВЛЕНИЯ первичного ключа основного отношения, на который ссылается некоторый внешний ключ подчиненного отношения? Например, при попытке обновления кода поставщика, для которого имеется хотя бы одна поставка.

Здесь также возможны три ситуации:

Каскадирование обновления, при котором при обновлении первичного ключа обновляются все соответствующие внешние ключи (будет обновлен код поставщика в основном отношении и все соответствующие ему внешние ключи в поставках).

Ограничение обновления, при котором обновляется первичный ключ в основном отношении только в том случае, если в подчиненном отношении нет соответствующих значений внешнего ключа, иначе обновление отменяется (обновление кода поставщика невозможно, пока существует хотя бы одна поставка этого поставщика).

Установка неопределенных значений, при которой внешний ключ подчиненного отношения устанавливается в неопределенное значение, а соответствующий первичный ключ в основном отношении обновляется (все значения внешнего ключа в поставках принимают Null-значение, а код поставщика в основном отношении обновляется).

2.8 Базовые понятия реляционной алгебры

Для получения информации из отношений необходим язык манипулирования данными (ЯМД), способный выполнять соответствующие операции над отношениями. Наиболее важной частью ЯМД является его раздел для формулировки запросов. Поскольку запросы в общем случае представляют собой произвольные функции над отношениями, необходимо решить вопрос о требуемой выразительности языка запросов. Для исследования этого вопроса были разработаны три типа теоретических языков: реляционная алгебра, реляционное исчисление с переменными-кортежами и реляционное исчисление с переменными-доменами.

Языки запросов первого типа – алгебраические языки – основаны на классической теории множеств (с некоторыми уточнениями) и позволяют выражать запросы средствами специализированных операторов, применяемых к отношениям. Языки запросов второго и третьего типов – языки исчисления –

основаны на классическом логическом аппарате исчисления предикатов первого порядка и позволяют выражать свойства желаемого результата, фактически не указывая, как его получить. Оба механизма (алгебра и исчисление) обладают важным свойством: они замкнуты относительно понятия отношения. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных баз данных и результатом вычисления также являются отношения. В результате любое выражение или формула могут пониматься как отношения. Языки этих трех типов были предложены Е.Ф. Коддом и служат теоретической основой для описания действий, выполняемых над отношениями. Определенные в них элементарные операции реализуются в любом реальном языке запросов независимо от его внешнего оформления.

Реальные языки (SQL, QBE и др.) обеспечивают не только функции соответствующего теоретического языка или их комбинации, но и реализуют некоторые дополнительные операции - арифметические операции, команды присваивания и печати и т.п. Конкретный язык манипулирования реляционными БД называется реляционно полным, если любой запрос, выраженный с помощью одного выражения реляционной алгебры или одной формулы реляционного исчисления, может быть выражен с помощью одного оператора языка. При этом для любого допустимого выражения реляционной алгебры можно построить эквивалентную формулу реляционного исчисления, и наоборот, то есть эти механизма эквивалентны.

2.8.1 Операции над отношениями

В основе реляционной алгебры лежит идея о том, что, так как отношение – это множество кортежей, то и средства манипулирования отношениями должны быть такими же, как традиционные теоретико–множественные операции, дополненные специфическими для баз данных операциями.

Расширенный начальный вариант алгебры, определенный Коддом, состоит из восьми алгебраических операций, которые делятся на два класса – теоретико–множественные операции и специальные реляционные операции.

– объединение – возвращает отношение, содержащее все кортежи, принадлежащие или одному из двух определенных отношений, или обоим;

– пересечение – возвращает отношение, содержащее все кортежи, принадлежащие одновременно двум определенным отношениям;

– вычитание – возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух определенных отношений и не принадлежат второму;

– декартово произведение – возвращает отношение, содержащее всевозможные кортежи, являющиеся сочетанием двух кортежей, принадлежащих соответственно двум определенным отношениям;

К специальным операциям относятся:

– выборка (селекция, ограничение) – возвращает отношение, содержащее все кортежи из определенного отношения, удовлетворяющие определенным условиям;

– проекция – возвращает отношение, содержащее все кортежи (называемые как подкортежи) определенного отношения после исключения из него некоторых атрибутов;

– деление – для двух отношений, бинарного и унарного, возвращает отношение, содержащее все значения одного атрибута бинарного отношения, соответствующее (в другом атрибуте) всем значениям в унарном отношении.

– соединение (естественное) – возвращает отношение, кортежи которого – это сочетание двух кортежей (принадлежащих соответственно двум определенным отношениям), имеющих общее значение для одного или нескольких атрибутов этих двух отношений (и такие общие значения в результирующем кортеже появляются только один раз);

Для алгебры, у которой операции замкнуты относительно понятия отношения, каждая операция должна производить отношение с двумя составляющими: телом и заголовком. Только в этом случае будет действительно возможно строить вложенные выражения. Операции объединения, пересечения взятия разности (вычитания) требуют от отношений-операндов совместимых по типу. Два отношения называются ***совместимыми по типу***, если каждое из них имеет одинаковое множество имен атрибутов и если соответствующие атрибуты определены на одном и том же домене. Если необходимо выполнить операцию объединения, пересечения или вычитания двух отношений, которые почти совместимы по типу, за исключением некоторых различий в именах атрибутов, можно использовать оператор переименования, чтобы сделать эти отношения полностью совместимыми по типу, прежде чем выполнить необходимую операцию.

Операции реляционной алгебры основаны на элементах теории множеств.

Множества обычно обозначаются заглавными латинскими буквами. Если элемент x принадлежит множеству A, то это обозначается:

$x \in A$.

Если каждый элемент множества B является также и элементом множества A, то говорят, что множество B является подмножеством множества A:

$B \subset A$.

Подмножество B множества A называется собственным подмножеством, если

$B \neq A$.

Теоретико-множественные операции реляционной алгебры.

Объединением двух совместимых по типу отношений A и B называется отношение C с тем же заголовком, как в исходных отношениях, и с телом,

состоящим из множества всех кортежей, принадлежащих *A* или *B* или обоим отношением (за исключением повторяющихся) (рис.2.12 а).

Синтаксис операции объединения:

Пусть заданы два отношения *A={a}, B={b}*, где *a* и *b* – соответственно кортежи отношений *A* и *B*, то объединение

$$C=A \cup B = \{c \mid c \in A \vee c \in B\},$$ или ***C=A UNION B,***

где с – кортеж нового отношения *C*, $\vee$ – операция логического сложения «ИЛИ» (выражение читается – для кортежей с верно, что кортежи с принадлежат отношению *A* или отношению *B* .

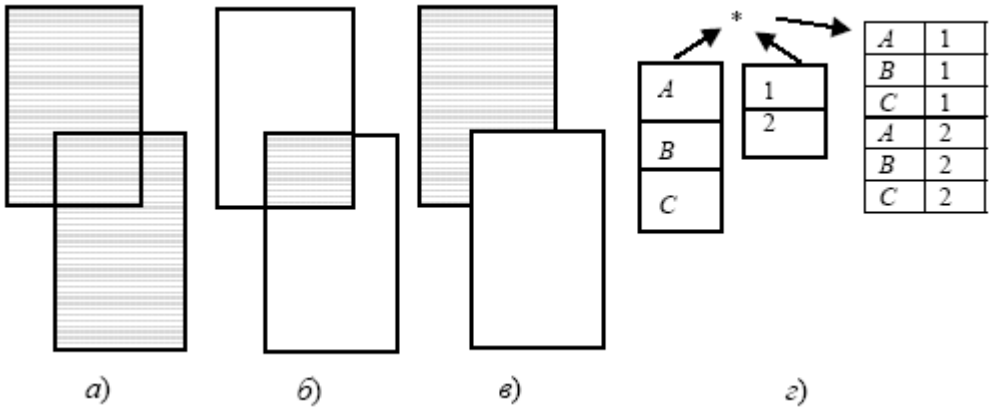


Рис. 2.12 – Операции над отношениями а) Объединение, б) Пересечение, в) Вычитание, г) Декартово произведение.

Объединение, как и любое отношение, не может содержать одинаковых кортежей. Поэтому, если некоторый кортеж входит и в отношение *A*, и отношение *B*, то в объединение он входит один раз.

Пример. Пусть даны два отношения *A* и *B* с информацией о сотрудниках:

Отношение *A*.

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Отношение *B*.

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

Объединение отношений *A* и *B* будет иметь вид:

Отношение *C=A UNION B*.

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000

3	Сидоров	3000
2	Пушников	2500
4	Сидоров	3000

Как видно из приведенного примера, потенциальные ключи, которые были в отношениях *A* и *B* не наследуются объединением этих отношений. Поэтому, в объединении отношений *A* и *B* атрибут "Табельный номер" может содержать дубликаты значений. Если бы это было не так, и ключи наследовались бы, то это противоречило бы понятию объединения как "объединение множеств". Конечно, объединение отношений *A* и *B* имеет, как и любое отношение, потенциальный ключ, например, состоящий из всех атрибутов.

Пересечением двух совместимых по типу отношений *A* и *B* называется отношение *C* с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям *A* и *B*.

Синтаксис операции пересечения:

$$C=A \cap B = \{c \mid c \in A \wedge c \in B\},$$ или ***C=A INTERSECT B,***

где $\wedge$– операция логического умножения (логическое «И»).

Пример. Для тех же отношений *A* и *B*, что и в предыдущем примере пересечение имеет вид:

Отношение *C=A INTERSECT B.*

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000

Для этой операции, в отличие от операции объединения, потенциальные ключи могли бы наследоваться пересечением отношений. Однако это не так. Вообще, ***никакие реляционные операторы не передают результатирующему отношению никаких данных о потенциальных ключах***. В качестве причины этого можно было бы привести тривиальное соображение, что так получается более просто и симметрично - все операторы устроены одинаково. На самом деле причина более глубока, и заключается в том, что потенциальный ключ - семантическое понятие, отражающее различимость объектов предметной области. Наличие потенциальных ключей не выводится из структуры отношения, а явно задается для каждого отношения, исходя из его смысла. Реляционные же операторы являются формальными операциями над отношениями и выполняются одинаково, независимо от смысла данных, содержащихся в отношениях. Поэтому, реляционные операторы ничего не могут "знать" о смысле данных. Трактовка результата реляционных операций - дело пользователя.

Вычитанием двух совместимых по типу отношений *A* и *B* называется отношение *C* с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из кортежей, принадлежащих отношению *A* и не принадлежащих отношению *B*.

Синтаксис операции вычитания:

$$C=A-B = \{c \mid c \in A \wedge c \notin B\},$$ или ***C=A MINUS B,***

Пример. Для тех же отношений *A* и *B*, что и в предыдущем примере вычитание имеет вид:

Отношение *C=A MINUS B.*

Табельный номер	Фамилия	Зарплата
2	Петров	2000
3	Сидоров	3000

Декартово произведение.

Прежде чем определить саму операцию, введем дополнительно понятие конкатенации, или сцепления, кортежей.

Сцеплением, или конкатенацией, кортежей $a = (a_1,a_2,\ldots,a_n)$ и $b = (b_1,b_2,\ldots,b_m)$ называется кортеж, полученный добавлением значений второго в конец первого. Сцепление кортежей *a* и *b* обозначается как (a,b).

$$(a,b) = (a_1,a_2,\ldots,a_n, b_1,b_2,\ldots,b_m),$$

где n – число элементов в первом кортеже a, m – число элементов во втором кортеже b.

Все предыдущие операция не меняли степени или арности отношений – это следует из определения эквивалентности схем отношений. Операция расширенного декартова произведения меняет степень результирующего отношения.

Декартовым произведением двух отношений $A(A_1, A_2,\ldots A_n)$ и $B (B_1, B_2,\ldots B_m)$ называется отношение *C*, заголовок которого является сцеплением заголовков отношений *A* и *B*:

$(A_1, A_2,\ldots A_n, B_1, B_2,\ldots B_m)$,

а тело состоит из кортежей, являющихся **сцеплением кортежей** отношений *A* и *B*:

$(a_1,a_2,\ldots,a_n, b_1,b_2,\ldots,b_m)$,

таких, что $(a_1,a_2,\ldots,a_n) \in A$, $(b_1,b_2,\ldots,b_m) \in B$, (рис.12 г.)

Синтаксис операции декартового произведения:

То есть. если $A=\{a\}$, $B=\{b\}$, то расширенное декартово произведение

$$C=A \otimes B = \{(a,b) \mid a \in A \wedge b \in B\},$$ или ***C=A TIMES B***

Мощность произведения *A TIMES B* равна произведению мощностей отношений *A* и *B*, т.к. каждый кортеж отношения *A* соединяется с каждым кортежем отношения *B*.

Если в отношения *A* и *B* имеются атрибуты с одинаковыми наименованиями, то перед выполнением операции декартового произведения такие атрибуты необходимо переименовать.

Перемножать можно любые два отношения, совместимость по типу при этом не требуется.

Пример. Пусть даны два отношения *A* и *B* с информацией о поставщиках и деталях:

Отношение *A* (Поставщики).

Номер поставщика	**Наименование поставщика**
1	Иванов
2	Петров
3	Сидоров

Отношение *B* (Детали).

Номер детали	**Наименование детали**
1	Болт
2	Гайка
3	Винт

Декартово произведение отношений *A* и *B* будет иметь вид:

Отношение *C=A TIMES B.*

Номер поставщика	**Наименование поставщика**	**Номер детали**	**Наименование детали**
1	Иванов	1	Болт
1	Иванов	2	Гайка
1	Иванов	3	Винт
2	Петров	1	Болт
2	Петров	2	Гайка
2	Петров	3	Винт
3	Сидоров	1	Болт
3	Сидоров	2	Гайка
3	Сидоров	3	Винт

Специальные реляционные операторы.

***Выборка*.**

Для определения этой операции необходимо ввести дополнительные обозначения.

Пусть α – булевское выражение, составленное из термов сравнения с помощью связок AND, OR, NOT и круглых скобок. В качестве термов сравнения допускаются:

1. терм $X\,\theta\,x$,

где X – имя некоторого атрибута, принимающего значения из домена D; x – константа, взятая из того же домена D, $X \in D$; θ – одна из допустимых для данного домена D операций сравнения (=, ≠, <, ≤, >, ≥);

2. терм $X \theta Y$,

где X, Y – имена некоторых θ-сравнимых атрибутов, то есть атрибутов, принимающих значение из одного и того же домена D.

Тогда **выборкой (θ-выборкой)** из отношения A по атрибутам X и Y называется отношение S, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество всех кортежей t отношения A, для которых для которых истинно условие выбора (проверка условия *"X θ Y"* дает значение истина) или ограничения.

Синтаксис операции выборка:

S=A[α(t)] = {t | t ∈ A ∧ α(t) = "Истина"}, или $S=\sigma_{X\,\theta\,Y}(A)$ или ***S=A WHERE XθY***

Атрибуты X и Y должны быть определены на одном и том же домене, а оператор должен иметь смысл для этого домена.

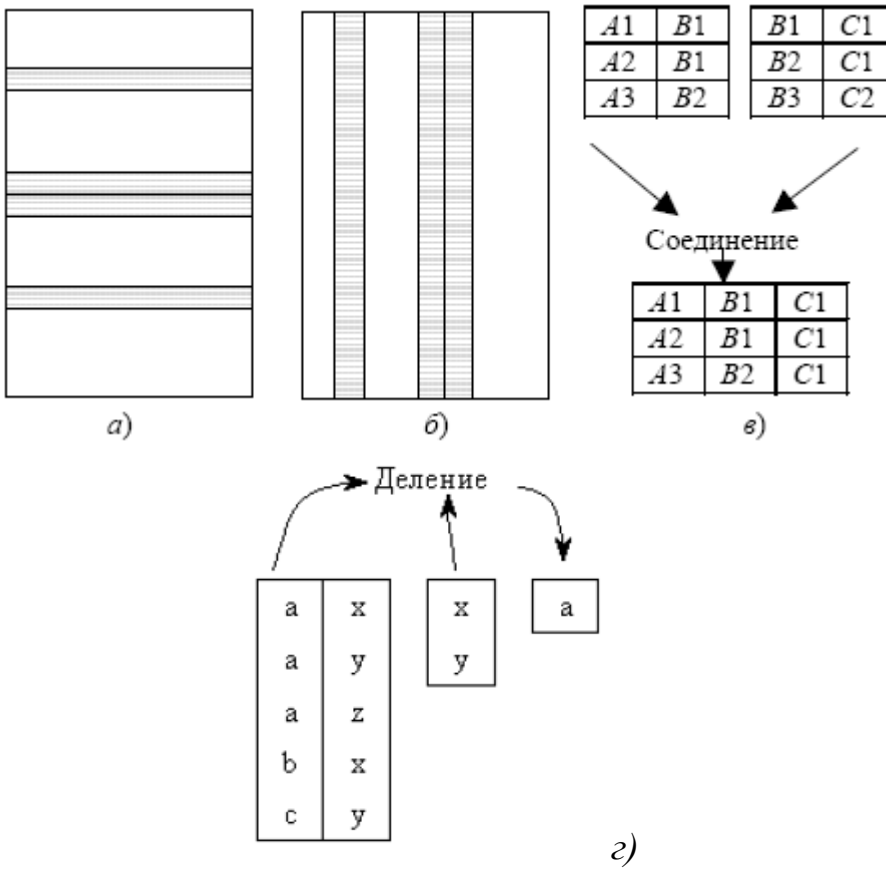


Рис. 2.13 – Операции над отношениями а) выборка, б) проекция, в) соединение, г) деление.

На основании свойства замкнутости можно однозначно расширить определение до формы, в которой условие в выражении *where* будет содержать произвольное число логических сочетаний таких простых сравнений (с использованием логических связок AND, OR, NOT и круглых скобок).

Пример. Пусть дано отношение A с информацией о сотрудниках:

Отношение *A*.

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Результат выборки $\sigma_{\text{зарплата}<3000}(A)$ или *A WHERE Зарплата<3000* будет иметь вид:

Отношение *S=A WHERE Зарплата<3000*.

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000

Смысл операции выборки очевиден - выбрать кортежи отношения, удовлетворяющие некоторому условию. Таким образом, операция выборки дает "горизонтальный срез" отношения по некоторому условию.

Проекцией отношения *A* по атрибутам *X, Y, ..., Z*, где каждый из атрибутов принадлежит отношению *A*, называется отношение *S* с заголовком *{X, Y, ..., Z}* и телом, содержащим множество кортежей вида {*x, y, ..., z}*, таких, для которых в отношении *A* найдутся кортежи со значением атрибута *X* равным *x*, атрибута *Y* равным *y*, …, атрибута *Z* равным *z*

Синтаксис операции проекции:

S=A[X,Y,Z] или ***S=πx, y, …, z (A)***

Операция проекции дает "вертикальный срез" отношения, в котором удалены все возникшие при таком срезе дубликаты кортежей. То есть с помощью оператора проекции получено «вертикальное» подмножество данного отношения, т.е. подмножество, получаемое исключением всех атрибутов, не указанных в списке атрибутов, и последующим исключением дублирующих кортежей (подкортежей) из того, что осталось.

Никакой атрибут не может быть указан в списке атрибутов более одного раза.

Пример. Пусть дано отношение *A* с информацией о поставщиках, включающих наименование и месторасположение:

Отношение *A* (Поставщики).

Номер поставщика	Наименование поставщика	Город поставщика
1	Иванов	Уфа
2	Петров	Москва
3	Сидоров	Москва
4	Сидоров	Челябинск

Результат Проекции $r=\pi_{Город\ поставщика}\ (A)$ или *A[Город поставщика]* будет иметь вид:

Отношение *A[Город поставщика]*.

Город поставщика
Уфа
Москва
Челябинск

Деление.

Пусть отношения *A* и *B* имеют заголовки:

$\{X_1, X_2, \ldots, Xm, Y_1, Y_2, \ldots, Y_n\}$ и $\{ Y_1, Y_2, \ldots, Y_n \}$

соответственно, то есть атрибуты Y_j $(j=1, \ldots, n)$ - общие для двух отношений, и отношение *A* имеет дополнительные атрибуты X_i $(i=1, \ldots, m)$, а отношение *B* не имеет дополнительных атрибутов (отношения *A* и *B* представляют соответственно делимое и делитель). Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Пусть теперь выражения $\{X_i\}$ и $\{Y_j\}$ обозначают два составных атрибута *X* и *Y* соответственно.

Тогда **делением** отношений *A* на *B* (*A* divideby *B*) называется отношение S с заголовком $\{X\}$ и телом, содержащим множество всех кортежей $\{X:x\}$, таких что существует кортеж $\{X:x,Y:y\}$, который принадлежит отношению *A* для всех кортежей $\{Y:y\}$, принадлежащих отношению *B*. Иначе это можно сформулировать так: результат содержит такие *X*-значения из отношения *A*, для которых соответствующие *Y*-значения из *A* включают все *Y*-значения из отношения *B*.

Деление отношений аналогично делению чисел с остатком (рис. 13 г).

Синтаксис операции деления:

S=A÷B или ***S=A divideby B***

Типичные запросы, реализуемые с помощью операции деления, обычно в своей формулировке имеют слово "все" - "какие поставщики поставляют все детали?".

Пример. В примере с поставщиками, деталями и поставками ответим на вопрос, "какие поставщики поставляют все детали?".

В качестве делимого возьмем проекцию *X=PD[PNUM,DNUM]*, содержащую номера поставщиков и номера поставляемых ими деталей:

Проекция *X=PD[PNUM,DNUM]*.

Номер поставщика PNUM	**Номер детали DNUM**
1	1
1	2
1	3
2	1
2	2
3	1

В качестве делителя возьмем проекцию *Y=D[DNUM]*, содержащую список номеров всех деталей (не обязательно поставляемых кем-либо):

Проекция *Y=D[DNUM]*; Отношение *S=X DEVIDEBY*.

Номер детали DNUM
1
2
3

Номер поставщика PNUM
1

Деление *A divideby B* дает список номеров поставщиков, поставляющих все детали. Так в результате деления оказалось, что только поставщик с номером 1 поставляет все детали.

Соединение.

Операция соединения отношений, наряду с операциями выборки и проекции, является одной из наиболее важных реляционных операций.

Обычно рассматривается несколько разновидностей операции соединения:

–общая операция соединения

–θ –соединение (тэта-соединение)

–экви-соединение

–естественное соединение

Наиболее важным из этих частных случаев является операция естественного соединения. Все разновидности соединения являются частными случаями общей операции соединения.

Общая операция соединения.

Соединением отношений *A* и *B* по условию *m* называется отношение S

$$S = \sigma_{X\ m\ Y} \text{ или } S = A \text{ TIMES } B \text{ WHERE } X\ m\ Y,$$

где *m* представляет собой логическое выражение, в которое могут входить атрибуты отношений *A* и *B* и (или) скалярные выражения.

Таким образом, операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях *A* и *B* имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

Тэта-соединение (частный случай операции общего соединения).

Эта операция предназначается для тех случаев, когда нам нужно соединить вместе два отношения на основе некоторых условий. Пусть отношения *A* и *B* не имеют общих имен атрибутов и θ определяется как одна из допустимых для данного домена *D* операций сравнения ($=, \neq, <, \leq, >, \geq$). Тогда θ **-соединением** отношения *A* по атрибуту *X* с отношением *B* по атрибуту *Y* (*A* join *B* on *X* θ *Y*) называется отношение *S* с тем же заголовком, что и при декартовом произведении отношений *A* и *B*, и с телом, содержащим множество кортежей *t*, таких, что *t* принадлежит этому декартову произведению и вычисление условия "*X* θ *Y*" дает значение истина для этого кортежа. Атрибуты

X и *Y* должны быть определены на одном и том же домене, а операция должна иметь смысл для этого домена.

Синтаксис операции θ -соединение:

S=A[X θ Y]B=S= $\sigma_{X\theta Y}$ или ***S=A TIMES B WHERE X θ Y*** или $S = A \underset{X\,\theta\,Y}{\triangleright\!\triangleleft} B$

или ***S= A JOIN B ON (X θ Y).***

Пример. Рассмотрим некоторую компанию, в которой хранятся данные о поставщиках и поставляемых деталях. Пусть поставщикам и деталям присвоен некий статус. Пусть бизнес компании организован таким образом, что поставщики имеют право поставлять только те детали, статус которых не выше статуса поставщика (смысл этого может быть в том, что хороший поставщик с высоким статусом может поставлять больше разновидностей деталей, а плохой поставщик с низким статусом может поставлять только ограниченный список деталей, важность которых (статус детали) не очень высока).

Отношение *A (Поставщики)*

Наименование поставщика	**X (Статус поставщика)**
Иванов	4
Петров	1
Сидоров	2

Отношение *B (Детали)*

Наименование детали	**Y (Статус детали)**
Болт	3
Гайка	2
Винт	1

Ответ на вопрос "какие поставщики имеют право поставлять какие детали?" дает θ -соединение $S=A[X \geq Y]B$:

Отношение "*Какие поставщики, поставляют, какие детали*"

Наименование поставщика	**X (Статус поставщика)**	**Наименование детали**	**Y (Статус детали)**
Иванов	4	Болт	3
Иванов	4	Гайка	2
Иванов	4	Винт	1
Петров	1	Винт	1
Сидоров	2	Гайка	2
Сидоров	2	Винт	1

Экви-соединение.

Наиболее важным частным случаем θ -соединения является случай, когда θ -есть просто равенство.

Если θ обозначает "равно", то θ -соединение называется экви-соединением. Из определения следует, что результат экви-соединения должен включать два таких атрибута, значения которых должны быть равны в каждом кортеже отношения. Если один из этих атрибутов исключается (что может быть сделано операцией проекции), результатом будет просто естественное соединение.

Синтаксис операции экви-соединения:

S=A[X = Y]B=S= $\sigma_{X=Y}$ или ***S=A TIMES B WHERE X = Y*** или $S = A \underset{X=Y}{\triangleright\triangleleft} B$

или ***S= A* JOIN *B* ON (*X* = *Y*)**

Пример. Пусть имеются отношения *P*, *D* и *PD*, хранящие информацию о поставщиках, деталях и поставках соответственно (для удобства введем краткие наименования атрибутов):

Отношение *P (Поставщики)*

Номер поставщика PNUM	**Наименование поставщика PNAME**
1	Иванов
2	Петров
3	Сидоров

Отношение *D (Детали)*

Номер детали DNUM	**Наименование детали DNAME**
1	Болт
2	Гайка
3	Винт

Отношение *PD (Поставки)*

Номер поставщика PNUM	**Номер детали DNUM**	**Поставляемое количество VOLUME**
1	1	100
1	2	200
1	3	300
2	1	150
2	2	250
3	1	1000

Ответ на вопрос, какие детали поставляются поставщиками, дает экви-соединение *P[PNUM=PNUM]PD*. На самом деле, т.к. в отношениях имеются одинаковые атрибуты, то требуется сначала переименовать атрибуты, а потом выполнить экви-соединение. Запись становится более громоздкой:

(P RENAME PNUM AS PNUM1)[PNUM1= PNUM2](PD RENAME PNUM AS PNUM2)

Обычно, такой сложной формой записи не пользуются. Результат операции:

Отношение "*Какие детали поставляются какими поставщиками*"

Номер поставщика PNUM1	**Наименование поставщика PNAME**	**Номер поставщика PNUM2**	**Номер детали DNUM**	**Поставляемое количество VOLUME**
1	Иванов	1	1	100
1	Иванов	1	2	200
1	Иванов	1	3	300
2	Петров	2	1	150
2	Петров	2	2	250
3	Сидоров	3	1	1000

Недостатком экви-соединения является то, что если соединение происходит по атрибутам с одинаковыми наименованиями (а так чаще всего и происходит!), то в результатирующем отношении появляется два атрибута с одинаковыми значениями. В нашем примере атрибуты *PNUM1* и *PNUM2* содержат дублирующие данные. Избавиться от этого недостатка можно, взяв проекцию по всем атрибутам, кроме одного из дублирующих. Именно так действует естественное соединение.

Естественное соединение.

Пусть отношения *A* и *B* имеют заголовки

$\{X_1, X_2, \ldots, X_m, Y_1, Y_2, \ldots, Y_n\}$ и $\{Y_1, Y_2, \ldots, Y_n, Z_1, Z_2, \ldots, Z_p\}$

соответственно; т.е. атрибуты Y_j (j=1,…, n) (и только они) – общие для двух отношений (т.е. атрибуты с одинаковыми именами и определенные на одинаковых доменах); X_i (i=1,..., m) – остальные атрибуты отношения *A*; Z_k (k=1,..., p) – остальные атрибуты отношения *B*. Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Теперь рассмотрим выражения $\{X_i\}$, $\{Y_j\}$ и $\{Z_k\}$ как три составных атрибута *X*, *Y* и *Z* соответственно. Тогда **естественным соединением** отношений *A* и *B* (*A JOIN B*) называется отношение *S* с заголовком {*X*, *Y*, *Z*} и телом, содержащим множество всех кортежей {*x*, *y*, *z*}, таких, для которых в отношении *A* значение атрибута *X* равно *x*, а атрибута *Y* равно *y*, и в отношении *B* значение атрибута *Y* равно *y*, а атрибута *Z* равно *z* (рис.2.13 в).

Синтаксис операции естественного соединения:

S*=πx, y, …, z (*Y1, Y2, …, Yn*)(*A*⊗*B*)** или $S = A \triangleright\!\triangleleft B$ или ***S= A JOIN B

Если отношения *A* и *B* не имеют общих имен атрибутов, то выражение *A JOIN B* эквивалентно *A TIMES B* (т.е. естественное соединение превращается в этом случае в декартово произведение).

В синтаксисе естественного соединения не указываются, по каким атрибутам производится соединение. Естественное соединение производится по всем одинаковым атрибутам.

Естественное соединение эквивалентно следующей последовательности реляционных операций:

1. Переименовать одинаковые атрибуты в отношениях.
2. Выполнить декартово произведение отношений.
3. Выполнить выборку по совпадающим значениям атрибутов, имевших одинаковые имена.
4. Выполнить проекцию, удалив повторяющиеся атрибуты.
5. Переименовать атрибуты, вернув им первоначальные имена.

Пример. В предыдущем примере ответ на вопрос "какие детали поставляются поставщиками", более просто записывается в виде естественного соединения трех отношений *P JOIN PD JOIN D* (для удобства просмотра порядок атрибутов изменен, это является допустимым по свойствам отношений):

Отношение *S=P JOIN PD JOIN D "Какие детали поставляются, какими поставщиками"*

Номер поставщика PNUM	**Наименование поставщика PNAME**	**Номер детали DNUM**	**Наименование детали DNAME**	**Поставляемое количество VOLUME**
1	Иванов	1	Болт	100
1	Иванов	2	Гайка	200
1	Иванов	3	Винт	300
2	Петров	1	Болт	150
2	Петров	2	Гайка	250
3	Сидоров	1	Болт	1000

Помимо указанных операций соединения в реляционной алгебре используются и ряд других соединений:

• ***Полусоединение*** – это соединение, которое выбирает поля только из одной таблицы, а вторую использует лишь для проверки условия соединения. Синтаксис: ***S=A⊳B***;

• ***Левое открытое (внешнее) соединение***: включает все записи левой (первой в списке) таблицы и те записи второй таблицы, которые удовлетворяют условию соединения. Синтаксис: ***S=A⊃⊲B*** *или* ***S= A LEFT JOIN B;***

• ***Правое открытое (внешнее) соединение***: включает все записи правой (второй в списке) таблицы и те записи первой таблицы, которые удовлетворяют условию соединения. Синтаксис: ***S=A⊳⊂B*** *или* ***S= A RIGHT JOIN B;***

• ***Полное соединение*** (FULL JOIN) – это объединение (UNION) левого и правого открытого соединения. Синтаксис: ***S=A⊃⊂B*** или **S= *A UNION B.***

Приведем несколько примеров использования реляционной алгебры для выражения словесных запросов в виде формулы. Они используют примерные данные базы поставщиков и деталей (рис. 2.14).

1. Получить имена поставщиков, которые поставляют деталь Д2:

$$\pi_{ПФАМ}\left(\sigma_{ДНОМ='Д2'}\left(ПД \triangleright\triangleleft ПОСТАВЩИК\right)\right)$$

2. Получить имена поставщиков, которые поставляют по крайней мере одну красную деталь:

$$\pi_{ПФАМ}\left(\pi_{ПНОМ}\left(\sigma_{ЦВЕТ='Красный'}\left(ДЕТАЛЬ\right)\triangleright\triangleleft ПД\right)\right.$$
$$\left.\triangleright\triangleleft ПОСТАВЩИК\right)$$

Отношение *ПОСТАВЩИК*

ПНОМ	ПФАМ	СТАТУС	ГОРОД
П1	Иванов	20	Воронеж
П2	Петров	15	Москва
П3	Сидоров	10	Москва
П4	Зайцев	30	Воронеж
П5	Волков	20	Киев

Соединение *ПД*

ПНОМ	ДНОМ	ШТ
П1	Д1	300
П1	Д2	200
П1	Д3	400
П1	Д4	200
П1	Д5	100
П1	Д6	100
П2	Д1	300
П2	Д2	400
П3	Д3	200
П4	Д2	200
П4	Д4	300
П4	Д5	400

Отношение *ДЕТАЛЬ*

ДНОМ	ДНАЗВ	ЦВЕТ	ВЕС	ГОРОД
Д1	Гайка	Красный	12	Воронеж
Д2	Болт	Зеленый	17	Москва
Д3	Шайба	Голубой	17	Минск
Д4	Шайба	Красный	14	Воронеж
Д5	Шуруп	Голубой	12	Москва
Д6	Гвоздь	Красный	19	Воронеж

Рис. 2.14. База данных поставщиков и деталей.

3. Получить имена поставщиков, которые поставляют все детали:

$$\left(\left(ПД\ [ПНОМ,\ ДНОМ\ ]\ divideby\ ДЕТАЛЬ\ [ДНОМ\ ]\right)\right.$$
$$\left.join\ ПОСТАВЩИК\right)[ПФАМ\ ]$$

или

$$\pi_{ПФАМ}\left(\left(\pi_{ПНОМ\ ,ДНОМ}\left(ПД\right)\div\pi_{ДНОМ}\left(ДЕТАЛЬ\right)\right)\triangleright\triangleleft ПОСТАВЩИК\right)$$

4. Получить номера поставщиков, поставляющих по крайней мере все те детали, которые поставляет поставщик П2:

$$ПД\ [ПНОМ,\ ДНОМ\ ]\ divedeby\left(ПД\ where\ ПНОМ\ ='П2'\right)[ПНОМ\ ]$$

или

$$\pi_{ПНОМ}\left(\pi_{ПНОМ\ ,ДНОМ}\left(ПД\right)\div\sigma_{ПНОМ='П2'}\left(ПД\right)\right)$$

5.Получить имена поставщиков, которые не поставляют деталь Д2:

$$\pi_{ПФАМ}\left(\left(\pi_{ПНОМ}\left(ПОСТАВЩИК\right)-\pi_{ПНОМ}\left(\sigma_{ДНОМ='Д2'}\left(ПД\right)\right)\right)\right.$$
$$\left.\triangleright\triangleleft ПОСТАВЩИК\right)$$

в выражении используются операции вычитания, естественного соединения, выборки с условием и проекции.

или

$$\left(\left(ПОСТАВЩИК\left[ПНОМ\right]minus\left(ПД\ where\ ДНОМ='Д2'\right)\left[ПНОМ\right]\right)\right.$$
$$\left.join\,ПОСТАВЩИК\right)\left[ПФАМ\right]$$

Таблица 2.4 – Сводная таблица операций реляционной алгебры

Название операции	**Обозначение**		
Объединение	$\bigcup$	**A $\bigcup$ B**	***A* UNION *B***
Пересечение	$\bigcap$	**A $\bigcap$ B**	***A* INTERSECT *B***
Декартово произведение	$\times\ \otimes$	**A ⊗ B**	**A TIMES B**
Разность	–	**A–B**	**A MINUS B**
Выборка (селекция)	σ	$\sigma_{X\,\theta\,Y}$**(A)**	**A WHERE X θ Y**
Проекция	π	$\pi_{x,\,y,\,\ldots,\,z}$ **(A)**	
Деление	÷	**A÷B**	**A divideby B**
Соединение по условию	$\triangleright\triangleleft_F$	$\sigma_{\,F}$	***A TIMES B WHERE F***
Тета-соединение (оно же, Θ-join, экви-соединение))	$\triangleright\triangleleft$ X θ Y	$\sigma_{X\,\theta\,Y}$	***A* JOIN *B* ON (*X* θ *Y*)**
Естественное соединение,	$\triangleright\triangleleft$	**A▷◁B**	***A* JOIN *B***
Полусоединение	$\triangleright$	***A*▷*B***	
Левое внешнее соединение	$\supset\triangleleft$	***A*⊃◁*B***	***A* LEFT JOIN *B***
Правое внешнее соединение	$\triangleright\subset$	***A*▷⊂*B***	***A* RIGHT JOIN *B***
Полное внешнее соединение	$\supset\subset$	***A*⊃⊂*B***	***A* UNION *B*.**

2.8.2 Реляционное исчисление

Реляционное исчисление основывается на механизме исчисления предикатов первого порядка. ***Реляционное исчисление*** – это система обозначений для получения необходимого отношения в терминах данных отношений. Реляционная алгебра и реляционное исчисление отличаются только внешне, на самом деле они эквивалентны. Однако языки исчисления – это не процедурные языки, поскольку их средствами можно выразить все, что необходимо и необязательно указывать, как это получить. Выражение в исчислении описывает лишь свойства желаемого результата, фактически не указывая, как его получить. Выражения реляционной алгебры, напротив, определяют конкретный порядок выполнения операций.

Основными понятиями исчисления являются понятие *переменной* с некоторой областью допустимых значений и понятие *правильно построенной формулы* (WFF – well formulated formula), опирающейся на **предикаты, переменные и кванторы.**

В зависимости от области определения переменной различают *исчисление кортежей* и *исчисление доменов*. В **исчислении кортежей** областью определения переменных являются отношения базы данных, т.е. допустимым значением каждой переменной является кортеж некоторого отношения. В **исчислении доменов** областью определения переменных являются домены, на которых определены атрибуты отношений базы данных, то есть допустимым значением каждой переменной является значение некоторого домена.

Предикатом $P(x_1,x_2,...,x_n)$ называется функция, принимающая значения "true" ("истина") или "false" ("ложь") от аргументов $x_1,x_2,...,x_n$, определенных в областях $D_1,D_2,...,D_n$.

При построении высказываний используются логические связки, называемые соответственно конъюнкцией, дизъюнкцией, отрицанием, импликацией и эквивалентностью. Кроме того, применяются термы сравнения, имеющие вид *x*X* , где *— символ операции сравнения (=,=>,<=,<>). Выражение *x*X (f(x)>a)* означает, что среди элементов множества *X* найдется, по крайней мере, один, при котором оказывается истинным неравенство, заключенное в скобки.

Выражение *x*X (f(x)>a)* показывает, что для всех элементов множества *X* функция *f(x)* больше заданного значения *a*. Неравенство *(f(x)>a)* представляет собой предикат: "функция от x больше константы a". Предикат принимает значение "истина" ("1") или "ложь" ("0").

Областью определения аргумента *x* предиката является множество *X*. Если указанный предикат обозначить *P(x)* и опустить явное указание области определения *X*, то получим запись *P(x)* и *x P(x)*.

Пример. Задано отношение *R* требуется найти всех сотрудников с зарплатой меньше 3000.

Отношение *R*

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Соответствующий запрос с использованием предиката (реляционного исчисления кортежей) будет иметь вид:

(r ∈R){r.Фамилия| r.Зарплата<3000 }

в данном выражении *P(r)=r.Зарплата<3000* является предикатом (условием – *where r.Зарплата<3000*).

Соответственно выражение

- *{X|P(X)}* для реляционного исчисления кортежей означает *"те X, для которых верен предикат P"*.

  или

- $(r \in R)\{r|P(r)\}$для реляционного исчисления кортежей означает, что кортежи r принадлежат отношению R и для кортежей r верно утверждение P.

Квантор (от лат. *quantum* — сколько), логическая операция, дающая количественную характеристику области предметов, к которой относится выражение, получаемое в результате её применения и ограничивающих область истинности какого-либо предиката.

Чаще всего ограничиваются **квантором всеобщности** (обозначение: $\forall$, читается: «для всех…», «для любого…» или «любой…») и **квантором существования** (обозначение: $\exists$, читается: «существует…» или «найдется…»).

Понятие, обозначаемое словом «все», лежит в основе квантора всеобщности (или квантора общности). Если через *Гр (Х)* обозначен предикат «*Х есть грек*», определенный на множестве *М* всех людей, то из этого предиката с помощью слова «все» мы можем построить высказывание «*Все люди – греки*» (конечно, ложное высказывание). Это пример применения квантора всеобщности.

Вообще же квантор всеобщности определяется так. Пусть *Р (Х)* – какой-нибудь предикат. Тогда квантор всеобщности – это операция, которая сопоставляет *Р (Х)* высказывание

«*Все Х обладают свойством Р (Х)*».

Для этой операции («все») употребляется знак $\forall$. Высказывание (*) записывается так: $\forall (X)P(X)$ (читается: «*для всех Х Р от Х*»). В соответствии со смыслом слова «все» $\forall (X)P(X)$ – ложное высказывание, кроме того единственного случая, когда *Р (Х)* тождественно-истинный предикат.

Наряду с квантором всеобщности в логике предикатов рассматривается другой квантор – «двойственный» ему квантор существования, обозначаемый знаком $\exists$:

$\exists (X)P(X)$ (читается: «*существует такое Х, что Р от Х*») – высказывание, которое истинно тогда и только тогда, когда *Р* истинно по меньшей мере для одного объекта *а* из области определения *М*. Тем самым $\exists (X)P(X)$ – истинное высказывание для всех предикатов *Р (Х)*, кроме одного – тождественно-ложного.

С помощью квантора существования легко выражается суждение типа «*Некоторые Р суть Q*» (например, «*Некоторые англичане курят*», «*Некоторые нечетные числа – простые*» и т. п.), т. е. что по крайней мере один объект *а*, обладающий свойством *Р*, обладает также свойством *Q*. Этот факт записывается формулой $\exists (X)(P(X) \wedge Q(X))$ («*Существует такой Х, что Р от Х и Q от Х*»).

Пример. Задано отношение *R*, требуется найти поставщиков болтов.

Отношение *R*

Номер поставщика	**Наименование детали**

PNUM	DNAME
1	Болт
1	Гайка
1	Винт
2	Болт
2	Гайка
3	Болт

Соответствующий запрос с использованием квантора существования (реляционного исчисления кортежей) будет иметь вид:

$$(r \in R)(r.PNAM | \exists r(\sim r.DNAME = "Болт")$$

согласно выражению требуется выбрать из атрибута *PNAM*, номера поставщиков, для которых существует такое наименование детали, которое соответствует верному предикату (условию) «болт» в атрибуте *DNAME*.

Реляционное исчисление кортежей.

Правильно построенная формула WFF служит для выражения условий, накладываемых на кортежные переменные. WFF состоит из простых сравнений скалярных значений (значений атрибутов переменных или литерально заданных констант). Более сложные варианты WFF строятся с помощью логических операций NOT, AND, OR, IF…THEN, и двух кванторов EXISTS – квантор существования) и FORALL – квантором общности.

Если *f* – формула WFF, в которой участвует переменная *x*, то

EXISTS *x* (*f*)

и

FORALL *x* (*f*)

являются допустимыми формулами WFF. Первая формула означает: «Существует по крайней мере одно значение переменной *x*, что вычисление формулы *f* для этого *x* дает значение истина». Вторая формула означает: «Для всех значений переменной *x* вычисление формулы *f* дает значение истина».

Квантор существования EXISTS определяется формально как повторяющееся *OR* (*ИЛИ*).

Квантор существования FORALL определяется как повторяющееся *AND* (*И*).

Выражением реляционного исчисления кортежей называется конструкция вида TARGET_LIST WHERE WFF. Значением выражения является отношение, тело которого определяется WFF, а набор атрибутов и их имена – целевым списком.

Выражение записывается в виде:

$\{t \mid \psi(t)\}$

где *t* – единственная свободная переменная, обозначающая кортеж фиксированной длины (если необходимо указать арность кортежа, то

используют запись $t^{(i)}$; i – арность кортежа t); ψ – правильно построенная формула (WFF).

Примеры.

Приведем несколько примеров использования реляционного исчисления кортежей для выражения словесных запросов в виде формулы. Они используют примерные данные базы поставщиков и деталей (рис. 2.14).

1. Получить номера поставщиков из Москвы со статусом больше 20:

$\{t^{(1)} \mid (\exists u)\ (ПОСТАВЩИК(u) \wedge t[1] = u[ПНОМ] \wedge u[ГОРОД] = 'Москва' \wedge$

$\wedge u[СТАТУС] > 20)\}$

(ПА.ПНОМ) WHERE ПА.ГОРОД='Москва' AND ПА.СТАТУС>20

2. Получить имена поставщиков, которые поставляют деталь Д2:

(ПА.ПФАМ) WHERE EXISTS ПДА (ПДА.ПНОМ = ПА.ПНОМ

AND ПДА.ДНОМ = 'Д2') или

$\{t^{(1)} \mid (\exists u)\ (\exists v)\ (ПД(u) \wedge ПОСТАВЩИК(v) \wedge t[1] = v[ПФАМ] \wedge$

$\wedge u[ПНОМ] = v[ПНОМ] \wedge u[ДНОМ] = 'Д2')\}$

3. Получить имена поставщиков, которые поставляют, по крайней мере, одну красную деталь:

(ПА.ПФАМ) WHERE EXISTS ПДА (ПА.ПНОМ = ПДА.ПНОМ AND

EXISTS ДА(ДА.ДНОМ = ПДА.ДНОМ AND ДА.ЦВЕТ = 'Красный')) или

$\{t^{(1)} \mid (\exists u)\ (ПОСТАВЩИК(u) \wedge t[1] = u[ПФАМ] \wedge (\exists v)\ (ПД(v) \wedge$

$\wedge u[ПНОМ] = v[ПНОМ] \wedge (\exists w)\ (ДЕТАЛЬ(w) \wedge$

$\wedge w[ДНОМ] = v[ДНОМ] \wedge w[ЦВЕТ] = 'Красный'))\}$

4. Получить имена поставщиков, которые не поставляют деталь Д2:

(ПА.ПФАМ) WHERE NOT EXISTS ПДА

(ПДА.ПНОМ = ПА.ПНОМ AND ПДА.ДНОМ = 'Д2') или

$\{t^{(1)} \mid (\exists u)\ (\exists v)\ (ПД(u) \wedge ПОСТАВЩИК(v) \wedge t[1] = v[ПФАМ] \wedge$

$\wedge(\neg\,(u[ПНОМ] = v[ПНОМ] \wedge u[ДНОМ] = 'Д2')))\}$

5. Получить номера деталей, которые или весят более 16, или поставляются поставщиком П2, или и то и другое:

ДА.ДНОМ WHERE ДА.ВЕС > 16 OR

EXISTS ПДА (ПДА.ДНОМ = ДА.ДНОМ AND

ПДА.ПНОМ = 'П2') или

$$\{t^{(1)} \mid (\exists u)\,(ДЕТАЛЬ(u) \wedge t[1] = u[ДНОМ] \wedge u[ВЕС] > 16 \vee$$

$$\vee (\exists v)\,(ПД(v) \wedge v[ДНОМ] = u[ДНОМ] \wedge$$

$$\wedge v[ПНОМ] = 'П2'))\}$$

Реляционное исчисление доменов.

Реляционное исчисление, ориентированное на домены (или исчисление доменов), отличается от исчисления кортежей тем, что в нем используются переменные доменов вместо переменных кортежей, т.е. переменные, принимающие свои значения в пределах домена, а не отношения.

Основным формальным отличием исчисления доменов от исчисления кортежей является наличие дополнительного набора предикатов, позволяющих выражать так называемые **условия членства**. Если *R* - это *n*-арное отношение с атрибутами *t1, t2, ..., tn*, то условие членства имеет вид:

$$R\,(pair, pair, \ldots),$$

где каждая пара *pair* имеет вид *t:v*, при этом *v* – это либо литерально задаваемая константа, либо имя доменной переменной. Условие членства принимает значение *true* в том и только в том случае, если в отношении *R* существует кортеж, содержащий значения указанных атрибутов. Если *v* – константа, то на атрибут *t* задается жесткое условие, не зависящее от текущих значений доменных переменных; если же *v* – имя доменной переменной, то условие членства может принимать разные значения при разных значениях этой переменной. Например, вычисление выражения

ПД (ПНОМ:'П1', ДНОМ:'Д1')

дает значение *true*, если и только если в отношении *ПД* существует кортеж со значением *ПНОМ*, равным *П1*, и значением *ДНОМ*, равным *Д1*. Аналогично, условие членства

ПД (ПНОМ:ПНОМА, ДНОМ:ДНОМА)

принимает значение true, если и только если в отношении *ПД* существует кортеж со значением *ПНОМ*, эквивалентным текущему значению переменной домена *ПНОМА* (какому бы то ни было), и значением *ДНОМ*, эквивалентным текущему значению переменной домена *ДНОМА* (опять же, какому бы то ни было).

Во всех остальных отношениях формулы и выражения исчисления доменов выглядят похожими на формулы и выражения исчисления кортежей. В частности, конечно, различаются свободные и связанные вхождения доменных переменных.

Далее будем считать, что существуют переменные доменов с именами, образуемыми добавлением цифр 1, 2, 3, ... к соответствующим именам доменов. Кроме того, предполагается, что в базе данных поставщиков и деталей каждый атрибут имеет такое же имя, как и соответствующий ему домен, за исключением атрибутов *ПФАМ* и *ДНАЗВ*, для которых соответствующий домен называется просто *ИМЯ*.

Например, выражение

(ПНОМ1) WHERE ПОСТАВЩИК (ПНОМ:ПНОМА, ГОРОД: 'Воронеж') означает подмножество всех номеров поставщиков из города Воронеж.

С использованием традиционного синтаксиса языка предикатов реляционное исчисление с переменными доменов имеет вид:

$$\{x_1 x_2 \ldots x_k \mid \psi(x_1, x_2, \ldots, x_k)\},$$

где ψ - формула, обладающая тем свойством, что только ее свободные переменные доменов являются различными переменными $x_1, x_2, \ldots, x_k$.

Реляционное исчисление доменов является основой большинства языков запросов, основанных на использовании форм. В частности, на этом исчислении базируется известный язык QBE (Query-by-Example), который был первым (и наиболее интересным) языком в семействе языков, основанных на табличных формах.

Примеры.

Приведем некоторые рассмотренные выше примеры, но выраженные в терминах исчисления доменов.

1. Получить номера поставщиков из Москвы со статусом больше 20:

ПНОМ1 WHERE EXISTS СТАТУС1
(СТАТУС1>20 AND ПОСТАВЩИК (ПНОМ:ПНОМ1,
СТАТУС:СТАТУС1, ГОРОД:'Москва')) или
{ПНОМ1|∃(□□СТАТУС1) (СТАТУС1>20∧□□ПОСТАВЩИК(ПНОМ1, СТАТУС1, ГОРОД)□ ∧□□ГОРОД='Москва')}

2. Получить имена поставщиков, которые поставляют, по крайней мере, одну красную деталь

ИМЯ1 WHERE EXISTS ПНОМ1 EXISTS ДНОМ1

(ПОСТАВЩИК (ПНОМ:ПНОМ1, ПФАМ:ИМЯ1)

AND ПД (ПНОМ: ПНОМ1, ДНОМ: ДНОМ1)

AND ДЕТАЛЬ (ДНОМ:ДНОМ1, ЦВЕТ: 'Красный')) или

{ИМЯ1 | ∃ (□□ПНОМ1) (∃□□ДНОМ1)(ПОСТАВЩИК(ПНОМ1, ИМЯ1) ∧□ ∧□ПД(ПНОМ1, ДНОМ1) ∧ДЕТАЛЬ (ДНОМ1, ЦВЕТ) ∧□ЦВЕТ = 'Красный')}□

3. Получить номера поставщиков, поставляющих по крайней мере все детали, поставляемые поставщиком П2

ПНОМ1 WHERE FORALL ДНОМ1 (IF ПД (ПНОМ: 'П2',

ДНОМ:ДНОМ1) THEN ПД (ПНОМ:ПНОМ1, ДНОМ:ДНОМ1)) или

{ПНОМ1 | ∀(□ДНОМ1) ((ПД (ПНОМ, ДНОМ1) ∧

∧□ПНОМ= 'П2')□→□ПД (ПНОМ1, ДНОМ1))}

Глава 3. Введение в проектирование реляционных баз данных

3.1 Уровни моделей и этапы проектирования базы данных

В процессе проектирования базы данных отражение предметной области представлено моделями нескольких уровней.

Модель предметной области - это знания о предметной области. Знания могут быть как в виде неформальных знаний в мозгу эксперта, так и выражены формально при помощи каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т.п. Опыт показывает, что текстовый способ представления модели предметной области крайне неэффективен. Гораздо более информативными и полезными при разработке баз данных являются описания предметной области, выполненные при помощи специализированных концептуальных или инфологических (семантических) моделей выполненных на основе графических нотаций.

Инфологической моделью называется описание общей логической структуры предметной области (базы данных) выполненное без ориентации на используемые в дальнейшем программные и технические средства. Такой информацией являются сведения о классах объектов предметной области и их количеств в каждом классе, о фиксируемых свойствах этих объектов, о связях между ними, динамике их изменения. Инфологическая модель отражает объективные, «внутренние» характеристики предметной области, поэтому она сравнительно стабильна.

Концептуальная модель данных создается на основе информации записанной в спецификациях требований пользователей и обеспечивает интегрированное представление о предметной области. Концептуальная модель не зависит от физических аспектов хранения или представления информации и не учитывает особенности конкретной СУБД.

Даталогической или логической моделью называют модель данных логического уровня, поддерживаемую средствами СУБД или концептуальную модель с учетом поддержки её средствами СУБД. Эта модель представляет собой отображение логических связей между элементами данных безотносительно к их содержанию и среде хранения. Даталогическая модель строится с учетом ограничений конкретной СУБД. При построении даталогической модели учитываются особенности отображаемой предметной области. База данных предполагает интегрированное и взаимосвязанное хранение данных, поэтому для проектирования данной модели необходимо иметь соответствующее концептуальное описание предметной области.

Для привязки даталогической модели к среде хранения используется физическая модель.

Физическая (внутренняя) модель данных описывает данные средствами конкретной СУБД. Например, для реляционной СУБД, в физической модели отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для

ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

Физическая модель определяет используемые запоминающие устройства, способ расположения элементов данных в памяти, способы физической реализации логических отношений между элементами данных. Модель физического уровня (или внутреннего уровня) строится с учетом ограничений СУБД и операционной системы.

Внешняя модель состоит в описании и построении схем или логических структур с точки зрения конкретного пользователя.

В зависимости от широты охвата предметной области различают также глобальные и локальные модели. ***Глобальные модели*** отражают точку зрения администратора базы данных, ***локальные*** - взгляды различных пользователей.

Модель каждого из последующих уровней строится на основе фиксированных характеристик моделей предшествующих уровней.

Указанные модели имеют разный уровень абстракции. Соответственно последовательность уровней абстракции проектирования базы данных по мере убывания их зависимости от физического формата хранения: физический, логический, концептуальный.

Выделение моделей разных уровней абстракции позволяет:

–разделить сложный процесс отображения «предметная область – база данных» на несколько итеративных более простых отображений;

–обеспечить специализацию разработчиков баз данных; возможность работать разным категориям пользователей с моделью соответствующего уровня;

–предоставить возможность активного и конструктивного участия в разработке баз данных лицам, не имеющим профессиональных навыков в области обработки данных;

–создать предпосылки автоматизации проектирования баз данных путем формализованного перехода с одного уровня моделей на другой.

Этапы проектирования баз данных.

Число этапов проектирования базы данных зависит от количества уровней представления данных, или моделей данных. Соответственно различают следующие этапы проектирования: системный, инфологический, датологический, физический. Или с учетом трехуровневой архитектуры базы данных стандарта ANSI – концептуальное, внутренние и внешнее проектирование (рис.3.1).

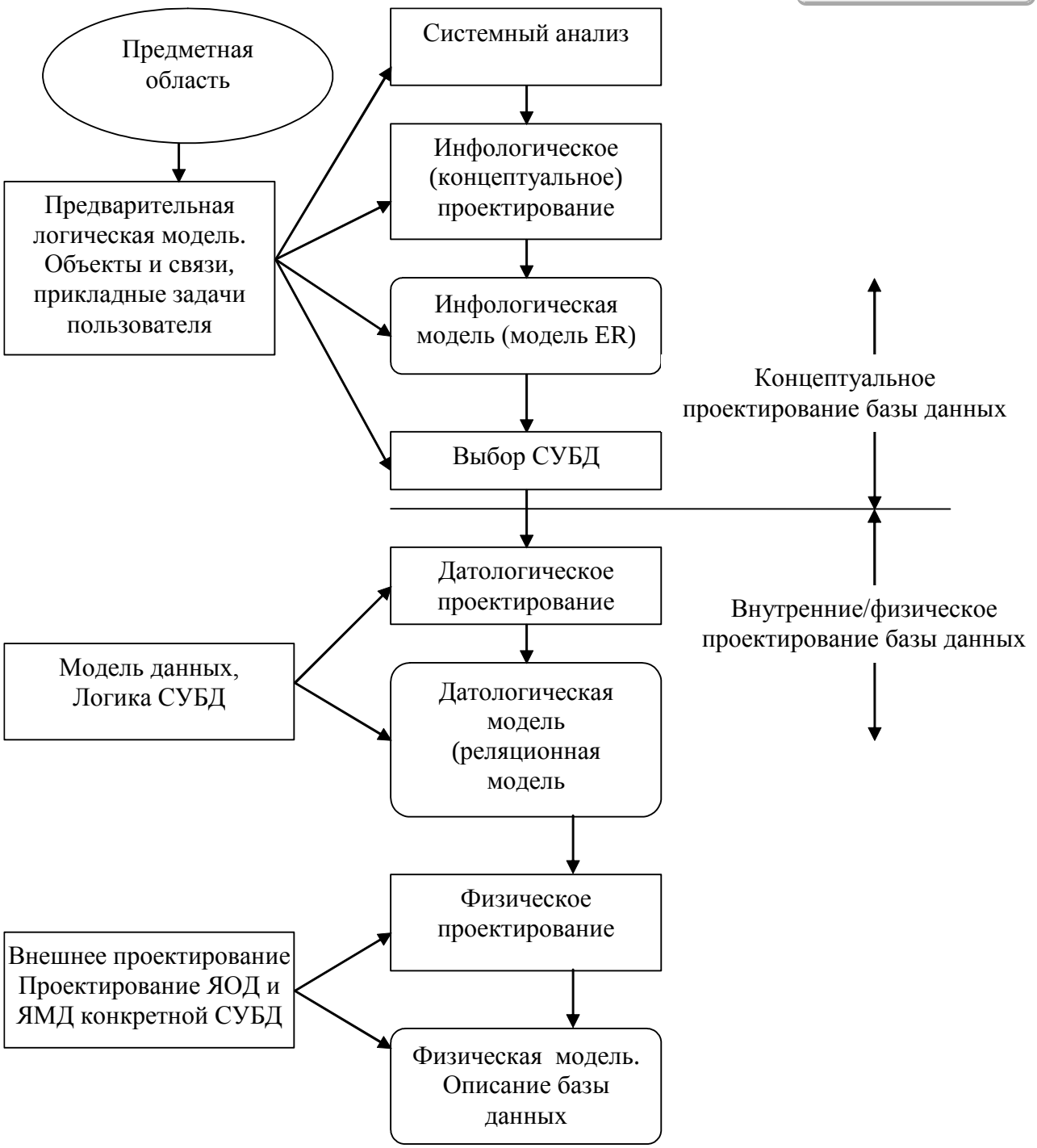


Рис.3.1. Этапы проектирования базы данных.

Системный анализ – цель выделение предметной области, как систему объектов и определение их взаимосвязей, определение парадигмы (образец) информационной модели (структура, способ представления информации, характер использования информации).

С точки зрения проектирования базы данных в рамках системного анализа, необходимо провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Желательно, чтобы данное описание позволяло корректно определить все взаимосвязи между объектами предметной области.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

Функциональный подход – он реализует принцип движения «от задач» и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая база данных. В этом случае можно четко выделить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

Предметный подход – когда информационные потребности будущих пользователей базы данных жестко не фиксируются. Они могут быть многоаспектными и весьма динамичными. Так как достаточно сложно точно выделить минимальный набор объектов предметной области, которые необходимо описывать. В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее. БД, конструируемая при этом, называется предметной, то есть она может быть использована при решении множества разнообразных, заранее не определенных задач.

Чаще всего на практике используется некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ заканчивается подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в базе данных, формулировкой конкретных задач, которые будут решаться с использованием данной базы данных с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения информацией базы данных.

Инфологическое (концептуальное) проектирование состоит в описании и построении инфологической модели, схем отражений предметной области, выполненном без ориентации на используемые программные и технические средства. то есть не зависящей от любых физических аспектов представления данных.

Инфологическое (концептуальное) проектирование основано на информации записанной в спецификациях требований пользователей и абсолютно не зависит от таких подробностей ее реализации, как тип выбранной целевой СУБД, набор создаваемых прикладных программ, используемые языки программирования, тип выбранной вычислительной платформы, а так же от любых других особенностей физической реализации.

Даталогическое проектирование основано на модели логического уровня и представляет собой процесс создания, описания и построения схем связей между элементами данных с учетом выбранной модели организации данных, в зависимости от типа целевой СУБД (реляционная, сетевая, объектно-ориентированная), но не зависимо от других физических аспектов реализации.

На этом этапе игнорируются все остальные аспекты выбранной СУБД – например, любые особенности среды хранения данных, физической организации структур хранения данных, построения индексов.

Инфологическое и даталогическое проектирование – итеративные процессы, которые начинаются в определенный момент и продолжаются в виде практически бесконечного ряда уточнений и улучшений. Их следует рассматривать, как некий процесс изучения. По мере углубления разработчиками понимания предметной области или принципов организации какого-либо предприятия и смысла используемых данных, они выражают это понимание в виде той или иной модели данных. Инфологическое и даталогическое проектирование базы данных – важнейший фактор общего успеха разрабатываемой системы. Если модели не являются точным отражением предметной области или методов работы и структуры предприятия, то могут возникнуть сложности с физической реализацией базы данных или обеспечением приемлемой производительности системы.

Физическое проектирование состоит в описании и построении схем хранения данных для определенной среды хранения, в частности реализации базы данных на вторичных запоминающих устройствах с указанием структур хранения и методов доступа, используемых для организации эффективной обработки данных. На этом этапе осуществляется выбор типа носителя, способ организации данных, методов доступа, определение параметров физического блока, управление работой памяти, считывание данных и т.д.

На этапе физического проектирования базы данных, прежде всего, необходимо выбрать конкретную целевую СУБД. Физическое проектирование неразрывно связанно с конкретной СУБД. Между даталогическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Вообще, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных. В случаи реляционной модели данных под этим подразумевается следующие:

–создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной модели данных;

–определение конкретных структур хранения данных и методов доступа к ним, обеспечение оптимальной производительности системы с базой данных;

–разработка средств защиты создаваемой системы.

Этапы инфологического и даталогического проектирования связанны с вопросами *что делать*, физического *как сделать*.

Внешнее проектирование состоит в описании и построении схем или логических структур с точки зрения конкретного пользователя. На этом этапе формализуются допустимые режимы обработки данных в рамках данной схемы или подсхемы. Для реляционных моделей это описание процедуры View конкретного приложения.

Проектирование баз данных представляет собой переход от исходного описания предметной области к схеме баз данных. Процесс проектирования структуры базы данных в целом, как и её частей, носит итеративный характер. Если на начальных стадиях проектирования трудно выбрать единственный вариант проектного решения, то довести до стадии моделирования и даже до стадии натурных испытаний можно, опробовав несколько различных вариантов.

Обычно сначала строится предварительная логическая структура, которая в общем виде отображает предметную область и функциональные требования, затем переходят к детальному логическому структурированию. Причем, предварительная структура будет служить своеобразным эталоном, с которым сравниваются получающиеся в результате проектирования структуры. Части структур, которые не соответствуют друг другу тщательно исследуются, и в структуру, которая оказалась неадекватной вносятся изменения.

Например, даталогическую модель, построенную от предметной области, анализируют с точки зрения возможности и эффективности выполнения заданных функций. В процессе анализа даталогическая модель может перестраиваться. После получения приемлемого варианта логической структуры базы данных переходят к физическому проектированию. Полученный вариант физической структуры проверяют на ограничение по ресурсам. В случае несоблюдения ограничений производят перепроектирование физической структуры. Если этого оказывается недостаточно, то возвращаются к даталогическому уровню и выполняют еще одну итерацию по проектированию структуры базы данных.

3.2 Инфологическое (концептуальное) проектирование

Инфологическое (концептуальное) проектирование – **цель** – определение системы атрибутов, типовых запросов, типовых процедур на основе использования специальных искусственно формализованных языковых средств (естественный, математический, алгоритмический и др).

Результат – инфологическая модель – неформальное описание создаваемой базы данных, адекватно отражающее предметную область и используемое в качестве источника информации для последующих этапов построения базы данных. Инфологическая модель не зависит от физических параметров среды хранения данных, технических и программных средств.

Первые две стадии определяются объектами и связями предметной области и прикладными задачами пользователя.

Например, рассмотрим бухгалтерскую операцию – внесение некоторой суммы на текущий счет. Учетный отдел банка сохраняет такие детали, как номер счета, сумма вклада, дата, номер кассира и опускает некоторые другие (количество посетителей в банке, длину очереди и др.). Таким образом, представление учетного отдела об операции будет содержать лишь те детали, которые будут признаны существенными.

Инфологическая модель должна отвечать следующим требованиям:

–обеспечивать адекватное отображение предметной области и, как следствие, давать возможность получить интегрированное представление о предметной области;

–представляться на языке, понятном как специалисту предметной области, так и администратору базы данных;

–содержать информацию о предметной области, достаточную для дальнейшего проектирования;

–гарантировать однозначное трактование модели;

–быть динамической;

–степень ее формализации должна быть высокой, чтобы обеспечить возможность автоматизированного перехода к моделям следующего уровня.

Разработка модели инфологического этапа позволяет при изменении используемых программных и технических средств не полностью перепроектировать базу данных, а только выполнить переход от инфологической модели к схеме, поддерживаемой новыми программно-техническими средствами.

Концептуальное проектирование базы данных является в значительной степени эвристическим процессом, и адекватность построенной модели предметной области проверяется в большинстве случаев эмпирически по анализу и проверке удовлетворения информационных потребностей пользователей.

В процедуре концептуального проектирования можно выделить следующие этапы:

1) Обзор и изучение области использования базы данных для формирования общего представления о предметной области.

2) Формирование и анализ круга функций и задач базы данных.

3) Определение основных субъектов-сущностей предметной области и отношений между ними.

4) Формализованное описание предметной области.

Обзор и изучение области использования базы данных для формирования общего представления о предметной области осуществляется разработчиком в непосредственном взаимоотношении с заказчиком. При этом изучается необходимая документация. На этой основе определяются основные процессы,

участники и информационные потоки в предметной области. Принципиальным моментом является фрагментирование предметной области, т.е. ее разделение на организационные, технологические, функциональные фрагменты. В результате выделяются, например, такие фрагменты предметной области как: подразделения, сотрудники организации; исполняющие документы; мероприятия; документы, обработка которых или подготовка которых реализует управленческие решения; регистрация, учет, обработка, хранение документов и др.

Так формируется представление о существующей («как-есть») технологии формирования, накопления и использования информации в рамках предметной области и анализируются совместно с заказчиком «узкие места» и недостатки в существующей технологии.

Главным итоговым результатом концептуального проектирования является *определение и описание основных объектов-сущностей предметной области и отношений между ними*.

3.2.1 Модель «Сущность – связь»

Одним из основных результатов Центральным компонентом инфологической (концептуальной) модели, как результат проектирования является ER-модель или модель «Сущность–связь» (ER–диаграмма, схема данных), описывающая объекты предметной области и связи между ними. Название ER-модель определяется аббревиатурой словосочетания ESSENCE (сущность) – RELATION (связь).

Модель «сущность-связь» (ER-модель) представляет собой высокоуровневою концептуальную модель данных, которая была разработана Ченом в 1976 году с целью упрощения задач проектирования баз данных. Данная модель представляет набор концепций, которые описывают структуру базы данных и связанные с ней транзакции обновления и извлечения данных..

Модель "сущность-связь" основывается на некой важной семантической информации о реальном мире и предназначена для логического представления данных. Она определяет значения данных в контексте их взаимосвязи с другими данными. Важным является тот факт, что из модели "сущность-связь" могут быть порождены все существующие модели данных (иерархическая, сетевая, реляционная, объектная), поэтому она является наиболее общей.

Следует отметить, что модель "сущность-связь" не является моделью данных в том смысле, как это было определено в разделе 6, поскольку не определяет операций над данными и ограничивается описанием только их логической структуры.

Основными понятиями модели "сущность-связь" являются: сущность, экземпляр, связь и атрибут.

Сущность - это содержание (описание) явления, процесса, объекта (предметной области реальной или представляемой), о котором собирается

информация, и которая должна сохраняться в проектируемой системе. Сущность имеет имя, уникальное в пределах системы. *Примеры* сущностей: люди, продукты, студенты и т.д. Любой фрагмент предметной области может быть представлен как множество сущностей, между которыми существует некоторое множество связей.

Следует различать тип сущности и конкретные ее экземпляры.

Тип сущности – признак принадлежности к некоторому классу (множеству) объектов (явлений). Тип сущности характеризуется множеством ее атрибутов.

Типы сущностей можно классифицировать как сильные и слабые.

Слабый тип сущности – тип сущности, существование которого зависит от какого-то другого типа сущности (зависимый тип сущности).

Сильный тип сущности – тип сущности, существование которого не зависит от какого-то другого типа сущности (независимый тип сущности).

Экземпляр сущности – конкретный объект, принадлежащий определенному классу объектов. Экземпляр сущности определяется значениями ее атрибутов. Примеры экземпляров сущностей: конкретный человек, конкретный продукт, конкретный студент и т.д.

Сущность является ***независимой***, если каждый её экземпляр может быть однозначно идентифицирован без установления связи с другой сущностью. Сущность является ***зависимой*** (дочерней), уникальная идентификация её экземпляров возможна только путём установления связи с другой сущностью.

Атрибут – поименованная характеристика сущности или данные, описывающие свойства сущности. Примеры атрибутов: фамилия, цвет, стоимость, дата. Атрибут в концептуальной модели имеет уникальное имя, роль и описание в пределах данной сущности.

Атрибуты сущности содержат значения, описывающие каждую сущность. В базе данных хранятся только наборы значений атрибутов, которые однозначно определяют один экземпляр сущности.

Домен атрибута – набор значений, которые могут быть присвоены атрибуту.

Различные атрибуты могут совместно использовать один и тот же домен. Полностью разработанная модель данных включает домены каждого атрибута, присутствующего в ER-модели (табл.3.1).

Таблица 3.1 – Сущность – «персона», её атрибуты и значения атрибутов

Сущность – «персона» **Атрибуты:**	**Экземпляр сущности «персона»** **Значения атрибутов:**
Фамилия	Иванов
Имя	Иван
Отчество	Иванович
Дата рождения	15.05.1967
Номер паспорта	40 03 012345
Номер телефона	123-45-67, 987-65-43

Атрибуты делятся на простые и составные, однозначные и многозначные, а также на производные.

Простой атрибут – атрибут, состоящий из одного компонента с независимым существованием.

Простые атрибуты не могут быть разделены на более мелкие компоненты (Пол, зар. плата).

Составной атрибут – атрибут, состоящий из нескольких компонентов, каждый из которых характеризуется независимым существованием (Например, атрибут *Адрес* состоит из города, улицы, дома и.т.д.)

Однозначный атрибут – атрибут, который содержит одно значение для одной сущности.

Многозначный атрибут – атрибут, который содержит несколько значений для одной сущности.

Например, сущность «*Отделение_компании*» может иметь несколько значений для атрибута *Номера_телефона* следовательно, атрибут *Номер_телефона* в этом случае будет многозначным.

Реляционная модель, в которую должна быть впоследствии преобразована данная инфологическая модель, не допускает многозначности атрибутов. Данное противоречие должно быть разрешено. Возможное решение – образование новой сущности. В приведенном ранее примере атрибут «номер телефона» становится кандидатом на создание сущности «Номер телефона».

Производный атрибут – атрибут, который представляет значение, производное от значения связанного с ним атрибута или некоторого множества атрибутов, принадлежащих некоторому (не обязательно данному) типу сущности.

Некоторые атрибуты могут быть связанны с определенной сущностью. Например, атрибут *Возраст* сотрудника является производным от атрибута *Дата_рождения*. В некоторых случаях значение атрибута является производным от многих сущностей одного и того же типа сущности. Например, атрибут *Количество_сорудников* сущности «*Отделение_компании*» может быть вычислен на основе подсчета количества сущностей «*Отделение_компании*.»

Ключ сущности (первичный ключ) - это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.

Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам.

Потенциальный ключ – атрибут или набор атрибутов, который уникально идентифицирует отдельные экземпляры типа сущности

Потенциальный ключ, должен содержать значения, которые уникальны для каждого отдельного экземпляра сущности данного типа. Например, каждое *Отделение_компании* обладает уникальным *Номером* и не существует отделений с одинаковыми номерами.

Тип сущности может иметь несколько потенциальных (первичных) ключей. Выбор первичного ключа сущности осуществляется исходя из соображений суммарной длины атрибутов в ключе, а так же наличия гарантий уникальности его значений в текущей момент времени и в обозримом будущем. При этом потенциальные ключи, которые не являются первичным, называются **альтернативными ключами.**

Составной ключ - потенциальный ключ, который состоит из двух или более атрибутов.

Между сущностями могут быть установлены связи.

Связь - это ассоциация, установленная между несколькими сущностями, и показывающая как взаимодействуют сущности между собой.

Односторонние связи между сущностями представляют с помощью трех типов ассоциаций: простой (1), сложной (М) и условной (С).

Ассоциация типа 1 (простая). Ассоциация этого типа определяет однонаправленную связь от элемента *A* к элементу *B*, при которой одному и тому же экземпляру *A* соответствует один и тот же экземпляр элемента *B*. При этом обратная связь не определена (рис.3.2).

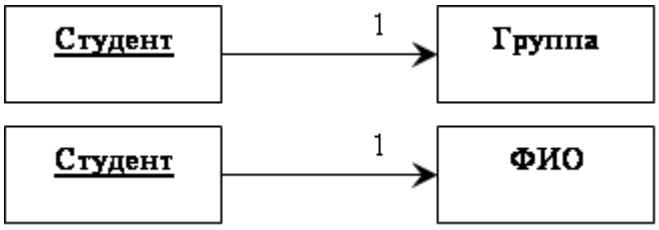


Рис.3.2. Ассоциация типа 1 (простая).

Данная идентификация является уникальной (атомарной) и определяет функциональную зависимость. Пример ассоциации типа 1: между элементами данных СТУДЕНТ № (номер студенческого билета) и ГРУППА (номер группы), СТУДЕНТ № и ФИО (фамилия, имя, отчество студента). Студент имеет только одну фамилию, имя, отчество, и он учится только в одной группе. Связи в обратном направлении не рассматриваются.

Ассоциация типа М (сложная). Определяет однонаправленную связь от элемента *A* к элементу *B*, при котором одному и тому же экземпляру *A* соответствует 0,1 или несколько экземпляров элемента *B*, при этом обратная связь не определена (рис.3.3).

Идентификация не обязательно является уникальной и представляет собой многозначную зависимость. Рассмотрим примеры ассоциации типа М между элементами данных ГРУППА и СТУДЕНТ №, ПРЕПОДАВАТЕЛЬ (фамилия, имя, отчество преподавателя) и ДИСЦИПЛИНА (читаемая им дисциплина, курс) (рис. 3.3). В данной группе могут обучаться много студентов, и данный преподаватель может читать много дисциплин. Связи в обратном направлении не рассматриваются.

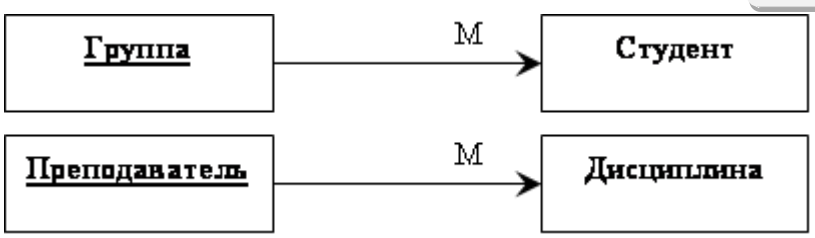


Рис.3.3. Ассоциация типа M (сложная).

Ассоциация типа C (условная). Для данного экземпляра элемента данных, от которого направлена связь, может не существовать соответствующего экземпляра элемента данных, к которому связь направлена, но если она существует, то относится к единственному экземпляру элемента данных. В данном случае говорят об условной ассоциации или ассоциации типа C. Идентификация, если существует, является уникальной (рис.3.4).

Идентификация, если существует, является уникальной. Приведем примеры для ассоциаций типа C для элементов данных ПРЕПОДАВАТЕЛЬ и АУДИТОРИЯ (аудитория, где проводится экзамен), ПРЕПОДАВАТЕЛЬ и ДАТА УВОЛЬНЕНИЯ (дата увольнения преподавателя) (рис.3.4). Преподаватель может проводить экзамен, а может и нет (например, по его предмету зачет), но если проводит, то только в одной аудитории. Аналогично, он может уволится из данного института или нет, но если он уволится, то дата увольнения будет одна.

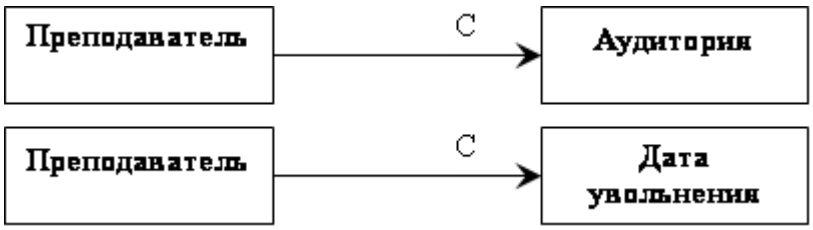
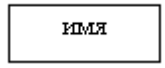
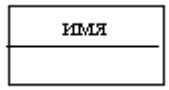
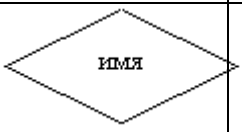
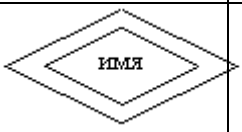


Рис.3.4. Ассоциация типа C (условная).

Если каждый экземпляр сущности обязательно участвует в связи, то говорят, что сущность имеет **обязательный класс** принадлежности. Если экземпляры данной сущности могут не участвовать в связи, то класс принадлежности этой сущности является **необязательным.**

Очень важным свойством модели "сущность-связь" является то, что она может быть представлена в виде графической схемы (диаграммы). Это значительно облегчает анализ предметной области. Существует несколько вариантов обозначения элементов диаграммы "сущность-связь", так называемые нотации, каждый из которых имеет свои положительные черты. В настоящее время для проектирования баз данных используются стандартизованные нотации IDEF1X**,** Баркера, UML основанные на нотациях Чена, Мартина. В таблицах 3.1, 3.2, 3.3 приводится список часто используемых обозначений в диаграммах.

Таблица 3.1 – Элементы диаграммы "сущность-связь" в нотации Чена

Элемент диаграммы	Обозначает
	Независимая сущность
	Зависимая сущность
	Родительская сущность в иерархической связи
	Связь
	Идентифицирующая связь
	Атрибут
	Первичный ключ
	Внешний ключ (понятие внешнего ключа вводится в реляционной модели данных)
	Многозначный атрибут
	Получаемый (производный) атрибут в иерархических связях

Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности (степень связи). Пример (рис.3.5):

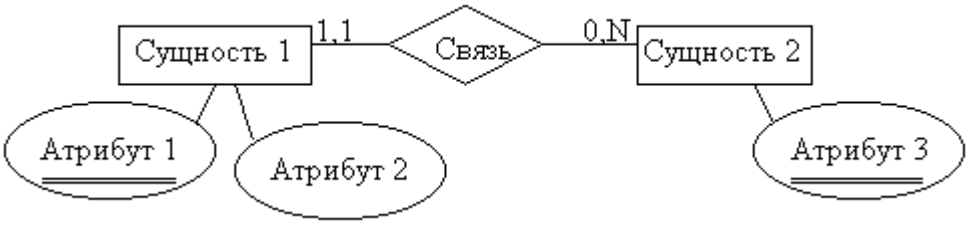


Рис.3.5 – Пример диаграммы "сущность-связь".

Связи изображаются линиями, соединяющими сущности, вид линии в месте соединения с сущностью определяет кардинальность связи. Существуют разнообразные способы обозначения связей на диаграммах "сущность-связь".

Таблица 3.2 – Обозначение связей в нотации Мартина

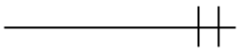
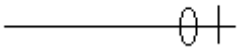
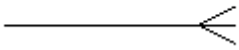
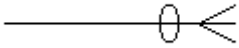
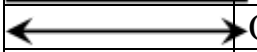
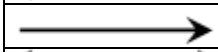
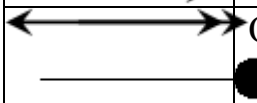
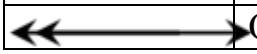
Элемент диаграммы	Обозначает
	Связь 1,1
	Связь степени 1, необязательный класс принадлежности (1,1)
	Связь степени 1, обязательный класс принадлежности (0,1)
	Связь M,N
	Связь степени N, необязательный класс принадлежности (0,N)
	Связь степени N, обязательный класс принадлежности (1,N)

Таблица 3.3 – Обозначение связей на диаграммах "сущность-связь"

Элемент диаграммы	Обозначает
	Двойная линия – обязательный класс принадлежности
	Тонкая линия – необязательный класс принадлежности
	Обозначение связи (отображений) (1,1)
	Обозначение ассоциаций
	Обозначение связи (1,M)
	Обозначение связи (M,1)

Пример (рис.3.6):

Рис.3.6. Пример обозначения связи в нотации Мартина.

На диаграммах "сущность-связь" обязательное участие в связи экземпляров сущности может отмечаться блоком с точкой или стрелкой внутри, смежным с блоком этой сущности I. При необязательном участии экземпляров

сущности в связи дополнительный блок к блоку сущности не пристраивается, а точка или стрелка размещается на линии связи (рис. 3.7).

Рис.3.7 – Пример обозначения связи.

Степень связи - количество сущностей, которые охвачены данной связью.

Охваченные некоторой связью сущности называются участниками этой связи. Количество участников – степень связи. ***Унарная*** связь (связь со степенью 1, рекурсивная) – это связь сущности с самой собой (сотрудник – начальник, обмен одной недвижимости на другую и т.п.). ***Бинарная*** – связь (связь со степенью 2) между двумя сущностями (рис.3.8). ***Тернарная*** – связь (связь со степенью 3) между тремя сущностями (рис.3.9) и т.д.

Рис.3.8 –Пример бинарной связи.

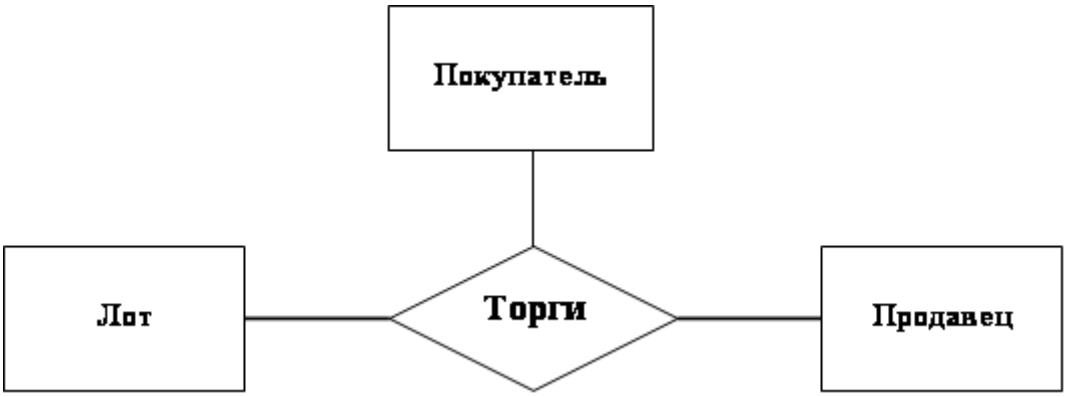


Рис.3.9 – Пример тернарной связи.

Полное участие в связи – это обязательная связь, неполное участие – не обязательная связь.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется **идентифицирующей**, в противном случае – **не идентифицирующей**.

Сущность-потомок в идентифицирующей связи зависит от сущности-родителя.

Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой сущностью (это определяется ее связями с другими сущностями).

Связь между двумя сущностями, или сущности самой с собой, может принадлежать к одному из следующих типов:

– идентифицирующая связь;
– не идентифицирующая связь;
– типизирующая связь;
– связь многие-ко-многим;
– рекурсивная связь.

Каждый тип связи определяет поведение атрибутов первичного ключа, когда они мигрируют из родительской сущности в подчиненную.

Рекурсивная связь – это не идентифицирующая связь между двумя сущностями, которая указывает, что экземпляр сущности может быть связан с другим экземпляром той же самой сущности (рис.3.10).

Рис.3.10 – Пример унарной (рекурсивной) связи.

В рекурсивной связи *Управляет*, которая представляет взаимосвязь подчиненных с руководителем, который то же входит в состав персонала. Таким образом сущность «Сотрудник» дважды входит в связь *Управляет*, первый раз в качестве руководителя, а второй раз в качестве сотрудника, которым управляют.

Связям могут присваиваться **ролевые имена (имя роли)**, которые предназначены для указания назначения участников связи. Ролевые имена (фактически) являются синонимом атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. Ролевое имя обязательно указывать в том случае, если два и более атрибута одной сущности определены на одном и том же домене (Дети (папа, мама) – два внешних ключа, Сотрудники (руководитель) – рекурсивная связь).

Ролевые имена имеют большое значение в рекурсивных связях, при определении функций каждого участника связи. На рисунке показан пример использования ролевых имен в рекурсивной связи *Управляет*. Первое участие сущности «Сотрудник» получило ролевое имя *Руководитель*, а второе – *Подчиненный*. Кроме того, ролевые имена могут так же использоваться, когда две сущности связанны несколькими связями.

Ролевые имена обычно не требуются, если функции сущностей - участниц связей определенны недвусмысленно.

Кардинальность связи: это число вхождений сущности в данную связь (или количество возможных связей для каждой из сущностей, участвующих в связи).

Ассоциации между парой связанных элементов, определенных в обе стороны, представляют собой **отображения**. Отображения являются традиционным средством для определения характера взаимосвязей между элементами данных, так как описывают двусторонние связи между ними.

В ряде случаев связи от атрибутов к ключам не определяются, а важное значение имеют ассоциации, определяющие связи ключ-атрибут. Однако для некоторых задач произвольный выбор характера инверсных ассоциаций может привести к противоречиям с реальными информационными потребностями пользователей.

Наиболее распространенными отображениями являются бинарные связи с показателем кардинальности «один к одному», «один ко многим», «многие к одному» «многие ко многим» (1:1, 1:M, M:1, M:M).

Типы связей (отображений):

Связь «один к одному» (1:1) – представляют такой тип связи, когда один экземпляр элемента данных А, от которого направлена связь, идентифицирует один и только один экземпляр элемента данных В, к которому направлена связь, и наоборот. Идентификация уникальна в обоих направлениях. Пример отображения 1:1 для элементов данных СТУДЕНТ и № БИЛЕТА (номер читательского билета).

Связь «один ко многим» (1:M) – под этим отношением подразумевается такой тип связи между элементами А и В, когда одному экземпляру элемента А соответствует 0,1 или несколько экземпляров элемента В. Однако каждому экземпляру элемента В соответствует только один экземпляр элемента А. Идентификация в прямом направлении не обязательно является уникальной. Однако в обратном направлении любой экземпляр элемента данных В, к которому направлена связь, идентифицирует один и только один экземпляр элемента данных А, от которого направлена связь. В примере (рис.24,25) элементы данных ГРУППА и СТУДЕНТ № связаны между собой отображением 1:M. В группе обучается много студентов, но каждый студент учится только в одной группе.

Связь «многие к одному» (M:1) – обратная по отношению к 1:M. Взаимосвязь между элементами данных является ассоциативной.

Связь «многие ко многим» (M:M) – с помощью отображения многие ко многим определяются такой тип связи, при которых каждому экземпляру А может соответствовать 0,1 или несколько экземпляров элемента В и наоборот. Связь, не идентифицирующая в обоих направлениях. Такими элементами данных являются ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА (рис.3.11,3.12).

Конкретный преподаватель может читать много дисциплин, и конкретная дисциплина может читаться многими преподавателями.

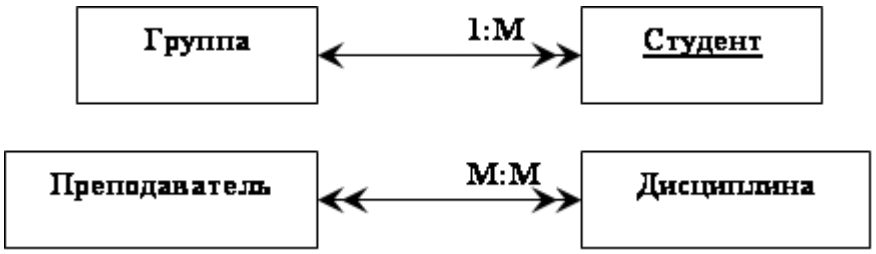


Рис.3.11– Отображение 1:M и M:M.

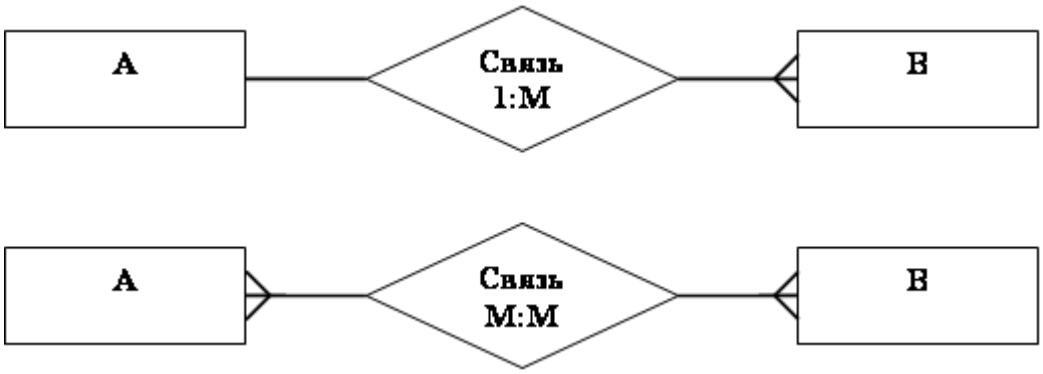


Рис.3.12– Отображение 1:M и M:M с использованием нотации Мартина.

Если каждый экземпляр сущности обязательно участвует в связи, то говорят, что сущность имеет **обязательный класс** принадлежности. Если экземпляры данной сущности могут не участвовать в связи, то класс принадлежности этой сущности является **необязательным.**

Примеры обязательного и необязательного классов принадлежности сущностей.

Рассмотрим примеры степени:

1. Связь «один-к-одному» (оба обязательных класса). Если рассматривать традиционный брак, то степень связи между сущностями *Мужчина* и *Женщина* будет 1:1. Кроме того, каждый мужчина, состоящий в браке, обязательно ДОЛЖЕН иметь жену. Таким образом, говорят, что сущность *Женщина* имеет обязательный класс принадлежности. И, наоборот, каждая женщина, состоящая в браке, ДОЛЖНА иметь мужа. То есть, сущность *Мужчина* также имеет обязательный класс принадлежности (рис.3.13).

Рис. 3.13– Связь 1:1, обязательные классы принадлежности.

2. Связь «один-к-одному» (один обязательный класс другой нет).

Сотрудник руководит отделом (рис. 3.14). Поскольку сотрудник может руководить только ОДНИМ отделом, а в отделе может быть только ОДИН руководитель, то степень связи в этом примере 1:1.

В каждом отделе ДОЛЖЕН быть руководитель, т.е. каждому экземпляру сущности *Отдел* ДОЛЖЕН соответствовать экземпляр сущности Сотрудник (сущность Сотрудник имеет обязательный класс принадлежности). С другой стороны, далеко не все сотрудники должны быть руководителями, т.е. сотрудник МОЖЕТ (но не должен) быть руководителем. Таким образом, есть экземпляры сущности *Сотрудник*, которым не соответствует ни один экземпляр сущности *Отдел* (необязательный класс принадлежности) (рис.3.14).

Рис. 3.14 – Связь 1:1, обязательный и необязательный классы принадлежности.

3. Связь «один-к-одному» (оба необязательных класса).

Человек читает книгу (рис. 3.15). Человек может читать сразу только ОДНУ книгу, а конкретная книга может быть читаема только ОДНИМ человеком, следовательно, степень связи 1:1. Человек МОЖЕТ читать книгу, а может ничего не читать. С другой стороны не каждая книга должна читаться, некоторые стоят на полке. Таким образом, обе сущности имеют необязательный класс принадлежности.

Рис. 3.15 – Связь 1:1, необязательные классы принадлежности.

4. Связь «один ко многим» (обязательные классы).

В процессе обучения студенты объединены в группы. Каждая группа может содержать множество студентов, а каждый студент может входить только в одну группу, т.е. степень связи 1:N (рис. 3.16). Каждая группа ДОЛЖНА содержать студентов, а каждый студент ДОЛЖЕН быть зачислен в конкретную группу, т.е. обе сущности имеют обязательные классы принадлежности.

Рис. 3.16 – Связь 1:N, обязательные классы принадлежности.

5. Связь «многие к одному» (один обязательный класс другой нет).

Поставщики продуктов имеют один юридический адрес, следовательно, ДОЛЖНЫ находиться в одном конкретном городе. А в одном городе МОГУТ находиться один, несколько или ни одного поставщика. То есть связь будет

N:1, сущность Города будет иметь обязательный, а сущность Поставщики – необязательный классы принадлежности (рис. 3.17).

Рис. 3.17– Связь N:1, обязательный и необязательный классы принадлежности.

Как уже указывалось выше, если существование сущности x зависит от существования сущности y, то x называется зависимой сущностью (иногда сущность x называют "слабой", а "сущность" y - сильной).

Степень связи для сильной сущности всегда будет 1 и обязательный класс принадлежности.

Класс принадлежности и степень связи для зависимой сущности могут быть любыми.

6. В магазине происходит продажа продуктов. Продукт не может быть продан, если его нет в магазине. Поэтому сущность *Продажи* является зависимой от сущности *Продукты* (рис. 3.18). Продукт МОЖЕТ быть продан в разные дни (а может быть вообще не продан), конкретная продажа связана только с одним продуктом. Таким образом, степень связи N:1, сущность *Продажи* имеет необязательный, а сущность *Продукты* - обязательный классы принадлежности (в самом деле, продажа без продукта теряет смысл).

Рис. 3.18 – Зависимая и независимая сущности.

7. Связь «многие-ко-многим».

Продукты в магазин поставляются поставщиками. Каждый продукт, имеющийся в магазине, ДОЛЖЕН быть поставлен одним или несколькими поставщиками, а каждый из поставщиков МОЖЕТ поставлять один или несколько продуктов или не поставлять ни одного. Т.е. степень связи M:N (рис. 3.19), а класс принадлежности для *Поставщиков* – обязательный, для *Продуктов* – необязательный.

Рис. 3.19 – Связь M:N, обязательный и необязательный классы принадлежности.

Между одними и теми же сущностями могут существовать несколько связей, например:

8. С одной стороны продукты в магазин поставляются заказчиками, с другой стороны, чтобы продукты были поставлены в магазин, необходимо заказать поставщикам необходимые продукты. Таким образом, между сущностями *Продукты* и *Поставщики* существуют связи «*Поставляют*» и

«*Заказаны*» (Рис. 3.20). Связь «*Поставляют*» рассмотрена в предыдущем примере. Рассмотрим подробнее связь «*Заказаны*». Каждый продукт ДОЛЖЕН быть заказан одному или нескольким поставщикам, каждый поставщик МОЖЕТ получить заказ на один или несколько продуктов или вообще не получить заказ.

9. Рассмотрим сущности *Врач* и *Пациент*. Пациент ДОЛЖЕН иметь одного лечащего врача, а врач МОЖЕТ лечить несколько пациентов. Кроме того, пациент МОЖЕТ иметь нескольких врачей-консультантов, а врач МОЖЕТ консультировать нескольких пациентов (Рис. 3.21).

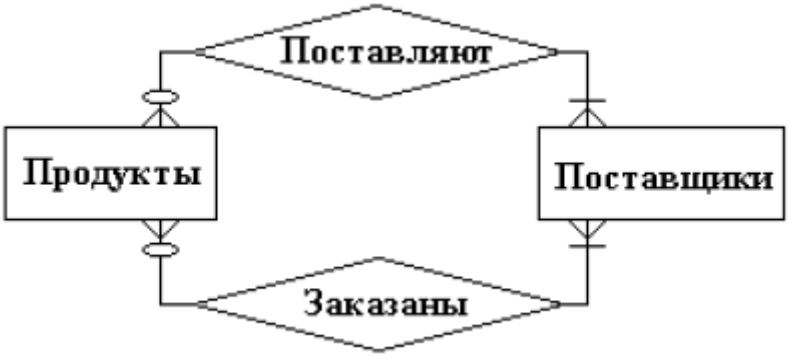


Рис. 3.20 – Несколько связей между двумя сущностями.

Рис. 3.21. Несколько связей между двумя сущностями.

3.2.2 Построение инфологической модели «сущность – связь»

Рассмотрим основные шаги, которые нужно выполнить для построения инфологической модели предметной области (модели «сущность – связь»).

1. Отбор документов и экспертных знаний, содержащих информацию об объектах, явлениях и процессах, имеющих место в предметной области.

Разработчик может не быть экспертом в рассматриваемой предметной области. Эксперт не обязан разбираться в принципах построения баз данных. Анализ документов и общение с экспертом помогают разработчику изучить «инфологический» аспект предметной области.

2. Выявление и определение сущностей выбранной предметной области.

Иногда сущности изначально ярко выражены. В некоторых случаях наличие сущности можно выявить, лишь изучив различные объекты и

установив, что они являются атрибутами одной и той же сущности (наличие которой изначально не было очевидным).

3. Выявление, определение и идентификация атрибутов сущностей.

Идентификатор должен гарантированно быть уникальным и по возможности не подвергаться изменениям. В роли идентификатора может выступать уже имеющийся атрибут или комбинация атрибутов, если они отвечают этим условиям. В противном случае должен быть добавлен специальный атрибут – идентификатор.

4. Определение наборов значений (доменов) для выявленных атрибутов.

Анализ доменов помогает обнаружить новые сущности, не выявленные ранее. Если домен достаточно специфичен, он может образовать новую сущность.

5. Выявление, описание и анализ связей между сущностями или атрибутами (классы принадлежности связей, а также атрибуты связей, если они необходимы).

6. Определение кардинальности связей между сущностями или атрибутами. Например, является ли некоторый набор связей отображением 1:n.

7. Объединение фрагментов в одну модель и организация данных в виде диаграммы "сущность-связь".

8. Преобразование и оптимизация модели. Уточнение модели и устранение выявленных противоречий.

Преобразование и оптимизация диаграммы сущность-связь подразумевает:

– устранение множественных атрибутов;
– преобразование *n*–арных связей в бинарные связи;
– перепроверка связей 1:1 и, возможно, их устранение;
– устранение связей типа M:M;
– устранение рекурсивных связей;
– устранение атрибутов из состава связи;
– дополнение ролевых имен степенями участия;

При необходимости проводят изменение статуса сущностей, атрибутов и связей. Например, преобразование сущности в атрибут или наоборот. При выявлении новых фактов – сущностей, атрибутов или связей повторяют уже проделанные действия ещё раз.

Пример построения модели «сущность – связь».

В качестве примера построим диаграмму, отображающую связь данных для подсистемы учета персонала предприятия.

1. Выделим интересующие нас сущности и связи.

Предприятие состоит из отделов, в которых работают сотрудники. Оклад каждого сотрудника зависит от занимаемой должности (инженер, ведущий инженер, бухгалтер, уборщик и т.д.). Предположим, что на предприятии допускается совместительство должностей, то есть каждый сотрудник может иметь более чем одну должность (и работать более чем в одном отделе), причем

может занимать неполную ставку. В то же время, одну и ту же должность могут занимать одновременно несколько сотрудников. В результате выделим следующие сущности, атрибуты и связи (рис. 3.22):

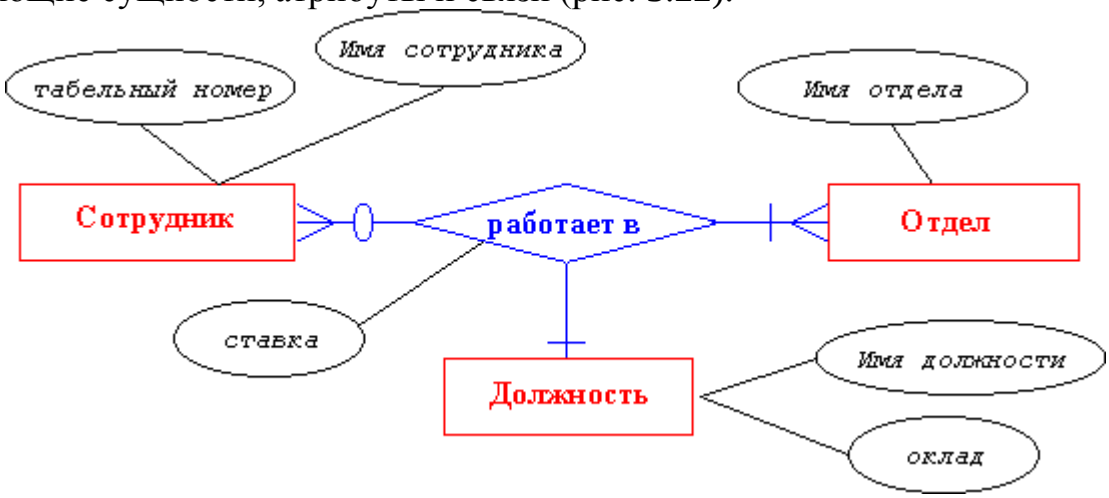


Рис. 3.22 – Выявление основных сущностей, атрибутов и связей предметной области.

ОТДЕЛ (ИМЯ_ОТДЕЛА),
СОТРУДНИК (ТАБЕЛЬНЫЙ_НОМЕР, ИМЯ СОТРУДНИКА),
ДОЛЖНОСТЬ (ИМЯ_ДОЛЖНОСТИ, ОКЛАД),

Сущности имеют связь РАБОТАЕТ_В с атрибутом СТАВКА между ними. Атрибут СТАВКА может принимать значения из интервала [0,1] (больше нуля, но меньше или равен единице), он определяет какую часть должностного оклада получает данный сотрудник.

В полученной модели связь реализуется на основе тренарной связи, которая, безусловно, несет полную информацию о предметной области. Действительно, она однозначно отображает тот факт, что оклад сотрудника зависит от его должности, отдела, где он работает, и ставки. Однако, в этом случае возникают некоторые проблемы с определением степени связи. Хотя, как было сказано, каждый работник может занимать несколько должностей, а в штате каждого отдела существуют вакансии с различными должностями. Тем не менее, класс принадлежности сущности ДОЛЖНОСТЬ (рис. 35) установлен в (1,1). Это объясняется тем, что ДОЛЖНОСТЬ ассоциируется фактически не с сущностями СОТРУДНИК и ОТДЕЛ, а со связью между ними.

Для оптимизации диаграммы следует устранить связь (1,1) сущности ДОЛЖНОСТЬ, убрать из связи РАБОТАЕТ_В атрибут СТАВКА, тренарную связь (каждый *n*-арный набор связей) заменить несколькими бинарными наборами.

Для оптимизации сущности СОТРУДНИК, ОТДЕЛ и связь РАБОТАЕТ_В агрегируются в некую новую абстрактную сущность, которая ассоциируется с сущностью ДОЛЖНОСТЬ с помощью связи степени n:1.

Отобразим ассоциации сотрудников, отделов и должностей с помощью бинарных связей (рис. 3.26).

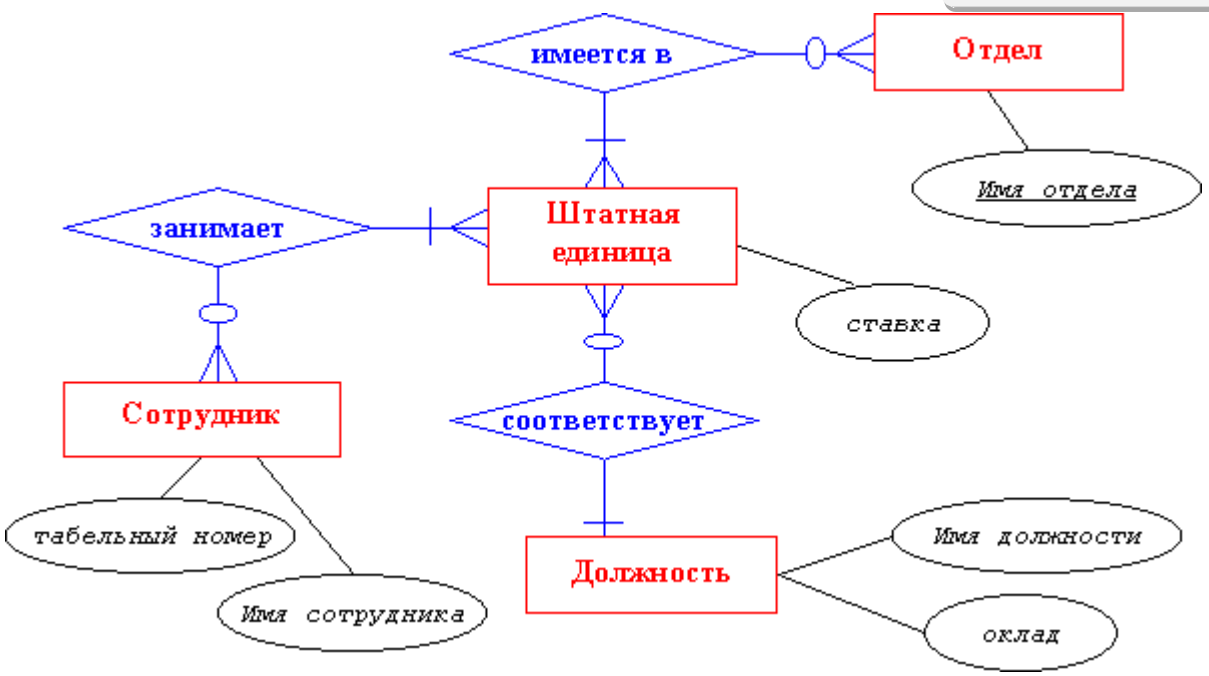


Рис. 3.26 – Замена тренарной связи бинарными связями (оптимизация модели).

В этом случае для адекватного описания семантики предметной области необходимо ввести еще одну сущность ШТАТНАЯ_ЕДИНИЦА, которая фактически заменяет собой связь РАБОТАЕТ_В в абстрактной сущности и поэтому имеет атрибут СТАВКА.

Переход от *n*-арной связи через агрегацию сущностей к набору бинарных связей можно рассматривать как последовательные этапы одного процесса, который приводит в дальнейшем к однозначному порождению реляционной модели данных. При построении диаграммы "сущность- связь" можно использовать любой из этих трех способов представления данных.

3.2.3 Даталогическое проектирование реляционной базы данных

Цель даталогического (логического) проектирования описание и построение схем связей между элементами данных, внешних схем, правил целостности безотносительно к их содержанию и среде хранения (определяется моделью данных иерархической, реляционной, сетевой).

Результат даталогического проектирования – представление логической структуры базы данных на языке описания конкретной СУБД. В логической структуре определяются все информационные единицы и связи между ними, типы данных и количественные характеристики, состав базы данных, исходные данные, имена, количество и структура файлов.

Глобальная даталогическая модель представляет собой отражение общего содержания баз данных, структурированное на логическом уровне и ориентированное на конкретную СУБД. СУБД накладывает количественные ограничения на логическую структуру базы данных. Прежде чем приступить к построению даталогической модели, необходимо детально изучить

особенности СУБД, уточнить ограничения, ознакомиться с правилами композиции структур более высокого уровня из составляющих их структур младшего уровня.

Даталогическое проектирование структурированных баз данных для различных систем имеет ряд общих моментов:

– минимальная логическая единица (элемент данных, поле и т.п.) семантически для всех систем одинакова и соответствует либо идентификатору объекта, либо его свойству;

– группировка элементов в структуры более высоких уровней и определение связей между ними производится в результате совместного анализа ограничений СУБД, особенностей предметной области и потребностей пользователей с учетом ограничений на ресурсы;

– совокупность типов объектов подлежащих отражению в базе данных и совокупность свойств, фиксируемых для конкретного объекта каждого типа должны быть заранее определены;

– при проектировании логической структуры базы данных присутствуют этапы преобразования исходной инфологической модели в модель допустимую для СУБД, и проверки адекватности полученной логической модели исходной модели;

– для каждой СУБД должен быть задан набор правил и приемов устранения аномальных с точки зрения системы ситуаций;

– отображение связей между элементами на уровне логической модели может выполняться либо путем совместного расположения взаимосвязанных элементов, либо путем объявления связи посредством дополнительного связующего звена. Последняя из перечисленных возможностей часто используются, чтобы «обойти» ограничения, накладываемые СУБД.

В процессе даталогического проектирования создается структура базы данных, соответствующая концептуальной схеме. Адекватность реализации концептуальной схемы определяется эмпирически и эвристически в ходе отладки и дальнейшей эксплуатации базы данных.

При даталогическом проектировании можно выделить последовательность процедур:

1. определение перечня отношений и типа зависимостей (связей);
2. определение перечня полей, типов полей, ключевых полей каждого отношения, установление связей между таблицами через внешние ключи;
3. определение или установление индексов для полей в таблицах;
4. разработка списков (словарей) для полей с перечислительным характером значений данных;
5. нормализация отношений, доработка перечня таблиц и их связей.
6. устранение противоречий между данными;
7. сокращение избыточности хранимых данных (разделение данных);
8. установление ограничений целостности по полям таблиц и связей.

Технологически процесс проектирования разделяют на процесс предварительного проектирования таблиц (отношений) и связей между ними и последующую нормализацию таблиц (см. раздел 3.2.4).

Поля таблиц (отношений) и связей между ними определяются на основе первоначально отработанных сущностей, атрибутов и связей концептуальной модели базы данных.

Для перехода от инфологической модели к реляционной даталогической выполняют операции по замене ER-диаграммы на описание атрибутов отношений. На основе анализа диаграмм "сущность-связь" формируются отношения проектируемой базы данных. При этом учитывается степень связи сущностей и класс их принадлежности, которые, в свою очередь, определяются на основе анализа диаграмм ER- экземпляров соответствующих сущностей.

При этом также следует учитывать, что реляционная модель организации данных по признаку множественности обеспечивает лишь два типа связей-отношений: «Один-ко-многим» и «Один-к-одному. Связи типа «Многие-ко-многим» реализуются через создание двух связей «Один-ко-многим», которые связывают исходные таблицы с третьей (общей) таблицей. Ключ связной таблицы состоит, по крайней мере, из двух полей, которые являются внешними ключами для связываемых отношений «Многие-ко-многим».

Предварительные отношения для бинарной связи могут быть получены путем просмотра нескольких логических альтернатив и выбора среди них наиболее подходящей. Перечень общих правил генерации отношений из диаграмм "сущность-связь" ER-типа можно получить, опираясь на класс принадлежности и степень отношения как на определяющие факторы.

Правило 1. Если степень бинарной связи равна 1:1 и класс принадлежности обеих сущностей является обязательным (рис.3.18), то требуется только одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей. Если степень связи равна 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным (рис.3.19), то одного отношения недостаточно. Так как будут пробелы на месте атрибутов тех экземпляров, которые не участвуют в связях.

Способ исключения пробелов состоит в использовании вместо одного отношения двух. Каждое отношение будет содержать информацию, касающуюся одной сущности. Кроме того, ключ сущности, класс принадлежности которой является необязательным, необходимо поместить в качестве атрибута в отношение, содержащее информацию о сущности, класс принадлежности которой является обязательным.

Правило 2. Если степень бинарной связи равна 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность необходимо выделение одного отношения, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме

того, ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

Воспользовавшись этим правилом можно получить следующие отношения:

S2(кл2, S2атр1, кл1),
S1(кл1, S1атр1, S1атр2).

В этом случае, когда степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательным (рис.3.19), одного отношения недостаточно. При использовании только одного отношения возможны два пути возникновения пробелов. Также недостаточным является использование двух отношений, так как возникают проблемы в связи с внесением ключа одной сущности в отношение, выделенное под другую сущность. Единственное решение заключается в выделении трех отношений: по одному для каждой сущности и одного для связи.

Правило 3. Если степень бинарной связи равна 1:1 и класс принадлежности ни одной сущности не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях, и одного для связи. Среди своих атрибутов отношение, выделяемое связи, будет иметь по одному ключу сущности от каждой сущности.

Предварительные отношения для бинарных связей степени 1:N.

Для случая бинарных связей степени 1:1 устанавливаются три отдельных правила генерации соответствующего набора предварительных отношений. Для случая бинарных связей степени 1:n требуется только два правила. Фактором, определяющим выбор и использование одного из этих двух правил, является класс принадлежности n-связной сущности; класс принадлежности 1-связной сущности не влияет на конечный результат в обоих случаях.

Рассмотрим пример связи, изображенной на рис.3.16.

Согласно диаграмме каждый экземпляр сущности ГРУППА может участвовать в связи только один раз, а экземпляры сущности СТУДЕНТ участвует в связи не менее одного раза. Следовательно, в отношении, посвященном связи ГРУППА и СТУДЕНТ, будут появляться пустые поля, где класс принадлежности сущности не обязательный, а в полях, отведенных для атрибута n-связной сущности будут повторы. Причем, повторы будут существовать и в том случае, если класс принадлежности n-связной сущности обязателен. Решить все эти проблемы, вне зависимости от класса принадлежности 1-связной сущности, можно согласно следующему правилу.

Правило 4. Если степень бинарной связи равна 1:n и класс принадлежности n-связной сущности является обязательным, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ сущности каждой сущности служит в качестве первичного

ключа для соответствующего отношения. Дополнительно ключ 1-связной сущности должен быть добавлен как атрибут в отношение, отводимое n-связной сущности.

Теперь рассмотрим случай необязательного класса принадлежности обеих сущностей.

В связи с этим возникает несколько проблем: пробелы в полях, отведенных тем экземплярам сущности, которые не участвуют в связи; повторяются поля в тех случаях, когда экземпляры КНИГА участвуют в связи более одного раза. Последняя проблема остается и в том случае, когда класс принадлежности односвязной сущности обязателен.

Если в этом случае применить правило 4 и сформировать 2 отношения, то не исчезнут пробелы в тех полях, которые отведены под ключ КНИГА, то есть у этой сущности необязательный класс принадлежности.

Решить все эти проблемы вне зависимости от класса принадлежности 1-связной сущности можно, следуя этому правилу.

Правило 5. Если степень бинарной связи равна 1:n и класс принадлежности n-связной сущности является необязательным, то необходимо формирование трех отношений: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одного отношения для связи. Связь должна иметь среди своих атрибутов ключ сущности от каждой сущности.

Предварительные отношения для бинарных связей степени M:N.

Если степень бинарной связи равна M:N, то для хранения данных требуются три отношения вне зависимости от класса принадлежности как первой, так и второй сущностей. При использовании одного или двух отношений неизбежно возникновение пробелов и/или повторяющихся групп данных в экземплярах этих отношений; какая из двух проблем возникает при использовании двух отношений зависит от классов принадлежности двух сущностей. Предлагается следующее правило генерации предварительных отношений для случая степени M:N.

Правило 6. Если степень бинарной связи равна M:N , то для хранения данных необходимо три отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее отношение должно иметь в числе своих атрибутов ключ сущности каждой сущности.

Алгоритм перехода от модели «сущность-связь» к реляционной модели.

Существует алгоритм однозначного преобразования модели «сущность-связь» в реляционную модель данных (т.е. осуществляется переход от инфологического моделирования к логическому проектированию схемы реляционной базы данных). Рассмотрим этот алгоритм.

1. Каждой сущности модели «сущность-связь» ставится в соответствие отношение реляционной модели. Каждая простая сущность превращается в таблицу. Имя сущности становится именем таблицы. При этом на имена отношений накладываются ограничения, присущие конкретной СУБД.

2. Каждый атрибут сущности становится атрибутом (полем) с тем же именем соответствующего отношения. На имена атрибутов отношения также накладываются ограничения выбранной СУБД. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (т.е. допустимость или недопустимость Null-значений). Поля, соответствующие необязательным атрибутам, могут содержать неопределенные значения; поля, соответствующие обязательным атрибутам, - не могут.

3. Первичный ключ сущности становится первичным ключом соответствующего отношения. Атрибуты, входящие в первичный ключ отношения, автоматически получают свойство отсутствия неопределенных значений (Not Null). Если имеется несколько возможных уникальных идентификаторов (ключей), выбирается наиболее используемый. Если в состав уникального идентификатора входят связи, к числу полей первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих полей используются имена концов связей и/или имена сущностей.

4. В каждое отношение, соответствующее сущности со стороны «многие» (связь 1:N), добавляется набор атрибутов сущности со стороны «один», являющихся первичным ключом сущности со стороны «один». В отношении, соответствующим сущности со стороны «многие», этот набор атрибутов становится внешним ключом.

5. Для моделирования необязательного класса принадлежности у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений. При обязательном классе принадлежности атрибуты получают свойство отсутствия неопределенных значений.

6. Разрешение связей типа M:N. Связи становится в соответствие отношение, имеющего атрибуты, которые в сущностях являются первичными ключами, а в новом отношении будут внешними ключами. Первичным ключом нового отношения будет совокупность внешних ключей.

В качестве примера преобразуем диаграмму «сущность – связь», отображающую связь данных для информационной системы учета продажи продуктов в магазине (рис. 3.27).

Модель имеет следующий список сущностей:

1. Продукты (КодПрод, Продукт, ЕдИзм, СрокХран(дней), УсловияХран).

2. Поставщики (КодПост, Поставщик, КодГорода, Адрес, ФИОдиректора, Телефон, Факс).

3. Продажи (ДатаПродажи, КодПрод, Количество, ЦенаПродажи).
4. Города(КодГорода, Город).

Рис. 3.27 –Диаграмма «сущность – связь» учета продажи продуктов в магазине.

1. В указанной модели мы имеем дело со следующими сущностями: **Продукты, Поставщики, Города, Продажи**. Следовательно, и в реляционной модели будут участвовать четыре отношения с такими же именами.

2. Далее задаются конкретные типы данных для каждого атрибута отношений (домены могли быть определены уже на этапе инфологического моделирования). В результате получаем отношения, приведенные на рис. 3.28.

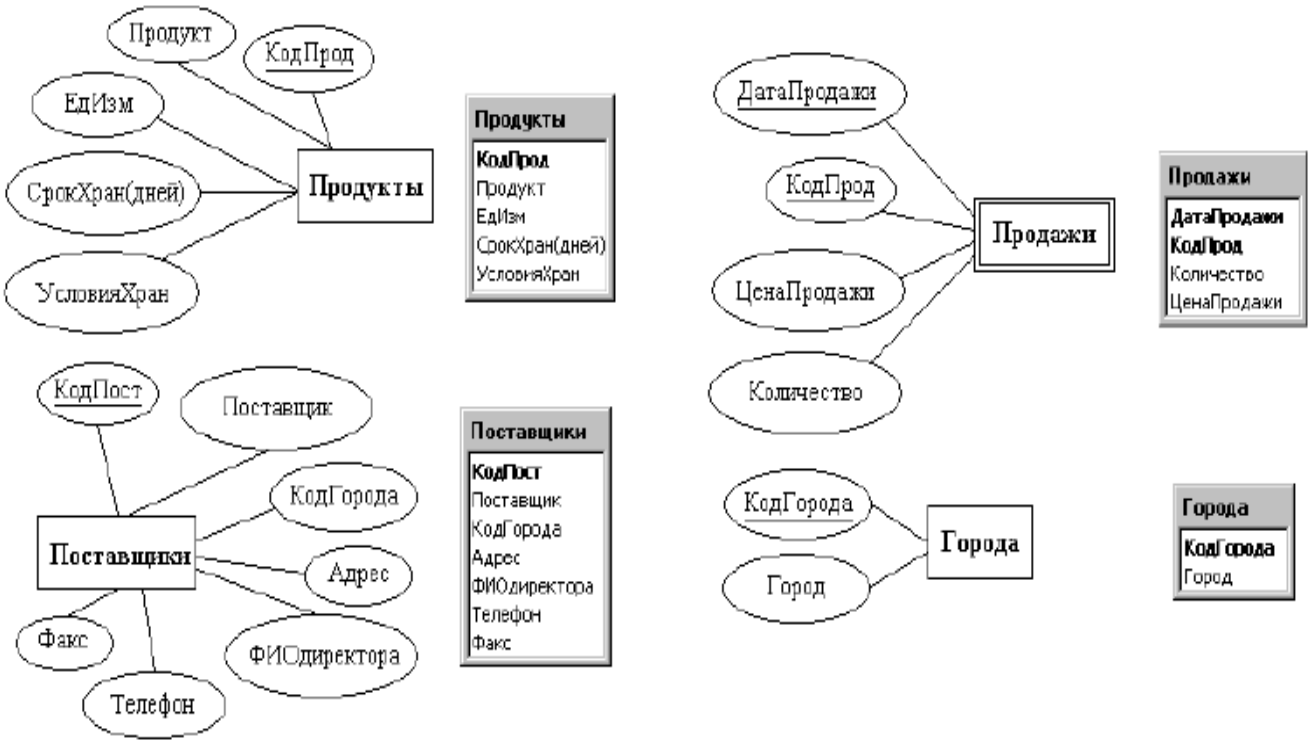


Рис. 3.28 – Переход от сущностей ER-модели к отношениям реляционной модели.

3. Степень связи между сущностями **Поставщики** и **Города** – N:1, поэтому первичный ключ *КодГорода* (сущности **Города**) должен войти в сущность **Поставщики** в качестве внешнего ключа (мы это сделали еще на этапе создания модели «сущность-связь»); степень связи между сущностями **Продажи** и **Продукты** – N:1, поэтому первичный ключ *КодПрод* (сущности **Продукты**) должен войти в сущность **Продажи** в качестве внешнего ключа (это сделано на этапе создания модели «сущность-связь»).

5. Для внешнего ключа *КодГорода* (отношение **Поставщики**) устанавливаем свойство допустимость Null-значений: «Да», т.к. в модели «сущность-связь» сущность **Поставщики** имела необязательный класс принадлежности; для внешнего ключа *КодПост* (отношение **Поставщики**) устанавливаем свойство допустимость Null-значений: «Нет», поскольку этот внешний ключ входит в состав первичного ключа.

6. В нашем примере две связи имеют степень M:N. Это связи **Поставляют** и **Заказаны**. Следовательно, дополнительно появляются еще два отношения **Поставки** и **Заказы** соответственно. Определяется описание и тип данных атрибутов этих отношений.

Окончательный вариант реляционной модели (схемы базы данных) приведен на рис. 3.29.

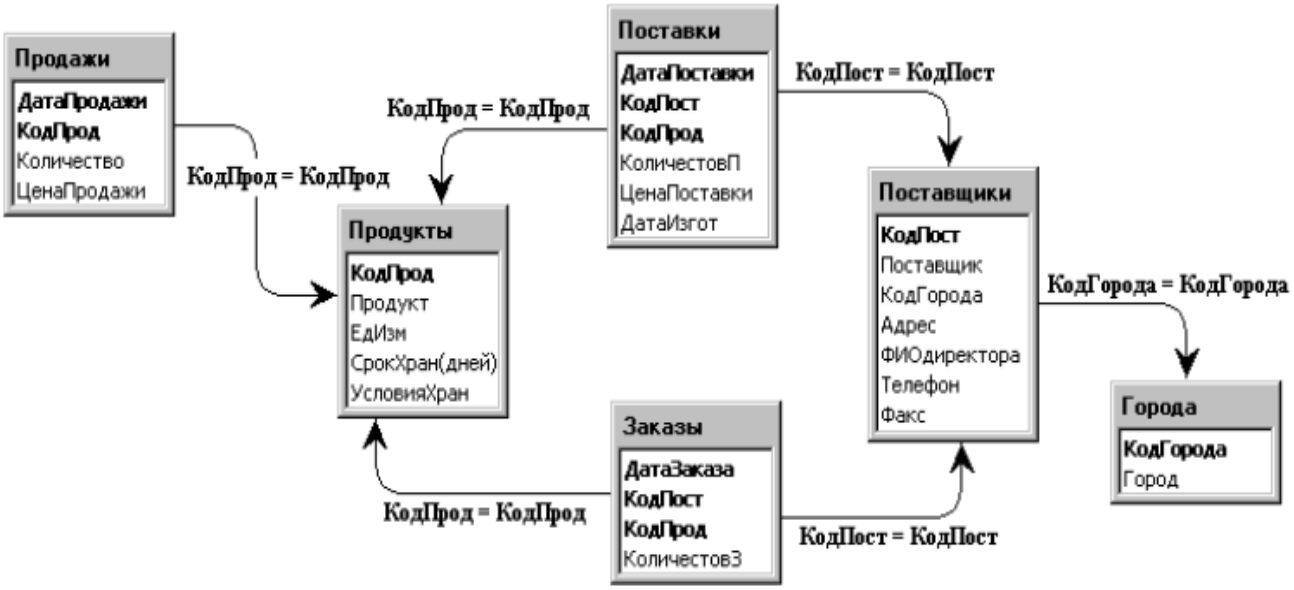


Рис. 3.29 – Реляционная модель данных учета продажи продуктов в магазине.

3.2.4 Проектирование реляционных баз данных на основе нормализации

На этапе инфологического моделирования была построена модель «сущность-связь», и с помощью алгоритма перехода к реляционной модели получены отношения, определены атрибуты, типы данных, ключевые поля и схема базы данных (рис. 3.29).

Полученное в результате инфологического проектирования исходное отношение называют **универсальными,** оно включают все данные и атрибуты.

Универсальное отношение может иметь избыточность данных, противоречивость данных и аномалии добавления, обновления и удаления данных.

Аномалии баз данных.

Если рассмотреть отношение ПРЕПОДАВАТЕЛЬ (рис. 3.30), то можно увидеть, что данные хранятся в ней с большой избыточностью.

Препод	Должн	Оклад	Предм	Учебник	Группа	ВидЗан
Иванов	преп	500	СУБД	Коннолли Т. Базы данных	123	Практ
Иванов	преп	800	СУБД	Коннолли Т. Базы данных	123	Лекция
Петров	ст.преп	500	СИ	Уэйт М. и др. Язык Си	256	Практ
Петров	ст.преп	800	СИ	Уэйт М. и др. Язык Си	256	Лекция
Сидоров	преп	500	Паскаль	Форсайт Р. Паскаль для всех	256	Лекция
Сидоров	преп	500	Паскаль	Форсайт Р. Паскаль для всех	256	Лекция
Егоров	преп	500	СУБД	Дьяков И.А. Базы данных	244	Лекция

Рис. 3.30 – Исходное отношение ПРЕПОДАВАТЕЛЬ.

Во многих строках повторяются фамилии преподавателей, должности, наименования предметов. Кроме того, в данном отношении хранятся вместе независимые друг от друга данные - и данные о преподавателях, и об окладах, и о группах. Пока никаких действий с отношением не производится, это не страшно. Но как только состояние предметной области изменяется, то, при попытках соответствующим образом изменить состояние базы данных, возникает большое количество проблем.

Исторически эти проблемы получили название аномалии обновления. Они возникают вследствие неадекватности модели данных предметной области, либо из-за трудности в реализации ограничений предметной области средствами СУБД.

Так как аномалии проявляют себя при выполнении операций, изменяющих состояние базы данных, то различают следующие виды аномалий:

Аномалии вставки (INSERT)

Аномалии обновления (UPDATE)

Аномалии удаления (DELETE)

В отношении ПРЕПОДАВАТЕЛЬ можно привести примеры следующих аномалий:

Аномалии вставки (добавления).

В отношение нельзя вставить данные об учебнике, который пока не используется в качестве учебного пособия. Действительно, если, например, появляется новый учебник, скажем, *Краморенко Базы данных*, то следует вставить в отношение кортеж (*null, null, null, null, Краморенко Базы данных, null, null*). Это сделать невозможно, т.к. атрибут *Препод.* (фамилия преподавателя) входит в состав потенциального ключа, и, следовательно, не может содержать null-значений.

Точно также нельзя вставить данные о предмете, который не ведет ни один преподаватель.

Причина аномалии - хранение в одном отношении разнородной информации (и о преподавателях, и о предметах, и об учебниках).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Аномалии обновления (UPDATE).

Фамилии преподавателей, наименования предметов, учебников, номера групп повторяются во многих кортежах отношения. Поэтому если преподаватель меняет фамилию, или предмет меняет наименование, или меняется номер группы, то такие изменения необходимо одновременно выполнить во всех местах, где эта фамилия, наименование или номер встречаются, иначе отношение станет некорректным (например, один и тот же предмет в разных кортежах будет называться по-разному). Таким образом, обновление базы данных одним действием реализовать невозможно. Для поддержания отношения в целостном состоянии необходимо написать триггер,

который при обновлении одной записи корректно исправлял бы данные и в других местах.

Причина аномалии - избыточность данных, также порожденная тем, что в одном отношении хранится разнородная информация.

Вывод - увеличивается сложность разработки базы данных. База данных, основанная на такой модели, будет работать правильно только при наличии дополнительного программного кода в виде триггеров.

Аномалии удаления (DELETE).

При удалении некоторых данных может произойти потеря другой информации. Например, если предмет СУБД удаляется из отношения, то следует удалить все строки, в которых он встречается, то будут потеряны данные о преподавателе Егорове. Если удалить преподавателя Иванова, то будет потеряна информация об учебнике Коннолли и о группе 123.

Причина аномалии - хранение в одном отношении разнородной информации (и о преподавателях, и о предметах, и об учебниках и о группах).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Функциональные зависимости и нормальные формы.

Для продолжения процесса проектирования необходимо проверить полученную реляционную модель на отсутствие противоречий в данных, отсутствие избыточных функциональных зависимостей между атрибутами и аномалий. При этом состав атрибутов отношений модели должен обеспечивать минимальное дублирование и непротиворечивость данных, обеспечивать их добавление, обновление и удаление.

Удовлетворение этих требований достигается нормализацией отношений базы данных. **Нормализация отношений** - это пошаговый обратимый процесс декомпозиции (разложения) исходных отношений базы данных на другие, более мелкие и простые отношения. При этом устанавливаются все возможные функциональные зависимости. Нормализация позволяет устранить недостатки проектирования и аномалии реляционной модели.

Нормализация особенно актуальна для больших проектов, содержащих множество сущностей с большим числом взаимосвязей.

Для проверки правильности разработанной реляционной модели применяют так называемые нормальные формы – правила, которым должны отвечать отношения (таблицы).

Нормальные формы пронумерованы и имеют вложенную взаимосвязь (рис. 3.31).

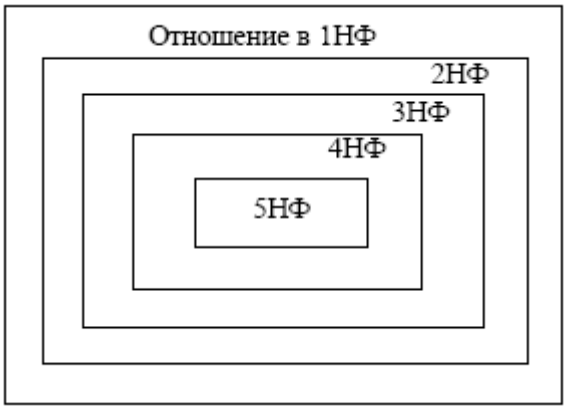


Рис. 3.31 – Нормальные формы.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

–первая нормальная форма (1НФ);

–вторая нормальная форма (2 НФ);

–третья нормальная форма (3 НФ);

–нормальная форма Бойса-Кодда (НФБК);

–четвертая нормальная форма (4 НФ);

–пятая нормальная форма, или нормальная форма проекции-соединения (5 НФ).

Основные свойства нормальных форм:

–каждая следующая нормальная форма в некотором смысле лучше предыдущей;

–при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

В основе процесса нормализации лежит декомпозиция отношения, находящегося в предыдущей нормальной форме, в два или более отношения, удовлетворяющих требованиям следующей нормальной формы.

Метод нормальных форм является классическим методом проектирования реляционных баз данных. Этот метод основан на фундаментальном в теории реляционных баз данных понятии функциональной зависимости между атрибутами отношений.

В процессе нормализации рассматриваются различные функциональные зависимости. Функциональные зависимости определяют не текущее состояние базы данных, а все возможные ее состояния, то есть они отражают те связи между атрибутами, которые присущи реальному объекту, моделируемые в базе данных.

Функциональные зависимости.

Функциональная зависимость. Атрибут *Y* некоторого отношения функционально зависит от атрибута *X* (атрибуты могут быть составными), если в любой момент времени каждому значению *X* соответствует одно значение *Y*. Функциональная зависимость обозначается $X \rightarrow Y$.

Это означает, что во всех кортежах с одинаковым значением атрибута *X*, атрибут *Y* будет иметь также одно и то же значение. Отметим, что *X* и *Y* могут быть составными - состоять из двух и более атрибутов.

В отношении на рис. 3.30 можно выделить функциональные зависимости между атрибутами ПРЕПОД→КАФ, ПРЕПОД→ДОЛЖН, ДОЛЖН→ОКЛАД и другие. Наличие функциональной зависимости в отношении определяется природой вещей, информация о которых представлена кортежами отношения. В отношении на рис. 40 ключевой атрибут является составным и состоит из атрибутов ПРЕПОД, ПРЕДМЕТ, ГРУППА.

Избыточная функциональная зависимость – это зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей, имеющихся в базе данных.

Частичной функциональной зависимостью называется зависимость не ключевого атрибута от части составного ключа. В рассматриваемом отношении атрибут ДОЛЖН находится в функциональной зависимости от атрибута ПРЕПОД, являющегося частью ключа. Тем самым атрибут ДОЛЖН находится в частичной зависимости от ключа отношения.

Полная функциональная зависимость. Неключевой атрибут функционально полно зависит от составного ключа если он функционально зависит от всего ключа в целом, но не находится в функциональной зависимости от какого-либо из входящих в него атрибутов. В рассматриваемом отношении атрибут ВИДЗАН находится в полной функциональной зависимости от составного ключа.

Транзитивная функциональная зависимость. Пусть *X, Y, Z* – три атрибута некоторого отношения. При этом $X \rightarrow Y$ и $Y \rightarrow Z$, но обратное соответствие отсутствует, т.е. $Z \nrightarrow Y$ *и* $Y \nrightarrow X$.

Тогда *Z* транзитивно зависит от *X*.

В отношении на рис.3.30 транзитивной зависимостью связаны атрибуты:

ПРЕПОД→ДОЛЖН→ОКЛАД

Между атрибутами может иметь место многозначная зависимость.

Многозначная зависимость. Пусть *X, Y, Z* – три атрибута отношения R. В отношении R существует многозначная зависимость $R.X \rightarrow\rightarrow R.Y$ только в том случае, если множество значений *Y*, соответствующее паре значений *X* и *Z*, зависит только от *X* и не зависит от *Z*.

Многозначные зависимости могут быть «один ко многим» (1:M), «многие к одному» (M: 1) или «многие ко многим» (M:M).

Например, пусть преподаватель ведет несколько предметов, а каждый предмет может вестись несколькими преподавателями, тогда имеет место

зависимость ПРЕПОД↔ПРЕДМЕТ. Так в рассматриваемом отношении (рис.3.30) видно, что преподаватель Иванов ведет практические занятия и лекции, а дисциплина СУБД – читается двумя преподавателями: Ивановым и Егоровым.

Другая многозначная зависимость ПРЕДМЕТ↔УЧЕБНИК, при изучении СУБД используются разные учебники. При этом ПРЕПОД и УЧЕБНИК не связны функциональной зависимостью, что приводит к появлению избыточности (для добавления еще одного учебника придется ввести в таблицу две новых строки). Дело улучшается при замене этой таблицы на две: (Предмет–Преподаватель и Предмет–Учебник).

Нормальные формы.

Нормализацией называется процесс разделения отношения на два или более отношений, в которых каждый факт появляется лишь в одном месте, с целью ликвидации дублирования данных, их противоречивости, удаления многозначных зависимостей и при этом в любом отношении должна быть функциональная зависимость вида $K \rightarrow F$, где K – первичный ключ, F – некоторый атрибут (поле таблицы). Цель нормализации состоит в приведения исходных(универсальных) отношений к нормальной форме, то есть в удалении «лишних» функциональных зависимостей.

Отношение находится в первой нормальной форме (1НФ) тогда и только тогда, когда значения всех его атрибутов атомарны, то есть каждая ее строка содержит только одно значение для каждого поля (атрибута) и ни одно из ее ключевых атрибутов не пусто. При этом каждый атрибут отношения неделим и не содержит повторяющихся групп.

Отношения в 1НФ часто называются просто нормализованными отношениями. Под атомарностью понимается степень структурирования и детализации информации в базе данных.

Глубина структурирования определяется практической необходимостью при манипулировании данными. Примером является глубина структурирования адреса. Можно хранить в одном поле весь адрес (*город, улица, дом, квартира*). Данный атрибут будет атомарным, если нет необходимости манипулировать отдельными городами или улицами, в противном случае этот атрибут не является атомарным и необходимо его дальнейшее разбиение на отдельные атрибуты (*город*), (*улица, дом, квартира*).

Отношение находится во второй нормальной форме (2НФ), если оно удовлетворяет определению 1НФ и все его атрибуты, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Отношение находится в третьей нормальной форме (3НФ), если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Можно также утверждать, что отношение находится в третьей нормальной форме, если оно удовлетворяет определению 2НФ и не одно из его

неключевых атрибутов не зависит функционально от любого другого неключевого атрибута.

Отношение находится в нормальной форме Бойса-Кодда (НФБК), если и только если любая функциональная зависимость между его атрибутами сводится к полной функциональной зависимости от возможного ключа (если в отношении несколько первичных ключей).

Ситуация, когда отношение будет находиться в 3НФ, но не в нормальной форме Бойса-Кодда (НФБК), возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут. Заметим, что на практике такая ситуация встречается достаточно редко, для всех прочих отношений 3НФ и НФБК эквивалентны.

Для определения 4 и 5 нормальных форм необходимо понятие декомпозиции.

Полной декомпозицией отношения называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым отношения.

То есть если после выполнения нескольких операций *Проекция* с каким-либо исходным отношением *R*, получают отношения *S, D, F*, тогда эти отношения являются полными декомпозициями отношения *R*. Или можно сказать, что результатом выполнения операции естественного соединения отношений *S, D, F* будет отношение *R*.

Отношение находится в четвертой нормальной форме, когда полная декомпозиция должна быть соединением ровно двух проекций. Или отношение находится в 4НФ, если в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями. Или отношение *R* находится в 4НФ, если в случае существования многозначной зависимости *A* ->> *B* все остальные атрибуты *R* функционально зависят от *A*.

Отношение находится в пятой нормальной форме тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный ключ. Отношение, не имеющее ни одной полной декомпозиции, также находится в 5НФ.

Нормализация отношений.

Для проведения нормализации отношений следует пользоваться следующими правилами:

Первое правило. Если отношение *R* имеет составной первичный ключ вида *K1, K2*, и включает также атрибут *F*, который функционально зависит от части этого ключа, например, от *K2*, но не от полного ключа. В этом случае следует сформировать другое отношение, содержащее атрибуты *K2* и *F* (первичный ключ – *K2*), и удалить *F* из первоначального отношения. То есть вместо исходного отношения *R(K,F1,F2)*, где первичные ключи *K1,K2 и K2→F*,

создается два отношения *S1(K1,K2)*, где первичные ключи *K1,K2* и *S2(K2,F)* с первичным ключом *K2*.

Второе правило. Если отношение *R* имеет первичный (возможный) ключ *K* и не являющееся возможным ключом атрибут *F1*, который функционально зависит от *K*, и другой неключевой атрибут *F2*, который функционально зависит от *F1*. В этом случае формируется другое отношение, содержащее атрибуты *F1* и *F2*, с первичным ключом *F1*, а *F2* удаляется из первоначального отношения. То есть вместо исходного отношения *R(K,F1,F2)* с первичным ключом *K* и $F1 \rightarrow F2$, создаются два отношения *S1(K,F1)*, где первичный ключ *K* и *S2(F1,F2)* с первичным ключом *F1*.

Третье правило. Если имеется отношение *R(K1, K2, K3, F4)*, находящееся в 3НФ, где *K1, K2* – возможный ключ, *K2, K3* – возможный ключ, *F4* – неключевой атрибут отношения *R*, и имеются функциональные зависимости:

$K1 \rightarrow K3;$

$K3 \rightarrow K1;$

$K1, K2 \rightarrow F4;$

$K2, K3 \rightarrow F4.$

Для приведения отношения *R* к НФБК, это отношение декомпозируется на два отношения:

S1(K1, K3) и *S2(K1, K2, F4)*

или *S1(K3, K1)* и *S2(K2, K3, F4)*.

Четвертое правило. Если имеется отношение *R(K1, A2, A3)*, находящееся в НФБК и имеются функциональные зависимости:

–зависимость множества значений атрибута *A2* от множества значений атрибута *K1* ($K1 \rightarrow\rightarrow A2$);

–зависимость множества значений атрибута *A3* от множества значений ключевого атрибута *K1* ($K1 \rightarrow\rightarrow A3$)

Для приведения отношения R к 4НФ, это отношение декомпозируется на два отношения:

S1(K1, A2) и *S2(K1, A3)*.

Пятое правило. Если имеется отношение *R(K1, K2, K3)*, находящееся в 4НФ, где *K1, K2, K3* – составной первичный ключ, и имеется зависимость соединения: *({K1, K2}, {K1, K3}, {K2, K3})*.

Для приведения отношения *R* к 5НФ, это отношение декомпозируется на три отношения:

S1(K1, K2), S2(K1, K3) и *S3(K2, K3)*.

В общем случае необходимо проводить нормализацию к пятой нормальной форме (5НФ). На практике отношение считается нормализованной, если оно находится минимум в третьей нормальной форме или в нормальной форме Бойса–Кодда.

Рассмотрим пример нормализации универсального отношения СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ представленного на рис. 3.32:

СОТР_НОМЕР (номер сотрудника)	**СОТР_ЗАРП (зарплата сотрудника)**	**ОТД_НОМЕР (номер отдела)**	**ПРО_НОМЕР (номер проекта)**	**СОТР_ЗАДАН (задание сотрудника по проекту)**
1	1000	1	1	Проект
2	1000			Расчет
3	2000	2		Отчет
4	2000		2	Проект
5	3000	3		Расчет
6	3000			Отчет

Рис. 3.32 – Универсальное отношение СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ.

В заданном универсальном отношении атрибуты ОТД_НОМЕР и ПРО_НОМЕР имеют одно значение для нескольких строк, соответственно отношение необходимо привести в 1 НФ, в которой все значения атомарны, когда в каждой позиции пересечения столбца и строки отношения расположено в точности одно значение (рис. 3.33).

СОТР_НОМЕР	**СОТР_ЗАРП**	**ОТД_НОМЕР**	**ПРО_НОМЕР**	**СОТР_ЗАДАН**
1	1000	1	1	Проект
2	1000	1	1	Расчет
3	2000	2	1	Отчет
4	2000	2	2	Проект
5	3000	3	2	Расчет
6	3000	3	2	Отчет

Рис. 3.33 – Отношение СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ в первой нормальной форме.

В отношении (рис. 3.33) первичный ключ:
СОТР_НОМЕР, ПРО_НОМЕР
Функциональные зависимости:
СОТР_НОМЕР → СОТР_ЗАРП
СОТР_НОМЕР → ОТД_НОМЕР
ОТД_НОМЕР → СОТР_ЗАРП
СОТР_НОМЕР, ПРО_НОМЕР → СОТР_ЗАДАН

Несмотря на то, что первичным ключом является составной атрибут СОТР_НОМЕР, ПРО_НОМЕР, атрибуты СОТР_ЗАРП и ОТД_НОМЕР функционально зависят от части первичного ключа, атрибута СОТР_НОМЕР.

В отношении (рис. 3.33) могут быть выявлены наиболее типичные проблемы, возникающие при использовании единственного отношения. В качестве наиболее типичных можно отметить аномалии вставки, удаления и

обновления. Так нельзя вставить в отношение кортеж, описывающий сотрудника, который еще не выполняет никакого проекта (первичный ключ не может содержать неопределенное значение). При удалении кортежа разрушается связь данного сотрудника с данным проектом, и утрачивается информация о том, что он работает в некотором отделе. При переводе сотрудника в другой отдел необходимо будет модифицировать все кортежи, описывающие этого сотрудника, или получится несогласованный результат. Выявленные проблемы отношения устраняются путем его дальнейшей нормализации.

Используя первое правило нормализации отношений разделим отношение СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ на два отношения СОТРУДНИКИ-ОТДЕЛЫ и СОТРУДНИКИ-ПРОЕКТЫ (рис. 3.34 и 3.35):

СОТР_НОМЕР	СОТР_ЗАРП	ОТД_НОМЕР
1	1000	1
2	1000	1
3	2000	2
4	2000	2
5	3000	3
6	3000	3

Рис. 3.34 – Отношение СОТРУДНИКИ-ОТДЕЛЫ во второй нормальной форме.

Первичный ключ:
СОТР_НОМЕР
Функциональные зависимости:
СОТР_НОМЕР → СОТР_ЗАРП
СОТР_НОМЕР → ОТД_НОМЕР
ОТД_НОМЕР → СОТР_ЗАРП

СОТР_НОМЕР	ПРО_НОМЕР	СОТР_ЗАДАН
1	1	Проект
2	1	Расчет
3	1	Отчет
4	2	Проект
5	2	Расчет
6	2	Отчет

Рис. 3.35 – Отношение СОТРУДНИКИ-ПРОЕКТЫ во второй нормальной форме.

Первичный ключ:
СОТР_НОМЕР, ПРО_НОМЕР
Функциональные зависимости:
СОТР_НОМЕР, ПРО_НОМЕР → СОТР_ЗАДАН

В результате нормализации каждое из этих двух отношений находится в 2НФ, и в них устранены отмеченные выше аномалии (легко проверить, что все указанные операции выполняются без проблем). При этом в каждом отношении, каждый неключевой атрибут полностью зависит от первичного ключа, а отношения будут иметь связь один-ко-многим по атрибуту СОТР_НОМЕР.

Рассмотрим еще раз отношение СОТРУДНИКИ-ОТДЕЛЫ, находящееся в 2НФ. Заметим, что функциональная зависимость СОТР_НОМЕР → СОТР_ЗАРП является транзитивной; она является следствием функциональных зависимостей СОТР_НОМЕР → ОТД_НОМЕР и ОТД_НОМЕР → СОТР_ЗАРП. Другими словами, заработная плата сотрудника на самом деле является характеристикой не сотрудника, а отдела, в котором он работает.

То есть нельзя занести в базу данных информацию, характеризующую заработную плату отдела, до тех пор, пока в этом отделе не появится хотя бы один сотрудник (первичный ключ не может содержать неопределенное значение). При удалении кортежа, описывающего последнего сотрудника данного отдела, удаляется информация о заработной плате отдела. Чтобы согласованным образом изменить заработную плату отдела, требуется предварительно найти все кортежи, описывающие сотрудников этого отдела. То есть в отношении СОТРУДИКИ-ОТДЕЛЫ по-прежнему существуют аномалии. Их можно устранить путем дальнейшей нормализации.

Используя второе правило нормализации отношений, удалим транзитивную зависимость разделив отношение СОТРУДНИКИ-ОТДЕЛЫ на два отношения СОТРУДНИКИ и ОТДЕЛЫ (рис. 3.36, 3.37):

СОТР_НОМЕР	ОТД_НОМЕР
1	1
2	1
3	2
4	2
5	3
6	3

Рис. 3.36 – Отношение СОТРУДНИКИ в третьей нормальной форме.

Первичный ключ:
СОТР_НОМЕР
Функциональные зависимости:
СОТР_НОМЕР →ОТД_НОМЕР

ОТД_НОМЕР	СОТР_ЗАРП
1	1000
2	2000
3	3000

Рис. 3.37 – Отношение СОТРУДНИКИ-ОТДЕЛЫ во второй нормальной форме.

ОТДЕЛЫ (ОТД_НОМЕР, СОТР_ЗАРП)
Первичный ключ:
ОТД_НОМЕР
Функциональные зависимости:
ОТД_НОМЕР →СОТР_ЗАРП

В результате нормализации каждое из этих двух отношений находится в 3НФ, свободно от отмеченных аномалий. Отношения будут иметь связь многие-к-одному по атрибуту ОТД_НОМЕР.

Рассмотрим пример отношения СОТРУДНИКИ-ПРОЕКТЫ рис. 3.38.

СОТР_НОМЕР	**СОТР_ФИО**	**ПРО_НОМЕР**	**СОТР_ЗАДАН**
1	Иванов	1	Проект
2	Петров	1	Расчет
3	Сидоров	1	Отчет
4	Смирнов	2	Проект
5	Федоров	2	Расчет
6	Волков	2	Отчет

Рис. 3.38 – Отношение СОТРУДНИКИ-ПРОЕКТЫ в третьей нормальной форме.

Возможные ключи:
СОТР_НОМЕР, ПРО_НОМЕР,
СОТР_ФИО, ПРО_НОМЕР.
Функциональные зависимости:
СОТР_НОМЕР →СОТР_ФИО,
СОТР_НОМЕР → ПРО_НОМЕР,
СОТР_ФИО → СОТР_НОМЕР,
СОТР_ФИО → ПРО_НОМЕР,
СОТР_НОМЕР, ПРО_НОМЕР → СОТР_ЗАДАН,
СОТР_ФИО, ПРО_НОМЕР→ СОТР_ЗАДАН,

В этом примере предполагается, что личность сотрудника полностью определяется как его номером, так и фамилией.

В соответствии с определением отношение СОТРУДНИКИ-ПРОЕКТЫ находится в 3НФ. Однако тот факт, что имеются функциональные зависимости атрибутов отношения от атрибута, являющегося частью первичного ключа, приводит к аномалиям. Например, для того, чтобы изменить имя сотрудника с данным номером, согласованным образом, потребуется модифицировать все кортежи, включающие его номер.

Используя третье правило нормализации отношений, для устранения указанной аномалии, и преобразования отношения из 3НФ в НФБК разделим отношение СОТРУДНИКИ-ПРОЕКТЫ на два. отношения СОТРУДНИКИ и СОТРУДНИКИ-ПРОЕКТЫ (рис. 3.39, 3.40):

СОТР_НОМЕР	СОТР_ФИО
1	Иванов
2	Петров
3	Сидоров
4	Смирнов
5	Федоров
6	Волков

Рис. 3.39 – Отношение СОТРУДНИКИ в нормальной форме Бойса-Кодда.

Возможные ключи:

СОТР_НОМЕР,

СОТР_ФИО.

Функциональные зависимости:

СОТР_НОМЕР → СОТР_ФИО,

СОТР_ФИО → СОТР_НОМЕР.

СОТР_НОМЕР	ПРО_НОМЕР	СОТР_ЗАДАН
1	1	Проект
2	1	Расчет
3	1	Отчет
4	2	Проект
5	2	Расчет
6	2	Отчет

Рис. 3.40 – Отношение СОТРУДНИКИ-ПРОЕКТЫ в нормальной форме Бойса-Кодда.

Возможный ключ:

СОТР_НОМЕР, ПРО_НОМЕР.

Функциональные зависимости:

СОТР_НОМЕР, ПРО_НОМЕР → СОТР_ЗАДАН.

Возможна альтернативная декомпозиция, если выбрать за основу СОТР_ФИО. В обоих случаях получаемые отношения СОТРУДНИКИ и СОТРУДНИКИ-ПРОЕКТЫ находятся в НФБК, и им не свойственны отмеченные аномалии.

Рассмотрим пример отношения ПРОЕКТЫ (ПРО_НОМЕР, ПРО_СОТР, ПРО_ЗАДАН). Отношение ПРОЕКТЫ содержит номера проектов, для каждого проекта список сотрудников, которые могут выполнять проект, и список заданий, предусматриваемых проектом. Сотрудники могут участвовать в нескольких проектах, и разные проекты могут включать одинаковые задания.

Тогда в отношении ПРОЕКТЫ существуют следующие две многозначные зависимости:

ПРО_НОМЕР →→ ПРО_СОТР,

ПРО_НОМЕР →→ ПРО_ЗАДАН.

Каждый кортеж отношения связывает некоторый проект с сотрудником, участвующим в этом проекте, и заданием, который сотрудник выполняет в рамках данного проекта (предполагается, что любой сотрудник, участвующий в проекте, выполняет все задания, предусмотренные этим проектом). По причине сформулированных выше условий единственным возможным ключом отношения является составной атрибут ПРО_НОМЕР, ПРО_СОТР, ПРО_ЗАДАН, и нет никаких других детерминантов. Следовательно, отношение ПРОЕКТЫ находится в НФБК. Но при этом оно обладает недостатками: если, например, некоторый сотрудник присоединяется к данному проекту, необходимо вставить в отношение ПРОЕКТЫ столько кортежей, сколько заданий в нем предусмотрено.

Используя четвертое правило нормализации отношений, для устранения указанной аномалии, и преобразования отношения из НФБК в 4НФ проведем декомпозицию отношения ПРОЕКТЫ в два отношения ПРОЕКТЫ-СОТРУДНИКИ и ПРОЕКТЫ-ЗАДАНИЯ:

ПРОЕКТЫ-СОТРУДНИКИ (ПРО_НОМЕР, ПРО_СОТР),

ПРОЕКТЫ-ЗАДАНИЯ (ПРО_НОМЕР, ПРО_ЗАДАН).

В результате нормализации отношения будут находится в 4НФ и свободны от отмеченных аномалий.

Во всех рассмотренных до этого момента нормализациях производилась декомпозиция одного отношения в два. Иногда это сделать не удается, но согласно пятому правилу нормализации, возможна декомпозиция в большее число отношений, каждое из которых обладает лучшими свойствами.

Рассмотрим, например, отношение СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ (рис. 3.41).

СОТР_НОМЕР	ОТД_НОМЕР	ПРО_НОМЕР
1	1	1
2	1	1
3	2	1
4	2	2
5	3	2
6	3	2

Рис. 3.41 – Отношение СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ в 4 нормальной форме.

Предположим, что один и тот же сотрудник может работать в нескольких отделах и работать в каждом отделе над несколькими проектами. Первичным ключом этого отношения является полная совокупность его атрибутов, отсутствуют функциональные и многозначные зависимости.

Поэтому отношение находится в 4НФ. Однако в нем могут существовать аномалии, которые можно устранить путем декомпозиции в три отношения.

Для отношения СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ введем следующие имена составных атрибутов:

СО = {СОТР_НОМЕР, ОТД_НОМЕР},

СП = {СОТР_НОМЕР, ПРО_НОМЕР},
ОП = {ОТД_НОМЕР, ПРО_НОМЕР}.

Предположим, что в отношении СОТРУДНИКИ-ОТДЕЛЫ-ПРОЕКТЫ существует зависимость соединения:

(СО, СП, ОП).

На примерах легко показать, что при вставках и удалениях кортежей могут возникнуть проблемы. Их можно устранить путем декомпозиции исходного отношения, согласно пятому правилу нормализации отношений, в три новых отношения:

СОТРУДНИКИ-ОТДЕЛЫ (СОТР_НОМЕР, ОТД_НОМЕР),
СОТРУДНИКИ-ПРОЕКТЫ (СОТР_НОМЕР, ПРО_НОМЕР),
ОТДЕЛЫ-ПРОЕКТЫ (ОТД_НОМЕР, ПРО_НОМЕР).

Пятая нормальная форма - это последняя нормальная форма, которую можно получить путем декомпозиции. Ее условия достаточно нетривиальны. 5НФ редко используется на практике. Очень тяжело определить само наличие зависимостей «проекции-соединения», потому что утверждение о наличии такой зависимости делается для всех возможных состояний БД, а не только для текущего экземпляра отношения R. Следует заметить, что зависимость соединения является обобщением как многозначной зависимости, так и функциональной зависимости.

3.3 Физическое проектирование базы данных

Физическое проектирование – описание и построение схем хранения данных для определенной СУБД (отображение датологической модели в модели данных СУБД).

На этапе физического проектирования базы данных, как уже указывалось выше, прежде всего, необходимо выбрать конкретную целевую СУБД. Физическое проектирование неразрывно связанно с конкретной СУБД. Между внутренним и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных. В случаи реляционной модели данных под этим подразумевается следующие:

–создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной модели данных.

–определение конкретных структур хранения данных и методов доступа к ним, обеспечение оптимальной производительности системы с базой данных.

–разработка средств защиты создаваемой системы.

На этом этапе осуществляется выбор типа носителя, способ организации данных во внешней памяти, методов доступа к данным, определение

параметров физического блока, управление работой памяти, считывание данных и т.д. (на основе ЯОД и ЯМД, SQL, конкретной СУБД).

К физическим (внутренним) моделям данных предъявляется ряд требований, основными из которых являются:

–сохранение семантики логической организации данных;

–максимальная экономия внешней памяти;

–минимальные затраты на ведение баз данных;

–максимальное быстродействие при поиске и выборке данных (минимальное время ответа системы на запрос).

В конечном счете любая конкретная СУБД основывается на конкретном комплексном решении. В данном разделе будут рассмотрены только фрагменты таких решений физического проектирования баз данных и в частности некоторые способы организации данных и методы доступа к ним.

Организация хранения данных.

Реляционные СУБД обладают рядом особенностей, влияющих на организацию данных во внешней памяти. К наиболее важным особенностям можно отнести следующие:

1) Наличие двух уровней системы: уровня непосредственного управления данными во внешней памяти (а также обычно управления буферами оперативной памяти, управления транзакциями и журнализацией изменений БД) и языкового уровня (например, уровня, реализующего язык SQL). При такой организации подсистема нижнего уровня должна поддерживать во внешней памяти набор базовых структур, конкретная интерпретация которых входит в число функций подсистемы верхнего уровня.

2) Поддержание отношений-каталогов. Информация, связанная с именованием объектов базы данных и их конкретными свойствами (например, структура ключа индекса), поддерживается подсистемой языкового уровня. С точки зрения структур внешней памяти отношение-каталог ничем не отличается от обычного отношения базы данных.

3) Регулярность структур данных. Поскольку основным объектом реляционной модели данных является плоская таблица, главный набор объектов внешней памяти может иметь очень простую регулярную структуру.

4) При этом необходимо обеспечить возможность эффективного выполнения операторов языкового уровня как над одним отношением (простые селекция и проекция), так и над несколькими отношениями (наиболее распространено и трудоемко соединение нескольких отношений). Для этого во внешней памяти должны поддерживаться дополнительные "управляющие" структуры - индексы.

5) Наконец, для выполнения требования надежного хранения баз данных необходимо поддерживать избыточность хранения данных, что обычно реализуется в виде журнала изменений базы данных.

Соответственно возникают следующие разновидности объектов во внешней памяти базы данных:

–строки отношений - основная часть базы данных, большей частью непосредственно видимая пользователям;

–управляющие структуры - индексы, создаваемые по инициативе пользователя (администратора) или верхнего уровня системы из соображений повышения эффективности выполнения запросов и обычно автоматически поддерживаемые нижним уровнем системы;

–журнальная информация, поддерживаемая для удовлетворения потребности в надежном хранении данных;

–служебная информация, поддерживаемая для удовлетворения внутренних потребностей нижнего уровня системы (например, информация о свободной памяти).

Хранение отношений.

Существуют два принципиальных подхода к физическому хранению отношений. Наиболее распространенным является покортежное хранение отношений (кортеж является единицей физического хранения). Естественно, это обеспечивает быстрый доступ к целому кортежу, но при этом во внешней памяти дублируются общие значения разных кортежей одного отношения и, вообще говоря, могут потребоваться лишние обмены с внешней памятью, если нужна часть кортежа.

Альтернативным (менее распространенным) подходом является хранение отношения по столбцам, то есть единицей хранения является столбец отношения с исключенными дубликатами. Естественно, что при такой организации суммарно в среднем тратится меньше внешней памяти, поскольку дубликаты значений не хранятся; за один обмен с внешней памятью в общем случае считывается больше полезной информации. Дополнительным преимуществом является возможность использования значений столбца отношения для оптимизации выполнения операций соединения. Но при этом требуются существенные дополнительные действия для сборки целого кортежа (или его части).

Наиболее широко распространено хранение по строкам, поэтому рассмотрим немного более подробно этот способ хранения отношений. Типовая структура страницы данных показана на рис. 3.42.

Рис. 3.42 – Структура страницы данных.

К основным характеристикам организации хранения отношений по строкам можно отнести следующие:

- Каждый кортеж обладает уникальным идентификатором, не изменяемым во все время существования кортежа
- Обычно каждый кортеж хранится целиком в одной странице. Из этого следует, что максимальная длина кортежа любого отношения ограничена размерами страницы. Возникает вопрос: как быть с "длинными" данными, которые в принципе не помещаются в одной странице? Применяются несколько методов. Наиболее простым решением является хранение таких данных в отдельных (вне базы данных) файлах с заменой "длинного" данного в кортеже на имя соответствующего файла.
- Как правило, в одной странице данных хранятся кортежи только одного отношения.
- Изменение схемы хранимого отношения с добавлением нового столбца не вызывает потребности в физической реорганизации отношения. Достаточно лишь изменить информацию в описателе отношения и расширять кортежи только при занесении информации в новый столбец.
- Используется поддержка на уровне хранения. Это достигается путем хранения соответствующей шкалы при каждом кортеже, который в принципе может содержать неопределенные значения.
- Использование кластеризации отношения по значениям одного или нескольких столбцов для повышения эффективности СУБД, совместная кластеризация нескольких отношений.

Хранение отношений по столбцам состоит в совместном хранении всех значений одного (или нескольких) столбцов. Для каждого кортежа отношения хранится кортеж той же степени, состоящий из ссылок на места расположения соответствующих значений столбцов. Одним из приемов является так называемое вертикальное разделение отношений, когда в разных узлах сети

хранятся разные проекции данного отношения. Хранение отношения по столбцам в некотором смысле является предельным случаем вертикального разделения отношений.

Методы доступа к данным. Индексирование. Хеширование.

Вопросы организации данных тесно связаны с операциями, при помощи которых эти данные обрабатываются. К числу таких операций относятся: обновление, добавление и удаление данных. В основе всех перечисленных операций лежит операция доступа, которую нельзя рассматривать независимо от способа организации и представления данных.

В задачах поиска предполагается, что все данные хранятся в памяти с определенной идентификацией и, говоря о доступе, имеют в виду, прежде всего, доступ к данным (называемым ключами), однозначно идентифицирующим связанные с ними совокупности данных.

Пусть необходимо организовать доступ к файлу, содержащему набор одинаковых кортежей, каждая из которых имеет уникальное значение ключевого поля. Самый простой способ поиска - последовательно просматривать каждую запись в файле до тех пор, пока не будет найдена та, значение ключа которой удовлетворяет критерию поиска. Очевидно, этот способ весьма неэффективен, поскольку записи в файле не упорядочены по значению ключевого поля. Сортировка записей в файле также неприменима, поскольку требует еще больших затрат времени и должна выполняться после каждого добавления записи.

Для выполнения операций поиска в отношениях в СУБД применяют индексирование.

Под индексом понимают средство ускорения операции поиска и прямого доступа к кортежу отношения по ключу. Отношение, для которого используется индекс, называют индексированным.

Индекс исполняет роль оглавления отношения, просмотр которого предшествует обращению к кортежам отношения. В некоторых системах индексы хранятся в индексных файлах, хранимых отдельно от табличных файлов.

При индексировании ключи вместе с указателями на соответствующие записи в файле копируют в другую структуру, которая позволяет быстро выполнять операции сортировки и поиска. При доступе к данным вначале в этой структуре находят соответствующее значение ключа, а затем по хранящемуся вместе с ним указателю получают запись из файла.

Обычно индекс определяется для одного отношения, и ключом является значение атрибута (возможно, составного). Если ключом индекса является возможный ключ отношения, то индекс должен обладать свойством уникальности, т.е. не содержать дубликатов ключа. На практике ситуация выглядит обычно противоположно: при объявлении первичного ключа отношения автоматически заводится уникальный индекс, а единственным способом объявления возможного ключа, отличного от первичного, является

явное создание уникального индекса. Это связано с тем, что для проверки сохранения свойства уникальности возможного ключа так или иначе требуется индексная поддержка.

Поскольку при выполнении многих операций языкового уровня требуется сортировка отношений в соответствии со значениями некоторых атрибутов, полезным свойством индекса является обеспечение последовательного просмотра кортежей отношения в диапазоне значений ключа в порядке возрастания или убывания значений ключа.

Общей идеей любой организации индекса, поддерживающего прямой доступ по ключу и последовательный просмотр в порядке возрастания или убывания значений ключа является хранение упорядоченного списка значений ключа с привязкой к каждому значению ключа списка идентификаторов кортежей. Одна организация индекса отличается от другой главным образом в способе поиска ключа с заданным значением.

Варианты решения проблемы организации физического доступа к информации зависят в основном от следующих факторов:

–вида содержимого в поле ключа записей индексного файла;

–типа используемых ссылок (указателей) на запись основного отношения;

–метода поиска нужных записей.

В поле ключа индексного файла можно хранить значения ключевых полей индексируемого отношения либо свертку ключа (так называемый хеш-код, см. раздел 9.4).

Для организации ссылки на кортеж отношения могут использоваться три типа адресов: абсолютный (действительный), относительный и символический (идентификатор).

Организацию индексирования отношений выполняют двумя схемами: одноуровневой и двухуровневой.

При **одноуровневой схеме** в индексном файле хранятся короткие записи, имеющие два поля: поле содержимого старшего ключа (хеш-кода ключа) адресуемого блока и поле адреса начала этого блока. В каждом блоке записи располагаются в порядке возрастания значения ключа или свертки. Старшим ключом каждого блока является ключ его последней записи.

Основным недостатком одноуровневой схемы является то, что ключи (свертки) записей хранятся вместе с записями. Это приводит к увеличению времени поиска записей из-за большой длины просмотра (значения данных в записях приходится пропускать).

Двухуровневая схема в ряде случаев оказывается более рациональной, в ней ключи (свертки) кортежей (записей) отделены от их содержимого. В этой схеме индекс основного отношения распределен по совокупности файлов: одному файлу главного индекса и множеству файлов с блоками ключей.

На практике для создания индекса для некоторой таблицы базы данных пользователь указывает поле таблицы, которое требует индексации. Ключевые поля таблицы во многих СУБД как правило индексируются автоматически.

Индексные файлы, создаваемые по ключевым полям таблицы, часто называются файлами **первичных индексов.**

Индексы, создаваемые пользователем для не ключевых полей, иногда называют **вторичными (пользовательскими) индексами**. Введение таких индексов не изменяет физического расположения записей таблицы, но влияет на последовательность просмотра записей. Индексные файлы, создаваемые для поддержания вторичных индексов таблицы, обычно называются файлами вторичных индексов.

Связь вторичного индекса с элементами данных базы может быть установлена различными способами. Один из них — использование вторичного индекса как входа для получения первичного ключа, по которому затем с использованием первичного индекса производится поиск необходимых записей.

Некоторыми СУБД, например Access, деление индексов на первичные и вторичные не производится. В этом случае используются автоматически создаваемые индексы и индексы, определяемые пользователем по любому из не ключевых полей.

Главная причина повышения скорости выполнения различных операций в индексированных таблицах состоит в том, что основная часть работы производится с небольшими индексными файлами, а не с самими таблицами. Наибольший эффект повышения производительности работы с индексированными таблицами достигается для значительных по объему таблиц. Индексирование требует небольшого дополнительного места на диске и незначительных затрат процессора на изменение индексов в процессе работы. Индексы в общем случае могут изменяться перед выполнением запросов к базе данных, после выполнения запросов к базе данных, по специальным командам пользователя или программным вызовам приложений.

На практике чаще всего используются два класса методов организации индексов и реализующих доступ к данным по ключу:

–методы поиска по дереву;

–методы хеширования.

B-деревья.

С точки зрения внешнего логического представления B-дерево - это сбалансированное сильно ветвистое дерево во внешней памяти (рис. 3.43). **Сбалансированность** означает, что длина пути от корня дерева к любому его листу одна и та же. **Ветвистость** дерева – это свойство каждого узла дерева ссылаться но большое число узлов-потомков. С точки зрения физической организации B-дерево представляется как мультисписочная структура страниц внешней памяти, т.е. каждому узлу дерева соответствует блок внешней памяти (страница).

Для B–дерева:

–сравнительно просто может быть организован последовательный доступ, так как все листья расположены на одном уровне;

–при добавлении и изменении ключей все изменения ограничиваются, как правило, одним узлом.

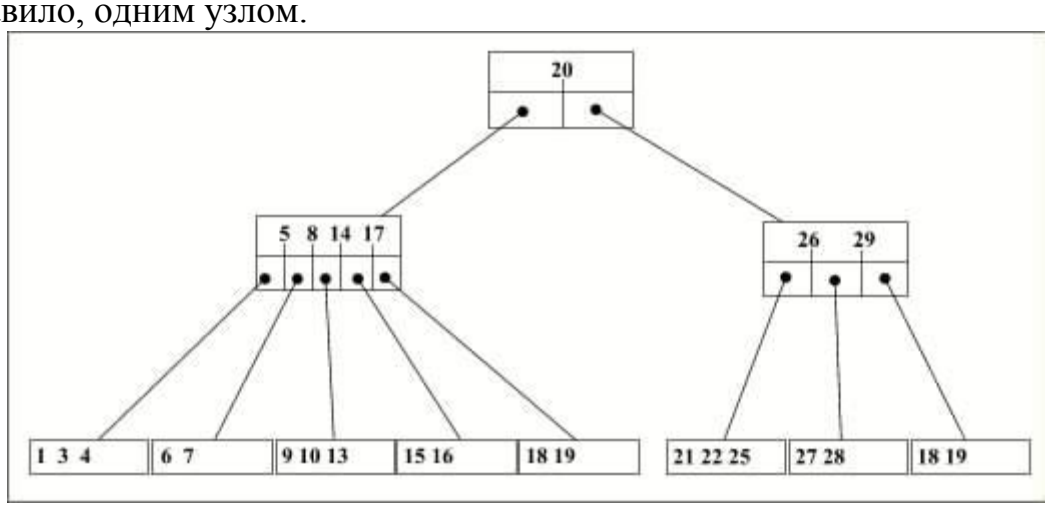


Рис. 3.43 – Сбалансированное дерево.

Поиск по B-дереву - это прохождение от корня к листу в соответствии с заданным значением ключа. Заметим, что поскольку деревья сильно ветвистые и сбалансированные, то для выполнения поиска по любому значению ключа потребуется одно и то же (и обычно небольшое) число обменов с внешней памятью. Более точно, в сбалансированном дереве, где длины всех путей от корня к листу одни и те же, если во внутренней странице помещается n ключей, то при хранении m записей требуется дерево глубиной $\log_n(m)$. Если n достаточно велико (обычный случай), то глубина дерева невелика, и производится быстрый поиск.

Основной "изюминкой" B-деревьев является автоматическое поддержание свойства сбалансированности.

Следует отметить, что B- деревья наилучшим образом подходят только для организации доступа к достаточно простым (одномерным) структурам данных. Для доступа к более сложным структурам, таким, например, как пространственные (многомерные) данные в последнее время используют R-деревья и другие виды деревьев.

Хеширование.

Метод хеширования используется тогда, когда все множество ключей заранее известно и на время обработки может быть размещено в оперативной памяти. В этом случае строится специальная функция, однозначно отображающая множество ключей на множество указателей, называемая хеш-функцией (от английского "to hash" - резать, измельчать). Имея такую функцию можно вычислить адрес записи в файле по заданному ключу поиска. В общем случае ключевые данные, используемые для определения адреса записи организуются в виде таблицы, называемой хеш-таблицей.

Общей идеей методов хэширования является применение к значению ключа некоторой функции свертки (хэш-функции), вырабатывающей значение меньшего размера. Свертка значения ключа затем используется для доступа к записи.

В самом простом, классическом случае, свертка ключа используется как адрес в таблице, содержащей ключи и записи. Основным требованием к хэш-функции является равномерное распределение значение свертки. При возникновении коллизий (одна и та же свертка для нескольких значений ключа) образуются цепочки переполнения. Главным ограничением этого метода является фиксированный размер таблицы. Если таблица заполнена слишком сильно или переполнена, но возникнет слишком много цепочек переполнения, и главное преимущество хэширования - доступ к записи почти всегда за одно обращение к таблице - будет утрачено. Расширение таблицы требует ее полной переделки на основе новой хэш-функции (со значением свертки большего размера).

Преимущество хранения хеш-кода вместо значения состоит в том, что длина свертки независимо от длины исходного значения ключевого поля всегда имеет некоторую постоянную и достаточно малую величину (например, 4 байта), что существенно снижает время поисковых операций. Недостатком хеширования является необходимость выполнения операции свертки (требует определенного времени), а также борьба с возникновением коллизий (свертка различных значений может дать одинаковый хеш-код).

Если в индексном файле хранятся хеш-коды ключевых полей индексированной таблицы, то алгоритм поиска нужной записи (с указанным ключом) в таблице включает в себя следующие три этапа:

–образование свертки значения ключевого поля искомой записи;

–поиск в индексном файле записи о блоке, значение первого поля которого больше полученной свертки (это гарантирует нахождение искомой свертки в этом блоке);

–последовательный просмотр записей блока до совпадения сверток искомой записи и записи блока файла. В случае коллизий сверток ищется запись, значение ключа которой совпадает со значением ключа искомой записи.

Если множество ключей заранее неизвестно или очень велико, то от идеи однозначного вычисления адреса записи по ее ключу отказываются, а хеш-функцию рассматривают просто как функцию, рассеивающую множество ключей во множество адресов.

Глава 4. Распределенная обработка данных

При использовании информационных технологий компьютерных сетей становится необходимой решение проблем удаленного доступа к данным, хранения, обработки, использования данных связанных с реализацией территориального распределения деятельности, производства.

При этом отличительная особенность баз данных одновременное, параллельное использование данных предполагает наличие специальных средств обеспечивающий одновременный и независимый доступ к данным на основе сетевых технологий.

Одной из важнейших сетевых технологий является распределенная обработка данных. Персональные компьютеры стоят на рабочих местах, и соединены каналами связи. Это дало возможность распределить их ресурсы по отдельным функциональным сферам деятельности и изменить технологию обработки данных в направлении децентрализации. Распределенная обработка позволила повысить эффективность удовлетворения изменяющейся информационной потребности информационного (управленческого) работника, и тем самым обеспечить гибкость принимаемых решений.

В системе распределенной обработки клиент может послать запрос к собственной локальной или к удаленной базе данных.

Удаленный запрос – это единичный запрос к одному серверу. Несколько удаленных запросов к одному серверу объединяются в ***удаленную транзакцию*** (см. раздел 11). Если отдельные запросы транзакции обрабатываются различными серверами, то транзакция называется ***распределенной***. При этом один запрос обрабатывается одним сервером.

Транзакция называется ***распределенной***, если отдельные запросы транзакции обрабатываются различными серверами.

Запрос называется ***распределенным,*** если распределенная СУБД обрабатывает один запрос несколькими серверами. Только обработка распределенного запроса поддерживает концепцию распределенной базы данных.

Распределенная обработка данных это наличие специальных средств для обеспечения одновременного и независимого доступа к данным, путем обработки распределенных транзакций и запросов несколькими серверами.

При этом представление, ввод-вывод данных, их обработка на логическом уровне выполняются на ПК клиента, а поддержание базы данных в актуальном состоянии – на сервере.

Распределенная обработка данных предоставляет пользователю ряд преимуществ:

–большое число взаимодействующих между собой пользователей, выполняющих функции сбора, регистрации, хранения, передачи и выдачи информации;

–снятие пиковых нагрузок с централизованной базы путем распределения обработки и хранения локальных баз данных на разных ПК;

–обеспечение симметричного обмена данными между удаленными пользователями;

–обеспечение доступа пользователя к вычислительным ресурсам сети ПК;

–минимальное время обслуживания запроса.

Свойства распределенной обработки данных реализуемых в СУБД:

– прозрачность относительно расположения данных – представление данных как локальных. Прозрачность обеспечивает доступ ко всем объектам независимо от их местоположения, благодаря чему пользователю доступны все сервисы СУБД и может производиться перераспределение компонентов без нежелательных последствий;

–прозрачность относительно сети – стабильная работа в условиях разнородных сетей;

– гетерогенность системы – независимость от архитектуры систем и их производительности;

–поддержка распределенных запросов – объединение данных из любых баз, даже если они размещены в разных системах;

– поддержка распределенных транзакций – (транзакция – логически завершенная единица работы, содержащая одну или более элементарных операций), выполнение распределенных транзакций и поддержка целостности при возникновении отказов;

– поддержка “трехфазного монитора транзакций” (third-party transaction monitor), благодаря которому транзакция выполняется не в два, а в три этапа – сначала посылается запрос о готовности к транзакции.

– поддержка распределенных изменений – возможность изменять данные в любых базах и разных системах, если есть право на доступ;

–обеспечение целостности данных;

–обеспечение безопасности – обеспечение защиты от несанкционированного доступа, потери или изменения данных;

– обеспечение независимости компонентов распределенной обработки данных (СУБД).

Режимы работы РОД определяется следующими признаками:

– многозадачность – одно или многопользовательская;

–правило обслуживания запросов (транзакций) – последовательное или параллельное;

– схема организации процессов обработки и размещения данных – централизованная, распределенная (децентрализованная), смешанная

4.1 Организация процессов обработки данных

Организация процессов обработки данных зависит от способа их распределения. Существуют централизованный, децентрализованный и смешанный способы распределения данных.

Централизованная организация данных является самой простой для реализации (рис. 4.1). На одном сервере находится единственная копия базы данных. Все операции с базой данных обеспечиваются этим сервером. Доступ к данным выполняется с помощью удаленного запроса или удалённой транзакции. Достоинством такого способа является лёгкая поддержка базы данных в актуальном состоянии,

Недостатки централизованной организации данных:

–размер базы данных ограничен размером внешней памяти;

–все запросы направляются к одному серверу с соответствующими затратами на стоимость связи и временную задержку;

–ограничение на параллельную обработку;

–недоступность для удаленных пользователей при появлении ошибок связи;

–выход из строя при отказе центрального сервера.

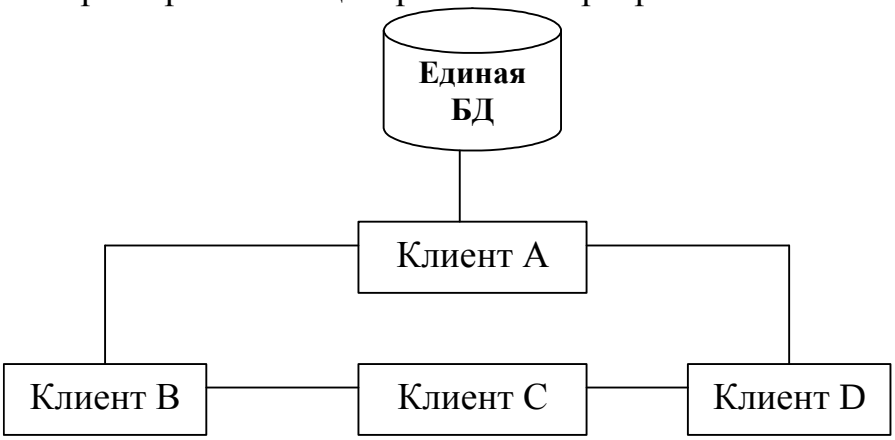


Рис. 4.1 – Централизованная организация данных.

Децентрализованная организация данных основана на разделение информационной базы на несколько физически распределенных баз данных размещенных на нескольких серверах. Работа с базой данных осуществляется на этих же или других ПК, и для доступа к удаленным данным необходимо использовать сетевую СУБД (**не путать с РОД**).

Каждый клиент пользуется своей базой данных, которая может быть либо частью общей информационной базы данных (рис. 4.2), либо копией информационной базы данных в целом (рис. 4.3), что приводит к ее дублированию для каждого клиента.

При распределении данных на основе разбиения база данных размещается на нескольких серверах. Существование копии отдельных частей не допустимо. Достоинства этого метода:

–большинство запросов удовлетворяются локальными базами, что сокращает время ответа;

–увеличиваются доступность данных и надежность их хранения;

–стоимость запросов на выборку и обновление снижается по сравнению с централизованным распределением;

–система останется частично работоспособной, если выйдет из строя один сервер.

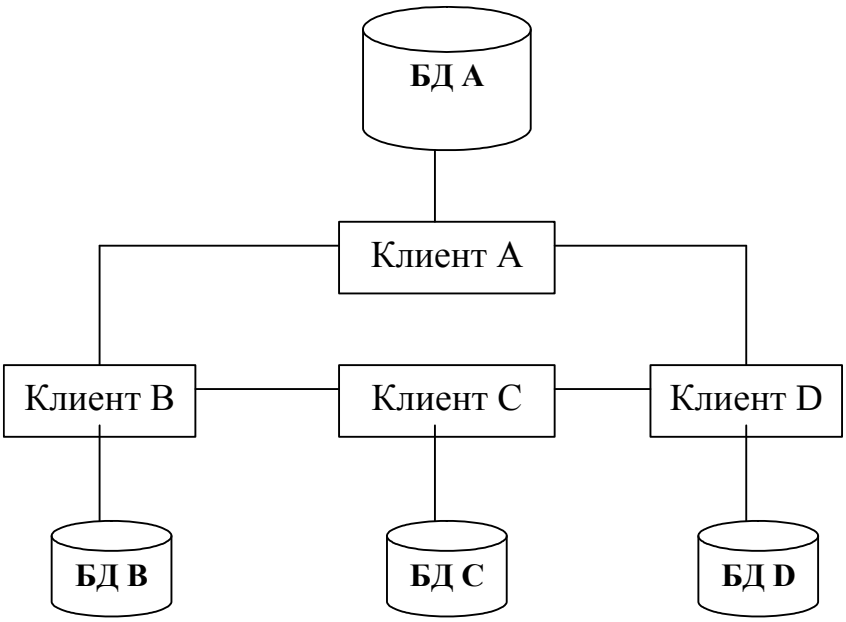


Рис. 4.2 – Децентрализованная организация данных способом распределения.

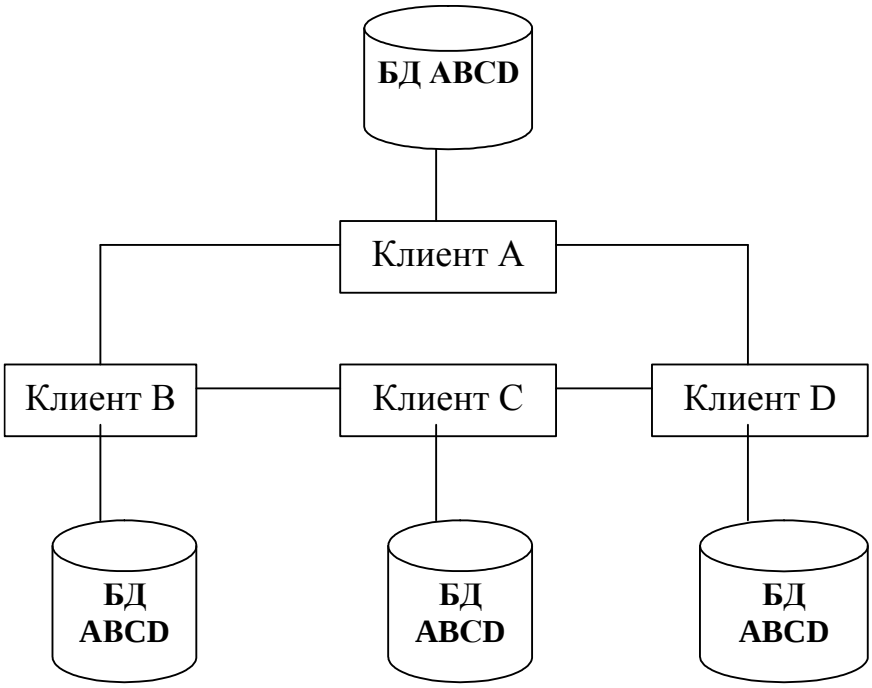


Рис. 4.3 – Децентрализованная организация данных способом дублирования.

Недостатки децентрализованной организации данных:

Достоинства метода:

–большинство запросов удовлетворяются локальными базами данных, что сокращает время ответа;

–увеличиваются доступность данных и надежность их хранения;

–стоимость запросов на выборку и обновление снижается;

–система остается частично работоспособной при выходе из строя одного из серверов.

Недостатки метода:

–часть удаленных запросов или транзакций может потребовать доступ ко всем серверам, что увеличивает время ожидания и цену обслуживания;

–необходимость иметь сведения о размещении данных в различных базах данных;

–необходимость синхронизации операций обновления, изменения и удаления данных одновременно во всех базах данных; усложнение системы обеспечения безопасности данных.

Деление базы данных наиболее целесообразно при совместном использовании локальных и глобальных компьютерных сетей.

Метод дублирования предполагает размещение на каждом сервере полной базы данных, что обеспечивает наибольшую надежность хранения.

Достоинства метода:

–быстрый доступ в результате локального выполнения запросов;

–высокая надежность хранения данных.

Недостатки метода:

–повышенные требования к объему внешней памяти;

–усложнение корректировки базы данных.

Метод дублирования используется, когда фактор надежности является критическим, база данных небольшая, интенсивность обновления невелика.

Смешанная организация хранения данных объединяет два способа распределения: разбиение и дублирование (рис. 4.4).

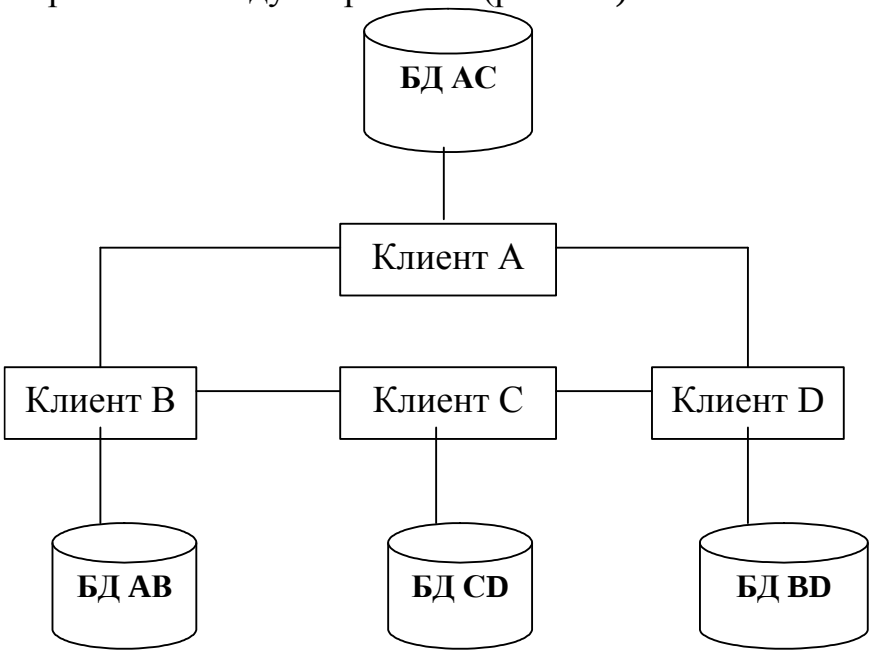


Рис. 4.4 – Смешанная организация данных.

Появляется необходимость хранить информацию о том, где находятся данные в сети. При этом достигается компромисс между объемом памяти под БД в целом и под БД на каждом сервере, чтобы обеспечить надежность и эффективность работы. Данный метод легко реализует параллельную обработку, т.е. обслуживание распределенного запроса или транзакции.

Несмотря на гибкость смешанного способа организации данных, остается проблема взаимозависимости факторов, влияющих на производительность системы, проблема ее надежности и выполнения требований к памяти. Смешанный способ организации данных можно использовать только при наличии сетевой СУБД.

Таким образом с точки зрения_организации процессов обработки данных **распределенная обработка данных** – это возможность распределить ресурсы компьютеров по отдельным функциональным сферам деятельности и изменить технологию обработки данных в направлении децентрализации.

4.2 Базовые технологии и архитектуры распределенной обработки данных

В базах данных коллективного пользования центральным технологическим звеном становятся **серверы баз данных**. Программные средства серверов баз данных обеспечивают реализацию многопользовательских приложений, централизованное хранение, целостность и безопасность данных. Производительность серверов баз данных на порядок выше по сравнению с файл-серверами, которые используются в локальных сетях.

Локальные вычислительные сети создавались для совместного использования дорогостоящего периферийного оборудования. Использование сервера баз данных обеспечило доступ многих пользователей к одним и тем же файлам. Это и стало предпосылкой создания сетевых СУБД.

Введение классификации моделей представления данных на иерархические, сетевые и реляционные отразилось на архитектуре систем управления базами данных и технологии их обработки. Архитектура системы управления базой данных (СУБД) описывает ее функционирование как взаимодействие процессов двух типов: клиента и сервера.

Технология **клиент-сервер** позволила совместить достоинства однопользовательских систем (высокий уровень диалоговой поддержки, дружественный интерфейс, низкая цена) с достоинством более крупных компьютерных систем (поддержка целостности, защита данных, многозадачность).

Основная идея технологии клиент – сервер заключается в том, чтобы серверы расположить на мощных машинах, а приложения клиентов, использующих язык, – на менее мощных машинах. Тем самым будут задействованы ресурсы более мощного сервера и менее мощных машин клиентов. Ввод-вывод к базе основан не на физическом дроблении данных, а на

логическом, то есть сервер отправляет клиентам не полную копию базы, а только логически необходимые порции, тем самым, сокращая трафик сети. В технологии клиент-сервер программы клиента и его запросы хранятся отдельно от СУБД. Сервер обрабатывает запросы клиентов, выбирает необходимые данные из базы данных, посылает их клиентам по сети, производит обновление информации, обеспечивает целостность и сохранность данных.

Клиент отвечает за целевую обработку данных и организацию взаимодействия с пользователем. Как правило, программное обеспечение клиента организованно в виде приложения работающего поверх СУБД и использует интерфейс внешнего уровня.

Сервер обеспечивает фундаментальные функции хранения и управления данными, обрабатывает запросы и выдает результаты клиенту для целевой обработки. Программное обеспечение сервера – реализует функции СУБД

Характеристика клиента и сервера как процессов.

Клиент как процесс взаимодействуя с пользователем имеет следующие характеристики:

1. Представляет интерфейс пользователя. Этот интерфейс является единственным средством, принимающим запросы пользователя или по поиску данных и анализу, а также средством представления результатом одного или нескольких запросов или команд. Обычно клиент представляет пользователю графический интерфейс пользователя;
2. Клиент формирует на определенном языке один или несколько запросов или команд для представления серверу. Клиент сервер могут использовать стандартный язык, например, SQL или другой язык совместимый с системой клиент – сервер.
3. Для сокращения числа запросов к серверу, или для проведения операции защиты данных и контроля доступа, клиент может применять кэширование и оптимизацию. Клиент может также проверять целесообразность запросов или команд, запрашиваемых пользователем. Иногда нет необходимости вообще направлять запрос серверу. В таких случаях клиент сам может производить обработку данных, запрошенных пользователем.
4. Клиент осуществляет связь с сервером по методологии коммуникации между процессами, передает запросы и команды серверу.
5. Клиент проводит анализ данных по результатам выполнения запроса или команды от сервера и затем представляет их пользователю.

Сервер в системе клиент – сервер может являться процессом или рядом процессов, которые должны присутствовать в операционной системе, предоставляющей слуги одному или нескольким клиентам.

Сервер имеет следующие характеристики.

1. Сервер предоставляет услуги клиенту. Характер и диапазон таких услуг зависит от назначения самой системы клиент – сервер. Услуги, предоставляемые сервером, предусматривают минимальные расчеты на

базе сервера (например, сервера принтера или файла), особенно интенсивных вычислений (например, серверы БД или обработки изображений).

2. Сервер просто реагирует на запросы или команды клиента и не являются инициатором диалога с клиентом. Он служит либо в качестве архива данных (сервер файла) или архива знаний (сервер БД) или для предоставления услуг (сервер принтера).
3. Идеальный сервер скрывает от клиента и пользователя всю структуру и состав системы клиент – сервер. Клиент, связанный с сервером, совершенно не должен знать платформу сервера (аппаратную и программную часть), а также технологию коммуникации (аппаратную и программную). Например, клиент на базе WINDOWS должен уметь вести диалог с сервером на базе UNIX или OS/2, независимо от типа операционной системы на сервере и технологии ЛВС, соединяющей клиента и сервера.

Основные виды технологии распределённой обработки данных.

1. *Технология клиент – сервер, ориентированная на автономный компьютер,* то есть и клиент и сервер размещены на одном ПК. По функциональным возможностям такая система аналогична централизованной СУБД. Ни распределённая обработка, ни распределённая СУБД не поддерживаются.
2. *Технология клиент – сервер, ориентированная на централизованное распределение.* При использовании этой технологии клиент получает доступ к данным одиночного удалённого сервера, данные могут только считываться, динамический доступ к данным реализуется посредством удалённых транзакций и запросов, их число должно быть невелико, чтобы не снизилась производительность системы.
3. *Технология клиент – сервер, ориентированная на ЛВС.* Эта технология характеризуется следующими особенностями:

–единственный сервер обеспечивает доступ к базе;

–клиент формирует процесс, отвечающий за содержательную обработку данных, их представление и логический доступ к базе;

–доступ к базе данных замедлен, т.к. клиент и сервер связаны через локальную сеть.

4. *Технология клиент – сервер, ориентированная на изменения данных в одном месте.* В случае применения этой технологии реализуется обработка распределённой транзакции; удалённые серверы не связаны между собой компьютерной сетью, то есть отсутствует сервер-координатор; клиент может изменять данные только в своей локальной базе; возникает опасность «зацикливания» то есть ситуации, когда задача А ждет записи, заблокированные задачей В, а задача В ждет записи, заблокированные задачей А. Поэтому распределенная СУБД должна иметь средства контроля совпадений противоречивых запросов.

5. *Технология клиент – сервер, ориентированная на изменение данных в нескольких местах.* В отличие от предыдущей технологии здесь имеется сервер-координатор, поддерживающий протокол передачи данных между различными серверами. Возможна разработка определенных транзакций в различных удаленных серверах. Это создает предпосылки разработки распределенной СУБД. Реализуется стратегия смешанного распределения путем передачи копий с помощью СУБД.
6. *Технология клиент – сервер, ориентированная на распределенную СУБД.* Она обеспечивает стратегию разбиения и дублирования, позволяет получить более быстрый доступ к данным. Распределенная СУБД обеспечивает независимость клиента от места размещения сервера, глобальную оптимизацию, распределенный контроль целостности азы, распределенное административное управление.

Во всех технологиях существует два способа связи программ клиента и сервера баз данных: прямой и непрямой. При **прямом соединении** прикладная программа клиента связывается непосредственно с сервером БД, а при **непрямом соединении** – доступ к удаленному серверу обеспечивается средствами локальной базы. Возможно объединение этих способов.

Использование технологии клиент – сервер позволяет перенести часть работы с сервера на ПК клиента, оснащенную инструментальными средствами для выполнения его профессиональных обязанностей. Тем самым данная технология позволяет независимо наращивать возможности сервера базы данных и совершенствовать инструментальные средства клиента.

Недостаток технологии клиент – сервер заключается в повышении требований к производительности сервера, в усложнении управления вычислительной сетью, а при отсутствии сетевой СУБД – в сложности организации распределенной обработки.

Один из основных принципов технологии клиент–сервер заключается в разделении функций стандартного приложения на три группы, имеющие различную природу. Первая группа – это представление и отображение данных, их функциональная обработка (ввод, изменение, удаление). Вторая группа объединяет чисто прикладные функции, характерные для данной предметной области. Наконец к третьей группе относятся фундаментальные функции хранения и управления данными (базами данных, файловыми системами, и т. д.)

Различия в реализациях приложений в рамках технологии «клиент-сервер» определяется тремя факторами.

Во-первых, тем, в какие виды программного обеспечения интегрирован каждый из этих компонентов. Во-вторых, тем, какие механизмы используются для реализации функций всех трех групп. В третьих – как логические компоненты распределяются между компьютерами в сети.

Выделяют три подхода или архитектуры технологии клиент-сервер, каждый из которых реализован в соответствующей модели:

–модель файл –сервер FS;

–модель сервера базы данных (Data Base Server – DBS);

–модель сервера приложений (Application Server – AS).

Архитектура «файл-сервер» (FS).

В модели файл – сервер обработка данных распределена в среде, обычно представляющую собой локальную вычислительную сеть (ЛВС). Пользовательские приложения, средства организации и управления базой данных, в том числе и СУБД располагаются на компьютере-клиенте, а база данных, представляющая собой набор специализированных структурированных файлов, на компьютере-сервере. Клиенты обращаются к файловому серверу только по мере необходимости получения доступа к нужным им файлам. Удаленный разделенный доступ к файлам обеспечивают сетевые операционные системы.

Таким образом, модель файл–сервер функционирует просто как совместно используемый жесткий диск.

При запросах на выборку данных со стороны рабочих станций, поскольку файловый сервер не понимает команд на ЯМД, СУБД должна запросить у файлового сервера все файлы целиком, соответствующие запрашиваемым данным. Таким образом, на рабочую станцию пересылаются не только данные необходимые пользователю, а весь файл в целом (рис. 4.5).

Достоинства модели – возможность обслуживания нескольких клиентов.

Недостатки файл-серверной архитектуры:

–большой объем сетевого трафика;

–на каждой рабочей станции должна находится полная копия СУБД;

–управление параллельностью, восстановлением и целостностью усложняется, поскольку доступ к одним и тем же файлам могут осуществлять сразу несколько экземпляров СУБД;

– низкий уровень защиты;

–большое время обработки запросов.

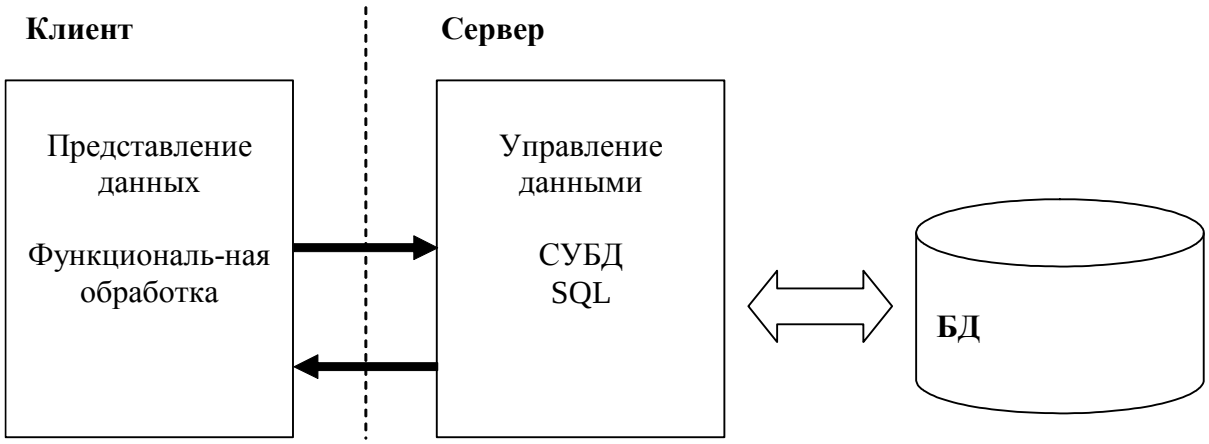


Рис. 4.5 – Модель файл–сервер.

Архитектура с выделенным сервером базы данных (двухзвенная) или сервер базы данных (выделенный сервер)((Data Base Server – DBS).

Средства управления базой данных и база данных размещены на машине-сервере. Доступ к базе данных обеспечивается, операторами языка. Запросы к информационным ресурсам направляются по сети удалённому компьютеру (например, серверу базы данных). Последний обрабатывает и выполняет запросы и возвращает клиенту блоки данных. Такая обработка включает проверку полномочий клиента, обеспечение требований целостности, поддержку системного каталога, а так же выполнение и обновление данных. Так же поддерживается управлением параллельностью и восстановлением данных (рис. 4.6).

Клиент управляет пользовательским интерфейсом и логикой приложения, действуя, как сложная рабочая станция, на которой выполняется приложения баз данных. Клиент принимает от пользователя запрос, проверяет синтаксис и генерирует запрос к базе данных на языке ЯМД. Затем он передает сообщение серверу, ожидает поступление ответа и форматирует полученные данные для представления их пользователю.

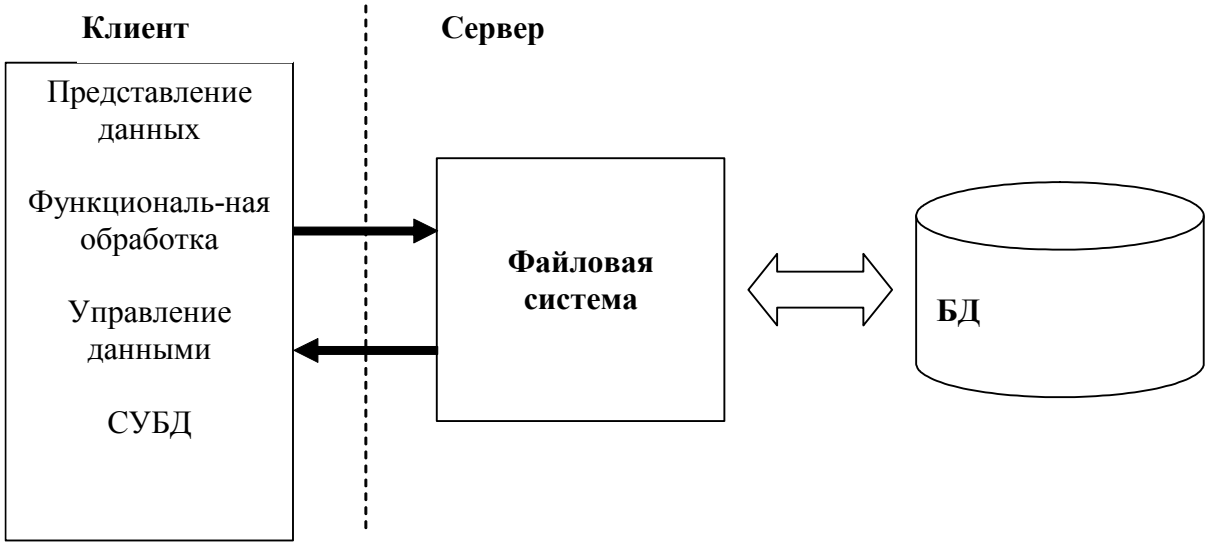


Рис. 4.6 – Модель сервера базы данных (DBS).

Достоинства модели:

– возможность обслуживания нескольких клиентов;

–низкая загрузка сети и машин-клиентов;

–защита данных средствами СУБД, что позволяет ограничивать доступ к данным и действия клиентов; используются управление транзакциями – блокировка одновременного обновления данных;

–повышается общая производительность системы. Поскольку клиенты и сервер находятся на разных компьютерах, их процессоры способны выполнять приложения параллельно. При этом настройка производительности компьютера с сервером упрощается, поскольку на нем выполняется только работа с базой данных.

–сокращаются коммуникационные расходы. Приложения выполняют часть операций на клиентских и посылают через сеть только запросы к базе

данных, что позволяет существенно сократить объем пересылаемых по сети данных.

Недостатки модели – так как правила обработки запросов на клиентских машинах могут различаться – низкий уровень управления целостностью и непротиворечивости информации.

Архитектура сервера приложений (трехзвенная)(Application Server – AS).

В трехзвенной архитектуре «неинтеллектуальный» клиент на рабочей станции управляет только пользовательским интерфейсом, тогда как средней уровень обработки данных управляет всей остальной логикой приложения. Третьем уровнем здесь является север базы данных. Функции доступа и обработки распределены между клиентом, специальной программой сервером приложением, на который переносится большая часть бизнес-логики и значительная часть программных компонентов управления данными и сервером, который обеспечивает функции СУБД (рис. 4.7).

Сервер приложений – это приложение выполняет некоторые функции, предоставляя другим соответствующий набор услуг.

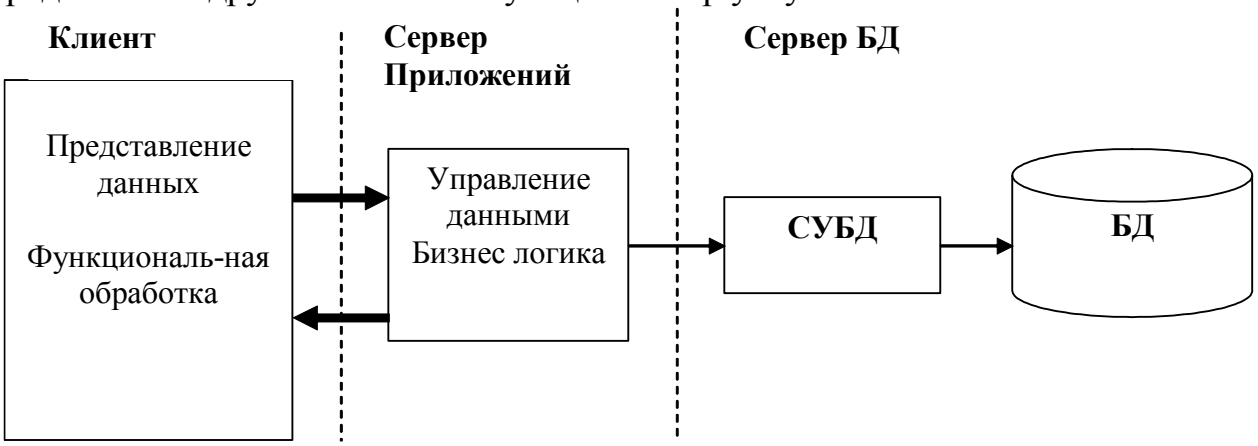


Рис. 4.7 – Модель сервера приложений (AS).

В AS-модели процесс, выполняющийся на компьютере-клиенте, отвечает, как обычно, за ввод и отображение данных, за интерфейс с пользователем. Обращаясь за выполнением услуг к прикладному компоненту, этот процесс играет роль клиента приложения (Application Client - AC). Прикладные функции выполняются группой процессов – серверов приложений (Application Server - AS), функционирующих на удаленном компьютере (или нескольких компьютерах). Доступ к информационным ресурсам, необходим для решения прикладных задач обеспечивается, как правило, операторами специального языка SQL или вызовами функций специальной библиотеки. Все операции над информационными ресурсами выполняются соответствующим компонентом, по отношению к которому AS играет роль клиента.Запросы к информационным ресурсам направляются по сети удалённому компьютеру (например, серверу базы данных). Последний обрабатывает и выполняет запросы и возвращает клиенту блоки данных.

Сервер базы данных обеспечивают функции СУБД по ведению и обслуживанию базы данных.

К достоинствам трехзвенной архитектуры можно отнести:

–централизованное ведение бизнес-логики;

–отсутствие необходимости устанавливать на клиентских машинах компоненту программного обеспечения управления к доступам данных;

–возможность отложенного обновления базы данных, запрошенных с сервера, в автономном режиме;

–повышается уровень непротиворечивости данных. Сервер может самостоятельно управлять проверкой целостностью данных, поскольку все ограничения определяются и проверяются только в одном месте. При этом каждому приложению не придется выполнять собственную поверку;

–архитектура обеспечивает наращивание функциональности приложений и числа обслуживаемых пользователей за счет переноса основных операций на отдельный уровень и распределения нагрузки аппаратных средств.

Архитектура серверов баз данных.

1. Один к одному – для обслуживания каждого запроса запускается отдельный серверный процесс.

2. Многопотоковая односерверная – обработку всех запросов выполняет один серверный процесс, для каждого клиентского запроса создается поток в рамках которого и локализуется запрос.

3. Сервер с параллельной обработкой запроса. Для обработки каждого запроса запускается несколько серверных процессов (горизонтальный параллелизм). Запрос разбивается на подзапросы, которые обслуживаются отдельными серверными процессами независимо от остальных (вертикальный параллелизм). Результаты объединяются и передаются клиенту.

4.3 Транзакции и обеспечение целостности баз данных.

Поддержание механизма транзакций - показатель уровня развитости СУБД. Корректное поддержание транзакций одновременно является основой обеспечения целостности баз данных (и поэтому транзакции вполне уместны и в однопользовательских персональных СУБД), а также составляют базис изолированности пользователей во многопользовательских системах. Часто эти два аспекта рассматриваются по отдельности, но на самом деле они тесно взаимосвязаны.

Транзакция это неделимая последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации) такая, что либо результаты всех операторов, входящих в транзакцию, полностью выполняются (отображаются в базе данных), либо воздействие всех этих операторов полностью отсутствует. Также транзакцией называют логически

завершенную единицу работы, содержащую одну или более элементарных операций обработки данных.

Для пользователя транзакция выполняется по принципу "*все или ничего*", то есть либо транзакция выполняется целиком и переводит базу данных из одного ***целостного состояния*** в другое ***целостное состояние***, либо, если по каким-либо причинам, одно из действий транзакции невыполнимо, или произошло какое-либо нарушение работы системы, база данных возвращается в исходное состояние, которое было до начала транзакции (происходит откат транзакции).

Соответственно каждая транзакция должна начинаться при целостном состоянии базы данных и должна сохранять это состояние после своего завершения.

При выполнении транзакций результаты выполнения всех операторов входящих в транзакцию либо отображаются в базе данных, либо воздействие этих операторов полностью отсутствует.

Транзакции являются *единицами восстановления* данных после сбоев - восстанавливаясь, система ликвидирует следы транзакций, не успевших успешно завершиться в результате программного или аппаратного сбоя. Эти два свойства транзакций определяют атомарность (неделимость) транзакции. В многопользовательских системах, кроме того, транзакции служат для обеспечения ***изолированной*** работы отдельных пользователей - пользователям, одновременно работающим с одной базой данных, кажется, что они работают как бы в однопользовательской системе и не мешают друг другу.

Транзакция обладает четырьмя важными свойствами, известными как ***свойства АСИД***:

– атомарность – операции транзакции образуют неделимый блок операций с определенным началом и концом. Этот блок или выполняется от начала до конца или не выполняется вообще. Если в процессе выполнения транзакции произошел сбой, происходит откат к исходному состоянию;

– согласованность – по завершении транзакции все задействованные ресурсы находятся в согласованном состоянии;

– изолированность – одновременный доступ транзакций различных приложений к разделяемым ресурсам, таким образом, что транзакции не влияют друг на друга;

– долговременность – все изменения данных (ресурсов), осуществленные в процессе выполнения транзакций, не могут быть потеряны.

Транзакция обычно начинается автоматически от точки сохранения, задаваемой инструкцией SAVEPOINT и с момента присоединения пользователя к СУБД и продолжается до тех пор, пока не произойдет одно из следующих событий:

–подана команда COMMIT WORK (зафиксировать транзакцию);

–подана команда ROLLBACK WORK (откатить транзакцию);

–произошло отсоединение пользователя от СУБД;

–произошел сбой системы.

Команда COMMIT WORK завершает текущую транзакцию и автоматически начинает новую транзакцию. При состоянии (ситуации) COMMIT гарантируется фиксация в базе данных всех изменений, сделанных текущей транзакцией, когда операторы транзакции выполняются и происходит нормальное завершение транзакции и база данных переходит в обновленное состояние.

Команда ROLLBACK WORK приводит к тому, что все изменения, сделанные текущей транзакцией откатываются, то есть отменяются так, как будто их вообще не было и база данных возвращается к исходному состоянию. При этом автоматически начинается новая транзакция.

При отсоединении пользователя от СУБД происходит автоматическая фиксация транзакций.

При сбое системы происходят более сложные процессы. Кратко суть их сводится к тому, что при последующем запуске системы происходит анализ выполнявшихся до момента сбоя транзакций. Те транзакции, для которых была подана команда COMMIT WORK, но результаты работы которых не были занесены в базу данных выполняются снова (накатываются). Те транзакции, для которых не была подана команда COMMIT WORK, откатываются.

Ограничения целостности баз данных.

Для иллюстрации возможного нарушения целостности базы данных рассмотрим следующий пример:

Пусть имеется система, в которой хранятся данные о проектах и работающих над ними сотрудниках (рис.3.32 – 3.41). Список проектов хранится в таблице СОТРУДНИКИ-ПРОЕКТЫ (СОТР_НОМЕР, ПРО_НОМЕР, СОТР_ЗАДАН), где СОТР_НОМЕР – идентификатор сотрудника, ПРО_НОМЕР - идентификатор проекта, СОТР_ЗАДАН – задание, выполняемое сотрудником. Список сотрудников хранится в таблице СОТРУДНИКИ (СОТР_НОМЕР, СОТР_ФИО), где СОТР_НОМЕР – идентификатор сотрудника, СОТР_ФИО – фамилия сотрудника.

Ограничение целостности этой базы данных состоит в том, что поле СОТР_НОМЕР в любой таблице не может заполняться произвольными значениями – это поле должно содержать конкретных сотрудников, реально работающих над проектами.

С учетом этого ограничения можно заключить, что вставка нового сотрудника в любую таблицу не может быть выполнена одной операцией. При вставке нового сотрудника в одну таблицу необходимо одновременно изменить значение поля СОТР_НОМЕР в другой таблице.

Если после выполнения первой операции и до выполнения второй произойдет сбой системы, то реально будет выполнена только первая операция и база данных остается в нецелостном состоянии.

Ограничение целостности – это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

Примерами ограничений целостности могут служить следующие утверждения:

–возраст сотрудника не может быть меньше 18 и больше 65 лет;

–каждый сотрудник имеет уникальный табельный номер;

–сотрудник обязан числиться в одном отделе.

База данных находится в согласованном (целостном) состоянии, если выполнены (удовлетворены) все ограничения целостности, определенные для базы данных.

В данном определении важно подчеркнуть, что должны быть выполнены не все вообще ограничения предметной области, а только те, которые определены в базе данных. Для этого необходимо, чтобы СУБД обладала развитыми средствами поддержки ограничений целостности. Если какая-либо СУБД не может отобразить все необходимые ограничения предметной области, то такая база данных хотя и будет находиться в целостном состоянии с точки зрения СУБД, но это состояние не будет правильным с точки зрения пользователя.

Вместе с понятием целостности базы данных возникает понятие **реакции системы на попытку нарушения целостности**. Система должна не только проверять, не нарушаются ли ограничения в ходе выполнения различных операций, но и должным образом реагировать, если операция приводит к нарушению целостности. Имеется два типа реакции на попытку нарушения целостности:

–отказ выполнить "незаконную" операцию;

–выполнение компенсирующих действий.

Например, если система знает, что в поле "*Возраст_Сотрудника*" должны быть целые числа в диапазоне от 18 до 65, то система отвергает попытку ввести значение возраста 66. При этом может генерироваться какое-нибудь сообщение для пользователя.

В противоположность этому, в примере, о добавлении в базу данных сотрудника, система допускает вставку записи о новом сотруднике (что приводит к нарушению целостности базы данных), но автоматически производит компенсирующие действия, изменяя значение поля СОТР_ЗАДАН в таблице СОТРУДНИКИ-ПРОЕКТЫ.

Работу системы по проверке ограничений можно изобразить на следующем рисунке 4.8:

Рис. 4.8 – Проверка ограничения целостности в базе данных.

В некоторых случаях система может не выполнять проверку на нарушение ограничений, а сразу выполнять компенсирующие операции. Действительно, в указанном примере, при вставке или удалении сотрудника проверку производить не нужно, т.к. результаты ее известны заранее - ограничение обязательно будет нарушено. В этом случае необходимо сразу приступать к компенсированию возникшего нарушения.

Классификация ограничений целостности.

Ограничения целостности можно классифицировать несколькими способами:

–по способам реализации;
–по времени проверки;
–по области действия.

Классификация ограничений целостности по способам реализации.

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

–***декларативная поддержка ограничений*** целостности заключается в определении ограничений средствами языка определения данных. Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей);

–***процедурная поддержка ограничений целостности*** заключается в использовании триггеров и хранимых процедур.

Хранимые процедуры (программный код) – это процедуры и функции, хранящиеся непосредственно в базе данных в откомпилированном виде и которые могут запускаться пользователями или приложениями, работающими с базой данных. Хранимые процедуры обычно пишутся либо на специальном процедурном расширении языка SQL (например, PL/SQL для ORACLE или Transact-SQL для MS SQL Server), или на некотором универсальном языке программирования, например, C++, с включением в код операторов SQL в соответствии со специальными правилами такого включения. Основное назначение хранимых процедур - реализация бизнес-процессов предметной области.

Триггеры – это хранимые процедуры, связанные с некоторыми событиями, происходящими во время работы базы данных. В качестве таких событий выступают операции вставки, обновления и удаления строк таблиц. Если в базе данных определен некоторый триггер, то он запускается автоматически всегда при возникновении события, с которым этот триггер связан. Очень важным является то, что пользователь не может обойти триггер. Триггер срабатывает независимо от того, кто из пользователей и каким способом инициировал событие, вызвавшее запуск триггера. Таким образом, основное назначение триггеров - автоматическая поддержка целостности базы данных.

Если для таблицы, имеющей триггер, есть ограничения, то они проверяются до выполнения триггера. Если ограничения не могут быть преодолены, то выражение не выполнится, а вслед за ним не сможет сработать триггер.

Триггер имеет следующие характеристики:

–он связан с таблицами;

–выполняется автоматически независимо от того, как происходит изменение данных (помощью команды UPDATE, в процессе работы пользовательского приложения, написанного на Visual Basic и т.д.).

–триггер не может быть вызван напрямую и не имеет параметров;

–он может иметь до 16 уровней вложенности. Это позволяет триггеру, который изменяет значение в таблице, вызвать другой триггер, который в свою очередь запустит на исполнение следующий;

–триггер является продвинутой формой правил, которые позволяют устанавливать более полный контроль над данными. Они предупреждают о вводе неправильных данных и данных, которые пытаются нарушить внутреннее соответствие в базах данных.

По сути, наличие ограничения целостности (как декларативного, так и процедурного характера) всегда приводит к созданию или использованию некоторого программного кода, реализующего это ограничение. Разница заключается в том, где такой код хранится и как он создается.

При декларировании (объявлении) ограничения текст ограничения хранится в виде некоторого объекта СУБД, а для реализации ограничения используются встроенные в СУБД функции, и тогда этот код представляет собой внутренние функции ядра СУБД.

Классификация ограничений целостности по времени проверки

По времени проверки ограничения делятся на:

–***немедленно проверяемые ограничения*** проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Например, проверка уникальности потенциального ключа проверяется в момент вставки записи в таблицу. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности откатывается;

–***ограничения с отложенной проверкой*** проверяется в момент фиксации транзакции оператором COMMIT WORK. Внутри транзакции ограничение может не выполняться. Если в момент фиксации транзакции обнаруживается нарушение ограничения с отложенной проверкой, то транзакция откатывается.

Классификация ограничений целостности по области действия

По области действия ограничения делятся на:

–***ограничения домена*** представляют собой ограничения, накладываемые только на допустимые значения домена. Например, ограничением домена "Возраст сотрудника" может быть условие "Возраст сотрудника не менее 18 и не более 65";

–***ограничения атрибута*** представляют собой немедленно проверяемые ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене. Ограничение атрибута в точности совпадают с ограничениями соответствующего домена;

–***ограничения кортежа*** представляют собой ограничения, накладываемые на допустимые значения отдельного кортежа отношения, и не являющиеся ограничением целостности атрибута;

–***ограничения отношения*** представляют немедленно проверяемые ограничения, накладываемые только на допустимые значения отдельного отношения, и не являющиеся ограничением целостности кортежа. Например, ограничение целостности сущности, задаваемое потенциальным ключом отношения или наличие функциональных зависимостей являются ограничением отношения, т.к. для его проверки необходимо иметь информацию обо всех кортежах отношения;

–***ограничения базы данных*** представляют ограничения, накладываемые на значения двух или более связанных между собой отношений (в том числе отношение может быть связано само с собой). Например, ограничение целостности ссылок, задаваемое внешним ключом отношения при связи данных размещенных в разных таблицах, является ограничением базы данных.

Проверка ограничений допускается как после выполнения каждого оператора, могущего нарушить ограничение, так и в конце транзакции. Во время выполнения транзакции можно изменить режим проверки ограничения.

Журнализация, сериализация и синхронизация транзакций.

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние базы данных после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

В любом случае для восстановления базы данных нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в базе данных требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений базы данных.

Журнал транзакций – это особая часть базы данных, недоступная пользователям СУБД, в который поступают записи обо всех изменениях основной части базы данных. **Журнализация** – сохранение во внешней памяти информации о модификациях базы данных, то есть успешно завершенная транзакция фиксируется во внешней памяти.

В разных СУБД изменения базы данных журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения базы данных (например, операции удаления строки из таблицы реляционной базы данных).

При журнализации соблюдается правило WAL (Write Ahead Log) пиши сначала в журнал) выполненная транзакция сохраняется перед тем, как сохранится измененный объект базы данных. То есть в журнале транзакций содержаться все данные для восстановления системы после сбоя.

Вместе с журналом транзакций используется архивная копия базы данных. При восстановлении по журналу выполняются повторно все операции, а для транзакций, которые не закончились к моменту сбоя, выполняется откат.

Самая простая ситуация восстановления – индивидуальный откат транзакции. Для этого, в одних СУБД, поддерживается локальный журнал операций модификации базы данных, выполненных в этой транзакции, а откат транзакции проводится путем выполнения обратных операций, следуя от конца локального журнала. В других СУБД индивидуальный откат транзакции выполняют по общесистемному журналу, для чего все записи от одной транзакции связывают обратным списком (от конца к началу).

При мягком сбое во внешней памяти основной части базы данных могут находиться объекты, модифицированные транзакциями, не закончившимися к моменту сбоя, и могут отсутствовать объекты, модифицированные транзакциями, которые к моменту сбоя успешно завершились (по причине использования буферов оперативной памяти, содержимое которых при мягком сбое пропадает). При соблюдении протокола WAL во внешней памяти журнала должны гарантированно находиться записи, относящиеся к операциям модификации обоих видов объектов. Целью процесса восстановления после мягкого сбоя является состояние внешней памяти основной части базы данных, которое возникло бы при фиксации во внешней памяти изменений всех завершившихся транзакций и которое не содержало бы никаких следов незаконченных транзакций. Для того чтобы этого добиться, сначала производят откат незавершенных транзакций, а потом повторно воспроизводят те операции завершенных транзакций, результаты которых не отображены во внешней памяти.

Для восстановления базы данных после жесткого сбоя используют журнал и архивную копию базы данных. Архивная копия - это полная копия базы данных к моменту начала заполнения журнала. Восстановление базы данных состоит в том, что исходя из архивной копии, по журналу воспроизводится работа всех транзакций, которые закончились к моменту сбоя. Для нормального восстановления базы данных после жесткого сбоя необходимо, чтобы сам журнал транзакций сохранился и не был поврежден. Соответственно, к сохранности журнала во внешней памяти в СУБД предъявляются особо повышенные требования.

Синхронизация (изолированность) транзакций.

Во многопользовательских системах с одной базой данных одновременно могут работать несколько пользователей или прикладных программ, соответственно, СУБД требуется управлять одновременно множеством транзакций. В связи со свойством сохранения целостности БД транзакции являются подходящими единицами изолированности пользователей. Действительно, если с каждым сеансом работы с базой данных ассоциируется транзакция, то каждый пользователь начинает работу с согласованным состоянием базы данных, т.е. с таким состоянием, в котором база данных могла бы находиться, даже если бы пользователь работал с ней в одиночку.

Предельной задачей системы при выполнении синхронных (параллельных транзакций) является обеспечение изолированности

пользователей, то есть создание достоверной и надежной иллюзии того, что каждый из пользователей работает с базой данных в одиночку.

При выполнении параллельных транзакций и обеспечения изолированности необходимо:

1) **Отсутствие потерянных изменений** (пропавшие обновления).

Пример. Во время совместного выполнения двух транзакций, транзакция 1 изменяет объект базы данных A. До завершения транзакции 1 транзакция 2 также изменяет объект A. Транзакция 2 завершается оператором ROLLBACK (например, по причине нарушения ограничений целостности). Тогда при повторном чтении объекта A транзакция 1 не видит изменений этого объекта, произведенных ранее. Такая ситуация называется ситуацией потерянных изменений. Естественно, она противоречит требованию изолированности пользователей.

Чтобы избежать ситуации потерянных изменений требуется, чтобы до завершения одной транзакции, изменивший базу данных, никакая другая транзакция не должна читать измененные данные до завершения транзакции и никакая другая транзакция не могла изменять объект. Отсутствие потерянных изменений является минимальным требованием к СУБД по части синхронизации параллельно выполняемых транзакций.

2) **Чтение «грязных» данных**, то есть до завершения одной транзакции, изменившей базу данных, никакая другая транзакция не может (не должна) читать изменяемые данные.

Пример. При совместном выполнения транзакций 1 и 2, транзакция 1 изменяет объект базы данных A. Параллельно с этим транзакция 2 читает объект A. Поскольку операция изменения еще не завершена, транзакция 2 видит несогласованные "грязные" данные (в частности, операция транзакции 1 может быть отвернута при проверке немедленно проверяемого ограничения целостности). Это тоже не соответствует требованию изолированности пользователей (каждый пользователь начинает свою транзакцию при согласованном состоянии базы данных и в праве ожидать видеть согласованные данные).

Чтобы избежать ситуации чтения "грязных" данных, до завершения транзакции 1, изменившей объект A, никакая другая транзакция не должна читать объект A (минимальным требованием является блокировка чтения объекта A до завершения операции его изменения в транзакции 1).

3) **Отсутствие неповторяющихся чтений** – ситуация изменения данных (незавершенная транзакция) и чтение этих данных другим пользователем (чтение несогласованных данных).

Пример. Транзакция 1 читает объект базы данных A. До завершения транзакции 1 транзакция 2 изменяет объект A и успешно завершается оператором COMMIT. Транзакция 1 повторно читает объект A и видит его измененное состояние.

Чтобы избежать неповторяющихся чтений, до завершения транзакции никакая другая транзакция не должна изменять объект A. В большинстве

систем это является максимальным требованием к синхронизации транзакций, однако отсутствие неповторяющихся чтений еще не гарантирует реальной изолированности пользователей.

Сериализация транзакций.

Для обеспечения синхронизации транзакций и изолированности пользователей, в СУБД используются специальные методы обеспечения совместного выполнения транзакций (сериализации транзакций). К таким методам относятся: синхронизационные захваты, гранулированные синхронизационные захваты, метод временных меток и ряд других.

Сериализация транзакций – механизм их выполнения, когда результат параллельного выполнения транзакций эквивалентен результату последовательного выполнения тех же транзакций.

Обеспечение такого механизма является основной функцией компонента СУБД, ответственного за управление транзакциями. Система, в которой поддерживается сериализация транзакций, обеспечивает реальную изолированность пользователей.

Основная реализационная проблема состоит в выборе метода сериализации набора транзакций, который не слишком ограничивал бы их параллельность. Тривиальным решением является действительно последовательное выполнение транзакций. Но существуют ситуации, в которых можно выполнять операторы разных транзакций в любом порядке с сохранением сериальности. Примерами могут служить только читающие транзакции, а также транзакции, не конфликтующие по объектам базы данных.

Между транзакциями могут существовать следующие виды конфликтов:

–W-W - транзакция 2 пытается изменять объект, измененный не закончившейся транзакцией 1;

–R-W - транзакция 2 пытается изменять объект, прочитанный не закончившейся транзакцией 1;

–W-R - транзакция 2 пытается читать объект, измененный не закончившейся транзакцией 1.

Практические методы сериализации транзакций основывается на учете этих конфликтов.

Существуют два базовых подхода к сериализации транзакций - основанный на синхронизационных захватах объектов базы данных и на использовании временных меток. Суть обоих подходов состоит в обнаружении конфликтов транзакций и их устранении.

Синхронизационные захваты.

Наиболее распространенным в централизованных СУБД (включающих системы, основанные на архитектуре "клиент-сервер") является подход, основанный на соблюдении двухфазного протокола синхронизационных захватов объектов базы данных. В общих чертах протокол состоит в том, что

перед выполнением любой операции в транзакции T над объектом базы данных R от имени транзакции T запрашивается разрешение на синхронизационный захват объекта R от имени транзакции в соответствующем режиме (в зависимости от вида операции). При этом выполнение транзакции разбивается на 2 фазы – в первой фазе происходит накопление захватов, во второй фазе – освобождение захватов (фиксация или откат).

Основными режимами синхронизационных захватов являются:

– совместный режим s-shared (общий)– разделяемый захват объекта и необходимый для операции чтения (то есть нескольким транзакциям разрешается чтение одного данного;

– монопольный режим (x- exclusive) – монопольный захват объекта (блокировка для других транзакций), необходимый для выполнения операций изменения. При этом несовместимы захваты на чтение и на запись и захват одного объекта разными транзакциями для записи.

Захваты объектов несколькими транзакциями по чтению совместимы, то есть нескольким транзакциям допускается читать один и тот же объект, захват объекта одной транзакцией по чтению не совместим с захватом другой транзакцией того же объекта по записи, и захваты одного объекта разными транзакциями по записи не совместимы. Транзакция, запросившая синхронизационный захват объекта базы данных, уже захваченный другой транзакцией в несовместимом режиме, блокируется до тех пор, пока захват с этого объекта не будет снят.

Для обеспечения сериализации транзакций синхронизационные захваты объектов, произведенные по инициативе транзакции, можно снимать только при ее завершении. Это требование порождает, указанный выше, протокол синхронизационных захватов –2PL.

Гранулированные синхронизационные захваты.

При применении этого метода синхронизационные захваты запрашиваются по отношению к объектам разного уровня: файлам, отношениям и кортежам. Требуемый уровень объекта определяется тем, какая операция выполняется (например, для выполнения операции уничтожения отношения объектом синхронизационного захвата должно быть все отношение, а для выполнения операции удаления кортежа - этот кортеж). Объект любого уровня может быть захвачен в режиме *S* или *X*.

Метод временных меток.

Альтернативный метод сериализации транзакций состоит в следующем: если транзакция *T1* началась раньше транзакции *T2*, то система обеспечивает такой режим выполнения, как если бы *T1* была целиком выполнена до начала *T2*.

Для этого каждой транзакции *T* предписывается временная метка *t*, соответствующая времени начала *T*. При выполнении операции над объектом *r*

транзакция *T* помечает его своей временной меткой и типом операции (чтение или изменение).

Перед выполнением операции над объектом *R* транзакция *T1* выполняет следующие действия:

–проверяет, не закончилась ли транзакция *T*, пометившая этот объект. Если *T* закончилась, *T1* помечает объект *r* и выполняет свою операцию;

–если транзакция *T* не завершилась, то *T1* проверяет конфликтность операций. Если операции неконфликтны, при объекте *R* остается или проставляется временная метка с меньшим значением, и транзакция *T1* выполняет свою операцию;

–если операции *T1* и *T* конфликтуют, то если *t(T) > t(T1)* (т.е. транзакция *T* является более "молодой", чем *T*), производится откат *T* и *T1* продолжает работу;

–если *же t(T) < t(T1) (T "старше" T1)*, то *T1* получает новую временную метку и начинается заново.

4.4 Технологии и средства доступа к распределенным базам данных

В качестве приложений обеспечивающих сервер приложений используют два вила программного обеспечения:

–ПО взаимодействия и доступа к базам данных ODBC – технологии, SQL шлюзы, OLE DB;

–ПО межмодульного взаимодействия – мониторы обработки транзакций, вызов удаленных процедур.

SQL-шлюзы.

В многозвенных системах при доступе к разнотипным СУБД, между клиентом и сервером размещается промежуточное звено SQL-шлюз, с помощью которых строятся общие запросы к разнородным данным, шлюз синтаксически анализирует запрос, оптимизирует его и преобразует в язык понятный нужной СУБД.

Технологии реализуются на мониторах обработки транзакций, которые обеспечивают автоматизированную поддержку приложений для обработки транзакций (управление информационно-вычислительными ресурсами в распределенной системе).

Технология SQL-DMO (SQL Distributed Management Objects). Технология – распределенного управления объектами SQL позволяет приложениям, написанным на языках программирования, поддерживающих OLE или COM, выполнять администрирование всеми компонентами SQL Server. SQL-DMO представляет собой интерфейс прикладного программирования API, используемый SQL Server Enterprise Manager. Поэтому приложения, используя

SQL-DMO, могут выполнять все функции, выполняемые программой SQL Server Enterprise Manager.

При использовании технологии SQL-DMO работа ведется с объектами, коллекциями, свойствами и методами объекта SQL-DMO. Кроме того, возможно выполнение и посылка на сервер команд SQL Server с помощью вызова методов ExecuteImmediate и ExecuteWithResults

Интерфейс DB -Library представляет собой специализированную библиотеку функций API, осуществляющих прямой доступ к SQL Server из среды систем программирования C++ и Visual Basic. Они предоставляют средства отправки запросов и получения информации от SQL Server,

Работа с библиотекой DB-LIB предполагает следующую стандартную последовательность действий: регистрацию в системе (установление соединения), выполнение нескольких действий на сервере и отключение сервера (закрытие соединения).

DB-LIB представляет собой специально предназначенный для SQL Server интерфейс прикладных программ и прикладного программирования. Поэтому он является наименее мобильным из числа рассматриваемых средств доступа, в смысле возможностей переноса приложений в другую серверную среду. С точки зрения производительности этот способ позволяет осуществить самый быстрый доступ к информации. Причиной этого является то, что он предоставляет оптимизированный интерфейс прикладного программирования и непосредственно использует язык запросов в систему SQL Server.

Уровень прикладного программирования DB-LIB не используется непосредственно в среде персональных СУБД, таких как Access и Visual FoxPro. В них используются промежуточные абстрактные уровни ODBC или OLE DB, упрощающие доступ к базам данных.

Технология DB-LIB использована в большом числе разработанных ранее информационных систем. В настоящее время эта технология не поддерживается фирмой Microsoft и вместо нее рекомендуется использовать средства OLE DB.

Архитектура и технология ODBC.

Технология ODBC разработана фирмой Microsoft для обеспечения возможности взаимосвязи между различными СУБД.

В двухзвенных моделях клиент-сервер, где несколько баз данных размещенных на удаленных серверах обслуживают ограниченное число пользователей (доступ к данным на удаленных серверах), в роли программного обеспечения используют ODBC драйверы (Open Date Base Connecting). Эта технология предусматривает создание дополнительного уровня между приложением и используемой СУБД. Основное назначение ODBC состоит в абстрагировании приложения от особенностей ядра серверной базы данных, с которой оно осуществляет взаимодействие, поэтому серверная база данных становится как бы прозрачной для любого клиентского приложения.

Службы ODBC обеспечивают получение от приложения запросов на выборку информации, перевод их на язык ядра адресуемой базы данных для доступа к хранимой в ней информации (рис. 4.9).

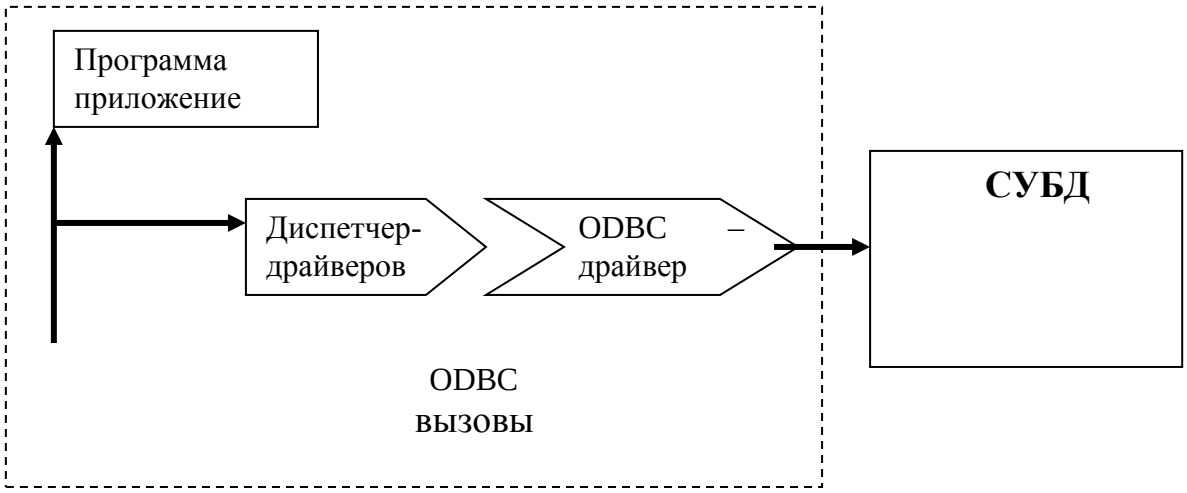


Рис. 4.9 – Архитектура ODBS.

Принцип работы ODBC:

– программа приложение непосредственно взаимодействует с диспетчером драйверов, посылая ему ODBC вызовы;

–диспетчер драйверов обеспечивает загрузку нужного ODBC драйвера, через который обращается к СУБД;

–ODBC драйвер выполняет все вызовы и переводит их на язык источника данных (ЯИД);

–СУБД хранит и выводит данные в ответ на запрос ODBC драйвера.

Технология ODBC представляет собой виртуальный источник данных, которым можно управлять с помощью SQL команд.

Задание ODBC источника данных осуществляется с помощью Администратора источника данных DSN (date source name).

ODBC драйвер взаимодействует с СУБД на высоком уровне и пользовательское приложение этого не видят. Соответственно приложение становится независимым от СУБД, но необходима инсталляция и настройка ODBC драйверов на каждом рабочем месте.

Достоинства технологии ODBC – простота разработки приложений, обусловленная высоким уровнем интерфейса, который обеспечивает доступ к данным любых типов СУБД и создание виртуального источника данных связанного с любым типом баз данных. Используя эту технологию, можно создавать клиент-серверные приложения, причем средствами персональных СУБД целесообразно разрабатывать клиентскую часть приложения, а средствами SQL Server - серверную часть.

Недостаток технологии – необходимость трансляции запросов в ЯИД, что снижает скорость доступа к данным. В системах клиент-сервер этот недостаток устраняется путем перемещения обработки запроса с компьютера-клиента на компьютер-сервер. При этом устраняются промежуточные звенья, являющиеся

основной причиной снижения скорости обработки информации с использованием средств рассматриваемой технологии.

Архитектура и технология OLE DB.

OLE DB – встраивание и связывание объектов в базах данных, используется для реализации трехзвенной архитектуры «сервер приложений» и предназначена для организации доступа к данным размещенных на удаленных серверах с использованием SQL.

OLE DB представляет собой интерфейс прикладного программирования API, который позволяет приложениям COM (Component Object Model объектная модель компонентов) потреблять данные из источника данных OLE DB. Источник данных OLE DB включает данные, хранимые в различных форматах, не только в формате баз данных SQL.

OLE DB обеспечивает взаимодействие и доступ к данным через провайдеров – поставщиков данных независимо от типа источника данных.

OLE DB включает провайдер данных и потребитель данных. Потребитель данных – это приложение или COM компонент (объектная модель компонентов) обращающийся к OLE DBю

Провайдер – сервер приложений, обеспечивает доступ к источнику данных OLE DB, отвечает на вызовы OLE DB и возвращает запрашиваемый объект. Провайдер OLE DB представляет собой компонент COM, позволяющий принимать вызовы OLE DB API и выполнять все необходимое для обработки запроса к источнику данных.

Для работы с OLE DB используется универсальный интерфейс высокого уровня ADO – как язык программирования баз данных. Основным объектом является объект – *Соединение*, который создает связь с провайдером данных; объект – *Набор данных* и объект *Команда* – выполнение процедуры.

В ранних версиях SQL Server приложения OLE DB должны были использовать провайдер OLE DB для ODBC, основанном на драйвере Microsoft SQL Server ODBC. Поскольку приложения по-прежнему могут использовать провайдер OLE DB для ODBC, поэтому более эффективно использование только провайдера OLE DB для SQL Server.

Интерфейс прикладного программирования OLE DB рекомендуется использовать для создания средств и утилит, или разработок системного уровня, нуждающихся в высокой производительности или доступе к свойствам SQL Server, недоступным с помощью технологии ADO. Основные возможности спецификации OLE DB обеспечивают полную функциональность доступа к данным, требуемую большинством приложений. Кроме того, OLE DB допускает индивидуальным провайдерам определять специфические механизмы поддержки дополнительных свойств процессора данных, доступного провайдеру. Приложения ADO могут не иметь доступа к некотором свойствам SQL Server, проявляемым через специфические свойства провайдера OLE DB для SQL Server. Поэтому приложения, желающие воспользоваться такими специфическими свойствами, должны использовать OLE DB API.

В SQL Server 7.0 процессор баз данных сервера использует OLE DB для связи: между внутренними компонентами, такими как процессор хранения и процессор отношений; между установками SQL Server при использовании удаленных хранимых процедур; как интерфейс к другим источникам данных для распределенных запросов.

Технология DAO определяет использование объектов, методов и свойств, которые существенно упрощают работу приложения с базой данных. Для обмена информацией с SQL Server применяются уровни доступа процессора баз данных Jet SQL Server и ODBC. Кроме того, при этом создается еще один уровень абстракции между приложением и функциями ODBC, используемыми при выполнении запросов.

При использовании технологии DAO работа с базами данных, таблицами, представлениями ведется с использованием коллекций объектов. При этом обеспечиваются большие удобства в работе с объектами баз данных. Например, для создания представления проще вызвать метод Add соответствующего представлению объекта, чем применять стандартные средства ODBC с указанием имен используемых для этих целей хранимых процедур.

При работе с базами данных технология DAO применима для большинства источников данных, поддерживаемых средствами ODBC. Его можно прямить также на других платформах и в системах программирования. В частности, технология DAO поддерживается версиями Visual I Basic.

В настоящее время технология DAO постепенно вытесняется технологией ADO, которая позволяет разрабатывать приложения Web для работы с базами данных.

Технология ADO основана на объектной модели, в которой объекты имеют наборы коллекций, методов и свойств, обеспечивающие поддержку баз данных. Объекты этой технологии предоставляют наиболее широкие возможности по интеграции приложений с базами данных. С использованием технологии ADO приложения для работы с базами данных можно создавать в различных системах программирования, например Visual Basic и Visual Basic for Application. Кроме того, объекты можно использовать при разработке Web - приложений для среды ASP (Active Server Pages - активных серверных страниц), или приложений ASP.

Объекты ADO доступны в среде ASP и функционируют на уровне OLE DB. При этом технология ADO обеспечивает установление соединений ODBC и работу с уровнем OLE DB. При организации доступа к базе данных из приложения ASP инициатором установки соединения с сервером базы данных является клиентское приложение. Причем приложения ASP выполняются и в среде Internet Information Server, а не на SQL Server. В целом, технологию ADO можно охарактеризовать как наиболее современную технологию разработки приложений для работы с распределенными базами по технологии клиент-сервер.

Технологии межмодульного взаимодействия.

Технологии ориентированы на архитектуру приложения, в которой один прикладной модуль, используя специальные протоколы, получают данные из другого модуля. Используется в сложной распределенной среде (тысячи клиентских мест, десятки источников данных).

Технологии реализуются на мониторах обработки транзакций (Transaction Processing Monitor, TP мониторы) – программные средства системы, обеспечивающие эффективное управление информационно-вычислительными ресурсами в распределенной системе. Они представляют собой гибкую, открытую среду для разработки и управления мобильными приложениями, ориентированными на оперативную обработку распределенных транзакций и обеспечивают координацию и управление транзакций, автоматизированную поддержку приложений для обработки транзакций (управление информационно-вычислительными ресурсами в распределенной системе).

Средства обработки транзакций обеспечивают изолированность пользователей друг от друга (принцип монопольной работы), целостность и согласованность состояния базы данных для любого пользователя.

В системе TP мониторов эти свойства и условия обеспечивают серверы распределенной базы данных, использующие двухфазный протокол, в котором перед началом распределенной транзакции все системы опрашиваются о готовности выполнить необходимые действия. Если каждый из серверов дает утвердительный ответ, транзакция выполняется. Если происходит сбой в любом месте, будет выполнен откат для всех частей транзакции.

Использование мониторов обработки транзакций является одним из методов достижения более высокой производительности для имеющейся конфигурации, особенно в режиме клиент-сервер. TP-мониторы представляют собой промежуточный слой программного обеспечения, который располагается между приложением и системой или системами СУБД. При этом приложение должно быть модифицировано так, чтобы оно могло выдавать транзакции, написанные на языке монитора транзакций, а не обращаться прямо к базе данных посредством обычных механизмов (подобных различным формам встроенного SQL). Программисты прикладных систем являются также ответственными за составление файла описания, который отображает транзакции в определенные обращения к базе данных народном языке обращений нижележащей СУБД (почти для всех СУБД под Unix это SQL).

TP мониторы при обращении клиента к серверу базы данных координируют и управляют соединением между клиентом и сервером, транзакциями, которые обращаются к серверам баз данных разных поставщиков, распределяет, планирует и выделяет приоритеты запросам от нескольких приложений одновременно.

TP монитор реализуется на основе транзакционной архитектуры, которая определяет интерфейс взаимодействия TP монитора с менеджерами ресурса СУБД – XA. Спецификация XA является частью общего стандарта распределенной обработки транзакций.

Использование мониторов транзакций практически не накладывает каких-либо ограничений на многообразие или сложность запросов доступа к нижележащей СУБД. Помимо достижения определенной гибкости за счет использования ТР-мониторов такая организация оказывается выгодной и с точки зрения увеличения производительности системы. ТР-монитор всегда представляет собой многопотоковую программу. Поскольку ТР-монитор открывает свое собственное соединение с СУБД, одновременно устраняя необходимость выполнения каждым прикладным процессом прямых запросов к СУБД, число одновременно работающих пользователей СУБД существенно сокращается. В подавляющем большинстве случаев СУБД обслуживает только одного "пользователя"- ТР-монитор.

ТР-мониторы позволяют также улучшить производительность за счет сокращения объема информации, пересылаемой между СУБД и прикладным процессом. Обработка запросов организуется в виде нитей операционной системы, а не в виде полноценных процессов. Поскольку определенную часть каждой транзакции составляют только минимально требуемые данные, общий объем пересылки данных обычно может быть сокращен. Это особенно важно, когда клиент и сервер соединены между собой посредством достаточно занятой сети и/или сети с относительно узкой полосой пропускания, например глобальной сети на спутниковых каналах связи.

В числе важнейших характеристик ТР мониторов – маштабируемость, поддержка функциональной полноты и целостности приложений, достижение максимальной производительности при обработке данных при невысоких стоимостных показателях, поддержка целостности данных в гетерогенной среде.

Глава 5. OLAP - системы и интеллектуальные информационные системы

5.1 Хранилища данных и OLAP - системы

5.1.1 Системы поддержки принятия решений

Эффективность функционирования любой системы в существенной, а в большинстве случаев, и определяющей степени зависит от эффективности используемой ей системы управления. При этом особую роль управление приобретает при работе со сложными динамическими системами, к которым в в частности относятся информационные системы и системы управления предприятием. Современные подходы к задачам управления сложными динамическими системами выдвигают ряд новых требований, часто являющиеся достаточно противоречивыми. Важнейшими их них являются:

–сверхбольшой объем данных;
– разнородность данных;
– глубина анализа;
– интерпретируемость данных;
– доступность (простота) инструментария.

Сверхбольшой объем данных связан с проблемой больших данных Big Data. Человечество накопило огромные массивы цифровых данных, однако они ничего не дают без соответствующих технологий извлечения из них полезных знаний. Разнородность данных, связанное с большим количеством плохо структурированной информации (тексты, -аудио, -видео, рисунки и т.п.) привела к проблеме Data Fusion – слияния данных и приведение их к формам, доступным для автоматического анализа данных.

Базовая функциональная и обеспечивающая структура «традиционной» информационной системе на базе оперативной передачи и анализа данных OLTP (Online Transactions Proceeding), оказалась недостаточно эффективна на собственно анализе данных и особенно анализа многомерных данных, которые накапливаются в результате долговременного функционирования любого предприятия, и принятия управленческих решений на основе анализа данных.

Для решения проблемы «правильного» анализа для принятия управленческих решений используют системы поддержки принятия решений на основе когнитивного управления и когнитивных информационных технологий, которые ориентированы на создание качественно новых систем принятия решений, основанных на знаниях и принципах искусственного интеллекта (см. раздел 5.2).

Составной частью информационной системы управления предприятием (корпоративной информационной системы (см. глава 1) является система поддержки принятия решений, которая используется для автоматизации процесса выработки управляющих решений (рис.5.1).

Рис. 5.1 – Общая схема управления с явно выделенной системой выработки управляющих решений.

В частности, базовая структура информационной системы дополняется информационным хранилищем данных и автоматизированной системой поддержки принятия решений (рис.5.2) на основе OLAP – систем и технологий.

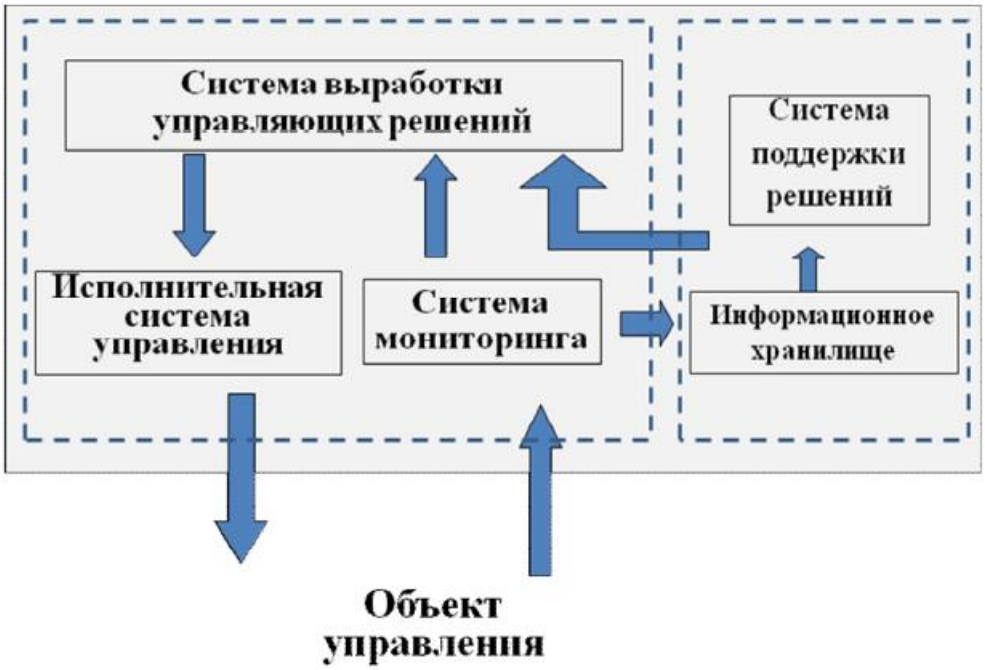


Рис.5.2 – Структура когнитивного управления.

Для задач СППР свойственны недостаточность имеющейся информации, ее противоречивость и нечеткость, преобладание качественных оценок целей и ограничений, слабая формализуемость алгоритмов решения. В качестве инструментов обобщения чаще всего используются средства составления аналитических отчетов произвольной формы, методы статистического анализа, экспертных оценок и систем, математического и имитационного моделирования. При этом применяются базы обобщенной информации, информационные хранилища, базы знаний о правилах и моделях принятия решений.

Системы поддержки принятия решений (DSS) - это компьютерные, интерактивные системы, разработанные, чтобы помочь руководителю в принятии решений. DSS включают и данные, и модели, чтобы помочь принимающему решения решить проблемы, особенно те, которые плохо формализованы. Данные часто извлекаются из системы диалоговой обработки запросов или базы данных. Модель может быть простой типа "доходы и убытки", чтобы вычислить прибыль при некоторых предположениях, или комплексной типа оптимизационной модели для расчета управления запасами или загрузки оборудования.

Структура системы поддержки принятия решений состоит из следующих подсистем (рис.5.3):

Подсистема данных содержит информацию, необходимую для принятия решений, часто широкий набор данных из внутренних и внешних источников в организации. Подсистема данных содержит всю доступную информацию о возможных видах собственности, включая описание, текущее значение, расходы и доходы.

Подсистема моделей определяет структуру процесса принятия решения и включает формулы, соотношения, логические сравнения и другие блоки для соответствующей обработки информации. Разработчик часто увлекается структурой модели решения в ущерб простоте, модульному подходу.

Подсистема модели данной системы должна состоять из алгоритмов расчета, которые представляют собой каждый объект собственности в терминах близости предпочтения клиента по таким параметрам как цена, местоположение, размер чистого дохода. Например, если цена объекта инвестиций находится в требуемом диапазоне и имеет для клиента наибольшее значение, то данный объект инвестиций получает 100 пунктов в списке предпочтений.

Подсистема диалогового управления занимается представление технологии и реализации требуемого пользовательского интерфейса.

Подсистема управления данными занимается хранением, извлечением и проверкой информации, необходимой для принятия решения.

Система управления моделью явно определяет взаимоотношения между переменными в решении и выполняет роль сценариев «А что если?». В системе DSS данная модель реализует сравнение значений в базе данных собственности со значениями, выбранными клиентом. Модель состоит из последовательности процедур обработки событий, которые реагируют на действия пользователя.

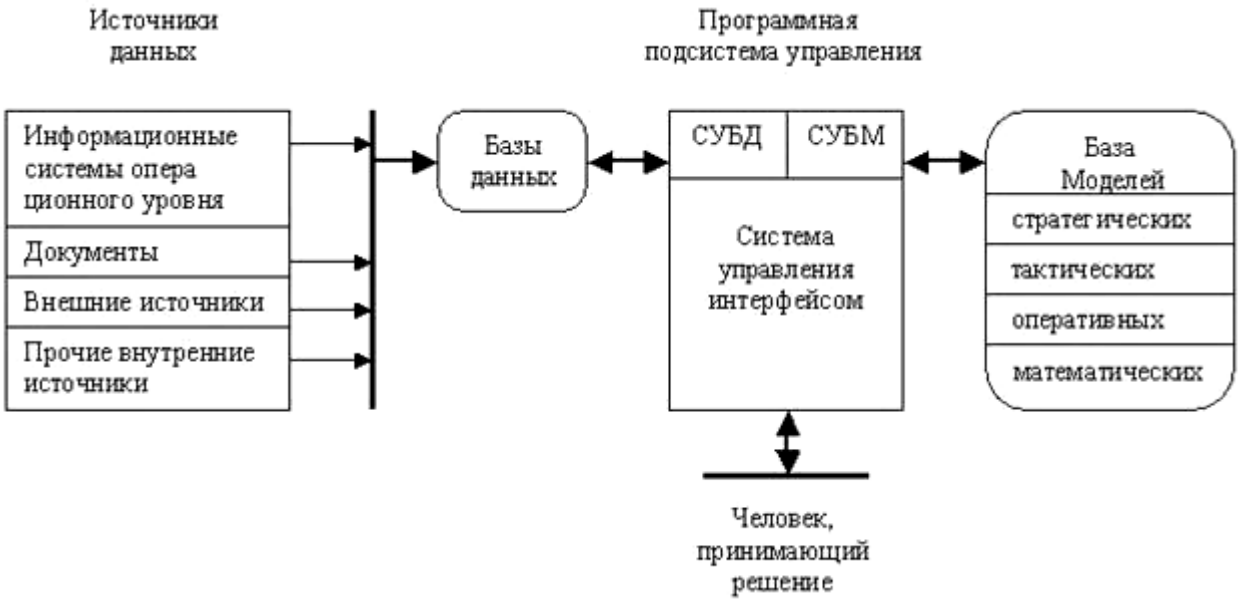


Рис.5.3 – Структура компоненты информационной системы поддержки принятия решений.

Система поддержки принятия решений требует трех первичных компонентов: модели управления, управления данными для сбора и ручной обработки данных и управления диалогом для облегчения доступа пользователя к DSS. Пользователь взаимодействует с DSS через пользовательский интерфейс, выбирая частную модель и набор данных, которые нужно использовать, а затем DSS представляют результаты пользователю через тот же самый пользовательский интерфейс. Модель управления и управление данными в значительной степени действуют незаметно и варьируются от относительно простой типовой модели в электронной таблице до сложной комплексной модели планирования, основанной на математическом программировании.

По критерию сложность решаемых задач и область применения все СППР можно разделить на три класса.

СППР первого класса обладают самыми большими функциональными возможностями. Они предназначены для использования в структурах государственного управления самого высокого уровня, в структурах управления крупных компаний при составлении планов реализаций комплексных целевых программ, применяются для обоснования решений, какие мероприятия должны быть включены в программу и как между ними должны быть распределены ресурсы на основании оценки влияния мероприятий на достижение конечной цели программы. СППР первого класса являются системами принятия решений совместного *коллективного пользования,* для таких систем базы знаний формируются многочисленными экспертами, являющимися специалистами в разных областях знаний.

Базы знаний для ***СППР второго класса***. формируются самим пользователем и они являются системами *индивидуального пользования.* Такие СППР предназначены для работы служащих среднего ранга, администраторов небольших огранизаций для решения оперативных задач управления.

Системы принятия решений, адаптирующиеся к опыту пользователя, выделяются в ***СППР третьего класса.*** Это системы индивидуального пользования, предназначенные для решения довольно часто встречающихся задач системного анализа и оперативного управления (в качестве примера можно привести выбор субъекта кредитования, выбор претендента на должность, выбор исполнителя работы и т.д.). Такие СППР обеспечивают получение решения конкретной задачи, ориентируясь на информацию о результатах практического применения принятых в прошлом решений этой же задачи.

Любое конкурентоспособное производство основывается на новейших достижениях и поэтому достаточно легко переориентируется на еще более совершенные технологии. Менеджеру любого ранга полезно обеспечить необходимую помощь для выработки и обоснования решений, которые будут адекватны изменяющимся условиям функционирования для управляемых ими систем. СППР являются мощным инструментарием для разработки альтернативных вариантов действий, последующего анализа последствий их внедрения и рационализации навыков руководителя в принятии решений, которая является одной из важнейших областей его деятельности.

5.1.2 Хранилища данных

Хранилище данных - это предметно-ориентированное, крупномасштабное, неизменчивое, поддерживающие хронологию (привязанное ко времени) собрание данных. Хранилища данных содержат заведомо избыточную информацию.

Витрина данных – упрощенный вариант хранилища данных, содержащий только тематически объединенные данные. Витрину данных часто формируют как надстройку над более общим хранилищем данных.

В отличие от базы данных, хранилище данных (информационное хранилище) представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного ситуационного анализа данных – реализации OLAP-технологии или технологий интеллектуального анализа данных (см. раздел 5.3).

Типичными задачами оперативного ситуационного анализа являются: определение профиля потребителей конкретных объектов хранения; предсказание изменений объектов хранения во времени; анализ зависимостей признаков ситуаций (корреляционно-регрессионный анализ).

Данные в хранилище попадают из оперативных систем (OLTP-систем), которые предназначены для автоматизации бизнес-процессов. Кроме того, хранилище может пополняться за счет внешних источников, например статистических отчетов, различных справочников и т.д. Хранилище данных кроме детализированной информации содержит в себе агрегаты, то есть

обобщающую информацию, например суммы продаж, количество, общие расходы и т.д.

Использование хранилищ данных обусловлено тем, что анализировать данные оперативных систем какого-либо крупного предприятия напрямую невозможно или очень затруднительно. Это объясняется различными причинами, в том числе разрозненностью данных и хранением их в форматах различных СУБД. Но даже если все данные хранятся на центральном сервере базы данных, аналитику бывает почти невозможно разобраться обычными методами в сложных структурах используемых данных.

Технологии хранилища данных обеспечивают сбор данных из существующих внутренних баз предприятия и внешних источников, формирование, хранение и эксплуатацию информации как единого целого, хранение аналитических данных (знаний) в форме, удобной для анализа и принятия управленческих решений. К внутренним базам данных предприятия относятся локальные базы подсистем информационной системы (бухгалтерский учет, финансовый анализ, кадры, расчеты с поставщиками и покупателями и т. д.). К внешним базам относятся любые данные, доступные через сеть Интернет и размещенные на web-серверах правительственных и законодательных органов, других учреждений.

Основное назначение хранилищ данных – обеспечение менеджеров и аналитиков достоверной информацией необходимой для оперативного анализа и поддержки процесса принятия управляющих решений. То есть выполнение сложных аналитических запросов к оперативной информации и предоставление разрозненных данных, "сырья" для анализа в относительно простой и понятной форме.

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.5.4).

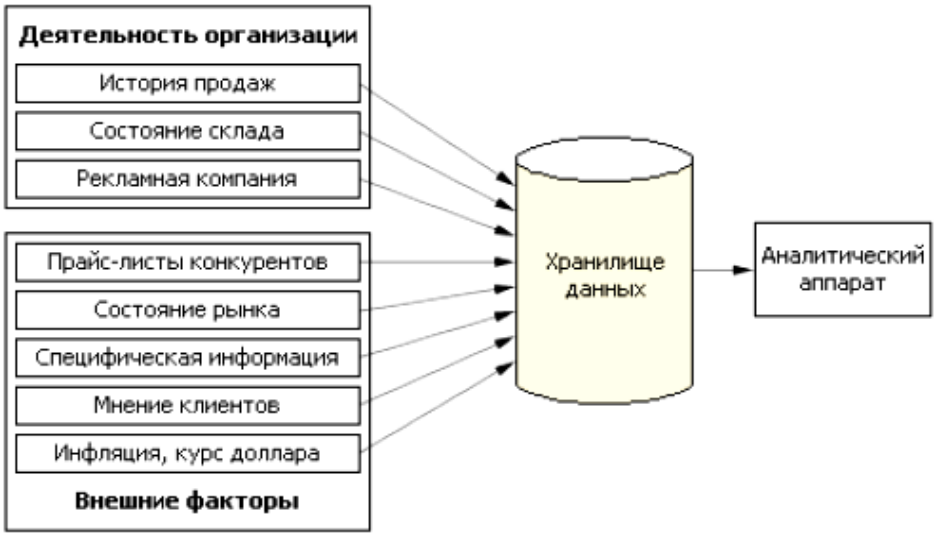


Рис.5.4 – Систематизированная информация в хранилище данных.

В основе концепции хранилищ данных лежат следующие принципы:

–интеграция ранее разъединенных детализированных данных в едином хранилище данных, их согласование и агрегация, в том числе данные: исторических архивов; данных из традиционных систем обработки данных; данных из внешних источников;

–разделение наборов данных, используемых для операционной обработки, и наборов данных, применяемых для решения задач анализа;

– хранилище данных обеспечивает не анализ данных, а подготовку данных для анализа;

– хранилище данных определяет, какие процессы должны выполняться в системе, но не где конкретно и как эти процессы должны выполняться;

– хранилище данных предполагает не просто единый логический взгляд на данные организации, а реализацию единого интегрированного источника данных.

Основные требования к данным, помещаемых в целевую базу данных хранилища данных следующие:

–*предметная ориентированность* – все данные о некотором предмете (бизнес-объекте) собираются (обычно из множества различных источников), очищаются, согласовываются, дополняются, агрегируются и представляются в единой, удобной для их использования в бизнес-анализе форме;

–*интегрированность* –все данные о разных бизнес-объектах взаимно согласованы и хранятся в едином общекорпоративном хранилище;

–*неизменчивость* –исходные (исторические) данные, после того как они были согласованы, верифицированы и внесены в общекорпоративное хранилище, остаются неизменными и используются исключительно в режиме чтения;

–*поддержка хронологии* –данные хронологически структурированы и отражают историю, за достаточный для выполнения задач бизнес-анализа и прогнозирования период времени.

Данные в хранилищах данных рассматриваются как самостоятельный объект предметной области, порожденной в результате функционирования ранее созданных информационных систем (производственного процесса) и являются продуктом производства, обладают сроком годности, местом складирования (хранения), совместимостью с данными из других производств, рыночной стоимостью, транспортабельностью, комплектностью, ремонтопригодностью и т. д.. То есть теми же свойствами и характеристиками, что и любой промышленный продукт.

Свойства хранилищ (складов) данных:

–эффективное хранение и обработка очень больших массивов информации;

–неоднородность программной среды – функционирование системы на основе неоднородных программных средств и решений и их совместимость;

–формирование интегрированного согласованного набора данных, которые могут поступать из разнородных баз данных, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников.

–распределенный характер организации;

–распределенный характер обработки данных – операционная аналитическая обработка может выполняться в любом узле сети независимо от места расположения основного хранилища;

–репликация и синхронизация данных;

–наличие многоуровневых справочников метаданных;

–повышенные требования к безопасности данных; доступ к данным контролируется на уровне отдельных строк (это соответствует классу B1 Оранжевой Книги). Обычные коммерческие СУБД используют защиту данных в стиле языка SQL на уровне таблиц и их столбцов соответствует классу C2.

Принципы управления хранилищами данных те же, что и СУБД.

Обычно выделяют следующие основные свойства, которыми должно обладать хранилище данных:

–неоднородность программной среды;

–распределенный характер организации;

–повышенные требования к безопасности данных;

–необходимость наличия многоуровневых справочников метаданных;

–потребность в эффективном хранении и обработке очень больших объемов информации.

Хранилище данных практически никогда не создается на пустом месте. Почти всегда конечное решение будет разнородным, то есть в нем будут использоваться автономно разработанные программные средства. Прежде всего это касается формирования интегрированного согласованного набора данных,

которые могут поступать из разнородных баз данных, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников. При построении склада данных приходится решать задачу построения единой, согласованно функционирующей информационной системы на основе неоднородных программных средств и решений. При выборе средств реализации склада данных приходится учитывать множество факторов, включающих уровень совместимости различных программных компонентов, легкость их освоения и использования, эффективность функционирования и т.д.

В концепции хранилище данных предопределено то, что операционная аналитическая обработка может выполняться в любом узле сети независимо от места расположения основного хранилища. Хотя при аналитической обработке данные только читаются, и потребность в синхронизации отсутствует, для достижения эффективности необходимо поддерживать репликацию данных в разных узлах сети. (На самом деле, все не так просто. Одним из требований к складам данных является то, чтобы свежая информация поступала на склад как можно быстрее. То есть потенциально любая модификация оперативной базы данных может инициировать добавление данных к складу данных, а тогда потребуется обновить и все реплики, для чего синхронизация все-таки нужна).

Собранная вместе согласованная информация об истории развития корпорации, ее успехах и неудачах, о взаимоотношениях с поставщиками и заказчиками, об истории и состоянии рынка дает возможность анализа прошлой и текущей деятельности корпорации и построения прогнозов для будущего. Эта информация настолько ценна для корпорации, что нельзя допустить возможности ее утечки (на самом деле, если склад данных одной корпорации попадет в руки аналитиков другой корпорации, то все аналитические прогнозы первой корпорации сразу станут неверными). В системах, основанных на складах данных, оказывается недостаточной защита данных в стиле языка SQL, которую обеспечивают обычные коммерческие СУБД (этот уровень защиты соответствует классу С2 в соответствии с классификацией Оранжевой Книги). Для обеспечения должного уровня защиты доступ к данным должен контролироваться не только на уровне таблиц и их столбцов, но и на уровне отдельных строк (это уже соответствует классу В1 Оранжевой Книги). Приходится также решать вопросы аутентификации пользователей, защиты данных при их перемещении в склад данных из оперативных баз данных и внешних источников, защиты данных при их передаче по сети.

Если роль метаданных (обычно содержащихся в таблицах-каталогах) в оперативных информационных системах достаточно ограничена, то для OLAP-систем наличие развитых метаданных и средств их предоставления конечным пользователям является одним из основных условий успешной реализации. Например, прежде, чем менеджер корпорации задаст системе свой вопрос, он должен понять, какая информация имеется, насколько она актуальна, можно ли ей доверять, сколько времени может занять формирование ответа и т.д. Для

пользователя OLAP-системы требуются метаданные, по крайней мере, следующих типов:

- Описания структур данных, их взаимосвязей.
- Информация о хранимых на складе данных и поддерживаемых им агрегатах данных.
- Информация об источниках данных и о степени их достоверности. Одна и та же информация могла попасть в склад данных из разных источников. Пользователь должен иметь возможность узнать, какой источник был выбран основным, и каким образом производились согласование и очистка данных.
- Информация о периодичности обновлений данных. Желательно знать не только то, какому моменту времени соответствуют интересующие его данные, но и когда они в следующий раз будут обновлены.
- Информация о владельцах данных. Пользователю OLAP-системы может оказаться полезной информация о наличии в системе данных, к которым он не имеет доступа, о владельцах этих данных и о действиях, которые он должен предпринять, чтобы получить доступ к данным.
- Статистические оценки времени выполнения запросов. До выполнения запроса полезно иметь хотя бы приблизительную оценку времени, которое потребуется для получения ответа, и объема этого ответа.

Известны примеры хранилищ данных, содержащих терабайты информации. По данным консалтинговой компании MetaGroup, около половины корпораций, использующих или планирующих использовать хранилища данных предполагает довести их объем до сотен гигабайт. Проблемой таких больших хранилищ является то, что накладные расходы на внешнюю память возрастают нелинейно при возрастании объема хранилища. Исследования, проведенные на основе тестового набора TPC-D, показали, что для баз данных объемом в 100 гигабайт потребуется внешняя память объемом в 4.87 раза большая, чем нужно собственно для полезных данных. При дальнейшем росте баз данных этот коэффициент увеличивается.

5.1.3 OLAP – системы и технологии

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

Основное назначение OLAP-систем - поддержка аналитической деятельности, произвольных запросов пользователей - аналитиков. Цель OLAP-анализа - проверка возникающих гипотез по характеру и причинам изменения данных для принятия управляющих решений.

В процессе анализа данных часто возникает необходимость построения зависимостей между различными параметрами, число которых может быть

значительным. Под измерением понимается последовательность значений одного из анализируемых параметров. Например, для параметра "время" это - последовательность дней, месяцев, кварталов, лет.

Возможность анализа зависимостей между различными параметрами предполагает возможность представления данных в виде многомерной модели - гиперкуба (рис.5.5), или OLAP-куба.

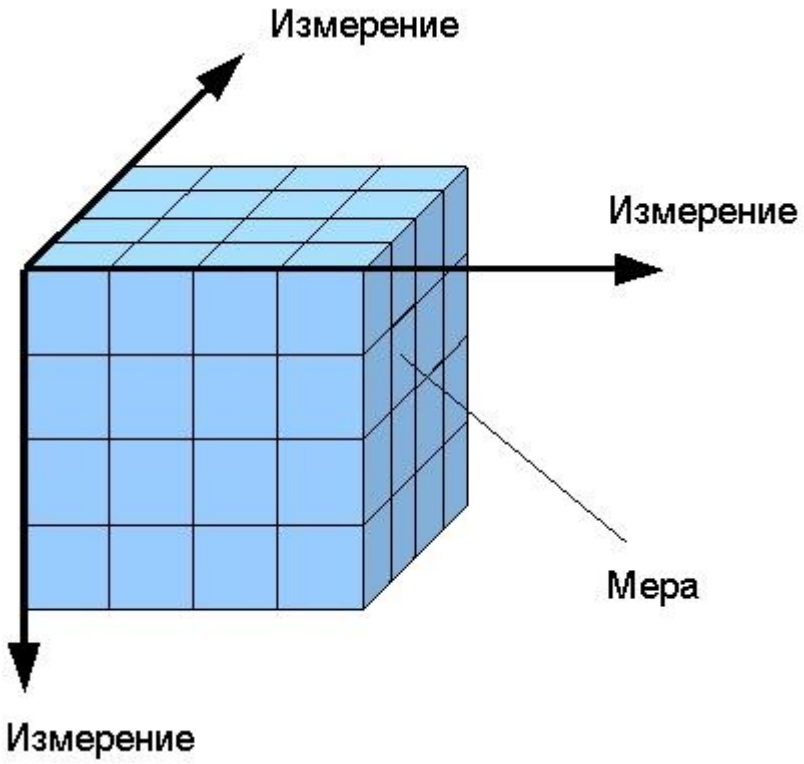


Рис. 5.5 – OLAP-куб.

Оси куба представляют собой измерения, по которым откладывают параметры, относящиеся к анализируемой предметной области, например, названия товаров и названия месяцев года.

На пересечении осей измерений располагаются данные, количественно характеризующие анализируемые факты - меры, например, объемы продаж, выраженные в единицах продукции.

В простейшем случае двумерного куба получается таблица, показывающая значения уровней продаж по товарам и месяцам.

Дальнейшее усложнение модели данных возможно по нескольким направлениям:

1) Увеличение числа измерений данные о продажах не только по месяцам и товарам, но и по регионам. В этом случае куб становится трехмерным.

2) Усложнение содержимого ячейки например, нас может интересовать не только уровень продаж, но и чистая прибыль или остаток на складе. В этом случае в ячейке будет несколько значений.

3) Введение иерархии в пределах одного измерения общее понятие "время" связано с иерархией значений: год состоит из кварталов, квартал из месяцев и т.д.

Каждое из измерений OLAP-куба может быть представлено в виде иерархической структуры. Например, измерение "Регион" может иметь следующие уровни иерархии: "страна - федеральный округ - область - город - район".

Некоторые измерения могут иметь несколько уровней иерархического представления, например измерение "время" - представление "год - квартал - месяц - день" и представление "год - неделя - день".

Точно так же в рамках измерения "География" можно ввести уровни "Страна", "Регион", "Область" и "Город".

Технологии OLAP.

Целью использования технологий OLAP является анализ данных и представление результатов этого анализа в виде, удобном для восприятия управляющим персоналом и принятия на их основе решений.

Требования предъявляемые к средствам реализации OLAP-систем.

1. Многомерное представление данных. Средства должны поддерживать многомерный на концептуальном уровне взгляд на данные.

2. Прозрачность. Это требование заключается в том, что пользователь не должен знать о том, какие конкретные средства используются хранения и обработки данных, как они организованы и откуда они берутся.

3. Доступность. Средства должны сами выбирать источник данных и связываться с ним для формирования ответа на данный запрос.

4. Согласованная производительность. Производительность не должна зависеть от количества измерений в запросе.

5. Поддержка архитектуры «клиент-сервер». Средства должны работать в архитектуре «клиент-сервер».

6. Равноправность всех измерений. Ни одно из измерений не должно быть базовым, все они должны быть равноправными.

7. Динамическая обработка разреженных матриц. Неопределенные значения должны храниться и обрабатываться наиболее эффективными способами.

8. Поддержка многопользовательского режима работы с данными. Все многомерные операции должны единообразно и согласованно применяться к любому числу любых измерений.

9. Поддержка операций на основе различных измерений. Все многомерные операции должны единообразно и согласованно применяться к любому числу любых измерений.

10. Простота манипулирования данными. Средства должны иметь максимально удобный и естественный пользовательский интерфейс.

11. Развитые средства представления данных. Средства должны поддерживать различные способы представления данных.

12. Неограниченное число измерений и уровней агрегации данных.

Полномасштабная OLAP-система должна выполнять сложные и разнообразные функции, включающие сбор данных из различных источников, их согласование, преобразование и загрузку в хранилище, хранение аналитической информации, регламентную отчетность, поддержку произвольных запросов, многомерный анализ и др.

В настоящее время существуют фактические стандарты построения OLAP-систем, основанных на концепции ХД. Эти стандарты опираются на современные исследования и общемировую практику создания хранилищ данных и аналитических систем.

В общем виде архитектура корпоративной OLAP-системы описывается схемой с тремя выделенными слоями (рис.5.6):

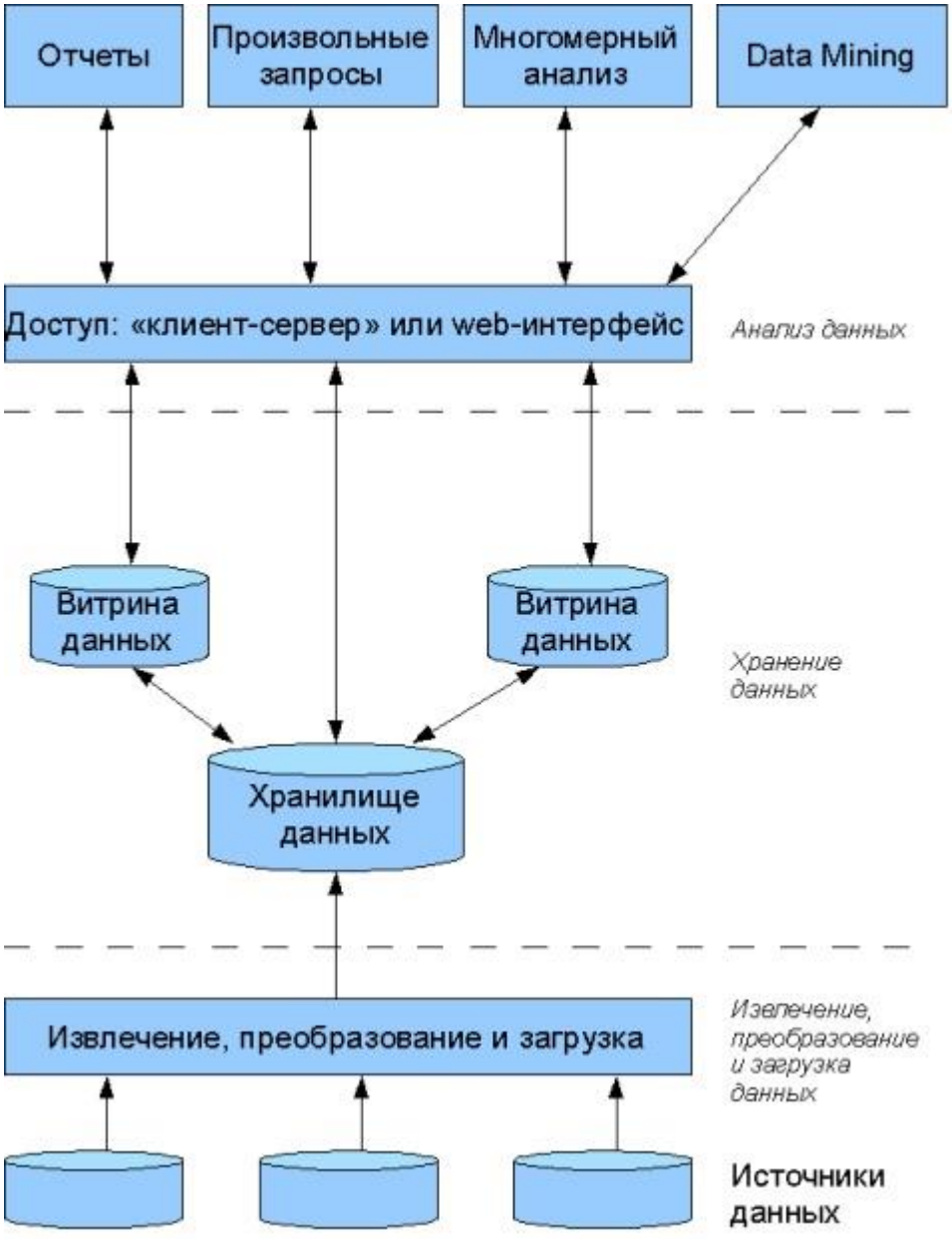


Рис. 5.6 – Архитектура корпоративной OLAP-системы.

–извлечение, преобразование и загрузка данных;

–хранение данных;

–анализ данных.

Данные поступают из различных внутренних OLTP-систем, от подчиненных структур, от внешних организаций в соответствии с установленным регламентом, формами и макетами отчетности. Вся эта информация проверяется, согласуется, преобразуется и помещается в хранилище и витрины данных. После этого пользователи с помощью специализированных инструментальных средств получают необходимую им информацию для построения различных табличных и графических представлений, прогнозирования, моделирования и выполнения других аналитических задач.

Слой извлечения, преобразования и загрузки данных включает подразделения и структуры организации всех уровней, поддерживающие базы данных оперативного доступа. Он представляет собой низовой уровень генерации информации, уровень внутренних и внешних информационных источников, вырабатывающих "сырую" информацию. Эта информация является рабочей для повседневной деятельности различных подразделений, которые ее вырабатывают и используют.

Загрузка данных из источников в хранилище осуществляется специальными процедурами, позволяющими:

–извлекать данные из различных баз данных, текстовых файлов;

–выполнять различные типы согласования и очистки данных;

–преобразовывать данные при перемещении их от источников к хранилищу;

–загружать согласованные и "очищенные" данные в структуры хранилища.

Слой хранения данных **реализуется на основе хранилища данных и** предназначен непосредственно для хранения значимой, проверенной, согласованной, непротиворечивой и хронологически целостной информации, которую с достаточно высокой степенью уверенности можно считать достоверной.

Слой анализа данных. Для организации доступа аналитиков к данным хранилища данных и витрины данных используются специализированные рабочие места, поддерживающие необходимые технологии как оперативного, так и долговременного анализа. Результаты работы аналитиков оформляются в виде отчетов, графиков, рекомендаций и сохраняются как на локальном компьютере, так и в общедоступном узле локальной сети.

Аналитическая деятельность в рамках корпорации достаточно разнообразна и определяется характером решаемых задач, организационными особенностями компании, уровнем и степенью подготовленности аналитиков.

В связи с этим современный подход к инструментальным средствам анализа не ограничивается использованием какой-то одной технологи. В

настоящее время принято различать следующие основные виды аналитической деятельности:

–стандартная отчетность;

–нерегламентированные запросы;

–многомерный анализ (OLAP);

–извлечение знаний (data mining).

Каждая из этих технологий имеет свои особенности, определенный набор типовых задач и должна поддерживаться специализированной инструментальной средой.

Основные виды OLAP – систем.

Аналитические системы OLAP предназначены для анализа больших объемов информации в интерактивном режиме на основе построения многомерных наборов данных – OLAP-кубов для создания аналитических данных, позволяющего руководителю принять обоснованное решение. Они обеспечивают:

–агрегирование и детализацию данных по запросу;

– выдачу данных в терминах предметной области;

–анализ деловой информации по многим параметрам (например, поставщик, его местоположение, поставляемый товар, цены, сроки поставки и т. д.);

– произвольные «срезы» данных по наименованию, выбираемых из внутренних и внешних источников (например, по наименованию товара);

– выполнение аналитических операций с использованием статистических и других методов.

Серверные OLAP-системы развили идею использования кэша для сохранения агрегатных данных. В них сохранение и изменение агрегатных данных, поддержка содержащего их хранилища осуществляются отдельным приложением (процессом), называемым *OLAP-сервером*. Клиентские приложения делают запросы к OLAP-серверу и получают требуемые агрегатные данные. Применение OLAP-серверов сокращает трафик сети и время обслуживания запросов, уменьшает требования к ресурсам клиентских приложений.

В масштабе предприятия обычно используются OLAP-серверы типа Oracle Express Server, MS SQL Server 2000 Analysis Services и др.

Гибридные OLAP-системы представляют собой сочетание инструментов, реализующих реляционную (см. главу 2) и многомерную модели данных. Зона многомерных данных создается специализированными средствами. Это позволяет резко снизить затраты ресурсов на создание и поддержание такой зоны, а время отклика на запросы (в том числе незапланированные) резко снижается, что позволяет обеспечить выполнение требований, предъявляемых к OLAP-системам.

Использование гибридной архитектуры в OLAP-системах — это наиболее приемлемый путь решения проблем, связанных с применением программных инструментальных средств в многомерном анализе.

5.2. Интеллектуальные информационные системы

5.2.1 Искусственный интеллект и нейронные сети

Помимо классических применений информационных систем (глава 1), успешно развивается и такое прогрессивное направление, которое называется «искусственный интеллект». Цель этого направления – создание автоматизированных информационных систем, выполняющих те же функции, что и творчески мыслящая личность, или, по крайней мере, в их простых проявлениях.

Интеллект – это мыслительные способности человека. Отдельные интеллектуальные способности человека могут быть воспроизведены в технических средствах (в том числе и в автоматах) путем создания систем искусственного интеллекта.

Под ***искусственным интеллектом*** (ИИ) обычно понимают способности компьютерных систем к таким действиям, которые назывались бы интеллектуальными, если бы исходили от человека. Чаще всего здесь имеются в виду способности, связанные с человеческим мышлением. Работы в области искусственного интеллекта не ограничиваются экспертными системами. Они также включают в себя создание роботов, систем, моделирующих нервную систему человека, его слух, зрение, обоняние, способность к обучению.

Искусственный интеллект (ИИ) – это свойство автоматических и автоматизированных систем брать на себя отдельные функции человеческого интеллекта, т. е. выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних условий (воздействий).

Основные направления в области ИИ:

1. Символьное (семиотическое, нисходящее), основанное на моделировании высокоуровневых процессов мышления человека, на представлении и использовании знаний.
2. Нейрокибернетическое (нейросетевое, восходящее), основанное на моделировании отдельных низкоуровневых структур мозга (нейронов).

Следовательно, сверхзадачей ИИ является построение компьютерной интеллектуальной системы, которая обладала бы уровнем эффективности решений неформализированных задач, сравнимых с человеческим или превосходящим его. В качестве критерия и конструктивного определения интеллектуальности был предложен мысленный эксперимент, известный как тест Тьюринга.

Наибольшее развитие получили системы искусственного интеллекта, построенные на базе средств компьютерной техники и предназначенные для

восприятия, обработки и хранения информации, а также формирования решений по целесообразному поведению в различных ситуациях, воспроизводящих (модулирующих) состояние некоторой среды (мира, природы, общества, производства и т.п.).

Нейронные сети.

Интеллектуальные системы на основе искусственных нейронных сетей позволяют с успехом решать проблемы распознавания образов, выполнения прогнозов, оптимизации, ассоциативной памяти и управления. Искусственные нейросети являются электронными моделями нейронной структуры мозга, который, главным образом, учится на опыте. Множество проблем, не поддающихся решению традиционными компьютерными методами, могут быть эффективно решены с помощью нейросетей.

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 соединенных 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 соединенными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т.п. С математической точки зрения обучение нейронных сетей – это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования нейронная сеть – способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов. Известные применения:

– *распознавание образов и классификация.* В качестве образов могут выступать различные по своей природе объекты: символы текста, изображения, образцы звуков и т.д. При обучении сети предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор значений признаков. При этом совокупность всех признаков должна *однозначно определять класс,* к которому относится образец. В случае если признаков недостаточно, сеть может соотнести один и тот же образец с несколькими классами, что неверно. По окончании обучения сети ей можно предъявлять неизвестные ранее образы и получать ответ об их принадлежности к определенному классу;

– *принятие решений и управление.* Эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла. При этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы;

– *прогнозирование.* Способности нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких-то существующих в настоящий момент факторов. Следует отметить, что прогнозирование возможно только тогда, когда *предыдущие изменения действительно в какой-то степени предопределяют будущие.* Например, прогнозирование котировок акций на основе котировок за прошлую неделю может оказаться успешным (а может и не оказаться), тогда как прогнозирование результатов завтрашней лотереи на основе данных за последние 50 лет почти наверняка не даст никаких результатов;

– *аппроксимация.* Нейронные сети могут аппроксимировать непрерывные функции. Доказана обобщенная аппроксимационная теорема: с помощью

линейных операций и каскадного соединения можно из произвольного нелинейного элемента получить устройство, вычисляющее любую непрерывную функцию с некоторой наперед заданной точностью. Это означает, что нелинейная характеристика нейрона может быть произвольной: от сигмоидальной до произвольного волнового пакета или вейвлета, синуса или многочлена. От выбора нелинейной функции может зависеть сложность конкретной сети, но с любой нелинейностью сеть остается универсальным аппроксиматором и при правильном выборе структуры может достаточно точно аппроксимировать функционирование любого непрерывного автомата;

– *сжатие данных* и *ассоциативная память.* Способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс – восстановление исходного набора данных из части информации – называется (авто)ассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный

сигнал/образ из зашумленных/поврежденных входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому. 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.

5.2.2 Понятие и классификация ИИС

Интеллектуальная информационная система (ИИС) – автоматизированная информационная система, основанная на знаниях, или комплекс программных, лингвистических, логико-математических средств, , специальных методов и персонала, имеющая возможность хранения, обработки и выдачи информации, а также самостоятельной настройки своих параметров для реализации основной задачи – осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке.

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.

Кроме того, информационно-вычислительными системами с интеллектуальной поддержкой для решения сложных задач называют те системы, в которых логическая обработка информации превалирует над вычислительной.

Таким образом, любая информационная система, решающая интеллектуальную задачу или использующая методы искусственного интеллекта, относится к интеллектуальным.

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.

Система считается ***интеллектуальной,*** если в ней реализованы следующие функции:

1. *Функция представления и обработки знаний.* ИС должна быть способна накапливать знания об окружающем мире, классифицировать и

оценивать их с точки зрения прагматики и непротиворечивости, инициировать процессы получения новых знаний, соотносить новые знания со знаниями, хранящимися в базе знаний.

2. Функция рассуждения. ИС должна быть способна формировать новые знания с помощью логического вывода и механизмов выявления закономерностей в накопленных знаниях, получать обобщенные знания на основе частных знаний и логически планировать свою деятельность.

3. *Функция общения.* ИС должна общаться с человеком на языке, близком *к* естественному языку (ЕЯ), и получать информацию через каналы, аналогичные тем, которые использует человек при восприятии окружающего мира (прежде всего зрительной, звуковой), уметь формировать «для себя» или по просьбе человека объяснения собственной деятельности (т.е. отвечать на вопросы типа «Как я это сделал?»), оказывать человеку помощь за счет знаний, которые хранятся в ее памяти, и логических средств рассуждения.

Основные сферы использования 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 планирование;

–инвестиционный анализа, оценка рисков;

–формирование портфеля ценных бумаг и т.п.

Базовые свойства интеллектуальных информационных систем:

–решение задач, описанных 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 накопленного опыта 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 интеллектуальных информационных систем характерны следующие признаки:

–развитые коммуникативные способности: возможность обработки произвольных запросов в диалоге на языке максимально приближенном к естественному;

–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 интеллектуальных информационных систем 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;

–способность автоматического обнаружения закономерностей бизнеса в ранее накопленных фактах и включения их в базу знаний (т.н. машинное обучение).

Важной отличительной особенностью ИИС является ***естественно-языковой интерфейс*** предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое. Естественно-языковой интерфейс используется для:

− доступа к интеллектуальным базам данных;

− контекстного поиска документальной текстовой информации;

− голосового ввода команд в системах управления;
− машинного перевода с иностранных языков.

Классификация интеллектуальных информационных систем.

Для интеллектуальных информационных систем характерны следующие признаки:

− развитые коммуникативные способности;
− умение решать сложные плохо формализуемые задачи;
− способность к самообучению;
− адаптивность.

Коммуникативные способности ИИС характеризуют способ взаимодействия (интерфейса) конечного пользователя с системой, в частности возможность формулирования произвольного запроса в диалоге с ИИС на языке, максимально приближенном к естественному.

Сложные плохо формализуемые задачи – это задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Способность к самообучению – это возможность автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

Адаптивность – способность к развитию системы в соответствии с объективными изменениями модели проблемной области.

Классификация по коммутативным способностям (системы с интеллектуальным интерфейсом).

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом. В более широком плане сказанное распространяется и на поиск мультимедийной информации, включающей, помимо текстовой, и цифровую информацию.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует, и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам

документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации. Например, состояние сложного управляемого объекта отображается в виде человеческого лица, на котором каждая черта отвечает за какой-либо параметр, а общее выражение лица дает интегрированную характеристику ситуации. Системы когнитивной графики широко используются также в обучающих и тренажерных системах на основе использования принципов виртуальной реальности, когда графические образы моделируют ситуации, в которых обучаемому необходимо принимать решения и выполнять определенные действия.

Классификация по типу решаемых задач.

Экспертные системы предназначены для решения задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области (см. раздел 5.4).

Многоагентные системы – это динамические системы, для которых характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе. Для многоагентных систем характерны следующие особенности:

− проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий;

− распределенное решение проблем, которые разбиваются на параллельно решаемые подпроблемы, соответствующие самостоятельным источникам знаний;

− применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы;

− обработка больших массивов данных, содержащихся в базе данных;

− использование различных математических моделей и внешних процедур, хранимых в базе моделей;

− способность прерывания решения задач в связи с необходимостью получения дополнительных данных и знаний от пользователей, моделей, параллельно решаемых подпроблем.

Классификация по способности к самообучению.

В основе ***самообучающихся систем*** лежат методы автоматической классификации примеров ситуаций реальной практики.

Характерными признаками самообучающихся систем являются:

− самообучающиеся системы «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);

− самообучающиеся системы «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

Индуктивные системы используют обобщение примеров по принципу от частного к общему. Процесс классификации примеров осуществляется следующим образом:

1. Выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например, в соответствии с максимальным числом получаемых подмножеств примеров).
2. По значению выбранного признака множество примеров разбивается на подмножества.
3. Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу.
4. Если какое-то подмножество примеров принадлежит одному подклассу, то есть у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается (при этом остальные признаки классификации не рассматриваются).
5. Для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается, начиная с пункта 1 (каждое подмножество примеров становится классифицируемым множеством).

Нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Каждый процессор такой сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам.

В экспертных системах, ***основанных на прецедентах*** (аналогиях), база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты. Поиск решения проблемы в экспертных системах, основанных на прецедентах, сводится к поиску по аналогии (то есть абдуктивный вывод от частного к частному).

Адаптивная информационная система – это информационная система, которая изменяет свою структуру в соответствии с изменением модели проблемной области. При этом:

− адаптивная информационная система должна в каждый момент времени адекватно поддерживать организацию бизнес-процессов;

− адаптивная информационная система должна проводить адаптацию всякий раз, как возникает потребность в реорганизации бизнес-процессов;

− реконструкция информационной системы должна проводиться быстро и с минимальными затратами.

Ядром адаптивной информационной системы является постоянно развиваемая модель проблемной области (предприятия), поддерживаемая в специальной базе знаний – репозитории. На основе ядра осуществляется генерация или конфигурация программного обеспечения. Таким образом,

проектирование и адаптация ИС сводится, прежде всего, к построению модели проблемной области и ее своевременной корректировке.

5.3 Интеллектуальный анализ данных

Интеллектуальный анализ данных (ИАД), (Data Mining, DM) – это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей (шаблонов информации), то есть процесс извлечения информации, которая может быть охарактеризована как знания.

Интеллектуальный анализ данных является кратким обозначением довольно широкого спектра процедур автоматического анализа данных интеллектуальными информационными системами.

Основные задачи, решаемые средствами ИАД:

– выявление скрытых закономерностей и взаимосвязей в динамики состояния различных производственных и организационных бизнес-процессов;

– выявление наиболее значимых факторов влияния на качество функционирования бизнес-процессов;

– ранжирование важности измерений при классификации объектов для проведения анализа;

– оценка вероятности выхода показателей качества выпускаемой продукции и значений параметров состояния бизнес-процессов за допустимые пределы;

– прогнозирование изменения показателей качества и объемов выпуска товарной продукции в зависимости от выбора стратегии и режимов управления;

– прогнозирование спроса (уровня продаж, заказов) и потребительских характеристик товара;

– формирования оптимальных решений и вариантов управления бизнес-процессами;

– оценку влияния принимаемых решений на достижение успеха предприятия.

И целый ряд других задач.

Соответственно, назначение DM состоит в решении задач в интересах систем поддержки принятия решений на основе количественных и качественных исследований сверхбольших массивов разнородных ретроспективных данных. Перечень базовых задач, решаемых средствами DM, полностью совпадает со список задач, с которым сталкивается менеджер при управлении практически любым предприятием или бизнес-процессом, финансами, коммерческой деятельностью и т.п. А именно - поиск закономерностей, взаимосвязей, факторов влияния, угроз, прогнозирование и поиск возможных решений.

Технологии ИАД (Data Mining) позволяют наблюдать за текущей информацией с целью поиска в ней отклонений и тенденций без вникания в смысл самих данных. Такие технологии применяют, например, для оценки

поведения покупателей, чтобы внести изменения в рекламную политику фирмы, для корректировки выпуска продукции, изменения ценовой политики и т. д.

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.

Стадии интеллектуального анализа данных базируется на пяти основных этапах (рис.5.8):

Рис.5.8 – Этапы интеллектуального анализа данных.

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. 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.

На этапе очистки данных удаляется:

–*противоречивая информация* (например, разные данные об одном объекте за один и тот же период времени);

–*пропуски в данных* (например, отсутствие данных за какой-либо период);

–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;

–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.

–ошибки ввода данных. Количество ошибок разного типа достаточно 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 также относятся: заполнение пропусков, подавление аномальных значений, сглаживание, исключение дубликатов и др.

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) для решения задач ИАД. 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.

Полученные модели являются, по сути, формализованными знаниями эксперта, а, следовательно, их можно *тиражировать*. В этом заключается самое главное преимущество ИАД. ТО есть 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.

5.4 Экспертные системы

Экспертная система (ЭС) – это сложный программный комплекс, аккумулирующий и тиражирующий знания специалистов в конкретной предметной области и выполняющий функции эксперта при решении задач из этой области, консультируя менее квалифицированных пользователей.

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

Достоинство экспертных систем заключается в возможности принятия решений в уникальных ситуациях, для которых алгоритм заранее не известен и формируется по исходным данным в виде цепочки рассуждений (правил принятия решений) из базы знаний. Причем решение задач предполагается осуществлять в условиях неполноты, недостоверности, многозначности исходной информации и качественных оценок процессов.

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

–в экспертных системах в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма);

– экспертные системы применяются для решения только трудных практических задач, для решения которых нужны экспертные знания;

– экспертные системы 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);

Экспертная система является инструментом, усиливающим интеллектуальные способности эксперта, и может *выполнять следующие роли:*

–консультанта для неопытных или непрофессиональных пользователей;

–ассистента в связи с необходимостью анализа экспертом различных вариантов принятия решений;

–партнера эксперта по вопросам, относящимся к источникам знаний из смежных областей деятельности.

Экспертная система включает в себя два основных компонента: базу знаний (хранилище единиц знаний) и программный инструментарий доступа и

обработки знаний (рис.5.9). Программный инструментарий состоит из механизмов вывода заключений, приобретения знаний, объяснения получаемых результатов и интеллектуального интерфейса.

Центральным компонентом экспертной системы является база знаний, которая выступает по отношению к другим компонентам как содержательная подсистема, составляющая основную ценность.

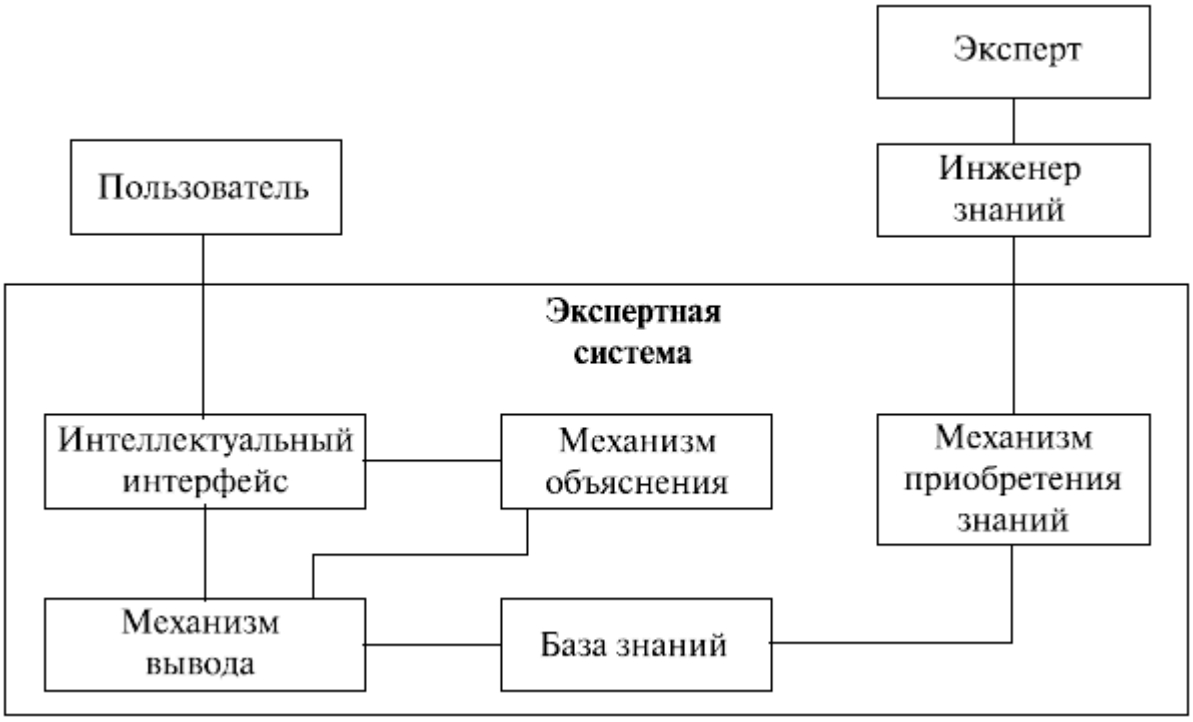


Рис.5.9 – Обобщенная структура экспертной системы.

База знаний - это совокупность единиц знаний, которые представляют собой формализованное с помощью некоторого метода представления знаний отражение объектов проблемной области и их взаимосвязей, действий над объектами и, возможно, неопределенностей, с которыми эти действия осуществляются.

В качестве *методов представления знаний* чаще всего используются либо правила, либо объекты (фреймы), либо их комбинация.

Рассмотрим классификацию методов представления знаний:

–логическая модель реализует и объекты, и правила с помощью предикатов первого порядка, является строго формализованной моделью с универсальным дедуктивным и монотонным методом логического вывода "от цели к данным";

–продукционная модель реализует эвристические методы вывода на правилах и может обрабатывать неопределенности в виде ус-ловных вероятностей или коэффициентов уверенности, а также выполнять монотонный или немонотонный вывод;

–семантическая сеть отображает разнообразные отношения объектов; фреймовая модель, как частный случай семантической сети, использует для реализации операционного знания присоединенные процедуры;

–объектно-ориентированная модель, как развитие фреймовой модели, реализуя обмен сообщениями между объектами, в большей степени ориентирована на динамические задачи и отражение поведенческой модели.

Интеллектуальный интерфейс. Обмен данными между конечным пользователем и ЭС выполняет программа интеллектуального интерфейса, которая воспринимает сообщения пользователя и преобразует их в форму представления базы знаний и, наоборот, переводит внутреннее представление результата обработки в формат пользователя и выдает сообщение на требуемый носитель. Важнейшим требованием к организации диалога пользователя с ЭС является естественность, которая не означает буквально формулирование потребностей пользователя предложениями естественного языка, хотя это и не исключается в ряде случаев. Важно, чтобы последовательность решения задачи была гибкой, соответствовала представлениям пользователя и велась в профессиональных терминах.

Механизм логического вывода (решатель). Этот программный инструментарий получает от интеллектуального интерфейса преобразованный во внутреннее представление запрос, формирует из базы знаний конкретный алгоритм решения задачи, выполняет алгоритм, а полученный результат предоставляется интеллектуальному интерфейсу для выдачи ответа на запрос пользователя. В основе применения любого механизма вывода лежит процесс нахождения в соответствии с поставленной целью и описанием конкретной ситуации (исходных данных), относящихся к решению единиц знаний (правил, объектов, прецедентов и т.д.) и связыванию их при необходимости в цепочку рассуждений, приводящую к определенному результату. Для представления знаний в форме правил это может быть прямая или обратная цепочка рассуждений. это программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в базе знаний.

Механизм объяснения. В процессе или по результатам решения задачи пользователь может запросить объяснение или обоснование хода решения. С этой целью ЭС должна предоставить соответствующий механизм объяснения.

Объяснительные способности ЭС определяются возможностью механизма вывода запоминать путь решения задачи. Тогда на вопросы пользователя "Как?" и "Почему?" получено решение или запрошены те или иные данные, и система всегда может выдать цепочку рассуждений до требуемой контрольной точки, сопровождая выдачу объяснения заранее подготовленными комментариями. В случае отсутствия решения задач объяснение должно выдаваться пользователю автоматически.

Полезно иметь возможность и гипотетического объяснения решения задачи, когда система отвечает на вопросы, что будет в том или ином случае. Однако не всегда пользователя интересует полный вывод решения,

содержащий множество ненужных деталей. В этом случае система должна уметь выбирать из цепочки только ключевые моменты с учетом их важности и уровня знаний пользователя. Для этого в базе знаний необходимо поддерживать модель знаний и намерений пользователя.

Если же пользователю все еще не понятен полученный ответ, то система должна быть способна в диалоге на основе поддерживаемой модели проблемных знаний обучать пользователя тем или иным фрагментам знаний, т.е. раскрывать более подробно отдельные понятия и зависимости, если даже эти детали непосредственно в выводе не использовались.

Механизм приобретения знаний. База знаний отражает знания экспертов (специалистов) в данной проблемной области о действиях в различных ситуациях или процессах решения характерных задач. Выявлением подобных знаний и последующим их представлением в базе знаний занимаются специалисты, называемые инженерами знаний. Для ввода знаний в базу и их последующего обновления ЭС должна обладать механизмом приобретения знаний. В простейшем случае используется интеллектуальный редактор, который позволяет вводить единицы знаний в базу и проводить их синтаксический и семантический контроль, например, на непротиворечивость. В более сложных случаях инженер знаний должен извлекать знания путем специальных сценариев интервьюирования экспертов, или из вводимых примеров реальных ситуаций, как в случае индуктивного вывода, или из текстов, или из опыта работы самой интеллектуальной системы.

Экспертная система работает в двух режимах:

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.

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 базы знаний 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.

Классификация и области применения экспертных систем.

Классификация по решаемой задаче.

Интерпретация данных – выбор решения из фиксированного множества альтернатив на базе введенной информации о текущей ситуации. Основное назначение - определение сущности рассматриваемой ситуации, выбор гипотез, исходя их фактов, определение смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных. Типичным примером является экспертная система анализа финансового состояния предприятия.

Диагностика – выявление причин, приведших к возникновению ситуации, обнаружение неисправности в некоторой системе. Требуется предварительная интерпретация ситуации с последующей проверкой дополнительных фактов, например, выявление факторов снижения эффективности производства. 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 диагностирующей системы

Мониторинг. Основная задача мониторинга - слежение за текущей ситуацией с возможной последующей коррекцией и непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Для этого выполняется диагностика, прогнозирование, а в случае необходимости - планирование и коррекция действий пользователей, например, мониторинг сбыта готовой продукции. Главные проблемы - "пропуск" тревожной ситуации и инверсная задача "ложного" срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимость учета временного контекста.

Проектировани. - определение конфигурации объектов с точки зрения достижения заданных критериев эффективности и ограничений, в подготовке спецификаций на создание "объектов" с заранее определенными свойствами, например, проектирование бюджета предприятия или портфеля инвестиций 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. – предсказание последствий развития текущих ситуаций на основе математического и эвристического моделирования,

например, прогнозирование сбыта товара. 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 – выбор последовательности действий пользователей, нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции по достижению поставленной цели, например, планирование процессов поставки продукции. В таких 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 компьютера 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.

Классификация 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.

Классификация 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.

В ОГЛАВЛЕНИЕ

Список использованных источников

1. Автоматизированные информационные технологии в экономике: учебник /под ред. Г. А. Титоренко. – М.: Юнити, 2005. –399 с.

2. Барановская Т.П., Лойко В.И. Информационные системы и технологии в экономике. – М.: Финансы и статистика, 2005. – 414 с.

3. Белов В.С. Информационно-аналитические системы. Основы проектирования и применения: учебное пособие, руководство, практикум – М.: Из-во Московский государственный университет экономики, статистики и информатики, 2004. – 116 с.

4. Гайдамакин Н.А, Автоматизированные информационные системы, базыи банки данных. Вводный курс: Учебное пособие. –М.: Гелиос АРВ, 2020. – 368 с.

5. Горбенко А.О. Информационные системы в экономике. – М.: Лаборатория знаний, 2020. – 295 с.

6. Елманова Н.З., Федоров А.И. Введение в OLAP-технологии Microsoft: учебное пособие. – Москва : Диалог-МИФИ, 2002. – 272 с.

7. Информационные системы: Учебник для вузов. /И.С. Телина, Ю.С. Избачков, В.Н. Петров [и др.]. – СПб.: Изд-во Питер, 2011. – 511 с.

8. Исаев Г. Н. Информационные системы в экономике: учебник для студентов вузов. – М.: Из-во «Омега-Л», 2013. – 462 с.

9. Карминский А.М., Черников Б.В. Информационные системы в экономике. Учебное пособие. Часть 1. – М.: Финансы и статистика, 2006. – 336 с.

10. Карминский А.М., Черников Б.В. Информационные системы в экономике. Учебное пособие. Часть 2. – М.: Финансы и статистика, 2006. – 240 с.

11. Кияев В.И., Граничин О.Н. Информационные технологии в управлении предприятием. [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/goods_store/ebooks/9724 (дата обращения: 12.06.2019).

12. Козлов, А.Н. Интеллектуальные информационные системы: учебник. – Пермь: Изд-во ФГБОУ ВПО Пермская ГСХА, 2013.– 278 с.

13. Кузнецов С.Д. Базы данных: языки и модели. – М.: Бином, 2008. – 720 с.

14. Макаренко С. И. Интеллектуальные информационные системы: учебное пособие. – Ставрополь: СФ МГГУ им. М. А. Шолохова, 2009. – 206 с.

15. Методы и модели анализа данных: OLAP и Data Mining: учебное пособие / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод. – СПб.: БХВ-Петербург, 2004. – 336 с.

16. Мишин И.Н. Информатика с основами баз данных. Учебное пособие для вузов /И.Н. Мишин. – Смоленск : ФГОУ ВО Смоленская ГСХА, 2016. – 175 с.

17. Остроух А.В. Интеллектуальные информационные системы и технологии: Монография / А.В. Остроух, Н.Е. Суркова. – Красноярск: Научно-инновационный центр, 2015. – 370 с.

18. Проектирование экономических информационных систем: учебник / Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов / под ред. Ю.Ф. Тельнова. - М.: Финансы и статистика, 2002. - 512 с.

19. Титоренко Г.А. Информационные системы в экономике. – М.: Юнити, 2008. – 463 с.

20. Федотова Е. Л. Информационные технологии и системы: учебное пособие.– М.: Изд-во ИНФРА-М, 2009. – 352 с.

21. Ясенев В.Н. Информационные системы в экономике: Учебное пособие. – М.: КНОРУС, 2021. 428 с.