Информационные технологии и базы данных в прикладных коммуникациях.

Классификация информационных технологий

Информационная технология представляет собой процесс, состоящий из четко регламентированных правил выполнения операций над информацией, и зависит от многих факторов. Для того чтобы правильно понять, оценить, грамотно разработать и затем выбрать для использования нужную информационную технологию, необходима их предварительная классификация. Выбор классификационных признаков зависит от сферы применения ИТ. Для сферы услуг возможно использование следующего набора критериев классификации, как это показано на рис. 1.2:

1) тип предметной области;
2) степень охвата задач управления;
3) класс реализуемых технологических операций;
4) тип пользовательского интерфейса;
5) способ построения сети.

1) Тип предметной области предполагает выделение функциональных классов задач соответствующих предприятий и организаций, которые решаются с использованием современной информационной технологии. К ним относятся задачи бухгалтерского учета и аудита, социально-культурной и туристической сферы, страховой и налоговой деятельности и т.д. Технология, реализуемая в конкретной предметной области, не зависит от используемых средств вычислительной техники и представляет собой последовательность технологических операций по преобразованию первичной информации в результатную.

2) По степени охвата задан управления в соответствии с характером обработки информации на различных уровнях управления экономической системой (оперативном, тактическом и стратегическом) выделяют технологии обработки данных, автоматизации функций управления, поддержки принятия решений. Они предусматривают применение экономико-математических методов, моделей и специализированных пакетов прикладных программ для аналитической работы и формирования прогнозов, составления бизнес- планов, обоснованных оценок и выводов по изучаемым процессам. К данной классификационной группе относятся также: технология электронного офиса, предназначенная для автоматизации и решения офисных задач, и технология экспертной поддержки, основанная на использовании экспертных систем и баз знаний конкретной предметной области.

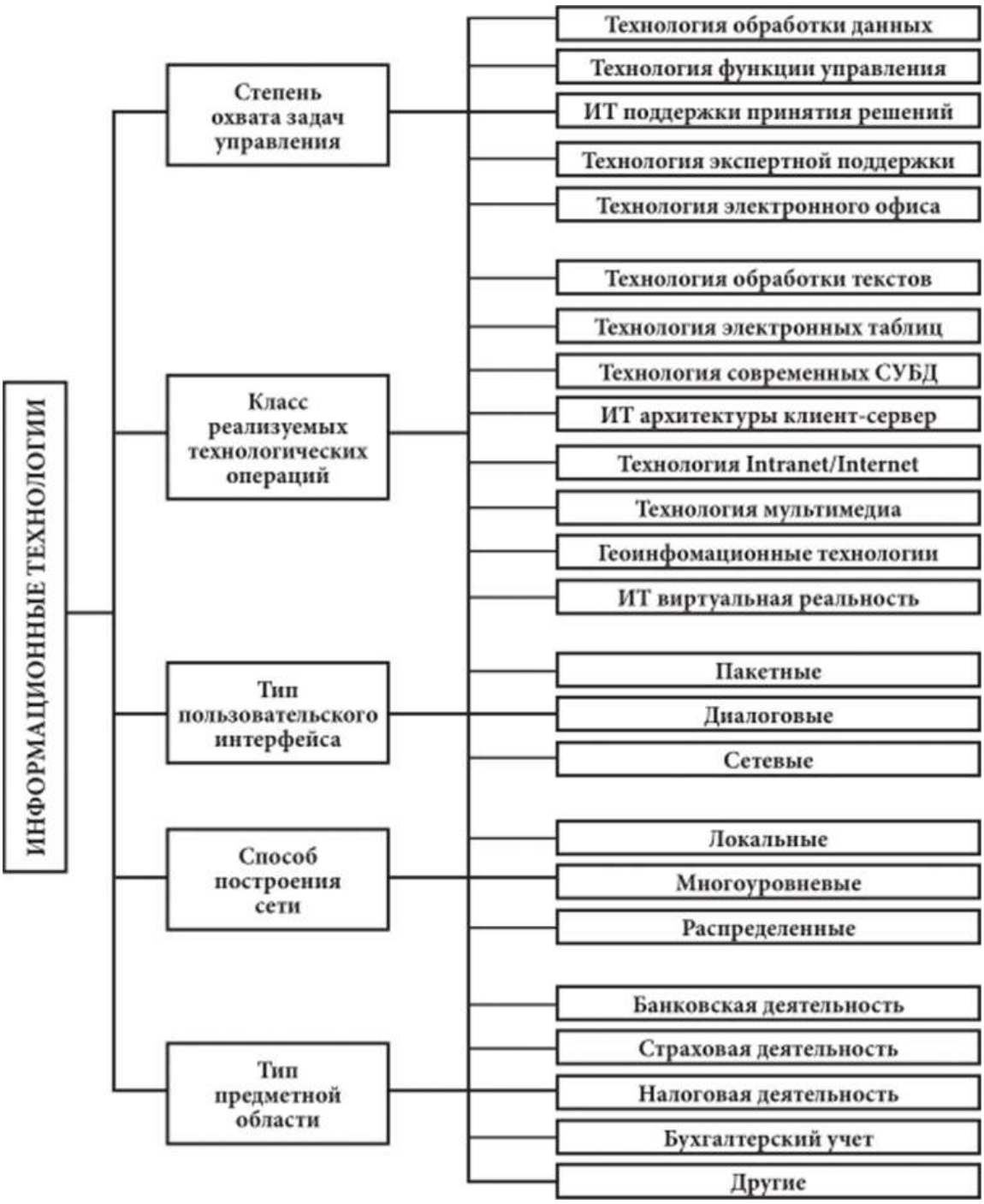


Рис. 1.2. Классификация информационных технологий

Информационная технология обработки данных предназначена для решения хорошо структурированных задач, по которым имеются необходимые входные данные и известны алгоритмы и другие стандартные процедуры их обработки. Эта технология применяется на уровне оперативной (исполнительской) деятельности персонала невысокой квалификации в целях автоматизации некоторых рутинных, постоянно повторяющихся операций управленческого труда: обработка данных о деятельности предприятия, создание отчетов периодических и по запросам, оформление различных документов (рис. 1.3).

Рис. 1.3. Основные компоненты информационной технологии обработки данных

Она предназначена для оперативного управления хозяйственными процессами с временным интервалом от одного до нескольких дней и реализует учет и оперативное регулирование хозяйственных операций, подготовку стандартных документов для внешней среды (счетов, накладных, платежных поручений), регистрацию и обработку событий, например оформление и мониторинг выполнения заявок, приход и расход материальных ценностей на складе, ведение табеля учета рабочего времени и т.д. Эти задачи имеют итеративный, регулярный характер, выполняются непосредственными исполнителями хозяйственных процессов (менеджерахми, кладовщикахми, адхмини- страторахми и т.д.) и связаны с оформлениехМ и пересылкой докухментов в соответствии с четко определенными алгоритмами. Результаты выполнения хозяйственных операций через экранные форхмы вводятся в базу данных.

Целью информационной технологии управления является удовлетворение информационных потребностей всех руководителей, имеющих дело с принятием решений. Она необходима на любом уровне управления. Эта технология предназначена для работы в среде информационной системы управления и используется при худшей структурированности решаемых задач, если их сравнивать с задачами, решаемыми с помощью информационной технологии обработки данных (рис. 1.4).

Рис. 1.4. Основные компоненты ИТ управления

ИТ управления поставляют информацию о прошлом, настоящем и вероятном будущем фирмы в виде регулярных или специальных управленческих отчетов, удобных для поддержки принятия решения. Содержимое базы данных должно состоять из данных, накапливаемых на основе оценки операций, проводимых фирмой, а также планов, стандартов, бюджетов и других нормативных документов, определяющих планируемое состояние объекта управления.

Эти технологии ориентированы 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 периодическая повторяемость формирования результатных документов и четко определенный алгоритм решения задач, например свод заявок для формирования тура и определение потребности в комплектующих услугах. Решение подобных задач предназначено для руководителей различных служб предприятий (отделов маркетинга и сбыта, питания, лечения и т.д.). Задачи решаются на основе накопленной базы оперативных данных.

Информационные технологии поддержки принятия решений используются в основном на верхнем уровне управления< (руководства фирм, предприятий, организаций) для формирования стратегических целей, планирования привлечения ресурсов, источников финансирования, выбора места размещения предприятий и т.д. Реже задачи этого класса решаются на тактическом уровне, например при выборе поставщиков или заключении контрактов с клиентами.

Для задач систем поддержки принятия решения свойственны недостаточность имеющейся информации, ее противоречивость и нечеткость, преобладание качественных оценок целей и ограничений, слабая формализованность 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 используются базы обобщенной информации, информационные хранилища, базы знаний о правилах и моделях принятия решений.

Главной особенностью информационной технологии поддержки принятия решений является качественно новый метод организации взаимодействия человека и компьютера. Выработка решения, что является основной целью этой технологии, происходит в результате итерационного процесса, в котором участвуют: система поддержки принятия решений в роли вычислительного звена и объекта управления; человек как управляющее звено, задающее входные данные и оценивающее полученный результат вычислений на компьютере.

Окончание итерационного процесса происходит по воле человека. В этом случае можно говорить о способности информационной системы совместно с пользователем создавать новую информацию для принятия решений.

Информационная технология электронного офиса ориентирована на автоматизацию и решение офисных задач, преобразуя офис в предприятие по переработке информации (рис. 1.5).

Электронный офис предусматривает организацию и поддержку коммуникационных процессов как внутри организации, так с внешней средой на базе вычислительных сетей и других современных средств передачи информации. Входящие в его состав интегрированные пакеты прикладных программ содержат специализированные программы и информационные технологии, которые обеспечивают комплексную автоматизацию задач предметной области.

Рис. 1.5. Основные компоненты ИТ электронного офиса

Современные компьютерные технологии в офисе обычно представляются двумя непересекающимися группами. Первая группа обеспечивает поддержку рутинных работ с документами, объединенных понятием делопроизводства: создание документов, рассылка документов, хранение документов, обеспечение санкционированного доступа к документам, поиск нужных документов и их фрагментов, то есть обычные для большинства офисов функции управления электронным документооборотом. Вторая группа обеспечивает средства поддержки принятия решений, которые специфичны для той отрасли или проблемы, которые решаются в данном офисе, в данной фирме, и включают методические и программные средства.

Основной целью ИТ электронного офиса является создание для специалистов и руководителей благоприятных условий выполнения профессиональных функций, качественного и своевременного информационного обслуживания за счет полного автоматизированного набора управленческих процедур, реализуемых в условиях конкретного рабочего места и офиса в целом.

Эта технология поддерживает выполнение следующих функций:

- - обработка текстов на компьютерах с помощью различных текстовых процессоров;
- - производство высококачественной печатной продукции;
- - архивация документов;
- - электронные календари и записные книжки для ведения деловой информации;
- - электронная почта;
- - видео- и телеконференции.

В последнее время начинает использоваться технология виртуального офиса, которая основывается на работе локальной сети, соединенной с территориальной или глобальной сетью, а также на основе технологии беспроводной связи WiFi. Благодаря этому абонентские системы сотрудников учреждения независимо от того, где они находятся, оказываются включенными в общую для них сеть и имеют возможность для работы с документами, материалами, базами данных конкретной организации или учреждения в домашних условиях, в гостинице, транспортных средствах.

Информационная технология экспертных систем (или экспертная поддержка) реализована на использовании экспертных систем и баз знаний конкретной предметной области и предназначена для автоматизации труда специалистов-аналитиков. Они дают возможность специалисту получать консультации экспертов по любым проблемам, о которых этими системами накоплены знания. Исследователи находят, что использование этой массы знаний более эффективно, чем использование специальных решающих процедур. Экспертные системы являются консультантами в принятии решений, так как содержат факты, знания и правила, которые взаимодействуют в проблемной области [Гольдштейн Г.Я.]. Пример общей структуры экспертной системы классификации на основе правил приведен на рис. 1.6.

Главная идея технологии экспертных систе^м заключается в том, чтобы получить от эксперта его знания и, загрузив их в память компьютера, использовать всякий раз, когда в этом возникнет необходимость. Являясь одним из основных приложений искусственного интеллекта, экспертные системы представляют собой компьютерные программы, трансформирующие опыт экспертов в какой-либо области знаний в форму эвристических правил (эвристик). Эвристики не гарантируют получения оптимального результата с такой же уверенностью, как обычные алгоритмы, используемые для решения задач в рамках технологии поддержки принятия решений. Однако они могут выдать в достаточной степени приемлемые решения для их практического применения. Все это делает возхможным использовать технологию экспертных систем в качестве советующих систем.

Рис. 1.6. Структура информации в экспертной системе классификации

3) По классам реализуемых технологических операций. В этом случае ИТ используется в соответствии с решением задач прикладного характера и имеющимся прикладным программным обеспечением, таким как обработка текстов, электронные таблицы, современные системы управления базами данных (СУБД), технологии мультимедиа и виртуальная реальность, геоинформационные технологии и другие.

Технология современных СУБД реализует три основные функции: ввод и редактирование данных, обработка информации базы данных и вывод информации из БД. Каждая конкретная СУБД (Access, Oracle, SQL Server, Interbase и т.д.) имеет свои особенности, которые необходимо учитывать. Однако обобщенную технологию работы пользователя в СУБД, реализующей реляционную модель данных, можно представить следующим образом (рис. 1.7).

Рис. 1.7. Технология работы пользователя в СУБД

Технология мультимедиа (multimedia - многокомпонентная среда) позволяет использовать текст, графику, аудио- и видеоинформацию, мультипликацию в интерактивном режиме и тем самым расширяет рамки применения компьютера в управлении.

Технологии виртуальной реальности - это новые технологии неконтактного информационного взаимодействия, создающие мультимедийными средствами иллюзию присутствия в реальном времени в стереоскопически представленном «экранном мире». В таких системах непрерывно поддерживается иллюзия места нахождения пользователя среди объектов виртуального мира. Вместо обычного дисплея используются очки-телемониторы, в которых воспроизводятся непрерывно изменяющиеся события виртуального мира. Управление осуществляется с помощью реализованного в виде «информационной перчатки» специального устройства, определяющего направление перемещения пользователя относительно объектов виртуального мира. Кроме этого, в распоряжении пользователя есть устройство создания и передачи звуковых сигналов.

Географические информационные (геоинформационные) технологии обеспечивают обработку пространственно-временных данных, основой интеграции которых служит географическая информация. Геоинформационные

системы (ГИС), построенные на базе этой технологии, обеспечивают ввод, хранение, обновление, обработку, анализ и визуализацию всех видов географически ориентированной информации [В.Н. Бочарников]. ГИС представляют собой электронную карту с размещенными на ней различного рода объектами: зоны отдыха, маршруты, экскурсионные объекты, остановки транспорта и т.п.

4) *По типу пользовательского интерфейса.* ИТ, в зависимости от возможностей доступа пользователя к информационным, вычислительным и программным ресурсам, подразделяются на пакетную, диалоговую и сетевую.

Технология обработки информации на компьютере может заключаться в выполнении заранее определенной последовательности операций и не требовать вмешательства пользователя в процесс обработки. В данном случае отсутствует диалог пользователя с системой, информация будет обрабатываться *в пакетном режиме,* в котором данные вначале накапливаются, формируется пакет данных, а затем пакет последовательно обрабатывается рядом программ. Недостатком этого режима является низкая оперативность принятия решений и обособленность пользователя от системы, поскольку пакетная ИТ не предоставляет возможности пользователю влиять на обработку данных, в то время как *диалоговая технология* позволяет ему взаимодействовать с вычислительными средствами в интерактивном режиме, оперативно получая информацию для принятия управленческих решений.

Эта технология особенно удобна, когда пользователь может выбирать перспективный вариант из числа предлагаемых системой. Таким образом, диалоговый режим (интерактивный) является развитием пакетного режима. Если применение пакетного режима позволяет уменьшать вмешательство пользователя в процесс задачи, то диалоговый режим предполагает отсутствие жестко закрепленной последовательности операций обработки данных.

Интерфейс сетевой предоставляет пользователю телекоммуникационные средства доступа к территориально удаленным информационным и вычислительным ресурсам, обеспечивая при этом взаимодействие многих пользователей.

Кроме того, при классификации информационных технологий по типу пользовательского интерфейса говорят о системном и прикладном интерфейсе.

Системный интерфейс - это набор приемов взаимодействия с компьютером, который реализуется операционной системой или его надстройкой.

Прикладной интерфейс связан с реализацией некоторых функциональных информационных технологий и является основным при работе пользователя с ИС.

5) По способу построения сети. В настоящее время наблюдается тенденция к объединению различных типов информационных технологий в единый компьютерно-технологический комплекс, который носит название интегрированного. Особое место в нем принадлежит средствам коммуникации, обеспечивающим не только чрезвычайно широкие технологические возможности автоматизации управленческой деятельности, но и являющимся

основой создания самых разнообразных сетевых вариантов ИТ: локальных, многоуровневых, распределенных, глобальных вычислительных сетей, архитектуры «клиент-сервер».

Все сетевые варианты ориентированы на технологическое взаимодействие, которое организуется за счет средств передачи, обработки, накопления, хранения и защиты информации. Способ построения сети зависит от требований управленческого аппарата к оперативности информационного обмена и управления всеми структурными подразделениями фирмы.

Локальные вычислительные сети (ЛВС) охватывают ограниченную территорию в пределах удаленности станций не более десятков или сотен метров друг от друга и представляют собой самую распространенную и элементарную форму сетей. Они применяются в учебных заведениях, в конторах, в офисах мелких и крупных компаний, отличаются простотой создания и администрирования.

ЛВС соединяют вместе группу персональных компьютеров или связывают их с более мощным компьютером, который выполняет функции сетевого сервера. Все компьютеры в локальной сети могут использовать специализированные приложения, хранящиеся на сетевом сервере. Каждый персональный компьютер (ПК) в локальной сети называется рабочей станцией или сетевым узлом. Локальные сети позволяют отдельным пользователям легко и быстро взаимодействовать друг с другом, получить доступ к информационным и программным ресурсам на сервере и обеспечить совместное использование в организациях дорогостоящих ресурсов, таких как принтеры, накопители CD-ROM, жесткие диски и приложения (например, текстовые процессоры или программное обеспечение баз данных).

Существует также одна небольшая подгруппа ЛВС - персональная сеть пользователя, которая объединяет все его персональные цифровые устройства: телефоны, смартфоны, принтеры, фотоаппараты, ПК, КПК, ноутбуки и т.п. Наиболее известным стандартом для создания персональных сетей в настоящее время является Bluetooth.

Вычислительные сети подразделяются на одноранговые (одноуровневые), иерархические (многоуровневые), клиент-серверные, распределенные и глобальные сети.

Одноранговая сеть представляет собой сеть равноправных компьютеров, каждый из которых может выполнять функции и клиента, и сервера. В общем случае под клиентом понимается объект (устройство или программа), запрашивающий некоторые услуги, а под сервером - объект, предоставляющий эти услуги. Сервер обычно представляет собой высокопроизводительный компьютер, возможно, с несколькими параллельно работающими процессорами, с винчестерами большой емкости, с высокоскоростной сетевой картой. Он оснащается соответствующим программным обеспечением, централизованно управляет работой сети и/или предоставляет другим компьютерам сети свои ресурсы (данные, прикладные программы, накопители, принтеры и т.д.). Компьютеры пользователей, с которых осуществляется доступ к информации на сервере, называются рабочими станциями или

клиентами. Например, если на вашем компьютере установлен принтер, то с его помощью смогут распечатывать свои документы все остальные пользователи сети, а вы, в свою очередь, сможете работать с Интернетом, подключение к которому осуществляется через соседний компьютер.

В иерархических локальных сетях имеется один или несколько серверов, на которых хранится информация, совместно используемая различными пользователями. Сервер в иерархических сетях является постоянным хранилищем разделяемых ресурсов. Сам сервер может быть клиентом только сервера более высокого уровня иерархии.

В сети типа *«клиент-сервер»* существует два типа машин: клиенты и сервер. Основная задача сервера - предоставлять свои ресурсы клиентам, то есть всем другим компьютерам сети. Ресурсы, предоставляемые серверами, определяются их типами: серверы баз данных поставляют данные, серверы приложений - прикладные программы, почтовые сервера - почтовые сервисы и т.п. Чаще используется сервер баз данных, который по запросам прикладных программ, установленных на рабочей станции, поставляет данные из базы.

При наличии в сети нескольких специализаций серверов получают сеть распределенных вычислений. В *распределенной сети* управление распределено между всеми серверами, которые могут быть различных типов и географически удалены друг от друга на большие расстояния. Пользователи не закреплены за отдельными серверами и имеют доступ ко всем ресурсам сети: аппаратным, программным и информационным. Распределенные сети очень большого масштаба (например, Internet) часто называют глобальными.

Глобальная сеть Internet является единственной в своем роде с уникальной технологией, специфика которой заключается в том, что она предоставляет пользователям громадные возможности выбора источников информации: базовая информация на серверах сети; оперативная информация, пересылаемая по электронной почте; разнообразные базы данных ведущих библиотек, научных и учебных центров, музеев; информация о компакт-дисках, видео- и аудиокассетах, книгах и журналах, распространяемых через интернет-магазины, и др.

Интересным примером связи локальных и глобальных сетей является *виртуальная частная сеть,* получающаяся в результате объединения двух или нескольких территориально разделенных ЛВС с помощью общедоступных каналов глобальных сетей, например через Интернет.

Технология Internet в настоящее время является самым большим и популярным межсетевым объединением в мире, ее можно определить как сеть сетей, или как глобальную информационную систему, которая объединяет десятки тысяч компьютерных сетей и миллионы пользователей во всем мире. Технология, разработанная для Internet и Всемирной паутины (WorldWideWeb, WWW), воплощает идею глобальной информационной базы данных, реализованную в пределах современных возможностей.

В Internet существует понятие *интрасетей (Intranet)* - корпоративных сетей в рамках Internet. Технологию Intranet также можно назвать web-системой, локализованной в пределах одной организации и представляющей

собой технологию управления корпоративными коммуникациями. В этом ее отличие от Интернета, который является технологией глобальных коммуникаций. Обычно Интранет определяют как частную компьютерную сеть, в которой используются стандарты и протоколы Интернета. При этом достигается независимость этих сетей от используемых программно-аппаратных средств и возможность их развития.

Для преобразования локальной или региональной компьютерной сети в Интранет не потребуется распродавать старое оборудование, можно обойтись уже существующими ресурсами. Обычная компьютерная сеть имеет те же самые функции, однако они реализуются через различные пользовательские интерфейсы, которые характерны для каждой из компьютерных платформ. В Интранет все эти функции формируются на основе единых стандартов технологий Интернета.

Достоинства интранет-технологии следуют из особенностей, заложенных при создании WWW-технологий:

• использование гипертекста обеспечивает связность разнородной информации;

• броузер (клиентская программа WWW-сервера) предоставляет единый, более простой интерфейс пользователя.

Интранет имеет пять основных функций - это электронная почта, совместное использование файлов, каталогизация, поиск, управление сетью. Эти функции позволяют организации публиковать, хранить, извлекать и управлять информацией. При этом формируется единое информационное пространство для всех сотрудников, которые могут находиться на различных этажах здания центрального офиса компании, в различных регионах и даже в разных странах.

Революционные изменения в автоматизацию сферы сервиса и туризма вносит новая тенденция использования телекоммуникаций сети Интернет, которая предоставляет возможность предприятиям не приобретать ИТ-ресурсы в собственность, а арендовать их у компании сервис-провайдера - *Application Service Provider (ASP).* Сервис- провайдер становится новым звеном в производственной цепочке «производители инструментальных средств разработки и СУБД - независимые разработчики, создающие системы на их базе - сервис-провайдеры (ASP) - конечный пользователь». Обязанность сервис-провайдера заключается в приспособлении IT-ресурсов к нуждам пользователей-арендаторов, которые получают к ним доступ через Интернет.

Технология ASP исключает затраты на приобретение лицензий на программное обеспечение, аппаратное и телекоммуникационное оборудование, содержание обслуживающего персонала, выделенную линию и обновление системы. Клиент только арендует ИТ-ресурсы, его деньги работают на него, они не связаны собственностью и не уменьшаются вместе с амортизационными списаниями и старением оборудования.

Сервис-провайдер организует технологический узел и размещает там ИТ-ресурсы, на основе которых предлагает сервисы: Web- хостинг, выделенные серверы, аренда программных приложений (рис. 1.8).

Рис. 1.8. ASP-модель использования IT-инфраструктуры (по А. Печникову)

Web-хостинг является простейшей формой аренды ИТ-ресурсов, при которой поставщик услуг хостинга, устанавливая один мощный сервер, предоставляет услугу аренды виртуального web-сервера одновременно сотням клиентов, не снижая при этом качество сервиса для каждого арендатора в отдельности. Арендатор при этом получает возможность удаленного управления Web-сервером, которая может осуществляться с использованием любого коммутируемого канала связи, что гораздо практичнее, чем поддерживать и обслуживать сам сервер и выделенный канал в сети.

Выделенные серверы используются для проектов, требующих существенных вычислительных ресурсов. В этих случаях ASP предлагают в аренду специально выделенный для клиента сервер, подключенный к Интернету по высокоскоростным каналам. За арендную плату клиент получает полностью готовую ИТ-инфраструктуру компании, обладая полным контролем над ресурсами сервера, имеет возможность конфигурировать систему в соответствии со своими нуждами, устанавливать любое ПО и т. д. При этом нет больших стартовых затрат, являющихся для многих пороговым ограничением при начале проектов.

Аренда программных приложений является основной и самой распространенной услугой ASP, которая обеспечивает аренду сервисов, построенных на основе программных приложений. С помощью Интернета можно арендовать только те приложения, которые реализованы по архитектуре «клиент-сервер». Все арендуемые приложения можно разделить на несколько групп:

• приложения для персонального использования: игры, редакторы, образовательные и антивирусные программы;
• офисные приложения для совместной деятельности: электронная почта, системы для групповой работы, телеконференции и т. д.;
• приложения для электронной коммерции и бизнеса: интернет-магазины, интернет-витрины, торговые порталы, B2B-системы, расчетные (биллинговые) системы.

В настоящее время у потребителей услуги аренды приложений наибольшим спросом пользуются следующие программные приложения:

• управления взаимоотношениями с клиентами (CRM);
• корпоративного планирования ресурсов (ERP);
• планирования работы с поставщиками (SCM);
• управления транспортными потоками (TSM);
• управления хранилищами данных (WMS);
• сервисы, которые нужны не регулярно, а периодически: система бронирования билетов для организации личного отдыха, заполнение налоговой декларации, расчет строительной сметы, юридическая справочная система и многое другое.

Практически все серьезные разработчики ИС имеют версию своих информационных систем, которые разработаны с применением технологий ASP и «клиент-сервер». В России эти продукты представлены системами "Kei Hotel", "Nimeta", "Libra OnDemand CRM" и другими. Этот вид систем называют также системами по подписке.

Специалисты сферы ИТ выделяют ряд современных взаимосвязанных и усиливающих друг друга доминирующих *тенденций развития* информационных технологий.

Первая тенденция связана с постоянным усложнением информационных продуктов или услуг и возрастанием их роли. Отмечается способность к *параллельному взаимодействию* всех типов информации (текста, образов, цифр, звуков) с ориентацией на одновременное восприятие человеком посредством органов чувств.

Вторая тенденция заключается в максимальном *уменьшении всех промежуточных звеньев* на пути от источника информации к ее потребителю, например, становится возможным непосредственное общение автора и читателя, продавца и покупателя, певца и слушателя, преподавателя и обучающегося, специалистов на предприятии через систему видеоконференций, электронный киоск, электронную почту.

Третьей тенденцией, которую многие рассматривают в качестве ведущей, является тенденция к глобализации информационных технологий в результате использования спутниковой связи и всемирной сети INTERNET, благодаря чему люди смогут общаться между собой и с общей базой данных, находясь в любой точке планеты .

Текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов.

Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Первая версия была написана Ричардом Броди (Richard Brodie) для IBM PC, использующих DOS, в 1983 году. Позднее выпускались версии для Apple Macintosh (1984), SCO UNIX и Microsoft Windows (1989). Текущей версией является Microsoft Office Word 2019 для Windows и macOS, а также веб-версия Word Online (Office Online), не требующая установки программы на компьютер.

Текстовый процессор Word

Введение в Microsoft Word

Текстовый редактор Microsoft Word — это мощное средство для создания текстовых документов профессиональной направленности или личного характера.

С помощью MS Word, можно:

• использовать и создавать шаблоны профессионально оформленных отчетов, факсов и других документов;

• формировать, редактировать и форматировать документ на экране;

• комбинировать части существующих документов для создания новых документов;

• оформлять документы с помощью таблиц, списков и графики;

• сохранять документы на диске для их дальнейшего использования.

Интерфейс программы MS Word

В графическом интерфейсе MS Word 2010 используются различные элементы управления, сгруппированные с помощью объекта, называемого «лентой».

Она предназначена для доступа к командам и состоит из вкладок, связанных с определенными целями или объектами. Каждая вкладка, в свою очередь, состоит из нескольких групп взаимосвязанных элементов управления. По сравнению с меню и панелями инструментов, используемых в предыдущих версиях MS Word, «лента» вмещает значительно больше содержимого — кнопок, коллекций, элементов диалоговых окон и т. д.(Рис.1)

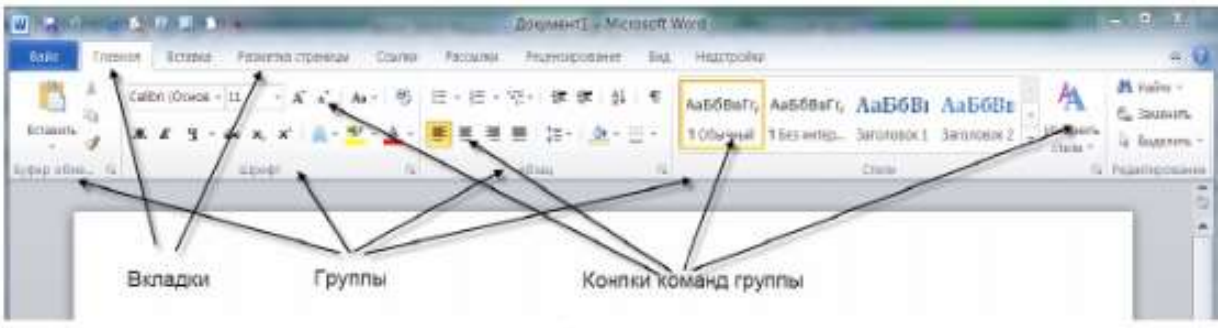


Рисунок 1. Структура ленты

Другие элементы интерфейса

Строка заголовка (Title bar) содержит информацию об имени программы и имени загруженного (активного) документа. При открытии нового документа ему присваивается временное имя ДокументN (DocumentN), где N — это число.

Масштабная линейка (Ruler) используется для установки отступов в абзацах и задания смещения текста внутри строки.

Полосы прокрутки (Scroll bars) (вертикальная и горизонтальная) предназначены для просмотра части документа, не помещающегося целиком в окне.

Строка состояния (Status bar) представляет собой горизонтальную полосу в нижней части окна документа. В строке состояния отображаются данные о текущем состоянии содержимого окна и другие сведения, зависящие от контекста.

Курсор ввода перемещается по мере ввода текста и указывает место в редактируемом документе, куда будет произведена вставка текста.

Правила ввода текста

Текст в MS Word состоит из символов. Символы группируются в слова, слова объединяются в абзацы. Для разделения слов используется клавиша пробел, для разделения абзацев используется клавиша ENTER.

Одно из преимуществ работы с MS Word — свойство WYSIWYG (What You See Is What You Get) — то, что вы видите на экране, то вы и получите на печатном листе.

От корректности ввода информации зависит процесс оформления готового документа, поэтому необходимо использовать следующие правила ввода:

1. Текст ВСЕГДА следует вводить от левого поля страницы.

2. По достижении правого поля страницы происходит автоматический перевод курсора на новую строку, поэтому не следует нажимать клавишу ENTER в конце каждой строки.

3.Клавиша ENTER нажимается только в конце абзаца, в том случае, если вы хотите создать новый абзац.

После нажатия клавиши ENTER MS Word рассматривает ранее напечатанный текст в качестве абзаца. Фактически, MS Word создает абзац всякий раз, когда вы нажимаете клавишу ENTER, поэтому пустые строки тоже считаются абзацами.

Вы можете инициировать перевод строки, не создавая жесткого прерывания, используя клавиши SHIFT+ENTER. Это позволяет создать список, который MS Word будет рассматривать как один абзац.

Любое форматирование абзаца, относящееся к одной строке, будет также относиться и ко всем строкам в списке.

4. Не вставляйте пустые строки при помощи клавиши ENTER. Требуемое расстояние между строками и абзацами достигается за счет операций форматирования.

5. Для смещения текста по строке используется клавиша табуляции — TAB.

7. Клавиша пробела используется ТОЛЬКО для разделения слов в предложении.

8 После символа пунктуации обязательно ставится пробел.

Перед символом пунктуации пробел не ставится. Исключение составляют кавычки и скобки: пробел ставится перед открывающей и после закрывающей скобки или кавычки:

Перед запятой пробела – нет, после – есть

Скобки пишутся без пробела к тексту

9 Для отмены последнего произведенного действия или для возврата назад на определенное количество операций нажмите кнопку Отменить (Undo) (↶) на панели быстрого доступа столько раз, на сколько операций нужно вернуться. Либо нажмите на клавиатуре комбинацию клавиш CTRL+Z.

10 Для ввода заглавного символа используется комбинация клавиш SHIFT+СИМВОЛ, а для ввода целой строки заглавных символов используется клавиша CAPSLOCK.

Перемещение по документу

Для редактирования текста документа необходимо научиться перемещаться в нужную позицию документа.

Для перемещения по документу, можно использовать:

• мышь;

• клавиатуру.

Используйте вертикальную полосу прокрутки для поиска нужного фрагмента документа.

Используйте клавиши для поиска нужного фрагмента документа.

При этом курсор автоматически перемещается на новую позицию.

Описание клавиш для перемещения по документу приведено в Табл. 1.

Табл. 1. Клавиши для перемещения по документу

Клавиша(и)	**Действие**
↑/↓	Перемещает курсор на строку вверх/вниз
←/→	Перемещает курсор на один символ влево/вправо
HOME	Перемещает курсор в начало строки
END	Перемещает курсор в конец строки
CTRL + ←	Перемещает курсор на одно слово влево
CTRL + →	Перемещает курсор на одно слово вправо
CTRL + ↑	Перемещает курсор вверх на один абзац
CTRL +↓	Перемещает курсор вниз на один абзац
PAGE UP	Перемещает курсор на одну экранную страницу вверх
PAGE DOWN	Перемещает курсор на одну экранную страницу вниз
CTRL + PAGE UP	Перемещает курсор в начало предыдущей физической страницы
CTRL + PAGE DOWN	Перемещает курсор в начало следующей физической страницы
CTRL + HOME	Перемещает курсор в начало документа
CTRL + END	Перемещает курсор в конец документа

Выделение текста

Для того чтобы произвести операции копирования, перемещения или удаления фрагмента текста, его необходимо выделить. Выделенный текст подсвечивается черным цветом.

Если вы выделили текст, то любой вводимый с клавиатуры текст заменит выделенный. Например, если выделить пять страниц текста и нажмете пробел, то все пять страниц будут заменены одним пробелом.

Существуют несколько способов выделения текста:

• с помощью мыши;

• с помощью клавиатуры;

• при помощи области выделения.

Выделение текста с помощью клавиатуры

1 Переместить указатель мыши в позицию, с которой надо начать выделение.

2 Удерживая нажатой клавишу SHIFT, воспользоваться клавишами для перемещения по тексту (см. Табл. 1)

Для выделения всего документа целиком использовать комбинацию клавиш CTRL + A, либо CTRL + 5 (на дополнительной клавиатуре).

Выделение текста с использованием области выделения

Область выделения — это пустая область слева от основного текста (левое поле страницы документа). Попадая в область выделения, курсор принимает вид стрелки, направленной вправо вверх ⇗

.

1 Переместить указатель мыши (УМ) в крайнее левое положение относительно строки, которую необходимо выделить.

Выделение строки текста. Переместить УМ в область выделения, подвести к строке, которую необходимо выделить, и щелкнуть ЛКМ.

Чтобы выделить несколько строк, нажать ЛКМ и, удерживая нажатой, выделить требуемое количество строк.

Выделение абзаца. Переместить УМ в область выделения, подвести к абзацу и дважды щелкнуть ЛКМ. Чтобы выделить несколько абзацев, нажмите ЛКМ и, удерживая нажатой, выделите требуемое количество абзацев

Выделение документа. ПереместитьУМ в область выделения, сделать тройной щелчок ЛКМ, или, удерживая нажатой клавишу CTRL, один раз щелкнуть ЛКМ.

Управление документами

При запуске программы MS Word автоматически создается пустой документ, с которым можно работать. При создании или открытии документа в MS Word, документ открывается, по умолчанию, в отдельном окне.

Создание нового документа

Все документы в программе MS Word создаются на основе специальной папке Шаблоны (Templates).

По умолчанию, для создания нового документа используется шаблон Normal.dotx. Этот шаблон позволяет создавать простые чистые документы на листах бумаги формата А4, со стандартными размерами полей страницы.

В случае необходимости можно воспользоваться одним из готовых шаблонов (к примеру, шаблоном резюме или факса) или создать собственные шаблоны документов (регистрационные бланки, поздравительные открытки и т.д.).

Сохранение документа MS Word

1. Для сохранения документа выполнить одно из следующего:

Вкладка Файл➔ Сохранить.

Либо нажать кнопку Сохранить.

Откроется диалоговое окно Сохранение документа .

2. В окне списка файлов выбрать нужную папку и открыть ее двойным щелчком мыши.

3. В поле Имя файла ввести название документа.

Допускается использование длинных описательных имен файлов.

В имени файла нельзя использовать следующие символы: косая черта (/), обратная косая черта (\), знак больше (>), знак меньше (<), звездочка (*), знак вопроса (?), двойные кавычки (“), вертикальная черта (|), двоеточие (:) и точка с запятой (;).

4. Нажать кнопку Сохранить.

Открытие существующего документа

1. Для открытия ранее созданного документа выполнить одно из следующего:

Вкладка Файл ➔Открыть.

Откроется диалоговое окно Открытие документа (Open)

2. Выбрать нужную папку и открыть ее двойным щелчком мыши.

Открывать папки до тех пор, пока не откроется папка, содержащая нужный документ.

3 Из списка файлов выбать нужный документ.

4 Нажатье кнопку Открыть.

Режимы просмотра документа

Для удобства работы MS Word предоставляет различные режимы редактирования документов:

- Черновик.
- Веб-документ.
- Разметка страницы.
- Структура.
- Режим чтения.

Режим Черновик (в предыдущих версиях «Обычный») предназначен для ввода, редактирования и форматирования текста.

В этом режиме форматирование текста отображается полностью, а разметка страницы — в упрощенном виде (на горизонтальной линейке отображаются границы левого и правого поля). Границы страниц, колонтитулы, фон, графические объекты и рисунки, для которых не определен стиль обтекания В тексте, не отображаются. Это представление является наиболее эффективным для концентрации внимания на тексте.

Режим веб-документа (Web Layout View) наиболее удобен для создания веб-страниц и документов, предназначенных для просмотра с помощью веб-обозревателя по локальной сети или через Интернет.

В этом режиме отображается фон, текст переносится по границе окна, а рисунки занимают те же позиции, что и в окне веб-обозревателя.

В режиме разметки страницы (Print Layout View) отображается действительное положение текста, рисунков и других элементов на печатной странице.

Этот режим удобно использовать для изменения колонтитулов и полей, а также работы с колонками и графическими объектами.

Режим структуры позволяет видеть структуру документа (иерархию заголовков), а также перемещать и копировать большие фрагменты текста посредством их заголовков.

В режиме структуры удобно работать с большими и можно работать с главными документами. Использование главных документов упрощает создание и обновление больших документов, например отчетов, включающих несколько частей, или книг, состоящих из нескольких глав.

Режим чтения позволяет работать в приложении без ленты, полос прокрутки —виден только лист документа. Рекомендуется выбирать режим чтения, если длинный документ, и нужно максимально использовать пространство экрана. При этом можно использовать команды контекстного меню и "горячие" клавиши.

Для переключения в нужный режим использовать один из вариантов:

1. Вкладка Вид (View)➔ Режимы просмотра документа

2. Щелкнуть кнопку Режим *** (вторая слева кнопка) на горизонтальной полосе прокрутки (внизу).

Чтобы вернуться к нормальному виду представления документа, использовать кнопку Esc.

Форматирование символов и абзацев

Для привлечения внимания к фрагменту текста, слову, символу, его можно красиво оформить, выделить, т.е. отформатировать.

Минимальным форматируемым в MS Word фрагментом текста является символ. Символ — это отдельная буква, цифра, знак пунктуации или специальный знак. Символы объединяются в слова – группы символов, разделенные пробелами. Из слов можно составлять предложения.

Абзац — это последовательность предложений, объединенных вместе для выражения отдельной мысли, идеи или образа.

При работе с MS Word абзацем могут быть: заголовок статьи, элемент списка, пустая строка между двумя абзацами и т.п.

Форматирование символов

Форматирование символов можно осуществить с помощью:

• кнопок на вкладке Главная в группе Шрифт,

• окна диалога, открывающегося путем нажатия кнопки справа от названия группы Шрифт.

Можно определить различные способы форматирования для вновь вводимого символа или уже написанного.

Форматирование символов с помощью вкладки Главная

Форматирование символов с помощью окна диалога Шрифт

Форматирование символов с помощью мини-панели

Форматирование абзаца

Абзац MS Word — это любой фрагмент текста, который заканчивается маркером абзаца (¶).

При нажатии на клавишу ENTER, образуется новый абзац.

Маркер абзаца виден при включенном режиме непечатных символов: кнопка Непечатаемые знаки (Shoe/Hide ¶) на вкладке Главная группа Абзац должна быть нажата.

Команды форматирования абзаца применяются только к выделенным абзацам или к абзацу, в котором находился курсор. Другие абзацы в документе остаются неизменными.

При нажатии клавиши ENTER, создается новый абзац, с теми же параметрами форматирования, что и у предыдущего, если специально не указано другое.

Параметры абзаца:

• выравнивание текста (по правому краю, по левому краю, по центру, по ширине);

• отступы (справа, слева, первой строки, выступ);

• межстрочный интервал, интервалы до и после абзаца в пт.

Дополнительные возможности форматирования

Копирование формата

Для того чтобы перенести оформление (форматирование) с одного фрагмента текста на другой, необходимо выполнить копирования формата по образцу.

Копирование формата с помощью вкладки Главная

1. Выделить фрагмент текста, формат которого надо копировать - оригинал.

2. Нажать кнопку **Формат** по образцу (Format Painter) на вкладке **Главная**. При этом в области текста указатель мыши превращается в кисточку  ;

3. Подведите видоизмененный к началу фрагмента текста, на который будет копироваться формат, и выделите его, удерживая левую кнопку мыши нажатой.

4. После копирования кнопка **Формат** по образцу (Format Painter) сама вернется в исходное положение.

* При использовании двойного щелчка ЛКМ по кнопке, она не вернется в исходное положение и можно будет копировать формат оригинала на различные фрагменты текста. По завершении работы с копированием форматов нажмите ESC.

Очистка формата

Для отказа от всего примененного форматирования можно воспользоваться командой **Очистить формат**. Для этого:

1. Выделить фрагмент текста, формат которого необходимо очистить.
2. Открыть раздел **Показать форматирование** (Reveal Formatting).
3. В группе **Выделенный текст** (Selected text), щелкнуть по кнопке списка текстового поля, в котором отображается выделенный текст.
4. Выбрать из списка **Очистить формат** (Clear Formatting).

* Также можно воспользоваться разделом ОЗ **Стили**. В этом случае, выделить текст, с которого необходимо снять форматирование и на вкладке **Главная** в группе **Стили** нажать на кнопку справа от названия группы. В открывшемся разделе выбрать **Очистить все**.

Табуляция

Табуляция — это перемещение текста, находящегося справа от курсора и самого курсора, на фиксированное расстояние или к определенному положению на линейке.

Табуляция позволяет изменять отступы и производить выравнивание фрагментов текста в строке. При нажатии клавиши TAB курсор (и любой текст справа от него) перемещается к следующей позиции табуляции или на фиксированное расстояние. По умолчанию MS Word устанавливает позиции табуляции через каждые 0,5 дюйма (1.27 см) по всей ширине страницы. Можно изменять расположение позиций табуляции и управлять способом выравнивания текста по позиции табуляции.

Типы позиций табуляции

Существует **пять** основных типов позиций табуляции:

• **Выравнивание по левому краю**. Левый край текста выравнивается по позиции табуляции. В программе MS Word по умолчанию позиции табуляции выровнены влево.

• **Выравнивание по правому краю**. Правый край текста выравнивается по позиции табуляции.

• **Выравнивание по центр**у. Текст выравнивается по центру по позиции табуляции.

• **Выравнивание по десятичному разделителю**. Десятичный разделитель (точка или запятая) выравнивается по позиции табуляции (используется для выравнивания колонок или чисел).

• **С чертой**. Вставка вертикальной черты в позиции табуляции. Не используется для выравнивания текста.

Рис. 3.1 отображает результаты применения четырех вариантов выравнивания табуляции. Также показаны четыре различных символа, которые изображены на Линейке для того, чтобы указывать позиции табуляции.

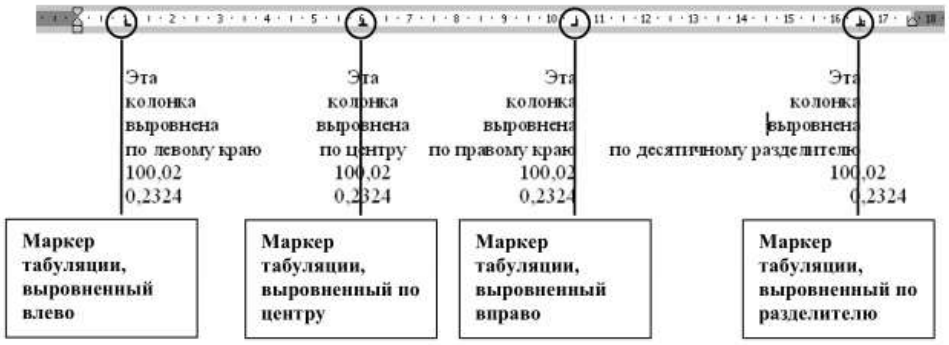


Рис. 2. Типы позиций табуляции

* Когда вы нажимаете клавишу TAB один раз для абзаца, MS Word сдвигает первую строку абзаца. Когда вы нажимаете клавишу TAB снова, MS Word сдвигает целый абзац, сохраняя красную строку

Быстрее всего установить табуляцию с помощью Линейки (Ruler).

Окно диалога Табуляция (Tabs) позволяет более точно установить позиции табуляции и задать дополнительные параметры табуляции.

Списки

Список — это набор абзацев (элементов списка), особым образом отформатированных с помощью номеров или специальных маркеров.

Возможно использование списков двух типов:

маркированных и нумерованных.

Пример. Маркированный список.

• табурет;
• тумба;
• шкаф;
• диван.

Нумерованные могут быть одноуровневыми и многоуровыми списками.

Пример. Одноуровневый список.

1. Табурет.
2. Тумба.
3. Шкаф.
4. Диван.

Пример. Многоуровневый список.

1. Мебель.

1.1. Стол.
1.1.1. Стол обеденный.
1.1.2. Стол кухонный.
1.1.3. Прикроватный столик.
1.2. Шкаф.
1.2.1. Шкаф-купе.
1.2.2. Трехстворчатый шкаф.
1.3. Стул.
1.3.1. Офисный стул.
1.3.2. Детский стул.
2. Посуда.
2.1. Чайный сервиз.
2.2. Одноразовая посуда.

При создании нумерованного списка поддерживается автоматическая нумерация каждого абзаца, т.е. при удалении/добавлении одного (или более) абзаца MS Word автоматически пересчитывает нумерацию.

При создании маркированного списка маркеры появляются перед каждым абзацем, до тех пор, пока вы не откажетесь от маркированного списка.

Работа с таблицами

Таблица состоит из строк и столбцов, на пересечении которых находятся ячейки, которые могут быть заполнены текстом и графикой.

Вы можете использовать таблицу для организованного размещения текста и графики на странице, а также производить сортировку или выполнять простые вычисления над данными, размещенными в таблице.

Все данные заносятся в ячейки таблицы. В ячейку таблицы вы можете добавить другую таблицу, т.е. работать с вложенными таблицами.

Внутри каждой ячейки вы можете задать свои способы форматирования, текста и/или абзаца. Когда вы вводите информацию в ячейку, высота строки и/или ширина столбца увеличивается автоматически. При удалении информаци и, высота строки и/или ширина столбца будет уменьшаться автоматически.

Таблица в программе MS Word является плавающей, т.е. вы можете расположить ее в любом месте в документе. Таблица может быть длинной, в том числе размещаться на нескольких страницах.

После добавления таблицы в документ на ленте появляются две новые вкладки: Конструктор и Макет, содержащие команды для работы с таблицей.

Схема для создания и работы с таблицей в документе

1. Решите, сколько строк/столбцов будет в вашей таблице, какая структура будет у таблицы (простая/сложная).
2. Создайте таблицу .
3. Заполните ячейки таблицы данными.
4. Добавьте/удалите строки/столбцы.
5. Отформатируйте таблицу: выделите шапку таблицы (одна или несколько первых срок, содержащие подписи значений столбцов), задайте границы для ячеек, общую рамку, укажите выравнивание данных в ячейках.

6. Увеличьте/уменьшите ширину строк/столбцов

7. Установите возможность повтора заголовков, в этом случае если ваша таблица перенесется на следующую страницу, то заголовки столбцов будут повторяться автоматически на всех страницах, где расположена таблица.

Создание таблицы

Таблица может быть простой или сложной. В простой таблице количество ячеек во всех столбцах и строчках одинаковое. Сложная таблица содержит в себе объединенные ячейки.

Для создания таблицы используйте на вкладке Вставка в группе Таблицы кнопку Таблица. После нажатия на нее вы можете для добавления таблицы выбрать один из следующих путей:

• макет Вставка таблицы – для быстрой вставки таблицы;

• кнопку Вставить таблицу… - откроется ОД Вставка таблицы;

• кнопку Нарисовать таблицу – для ручного рисования «карандашом» таблицы;

• кнопку Экспресс-таблицы – для применения встроенных форматов таблицы.

Вы можете создать простую таблицу, которую в последующем можете перестроить в сложную. При помощи карандаша вы сразу можете создать таблицу сложной структуры.

При создании таблицы не обязательно использовать или устанавливать все параметры сразу. Достаточно создать одну ячейку таблицы, которую в последующем можно увеличить до таблицы нужной структуры.

Создание таблицы с помощью макета

1. Переместите курсор мышки в тексте, после которого вы хотите добавить таблицу.

2. В группе Таблицы нажмите кнопку Таблица.

3. В макете Вставка таблицы переместите указатель мыши на нужное количество столбцов (ячейки вправо) и нужное количество строк (ячейки вниз)

4. Щелкните левой кнопкой мыши после того, как выделите нужное количество строк/столбцов.

MS Word добавит таблицу в ваш документ.

Создание таблицы с помощью окна диалога Вставка Таблицы.

Для создания больших, заранее оформленных таблиц используйте следующие шаги:

1. Переместите курсор мышки в тексте, после которого вы хотите вставить таблицу.

2. В группе Таблицы нажмите кнопку Таблица - Вставить таблицу…

Откроется окно диалога Вставка таблицы (Insert Table)

3. Задайте число столбцов/строк в соответствующих полях с помощью кнопок увеличить/уменьшить значение или введя с клавиатуры требуемое количество.

* Установите автоподбор ширины столбцов таблицы, выбрав из параметров Постоянная (Fixed Column width), По содержимому (AutoFit to Contents), По ширине окна (AutoFit to window). Для параметра Постоянная (Fixed Column width) вы можете дополнительно настроить фиксированную или автоматически подбираемую ширину столбцов в сантиметрах (значение Авто (Auto) устанавливает ширину столбцов пропорциональную их числу).

4. Нажмите ОК.

MS Word добавит таблицу в документ.

Создание таблицы при помощи карандаша

Когда вы используете карандаш, вы рисуете таблицу как, на бумаге.

1. В группе Таблицы нажмите кнопку Таблица ? Нарисовать таблицу (Draw Table). Указатель мышки примет вид карандашика .

2. Переместите УМ в то место в документе, где вы хотите нарисовать

таблицу 3. Нарисуйте сначала внешние границы таблицы: нажмите ЛКМ и, удерживая нажатой, потяните по диагонали вниз и вправо.

4. Затем рисуйте карандашом строки, столбцы.

* Вы можете стереть лишнюю линию с помощью кнопки Ластик (Eraser) на вкладке Конструктор в группе Нарисовать границы:

нажмите на кнопку и проведите по линии, которую хотите удалить.

* Рисуя таблицу с помощью карандаша, вы можете сразу создать таблицу сложной структуры.

* Вы можете указать тип, толщину, цвет проводимой линии с помощью кнопок на вкладке Конструктор в группе Нарисовать границы. Выберите необходимые установки, а потом проведите линию карандашом

Выделение элементов таблицы

Прежде чем использовать команды редактирования или форматирования таблицы, необходимо выделить ячейки, строки или столбцы, к которым эти команды будут применяться.

представлены положения и вид указателя мыши для выделения элемента таблицы:

* Если вам нужно выделить несколько элементов, нажмите левую кнопку мыши и, удерживая нажатой, укажите нужное количество элементов.

Вставка/удаление элементов таблицы

В процессе работы с таблицей бывает необходимо добавлять или удалять строки таблицы, столбцы или ячейки. Вы также можете вставить другую таблицу в ячейку — создать вложенные таблицы.

* Для добавления/удаления нескольких элементов таблицы, сначала необходимо выделить требуемое количество элементов.

Выделитьстроку

Выделить ячейку

Выделить столбец

Выделить таблицу

Используйте маркер для изменения размера таблицы

Ластик
Толщина линии
Тип линии
Цвет линии
Элемент Действия Вставка
Строка Курсор должен находиться в строке, перед (после) которой вы ➔ выберите вариант Вставить сверху / Вставить снизу.
Для добавления строки в самый конец таблицы, вы можете использовать следующее: переместитесь в последнюю ячейку таблицы ➔ нажмите TAB.
Можно использовать правую кнопку мыши ? Вставить ?Вставить строки сверху / снизу.
Столбец Курсор должен находиться в столбце, левее (правее) которого вы хотите
добавить пустой столбец. Вкладка Макет ? группа Строки и столбцы ? выберите вариант Вставить слева / Вставить справа. Можно использовать ПКМ ? Вставить ?Вставить столбцы слева / справа.
Ячейка Курсор должен находиться в ячейке, до (выше) которой хотите добавить
новую ячейку. Вкладка Макет ? группа Строки и столбцы ? Нажмите кнопку запуска ОД (справа от названия группы) ? откроется ОД
Добавление ячеек? выберите способ смещения ячеек
Вложенная
Таблица
Курсор должен находиться в ячейке, куда вы хотите добавить новую таблицу ? Вкладка Вставка ? Таблица ? Вставить таблицу… ?
Установить параметры таблицы ? ОК
Удаление
Строка Курсор должен находиться на строке, которую вы хотите удалить. Вкладка
Макет ? группа Строки и столбцы ? Удалить (Delete) ? Удалить строки (Rows). Можно использовать ПКМ ? Удалить строки.
Столбец Курсор должен находиться в столбце, который вы хотите удалить. Вкладка
Макет ? группа Строки и столбцы ? Удалить (Delete) ? Удалить столбцы (Columns). Можно использовать ПКМ ? Удалить столбцы.
Ячейка Курсор должен находиться в ячейке, которую хотите удалить. Вкладка
Макет ? группа Строки и столбцы ? Удалить (Delete) ? Удалить ячейки… Можно использовать ПКМ ? Удалить ячейки…(Delete Cells…) ? выберите способ смещения ячеек
Вложенная
Таблица
Курсор должен находиться внутри вложенной таблицы. Вкладка Макет ? группа Строки и столбцы ? Удалить (Delete) ? Удалить таблицу.
? Вместе с элементом таблицы удаляется и его содержимое.

? Если вы хотите удалить данные в ячейке, строке, столбце, выделите нужный элемент и нажмите DELETE.
? При использовании операций на вставку/удаление ячеек таблицы изменяется структура таблицы, поэтому старайтесь их не использовать!
51
12.6. Объединение/разбиение ячеек таблицы
Для придания простой таблице более сложной формы, используются операции объединения и разбиения ячеек.
12.6.1. Объединение ячеек
Две или более смежных ячейки, можно объединить в одну.
Например, путем объединения нескольких ячеек, расположенных в одной строке, можно создать заголовок таблицы, общий для нескольких столбцов.
Ячейки могут быть объединены как по горизонтали, так и по вертикали. Вы можете сначала объединить ячейки, а потом ввести текст, либо ввести текст, а потом объединить ячейки.
Для объединения ячеек используйте следующие шаги:
1. Выделите ячейки, которые вы хотите объединить.
2. Вкладка Макет ? Группа Объединить ? Объединить ячейки (Merge Cells)
3. Продолжайте выделять и объединять ячейки, чтобы создать сложную таблицу.
В Табл. 4.2 приведен пример таблицы со сложной структурой, в которой было использовано вертикальное и горизонтальное объединение ячеек.
Табл. 4.2. Изменение структуры и форматирование таблицы
Исходная таблица
Действие Результат
1. Выделите ячейки, содержащие слово Дата, и пустую ячейку слева.
2. ПКМ ? Объединить ячейки (Merge Cells)
3. Задайте Выравнивание ячеек По центру (Align center)
1. Выделите ячейки, содержащие слово Итого, и пустую ячейку снизу.
2. ПКМ ? Объединить ячейки (Merge Cells)
3. Задайте Выравнивание ячеек по центру (Align center)
12.6.2. Разбиение ячеек.
1. Поместите курсор в ячейку, которую вы хотите разбить.
? Или выделите несколько смежных ячеек.
52

2. Вкладка Макет ? Группа Объединить ? Разбить ячейки…(Split Cells...) или ПКМ ? Разбить ячейки…
3. Укажите требуемое число строк и столбцов в появившемся ОД.
4. Нажмите ОК.

Форматирование таблицы

Для форматирования текста, размещенного в элементе таблицы, необходимо применить форматирование к этому элементу. Вы можете сначала задать параметры форматирования для элемента таблицы, а потом впечатать текст. Либо вы можете определить параметры форматирования для элемента таблицы, в котором уже есть содержимое, в этом случае нужно выделить элемент таблицы, для которой будете задавать параметры форматирования.

Форматирования текста в ячейках таблицы

Для текста и абзацев в ячейке вы можете задать изменить следующие настройки форматирования:

- Шрифт, его размер, подчеркивание
- Расстояние между строк,
- отступы в абзаце (красная строка)
- Выравнивание текста
- Заливка, границы
- Ориентация текста в ячейке

Автоматическое форматирование таблицы

Поместите курсор внутрь таблицы ➔ вкладка Конструктор ➔ Группа Стили таблиц. Примените.

Установка шапки таблицы

Когда ваша таблица очень большая и занимает несколько печатных страниц, желательно, копировать шапку таблицы (заголовки столбцов) на каждой новой странице.

Для создания постоянной шапки таблицы выполните следующее:

1. Выделите шапку таблицы.
2. Вкладка Макет ➔ Повторить строки заголовков.

Теперь, когда одна страница с таблицей закончится, и таблица автоматически перейдет на следующую страницу, MS Word автоматически добавит заголовки столбцов.

* Вы также можете использовать Вкладка Макет ➔ группа Таблица (Table) ➔ Свойства (Properties) ➔ вкладка Строка (Rows) ➔ флажок Повторять как заголовок на каждой странице (Repeat as header row at the top of each page).

Изменение ширины/высоты столбца/строки

Когда вы вводите текст в ячейку, высота строки и ширина столбца, по умолчанию, изменяется автоматически. Вы можете установить ширину/высоту столбца/строки по вашему желанию. Для этого выполните следующее:

1. Подведите указатель мыши к границе элемента таблицы, ширину/высоту которого вы хотите изменить.

* Для изменения ширины столбца, подведите указатель мыши к правой границе столбца, так, чтобы указатель мыши принял вид двунаправленной стрелочки .

* Для изменения высоты строки, подведите указатель мыши к нижней границе строки, так, чтобы указатель мыши принял вид двунаправленной стрелочки .

2. Нажмите левую кнопку мыши и, удерживая нажатой, потяните в сторону увеличения/уменьшения ширины/высоты элемента таблицы.

* Если вы хотите изменить ширину/высоту одной ячейки, выделите сначала эту ячейку, а потом измените границу.

* Если вы хотите, чтобы все столбцы имели одинаковую ширину / строки одинаковую высоту, используйте следующее:

выделите столбцы → вкладка Макет → группа Размер ячейки → кнопки Выровнять ширину столбцов / Выровнять высоту строк.

* Вы можете установить автоматический подбор размера ячейки в зависимости отводимого текста, таблицы по ширине окна или фиксированную ширину столбца: вкладка Макет → Группа Размер ячейки → Кнопка Автоподбор → Выберите требуемую команду.

Работа с формулами

Текстовый редактор MS Word позволяет производить простые расчеты в таблицах. Однако лучше использовать MS Word как текстовый редактор, а более сложные вычисления производить в программе MS Excel.

Для ввода формулы выполните следующие действия:

1. Переместите курсор в ячейку, где вы хотите получить результат вычислений.

2. Вкладка → Макет → Группа Данные → Нажмите кнопку Формула.

Откроется окно диалога Формула.

3. Введите формулу или выберите ее из окна Вставить функцию. В качестве аргумента функции укажите LEFT, если оперируете данными слева от ячейки с формулой или ABOVE – если сверху. В ячейке отобразится результат вычислений.

* Если в вашей таблице не было ячеек с числовыми значениями, то формула вернет ответ "!Ошибка в формуле" (!Unexpected End of Formula).

* При чередовании ячеек с числами и текстом MS Word вычисляет только сумму ближайшего числа к ячейке с результатом.

* Если значения в ячейках изменились, формула автоматически непересчитывается. В этом случае нужно пересчитать (обновить) формулу: выделите ячейку с формулой → нажмите **F9**. Если в вашей таблице много формул, которые нужно пересчитать, выделите сначала всю таблицу, а потом нажмите **F9.**

Перемещение таблицы

1. Переместите указатель мышки к маркеру выделения таблицы (Рис. 4.6).

2. Нажмите ЛКМ и, удерживая нажатой, переместите в новое место в документе. MS Word поместит таблицу, куда вы определили.

* Если вы переместили таблицу внутрь текста, то можете указать возможность обтекания текстом границ таблицы: Вкладка Макет → Группа Таблица → Свойства → вкладка Таблица → Обтекание Вокруг.

Удаление таблицы

1. Переместить курсор внутри таблицы.
2. Вкладка Макет →Группа Строки и столбцы → Удалить Удалить таблицу.

В этом случае вы удалите и содержимое, и саму таблицу.

* Если в таблице есть вложенные таблицы, и вы удаляете основную (внешнюю) таблицу, вложенная тоже удаляется. В случае, если вы удаляете вложенную, внешняя таблица остается.

* Удалить таблицу также можно, нажав на кнопку Ластик (Eariser), и удерживая нажатой левую кнопку мыши, выделить всю таблицу.

Стили

Стиль — это набор параметров форматирования символов, абзацев, таблиц, списков которые хранятся под определенным именем.

Применение созданного ранее стиля абзаца к выделенному абзацу позволяет задать сразу несколько параметров форматирования абзаца и т.д.

Использование стилей позволяет быстро выполнять оформление текста в едином художественном стиле, а также использовать некоторые автоматизированные способы работы с документом, как, например, создание оглавления.

Существуют два типа стилей текста: стиль абзацев и стиль символов.

Стиль абзаца может быть применен только к абзацу или нескольким абзацам, а стиль символа — только к символу или нескольким символам.

Вы также можете использовать стили таблиц и списков.

Стили, доступные в данном документе, объединены в список стилей.

Вы можете легко просмотреть список стилей на вкладке Главная в группе Стили (Рис. 5.1):

Рис. 5.1. Поле стилей

* Также можно использовать раздел Стили области задач. Для его открытия нажмите кнопку справа от названия группы.

Применение стилей

1. Поместите курсор в абзац или выделите несколько абзацев, которые требуется отформатировать определенным стилем.

* Чтобы применить стиль символа, выделите символ или группу символов, которые надо отформатировать.

2. На вкладке Главная в группе Стили наведите указатель мыши на выбранный стиль, просмотрите автоматическое изменение текста согласно стиля, выберите нужный тип и щелкните ЛКМ.

* Если нужный стиль отсутствует в списке откройте раздел Стили, нажав кнопку справа от названия группы.

Создание стилей

Вы можете создать собственные стили и использовать их для оформления данного и других документов.

Создание стиля с помощью окна диалога Создание стиля.

1. Создайте новый абзац.

2. Выберите вкладку Главная ➔ Группа Стили ➔ нажмите кнопку для открытия раздела Стили.

Откроется раздел Стили.

3. Нажмите кнопку Создать стиль .

Откроется окно диалога Создание стиля (New Style) (Рис. 5.2)

4. Задайте необходимые параметры в окне диалога и нажмите OK.

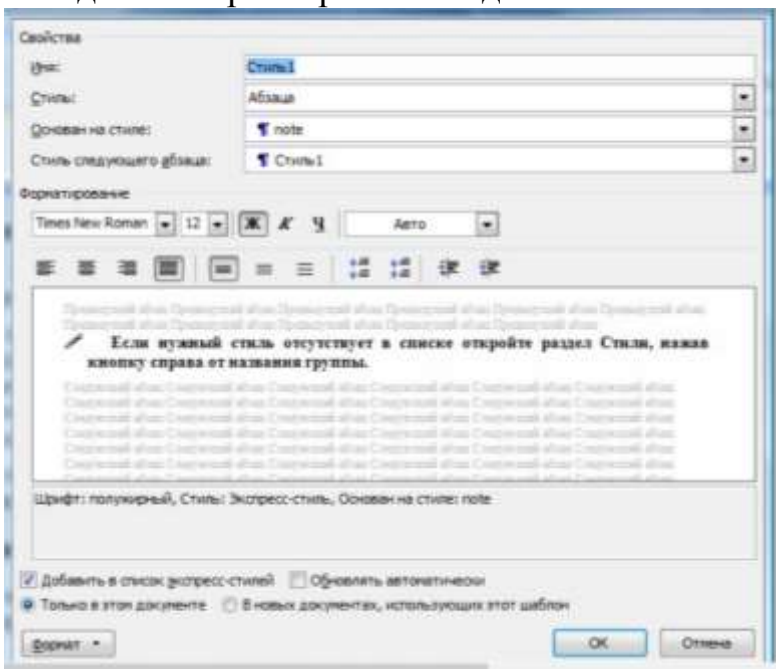


Рис. 5.2. ОД Создание стиля

Описание полей, которые можно настроить при создании нового стиля в ОД Создание Стиля (New Style) приведено в Табл. 5.1 .

Имя (Name)-Позволяет задать имя вновь создаваемому стилю

Стиль (Style Type)- Позволяет задать тип стиля.

Чтобы создать новый стиль абзаца,выберите Абзаца (Paragraph).

Чтобы создать новый стиль символов, выберите Символа (Symbol). Этот параметр недоступен при изменении существующего стиля, т.к. тип существующего стиля изменить нельзя

Основан на стиле (Style based on) - Позволяет указать имя существующего стиля, на котором основан новый или изменяемый стиль. При создании стиля для оформления заголовков, укажите в данном поле один из существующих стилей заголовков, соответствующего уровня иерархии.

Стиль следующего абзаца (Style for the following paragraph) - Позволяет задать стиль, который будет автоматически применен к следующему абзацу. При нажатии клавиши ENTER в конце абзаца, оформленного новым или измененным стилем, к следующему абзацу будет применен Стиль следующего абзаца

Форматирование (Formatting)- Позволяет задать форматирование с помощью кнопок: шрифт, размер шрифта, цвет текста, начертание текста (полужирный, наклонный, подчеркнутый), выравнивание абзаца, междустрочный интервала, интервалы до или после абзаца, отступы абзаца. В этой области так же находятся поле образца будущего стиля и краткое описание примененного форматирования.

Добавить в список экспресс-стилей - Добавляет стиль в экспресс-стили — наборы стилей, предназначенных для совместного использования, которые позволят в дальнейшем задавать формат для целых документов в зависимости от их предназначения.

Только в этом документе - Это делает стиль доступным для текущего документа.

В новых документах, использующих этот шаблон - Позволяет добавить стиль в шаблон, присоединенный к активному документу. Это делает стиль доступным для всех документов, основанных на этом шаблоне.

Обновлять автоматически (Automatically update) - Позволяет задать автоматическое переопределение стиля в случае применения дополнительного форматирования к любому абзацу, оформленному этим стилем. При этом обновляется форматирование всех абзацев документа, оформленных этим стилем.

Формат ▼ -(Format ▼) - Позволяет получить доступ к списку команд форматирования, использующихся для определения стиля.

При выборе атрибута, открывается соответствующее ОД. Внесите все необходимые изменения в ОД и нажмите кнопку OK, для корректного завершения работы с ОД и сохранения внесенных изменений

* Созданный вами стиль можно удалить.

* Встроенные стили удалить нельзя.

1. Табличный процессор Excel

При всей многогранности и сложности, информационные технологии базируются на нескольких фундаментальных видах работ, выполняемых с данными на разных этапах.

К ним относятся:

- получение (сбор) данных,
- обработка данных,
- хранение данных,
- представление данных.

В различных задачах на первое место выходят те или иные виды работ, однако в целом, они неотделимы друг от друга.

Действительно, невозможно представить данные, если они не собраны и сохранены,

Нет смысла собирать и хранить данные, если не предполагается их обработка и использование, подразумевающее доступ к собранным данным и сохранение результатов обработки.

Обработка данных предполагает их изменение и может быть разделена на несколько видов в зависимости от того, что изменяется в обрабатываемых данных.

Можно выделить **три** основных вида:

1. Обработка, которая связана с изменением значений данных. **Например**, для числовых типов это соответствует вычислениям, которые выполняются с помощью базовых арифметических операций MSExcel (сложение, умножение и т.д.) или с помощью функций, которые вместе с числовыми константами и переменными величинами объединяются в арифметическое выражение с помощью «формул».

Аналогично с помощью логических функций (И, ИЛИ, НЕ) и логических констант (переменных), создаются логические выражения.

Подобным образом конструируются и текстовые выражения.

2. Обработка, которая связана с изменением **структуры, состава** или **взаимного расположения данных**, но не связанная с изменением их значений.

Примерами такой обработки служат «**сортировка**» и «**группировка**» данных.

Такие виды обработки выполняются, как правило, с помощью соответствующих инструментов MSExcel, доступ к которым осуществляется через **меню.**

3. **Комбинированная** обработка, которая объединяет в себе оба предыдущих вида. К ней относятся: например, «**консолидация**» данных,

«**промежуточные итоги**» или обработка, выполняемая инструментом «**сводные таблицы**».

Следует отметить, что иногда одинаковые, по сути, виды обработки можно выполнить разными способами - с помощью специализированного инструмента или с помощью **функций**. Это относится к вычислению «промежуточных итогов».

Общее правило состоит в том, что «инструменты» MSExcel используются, как правило, для **агрегированных** данных (то есть взаимосвзанных - массивов, диапазонов и т. п.), но обладают ограниченными возможностями в смысле вычислений, в то время как «формулы» и «функции» ориентированы преимущественно на выполнение сложных расчетных алгоритмов (вычислений), с **произвольно расположенными** данными. Это существенное отличие нивелируется возможностью копирования и переноса формул.

Табличный процессор Excel

Вводные сведения

1.1. Файл Excel

Одним из главных свойств программы Excel является возможность работы с большим количеством **форматов** файлов. Программы Excel 2007, Excel 2010 поддерживают файлы всех форматов, созданных в более ранних версиях Excel.

Excel 2007 поддерживает следующие основные форматы файлов:

XLSX – файлы рабочих книг, не содержащих макросы;

XLSM - файлы рабочих книг, содержащих макросы;

XLTX- файлы шаблонов рабочих книг, не содержащих макросы;

XLTM- файлы шаблонов рабочих книг, содержащих макросы;

XLSB – формат двоичных файлов;

XLSK – формат файлов резервного копирования.

Файл Excel (рабочая книга) содержит по умолчанию три листа, на которых могут быть размещены таблицы или диаграммы.

Табличный лист содержит 1 048 576 (2^20) строк и 16 384 (2^14) столбцов, таким образом, лист содержит около 17 млрд ячеек.

В Excel 2007 использован новый пользовательский интерфейс, который заменил интерфейс, основанный на системе меню и панелях инструментов. Новый интерфейс получил название «ленточный», поскольку его основными элементами стали ленты и вкладки. Из новшеств интерфейса можно также отметить шесть добавленных шрифтов и панель быстрого доступа. Лента Excel 2007 показана на рис. 1.

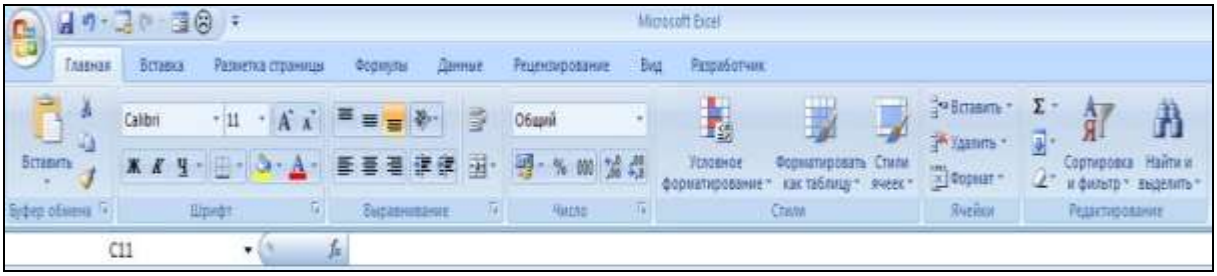


Рис. 1. Ленточный интерфейс

1.2. Ввод данных

В Excel различаются следующие «типы» данных: *Число, Текст, Дата и время.*

Числа по умолчанию выравниваются по правому краю, содержат символы от 0 до 9, +, минус,(), %,E и знаки денежных единиц, в случае недостаточной ширины ячейки в ней отображаются символы #.

Текст может содержать до 32000 символов и по умолчанию выравнивается по левому краю. В неоднозначных случаях для идентификации данных как текста, их предваряют апострофом.

При вводе **дат** и **времени** Excel преобразует их в «порядковые числа». Самая ранняя дата, с которой оперирует Excel это 1 января 1900 года. Этой дате присваивается порядковый номер – 1. Последующие даты имеют порядковые номера, зависящие от того, сколько дней прошло от 1 января 1900 года до вводимой даты.

Такая система представления значительно облегчает использования дат в формулах.

При вводе **времени** Excel трактует время как дробную часть суток. Так полдень соответствует значению 0,5.

Редактирование введенных данных осуществляется либо в строке формул, при установке курсора на соответствующую ячейку, либо при нажатии клавиши F2, которая позволит отредактировать данные непосредственно в ячейке.

Для упрощения **ввода данных** используется ряд приемов, из которых стоит обратить внимание на использования **списков** (встроенные списки позволяют вводить названия дней недели и месяцев при протаскивании **маркера автозаполнения**).

Заполнение ячеек данными арифметической или геометрической **прогрессии** с произвольным шагом также облегчает ввод данных.

При вводе даты ячейки могут быть заполнены по дням, месяцам, годам или рабочим дням. Применение прогрессии и заполнение датами обеспечивается использованием контекстного меню и протаскиванием маркера автозаполнения.

1.3. Адресация в Excel

В Excel различают следующие виды адресации:

- абсолютную,
- относительную,
- смешанную,
- с помощью имен.

В таблице на **рис. 2**, необходимо вычислить стоимость товаров и найти долю стоимости каждого товара в общей сумме стоимостей.

При **копировании формулы**, записанной в ячейку D2 в диапазон D3:D6, адреса ячеек автоматически **изменяются** относительно активной ячейки, т. е. в ссылках ячейки D2 применена *относительная адресация*.

Попытка использования относительной адресации в диапазоне E2:E8 привела к ошибке (деление на ноль), поскольку при относительной адресации в ячейке E3 будет сформирована формула D3/D7, т. е. сделана попытка деления на пустую (нулевую) ячейку.

Ссылка на ячейку **D7** не должна изменяться, другими словами быть *абсолютной*, а поскольку копирование производится в пределах одного столбца, достаточно сделать ее смешанной, как показано в выноске рис. 2.

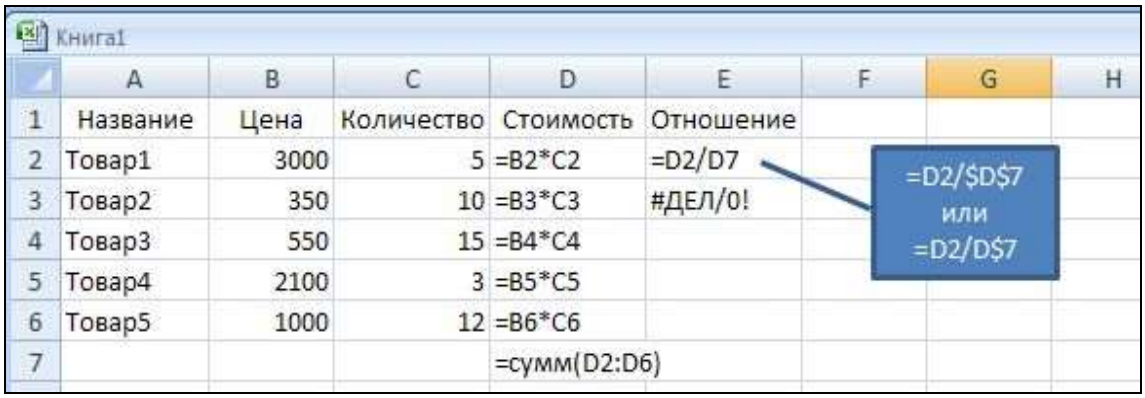


Рис. 2. Типы адресации

Изменить тип ссылки можно последовательным нажатием клавиши F4 при установке курсора перед ссылкой, подлежащей изменению.

Ячейка или **диапазон** могут иметь **имя**, которое может быть использовано в формуле. Так, если присвоить ячейке D7 Итог, то формула в ячейке E2 преобразуется к виду: D2/Итог.

Чтобы присвоить имя ячейке или диапазону, нужно воспользоваться командой **Формулы – Определенные имена – Присвоить имя**.

1.4. Использование формул

Вычисления в Excel производятся по формулам.

Формула представляет собой арифметическое или логическое выражение. Это совокупность констант, ссылок, функций, имен диапазонов, соединенных знаками арифметических или логических операций.

Результатом вычисления арифметического выражения является число, а логического – значение ИСТИНА или ЛОЖЬ.

В случае использования в выражении текстовых операндов результатом также может быть текст.

Формуле предшествует знак равенства (=).

Арифметические операции в порядке уменьшения их приоритета следующие: [-] (одноместный минус),
[%] (процент),
[^] (возведение в степень),
[*] , [/] (умножение и деление),
[+] , [–] (сложение и вычитание).

Логические операции:[>], [<], [>=], [<=], [=], [<>] (не равно).

Для строк используется операция конкатенации [&].

Кроме этих операций в Excel используются операции диапазон [:], объединение [;] и пересечение [ ] – знак пробела (рис. 3).

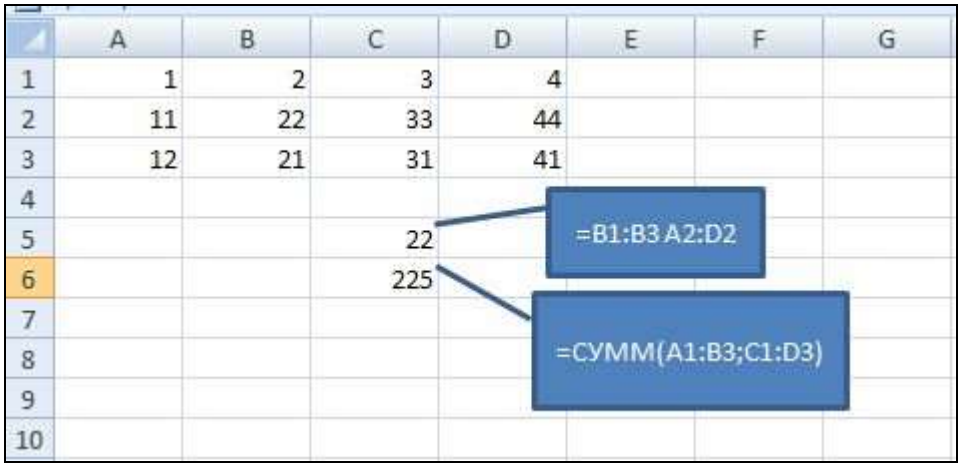
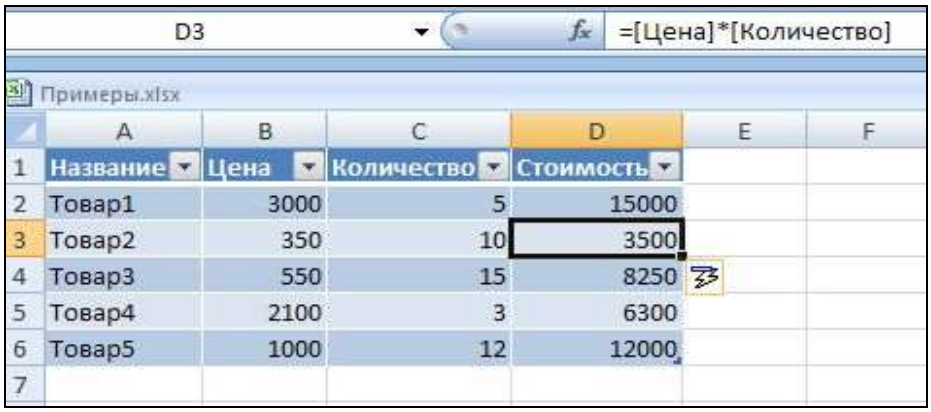


Рис. 3. Операции над диапазонами

Excel 2007 предлагает новый способ записи ссылок на данные, организованные в виде таблицы (Вставка – Таблицы – Таблица) (рис. 4).

Такая таблица имеет собственное имя, например Таблица1, поэтому можно ссылаться на все ячейки этой таблицы, используя ее имя: =СУММ(Таблица1). Данная формула просуммирует все ячейки таблицы.

В формулах можно использовать также заголовки столбцов таблицы, например=[Количество]*[Цена], если «Количество» и «Цена» – заголовки соответствующих столбцов таблицы.



	A	B	C	D	E	F
1	Название	Цена	Количество	Стоимость		
2	Товар1	3000	5	15000		
3	Товар2	350	10	3500		
4	Товар3	550	15	8250		
5	Товар4	2100	3	6300		
6	Товар5	1000	12	12000		
7						

Рис. 4. Организация данных в виде таблицы

Еще одно преимущество использования такого вида таблиц заключается в упрощенном вводе формул. Достаточно ввести формулу в одну ячейку столбца, в остальные ячейки она распространяется автоматически.

Формулы могут ссылаться на ячейки других листов – Лист2!C3,
других книг – [Отчет]Лист2!C3,
и ячейки закрытых книг – ‘E:\Мои документы\[Отчет]Лист2!’C3.

При возникновении синтаксических ошибок, Excel выдает сообщение об ошибке, например #ДЕЛ/0!, и прекращает вычисления.

Для отслеживания ошибок можно воспользоваться просмотром взаимосвязи ячеек, этот механизм включается командой **Формулы – Зависимости формул – Влияющие или Зависимые ячейки**.

При этом для активной ячейки будут показаны ячейки, которые влияют на результат вычислений, и ячейки, которые зависят от полученного результата.

Помимо синтаксических ошибок в ходе вычислений могут возникать так называемые «смысловые» ошибки, которые не фиксируются Excel (например, ввод значения 1000 в поле «вес»).

Для их предотвращения используется механизм применения ограничений на величину или тип вводимых данных.

Этот механизм реализован с помощью команды **Данные – Работа с данными – Проверка данных**.

Для диапазона, предназначенного для ввода данных, можно установить границы вводимых данных, указать сообщения, которые пользователь увидит при попытке ввода некорректных данных и в начале работы с диапазоном ввода.

Особенно удачным приемом, минимизирующим ошибки ввода, является создание списка возможных значений вводимых данных.

1.5. Форматирование

Форматирование позволяет улучшить восприятие табличной информации. В Excel 2007 значительно расширены возможности условного форматирования.

Условное форматирование позволяет применять различные форматы в зависимости от значений, содержащихся в ячейках.

Форматированию подвергаются как значения, хранящиеся в ячейках таблицы, так и структурные ее составляющие.

Диалоговое окно «Формат ячейки» позволяет установить вид числовых данных (количество знаков после запятой, цвет данных с отрицательными значениями, разделение троек разрядов), выбрать формат дат и денежных единиц (**Главная – Число**).

Выравнивание (**Главная – Выравнивание**) дает возможность выполнить традиционные действия с содержимым ячейки (выровнять по левому, правому краям или по центру), а также объединить ячейки, перенести данные по словам и применить требуемую ориентацию.

Шрифтовое оформление осуществляется с использованием команды **Главная – Шрифт**.

Изменение свойств структурных составляющих – ширины столбцов и высоты строк осуществляется с помощью команды **Главная – Ячейки**. Эта же команда позволяет осуществлять автоподбор ширины столбцов и высоты строк.

Команда **Главная – Стили** дает возможность применять стили как целиком к таблице, так и к отдельным ячейкам.

Команда **Главная – Стили – Условное форматирование** позволяет применять правила выбора ячеек в зависимости от их содержимого, применять гистограммы в ячейках, использовать наборы значков и цветовые шкалы. На рис. 5 показано использование набора значков для выделения значений <33%, от 33% до 66% и >66% выполнения дипломной работы. Снятие форматирования обеспечивается командой **Главная – Редактирование – Очистить форматы**. Копирование формата происходит с помощью пиктограммы «Формат по образцу».

	A	B	C
1	Выполнение дипломной работы		
2	Фамилия	15.апр	15.май
3	Анненков	15%	60%
4	Борисова	50%	80%
5	Валеев	10%	10%
6	Грибов	0%	0%
7	Давыдова	55%	80%
8	Емельянова	45%	90%
9	Павлова	30%	55%
10	Родионов	0%	25%
11	Семенов	45%	85%
12	Тимофеева	25%	75%
13	Циммерман	35%	90%
14	Яковенко	10%	40%
15			

Рис. 5. Условное форматирование

1.6. Диаграммы в Excel

Создание диаграмм – это способ **наглядного представления данных,** приведенных в таблице в виде чисел. Анализ чисел и их сравнение существенно упрощается при графическом отображении данных.

Диаграмма представляет собой объект, который формируется на основе **рядов данных**, расположенных в строках или столбцах таблицы. Каждому ряду соответствует свой *маркер* диаграммы. Маркеры показаны в *легенде*, расположенной рядом с диаграммой.

Диаграммы могут располагаться непосредственно в рабочем листе, такая диаграмма называется *внедренной*, или на отдельном листе, который называется *листом диаграммы*.

При выделении объекта внедренной диаграммы или при переходе на лист диаграммы, активизируются три новые ленточные вкладки под общим названием «Работа с диаграммами». Эти вкладки содержат все необходимые команды, чтобы форматировать, редактировать диаграмму и ее элементы.

Excel содержит около сотни вариантов построения диаграмм, выбор которых зависит от конкретной задачи.

Для создания диаграмм нужно воспользоваться командой **Вставка – Диаграмма**.

На рис. 6 представлена гистограмма и ее элементы, а также исходная таблица и ленточные вкладки для работы с диаграммами («Конструктор», «Макет» и «Формат»).

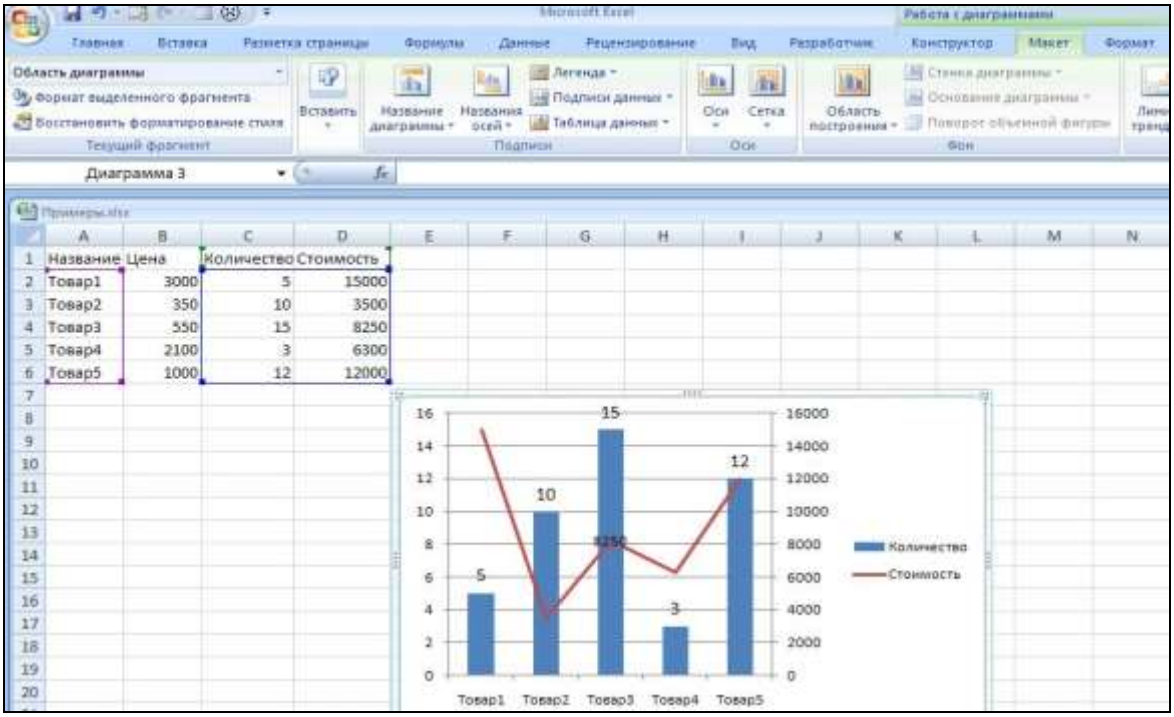


Рис. 6. Внедренная диаграмма

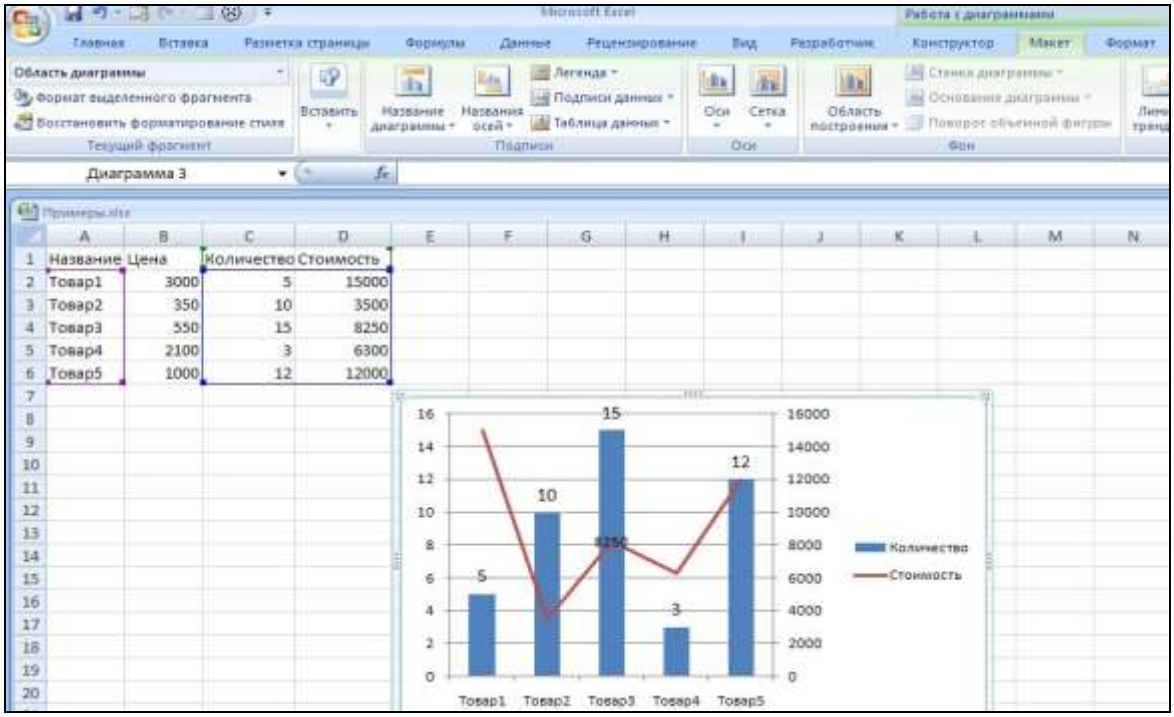


Рис. 6. Внедренная диаграмма

Ряды данных, отображенные в диаграмме, расположены в столбцах таблицы. Для их выделения использованы несмежные области A1:A6 и C1:D6. Диапазон A1:A6 используется для подписей по *оси категорий*.

Поскольку данные, отображаемые в диаграмме, резко разнятся по значениям, применена **вспомогательная ось** для визуализации данных столбца «Стоимость» и изменен **тип диаграммы** (график) для показа данных столбца «Количество».

Изменить расположение диаграммы можно, применив команду **Работа с диаграммами – Конструктор – Расположение – Переместить диаграмму**.

Вычислительные возможности Excel

1.7. Работа с функциями

Встроенные функции позволяют быстро и просто выполнять необходимые вычисления. Excel имеет более трехсот встроенных функций. При необходимости пользователь может создать собственную (пользовательскую) функцию.

Для удобства использования функции сгруппированы по категориям:

- финансовые;
- ссылки и массивы;
- проверка свойств и значений;
- дата и время;
- работа с базой данных;
- определённые пользователем;
- математические;
- текстовые;
- инженерные;
- статистические;
- логические;
- аналитические.

Ввод функций может осуществляться вручную или с помощью **мастера функций**.

В первом случае нужно следить за правильностью соблюдения синтаксиса функции, а именно:

- после имени функции в скобках должны быть записаны необходимые аргументы через точку с запятой.
- при этом необходимо следить за числом и типом аргументов.
- при ручном вводе функций можно использовать автозавершение при нажатии клавиши Tab.

Значительно удобнее использовать мастер функций, который активизируется при использовании команды **Формулы – Библиотека функций** или кнопки **Вставить функцию**.

Первое диалоговое окно мастера функций предназначено для выбора категории и функции из этой категории (рис. 7), а второе – для выбора аргументов (рис. 8).

Аргументы вводятся в поля, слева от которых находятся названия аргумента, а справа – тип аргумента. Названия обязательных аргументов записываются жирным шрифтом.

Внизу приведена информация, поясняющая смысл аргумента и текущее значение вводимой функции.

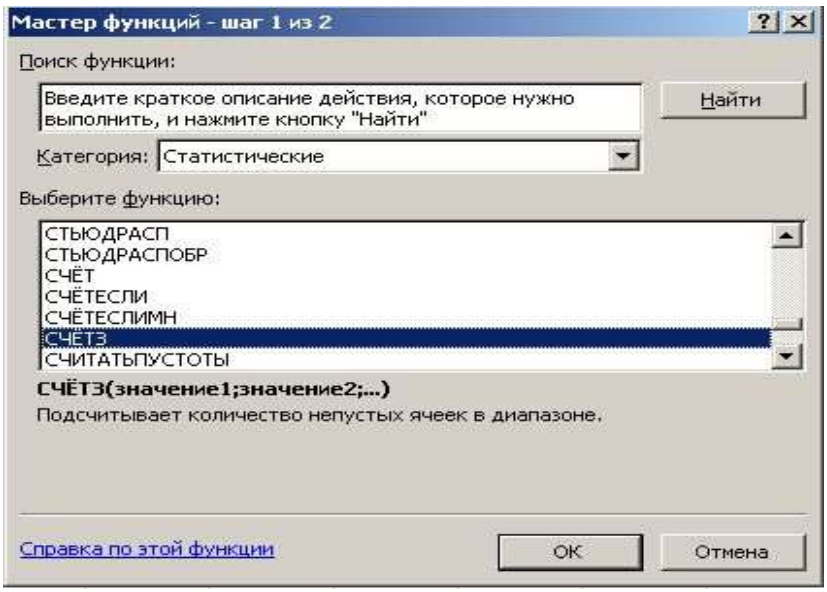


Рис. 7. Выбор категории и функции

На каждом шаге формирования функции можно получить справку по функции с помощью ссылки **Справка по этой функции**.

Формирование функции заканчивается нажатием клавиши **Enter** или **OK**. Для возврата на второй шаг мастера функций после нажатия клавиши **Enter** нужно щелкнуть мышью на имени функции в строке формул.

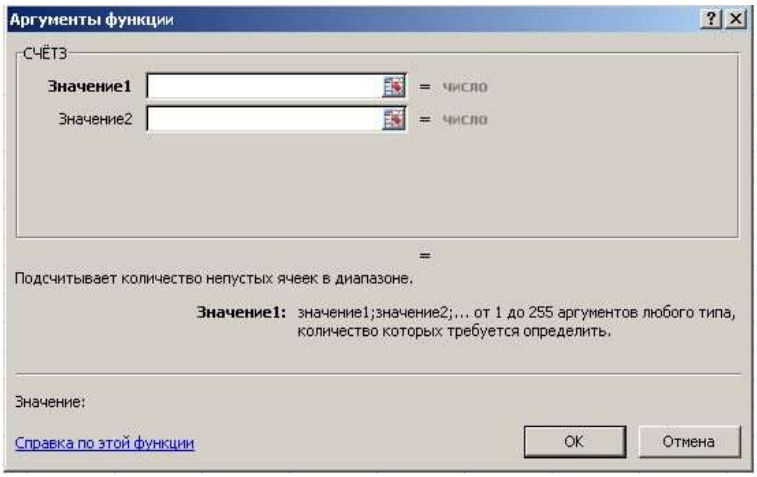


Рис. 8. Выбор аргумента функции

1.7.1. Математические функции

Исходные данные для примеров приведены на рис. 9.

Примеры некоторых математических функций приведены в табл. 1.

Таблица 1

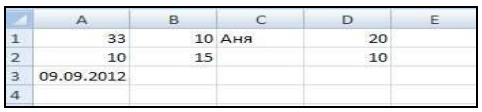
	Функция	Действие	Пример
	ПРОИЗВЕД(арг1; арг2;..)	Возвращает произведение аргументов	ПРОИЗВЕД(A1;A2)→330
	СУММЕСЛИ (диапазон; критерий; диапазон суммирования)	Возвращает сумму значений из заданной области, удовлетворяющих критерию	СУММЕСЛИ(A1:D2;10)→30 СУММЕСЛИ (A1:D2;”>16”)→53 СУММЕСЛИ (A1:A2;10;B1:B2)→15
	ОСТАТ(число;делитель)	Возвращает остаток от деления	ОСТАТ(A2;6)→4
	ОТБР(число;[число разрядов])	Отбрасывает дробную часть числа	ОТБР(12,34)→12
	ОКРУГЛ(число; кол-во цифр)	Округляет число до указанного количества разрядов	ОКРУГЛ(123,456;0)→123 ОКРУГЛ(123,456;1)→123,5 ОКРУГЛ(123,456;-1)→120
	МОПРЕД(диапазон)	Вычисляет определитель матрицы	МОПРЕД(A1:B2)→395
	СЛЧИС()	Генерирует случайные числа в диапазоне от 0 до 1	

	A	B	C	D	E
1	33	10	Аня	20	
2	10	15		10	
3	09.09.2012				
4					

Рис. 9. Исходные данные

1.7.2. Логические функции

Основные логические функции приведены в табл. 2.
Исходные данные для примеров приведены на рис. 9.



	A	B	C	D	E
1	33	10	Аня	20	
2	10	15		10	
3	09.09.2012				
4					

Рис. 9. Исходные данные

Таблица 2

	Функция	Действие	Пример
	И(арг1;арг2;...)	Возвращает значение ИСТИНА, если все аргументы имеют значение ИСТИНА	И(A1>20;B1>5; C1=”Аня”)→ ИСТИНА
	ИЛИ(арг1;арг2; …)	Возвращает значение ИСТИНА, если хотя бы один аргумент имеет значение ИСТИНА	ИЛИ(A1>20;B1<5; C2=”Аня”)→ ИСТИНА
	НЕ(арг)	Меняет логическое значение аргумента на противоположное	НЕ(A1>20)→ЛОЖЬ
	ЕСЛИ(арг1;арг2;арг3)	Возвращает арг2, если арг1 имеет значение ИСТИНА и возвращает арг3, если арг1 имеет значение ЛОЖЬ	ЕСЛИ(A1>20;B1+1;B1–1)→11

1.7.3. Функции категории «Ссылки и массивы»

Примеры функций из этой категории приведены в табл. 3.
Исходные данные для примеров приведены на рис. 9.

Таблица 3

	Функция	Действие	Пример
	ВЫБОР (номер_индекса; значение1; значение2; значение3;…)	Выбирает значение из списка значений, порядковый номер которого совпадает с аргументом номер	ВЫБОР(2;A1;B1)→ B1→10

		индекса	
	ПРОСМОТР (искомое значение; вектор просмотра; вектор результата)	Ищет значение в векторе результата, соответствующее значению вектора просмотра	ПРОСМОТР (A2;B1:B2; D1:D2) →D1→20

1.7.4. Функции даты и времени

Примеры функций из этой категории приведены в табл. 4.
Исходные данные для примеров приведены на рис. 9.

	A	B	C	D	E
1	33	10	Аня	20	
2	10	15		10	
3	09.09.2012				
4					

Рис. 9. Исходные данные

Таблица 4

	Функция	Действие	Пример
	СЕГОДНЯ()	Возвращает сегодняшнюю дату	СЕГОДНЯ()→08.03.2012
	ТДАТА()	Возвращает сегодняшнюю дату и время	ТДАТА()→08.03.2012 20:24
	ДАТА(год;месяц;день)	Возвращает дату	ДАТА (2012;3;25)→25.03.2012
	ДЕНЬНЕД(дата;тип)	Возвращает номердня недели	ДЕНЬНЕД(A3;2)→7
	ГОД(дата)	Возвращает год	ГОД(A3)→2012
	МЕСЯЦ(дата)	Возвращает месяц	МЕСЯЦ(A3)→9
	ДЕНЬ(дата)	Возвращает день	День(A3)→9

1.7.5. Статистические функции

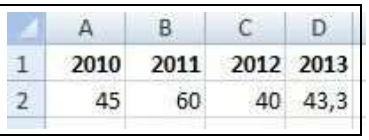
Примеры функций из этой категории приведены в табл. 5. Исходные данные для примеров приведены на рис. 9.

	A	B	C	D	E
1	33	10	Аня	20	
2	10	15		10	
3	09.09.2012				
4					

Рис. 9. Исходные данные

Таблица 5

	Функция	Действие	Пример
	МИН(арг1;арг2;…)	Возвращает минимальное значение в списке аргументов	МИН(A1:B2)→10
	МАКС(арг1;арг2;…)	Возвращает максимальное значение в списке аргументов	МАКС(A1:B2)→33
	СРЗНАЧ(арг1;арг2;…)	Возвращает среднее значение списка аргументов	СРЗНАЧ(A1:B2)→17
	СЧЕТ(арг1;арг2;…)	Возвращает количество чисел в списке аргументов	СЧЕТ(A1:D2)→6
	СЧЕТЗ(арг1;арг2;…)	Возвращает количество непустых значений в списке аргументов	СЧЕТЗ(A1:D2)→7
	СЧЕТЕСЛИ(диапазон;условие)	Возвращает количество чисел в диапазоне, удовлетворяющих условию	СЧЕТЕСЛИ(A1:B2;A2)→2
	РАНГ(число;ссылка;порядок)	Возвращает ранг числа (в диапазоне)	РАНГ(A1;A1:A2)→1
	ПРЕДСКАЗАНИЕ(x; изв_y; изв_x)	Возвращает значение функции в точке x на основе линейной регрессии для известных значений x и y	В ячейку D3 введена функция ПРЕДСКАЗ(D1;A2:C2;A1:C1)→43,3

	A	B	C	D
1	**2010**	**2011**	**2012**	**2013**
2	45	60	40	43,3

1.7.6. Финансовые функции

Примеры функций вычисления параметров ссуды приведены в **табл. 6**.

В функциях используются следующие аргументы:

- ставка – процентная ставка за один период;
- кпер – общее количество выплат (периодов);
- период – заданный период времени, который должен быть меньше или равен значению кпер;
- плт – взнос, который выплачивается каждый период;
- бс – необязательный аргумент, равный будущей стоимости после последней выплаты, если аргумент опущен, он полагается равным нулю;
- пс – приведенная стоимость;
- тип – необязательный аргумент, указывает, когда должна производиться выплата, если выплата производится в конце периода, аргумент равен нулю и может быть опущен, в противном случае он равен единице.

Таблица 6

	Функция	Действие	Пример
	ПЛТ(ставка; кпер; пс; бс; тип)	Возвращает объем выплат по ссуде	Объем ежемесячных выплат по ссуде 55000 р., взятой на 3 года при процентной ставке 6% годовых, вычисляется по формуле: ПЛТ(0,06/12;3*12;-55000) и составит→1 673,21 р.
	КПЕР(ставка; плт; пс; бс; тип)	Возвращает количество выплат по ссуде	Ссуда 55000 р. взята под 6% годовых. Объем ежемесячных выплат по ссуде 1 673,21 р. Количество периодов выплат по ссуде вычисляется по формуле: КПЕР(0,06/12;1673,21;-55000) и составит →36 (месяцев)
	ПС(ставка; кпер; плт; бс; тип)	Возвращает приведенную сумму ссуды	При процентной ставке 6% и ежемесячных выплатах 1 673,21р, объем ссуды, взятой на 36 месяцев вычисляется по формуле: ПС(0,06/12;36;1673,21) и составит→55000 р.
	ОСПЛТ(ставка;	Возвращает	Следующая формула возвращает основную часть выплаты за первый месяц по ссуде 55 000 р., взятой под

	период; кпер; пс; бс; тип)	основную часть выплат по ссуде за определенный период	6% годовых сроком на три года: ОСПЛТ(0,06/12;1;36;-55000)→1 398,21 р.
	ПРПЛТ(ставка; период; кпер; пс; бс; тип)	Возвращает часть выплат по ссуде, которая идет на выплату процентов	Следующая формула возвращает объем выплат по процентам за первый месяц по ссуде 55000 р., взятой под 6% годовых сроком на три года: ПРПЛТ(0,06/12;1;36;-55000)→ 275,00 р.

2. ОБРАБОТКА ДАННЫХ В EXCEL

2.1. Сортировка

Сортировка данных позволяет упорядочить данные в соответствии с выбранным полем таблицы.

В Excel 2007 возможно применить любое количество уровней сортировки.

Для сортировки данных нужно воспользоваться командой **Данные – Сортировка** и **фильтр – Сортировка** (рис. 10).

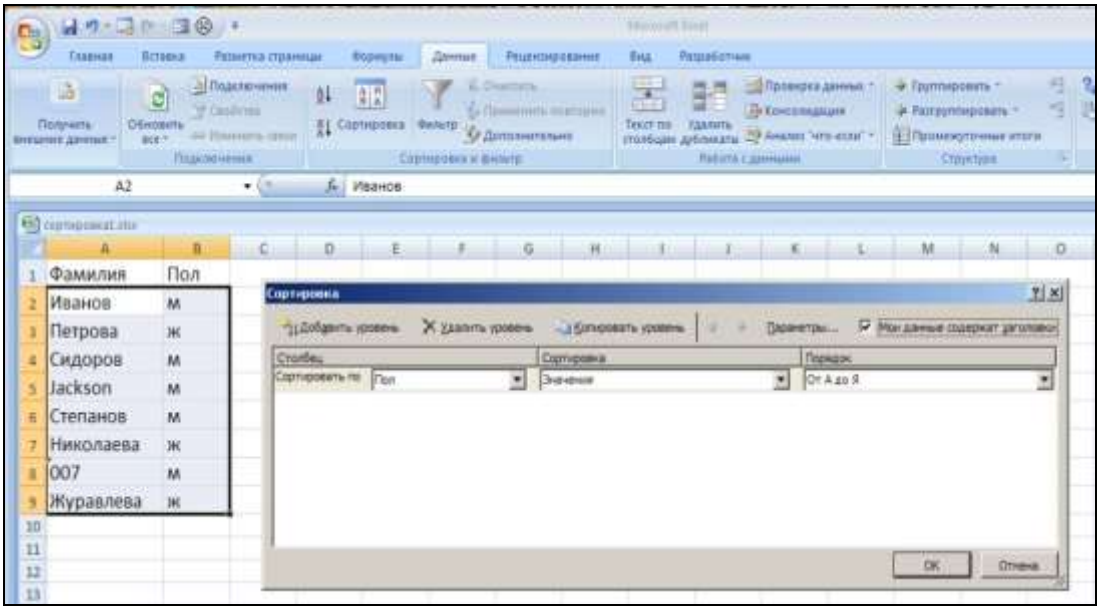


Рис. 10. Использование диалогового окна «Сортировка»

Исходная таблица приведена в диапазоне A1:B9.

Результаты работы команды «Сортировка» представлены на рис. 11.

В диапазоне **D1:E9** приведена таблица, отсортированная по полю **Фамилия по возрастанию** (по алфавиту).

В диапазоне **G1:H9** приведена таблица, отсортированная по полю **Пол по возрастанию.**

В диапазоне **J1:K9** приведена таблица, отсортированная по полю **Пол** по **возрастанию,** а затем по полю **Фамилия по возрастанию**.

	A	B	D	E	G	H	J	K
1	Фамилия	Пол	Фамилия	Пол	Фамилия	Пол	Фамилия	Пол
2	Иванов	м	007	м	Петрова	ж	Журавлева	ж
3	Петрова	ж	Jackson	м	Николаева	ж	Николаева	ж
4	Сидоров	м	Журавлева	ж	Журавлева	ж	Петрова	ж
5	Jackson	м	Иванов	м	Иванов	м	007	м
6	Степанов	м	Николаева	ж	Сидоров	м	Jackson	м
7	Николаева	ж	Петрова	ж	Jackson	м	Иванов	м
8	007	м	Сидоров	м	Степанов	м	Сидоров	м
9	Журавлева	ж	Степанов	м	007	м	Степанов	м

Рис. 11. Результаты работы команды «Сортировка»

Обработка отсортированных данных с помощью команды «Промежуточные итоги» (**Данные – Структура – Промежуточные итоги**) начинается с активизации данной команды (рис. 12).

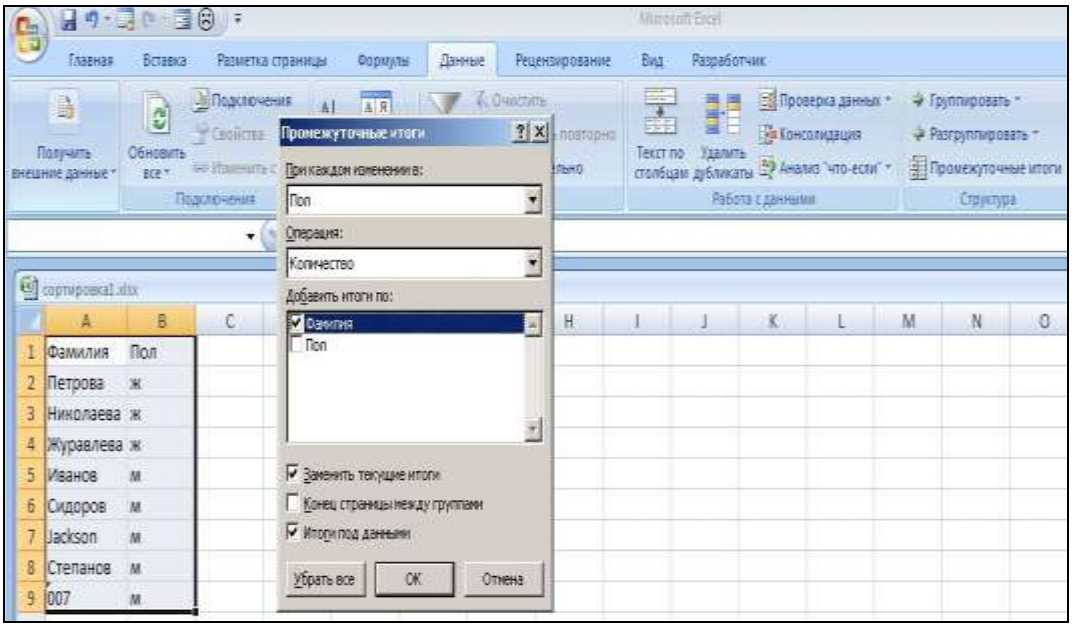


Рис. 12. Использование команды «Промежуточные итоги»

В результате использования команды «Промежуточные итоги» данные структурируются и обрабатываются с помощью выбранной функции (рис. 13).

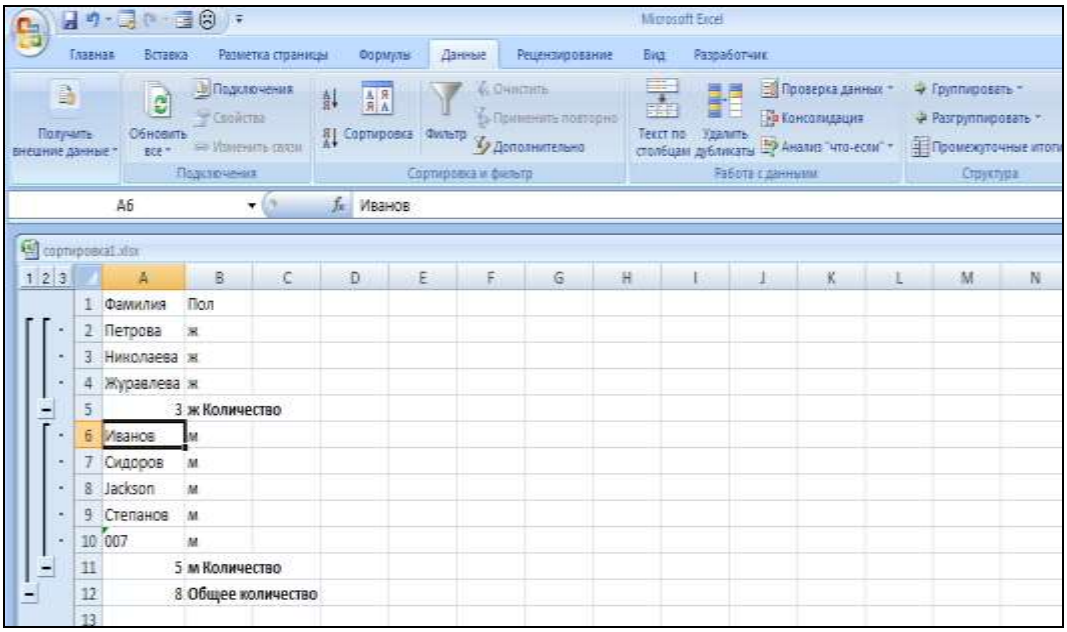


Рис. 13. Результат работы команды «Промежуточные итоги»

Используя управление уровнями структуры, можно получить итоги вычислений по отсортированным группам (по полу), скрыв сами значения (рис. 14) или результирующие вычисления, оставив только верхний уровень структуры (1).

Рис. 14. Результат сокрытия промежуточных данных

2.2. Фильтрация данных

2.2.1. Автофильтр

Фильтрация позволяет скрыть данные, не удовлетворяющие заданным критериям, и отобразить только те данные, которые критериям удовлетворяют.

Фильтры различаются по способам создания критериев.

В **автофильтре** критерии заданы разработчиками и пользователь должен только выбрать необходимый для решения конкретной задачи, при использовании **расширенного фильтра** критерии создаются самим пользователем.

Автофильтр активизируется выбором команды **Данные – Сортировка** и **фильтрация – Фильтр**.

Диалоговые окна со списком критериев представлены на рис. 15.

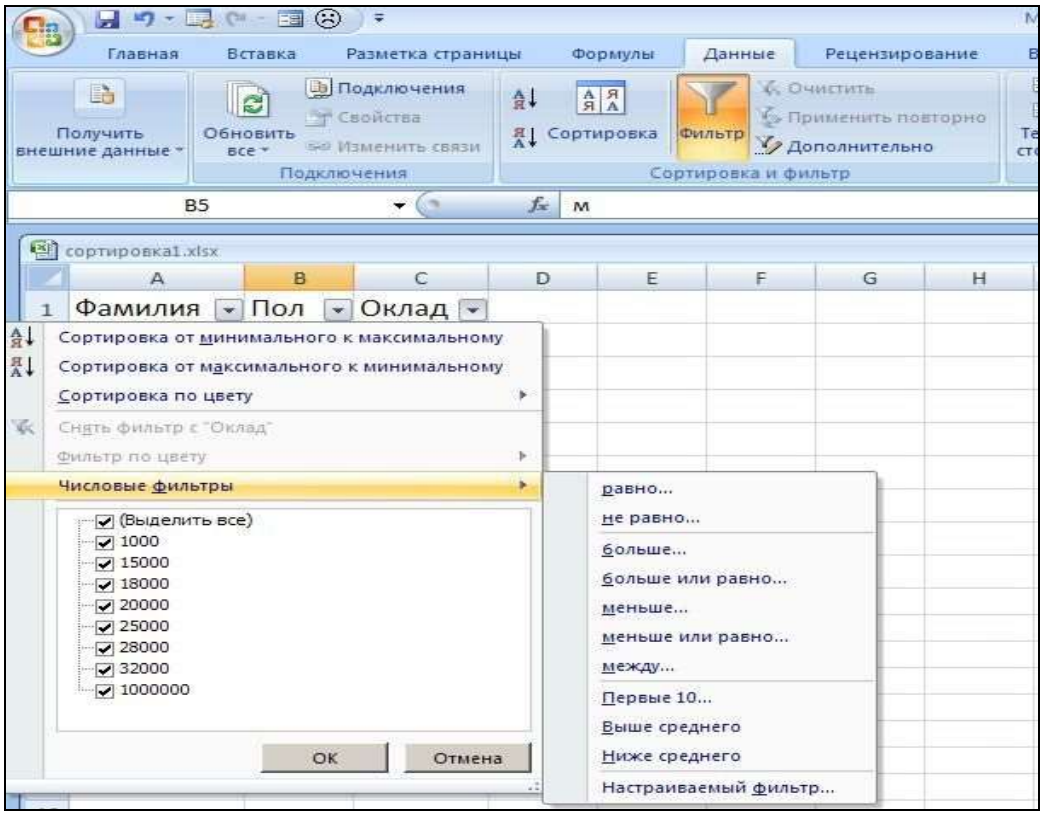


Рис. 15. Активизация фильтра

Настраиваемый фильтр позволяет создавать критерии для одного поля, связанные логическими операциями И и ИЛИ.

Пример приведен на рис. 16.

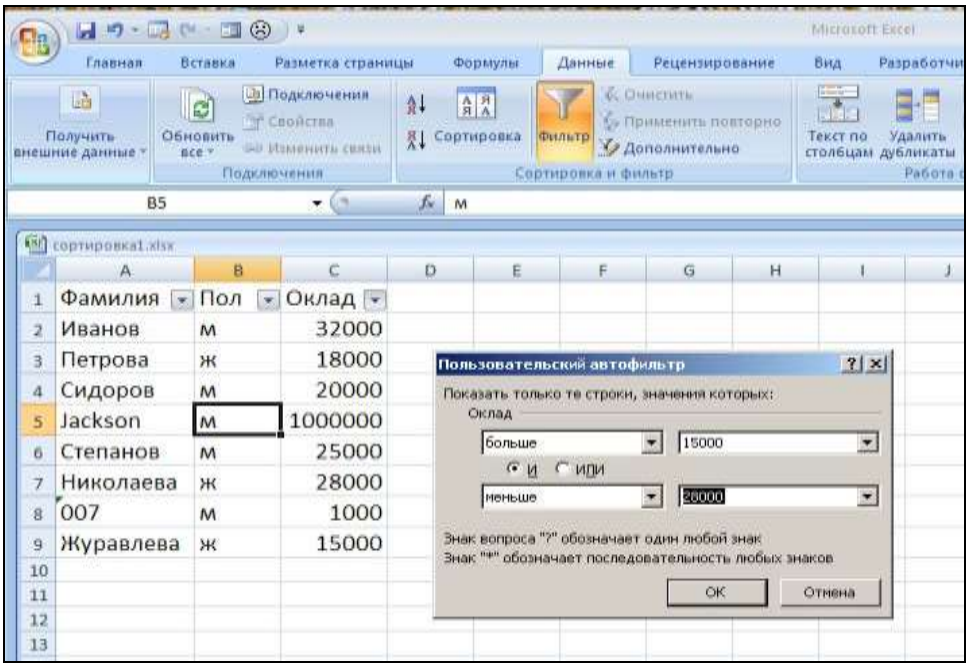


Рис. 16. Пример использования настраиваемого фильтра

Для обработки отфильтрованных данных используется функция ПРОМЕЖУТОЧНЫЕ.ИТОГИ(код;диапазон). Значения кодов можно посмотреть в справке (1 – СРЗНАЧ, 2 – СЧЕТ, 3 – СЧЕТЗ и т. д.).

2.2.2. Расширенный фильтр

Использование расширенного фильтра обусловлено тем, что он позволяет создавать более сложные критерии, а именно, связанные логическими операциями И и ИЛИ для одного и нескольких полей.

Активизация расширенного фильтра происходит выбором команды **Данные – Сортировка** и **фильтрация – Дополнительно** (рис. 17).

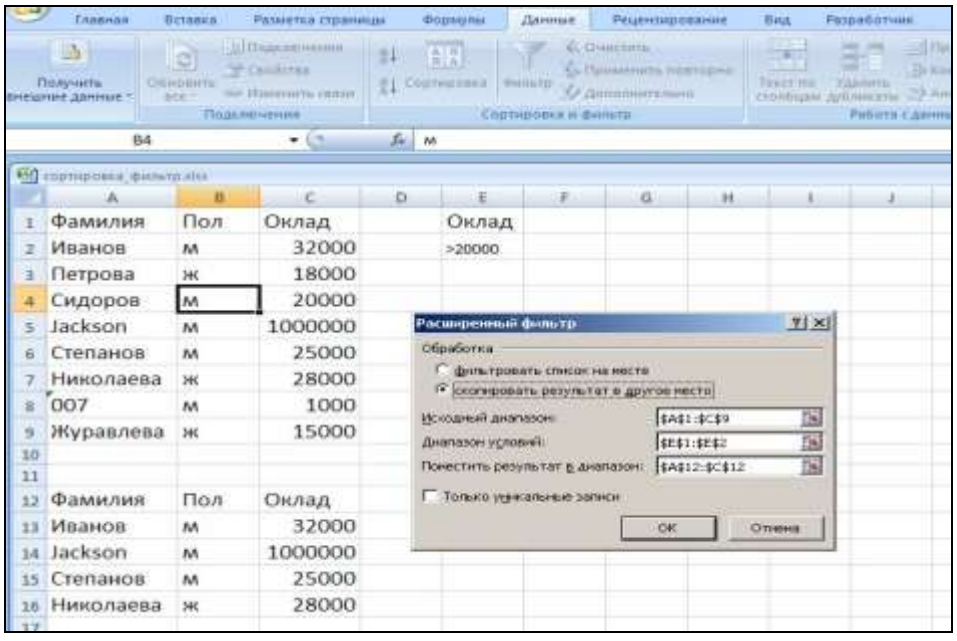
Для функционирования расширенного фильтра необходимо создать критерий для фильтрации и поместить его в табличную область, не соприкасающуюся с исходным диапазоном, и диапазоном, в который помещается результат.

Область критериев должна содержать не менее чем две строки. В первой строке указывается название поля, по которому происходит фильтрация, а во второй и следующих указываются значения критерия.

В примере, приведенном на **рис. 18**, область критериев находится в диапазоне E1:E2. В E1 записано название поля – Оклад, а в E2 – значение >20000.

Таким образом, из всех данных будут отфильтрованы только те записи, у которых значение поля Оклад >20000.

Область критериев должна быть указана в диалоговом окне «Расширенный фильтр» в поле «Диапазон условий».



	A	B	C	D	E
1	Фамилия	Пол	Оклад		Оклад
2	Иванов	м	32000		>20000
3	Петрова	ж	18000		
4	Сидоров	м	20000		
5	Jackson	м	1000000		
6	Степанов	м	25000		
7	Николаева	ж	28000		
8	007	м	1000		
9	Журавлева	ж	15000		
10					
11					
12	Фамилия	Пол	Оклад		
13	Иванов	м	32000		
14	Jackson	м	1000000		
15	Степанов	м	25000		
16	Николаева	ж	28000		

Рис. 17. Активизация расширенного фильтра

Результаты фильтрации могут быть помещены в отдельный диапазон, для чего он должен быть указан в диалоговом окне «Расширенный фильтр» в поле «Поместить результат в диапазон». В этом же окне предварительно должен быть установлен переключатель «Скопировать результат в другое место».

Результаты фильтрации помещены в примере в диапазон A12:C12. Выходной диапазон определяется только одной пустой верхней левой ячейкой в том случае, если он должен содержать все поля исходной таблицы. Для получения результатов фильтрации только в отдельных полях таблицы, их названия нужно поместить в выходной диапазон и указать его в диалоговом окне «Расширенный фильтр» в поле «Поместить результат в диапазон».

Значения критериев, связанных логической операцией И, должны располагаться в одной строке, а связанные логической функцией ИЛИ – в разных.

На рис. 18 приведены примеры таких критериев.

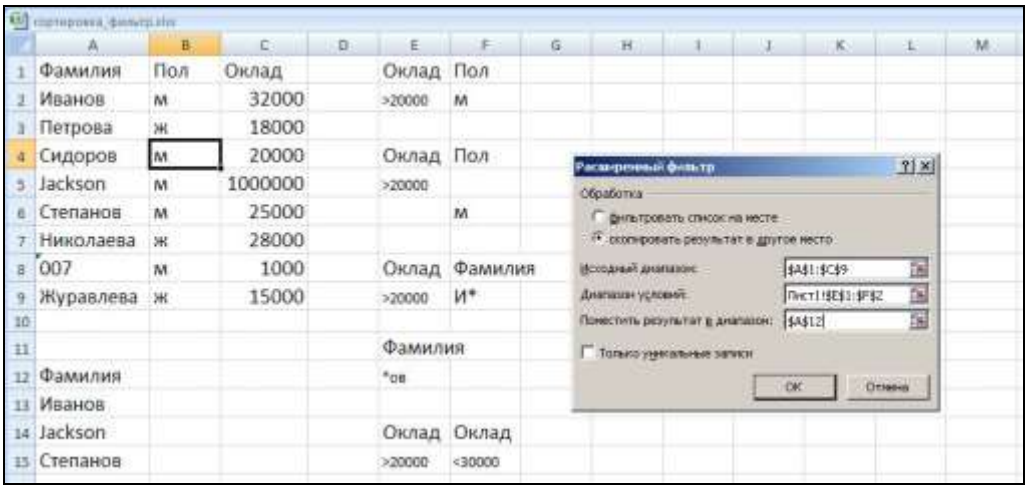


Рис. 18. Примеры использования критериев

При использовании критерия, расположенного в диапазоне E1:F2, выбираются записи, в которых значение поля Оклад >20000, а поля Пол – м, т. е. выбираются мужчины с окладом >20000.

В качестве выходного диапазона указана ячейка A12, в которую помещено название поля Фамилия.

Для последующих примеров в качестве выходного диапазона взят диапазон A12:B12, в который помещены названия полей Фамилия и Оклад.

Критерий, расположенный в диапазоне E4:F6, позволяет найти записи, в которых в поле Оклад значение >20000, или те, в которых поле Пол находится значение «м». Результат фильтрации:

Фамилия	Оклад
Иванов	32000
Сидоров	20000
Jackson	1E+06
Степанов	25000

Ник олаева	2 8000
007	1 000

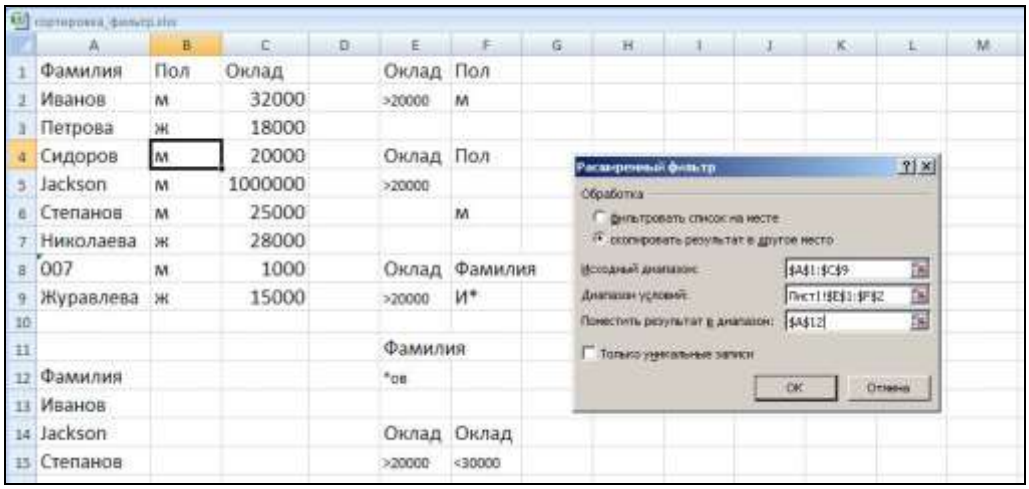


Рис. 18. Примеры использования критериев

Критерий, расположенный в диапазоне E8:F9, позволяет найти записи, в которых в поле Оклад значение >20000 и в поле Фамилия значения начинаются с буквы И. Результат фильтрации:

Фамилия	Оклад
Иванов	32000

Критерий, расположенный в диапазоне E11:E12, позволяет найти записи, в которых в поле Фамилия есть сочетание символов «ов». Результат фильтрации:

Фамилия	Оклад
Иванов	32000
Петрова	18000
Сидоров	20000
Степанов	25000

Критерий, расположенный в диапазоне E14:F15, позволяет найти записи, в которых в поле Оклад значение >20000 и <30000. Результат фильтрации:

Фамилия	Оклад

Степанов	5000	2
Николаева	8000	2

2.3. Связывание

Связывание позволяет сберечь память и время на обновление вычислений.

Связывание может быть осуществлено непосредственно с помощью формул, например:

обращение к ячейке другого листа: =A5*Лист1!C2

обращение к ячейке другой книги: =A5*[Книга1.xlsx]Лист1!C2

обращение к ячейке закрытой книги: =A5*’C:\Мои документы\[Книга1.xlsx]Лист 1’!C2

Связывание может быть осуществлено с помощью команды

Главная – Буфер обмена – Вставить – Специальная вставка – Вставить связь.

2.4. Консолидация

Консолидация позволяет объединить данные из исходных листов в итоговом листе.

Консолидация может быть динамической или статической.

При **динамической** консолидации сохраняется связь с исходными листами и изменения, происходящие в исходных листах, будут автоматически отображаться в итоговом листе.

При **статической** консолидации связь с исходными диапазонами не сохраняется.

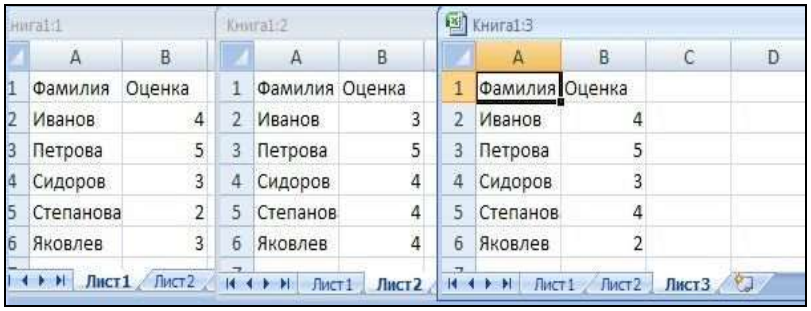
Консолидация данных может производиться **по расположению** или **по категории.**

При консолидации по расположению объединяются данные, находящиеся в одинаково расположенных ячейках.

Консолидация по категории предусматривает использование в качестве основы для объединения листов заголовки строк и столбцов.

Консолидация данных происходит с помощью команды **Данные – Работа с данными – Консолидация.**

Исходные листы с данными для консолидации по расположению представлены на рис. 19.



Книга1:1

	A	B
1	Фамилия	Оценка
2	Иванов	4
3	Петрова	5
4	Сидоров	3
5	Степанова	2
6	Яковлев	3

Лист1 Лист2

Книга1:2

	A	B
1	Фамилия	Оценка
2	Иванов	3
3	Петрова	5
4	Сидоров	4
5	Степанов	4
6	Яковлев	4

Лист1 Лист2

Книга1:3

	A	B	C	D
1	Фамилия	Оценка		
2	Иванов	4		
3	Петрова	5		
4	Сидоров	3		
5	Степанов	4		
6	Яковлев	2		

Лист1 Лист2 Лист3

Рис. 19. Исходные листы с данными для консолидации

Консолидируемые данные имеют одинаковое расположение во всех исходных листах.

На итоговом листе требуется найти **среднюю оценку** для каждого студента.

На этом листе нужно внести заголовки столбцов «Фамилия» и «Оценка» и установить курсор в ячейку А2, затем активизировать команду «Консолидация».

Диалоговое окно «Консолидация» и его использование представлено на **рис. 20.**

Для **осуществления консолидации** необходимо **выбрать функцию,** которая будет использована для консолидированных данных и указать **диапазоны данных** на исходных листах, подлежащих консолидации.

Динамическая консолидация происходит при установке флажка «Создавать связи с исходными диапазонами».

В этом случае образуется структура, уровни которой показаны слева от табличной области.

Раскрыв ее, можно увидеть значения консолидируемых данных в исходных листах и проследить за динамикой изменения исходных и результирующих данных.

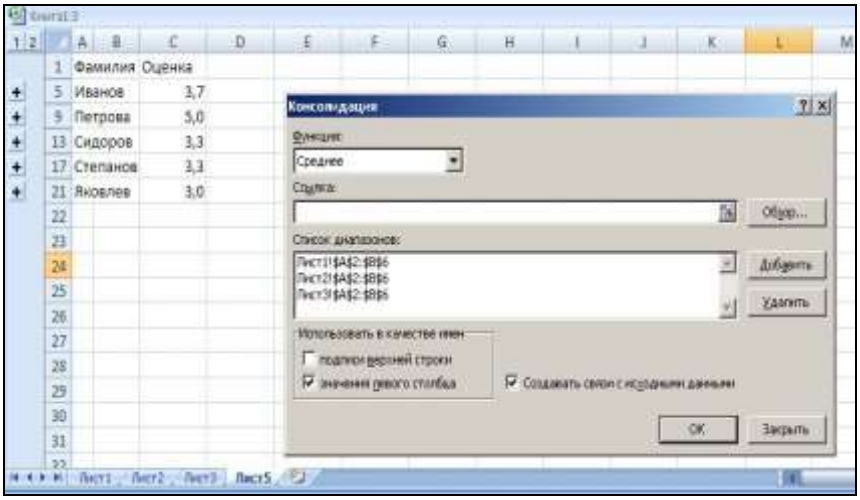


Рис. 20. Диалоговое окно «Консолидация»

Консолидация по **категории** используется, если исходные данные не расположены в одинаковых ячейках.

Пример приведен на рис. 21.

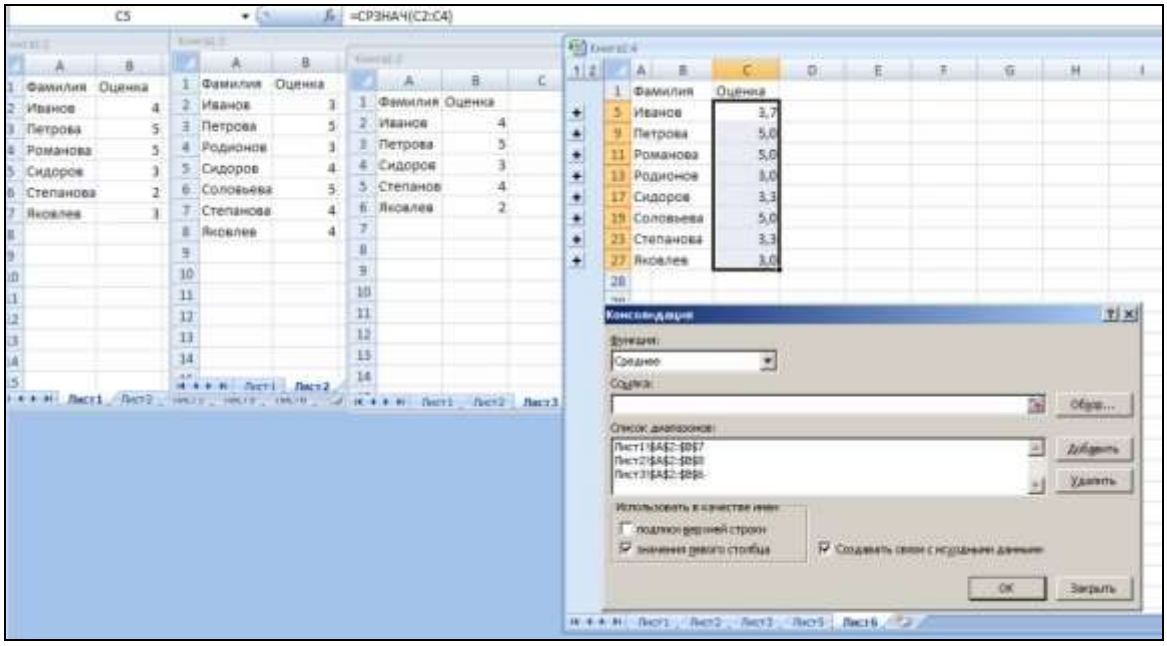


Рис. 21. Консолидация по категории

2.5. Сводные таблицы

Сводные таблицы представляют собой интегральный инструмент обработки, который включает в себя вычисление функций, использование консолидации, преобразование структуры таблицы, фильтрацию и другие операции.

ФИРМА	МАРКА	ГОД ВЫПУСКА	ЦЕНА	ДАТА ПРОДАЖИ
Альфа	Опель	2000	12000	15.01.2009
Альфа	Опель	1998	10000	20.02.2009
Альфа	Опель	1995	9000	22.02.2009
Альфа	Вольво	2003	15000	03.03.2009
Бета	Вольво	2001	14000	26.03.2009
Бета	Вольво	1999	11000	18.04.2009
Бета	Опель	2005	13000	23.05.2009
Бета	Опель	2001	12000	25.05.2009
Гамма	Вольво	1998	11000	30.05.2009
Гамма	Опель	2006	15000	03.06.2009
Гамма	Опель	2002	13000	05.06.2009
Гамма	Вольво	1997	10000	05.06.2009

Рис. 22. Исходная таблица

Для создания сводной таблицы нужно использовать команду

Вставка – Таблицы – Сводная таблица.

Сводная таблица позволяет конструировать из исходной таблицы новую таблицу с произвольно расположенными строками и столбцами, применять стандартные функции для обработки данных таблицы или использовать вычисления, необходимые пользователю.

Возможность фильтрации, группировки данных, построения диаграмм и другие возможности делают механизм сводных таблиц одним из самых эффективных механизмов обработки данных.

Сводная таблица строится из исходной (рис. 22) посредством заполнения панели «Список полей сводной таблицы», которая определяет макет сводной таблицы.

На рис. 23 представлена панель «Список полей сводной таблицы» и сводная таблица, построенная на его основе.

При создании сводной таблицы появляются вкладка «Работа со сводными таблицами – Параметры» и «Работа со сводными таблицами – Конструктор».

На вкладке «Работа со сводными таблицами – Параметры» можно изменять активное поле, проводить групповые операции, а также управлять показом заголовков полей и другими элементами сводной таблицы.

Команда **Активное поле – Параметры поля – Операция** позволяет изменять функцию, которая применяется в сводной таблице, а также имя поля.

На вкладке «Работа со сводными таблицами – Конструктор» можно выбрать стиль представления сводной таблицы, а также отображение промежуточных и итоговых вычислений.

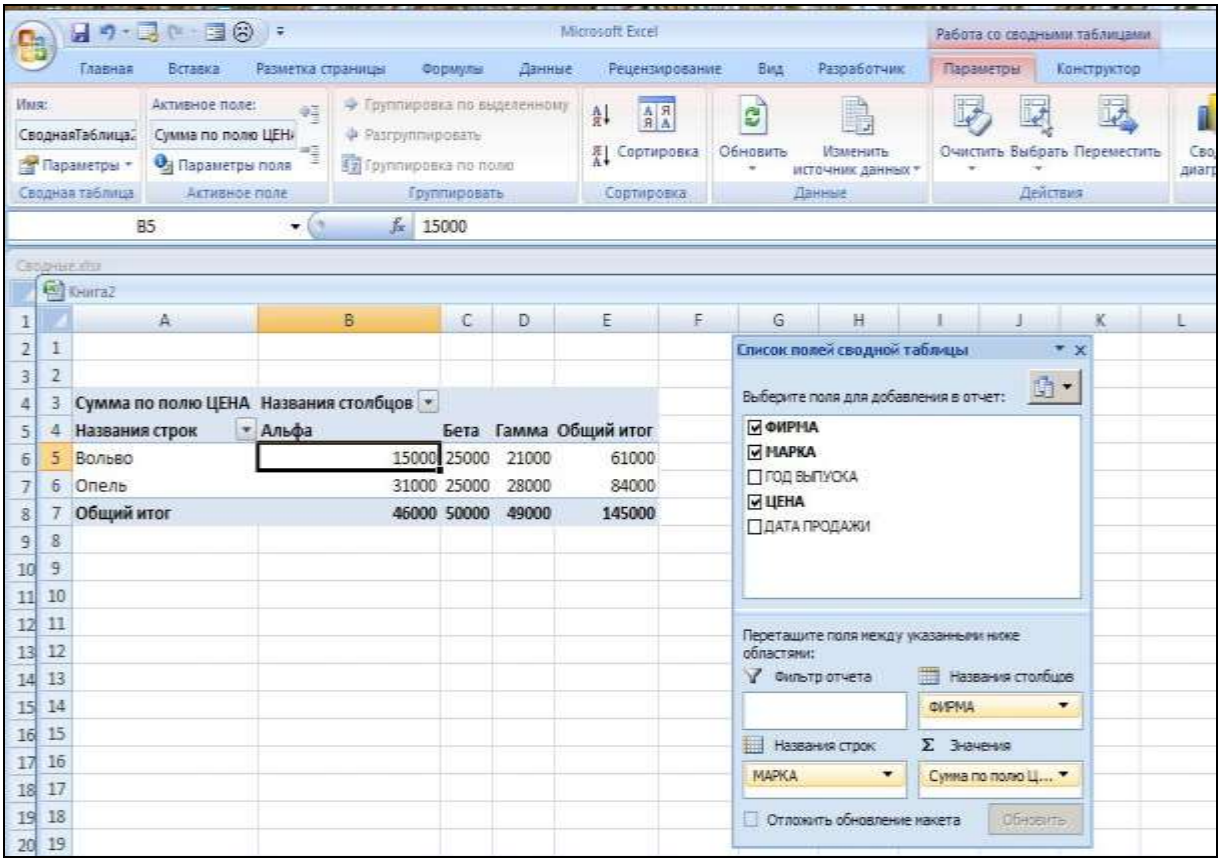


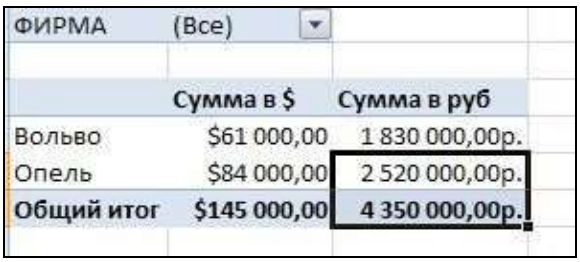
Рис. 23. Макет и сводная таблица

Для создания вычисляемого поля нужно воспользоваться командой **Работа со сводными таблицами – Параметры – Сервис – Формулы – Вычисляемое поле.**

В диалоговом окне нужно дать полю имя и ввести формулу для вычислений.

Пример приведен на **рис. 24**.

В примере использована формула =30*Цена.



ФИРМА	(Все)	
	Сумма в $	Сумма в руб
Вольво	$61 000,00	1 830 000,00р.
Опель	$84 000,00	2 520 000,00р.
Общий итог	**$145 000,00**	**4 350 000,00р.**

Рис. 24. Пример вычисляемого поля

ФИРМА	МАРКА	ГОД ВЫПУСКА	ЦЕНА	ДАТА ПРОДАЖИ
Альфа	Опель	2000	12000	15.01.2009
Альфа	Опель	1998	10000	20.02.2009
Альфа	Опель	1995	9000	22.02.2009
Альфа	Вольво	2003	15000	03.03.2009
Бета	Вольво	2001	14000	26.03.2009
Бета	Вольво	1999	11000	18.04.2009
Бета	Опель	2005	13000	23.05.2009
Бета	Опель	2001	12000	25.05.2009
Гамма	Вольво	1998	11000	30.05.2009
Гамма	Опель	2006	15000	03.06.2009
Гамма	Опель	2002	13000	05.06.2009
Гамма	Вольво	1997	10000	05.06.2009

Рис. 22. Исходная таблица

Для группировки данных из таблицы (рис. 22) нужно установить курсор на ячейку с датами (A5) и воспользоваться командой **Работа со сводными таблицами – Параметры – Группировать**.

Результаты группировки по полю представлены на рис. 25, а результаты группировки по выделенному на рис. 26.

	A	B	C	D
1	ФИРМА	(Все)		
2				
3	Сумма			
4		Вольво	Опель	Общий итог
5	⊟Кв-л1			
6	янв		12000	12000
7	фев		19000	19000
8	мар	29000		29000
9	⊟Кв-л2			
10	апр	11000		11000
11	май	11000	25000	36000
12	июн	10000	28000	38000
13	Общий итог	61000	84000	145000

Рис. 25. Группировка по полюДата продаж

	A	B	C	D
1	ФИРМА	(Все)		
2				
3	Сумма			
4		Вольво	Опель	Общий итог
5	⊟зима-лето			
6	15.01.2009		12000	12000
7	20.02.2009		10000	10000
8	22.02.2009		9000	9000
9	03.06.2009		15000	15000
10	05.06.2009	10000	13000	23000
11	⊟весна			
12	03.03.2009	15000		15000
13	26.03.2009	14000		14000
14	18.04.2009	11000		11000
15	23.05.2009		13000	13000
16	25.05.2009		12000	12000
17	30.05.2009	11000		11000
18	Общий итог	61000	84000	145000

Рис. 26. Группировка по выделенному

3. ИНТЕРАКТИВНАЯ РАБОТА С ДАННЫМИ

3.1. Работа с макросами

Excel имеет возможности автоматизации вычислений, создания приложений и их интерфейсов, а также обеспечения взаимодействия с другими компонентами MS Office.

Эти возможности обусловлены применением программирования. Программирование в Excel основано на использовании языка VBA (VisualBasicforApplications), который является языком визуального проектирования и на объектной модели Excel.

Файл Excel, содержащий программный код, должен иметь расширение .xlsm, т. е. сохраняться как «Книга Excel с поддержкой макросов».

Основной программной единицей VBA является процедура.

Процедуры разделяются на процедуры-подпрограммы и процедуры-функции.

Макросы относятся к процедурам-подпрограммам и чаще всего создаются в автоматическом режиме, однако могут быть созданы и вручную. Макросы хранятся в контейнерном элементе, называемом модулем. При

создании макроса в автоматическом режиме модуль создается автоматически.

Для выполнения макроса достаточно вызвать его по имени. Имя макроса не должно содержать пробелов и должно начинаться с буквы. Чтобы макрос автоматически выполнялся при открытии рабочей книги, ему нужно дать имя Auto_Open, а при закрытии – Auto_Close.

Программный код макроса, как любой процедуры-подпрограммы, содержит код, ограниченный ключевыми словами Sub и EndSub:

SubимяМакроса()
Код макроса
EndSub

Программный код функции содержит код функции, ограниченный ключевыми словами Function и EndFunction:

FunctionимяФункции(арг1, арг2,…)
Код функции
EndFunction

Функции, в отличие от макросов, не производят каких-либо действий с помощью команд Excel и не изменяют рабочее пространство, поэтому создаются только вручную и их функционирование заключается в преобразовании аргументов. Функции относятся к категории «Пользовательских» и их вызов ничем не отличается от вызова функций других категорий.

Для создания макроса нужно воспользоваться командой **Разработчик– Код – Запись макроса**, дать макросу имя, указать сочетание клавиш для его запуска и объект Excel, выбранный для его хранения, затем выполнить действия, которые должны быть записаны в макросе, и остановить запись с помощью команды **Разработчик – Код – Остановить запись**. Выполнить макрос можно с помощью команды **Разработчик – Код – Макросы – Выполнить** или воспользоваться выбранным сочетанием клавиш.

Увидеть содержимое макроса можно с помощью редактора VBA. Его можно вызвать командой **Разработчик – Код – VisualBasic** или **Разработчик – Код – Макросы – Изменить** или сочетанием клавиш Alt+F11. Окно редактора VBA показано на рис. 27.

В верхнем левом углу окна редактора находится окно проектов (ProjectExplorer), отображающее иерархическую структуру открытых в данный момент проектов VBA. Проект VBA содержит совокупность программных кодов и объектов VBA, обеспечивающих работу данного проекта и сохраненных в одной рабочей книге.

Компоненты, содержащие коды, представлены в виде иерархии папок – MicrosoftExcelObjects. К ним относятся листы книги Excel модули (Modules) и формы (Forms), создаваемые пользователем.

Стандартный модуль с кодом макроса находится справа от окна проектов. При необходимости могут быть созданы новые модули с помощью команды VBA **Insert – Module**. В модулях могут располагаться не только коды макросов, но и другие коды по усмотрению разработчика.

Место расположения кодов не принципиально, поскольку можно обращаться к кодам независимо от того, где они находятся.

Под окном проектов находится окно свойств (Properties), в котором отображаются свойства выделенного объекта.

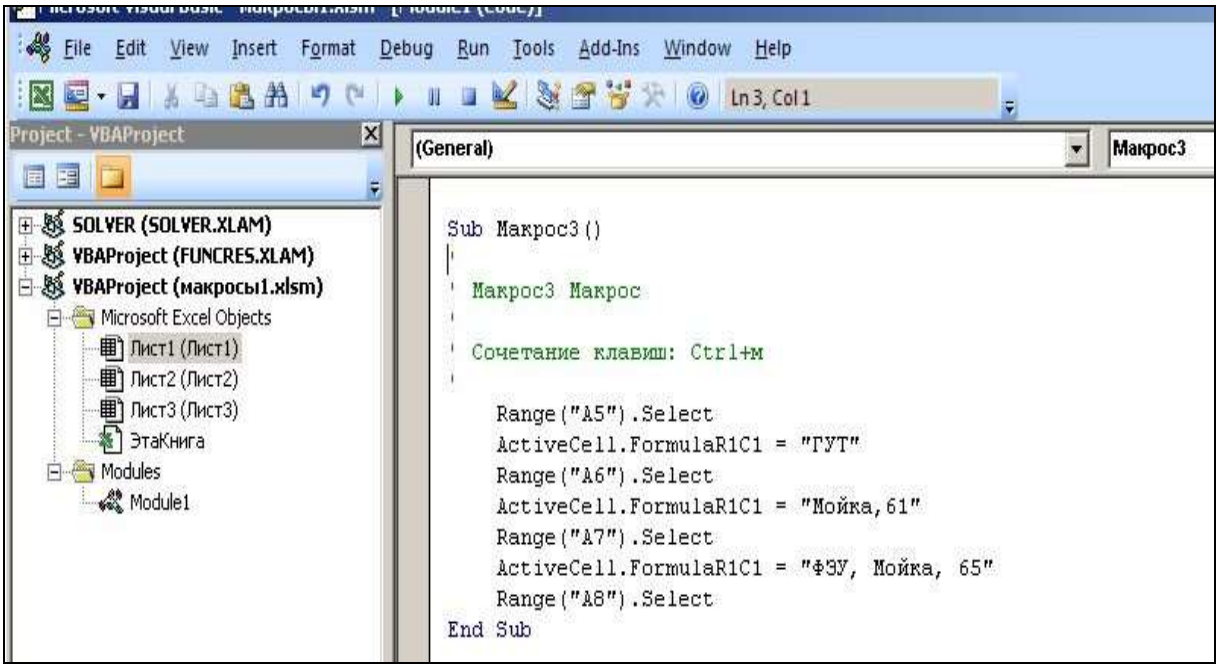


Рис. 27. Окно редактора VBA

1. БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Современные информационные системы (ИС), основанные на концепции банков данных и баз знаний, характеризуются большими объемами хранимой информации, сложной организацией, необходимостью удовлетворять разнообразные требования пользователей. Важным компонентом этой концепции является единая методология проектирования БД. БД, являясь информационной моделью непрерывно меняющегося реального мира, также должны меняться, чтобы адекватно отображать действительность. Поэтому для сопровождения и эксплуатации информационных систем требуется постоянное использование процедур проектирования БД.

Методология проектирования БД может рассматриваться как совокупность методов и средств, последовательное применение которых обеспечивает разработку проекта БД, удовлетворяющего заданным целям. Рассматриваемая методология позволяет пользователю лучше понять, как следует специфицировать требования к данным.

1.1. Основные понятия баз данных

Основным назначением ИС является оперативное обеспечение пользователя информацией о внешнем мире. Однако пользователя интересует не весь окружающий мир, а некоторый его фрагмент, который и находит свое воплощение в автоматизированной ИС.

Предметная область – отражение в ИС совокупности и объектов реального мира с их связями, относящимися к некоторой области знаний и имеющими практическую ценность для пользователя.

Понятие предметной области было введено в начале 80-х гг. прошлого века, когда учеными в области ИС была осознана необходимость использовать семантические модели для представления информации в компьютерных системах. Так же как требования к компьютерной системе формируются средствами естественного языка, так и информация в компьютерных системах представляется средствами особого языка с определенной семантикой. Такой подход впервые был представлен П. Ченом в 1976 г.

Исходя из понятия предметной области, перейдем к понятию БД. **База данных** объединяет в себе информационное представление объектов предметной области, связей между ними и операций над ними, образуя тем самым информационную и функциональную модели предметной области и описывая ее состояние с определенной точностью.

Информация о внешнем мире представляется в любой ИС, в том числе и в БД в форме данных. Это ограничивает возможности смысловой интерпретации информации и конкретизирует семантику ее представления в системе.

Попытаемся дать определение БД исходя из понятия «данные» и «информация».

Термин информация происходит от латинского слова informatio, что означает «сведения, разъяснения, изложение». **Информация** – любые сведения, не известные ранее получателю информации, пополняющие его знания.

Одно и то же информационное сообщение может содержать разное количество информации для разных людей в зависимости от накопленных ими знаний, от уровня понимания этого сообщения и интереса к нему. Так, сообщение, составленное на японском языке, не несет никакой новой информации человеку, не знающему этого языка, но может быть высокоинформативным для человека, владеющего японским. Никакой новой информации не содержит и сообщение, изложенное на знакомом языке, если его содержание непонятно или уже известно.

Информация может существовать и передаваться в виде:

- текстов, рисунков, чертежей, фотографий;
- световых или звуковых сигналов;
- радиоволн;
- электрических и нервных импульсов;
- магнитных записей;
- жестов и мимики;
- запахов и вкусовых ощущений;
- хромосом и т. д.

Одним из важнейших видов информации является **экономическая информация**. Экономическая информация сопровождает процессы производства, распределения, обмена и потребления материальных благ и услуг. Экономическая информация – совокупность сведений, отражающих социально-экономические процессы и служащих для управления этими процессами и коллективами людей в производственной и непроизводственной сфере.

Информация есть характеристика не сообщения, а соотношения между сообщением и его потребителем. Без наличия потребителя, хотя бы потенциального, говорить об информации бессмысленно. Информация извлекается получателем из соответствующих данных.

Данные – информация о каком-либо событии, процессе или объекте, зафиксированная в определенной форме, пригодной для восприятия, передачи, преобразования, хранения или использования.

Так, написанные на листе 10 номеров телефонов в виде последовательности десяти чисел представляют собой данные, но не несут в себе информации. В то же время если напротив каждого номера указать фамилию человека и его адрес, то цифры обретут определенность и превратятся из данных в информацию.

Традиционно фиксация данных осуществляется при помощи конкретного средства общения (например, при помощи естественного языка или изображений) на конкретном носителе (например, бумаге или дискете). Обычно данные и их интерпретация (семантика) фиксируются совместно.

Примером может служить утверждение «Оценка студента – 5». Здесь «5» – данное, а «Оценка студента» – его семантика.

В окружающем мире информация может быть представлена в виде данных разного типа: числовых, текстовых, графических, аудио, видео и т. п. Именно данные и являются предметом хранения в БД.

Первоначально (начало 60-х гг.) использовалась файловая система хранения данных. Для решения преимущественно инженерных задач, характеризующихся небольшим количеством данных и значительным объемом вычислений, данные хранились непосредственно в программе. Применялся последовательный способ организации данных с их высокой избыточностью, идентичностью логической и физической структур и полной зависимостью. С появлением экономико-управленческих задач, отличающихся большими объемами данных и малой долей вычислений, указанная организация данных оказалась неэффективной. Требовалось их упорядочение, которое, как выяснилось, возможно было проводить по двум критериям: использование (информационные массивы) и хранение (базы данных). Первоначально применяли информационные массивы, но вскоре выяснилось превосходство БД. Использование файлов для хранения только данных было предложено Мак Гри в 1959 г. Были разработаны методы доступа (в том числе произвольного) к таким файлам, при этом физическая и логическая структуры уже различались, а физическое расположение данных можно было менять без изменения логического представления.

В 1963 г. С. Бахманом была построена первая промышленная БД, при этом доступ к данным осуществлялся при помощи соответствующего программного обеспечения (ПО).

Таким образом, под **базой данных** понимают совокупность хранящихся вместе данных о конкретной предметной области при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Целью создания БД как разновидности информационной технологии и формы хранения данных является построение системы данных, не зависящих от принятых алгоритмов и ПО, применяемых технических средств и физического расположения данных в ЭВМ, которые обеспечивают непротиворечивую и целостную информацию при нерегламентируемых запросах.

БД предполагает многоцелевое ее использование (несколько пользователей, множество форм документов и запросов одного пользователя). Основные функции базы данных: сбор, хранение, поиск, извлечение и представление данных.

1.2. Модели и виды баз данных

Модель данных – множество элементов (объектов, типов данных) и связей между ними, ограничений (например, целостности, синхронизации многопользовательского доступа, авторизации) операций над типами данных и отношениями.

Модель данных определяет логическая структуру данных, которая представляет присущие этим данным свойства, не зависимые от аппаратного и программного обеспечения и не связанные с функционированием компьютера.

Модель и структура БД зависит от выбранной модели представления данных. Для каждой модели данных создан специальный класс программных продуктов, позволяющие создавать и управлять БД. Этот класс ПО получил название **системы управления базами данных (СУБД).** Основная особенность СУБД – это наличие процедур для ввода и хранения не только самих данных, но и описаний их структуры.

Можно выделить следующие модели данных: иерархическая, сетевая, реляционная, объектно-ориентированная, многомерная.

Иерархическая модель

Иерархическая модель была исторически первой структурой БД, видимо, в связи с тем, что древовидные иерархические структуры широко используются в повседневной человеческой деятельности. Это всевозможные классификаторы, ускоряющие поиск требуемой информации, иерархические функциональные структуры управления.

Иерархическая модель состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию. Такие БД могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй – объекты второго уровня и т. д.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня). При этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок.

Например, если иерархическая база данных содержала информацию о покупателях и их заказах, то будут существовать объект «покупатель» (родитель) и объект «заказ» (дочерний).

Достоинством такой модели является скорость поиска. Так запрос, направленный вниз по иерархии, прост (например: какие заказы принадлежат этому покупателю).

В то же время запрос, направленный вверх по иерархии, более сложен (например: какой покупатель поместил этот заказ). Кроме того, трудно представить неиерархические данные при использовании этой модели (подходит не для любой предметной области). Сложными становятся операции включения новых объектов и удаления устаревших объектов (в особенности обновление и удаление связей).

Наиболее известной иерархической СУБД до сих пор остается Information Management System (IMS) фирмы IBM.

Сетевая модель

Сетевые модели – это достаточно сложные структуры, состоящие из «наборов» – поименованных двухуровневых деревьев. «Наборы» соединяются при помощи «записей-связок», образуя цепочки и т. д.

К основным понятиям сетевой модели относятся: уровень, элемент (узел), связь. Узел – это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. В сетевой структуре каждый элемент может быть связан с любым другим элементом. В сетевых БД имеются указатели в обоих направлениях, которые соединяют родственную информацию.

Несмотря на то, что эта модель решает некоторые проблемы, связанные с иерархической моделью, она имеет ряд недостатков:

– сложность самой модели данных влечет за собой и сложность выполнения запросов;

– поскольку логика процедуры выборки данных зависит от физической организации этих данных, эта модель не является полностью независимой от приложения.

При различных способах реализации сетевых моделей наибольшее распространение получила модель КОДАСИЛ (CODASYL). Эта модель считается наиболее развитой сетевой моделью данных, постоянно развивается, поддерживается и сопровождается, являясь как бы стандартом. Разрабатывались и соответствующие СУБД: DMS корпорации UNIVAC, IDMS (Cullinane), DBMS (DEC), IDS (Honeywell). В настоящее время широко известна db_Vista, работающая на персональных компьютерах в DOS и Windows.

Реляционная модель

Существенный скачок в развитии технологии баз данных произошел в связи с предложенной Э. Коддом в 1970 г. парадигмой реляционной модели данных. Теперь логические структуры могли быть получены из одних и тех же физических данных, т. е. доступ к одним и тем же физическим данным мог осуществляться различными приложениями по разным путям. Стало возможным обеспечение целостности и независимости данных.

Слово «реляционный» происходит от английского «relation» (отношение). В реляционных БД (РБД) все данные представлены в виде простых таблиц, разбитых на строки и столбцы, на пересечении которых расположены данные. Запросы к таким таблицам возвращают таблицы, которые сами могут становиться предметом дальнейших запросов. Каждая база данных может включать несколько таблиц.

К недостаткам РБД относят:

- для сложной предметной области РБД состоит из очень большого числа таблиц, связанных между собой весьма сложным образом. Разработчику реализация такой БД будет чрезвычайно сложной задачей;

- ПО, разработанное для РБД, обычно оказывается жестко завязанным на структуру реляционных таблиц. Если в дальнейшем будет принято решение изменить структуру таблиц, то все программное обеспечение, разработанное для прежней структуры таблиц, придется изменять.

Более подробно РБД будут рассмотрены в подразд. 1.4.

Объектно-ориентированная модель

Объектно-ориентированная база данных (ООБД) – база данных, в которой данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями.

Возникновение ООБД определяется прежде всего потребностями практики: необходимостью разработки сложных информационных прикладных систем с высокой производительностью.

Однако ООБД на современном уровне развития имеют пока много недостатков. Среди них: сложность структуры; отсутствие отработанного языка; отсутствие поддержки авторизации.

Имеется ряд коммерческих объектно-ориентированных СУБД. В их числе GemStone (Servio Corporation), ONTOS (ONTOS), Object Store (Object Design, Inc.), Objectivity/DB (Objectivity, Inc.), Versant (Versant Object Technology, Inc.), Object Database (Object Database, Inc.), Itasca (Itasca Systems, Inc.), 02 (02 Technology).

Многомерная модель

Многомерные модели рассматривают данные либо как факты с соответствующими численными параметрами, либо как текстовые измерения, которые характеризуют эти факты. В розничной торговле, к примеру, покупка – это факт, объем покупки и стоимость – параметры, а тип приобретенного продукта, время и место покупки – измерения. Запросы агрегируют значения параметров по всему диапазону измерения, и в итоге получают такие величины, как общий месячный объем продаж данного продукта.

Многомерные БД рассматривают данные как кубы, которые являются обобщением электронных таблиц на любое число измерений. Кроме того, кубы поддерживают иерархию измерений и формул без дублирования их определений. Набор соответствующих кубов составляет многомерную базу данных (или хранилище данных).

Многомерные модели данных имеют три важные области применения, связанные с проблематикой анализа данных:

- хранилища данных, интегрирующие для анализа информацию из нескольких источников на предприятии;
- системы оперативной аналитической обработки (online analytical processing – OLAP), позволяющие оперативно получить ответы на запросы, охватывающие большие объемы данных в поисках общих тенденций;
- приложения добычи данных (Data Mining), которые служат для выявления знаний за счет полуавтоматического поиска ранее не известных шаблонов и связей в базах данных.

Первым продуктом, выполняющим OLAP-запросы, был Express (компания IRI). Сейчас среди многомерных СУБД можно выделить D3 Server, UniVerse, Unidata и др.

Отдельные БД могут объединять все данные, необходимые для решения одной или нескольких прикладных задач, или данные, относящиеся к какой-либо предметной области (например, финансам, образованию, кулинарии

и т. п.). Первые обычно называют **прикладными БД**, а вторые – **предметными БД** (соотносящимся с предметами организации, а не с ее информационными приложениями). (Первые можно сравнить с базами материально-технического снабжения или отдыха, а вторые – с овощными и обувными базами.)

Предметные БД позволяют обеспечить поддержку любых текущих и будущих приложений, поскольку набор их элементов данных включает в себя наборы элементов данных прикладных БД. Такая гибкость и приспосабливаемость позволяет создавать на основе предметных БД достаточно стабильные информационные системы, т. е. системы, в которых большинство изменений можно осуществить без вынужденного переписывания старых приложений.

Основывая же проектирование БД на текущих и предвидимых приложениях, можно существенно ускорить создание высокоэффективной информационной системы, т. е. системы, структура которой учитывает наиболее часто встречающиеся пути доступа к данным. Поэтому прикладное проектирование до сих пор привлекает некоторых разработчиков. Однако по мере роста числа приложений таких информационных систем быстро увеличивается число прикладных БД, резко возрастает уровень дублирования данных и повышается стоимость их ведения.

В настоящее время существует большое число программных продуктов, относящихся к этому классу. Это и открытые СУБД, используемые для небольших проектов и малых компаний (PostgreSQL, MySQL), и целые платформы (Oracle, MSSQL, DB2, Sybase) для крупных проектов.

1.3. Системы управления базами данных

Постоянное изменение основных свойств вычислительной техники и развитие СУБД обусловливали возникновение различных моделей взаимодействия пользователей с БД. Дадим краткую характеристику этим моделям в хронологическом порядке.

Модель с централизованной архитектурой

При использовании этой технологии БД, СУБД и прикладную программу (приложение) располагают на одном компьютере. Для такого способа организации не требуется поддержки сети, и все сводится к автономной работе.

Работа с этой моделью построена следующим образом. На компьютере хранится БД, установлены СУБД и приложение для работы с БД. Пользователь запускает приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к БД. Все обращения идут через СУБД, которая хранит внутри себя все сведения о физической структуре БД. СУБД инициирует обращения к данным, обеспечивая выполнение запросов пользователя. Результат СУБД возвращает в приложение, а приложение отображает результат выполнения запросов. Таким образом, в этой модели реализуется однопользовательский режим работы.

В данной модели реализуем и многопользовательский режим. С этой целью к мэйнфрейму подключалось несколько терминалов, но тогда приходилось обслуживать в рамках ресурсов одного компьютера весь комплекс

возникающих задач – от обработки и хранения данных до отображения информации и приема запросов пользователей. Таким образом, основным недостатком этой модели является резкое снижение производительности при увеличении числа пользователей.

Подобная архитектура использовалась в первых версиях СУБД DB2, Oracle, Ingres.

Модель с автономными персональными ЭВМ

Каждый пользователь имеет свою автономную персональную ЭВМ. База данных и СУБД копируются на компьютере каждого пользователя, и он работает с базой данных на своей ЭВМ.

Для данной модели характерна полная децентрализация данных. Основным недостатком модели является невозможность оперативного обновления данных на всех компьютерах при изменении их одним из пользователей.

Архитектура «файл–сервер»

Эта архитектура предполагает назначение одного из компьютеров сети в качестве выделенного сервера, на котором будут храниться файлы базы данных. В соответствии с запросами пользователей файлы с «файл–сервера» передаются на рабочие станции пользователей, где и осуществляется основная часть обработки данных. Центральный сервер выполняет в основном только роль хранилища файлов, не участвуя в обработке самих данных.

В этой модели БД в виде набора файлов находится на жестком диске файлового сервера. Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлены СУБД и приложение для работы с БД. На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый приложением пользовательский интерфейс, они через СУБД инициируют обращение к БД. СУБД инициирует обращения к данным, находящимся на файловом сервере, в результате которых часть файлов БД копируется на клиентский компьютер и обрабатывается, что обеспечивает выполнение запросов пользователя.

В случае изменения данные отправляются назад на файловый сервер для обновления БД. Результат СУБД возвращает в приложение. Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Основным недостатком модели является то, что при одновременном обращении множества пользователей к одним и тем же данным производительность работы системы резко падает из-за необходимости ждать, пока пользователь, работающий с данными, завершит свою работу. Кроме того, вся тяжесть вычислительной нагрузки при доступе к БД ложится на приложение клиента, так как при выдаче запроса на выборку информации из таблицы вся таблица БД копируется на клиентскую машину и выборка осуществляется на клиенте. Таким образом, не оптимально расходуются ресурсы клиентского компьютера и сети. В результате возрастает сетевой трафик и увеличиваются требования к аппаратным мощностям пользовательского компьютера.

Примеры: dBase, Microsoft Access, FoxPro, Paradox.

Архитектура «клиент–сервер»

Использование архитектуры «клиент–сервер» предполагает наличие некоторого количества компьютеров, объединенных в сеть, один из которых выполняет особые управляющие функции (является сервером сети). Архитектура «клиент–сервер» разделяет функции приложения пользователя (клиента) и сервера. Приложение-клиент формирует запрос к серверу, на котором расположена БД. Удаленный сервер принимает запрос и переадресует его серверу БД.

Работа построена следующим образом. БД в виде набора файлов и СУБД находятся на жестком диске сервера сети. Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлено клиентское приложение для работы с БД. На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый интерфейс, он инициирует обращение к СУБД (по сети от клиента к серверу передается лишь текст запроса). СУБД хранит внутри себя все сведения о физической структуре БД, расположенной на сервере. СУБД инициирует обращения к данным, находящимся на сервере, где осуществляется вся обработка данных, и лишь результат выполнения запроса копируется на клиентский компьютер. Таким образом, СУБД возвращает результат в приложение. Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Клиент-серверные СУБД, в отличие от файл-серверных, обеспечивают разграничение доступа между пользователями и мало загружают сеть и клиентские машины.

Недостаток клиент-серверных СУБД – в самом факте существования сервера (что плохо для локальных программ) и больших вычислительных ресурсах, потребляемых сервером. Кроме того, большое количество клиентских компьютеров, расположенных в разных местах, вызывает определенные трудности со своевременным обновлением клиентских приложений на всех компьютерах-клиентах.

Примеры: Firebird, Interbase, MS SQL Server, Oracle, PostgreSQL, Gupta, Informix, Sybase, DB2.

Трехзвенная (многозвенная) архитектура

В трехзвенной архитектуре вся бизнес-логика, ранее входившая в клиентские приложения, выделяется в отдельное звено, называемое сервером приложений. При этом клиентским приложениям остается лишь пользовательский интерфейс. Так, в качестве клиентского приложения может выступать Web-браузер.

В результате работа построена следующим образом. БД и СУБД располагаются на сервере сети. Существует специально выделенный сервер приложений, на котором располагается программное обеспечение (ПО) делового анализа (бизнес-логика). На каждом из множества клиентских компьютеров установлен так называемый «тонкий клиент» – клиентское приложение, реализующее интерфейс пользователя. Используя

предоставляемый приложением интерфейс, пользователь инициирует обращение к ПО делового анализа, расположенному на сервере приложений.

Сервер приложений анализирует требования пользователя и формирует запросы к БД. СУБД хранит внутри себя все сведения о физической структуре БД, расположенной на сервере. СУБД инициирует обращения к данным, находящимся на сервере, в результате которых результат выполнения запроса копируется на сервер приложений. Сервер приложений возвращает результат в клиентское приложение (пользователю). Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Таким образом, теперь при изменении бизнес-логики более нет необходимости изменять клиентские приложения и обновлять их у всех пользователей. Кроме того, максимально снижаются требования к аппаратуре пользователей.

Встраиваемые СУБД

Встраиваемая СУБД – библиотека, которая позволяет унифицированным образом хранить большие объемы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объемами данных (например, геоинформационные системы).

Примеры: OpenEdge, SQLite, BerkeleyDB, Sav Zigzag, Progress Enterprise Server, Centura SQLBase, IB Database, Sybase SQL Anywhere, Empress IMC.

1.4. Реляционные базы данных

В конце 60-х гг. появились работы, в которых обсуждались возможности применения различных табличных моделей данных, т. е. возможности использования привычных и естественных способов представления данных. Наиболее значительной из них была статья сотрудника фирмы IBM Э. Ф. Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), где впервые был применен термин «реляционная модель данных».

Будучи математиком по образованию, Э. Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение – relation (англ.)

Основным понятием РБД является сущность. **Сущность** (entity) – любой объект предметной области, информацию о котором необходимо хранить в БД. Сущность предметной области является результатом абстрагирования реального объекта предметной области путем выделения и фиксации набора его свойств (атрибутов). **Атрибуты сущности** – поименованные свойства, характеризующие сущность. Например, сущность СТУДЕНТ имеет атрибуты ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, № СТУДЕНЧЕСКОГО БИЛЕТА и т. п.

Сущности вступают в связи друг с другом через свои атрибуты. Каждая группа атрибутов, описывающих одно реальное проявление сущности, представляет собой экземпляр (instance) сущности. Иными словами, **экземпляры сущности** – это реализации сущности, отличающиеся друг от друга и допускающие однозначную идентификацию.

Связь – ассоциирование двух или более сущностей. Если бы назначением БД было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации БД – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных БД нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность логических моделей.

В дальнейшем будем рассматривать РБД как совокупность сущностей, отношений и связей между ними, содержащих всю информацию из предметной области. Однако пользователи могут воспринимать такую базу данных как совокупность таблиц.

Поскольку сущности в реляционной БД хранятся в форме таблиц, необходимо указать на соответствие терминов и сферы их использования.

Как видно из табл. 1.1, атрибуты сущности хранятся в столбцах таблицы и называются **поля**, а данные (экземпляры сущности) хранятся в строках и называются **записи.**

Таблица 1.1

Предметная область	Реляционная модель	Табличная структура
Сущность	Таблица	Таблица
Атрибут	Поле	Столбец
Экземпляр	Запись	Строка

Таблицы реляционной БД обладают следующими свойствами:

1. Каждая таблица состоит из однотипных строк и имеет уникальное имя.
2. Все записи должны иметь одинаковую структуру, т. е. одно и то же количество полей с соответственно совпадающими именами.
3. Все поля должны быть атомарны (неделимы), т. е. в каждой позиции таблицы на пересечении строки и столбца всегда имеется в точности или одно значение или нет ничего.
4. Все записи должны быть уникальны.
5. Поля имеют уникальные имена, и в каждом из них размещаются однородные (однотипные) значения данных.

Для манипулирования таблицами Э. Ф. Кодд создал инструмент – **реляционную алгебру**. Каждая операция этой алгебры использует одну или несколько таблиц в качестве ее операндов и продуцирует в результате новую таблицу, т. е. позволяет «разреза́ть» или «склеивать» таблицы.

Один из основных вопросов, который может возникнуть на этом этапе: почему нельзя обойтись одной большой таблицей, в которой будет храниться вся информация обо всех сущностях предметной области?

Рассмотрим пример. Пусть имеется таблица ПРЕПОДАВАТЕЛИ (табл. 1.2). На первый взгляд, данная таблица вполне хороша, так как хранит в себе все требуемые атрибуты.

Таблица 1.2. Преподаватели

Фамилия	*Должность*	*Кафедра*	*Телефон*	*Предмет*	*асы*
Иванов	Доцент	БИ	351	Информатика	00
Петров	Профессор	БИ	351	Базы данных	00
Петров	Профессор	БИ	351	Информатика	00

Однако при использовании такой универсальной таблицы возникает ряд проблем (аномалий):

1. **Аномалия обновления.** Проблема, связанная с избыточностью данных. Данные практически всех полей многократно повторяются. Повторяются и некоторые наборы данных (Фамилия–Должность–Кафедра–Телефон, Предмет–Часы). Это приводит к тому, что одни и те же данные необходимо вводить несколько раз, что многократно увеличивает время ввода и может привести к опечаткам. Кроме того, при необходимости изменения номера телефона кафедры его придется изменять в таблице многократно.

2. **Аномалия вставки (включения).** Невозможность ввести данные в таблицу ввиду отсутствия других данных. В таблицу невозможно добавить сведения о преподавателе, если он не ведет никакой предмет.

3. **Аномалия удаления.** Непреднамеренная потеря данных в связи с удалением других данных. Так, при удалении сведений о преподавателе (например, при его увольнении) пропадают все данные о предмете, который он вел.

Для того чтобы устранить аномалии, необходимо провести процедуру нормализации. **Нормализация** – пошаговый процесс разложения (декомпозиции) исходных таблиц на более простые таблицы, связанные между собой.

Принципиально важным достоинством реляционных БД является, то, что связи между таблицами не зависят от физической структуры данных. Таким образом, они являются не физическими, привязанными к местоположению данных (как, например, при использовании табличных процессоров, где значение ячейки A1 одного листа связывается со значением ячейки B2 другого листа), а **логическими**. В связи с этим при выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию.

Между двумя сущностям, например А и В, возможны четыре вида логических связей. В БД они называются **структурными** связями.

«Один-к-одному» (1:1)

Каждой записи таблицы А соответствует только одна запись таблицы В, и наоборот.

Если между двумя таблицами связь 1:1, то эти таблицы можно объединить в одну. Обычно связь используется для разделения доступа к данным для различных пользователей.

«Один-ко-многим» (1:М)

Каждой записи таблицы А может соответствовать несколько записей таблицы В, а одной записи таблицы В соответствует только одна запись таблицы А.

«Многие-к-одному» (М:1)

Каждой записи таблицы А соответствует только одна запись таблицы В, а одной записи таблицы В может соответствовать несколько записей таблицы А.

«Многие-ко-многим» (М:М)

Каждой записи таблицы А может соответствовать несколько записей таблицы В, а каждой записи таблицы В может соответствовать несколько записей таблицы А.

Рассмотрим в качестве примера сущности СТУДЕНТЫ и АДРЕСА:

1) связь между данными сущностями будет 1:1, в случае если один студент имеет только один адрес, то по одному адресу может проживать только один студент;
2) в случае если в БД фиксируется адрес прописки и адрес проживания, то получим связь 1:М, так как один студент может иметь несколько адресов, а по одному адресу проживает только один студент;
3) в случае если в БД фиксируется только адрес проживания, но возможна ситуация, когда студенты проживают в общежитии (т. е. по одному адресу), получим связь М:1;
4) если возможна ситуация и п. 2 и п. 3, то получим связь М:М.

Как видно из данного примера в одной и той же предметной области могут быть разные логические связи между одними и теми же сущностями. Поэтому только Заказчик (пользователь) БД, который разбирается в предметной области, может их правильно определить на этапе проектирования.

Для представления типов связи обычно используют **ER-диаграммы** (от англ. Entity-Relationship, т. е. сущность-связь). В СУБД MS Access ER-диаграмма представлена в виде **Схемы данных.**

Для реализации связей используются специальные **ключевые поля**. Различают первичные и внешние ключи.

Первичный ключ – одно или несколько полей таблицы, однозначно идентифицирующих каждую запись.

Первичный ключ обладает следующими свойствами:

- значения первичного ключа не могут повторяться;

- первичный ключ не может принимать неопределенное значение (быть пустым).

В случае если в таблице нет ни одного поля, подходящего под эти требования можно создать **составной первичный ключ**, состоящий из нескольких полей, которые в своей совокупности не повторяются. При этом необходимо соблюдать требование минимальности: ни один из атрибутов не может быть исключен из набора без нарушения уникальности.

В качестве первичных ключей нежелательно использовать поля, содержащие длинный текст (например, название вуза). В этом случае проще добавить в таблицу новое поле (например, код вуза).

Также не рекомендуется создавать составные ключи, состоящие из более чем трех полей. Такая конструкция будет неудобна в использовании.

Внешний ключ – одно или несколько полей, связанных с первичным ключом из другой таблицы. При этом данные во внешнем ключе могут повторяться.

При создании связей первичные и внешние ключи используются следующим образом:

1) *«один-к-одному»* – в таблицах А и В создаются одинаковые первичные ключи;

2) *«один-ко-многим»* и *«многие-к-одному»* – на стороне «один» создается первичный ключ, а на стороне «много» – соответствующий ему внешний ключ;

3) *«многие-ко-многим»* – связь М:М невозможно реализовать напрямую. Для установления связи используют третью таблицу, которую будем называть отношением. **Отношение** – это процесс или событие, в котором участвуют несколько сущностей.

Приведем примеры, используя сущности СТУДЕНТЫ и АДРЕСА.

«Один-к-одному» (рис. 1.1)

В таблице Студенты создадим первичный ключ № студ. билета. В таблице Адреса создадим такой же первичный ключ и соединим их между собой.

Рис. 1.1

«Один-ко-многим» (рис. 1.2)

В таблице Студенты создадим первичный ключ № студ. билета. В таблице Адреса создадим такое же поле, но которое будет являться внешним ключом (не уникальным), и соединим их между собой.

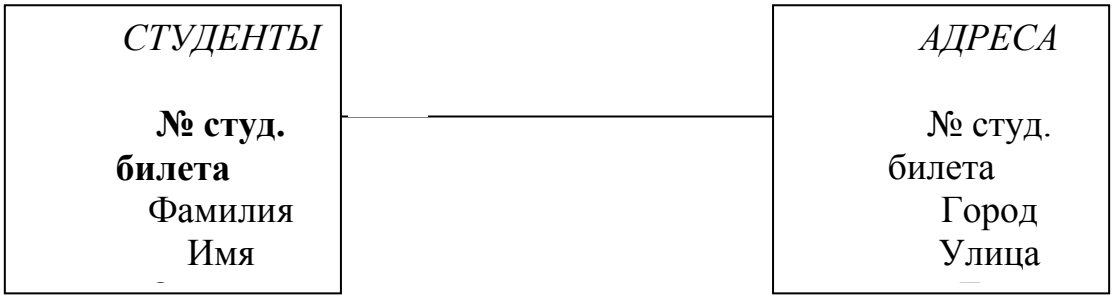


Рис. 1.2

«Многие-к-одному» (рис. 1.3)

В таблице Адреса создадим новое поле первичного ключа Код Адреса. В таблице Студенты создадим такое же поле, но оно будет являться внешним ключом, и соединим их между собой.

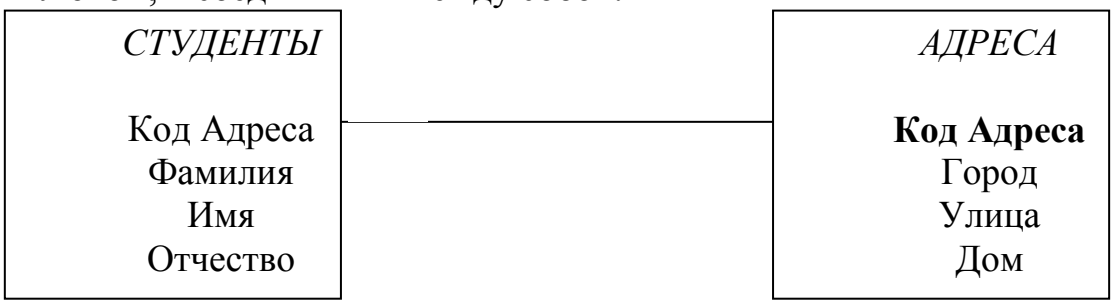


Рис. 1.3

«Многие-ко-многим» (рис. 1.4)

В таблице Студенты создадим первичный ключ № студ. билета. В таблице Адреса создадим первичный ключ Код Адреса. Создадим третью таблицу-отношение и назовем ее ПРОЖИВАНИЕ. Добавим в эту таблицу внешние ключи № студ. билета и Код Адреса и свяжем соответствующие пары первичных и внешних ключей.

При этом получим еще пару связей 1:М между таблицами Студенты–Проживание и Адреса–Проживание.

Одной из основных ошибок при определении связей является проверка типа связи только в одну сторону. Например, делается вывод о том, что между сущностями СТУДЕНТЫ и АДРЕСА связь «один-ко-многим», так как один студент может иметь несколько адресов. При этом забывается необходимость проверки от Адресов к Студентам (по одному адресу может проживать несколько студентов), что приводит к неправильной модели предметной области и, как следствие, к неработоспособной БД.

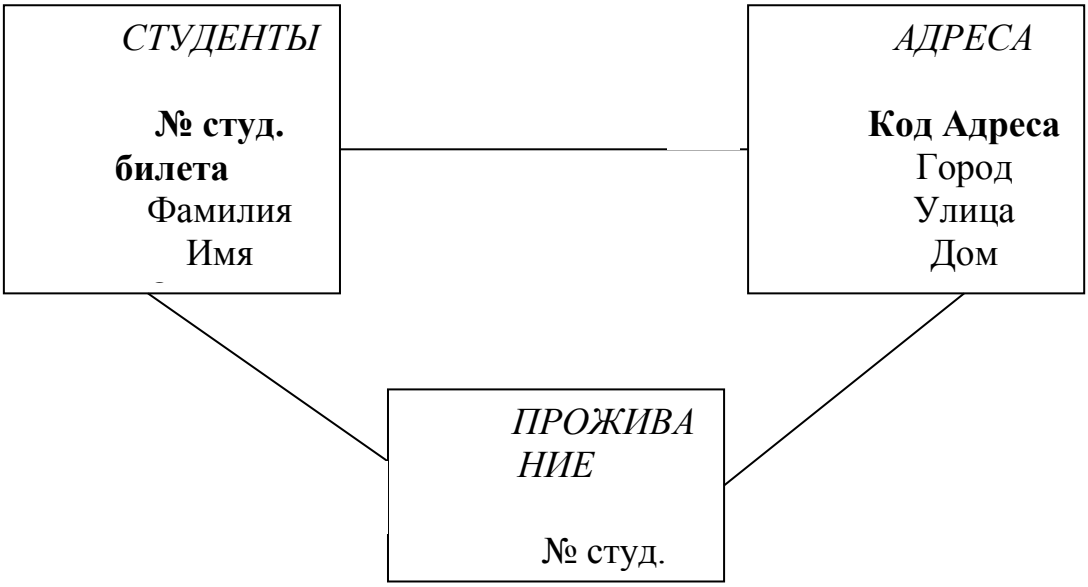


Рис. 1.4

Вернемся к примеру (табл. 1.2) и проведем нормализацию таблицы (результат нормализации представлен на рис. 1.5).

Во-первых, выделим независимые сущности: ПРЕПОДАВАТЕЛИ, КАФЕДРЫ и ПРЕДМЕТЫ. Затем определим связи между ними.

На одной кафедре могут работать несколько преподавателей, но один преподаватель может работать только на одной кафедре. Следовательно, связь между Кафедрами и Преподавателями «один-ко-многим».

Один предмет могут вести несколько преподавателей, а один преподаватель может вести несколько предметов. Следовательно, связь между Преподавателями и Предметами – «многие-ко-многим».

Реализуем выявленные связи. Для связи Кафедры–Преподаватели определим в качестве первичного ключа таблицы Кафедры (сторона «один») поле **№ кафедры**. Несмотря на то что Название кафедры тоже можно было бы использовать в качестве первичного ключа, оно содержит длинный текст, и поэтому неудобно в использовании. В таблице Преподаватели создадим такое же поле, но выступающее уже в качестве внешнего ключа и соединим их между собой.

Как было уже сказано, связь «многие-ко-многим» реализуется через таблицу-отношение. В качестве такой таблицы используем **Расписание**. В таблице Предметы создадим новое поле первичного ключа **Код Предмета**. В таблице Преподаватели первичным ключом будет новое поле **Личный Код.** Фамилия преподавателя не может быть первичным ключом, так как может содержать повторяющиеся данные. Добавим соответствующие внешние ключи в таблицу Расписание и свяжем соответствующие пары первичных и внешних ключей.

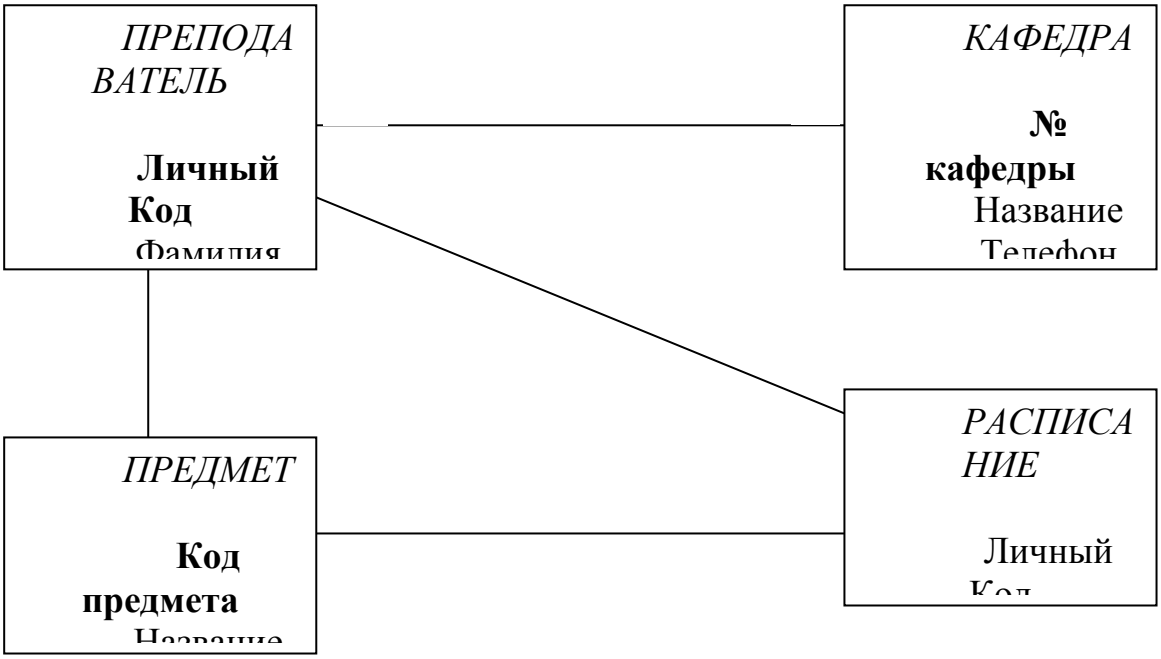


Рис. 1.5

Теперь проверим созданную ER-модель на аномалии. Аномалия обновления отсутствует, так как избыточность была ликвидирована, и все данные хранятся только один раз. Аномалия вставки ликвидирована, так как мы можно добавлять данные о новых предметах и преподавателях независимо друг от друга. Аномалия удаления также ликвидирована, так как удаление, например, сведений о предмете не приводит к удалению сведений о преподавателе.

Одна из основных проблем в любой модели БД – это проблема целостности данных. **Целостность данных** – устойчивость хранимых данных к неверным изменениям или разрушению, связанным с ошибочными действиями пользователей.

Целостность данных предполагает:

1) надежность данных – отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте;
2) ссылочную целостность – защиту данных при удалении или обновлении связанных данных разных таблиц.

Надежность данных достигается соблюдением двух условий: достоверности и непротиворечивости.

Достоверность данных – правильность данных в любой момент времени. БД не может контролировать правильность каждого отдельного вводимого значения (хотя каждое значение можно проверить на правдоподобность). Например, нельзя обнаружить, что вводимое значение 5 (представляющее номер месяца) в действительности должно быть равно 3. С другой стороны, значение 20 явно будет ошибочным.

Непротиворечивость данных – отсутствие противоречия между введенными данными и предметной областью, а также между вводимыми данными в разные поля одной записи.

Рассмотрим значение «100», вводимое в поле Год рождения. Данное значение достоверно, так как это число целое, и такой год рождения

вполне мог существовать. Сложнее дело обстоит с проверкой на непротиворечивость. Если предметной областью БД является экономический факультет СПбГУТ, то вводимое значение противоречит предметной области, если же предметная область – античные философы, то значение непротиворечиво. Однако на этом проверка не закончена. Даже если БД посвящена античным философами, но в обновляемой записи в поле Год смерти стоит число «90», то вводимые данные также противоречивы (число года рождения не может быть больше числа года смерти). Условие на непротиворечивость закладываются непосредственно заказчиком БД на этапе проектирования.

Ссылочная целостность определяет систему правил для поддержания связей между записями в связанных таблицах.

При связи «один-ко-многим» актуальными становятся вопросы:

- что делать при удалении записи в таблице на стороне «один»?
- что делать при изменении данных в поле первичного ключа?

Для обоих случаев есть два варианта: ограничение операции или ее каскадирование.

В случае **ограничения** пользователь не сможет изменить данные в первичном ключе, если есть связанная запись в другой таблице, а также не сможет удалить запись в таблице на стороне «один», если есть связанная запись на стороне «много». Это делается для того, чтобы, во-первых, не потерять связь между записями, а во-вторых, не получить «висячую» запись (при удалении сведений о преподавателе должны исчезнуть ссылки на него в расписании).

При **каскадировании** происходит автоматическое обеспечение ссылочной целостности. Изменение значения в первичном ключе приводит к автоматическому обновлению соответствующих значений во всех внешних ключах. Удаление же записи в главной таблице (сторона «один») приводит к автоматическому удалению связанных записей в подчиненных таблицах (сторона «много»).

1.5. Жизненный цикл базы данных

Организация структуры БД формируется исходя из следующих соображений:

– адекватность описываемому объекту/системе должна быть на уровне концептуальной и логической моделей;

– удобство использования для ведения учета и анализа данных должно быть на уровне физической модели.

Концептуальная модель – модель предметной области, состоящей из перечня взаимосвязанных понятий, используемых для описания этой области (вместе со свойствами и характеристиками, классификацией этих понятий по типам, ситуациям, признакам в данной области), и законов протекания процессов в ней. Концептуальная модель разрабатывается без какой-либо ориентации на конкретную СУБД, ее цель – получение семантических (смысловых) моделей, отражающих информационное содержание предметной области.

Логическая модель описывает понятия предметной области и их взаимосвязи и является прототипом будущей БД. Она представляет собой запись концептуальной модели на языке СУБД. Ее цель – описать объекты предметной области абстрактными понятиями модели данных так, чтобы это описание не противоречило семантике предметной области и было по возможности наилучшим.

Основным понятием логической модели является сущность.

Физическая модель является компьютерно-ориентированной моделью и описывает физическое расположение хранимых данных, процедур и методов доступа к ним.

Таким образом, при проектировании БД решаются две основных проблемы:

- каким образом отобразить объекты предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности лучшим. Часто эту проблему называют проблемой логического проектирования;
- как обеспечить эффективность выполнения запросов к базе данных, т. е. каким образом, имея в виду особенности конкретной СУБД, расположить данные во внешней памяти, создание каких дополнительных структур (например, индексов) потребовать и т. д. Эту проблему называют проблемой физического проектирования баз данных.

В данном пособии, говоря о проектировании, будем иметь в виду только концептуальное и логическое проектирование. Вопросы физического проектировании будут частично рассмотрены в разд. 2, посвященном созданию БД.

Жизненный цикл базы данных охватывает следующие основные этапы:

1) *проектирование* – происходит построение концептуальной и логической модели БД. Проектирование осуществляется заказчиком (конечным пользователем) БД. Этап проектирования полностью осуществляется «на бумаге» и никак не связан с использованием СУБД;

2) *создание* (реализация) БД – осуществляется разработчиком на компьютере посредством конкретной СУБД. Основой для создания является техническое задание, составленное заказчиком на предыдущем этапе;

3) *наполнение* – осуществляются конвертирование и загрузка данных, а также ввод данных. Под конвертированием данных понимают перенос любых существующих данных в новую БД. Этот шаг выполняется только в том случае, если новая БД заменяет старую. В противном случае данный этап предполагает непосредственный ввод данных;

4) *тестирование* – выполняется в целях поиска ошибок. Прежде чем использовать новую БД на практике, ее следует тщательно протестировать. Этого можно добиться путем разработки продуманной стратегии тестирования с использованием реальных данных, которая должна быть построена таким образом, чтобы весь процесс тестирования выполнялся строго последовательно и методически правильно;

5) *эксплуатация и сопровождение* – основной этап, включающий в себя использование БД и поддержка ее нормального функционирования по окончании развертывания;

6) *модернизация* – в процессе эксплуатации могут появиться новые функции, которые не были автоматизированы в БД, в этом случае на этапе модернизации может возникнуть необходимость создания нового шаблона документа, нового запроса и т. д. Важно подчеркнуть, что речь идет только о небольших добавлениях в первоначальный проект. Если же кардинально изменилась предметная область, можно говорить о необходимости проектирования новой БД.

2. ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

2.1. Этапы проектирования базы данных

Этап проектирования является первым и одним из важнейших этапов жизненного цикла БД. От корректного проектирования зависит то, насколько эффективно БД будет реализовывать свою цель – автоматизировать бизнес-процессы предприятия. Неоптимальные решения и прямые ошибки, заложенные на этапе проектирования, впоследствии очень трудно устраняются, поэтому этот этап является основополагающим.

Методически правильно начинать работу с карандашом и листом бумаги в руках, не используя компьютер. На данном этапе он просто не нужен. Проектирование БД всегда осуществляется заказчиком и заканчивается написанием технического задания (ТЗ) для разработчика. Однако для этого заказчик должен владеть соответствующей терминологией и знать, хотя бы в общих чертах, технические возможности основных СУБД. К сожалению, на практике это встречается не всегда. В связи с этим обычно используют следующие подходы:

- демонстрируют заказчику работу аналогичной БД, после чего согласовывают спецификацию отличий;
- если аналога нет, выясняют круг задач и потребностей заказчика, после чего разработчик помогает ему подготовить техническое задание.

Существует два основных подхода к проектированию БД: «**нисходящий**» и «**восходящий**».

При восходящем подходе работа начинается с самого нижнего уровня – уровня определения атрибутов сущностей, которые на основе анализа существующих между ними связей группируются в отношения, представляющие типы сущностей и связи между ними.

Восходящий подход лучше всего подходит для проектирования простых БД с относительно небольшим количеством атрибутов. Однако использование этого подхода существенно усложняется при проектировании БД с большим количеством атрибутов, установить среди которых все существующие функциональные зависимости довольно затруднительно.

Более подходящей стратегией проектирования сложных БД является использование нисходящего подхода. Начинается этот подход с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов. Нисходящий подход демонстрируется в концепции модели «сущность–связь». В этом случае работа начинается с идентификации сущностей и связей между ними, интересующих данную организацию в наибольшей степени.

Помимо этих подходов для проектирования могут применяться другие подходы, например типа **«изнутри наружу»** или **«смешанная стратегия проектирования»**. Подход типа «изнутри наружу» похож на восходящий подход, но отличается от него начальной идентификацией набора

основных сущностей с последующим расширением круга рассматриваемых сущностей, связей и атрибутов, которые взаимодействуют с первоначально определенными сущностями. В смешанной стратегии сначала восходящий и нисходящий подходы используются для разных частей модели, после чего все подготовленные фрагменты собираются в единое целое.

Рассмотрим более подробно нисходящий подход. Этап проектирования состоит из нескольких шагов.

1. Описание предметной области.

Заказчик, как лицо, обладающее всей полнотой информации о протекающих бизнес-процессах и информационных потоках в компании, должен описать ту сферу деятельности, которая подлежит автоматизации. Здесь должны быть выявлены основные сущности, участвующие в описываемых бизнес-процессах, и степень их детализации, а также варианты использования этих сущностей (сами бизнес-процессы).

При этом очень важно не ограничиваться взаимодействием с головным подразделением заказчика, а провести обсуждение со всеми службами и подразделениями, которые могут оказаться поставщиками данных в базу или их потребителями.

Необходимая для проектирования информация может быть получена посредством:

- опроса и анкетирования отдельных сотрудников предприятия, особенно ведущих специалистов в наиболее важных областях ее деятельности;
- наблюдения за деятельностью предприятия;
- изучения документов, которые используются для сбора и представления информации;
- использования опыта проектирования других систем.

Рассмотрим в качестве примера предметную область «Приемная комиссия вуза». Основным заказчиком подобной БД может быть ректор вуза. В результате его интервьюирования могут быть получены данные о том, что основными сущностями являются: Абитуриенты, Специальности и Экзамены.

Основными вариантами использования будут являться: сбор анкетных данных об абитуриентах, фиксация оценок, полученных на экзаменах, расчет проходных баллов, зачисление студентов.

2. Построение ER-модели.

На этом шаге заказчику, скорее всего, понадобится помощь разработчика, который мог бы помочь сформулировать и нормализовать связи, отражающие схему информационного взаимодействия между сущностями в понятных ему терминах. Завершением данного шага является построение ER-модели.

В данном примере можно задать следующие вопросы:

Сколько абитуриентов может поступить на обучение по одной специальности?

На обучение по какому количеству специальностей может поступать один абитуриент?

Сколько экзаменов сдается при поступлении на обучение по разным специальностям одним абитуриентом?

Таким образом, получим следующие связи: Абитуриенты–Специальности – M:M, Абитуриенты–Экзамены – 1:M, Специальности–Экзамены – 1:M

3. Определение круга пользователей базы данных.

Необходимо учесть, что при проектировании важны даже не столько конкретные пользователи, сколько роли, которые они играют. Если есть три методиста с одинаковыми функциями и им всем нужен доступ к БД, то получаем лишь одну роль.

В этом примере, БД может быть полезна не только приемной комиссии, но и всем деканатам, службе, отвечающей за проживание студентов в общежитии, студенческому отделу кадров, военно-учетному столу, бухгалтерии.

4. Определение информационных потребностей пользователей.

Составив список пользователей БД, необходимо подробно описать все их (кого?) функции, которые могут быть автоматизированы при помощи БД. Для этого необходимо провести подробное интервьюирование всех пользователей и выписать из полного перечня их функций (например, из должностной инструкции) те, которые можно автоматизировать.

Например, формирование приказа на зачисление студентов может существенно повысить эффективность всех вышеперечисленных подразделений, так как избавит от необходимости многократного создания такого списка вручную.

5. Составление списка требуемых данных.

Для реализации каждой из функций нужно большое количество данных. Основная задача при этом – не включать в БД те данные, которые никому не нужны, а с другой стороны, не забыть то, что может понадобиться пользователям.

С этой целью для каждой из функций желательно получить списки:

- исходных данных, с которыми работают пользователи;
- выходных данных, которые необходимы пользователям для внутренней работы;
- выходных данных, которые не являются необходимыми для заказчика, но которые он должен предоставлять в другие организации (например, в городскую приемную комиссию).

Для составления такого перечня необходимо внимательно рассмотреть все используемые входные и выходные документы. Например, если для сдачи итоговой формы в министерство необходимо знать средний балл аттестата каждого абитуриента, то такие данные должны быть предусмотрены в БД.

Часть данных (например, фамилия абитуриента) будет повторяться для разных функций, поэтому необходимо составить список неповторяющихся данных (атрибутов сущностей), снабдив их кратким описанием. Отдельно стоит отметить те данные, которые могут быть получены на основании других данных путем вычислений.

В данном примере в результате проектирования был получен следующий список данных:

Фамилия, Имя, Отчество, Дата рождения, Пол абитуриента.

Название факультета, Название специальности, Шифр специальности.

Оценка, полученная на экзамене, Дата экзамена.

2.2. Этапы создания базы данных

2.2.1. Выбор СУБД

После получения ТЗ от пользователя, в котором представлена ER-модель и списки требуемых данных, разработчику необходимо перейти к реализации проекта на компьютере. При этом первой задачей является выбор используемой реляционной СУБД.

Рассмотрим процесс создания БД на основе реляционной СУБД MS Access. Microsoft Access в настоящее время является одной из самых популярных среди настольных СУБД. Однако, как и все файл-серверные СУБД он не лишен ряда недостатков. Не вызывают сомнения упреки в плохой защищенности данных, медленной работе, проблемах с дублированием данных при резервном копировании, трудностях администрирования, катастрофическом снижении скорости обработки при возрастании объемов данных и т. п.

Однако используемые для решения проблемы средства должны соответствовать сложности самой проблемы. Так, вряд ли имеет смысл тратить на разработку и внедрение информационной системы средства, существенно большие, чем весь годовой оборот предприятия, а для многих предприятий сферы малого и среднего бизнеса дело обстоит именно так.

Итак, где же используется на сегодняшний день MS Access? Прежде всего в государственных (муниципальных) учреждениях, сфере образования, сфере обслуживания, малом и среднем бизнесе. Специфика возникающих задач заключается в том, что объемы данных не являются катастрофически большими, частота обновлений не бывает слишком высокой, организация территориально обычно расположена в одном небольшом здании, количество пользователей колеблется от одного до 10–15 человек. В подобных условиях использование настольных СУБД для управления информационными системами является вполне оправданным, и они с успехом применяются.

Среди особенностей MS Access следует отметить:

- высокую степень универсальности и продуманности интерфейса, который рассчитан на работу с пользователями самой различной квалификации. В частности, реализована система управления объектами БД, позволяющая гибко и оперативно переходить из режима конструирования в режим их непосредственной эксплуатации;
- глубоко развитые возможности интеграции с другими программными продуктами, входящими в состав Microsoft Office (публикация в Word, анализ в Excel), а также с любыми программными продуктами, поддерживающими технологию OLE;
- богатый набор визуальных средств разработки, включающий средства автоматической, автоматизированной (мастера и шаблоны) и ручной разработки (конструкторы).
- хранение всех объектов, в том числе и программного кода, в одном файле;
- возможность публикации данных в сети Интернет;

- благодаря встроенному языку VBA в самом Access можно писать приложения, работающие с базами данных.

Любая СУБД должна выполнять ряд обязательных функций:

- определение структуры БД;
- заполнение БД;
- поиск и выборку данных;
- представление данных;
- поддержку языков БД;
- защиту физической и логической целостности данных;
- управление доступом к БД и др.

Основные элементы (объекты) MS Access включают в себя:

- таблицы (tables) для хранения данных;
- связи (relationships) между таблицами для обеспечения единства данных;
- формы (forms) для ввода и просмотра данных на экране;
- отчеты (reports) для вывода данных на печать;
- запросы (queries) для поиска необходимых данных по критерию;
- модули (modules) для хранения процедур на языке Visual Basic for Applications;
- макросы (macros) для автоматизации работы базы данных.
- страницы доступа к данным (data access pages)

2.2.2. Создание таблиц

После выбора подходящей для решаемой задачи СУБД необходимо перейти непосредственно к реализации БД в соответствии с ТЗ. Первым шагом будет преобразование сущностей и отношений в таблицы, а атрибутов сущностей – в поля таблиц.

Для создания таблиц потребуется перечень сущностей и отношений из ER-модели и перечень неповторяющихся данных (атрибутов сущностей).

При создании таблиц необходимо определить имена этих таблиц и параметры полей.

Для каждого поля необходимо:

- установить тип поля, размер и допустимый диапазон значений;
- определить вычислимость значений атрибута с использованием другой информации (если атрибут вычисляемый, то его необходимо исключить из таблицы);
- произвести внешнее кодирование, т. е. заменить длинные названия полей на короткие;
- создать механизмы повышения надежности ввода данных.

Все эти мероприятия служат одной важнейшей цели – снизить вероятность ошибок, допускаемых пользователями при вводе данных. Источниками неверных данных в таблице могут быть ошибки действий (ошибки пользователя) и ошибки создания (ошибки разработчика).

Остановимся более подробно на ошибках действий. Пользователь может допускать ошибки при вводе данных в двух случаях: случайно (опечатки) и

когда ему что-то непонятно. Таким образом, задачей разработчика является снижение вероятности таких ошибок до минимума.

Для этих целей в MS Access существует целый ряд инструментов, содержащихся в Конструкторе таблиц, которые будут перечислены ниже.

1. **Имя поля** – определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц), является обязательным параметром поля и не может превышать 64 символа. При этом полное имя поля содержит в себе еще и имя таблицы (например: Студенты.Фамилия). Чем более понятно имя поля, тем меньше ошибок допустит пользователь при вводе данных (например, имя поля ПочтКод не является интуитивно понятным для пользователя в отличие от имени Почтовый индекс).

2. **Тип поля** – определяет тип данных, которые могут содержаться в данном поле, является обязательным свойством и в MS Access может принимать следующие значения (табл. 2.1).

Таблица 2.1

Тип поля	Описание	Размер
Текстовый	Для хранения обычного неформатированного текста ограниченного размера. Над числами в текстовом поле нельзя производить вычисления	До 255 символов
Поле Memo	Хранит длинный текст или числа, например пометки или описание. Не допускает регламентирования вводимых данных	До 65 535 символов
Числовой	Для хранения действительных чисел, над которыми можно производить математические вычисления	1, 2, 4 или 8 байт
Дата/время	Для хранения календарных дат и времени	8 байт
Денежный	Для хранения денежных сумм с денежной единицей (точность до 15 знаков в целой части и 4 в дробной)	8 байт
Счетчик	Специальный тип для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием или случайных чисел	4 байта
Логический	Тип для хранения логических данных, которые могут принимать только два значения (например, «Да/Нет», «Истина/Ложь», «Включено/Выключено»)	1 бит
Поле объекта OLE	Специальное поле, предназначенное для хранения объектов, созданных другими приложениями (например, документы Word, электронные таблицы Excel, рисунки, звуки). Объекты используют протокол OLE (Object Linking and Embedding) и могут быть связанными или внедренными	До 1 Гбайт
Гиперссылка	Специальное поле для хранения адресов внешних объектов (URL, ссылки на документы Word и т. п.)	До 64 000 символов

Благодаря правильно выбранному типу поля снижается вероятность опечаток (так, например, в числовое поле нельзя случайно ввести букву вместо цифры).

3. **Размер поля** – определяет предельную длину данных, которые могут размещаться в данном поле. Для текстовых данных указывает максимальное количество символов, вводимых в поле. Для числовых данных размер поля – это поименованный диапазон значений (например, размер поля «байт» позволяет ввести целые числа от 0 до 255).

Размер поля также позволяет избежать случайных ошибок (например, при установке размера поля ПОЛ равным 1 пользователь не сможет ввести «муж», «мужской» и т. п.).

4. **Формат поля** – определяет способ форматирования данных в поле, позволяет улучшить восприятие данных (например, можно указать на необходимость отображения всех вводимых символов как прописные).

5. **Маска ввода** – регламентирует ввод каждого символа. Задает строку символов, каждый из которых кодирует обязательную или необязательную букву или цифру. Кроме этого, текстовые символы, прописанные в маске, будут автоматически отображаться (или храниться) в поле.

Например, маска ввода для поля Телефон может быть задана в виде (000) 000-0000 (0 – обязательная цифра). При этом в поле будут автоматически вводиться скобки и дефис.

6. **Подпись** – определяет заголовок столбца таблицы для данного поля. Подпись поля, по сути, является «псевдонимом» поля и может достигать 2048 символов. Имя поля должно быть коротким, так как при разработке БД использование длинных полей крайне неудобно. В то же время пользователю необходимо видеть длинные и понятные имена полей. Для этого используется подпись, заменяющая имя поля в таблицах, формах и отчетах. Если подпись не указана, то в качестве заголовка столбца используется Имя поля.

7. **Описание –** комментарий к полю, выводимый в строке состояния. Описание поля позволяет пользователю еще более подробно объяснить назначение поля и соответственно снизить вероятность ошибок.

8. **Значение по умолчанию –** то значение, которое вводится в ячейки поля автоматически.

В качестве значения по умолчанию имеет смысл использовать наиболее часто встречающееся значение. Это позволит существенно ускорить процесс ввода и уменьшить количество опечаток. Например, если в поле Страна по умолчанию задать «Россия», то для всех российских студентов в поле Страна это значение будет появляться автоматически.

9. **Условие на значение –** ограничение, используемое для проверки правильности ввода данных.

Существует условие на значение поля и записей. Условие на значение поля является логическим ограничением на вводимые данные в поле, поддерживающее непротиворечивость данных (например, Возраст > 18).

Условие на значение записей позволяет сравнить данные, хранимые в разных полях (например: Год смерти > Год рождения).

10. **Сообщение об ошибке –** текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка выполняется автоматически, если задано свойство Условие на значение).

11. **Обязательное поле** – свойство, определяющее обязательность заполнения данного поля при наполнении базы.

12. **Пустые строки** – свойство, разрешающее ввод пустых строковых данных (значение Null).

13. **Индексированное поле** – поле, обладающее свойством, когда все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, чтобы значения в записях проверялись по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных (уникальный индекс).

Важно отметить, что наличие или отсутствие вышеперечисленных свойств зависит от типа поля. Например, для полей, в которых хранятся рисунки, звукозаписи, видеоклипы и другие объекты OLE, большинство вышеуказанных свойств не имеют смысла.

В данном примере зададим следующие свойства полей:

Фамилия – текстовое, обязательное;

Имя – текстовое, обязательное;

Отчество – текстовое, обязательное;

Дата рождения – дата/время, обязательное;

Пол – текстовое, размер поля равен 1, обязательное;

Название факультета – текстовое, обязательное;

Название специальности – текстовое, обязательное;

Шифр специальности – текстовое, маска ввода – 000000 (шесть цифр), обязательное;

Оценка, полученная на экзамене – числовое, условие на значение >0 AND <30 (при 30-балльной шкале);

Дата экзамена – дата/время.

Таким образом, все требуемые атрибуты по таблицам распределены и заданы все необходимые параметры. Теперь необходимо реализовать заданные в ER-модели связи.

2.2.3. Связывание таблиц

Как уже было сказано в подразд. 1.4, для установки связей необходимо определить ключевые поля. Вернемся к нашему примеру. В данной БД будут следующие связи: Абитуриенты–Специальности (M:M), Абитуриенты–Экзамены (1:M), Специальности–Экзамены (1:M).

Для реализации этих связей в таблицах на стороне «один» необходимо определить первичные ключи. Поскольку в таблице Абитуриенты все поля могут повторяться, имеет смысл добавить новое поле Номер абитуриента и определить его в качестве первичного ключа.

В таблице Специальности имеется уникальное поле Шифр специальности. Зададим по нему первичный ключ.

Таблица Экзамены является отношением и должна хранить в себе внешние ключи Номер абитуриента и Шифр специальности. Добавим соответствующие поля в таблицу Экзамены.

Теперь можно установить связи, соединив в схеме данных соответствующие пары первичных и внешних ключей (рис. 6).

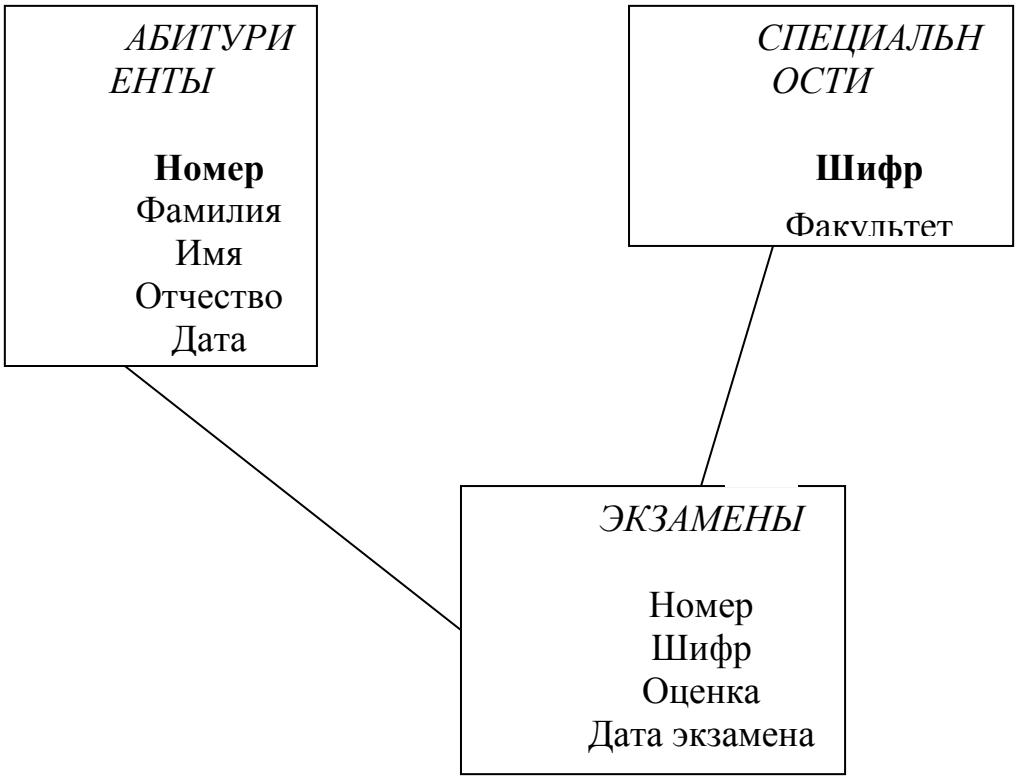


Рис. 6

Для соблюдения ссылочной целостности данных укажем на необходимость каскадного обеспечения целостности данных: каскадное обновление связанных полей и каскадное удаление связанных записей.

2.2.4. Разработка пользовательского интерфейса

Формы – это интерфейсный элемент, предназначенный для просмотра и ввода данных. Их цель – предоставить пользователю средства для заполнения только тех полей, которые ему заполнять положено. Одновременно с этим в форме можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и пр.) для автоматизации ввода. Источником формы может быть таблица (таблицы) или запрос (запросы). При этом происходит наследование свойств полей источника.

Формы позволяют пользователям вводить данные в таблицы БД без непосредственного доступа к самим таблицам. Это крайне важно по целому ряду причин.

Во-первых, для обеспечения конфиденциальности данных. Используя набор форм, можно эффективно разграничить доступ пользователей к данным в строгом соответствии с кругом персональных обязанностей. В банках,

например, одни сотрудники имеют доступ к таблицам данных о клиентах, другие – к их расчетным счетам, третьи – к таблицам активов банка. Если и есть специальные службы, имеющие доступ ко всем информационным ресурсам банка (в целях контроля и анализа), то они лишены средств для внесения изменений. Все сделано так, чтобы один человек не мог совершить фиктивную операцию, независимо от того, какую должность он занимает.

Во-вторых, в целях безопасности: снижается риск того, что неумелыми действиями они повредят данные в таблицах.

В-третьих, в целях удобства ввода: на экране будут отображаться не все записи, а только одна.

Если структура БД хорошо продумана, то исполнители, работающие с базой, должны навсегда забыть о том, что в базе есть таблицы, а еще лучше, если они об этом вообще ничего не знают. Таблицы – слишком ценные объекты базы, чтобы с ними имел дело кто-либо, кроме разработчика.

Преимущества форм раскрываются особенно наглядно, когда происходит ввод данных с заполненных бланков. В этом случае форму делают графическими средствами так, чтобы она повторяла оформление бланка. Это заметно упрощает работу пользователя, снижает его утомление и предотвращает появление опечаток.

Ну и, наконец, в формах можно производить вычисления, создавая новые вычисляемые поля, что невозможно сделать в таблицах.

Создаваемые формы должны соответствовать функциям пользователей, определяемых на этапе проектирования. Так, в данном примере необходимо осуществлять ввод и просмотр данных об абитуриентах, экзаменах и специальностях.

По выполняемым функциям выделяют следующие виды форм:

- для управления транзакциями (их функции: добавление новых записей в таблицу и изменение существующих);
- для доступа к данным (их функции: представление информации для анализа – диаграммы, формы со статистическими сведениями);
- управляющие, или кнопочные (их функции: управление доступом к объектам базы данных).

По дизайну и структуре формы следующие:

- «в столбец» (отображает все поля одной записи, она удобна для ввода и редактирования данных);
- ленточная (отображает одновременно группу записей, ее удобно использовать для оформления вывода данных для их сравнения и анализа);
- табличная (по внешнему виду ничем не отличается от таблицы, на которой она основана, однако позволяет выводить только необходимые поля, в том числе и вычисляемые).

Форма имеет три основных раздела: область заголовка, область данных и область примечания. Разделы заголовка и примечания имеют чисто оформительское назначение, их содержимое напрямую не связано с таблицей или запросом, на котором основана форма. В этих разделах имеет смысл располагать название формы или итоговые вычисления (например, количество

записей, суммарный доход, средний балл и т. д.). Раздел области данных имеет содержательное значение – в нем представлены элементы управления, при помощи которых выполняется отображение данных или их ввод.

Элементы управления бывают динамические и статические. Динамические элементы управления связаны с полями таблиц, а статические отображают свободные данные и хранятся внутри формы.

К динамическим элементам управления можно отнести:

- текстовые поля – служат для отображения данных из таблицы, ввода и редактирования данных, а также создания вычисляемых полей;
- флажки, переключатели, выключатели и их группы – служат для ввода и отображения данных из логических полей (например, отметка о зачете: да/нет);
- раскрывающиеся списки – позволяют осуществлять ввод посредством выбора значений из готового фиксированного списка;
- присоединенные рамки объектов – служат для отображения полей объектов OLE из таблицы (фотографий, рисунков, документов и т. д.)

К статическим элементам управления относят:

- надписи – произвольный текст;
- кнопки – служат для назначения им макросов или программ на VBA;
- вкладки – служат для создания многостраничных форм;
- линии, прямоугольники – для графического оформления формы;
- свободные рамки объектов – содержат внедренные в форму объекты (например, графическая эмблема компании). Для хранения рисунков можно использовать и элемент управления Рисунок.

Для создания вычисляемых полей в форме необходимо создать новое поле и разместить там соответствующее выражение. Для удобства создания формул можно использовать Построитель выражений.

Для анализа данных в формах применяют диаграммы. Диаграммы служат для наглядного графического представления информации, облегчая для пользователей сравнение и выявление тенденций и закономерностей в данных. При этом диаграмма может как сопровождать данные в форме, так и занимать всю форму. Для создания диаграмм используют приложение Microsoft Graph. Диаграммы могут быть глобальными (включающими все данные) или связанными с отдельной записью (отражающими данные только текущей записи и обновляющимися при переходе на другую запись).

Для создания кнопочных форм существует несколько путей, можно использовать:

1) элемент управления Кнопку для создания кнопок и назначения им макросов или процедур на языке VBA

2) мастер кнопок, позволяющий создать процедуру обработки события и назначить ее кнопке

3) диспетчер кнопочных форм для создания специальной таблицы (Switchboard Items), которая и хранит всю информацию о кнопочной форме.

Однако обычные формы, созданные на основе конкретной таблицы, не дают возможности одновременно видеть и вводить данные в связанные таблицы. Это крайне неудобно, так как не дает воспользоваться в полной мере всеми преимуществами установленных связей. Если необходимо обращаться к данным из разных таблиц, целесообразно создать **составные формы.**

При этом та форма, которая создана на основе таблицы со стороны «один», становится главной, а со стороны «много», – подчиненной. Встраивание подчиненной формы в главную можно осуществить в Конструкторе формы при помощи соответствующего элемента управления, методом Drag-and-Drop или при помощи Мастера форм.

В данном примере можно получить две составные формы: Студенты–Экзамены и Специальности–Экзамены.

Отчеты – это интерфейсный элемент, предназначенный для вывода данных на печать.

По своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т. п.)

В отличие от форм отчеты:

– предназначены только для печати и не предназначены для вывода в окне;

- позволяют изменять значения исходных данных;

- имеют два режима работы: предварительный просмотр и режим конструктора.

Приемы создания и редактирования отчетов те же, что и для форм. Элементы управления в данном случае выполняют функции элементов оформления, поскольку печатный отчет – не интерактивный объект, в отличие от электронных форм.

Структура готового отчета отличается от структуры формы только увеличенным количеством разделов. Кроме разделов заголовка, примечания и данных, отчет может содержать разделы верхнего и нижнего колонтитулов. Если отчет занимает более одной страницы, эти разделы необходимы для печати служебной информации, например номеров страниц. Чем больше страниц занимает отчет, тем важнее роль данных, выводимых на печать через

эти разделы. Если для каких то полей отчета применена группировка, количество разделов отчета увеличивается, поскольку оформление заголовков групп выполняется в отдельных разделах.

Группировка записей может осуществляться как по полному значению (каждой категории соответствует уникальное значение), так и по диапазону значений (по первым знакам, временно́му или числовому интервалам). В случае необходимости подведения итогов не по отчету в целом, а по группам выражение, включающее статистическую функцию, размещают не в примечании отчета, а в примечании группы.

2.2.5. Создание запросов

Запросы служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. При помощи запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию, а также преобразование данных по заданному алгоритму, создание новых таблиц, автоматическое наполнение таблиц данными, импортированными из других источников, вычисления в таблицах и многое другое.

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную результирующую таблицу. Основой запроса является критерий. Критерий – это совокупность условий, связанных логическими операторами (И, ИЛИ).

Для создания запросов используется язык SQL (Structured Query Language). Это универсальный компьютерный язык, применяемый для создания и модификации данных, а также управления ими в реляционных БД. SQL является информационно-логическим языком, а не языком программирования и основывается на реляционной алгебре.

Преимущества языка SQL:

- независимость от конкретной СУБД (несмотря на наличие диалектов и различий в синтаксисе, в большинстве своем тексты SQL-запросов могут быть достаточно легко перенесены из одной СУБД в другую);
- наличие стандартов и набора тестов (служит для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту, что способствует «стабилизации» языка);
- декларативность (при помощи SQL программист описывает только, какие данные нужно извлечь или модифицировать, а каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса).

Основные (элементарные) инструкции SQL:

- SELECT (возвращает результирующий набор записей);
- FROM (определяет источник);
- WHERE (определяет выражения для условий отбора);
- ORDER BY (определяет поля сортировки).

Язык SQL был задуман как средство работы конечного пользователя, но со временем он стал настолько сложным, что превратился в инструмент программиста.

В связи с этим появился и другой способ создания запросов. **QBE** (Query by example) – способ создания запросов к БД с использованием образцов в виде текстовой строки, названия документа или списка документов. Система QBE преобразует пользовательский ввод в формальный запрос к БД, что позволяет пользователю делать сложные запросы, освобождая от необходимости изучать более сложные языки запросов, таких как SQL. В Access QBE реализован в виде Конструктора запросов.

Рассмотрим основные разновидности запросов.

1. *Запрос на выборку.*

Позволяет выбрать данные из полей таблиц, на основе которых запрос сформирован, удовлетворяющие определенному критерию. Например, в случае написания условия *Between 170 And 200* извлекутся все записи, для которых значения указанного поля находятся в указанном интервале.

Если необходимо, чтобы данные, отобранные в результате работы запроса на выборку, были упорядочены по какому-либо полю или сразу по нескольким полям, применяют сортировку по возрастанию или по убыванию.

В нижней части бланка запроса по образцу имеется строка Вывод на экран. По умолчанию предполагается, что все поля, включенные в запрос, должны выводиться на экран, но это не всегда целесообразно. Например, бывают случаи, когда некое поле необходимо включить в запрос, например потому, что оно является полем сортировки, но в то же время нежелательно, чтобы пользователь видел его содержание. В таких случаях отображение содержимого на экране подавляют сбросом флажка Вывод на экран. Примером может быть запрос на вывод списка сотрудников предприятия, отсортированный по количеству дней, пропущенных по болезни. Он позволит каждому оценить свое положение в этом списке, но не позволит точно узнать, кто и сколько дней болел.

При необходимости в запросе на выборку можно создавать новые вычисляемые поля. Так, если в таблице существуют поля Цена и Количество проданных товаров, то мы можем определить Стоимость заказа, создав новое поле с выражением *Стоимость: [Цена]*[Количество]*.

Иногда возникает ситуация, когда критерий запроса постоянно нуждается в изменении при сохранении его структуры.

Например, необходимо регулярно узнавать успеваемость разных студентов. Поскольку пользователь сам изменять критерий запроса не может, разработчику придется создать запросы на все возможные фамилии, что невыполнимо.

В этом случае может помочь запрос с параметрами. При выполнении такого запроса критерий изменяется самим пользователем при каждом запуске.

Например, при записи в качестве условия *[Введите фамилию студента:]* пользователь увидит окно ввода с соответствующей надписью и введет интересующую его фамилию.

При необходимости над результатом выборки можно производить фильтрацию. Например, если необходимо определить пять самых высокооплачиваемых сотрудников, можно произвести в запросе сортировку по убыванию по полю Зарплата и указать на необходимость вывода пяти первых значений (заменив на панели инструментов Набор значений со слова *Все* на число 5).

Если необходимо определить промежуточные итоги по группам записей, можно создать запрос с группировкой. Для этого используется режим группировки и статистические функции *Sum* (сумма), *Avg* (среднее), *Count* (количество), *Max* (максимум), *Min* (минимум). Например, для того чтобы подсчитать количество лиц женского пола, необходимо сделать группировку по Полу и использовать функцию Count по фамилии.

Перекрестные запросы очень близки сводным таблицам в Excel и позволяют создавать результирующие таблицы на основе результатов расчетов, полученных при анализе группы таблиц. Результаты статистических расчетов группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй – в верхней строке.

В некоторых случаях необходимо сохранить результат запроса в БД для последующей обработки. При этом используются запросы на создание таблицы. При повторном выполнении такого запроса старая таблица будет удалена, а на ее месте появится новая таблица с тем же именем.

2. *Запрос на удаление.*

Позволяет автоматизировать процесс удаления записей из одной или нескольких таблиц в соответствии с критерием. Запрос можно использовать, например, при отчислении студентов по результатам сессии.

3. *Запрос на добавление.*

Добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц. Запрос может быть полезен, например, при переводе студентов, защитивших диплом из таблицы Студенты в таблицу Выпускники.

4. *Запрос на обновление.*

Данный тип запроса используется в тех случаях, когда необходимо выполнять отбор записей с последующим изменением для них значения определенного поля. Например, если нужно проиндексировать заработную плату сотрудников, можно делать это не вручную, меняя значения в поле Зарплата, а используя запрос на обновление.

В том случае, если необходимо извлечь данные сразу из нескольких таблиц используют **составные запросы.**

Они отличаются от простых только тем, что в качестве источника запроса используется не одна, а несколько таблиц.

При этом запросные связи, которые устанавливаются внутри данного запроса, могут отличаться от структурных связей в схеме данных.

При создании составных запросов необходимо правильно определять параметры объединения. Можно установить один из трех параметров:

1) объединение только тех записей, в которых связанные поля совпадают;

2) левое внешнее соединение (объединение всех записей со стороны «один» и только тех записей со стороны «много», в которых связанные поля совпадают);

3) правое внешнее соединение (объединение всех записей со стороны «много» и только тех записей со стороны «один», в которых связанные поля совпадают).

По умолчанию используется первый тип объединения.

Приведем пример. Пусть запросу необходимо извлечь информацию о фамилии, имени, отчестве и дате рождения абитуриента, а также об оценке, полученной им на экзамене. Добавив все требуемые поля в запрос, получим необходимую информацию. Однако если абитуриент подал документы, а экзамены еще не сдавал, то запрос не даст результата. Это объясняется тем, что в таблице Экзамены нет соответствующего внешнего ключа. Изменив параметр объединения на «левое внешнее», увидим всю информацию об абитуриенте, а вместо оценки – пустую ячейку.

Как уже было сказано ранее, для ускорения выполнения запросов рекомендуется использовать процедуру индексации полей, в которых часто происходит поиск. Записи в таблице представляют собой последовательную структуру и хранятся в том порядке, в каком осуществлялся ввод. В этом случае при выполнении запроса необходимо осуществлять поиск последовательным перебором всех значений, что занимает много времени.

Проблему можно решить, отсортировав все записи таблицы и перезаписав их в новом порядке. Но данную операцию пришлось бы делать каждый раз при добавлении и изменении записей в таблице, что значительно замедлило бы обработку информации.

Основной принцип индексации состоит в том, что от базовых таблиц никакой упорядоченности не требуется. Все записи в таблицы вносятся только в естественном порядке по мере их поступления, то есть в неупорядоченном виде. А для ускорения поиска используются индексные файлы. Индексный файл – это специальная служебная таблица. Она содержит значения атрибута, для которого создается индекс, и хранит ссылки на строки, в которых указано данное значение. Таким образом, получаем произвольный доступ ко всем записям (переходим от структуры последовательности к массивам).

Часто для ускорения процесса поиска той или иной записи необходимо создавать индекс на основании не одного, а нескольких полей таблицы одновременно, при этом индекс называется **составным.**

2.2.6. Использование средств автоматизации работы БД

В MS Access используются два инструмента для автоматизации работы с БД – макросы и модули. Эти категории объектов предназначены как для автоматизации повторяющихся операций при работе с системой управления базами данных, так и для создания новых функций путем программирования.

Макросы. В СУБД Microsoft Access макросы состоят из последовательности внутренних команд (макрокоманд) СУБД и являются одним из средств автоматизации работы с базой.

Макросы могут быть полезны для автоматизации часто выполняемых задач. Например, при нажатии пользователем кнопки можно запустить макрос, который распечатает отчет или откроет форму. В отличие от Word и Excel, в Access макросы создаются путем выбора последовательности стандартных макрокоманд. Количество возможных макрокоманд невелико, около 50. В основном это открытие, переход, запуск.

Макрос может быть как собственно макросом, состоящим из последовательности макрокоманд, так и группой макросов. Если макросов много, объединение родственных макросов в группы может упростить управление базой данных.

В некоторых случаях требуется выполнять макрокоманду или серию макрокоманд только при выполнении некоторых условий. Например, если в макросе проверяется соответствие данных в форме условиям на значение, то для одних значений может потребоваться вывести одно сообщение, а для других значений – другое сообщение. В подобных случаях условия позволяют определить порядок передачи управления между макрокомандами в макросе.

Модули. Модуль – это набор описаний, инструкций и процедур на языке VBA (Visual Basic for Application), собранных в одну программную единицу и сохраненных под общим именем. Это одно из средств, при помощи которых разработчик базы может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, а также уровень ее защищенности.

Существуют два основных типа модулей – модули класса и стандартные модули.

К модулям класса относят модули форм и отчетов, которые связаны с определенной формой или отчетом. Они часто содержат процедуры обработки событий, запускаемые в ответ на событие в форме или отчете. Процедуры обработки событий используются для управления поведением формы или отчета и их откликом на события, такие как нажатие кнопки.

В стандартных модулях содержатся общие процедуры, не связанные ни с каким объектом, а также часто используемые процедуры, которые могут быть запущены из любого окна БД. Основное различие между стандартным модулем и модулем класса, не связанным с конкретным объектом, заключается в области определения и времени жизни. Значение любой переменной или константы, определенной или существующей в модуле класса, не связанном с конкретным объектом, доступно только во время выполнения этой программы и только из этого объекта.

Таким образом, создание пользовательского интерфейса является одним из важнейших этапов создания БД. Если интерфейс легко осваивается персоналом, прост в использовании, интуитивно понятен и устойчив к ошибкам, то пользователи легко научатся извлекать пользу из представленной в нем информации. В то же время если интерфейс лишен указанных качеств, то работа с такой системой неизбежно будет сопровождаться теми или иными проблемами.

После создания всех объектов происходит наполнение и тестирование БД. Затем она переходит в стадию эксплуатации.