Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

004
Р 99
1214011232291 НБ УО "ПГУ"

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методический комплекс
для студентов экономических специальностей

В трех частях

Часть 3

С. Е. Рясова

Технологии баз данных и знаний

В двух книгах

Книга первая

Новополоцк
2017

УДК 004.3(075.8)
ББК 32.97я73
К63

Рекомендовано к изданию методической комиссией
финансово-экономического факультета
в качестве учебно-методического комплекса (протокол №12 от 28.12.2015 г.)

РЕЦЕНЗЕНТЫ:

зам. начальника отдела АСУ
(по программированию и эксплуатации программного обеспечения)
завода «Полимир» ОАО «Нафтан» И. Н. САС,
старший преподаватель кафедры технологии и методики преподавания
УО «ПГУ» Т. М. ЮПАТОВА

К63 **Компьютерные информационные технологии** : учеб.-метод. комплекс для студентов экон. специальностей : в 3 ч. / С. Е. Рясова. – Новополоцк : Полоцкий государственный университет, 2017. – Ч. 3 : Технологии баз данных и знаний : в 2 кн. Кн. 1. – 256 с.

ISBN 978-985-531-575-0.

В 2012 году вышла первая часть в 2-х книгах учебно-методического комплекса, автор С. Е. Рясова. В 2016 году – вторая часть, авторы С. Е. Рясова и Д. А. Оськин.

Приведены темы изучаемого курса, объем в часах лекционных и лабораторных занятий. Представлены методические указания и задания к лабораторным работам, сборник тестов по темам курса, вопросы к экзамену.

Предназначен для студентов экономических специальностей вуза.

УДК 004.3(075.8)
ББК 32.97я73

ISBN 978-985-531-575-0 (ч. 3, кн. 1)
ISBN 978-985-531-312-1

© Рясова С. Е., 2017
© Полоцкий государственный университет, 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В соответствии с учебной программой дисциплины «Компьютерные информационные технологии» для экономических специальностей настоящий учебно-методический комплекс включает в себя раздел «Технологии баз данных и знаний».

Материал, представленный в УМК, поможет студентам изучить концепции моделирования данных, принципы организации баз данных и их проектирования, назначение, архитектуру и функциональные возможности современных систем управления базами данных (СУБД), возможности языка баз данных SQL, технологии обработки многопользовательских баз данных, овладеть методами и средствами работы с базами данных.

Целью изучения раздела «Технологии баз данных и знаний» является получение знаний о современных технологиях организации, хранения и обработки данных, приобретение навыков по созданию баз данных и работе с ними.

Изучение материала этого раздела основано на использовании знаний, умений и навыков, полученных студентами после усвоения материала раздела «Техническое и программное обеспечение информационных технологий». Знания, умения и навыки, приобретенные в ходе изучения материала учебно-методического комплекса, позволят студентам эффективно использовать современное прикладное программное обеспечение в процессе решения учебных, научных и практических задач.

ЛЕКЦИОННЫЙ КУРС

1. Экономическая информация в автоматизированных информационных системах

1.1. Понятие экономической информации, ее виды, особенности, структурные единицы

Экономическая информация – это совокупность различных сведений экономического характера, используемых для планирования, учета, контроля, анализа и управления народным хозяйством и его звеньями. Экономическая информация включает сведения о трудовых, материальных и денежных ресурсах и деятельности экономических объектов (предприятий, организаций, банков, фирм и т. д.) на определенный момент времени. Эти сведения представляются натуральными и стоимостными показателями.

Экономическую информацию, циркулирующую в любом экономическом объекте, можно классифицировать **по разным признакам**:

– *функциям управления* – учетная, плановая, статистическая, оперативного управления и др.;

– *месту возникновения* – внутренняя и внешняя;

– *стадиям образования* – первичная и вторичная;

– *способу представления* – цифровая, алфавитно-цифровая, графическая;

– *стабильности* – переменная, условно-постоянная, постоянная;

– *полноте* – недостаточная, достаточная, избыточная;

– *истинности* – достоверная, недостоверная;

– *временному периоду возникновения* – периодическая и непериодическая.

Особенности экономической информации:

– многообразие источников возникновения и потребителей;

– объемность. Экономическая информация представляется для автоматизированной обработки в виде больших массивов документов;

– мир экономической информации – это прежде всего мир цифр, однако высок удельный вес и алфавитно-цифровой информации;

– при обработке преобладают арифметические операции, но значительный удельный вес и логических операций.

Структурные единицы экономической информации:

– реквизиты;

– показатели;

– документы;

– массивы.

Реквизиты выражают определенные свойства объекта и подразделяются на *реквизиты-признаки* и *реквизиты-основания*.

Реквизит-признак (РП) характеризует качественные свойства объекта (например, ФИО исполнителя, наименования работ, изделий, операций и т. д.).

Реквизит-основание (РО) дает количественную характеристику, выраженную в определенных единицах (например, количество изделий – в штуках, цена продукта – в рублях и т. д.).

Реквизиты имеют *наименования* и *значения*. Область значений описывается *форматом*. *Формат* определяет тип и максимальную длину значений. Тип может быть числовым, символьным и логическим. Для записи формата используются определенные символы. Например, 9 в формате означает позицию десятичной цифры, А – буквы, Х – любого символа, В – двоичной цифры (0 или 1), точка (.) – десятичной точки.

Таблица 1.1

Примеры составления и записи форматов

Код производителя	Продукт	Наличие нитратов в продукте
20	Картофель	0
104	Арбуз	1
5	Дыня	0
9(3)	А (9)	В

Реквизит-основание и логически связанные с ним реквизиты-признаки, имеющие экономический смысл, образуют *показатель*.

Таблица 1.2

Пример показателя

Цех	ФИО	Количество изделий, шт.
($РП_1$)	($РП_2$)	(РО)
7	Иванов И.И.	50

Здесь показатель – это «Количество изделий, изготовленное работником Ивановым И.И. цеха 7 – 50 шт.».

На основе показателей строятся *документы*. *Документ* – материальный объект, содержащий информацию, оформленную в установленном порядке, и имеющий в соответствии с действующим законодательством правовое значение. Экономические объекты широко применяют различные

документы (платежные поручения, акты, сводки, ведомости и т. д.) для отражения своей деятельности.

Совокупность документов, объединенных по определенному признаку, образует *массив*. Пример массива – множество финансовых отчетов предприятий некоторой отрасли.

1.2. Информационные системы, их классификация, информационное обеспечение

Высокий динамизм процессов в науке, технике, производстве приводит к усложнению производственно-хозяйственных связей и, в свою очередь, к росту объема информации, что затрудняет принятие управленческих решений, ведет к дублированию научных тем, экспериментов, технологий.

Возникает необходимость качественно нового подхода к работе с информацией. Разрабатываются автоматизированные информационные системы (АИС).

Система в широком смысле слова – это совокупность объектов и отношений между ними, образующая единое целое. Системе свойственны:

– делимость (система состоит из ряда элементов, отвечающих конкретным целям и задачам);

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

– целостность (функционирование множества элементов подчинено единой цели);

– структурированность (она обусловлена наличием связей между элементами, которые распределены по уровням иерархии).

На любой стадии развития общество требует для своего управления предварительно подготовленной, систематизированной информации.

Управление – это процесс целенаправленного воздействия на объект или систему, организующий функционирование объекта или системы по заданной программе. Систему, реализующую функции управления, называют *системой управления*. *Кибернетика* (наука об управлении) представляет эту систему как совокупность объекта управления и субъекта управления (управленческого аппарата). Управление связано с обменом информацией между компонентами системы, а также системы с окружающей средой.

Информационная система – это система информационного обслуживания работников управленческого аппарата, выполняющая технологические функции по сбору, накоплению, хранению и обработке информации. *Экономическая информационная система* (ЭИС) – система, функционирование

которой во времени заключается в сборе, обработке и распространении информации о деятельности некоторого экономического объекта. Важнейшие функции ЭИС – учет, анализ, контроль, регулирование, прогнозирование и планирование экономических процессов.

По уровню применения и административному делению различают ЭИС предприятия, района, города, области, государства. По сфере применения выделяют ЭИС: банковские, налоговые, страховые, статистические, бухгалтерские, фондового рынка, финансовые и др.

Для ЭИС соблюдаются следующие **принципы построения и функционирования**:

– эффективность – ЭИС должны обеспечивать функционирование объекта в соответствии с заданной целью;

– регламентность – большая часть информации в ЭИС поступает и обрабатывается по расписанию, со строгой периодичностью;

– самоконтроль – в ЭИС осуществляется непрерывная работа по обнаружению и исправлению ошибок в данных и процессах их обработки;

– интегральность – в ЭИС однократный ввод информации и ее многократное, многоцелевое использование;

– адаптивность – это способность ЭИС изменять свою структуру и закон поведения для достижения оптимального результата при изменяющихся внешних условиях.

Возрастание объемов информации в сфере управления, усложнение ее обработки повлекло за собой внедрение автоматизации сначала на отдельных операциях обработки информации, а затем расширение их применения. В результате создаются автоматизированные информационные системы.

Автоматизированная информационная система (АИС) – совокупность информации, экономико-математических методов и моделей, технических, программных, технологических средств, а также специалистов, предназначенная для обработки информации и принятия управленческих решений. Создание АИС способствует повышению эффективности функционирования экономического объекта и росту качества управления. Важнейший элемент АИС – информационное обеспечение. *Информационное обеспечение* предназначено для отражения информации, характеризующей состояние управляемого объекта, и является основой для принятия управленческих решений. Оно включает:

– системы показателей, описывающих деятельность экономического объекта;

– системы классификации и кодирования информации;

– документацию для отражения показателей;

– информационную базу.

Информационная база включает потоки внутренней и внешней информации, хранящейся на своих носителях. На ее основе функционирует АИС.

Примечание. Внутренняя информация возникает в самой системе и отражает финансово-хозяйственное состояние экономического объекта в различные временные интервалы.

Примечание. Внешняя информация касается состояния рынка и конкурентов, процентных ставок и цен, налоговой политики и политической ситуации.

1.3. Внемашинная организация экономической информации

Информационная база АИС состоит из двух взаимосвязанных частей: внемашинной и внутримашинной. К *внемашинной* относится та часть экономической информации, которая обслуживает систему управления в виде, воспринимаемом человеком без каких-либо технических средств. *Внутримашинная* содержится на машинных носителях и может состоять из отдельных независимых файлов или представлять собой базу данных.

Внемашинная часть представлена различными документами. Документы классифицируют по следующим признакам:

– сфере деятельности – на плановые, учетные, статистические, банковские, финансовые, бухгалтерские и др.;

– отношению к объекту управления – на входящие, исходящие, промежуточные, архивные;

– содержанию хозяйственных операций – на материальные, денежные, расчетные;

– назначению – на распорядительные, исполнительные, комбинированные;

– способу использования – на разовые и накопительные;

– способу заполнения – на заполняемые вручную или при помощи технических средств.

Развитие АИС потребовало унификации и стандартизации документов, по требованиям которых необходимо выделять:

– *заголовочную часть,* которая включает: наименование объекта, характеристику документа (индекс), наименование документа, зону для проставления кодов, постоянных реквизитов-признаков;

– *содержательную часть* в виде таблицы, где располагаются показатели;

– *оформляющую часть,* в которой содержатся подписи юридических лиц, ответственных за правильность его составления, а также дата составления.

Для представления информации, содержащейся в документах, в форме, удобной для ввода и обработки данных с помощью компьютеров, используются классификация и кодирование информации.

Классификация – это распределение множества объектов на подмножества в соответствии с установленными признаками сходства или различия. Признак сходства или различия, положенный в основу классификации, называется ее *основанием*. Совокупность правил классификации и результат классификации называется *системой классификации*. Существует две системы классификации – *иерархическая* и *фасетная*.

В *фасетной* системе заданное множество объектов делится на группировки одновременно по нескольким независимым признакам (фасетам).

Пример. Классификация промышленной продукции «Обувь» по фасетной системе:
– по материалу – резиновая, кожаная, матерчатая и др.;
– по качеству – модельная, рядовая;
– по половозрастному признаку – мужская, женская, детская.

Иерархическая система применяется в случае, когда какое-либо множество объектов подразделяется на классы, подклассы, группы последовательно по взаимоподчиненным основаниям.

Пример. Иерархическая система классификации для информационного объекта «Факультет», которая классифицирует информацию обо всех студентах по следующим классификационным признакам:
– факультет, на котором учится студент;
– возрастной состав студентов;
– пол студента;
– для женщин – наличие детей (рис. 1.1).

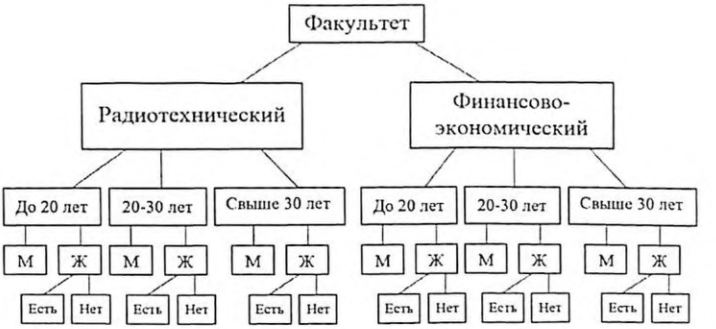


Рис. 1.1. Пример иерархической системы классификации

Кодирование информации – это образование и присвоение кодового обозначения объекту классификации, признаку классификации и/или классификационной группировке. Коды позволяют:

– уменьшить объем информации, вводимой в ЭВМ;

– облегчить запись на машинные носители, поиск и сортировку;

– обеспечить наглядность выходных документов.

Коды классифицируют по следующим признакам:

– форме представления – на цифровые и алфавитно-цифровые;

– длине – на однозначные и многозначные;

– методу образования – на порядковые, серийно-порядковые, разрядные, комбинированные.

В случае *порядкового метода* кодирования объектам присваиваются порядковые номера, начиная с единицы; *серийно-порядкового* – выделяется серия номеров, а внутри серии – присваиваются порядковые номера (пример – нумерация комнат на этаже). *Разрядный метод* применяется для кодирования объектов, определяемых несколькими соподчиненными признаками.

Каждому признаку классификации отводится определенное число разрядов. Классификационные группировки по младшим признакам кодируются в зависимости от кода старшего признака.

Пример. Необходимо составить код студента по разрядному методу. Студент обучается в вузе, где: 2 отделения (дневное и заочное), 5 факультетов; на дневном отделении – 4 курса, на заочном – 5 курсов; на каждом курсе – определенное количество групп (не более 6); в группах – не более 35 студентов. Тогда структура кода студента такова:

Код отделения	Код факультета	Код курса	Код группы	Порядковый номер студента в группе
X	*X*	*X*	*X*	*XX*

Длина кода студента – 6 разрядов.

Коды проставляются согласно классификаторам. *Классификаторы* – систематизированные своды наименований объектов, признаков классификации и их кодовых обозначений. Они используются:

– для ручного проставления кодов в документах. В этом случае они оформляются в виде справочников и используются экономистами для подготовки первичных документов к компьютерной обработке;

– для хранения на машинных носителях. Это позволяет автоматически декодировать информацию и формировать необходимые тексты в выходных документах.

Классификаторы подразделяются на:

– *общегосударственные* – разрабатываются централизованно и являются едиными для всей страны. Например, классификатор отраслей народного хозяйства ОКОНХ;

– *отраслевые* – едины для какой-то отрасли. Например, классификатор видов оплат и удержаний из заработной платы в пищевой промышленности;

– *локальные* – характерны для данного экономического объекта. Например, классификатор структурных подразделений предприятия.

1.4. Внутримашинная обработка экономической информации

1.4.1. Файловая организация данных, ее недостатки

В первые годы автоматизированной обработки информации, в 50-х – начале 60-х годов XX века, использовалась *файловая организация данных*.

Данные хранились в файлах последовательного доступа. Это заставляло прикладную программу обрабатывать файл целиком, когда необходимо было обратиться к определенной записи, что, конечно же, существенно замедляло скорость обработки данных. С появлением в 60-е годы устройств прямого доступа к данным – магнитных дисков – появилась возможность напрямую обратиться к нужной записи. Однако и это не дало существенного повышения скорости обработки и достоверности данных.

К недостаткам файловой организации данных также относятся:

– структура записей в файле задается в программе (приложении), которая работает с этим файлом;

– при изменении структуры файла необходимо изменять программу (приложение), т. е. наблюдается сильная зависимость программы от данных;

– если с файлом работают несколько приложений, то необходимо менять все приложения;

– невозможность нескольким пользователям изменить одновременно содержание файла, т. е. следующий пользователь может изменить файл, только если предыдущий закончил изменения и закрыл этот файл.

Эти недостатки файловой организации данных обусловили появление баз данных (БД), которые позволяют обеспечивать более эффективный доступ к данным и их обработку.

1.5. Понятие базы данных. Преимущества баз данных

База данных (БД) – именованная совокупность данных, отображающая состояние объектов, их свойства и взаимоотношения в некоторой предметной области.

Объектом может быть предмет, вещество, событие, лицо, явление, абстрактное понятие, т. е. все то, что может характеризоваться набором значений некоторой совокупности атрибутов.

Атрибут – это информационное отображение свойства объекта.

Пример. Объект «книга» характеризуется атрибутами: «наименование», «авторы», «количество страниц», «тираж», «цена» и др.

Предметная область – часть реального мира, которая описывается и моделируется с помощью БД.

Преимущества использования БД заключаются в следующем:

– возможность расширения и модификации данных;

– возможность обеспечения независимости данных в БД от программ, их обрабатывающих;

– возможность вести быстрый поиск необходимых данных по запросам пользователя;

– возможность обеспечения защиты секретных данных от постороннего вмешательства;

– возможность обеспечения целостности данных и др.

БД можно рассматривать как информационную модель объекта, от обоснованности, точности и достоверности которой во многом зависит эффективность управления объектом.

В БД информация хранится централизованно. Многие пользователи могут иметь возможность доступа, просмотра и изменения данных – все в одно и то же время, при этом пользуясь самой последней версией информации. Централизованное хранение позволяет легче изменять и согласовывать данные, экономить дисковое пространство.

Создание БД представляет трудоемкий процесс, требующий определенной квалификации. При разработке БД надо учитывать следующие требования:

1) многократное использование данных;

2) быстрый поиск и получение информации по запросам пользователей;

3) простоту обновления данных;

4) уменьшение излишней избыточности данных;

5) отсутствие дублирования данных в различных компонентах БД, обеспечивающее однократный ввод данных;

6) защиту данных от несанкционированного доступа, от искажения и уничтожения;

7) целостность БД – это требование полноты, непротиворечивости и сохранности данных.

1.6. Приложения БД. Компоненты БД

1.6.1. Приложения БД

Приложения базы данных включают такие объекты для работы с базой данных, как *формы*, *отчеты*, *Web-страницы* и *прикладные программы*. Формы, отчеты и Web-страницы можно создавать с помощью средств, поставляемых в комплекте с СУБД (например, в СУБД Access имеются средства конструирования форм и отчетов, называемые *элементами управления*). Прикладные программы должны быть написаны либо на входном языке СУБД (например, модули в Access), либо на одном из стандартных языков программирования и затем с помощью СУБД соединены с базой данных.

Дадим краткую характеристику объектов приложений баз данных, т. к. далее, на примере СУБД Access, мы рассмотрим их более детально.

Формы являются основным средством создания диалогового интерфейса приложения пользователя. Формы могут служить удобным средством для экранного представления данных, использоваться для ввода данных, а также для создания панелей управления в приложениях.

Web-страницы используются для просмотра, редактирования, обновления, удаления, отбора, группировки и сортировки изменяющихся данных базы данных в Microsoft Internet Explorer.

1.6.2. Компоненты БД

Любая база данных состоит из четырех основных компонентов:

- данные пользователя;
- метаданные;
- индексы;
- метаданные приложений.

Данные пользователя в большинстве современных баз данных представляются в виде набора таблиц, состоящих из строк (записей) и столбцов (полей).

Метаданные представляют собой описание структуры базы данных с помощью так называемых системных таблиц.

Индексы являются средством ускорения операций поиска необходимой информации в базах данных, а также используются при извлечении, модификации и сортировке данных.

Метаданные приложений описывают структуру и формат пользовательских форм, отчетов и других компонентов приложений базы данных.

1.7. Сверхбольшие базы данных

За историю существования человечества накопилось огромное количество информации, причем сегодня ее объемы продолжают увеличиваться. Обычные базы данных уже не справляются со всем объемом существующей информации. Следовательно, разрабатываются и применяются *сверхбольшие базы данных* (СБД).

Сверхбольшая база данных (англ. Very Large Database, VLDB) – это база данных, которая занимает чрезвычайно большой объем на устройстве физического хранения. Термин подразумевает максимально возможные объемы БД, которые определяются последними достижениями в технологиях физического хранения данных и в технологиях программного оперирования данными.

Количественное определение понятия «чрезвычайно большой объем» меняется во времени; в настоящее время считается, что это объем, измеряемый по меньшей мере петабайтами. Для сравнения, в 2005 г. самыми крупными в мире считались базы данных с объемом хранилища порядка 100 терабайт.

Специалисты отмечают необходимость особых подходов к проектированию сверхбольших БД. Для их создания нередко выполняются специальные проекты с целью поиска таких системотехнических решений, которые позволили бы хоть как-то работать с такими большими объемами данных. Как правило, необходимы специальные решения для дисковой подсистемы, специальные версии операционной среды и специальные механизмы обращения системы управления базами данных к данным.

В качестве примеров сверхбольших баз данных можно назвать Yahoo Everest, Sloan Digital Sky Survey (SDSS), WLCG (Worldwide Large Hadron Collider Computing Grid), BaBar.

На современном этапе развития науки и техники разработка и функционирование СБД связаны с рядом проблем:

1) большие объемы информации создают трудности в координации работы разных подразделений и взаимодействии с внешними подрядчиками;

2) усложнение обеспечения эффективного последовательного и быстрого доступа к данным на всех фазах обработки;

3) недостаточная производительность СБД;

4) серьезная нехватка квалифицированных специалистов;

5) высокая стоимость внедрения механизмов аналитики больших данных.

Ведущие разработчики представляют свои рекомендации по решению этих проблем.

Словарь терминов

Автоматизированная информационная система – совокупность информации, экономико-математических методов и моделей, технических, программных, технологических средств, а также специалистов, предназначенная для обработки информации и принятия управленческих решений.

Атрибут – это информационное отображение свойства объекта.

База данных – именованная совокупность данных, отображающая состояние объектов, их свойства и взаимоотношения в некоторой предметной области.

Документ – материальный объект, содержащий информацию, оформленную в установленном порядке, и имеющий в соответствии с действующим законодательством правовое значение.

Информационная система – это система информационного обслуживания работников управленческого аппарата, выполняющая технологические функции по сбору, накоплению, хранению и обработке информации.

Информационное обеспечение – предназначено для отражения информации, характеризующей состояние управляемого объекта, и является основой для принятия управленческих решений.

Классификация – это распределение множества объектов на подмножества в соответствии с установленными признаками сходства или различия.

Кодирование – это образование и присвоение кодового обозначения объекту классификации, признаку классификации и/или классификационной группировке.

Показатель – реквизит-основание и логически связанные с ним реквизиты-признаки, имеющие экономический смысл.

Предметная область – часть реального мира, которая описывается и моделируется с помощью БД.

Реквизит-основание – дает количественную характеристику, выраженную в определенных единицах (например, количество изделий – в штуках, цена продукта – в рублях и т. д.).

Реквизит-признак – характеризует качественные свойства объекта.

Реквизиты – выражают определенные свойства объекта и подразделяются на реквизиты-признаки и реквизиты-основания.

Сверхбольшая база данных – база данных, которая занимает чрезвычайно большой объем на устройстве физического хранения.

Система – это совокупность объектов и отношений между ними, образующая единое целое.

Управление – это процесс целенаправленного воздействия на объект или систему, организующий функционирование объекта или системы по заданной программе.

Экономическая информационная система – система, функционирование которой во времени заключается в сборе, обработке и распространении информации о деятельности некоторого экономического объекта.

Экономическая информация – это совокупность различных сведений экономического характера, используемых для планирования, учета, контроля, анализа и управления народным хозяйством и его звеньями.

Вопросы и задания для самоконтроля

1. Что собой представляет экономическая информация?
2. Как экономическая информация классифицируется по функциям управления?
3. Каковы особенности экономической информации?
4. Назовите и охарактеризуйте структурные единицы экономической информации.
5. Чем описывается область значений реквизитов?
6. Что такое «документ», «массив документов»?
7. Что такое «система» и каковы ее свойства?
8. Что собой представляет процесс управления?
9. Что такое «информационная система»?
10. Какие функции выполняет экономическая информационная система»?
11. Какие принципы построения и функционирования соблюдаются для ЭИС?
12. Что собой представляет автоматизированная информационная система?
13. Что входит в состав информационного обеспечения АИС?
14. Каким образом классифицируются документы?
15. Какие части выделяют в документе?
16. Что собой представляет процесс классификации объектов?
17. Какие системы классификации существуют?
18. Назовите методы кодирования информации?
19. Как используются классификаторы?

2. Модели данных

Первоначально исследования в области БД были направлены на разработку способов структуризации данных. Набор принципов, определяющих организацию логической структуры хранения данных в базе, называется *моделью данных*.

Модель данных включает в себя три аспекта:

1) *аспект структуры*: методы описания типов и логических структур данных в базе данных. Аспект структуры определяет, что из себя представляет база данных логически;

2) *аспект манипуляции*: методы манипулирования данными. Аспект манипуляции определяет способы перехода между состояниями базы данных (т. е. способы модификации данных) и способы извлечения данных из базы данных;

3) *аспект целостности*: методы описания и поддержки целостности базы данных. Аспект целостности определяет средства описаний *корректных состояний* базы данных.

Каждая БД и СУБД строится на основе некоторой модели данных. Все СУБД, построенные на одной и той же модели данных, относят к одному типу. Например, основой реляционных СУБД является реляционная модель данных, сетевых СУБД – сетевая модель данных, иерархических СУБД – иерархическая модель данных и т. д.

К числу классических относятся следующие модели данных:

- иерархическая;
- сетевая;
- реляционная.

Кроме того, в последние годы появились и стали более активно внедряться на практике следующие модели данных:

- постреляционная;
- объектно-ориентированная;
- объектно-реляционная модель и др.

2.1. Иерархическая модель данных

Иерархическая модель представляет собой перевернутое *дерево (граф)*, из *корня* и *узлов* (элементов данных) которого исходят *ветви* (соответствующие связям элементов данных). На самом верхнем уровне только один узел – *корень*. Каждый элемент связан с одним или несколькими элементами на более низком уровне (порожденными элементами, потомками)

и только с одним элементом на более высоком уровне (родителем, предком), за исключением корня. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо. Пример иерархической организации данных схематично представлен на рисунке 2.1.

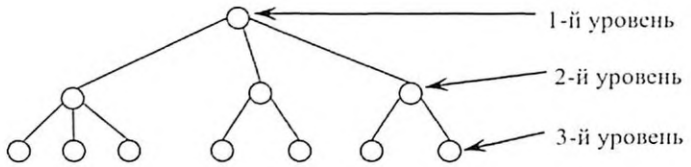


Рис. 2.1. Схематическое представление иерархической организации данных

Пример логической структуры иерархической БД приведен на рисунке 2.2.

Рис. 2.2. Пример логической структуры иерархической БД

Данные в базе с приведенной на рисунке 2.2 логической структурой могут выглядеть, например, как показано на рисунке 2.3.

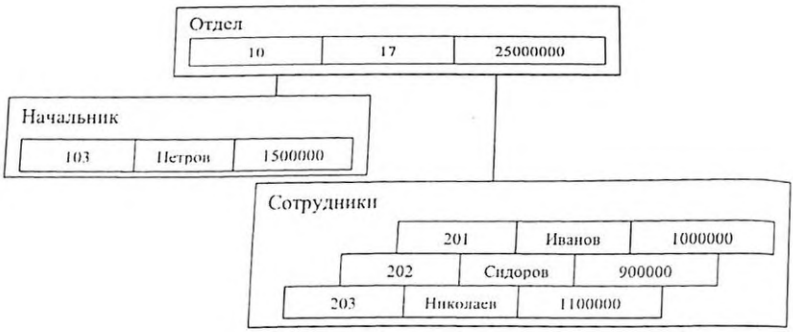


Рис. 2.3. Данные в иерархической базе

Основное правило контроля целостности данных для иерархических баз формулируется следующим образом: порожденный элемент (потомок) не может существовать без элемента более высокого уровня (родителя, предка), а у некоторых родителей может не быть порожденных элементов. Механизмы поддержания целостности данных между записями различных ветвей дерева отсутствуют.

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными:

– поиск указанного экземпляра БД (например, дерева со значением *10* в поле *Отд_номер*);

– переход от одного дерева к другому;

– переход от одной записи к другой внутри дерева (например, к следующей записи типа *Сотрудники*);

– вставка новой записи в указанную позицию;

– удаление текущей записи и т. д.

Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя. Ее применение ограничено, так как не любая предметная область может быть представлена с помощью этой модели.

Первые системы управления базами данных, появившиеся в середине 60-х годов XX в., позволяли работать с иерархическими базами данных. Наиболее известной была иерархическая система управления БД IMS фирмы IBM. Известны также другие системы: PC/Focus, Team-Up, Data Edge и др.

2.2. Сетевая модель данных

Сетевая модель представляет структуру, у которой один или несколько порожденных элементов имеют более одного исходного элемента. В сетевой структуре любой элемент может быть связан с любым другим элементом.

Примечание. Иерархическая модель данных является частным случаем сетевой.

Пример сетевой организации данных схематично представлен на рисунке 2.4.

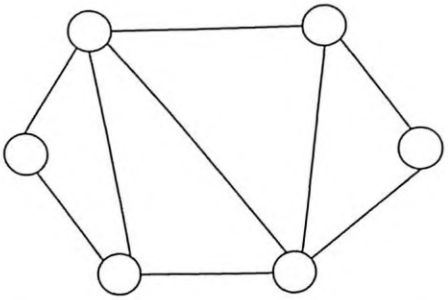


Рис. 2.4. Схематическое представление сетевой организации данных

Пример логической структуры сетевой БД приведен на рисунке 2.5.

Рис. 2.5. Пример логической структуры сетевой БД

Сетевая база данных состоит из наборов записей, которые связаны между собой так, что записи могут содержать явные ссылки на другие наборы записей. Тем самым наборы записей образуют сеть. Связи между записями могут быть произвольными, и эти связи явно присутствуют и хранятся в базе данных. Над данными в сетевой базе могут выполняться следующие операции:

- поиск записи в БД;
- переход от предка к первому потомку;
- переход от потомка к предку;
- создание новой записи;
- удаление текущей записи;
- обновление текущей записи;
- включение записи в связь;
- исключение записи из связи;
- изменение связей и т. д.

К известным сетевым системам управления базами данных относятся: DBMS, IDMS, TOTAL, VISTA, СЕТЬ, СЕТОР, КОМПАС и др.

В сравнении с иерархической моделью сетевая модель более универсальна, т. к. взаимосвязи большинства предметных областей имеют сетевой характер.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности данных вследствие допустимости установления произвольных связей между записями.

Сравнивая иерархические и сетевые базы данных, можно сделать следующий вывод. В целом иерархические и сетевые модели обеспечивают достаточно быстрый доступ к данным. Но поскольку в сетевых базах основная структура представления информации имеет форму сети, в которой каждая вершина (узел) может иметь связь с любой другой, то данные в сетевой базе более равноправны, чем в иерархической, т. к. доступ к информации может быть осуществлен, начиная с любого узла.

Однако следует отметить жесткость организации данных в иерархических и сетевых моделях. Доступ к информации осуществляется только в соответствии со связями, определенными при проектировании структуры конкретной базы данных. Базы данных с такими моделями сложно реорганизовывать. Недостатком этих моделей является и сложность механизма доступа к данным, а также необходимость на физическом уровне четко определять связи данных. А поскольку каждый элемент данных должен содержать ссылки на некоторые другие элементы, то для этого требуются значительные ресурсы памяти ЭВМ. Кроме того, для таких моделей характерна сложность реализации систем управления базами данных.

2.3. Реляционная модель данных

2.3.1. Общая характеристика реляционной модели данных

Реляционная модель данных (РМД) предложена сотрудником фирмы IBM Эдгаром Коддом в 1970 г. и основывается на понятии *отношение* (*relation*). Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица. Каждая строка таблицы имеет одинаковую структуру и состоит из полей.

Т. о., *реляционная модель данных* (РМД) некоторой предметной области представляет собой набор *отношений* (таблиц), изменяющихся во времени. При создании информационной системы совокупность отношений позволяет хранить данные об объектах предметной области и моделировать

связи между ними. Элементы РМД и формы их представления приведены в таблице 2.1.

Таблица 2.1

Элементы реляционной модели

Элемент реляционной модели	Форма представления
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значение поля в записи
Ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы

На рисунке 2.6 приведен пример представления отношения СОТРУДНИК.

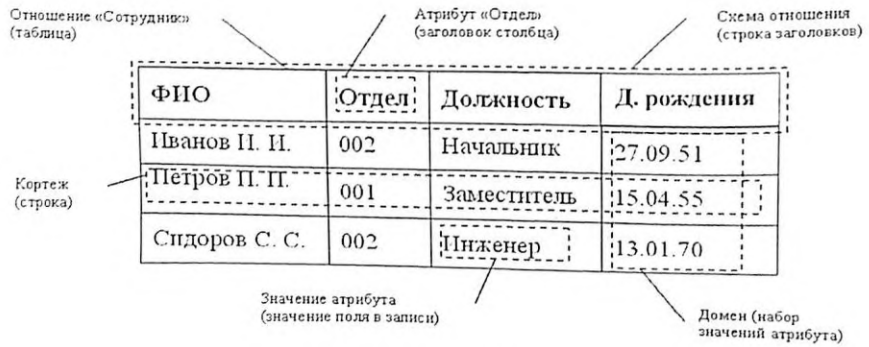


Рис. 2.6. Отношение «Сотрудник»

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные.

Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.

Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Кортеж представляет собой строку (запись) таблицы. В общем случае порядок кортежей в отношении не определен. Однако в реляционных СУБД для удобства кортежи все же упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому система автоматически сортирует кортежи по возрастанию или убыванию. Если пользователь не назначает атрибута упорядочения, система автоматически присваивает номер кортежам в порядке их ввода.

Отношение СОТРУДНИК содержит 3 кортежа. Каждый кортеж рассматриваемого отношения состоит из 4 элементов, каждый из которых выбирается из соответствующего домена.

Домен представляет собой множество всех возможных значений определенного атрибута отношения. Отношение СОТРУДНИК включает 4 домена. Первый домен содержит фамилии всех сотрудников, второй домен – номера всех отделов фирмы, третий домен – названия всех должностей, четвертый домен – даты рождения всех сотрудников. Каждый домен образует значения одного типа данных, например, числовые или символьные.

Примечание. В современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как «деньги»), специальных временных данных (дата, время, временной интервал).

Схема отношения (заголовок отношения) представляет собой список имен атрибутов. В частности, для приведенного примера схема отношения имеет вид СОТРУДНИК (ФИО, Отдел, Должность, Д_Рождения).

Первичным ключом отношения (*ключом отношения, ключевым атрибутом*) называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей. Например, в отношении СОТРУДНИК (ФИО, Отдел, Должность, Д_Рождения) ключевым является атрибут «ФИО». Ключ может быть *составным (сложным)*, т. е. состоять из нескольких атрибутов.

Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется тем, что в отношении нет повторяющихся кортежей, а это значит, что по крайней мере вся совокупность атрибутов обладает свойством однозначной идентификации кортежей отношения.

Ключи обычно используют для достижения следующих целей:

1) исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);

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

3) ускорения работы с кортежами отношения;

4) организации связывания таблиц (раздел 2.4).

Поскольку не всякой таблице можно поставить в соответствие отношение, приведем условия, выполнение которых позволяет считать таблицу отношением:

1) все строки таблицы должны быть уникальны, т. е. не может быть строк с одинаковыми первичными ключами;

2) имена столбцов таблицы должны быть различны, а значения их простыми, т. е. недопустима группа значений в одном столбце одной строки (вложенные таблицы);

3) все строки одной таблицы должны иметь одну структуру, соответствующую именам и типам столбцов;

4) порядок размещения строк в таблице может быть произвольным.

Достоинства реляционной модели данных:

- простота представления данных;
- удобство модификации отношений;
- минимальная избыточность данных, достигаемая при нормализации отношений (раздел 3.2).

Недостатки реляционной модели данных:

- невысокая скорость работы с данными;
- значительная фрагментация данных вследствие нормализации отношений, в то время как в большинстве задач необходимо объединение фрагментированных данных.

2.3.2. Логические связи между отношениями

При проектировании реальных БД информацию обычно размещают в нескольких таблицах. Таблицы при этом связаны семантикой информации. В реляционных БД для указания связей таблиц производят операцию их *связывания*.

Укажем преимущества, достигаемые в результате связывания таблиц:

- автоматический контроль целостности вводимых в базу данных в соответствии с установленными связями (выполняется многими СУБД). В конечном итоге это *повышает достоверность* хранимой в БД информации;
- установление связи между таблицами *облегчает доступ* к данным. Механизм: связывание таблиц при выполнении таких операций, как поиск, просмотр, редактирование, выборка и подготовка отчетов, обычно обеспечивает возможность обращения к произвольным полям связанных записей. Это уменьшает количество явных обращений к таблицам данных и число манипуляций в каждой из них;
- связи делают таблицы более информативными, чем они являются по отдельности. Они позволяют минимизировать избыточность данных в БД.

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

При связывании двух таблиц выделяют *основную* и *дополнительную* (подчиненную) таблицы.

Связь устанавливается посредством *ключей связи*, содержащих общую информацию для обоих отношений.

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

Существует четыре вида связи:

- один к одному (1:1);
- один ко многим (1:M);
- многие к одному (M:1);
- многие ко многим (M:M).

Связь вида 1:1. Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми. Поскольку значения в ключевых полях обеих таблиц не повторяются, обеспечивается взаимно-однозначное соответствие записей из этих таблиц. Сами таблицы, по сути, здесь становятся равноправными (рис. 2.7).

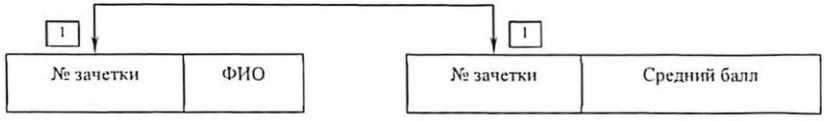


Рис. 2.7. Пример связи 1:1

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

Связь вида 1:M. Связь 1:M имеет место в случае, когда одной записи основной таблицы соответствует несколько записей дополнительной таблицы (рис. 2.8).

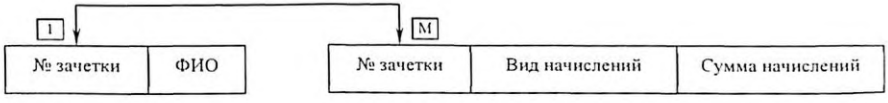


Рис. 2.8. Пример связи 1:M

Связь вида M:1. Связь M:1 имеет место в случае, когда одной или нескольким записям основной таблицы ставится в соответствие одна запись дополнительной таблицы (рис. 2.9).

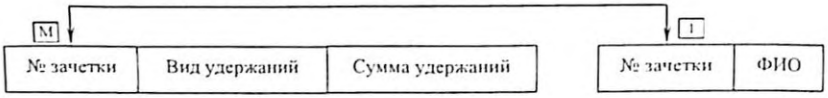


Рис. 2.9. Пример связи M:1

Связь вида M:M. Самый общий вид связи M:M возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей дополнительной таблицы (рис. 2.10).

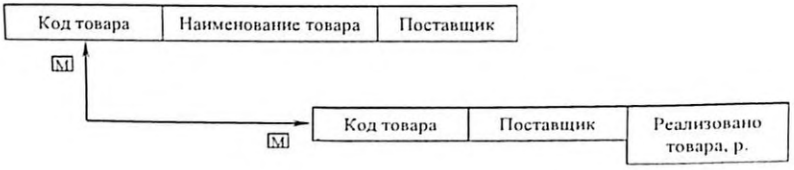


Рис. 2.10. Пример связи M:M

Каждый поставщик может поставлять несколько товаров; каждый товар может поставляться несколькими поставщиками.

Примечание. На практике в связь обычно вовлекается сразу несколько таблиц. При этом одна из таблиц может иметь различного рода связи с несколькими таблицами. В случаях, когда связанные таблицы, в свою очередь, имеют связи с другими таблицами, образуется иерархия или дерево связей.

Из перечисленных видов связи наиболее широко используется связь вида 1:M. Связь вида 1:1 можно считать частным случаем связи 1:M. Связь M:1, по сути, является «зеркальным отображением» связи 1:M. Оставшийся вид связи M:M характеризуется как слабый вид связи или даже как отсутствие связи.

2.3.3. Контроль целостности данных

Ввиду того, что связь вида 1:M используется наиболее широко, механизм контроля целостности данных будет рассмотрен применительно к этому виду связи.

Контроль целостности данных обычно означает анализ содержимого двух таблиц на соблюдение следующих правил:

– каждой записи основной таблицы соответствует ноль или более записей дополнительной таблицы;

– в дополнительной таблице нет записей, которые не имеют родительских записей в основной таблице;

– каждая запись дополнительной таблицы имеет только одну родительскую запись в основной таблице.

Опишем действие контроля целостности при манипулировании данными в таблицах. Рассмотрим три основные операции над данными двух таблиц:

– ввод новых записей;

– модификацию записей;

– удаление записей.

Ввод новых записей. При вводе новых записей возникает вопрос определения последовательности ввода записей в таблицы, такой, чтобы не допустить нарушение целостности данных. Исходя из приведенных правил, логичной является схема, при которой данные сначала вводятся в основную таблицу, а потом – в дополнительную.

В процессе заполнения основной таблицы контроль значений полей связи ведется как контроль обычного ключа, т. е. на совпадение со значениями тех же полей других записей. Заполнение полей связи дополнительной таблицы контролируется на предмет совпадения со значениями полей связи основной таблицы. Если вновь вводимое значение в поле связи дополнительной таблицы не совпадет ни с одним соответствующим значением в записях основной таблицы, то ввод такого значения должен блокироваться.

Модификация записей. Изменение содержимого полей связанных записей, не относящихся к полям связи, очевидно, происходит обычным образом. Нас будет интересовать механизм изменения полей связи.

При редактировании полей связи дополнительной таблицы очевидным требованием является то, чтобы новое значение поля связи совпадало с соответствующим значением какой-либо записи основной таблицы. Т. е. дополнительная запись может сменить родителя, но остаться без него не должна.

Редактирование поля связи основной таблицы должно подчиняться одному из следующих правил:

– редактировать записи, у которых нет подчиненных записей. Если есть подчиненные записи, то блокировать модификацию полей связи;

– изменения в полях связи основной таблицы мгновенно передавать во все поля связи всех записей дополнительной таблицы (каскадное обновление).

Удаление записей. В операциях удаления записей связанных таблиц большую свободу, очевидно, имеют записи дополнительной таблицы. Удаление их может происходить практически бесконтрольно.

Удаление записей основной таблицы должно подчиняться одному из следующих правил:

– удалять можно запись, которая не имеет подчиненных записей;

– запретить (блокировать) удаление записи при наличии подчиненных записей, либо удалять ее вместе со всеми подчиненными записями (каскадное удаление).

2.3.4. Достоинства и недостатки реляционной модели данных

Достоинства реляционных баз данных можно сформулировать следующим образом:

– упрощенная схема представления данных – в виде таблицы;

– простота инструментальных средств поддержки реляционной модели;

– оптимизация доступа к базе данных, поскольку системы сами выбирают наиболее эффективную последовательность действий;

– улучшение целостности и защиты, поскольку реляционная модель позволяет улучшить выражение требований целостности путем использования языка высокого уровня;

– возможности различных применений, в том числе и рассчитанных на неспециалистов в области программирования;

– обеспечение пользователя языками высокого уровня при работе с базой данных;

– обеспечение методологического подхода, поскольку главной целью модели базы данных является возможность описания реального мира, что проще всего осуществляется в реляционной модели.

Недостатки реляционной модели:

– жесткость структуры данных. Например, невозможно задать строку таблицы произвольной длины;

– сложность описания иерархических и сетевых связей.

В настоящее время многие известные системы управления базами данных используют именно реляционную модель представления данных – это dBase, FoxBase, FoxPro, Paradox, Oracle, Microsoft Access, Clarion, Clipper, Ingres и др.

2.4. Постреляционная модель данных

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

Примечание. *Приложение* или прикладная программа (*application*) – это программа или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи.

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

На примере информации о накладных и товарах для сравнения приведем представление одних и тех же данных с помощью реляционной и постреляционной моделей.

Пример. Для реляционной модели данных: таблица НАКЛАДНЫЕ содержит данные о номерах накладных (ИНВ№) и номерах покупателей (КЛИЕНТ№). В таблице НАКЛАДНЫЕ_ТОВАРЫ содержатся данные о каждой из накладных: номер накладной (ИНВ№), название товара (НАЗВ_ТОВАРА) и количество товара (КОЛИЧЕСТВО). Таблица НАКЛАДНЫЕ связана с таблицей НАКЛАДНЫЕ_ТОВАРЫ по полю ИНВ№. Для постреляционной модели данных: таблица НАКЛАДНЫЕ содержит всю информацию.

Как видно из рисунка 2.11, по сравнению с реляционной моделью в постреляционной модели данные хранятся более эффективно, а при обработке не требуется выполнять операцию соединения данных из двух таблиц.

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

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

НАКЛАДНЫЕ

ИНВ№	КЛИЕНТ№
0373	8723
8374	8232
7364	8723

НАКЛАДНЫЕ_ТОВАРЫ

ИНВ№	НАЗВ_ТОВАРА	КОЛИЧЕСТВО
0373	Сыр	3
0373	Рыба	2
8374	Лимонад	1
8374	Сок	6
8374	Печенье	2
7364	Йогурт	1

а)

НАКЛАДНЫЕ

ИНВ№	КЛИЕНТ№	НАЗВ_ТОВАРА	КОЛИЧЕСТВО
0373	8723	Сыр	3
		Рыба	2
8374	8232	Лимонад	1
		Сок	6
		Печенье	2
7364	8723	Йогурт	1

б)

Рис. 2.11. Структуры данных: а) реляционная модель, б) постреляционная модель

2.5. Объектно-ориентированная модель данных

Необходимо заметить, что общепринятого определения «объектно-ориентированной модели данных» (ООМД) не существует. Сейчас можно говорить лишь о некоем «объектном» подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются *объекты*.

Пример логической структуры объектно-ориентированной БД библиотечного дела приведен на рисунке 2.12.

Здесь объект типа БИБЛИОТЕКА является родительским для объектов-экземпляров классов АБОНЕНТ, КАТАЛОГ и ВЫДАЧА. Различные объекты типа КНИГА могут иметь одного или разных родителей. Объекты типа КНИГА, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств *isbn*, *удк*, *название* и *автор*.

Такая модель позволяет идентифицировать отдельные записи базы.

Поиск в объектно-ориентированной базе состоит в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в базе. Определяемый пользователем объект называют *объектом-целью*.

Базовыми понятиями этой модели являются следующие: *объекты*, *классы*, *методы*, *инкапсуляция*, *наследование*, *полиморфизм*. Эти понятия взяты из объектно-ориентированного программирования. В этой среде все состоит из объектов.

Объект обладает следующими свойствами:

- идентифицируется уникальным неизменным образом;
- принадлежит к определенному классу;
- может посылать сообщения другим объектам;
- имеет внутреннее состояние.

Таким образом, объектно-ориентированная база данных состоит из объектов, каждый из которых должен принадлежать к определенному классу, т. е. каждый объект – *экземпляр класса*. Объектно-ориентированная база данных состоит из *коллекции классов*. Структура и поведение объектов в объектной среде полностью определяются его классом. Класс, в свою очередь, является коллекцией объектов, при этом структура и поведение объектов одного класса одинаковы.

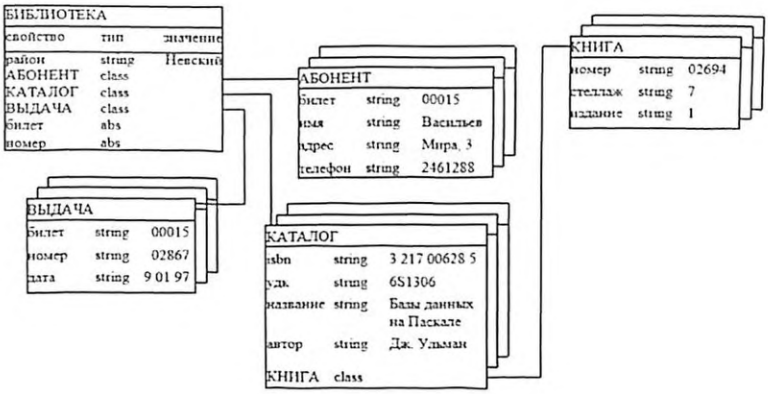


Рис. 2.12. Логическая структура БД библиотечного дела

Класс объекта состоит из его *интерфейса* и *закрытой области*.

Интерфейс класса – это то, что видно другим объектам. Он, в свою очередь, состоит из двух частей: *свойств класса* и *методов класса*. Аналогом свойств являются атрибуты отношений.

Пример. Клиент может иметь следующие свойства: номер, ФИО, адрес, телефон. К свойствам относятся также связи с другими объектами. Свойства сами могут быть объектами, что позволяет создавать составные объекты. Свойство ФИО может состоять из свойств: фамилия, имя, отчество.

Доступ к значениям свойств и манипулирование ими можно осуществлять только посредством *методов класса*. То есть поведение объекта задается с помощью методов его класса. Обычно они имеют форму операций и функций, которые могут содержать параметры. На уровне интерфейса видимым является только имя каждого метода и требуемые параметры. Методы служат для передачи объектам сообщений. Другими словами, метод представляет то, что, по мнению пользователя, должен делать объект. Например, клиент может сделать заказ, оплатить счет и т. п. Для каждого из этих видов деятельности должен быть соответствующий метод.

Закрытая область – это та часть определения класса, которая не видна другим объектам. Пользователю объекта предоставляется информация только о том, как работать с объектом при помощи его методов. Сама же работа объекта скрыта от пользователя.

Пример. Могут существовать дополнительные свойства с закрытыми значениями, а также скрытые связи и сообщения другим объектам.

Свойства объектов описываются некоторым стандартным типом (например, строковым – *string)*; значением свойства типа *string* является строка символов) или типом, конструируемым пользователем. Этот тип определяется словом *class*. Значением свойства типа *class* является объект, являющийся экземпляром соответствующего класса. Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в базе образуют связанную иерархию объектов.

Важным достоинством объектно-ориентированной базы является то, что пользователю не нужно знать о взаимодействии объектов: он просто обращается к конкретному объекту и использует конкретный метод. А то, что при этом осуществляется воздействие на другие объекты базы, скрыто от пользователя. Различные правила, руководящие использованием объектов, также могут быть скрыты от пользователя. Например, выбранный метод может, в свою очередь, обращаться к другим методам, например, методу проверки кредитоспособности выбранного клиента.

Чтобы определить класс объектов, нужно задать его свойства и методы, а также определить его взаимодействие с другими объектами. Понятие класса объекта во многом аналогично понятию типа. Поэтому при проектировании объектно-ориентированной базы данных нужно, прежде всего, осуществить процесс классификации, то есть выявить объекты с аналогичными свойствами и поведением и объединить их в классы.

Этих действий можно добиться и в реляционных базах. Но для этого надо создать специальные приложения, предоставляющие пользователю интерфейс, производящий определенные действия, основанные на работе других частей базы данных. При объектной ориентации подобная деятельность может быть частью определенного объекта, а не представлять собой отдельное приложение. Таким образом, используя объекты и методы, можно хранить и неоднократно использовать не только структуру объекта базы данных, но и его поведение.

Инкапсуляция. Каждый объект обладает некоторым внутренним состоянием (хранит внутри себя запись данных), а также набором методов – процедур, с помощью которых (и только таким образом) можно получить доступ к данным, определяющим внутреннее состояние объекта, или изменить их.

Таким образом, объекты можно рассматривать как самостоятельные сущности, отделенные от внешнего мира.

Доступ к объекту может осуществляться только через его интерфейс. Поведение объекта полностью определяется принадлежностью к конкретному классу.

Наследование подразумевает возможность создавать из классов объектов новые классы объектов, которые наследуют структуру и методы своих предков, добавляя к ним черты, отражающие их собственную индивидуальность. Наследование может быть простым (один предок) и множественным (несколько предков).

Полиморфизм подразумевает, что различные объекты могут по-разному реагировать на одинаковые внешние события в зависимости от того, как реализованы их методы.

Создание объектной модели начинается с *классификации* – выявления объектов с аналогичными свойствами и поведением и объединения их в классы. Например, в базе данных, диаграммы, можно начать классификацию с выделения объектов диаграмм, содержащих дату их создания. Процесс классификации позволяет выделить объекты с общими свойствами и методами. Однако, некоторые их свойства и методы различны. В этом случае производят *генерализацию* и *специализацию*.

Генерализация выявляет классы объектов с аналогичными свойствами и образует на основе этих свойств *абстрактный суперкласс*. Например, в базе данных, содержащей описание геометрических фигур, можно начать проектирование с выделения классов: треугольников, прямоугольников, окружностей, – а затем образовать из них абстрактный суперкласс *Фигуры*, состоящий из свойств, общих для всех фигур.

Специализация – процесс обратный генерализации. При использовании этих процессов создается иерархия классов. Иерархии указывают цепочку наследования.

Важным процессом в объектно-ориентированной базе является *агрегация*. С помощью агрегации классы объектов могут связываться друг с другом, образуя *класс агрегатов*.

Пример. Банковская база может содержать информацию о клиентах, счетах, филиалах, а также связи между ними. В объектно-ориентированной базе всю эту информацию можно инкапсулировать в одном агрегированном классе объектов.

Таким образом, создание объектно-ориентированной базы данных основано на процессах:

- классификации;
- генерализации;
- специализации;
- агрегации.

Эти процессы проводятся параллельно. Резюмируя все вышеизложенное, можно сказать следующее:

– объектно-ориентированная база данных – это попытка применить идеологию объектно-ориентированного программирования к технологии баз данных;

– объектно-ориентированная база данных состоит из объектов, причем каждый объект принадлежит к определенному классу;

– поведение объекта полностью определяется его принадлежностью к определенному классу;

– процесс проектирования объектно-ориентированной базы основан на выявлении классов объектов.

Основным достоинством объектно-ориентированной модели данных по сравнению с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель позволяет также идентифицировать отдельные записи в базе и определять функции их обработки. Учитывая эти достоинства, уже сегодня некоторые реляционные СУБД дополняют функциями, позволяющими воспользоваться преимуществами объектной технологии.

Основной недостаток объектно-ориентированной модели состоит в сложности понимания ее сути и низкой скорости выполнения запросов.

Объектно-ориентированная модель является очень перспективной в связи с распространением объектно-ориентированного подхода к разработке программных продуктов. На сегодняшний день ее распространение сдерживают два обстоятельства:

– отсутствие строгой математической модели объектно-ориентированной базы данных. Для реляционной модели такое строгое описание имеется;

– наличие огромного количества данных в имеющихся реляционных базах данных и существенные затраты на их конвертацию в объектно-ориентированную БД.

В силу этих обстоятельств внедрение объектно-ориентированного подхода в базы данных происходит эволюционно, без разрушения реляционной основы. На сегодняшний день многие СУБД позиционируются как объектно-реляционные. В их основе по-прежнему лежит реляционная модель, но она дополнена возможностью создания пользовательских типов столбцов с поддержкой принципов инкапсуляции и наследования.

В настоящее время ведется очень много экспериментальных и производственных работ в области объектно-ориентированных СУБД. Сегодня уже разработаны и успешно функционируют такие системы управления объектно-ориентированными БД как O2, ORION, GemStone и Iris.

2.6. Объектно-реляционная модель

В связи со значительным усложнением приложений появилась новая модель – *расширенная реляционная модель (Extended Relation Data Model – ERDM)*.

Эта модель включает в себя основные достоинства объектно-ориентированной модели и одновременно унаследовала простоту структуры реляционных моделей, и потому стала называться *объектно-реляционной моделью данных (ОРМД)*. В отличие от объектно-ориентированной модели объектно-реляционная модель основана на стратегии реляционной модели, в то время как ООМД основана на объектной стратегии. Исходя из этого, модель ОРМД наиболее приспособлена для бизнес-приложений, а модель ООМД используется в специальных инженерных и научных приложениях. Некоторые специалисты полагают, что в будущем произойдет слияние ООМД и ОРМД моделей.

Однако у объектно-реляционной и объектно-ориентированной моделей есть и ряд недостатков, основными из которых являются следующие:

– отсутствие унифицированной теории, которая есть в реляционных моделях;

– отсутствие формальной методологии проектирования баз данных, такой как нормализация в реляционных базах;

– отсутствие специальных средств создания запросов;

– отсутствие общих правил определения целостности.

2.7. Многомерная модель данных

По сравнению с реляционной моделью многомерная организация данных обладает более высокой наглядностью и информативностью. Многомерная модель данных используется, когда целью является анализ данных, а не выполнение транзакций (раздел 4.1), характерное для РМД. Технология многомерных баз данных – ключевой фактор интерактивного анализа больших массивов данных с целью поддержки принятия решения.

Многомерные модели рассматривают данные либо как *факты* с соответствующими численными параметрами, либо как *текстовые измерения*, которые характеризуют эти факты.

Пример. В розничной торговле покупка – это *факт*, объем покупки и стоимость – *параметры*, а тип приобретенного продукта, время и место его покупки – *измерения*. Запросы агрегируют значения параметров по всему диапазону измерения, и в итоге получают такие величины, как общий месячный объем продаж данного продукта.

Многомерные модели данных имеют три важных области применения, связанных с проблематикой анализа данных:

– в хранилищах данных, интегрирующих для анализа информацию из нескольких источников на предприятии;

– в системах оперативной аналитической обработки (OnLine Analytical Processing – OLAP), позволяющих оперативно получить ответы на запросы, охватывающие большие объемы данных в поисках общих тенденций;

– в приложениях добычи данных, служащих для выявления скрытых закономерностей или взаимосвязей в больших массивах данных.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными.

Для иллюстрации на рисунке 2.13 приведены реляционное и многомерное представления одних и тех же данных об объемах продаж автомобилей.

Модель	Месяц	Объем
«Жигули»	июнь	12
«Жигули»	июль	24
«Жигули»	август	5
«Москвич»	июнь	2
«Москвич»	июль	18
«Волга»	июль	19

а)

Модель	Июнь	Июль	Август
«Жигули»	12	24	5
«Москвич»	2	18	Нет
«Волга»	Нет	19	Нет

б)

Рис. 2.13. Представление данных: а) реляционное, б) многомерное

В многомерных моделях данные трактуют как *многомерные кубы*, что очень удобно для их анализа.

Примечание. В обычном обиходе этим термином обозначают фигуру с тремя измерениями, однако теоретически куб может иметь любое число измерений. На практике чаще всего кубы данных имеют от 4 до 12 измерений, т. к. современный инструментарий часто сталкивается с нехваткой производительности, когда так называемый *гиперкуб* имеет свыше 10-15 измерений.

Рассмотрим основные понятия многомерных моделей данных, к числу которых относятся *измерение* и *ячейка*.

Измерение (*Dimension*) – это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются *Дни*, *Месяцы*, *Кварталы* и *Годы*. В качестве географических измерений широко используются *Города*, *Районы*, *Регионы* и *Страны*. В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (*Cell*) или *показатель* – это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой. В зависимости от того, как формируются значения некоторой ячейки, обычно она может быть переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно) либо формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

В примере (рис. 2.13, б) каждое значение ячейки *Объем продаж* однозначно определяется комбинацией временного измерения (месяц продаж) и модели автомобиля. На практике зачастую требуется большее количество измерений.

Пример трехмерной модели данных приведен на рисунке 2.14.

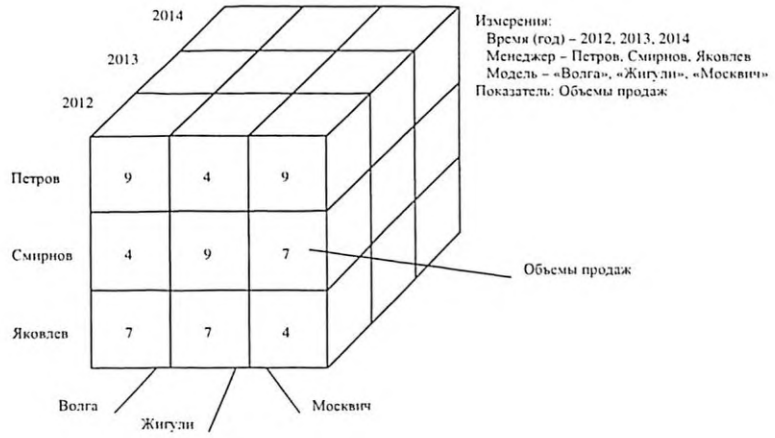


Рис. 2.14. Пример трехмерной модели данных

Если речь идет о многомерной модели с мерностью больше двух, то не обязательно визуально информация представляется в виде многомерных объектов (трех-, четырех- и более мерных гиперкубов). Пользователю и в этих случаях более удобно иметь дело с двухмерными таблицами или графиками. Данные при этом представляют собой *срезы* из многомерного хранилища данных, выполненные с разной степенью детализации.

Срез (*Slice*) представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений. Формирование *срезов* выполняется для ограничения используемых пользователем значений,

так как все значения гиперкуба практически никогда одновременно не используются.

Пример. Если ограничить значения измерения *Модель автомобиля* в гиперкубе (рис. 2.14) маркой «Жигули», то получится двумерная таблица продаж этой марки автомобиля различными менеджерами по годам.

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. При организации обработки аналогичных данных на основе реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размерности БД и существенное увеличение затрат оперативной памяти.

Недостатком многомерной модели данных является ее громоздкость для простейших задач обычной оперативной обработки информации.

Словарь терминов

Атрибут – свойство, характеризующее сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Домен – множество всех возможных значений определенного атрибута отношения.

Иерархическая модель – перевернутое *дерево* (*граф*), из *корня* и *узлов* (элементов данных) которого исходят *ветви* (соответствующие связям элементов данных).

Измерение – это множество однотипных данных, образующих одну из граней гиперкуба.

Инкапсуляция позволяет рассматривать объекты как самостоятельные сущности, отделенные от внешнего мира.

Кортеж – строка (запись) таблицы.

Многомерная модель данных используется, когда целью является анализ данных, а не выполнение транзакций, характерное для РМД.

Модель данных – набор принципов, определяющих организацию логической структуры хранения данных в базе.

Наследование – подразумевает возможность создавать из классов объектов новые классы объектов, которые наследуют структуру и методы своих предков, добавляя к ним черты, отражающие их собственную индивидуальность.

Объектно-реляционная модель включает в себя основные достоинства объектно-ориентированной модели и одновременно наследует простоту структуры реляционных моделей.

Отношение – двумерная таблица, содержащая некоторые данные.

Первичный ключ – атрибут отношения, однозначно идентифицирующий каждый из его кортежей.

Полиморфизм – различные объекты могут по-разному реагировать на одинаковые внешние события в зависимости от того, как реализованы их методы.

Постреляционная модель данных – расширенная реляционная модель, снимающая ограничение неделимости данных, хранящихся в ячейках таблиц.

Реляционная модель данных – некоторой предметной области представляет собой набор отношений (таблиц), изменяющихся во времени.

Сетевая модель – структура, у которой один или несколько порожденных элементов имеют более одного исходного элемента.

Срез – представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений.

Сущность – объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.

Ячейка – поле, значение которого однозначно определяется фиксированным набором измерений гиперкуба.

Вопросы и задания для самоконтроля

1. Что такое «модель данных»?
2. Назовите и охарактеризуйте три аспекта модели данных.
3. Охарактеризуйте иерархическую модель данных.
4. Охарактеризуйте сетевую модель данных.
5. Что собой представляет реляционная модель данных?
6. Назовите и охарактеризуйте элементы реляционной модели данных?
7. Для чего в реляционных отношениях используются ключи?
8. При соблюдении каких условий таблицу можно считать отношением?
9. Для чего в реляционных БД между таблицами устанавливаются связи?
10. Назовите и охарактеризуйте основные виды связей в реляционных БД.
11. К чему сводится контроль целостности данных в реляционных БД?
12. Каковы достоинства и недостатки реляционной модели данных?
13. Охарактеризуйте постреляционную модель данных.
14. Охарактеризуйте объектно-ориентированную модель данных.
15. Охарактеризуйте объектно-реляционную модель данных.
16. Охарактеризуйте многомерную модель данных.

3. Проектирование базы данных

Под *проектированием* понимают процесс создания описаний нового или модернизируемого объекта (изделия, процесса), достаточных для изготовления или реализации этого объекта в заданных условиях.

В развитии любого экономического объекта наступает момент осознания того, что для достижения дальнейших успехов в развитии необходимо данные, находящиеся в личном пользовании работников, *интегрировать для совместного использования в базе данных* и воспринимать их как корпоративный ресурс.

Проектирование базы данных – это процесс создания проекта базы данных, предназначенной для поддержки функционирования экономического объекта и способствующей достижению его целей. Оно представляет собой трудоемкий процесс, требующий совместных усилий аналитиков, проектировщиков и пользователей.

При проектировании базы данных необходимо учитывать тот факт, что база данных должна удовлетворять комплексу требований. Эти требования следующие:

1) целостность базы данных – требование полноты и непротиворечивости данных;
2) многократное использование данных;
3) быстрый поиск и получение информации по запросам пользователей;
4) простота обновления данных;
5) уменьшение излишней избыточности данных;
6) защита данных от несанкционированного доступа, искажения и уничтожения.

3.1. Жизненный цикл базы данных

Жизненный цикл базы данных (ЖЦБД) – это процесс проектирования, реализации и поддержки базы данных.

ЖЦБД состоит из следующих *этапов*:

- планирование разработки базы данных;
- определение требований к системе;
- сбор и анализ требований пользователей;
- проектирование базы данных;
- разработка приложений;
- реализация;
- загрузка данных;

– тестирование;
– эксплуатация и сопровождение.

Опишем главные задачи каждого этапа.

3.1.1. Планирование базы данных

Планирование базы данных – важный этап в процессе перехода от разрозненных данных к интегрированным.

Планирование разработки базы данных состоит в определении:
– объема работ;
– ресурсов;
– стоимости проекта;
– проверке осуществимости проекта.

Проверка осуществимости предполагает подготовку отчетов по трем вопросам:

– есть ли технология – необходимое оборудование и программное обеспечение – для реализации запланированной базы данных (технологическая осуществимость);

– имеются ли персонал, средства и эксперты для успешного осуществления плана создания базы данных (операционная осуществимость);

– окупится ли запланированная база данных (экономическая эффективность).

3.1.2. Определение требований к системе

На этом этапе определяются:
– цели базы данных;
– информационные потребности различных структурных подразделений и их руководителей;
– требования к оборудованию;
– требования к программному обеспечению.

3.1.3. Сбор и анализ требований пользователей

На данном этапе необходимо создать для себя модель движения важных материальных объектов и уяснить процесс документооборота.

По каждому документу необходимо установить периодичность использования, определить данные, необходимые для выполнения выделенных функций (анализируя существующую и планируемую документацию, выясняют, как получается каждый элемент данных, кем получается, где в дальнейшем используется, кем контролируется).

Собранная информация о каждой важной области применения приложения и пользовательской группе должна включать следующие компоненты:

- исходную и генерируемую документацию;
- подробные сведения о выполняемых транзакциях;
- список требований с указанием их приоритетов.

3.1.4. Проектирование базы данных

Полный цикл разработки базы данных включает ее концептуальное, логическое и физическое проектирование.

Концептуальное проектирование. Первая фаза процесса проектирования базы данных заключается в создании для анализируемой части предприятия концептуальной модели данных.

Проектирование сложных баз данных с большим количеством атрибутов осуществляется с помощью, так называемого, нисходящего подхода.

Этот подход начинается 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».

Логическое проектирование. На этом этапе осуществляется выбор типа модели данных. Концептуальная модель переводится в *логическую модель*, основанную уже на структурах, характерных для выбранной модели.

Процесс логического проектирования должен опираться на определенную модель данных, которая определяется типом СУБД, предполагаемой для реализации информационной системы.

Физическое проектирование. На этом этапе логическая модель расширяется характеристиками, необходимыми для определения способов физического хранения базы данных, типа устройств для хранения, методов доступа к данным базы, требуемого объема памяти, правил сопровождения базы данных и др.

3.1.5. Разработка приложений

Параллельно с проектированием системы базы данных выполняется разработка приложений. Главные составляющие данного процесса – это проектирование транзакций и пользовательского интерфейса.

Проектирование транзакций. *Транзакции* представляют некоторые события реального мира. Транзакция может состоять из нескольких операций, однако с точки зрения пользователя эти операции представляют собой единое

целое, переводящее базу данных из одного непротиворечивого состояния в другое.

Реализация транзакций опирается на тот факт, что СУБД способна обеспечивать сохранность внесенных во время транзакции изменений в БД и непротиворечивость базы данных даже в случае возникновения сбоя.

Проектирование транзакций заключается в определении:

- данных, которые используются транзакцией;
- функциональных характеристик транзакции;
- выходных данных, формируемых транзакцией;
- степени важности и интенсивности использования транзакции.

Проектирование пользовательского интерфейса. Интерфейс должен быть удобным и обеспечивать все функциональные возможности, предусмотренные в спецификациях требований пользователей.

Специалисты рекомендуют при проектировании пользовательского интерфейса использовать следующие основные элементы и их характеристики:

- содержательное название;
- ясные и понятные инструкции;
- логически обоснованные группировки и последовательности полей;
- визуально привлекательный вид окна формы или поля отчета;
- легко узнаваемые названия полей;
- согласованную терминологию и сокращения;
- согласованное использование цветов;
- визуальное выделение пространства и границ полей ввода данных;
- удобные средства перемещения курсора;
- средства исправления отдельных ошибочных символов и целых полей;
- средства вывода сообщений об ошибках при вводе недопустимых значений;
- особое выделение необязательных для ввода полей;
- средства вывода пояснительных сообщений с описанием полей;
- средства вывода сообщения об окончании заполнения формы.

3.1.6. Реализация

На данном этапе осуществляется физическая реализация базы данных и разработанных приложений, позволяющих пользователю формулировать требуемые запросы к БД и манипулировать данными в БД.

База данных описывается на языке определения данных выбранной СУБД. В результате компиляции его команд и их выполнения создаются схемы и пустые файлы базы данных. На этом же этапе определяются и все специфические пользовательские представления.

Прикладные программы реализуются с помощью языков третьего или четвертого поколений. Кроме того, на этом этапе создаются другие компоненты проекта приложения – например, экраны меню, формы ввода данных и отчеты.

Реализация этого, а также и более ранних этапов проектирования БД, может осуществляться с помощью *инструментов автоматизированного проектирования и создания программ*, которые принято называть *CASE-инструментами (Computer-Aided Software Engineering)*.

3.1.7. Загрузка данных

На этом этапе созданные в соответствии со схемой базы данных пустые файлы, предназначенные для хранения информации, должны быть заполнены данными.

Наполнение базы данных может протекать по-разному, в зависимости от того, создается ли база данных вновь или новая база данных предназначена для замены старой.

3.1.8. Тестирование

На этом этапе определяется законченность и корректность выполнения приложения базы данных.

Существует несколько стратегий тестирования:

- нисходящее тестирование;
- восходящее тестирование;
- тестирование потоков;
- интенсивное тестирование.

Нисходящее тестирование начинается на уровне подсистем с модулями, которые представлены заглушками, т. е. простыми компонентами, имеющими такой же интерфейс, как модуль, но без функционального кода. Каждый модуль низкого уровня представляется заглушкой. Постепенно все программные компоненты заменяются фактическим кодом и после каждой замены снова тестируются.

Восходящее тестирование выполняется в противоположном направлении по отношению к нисходящему. Оно начинается с тестирования модулей на самых низких уровнях иерархии системы, продолжается на более высоких уровнях и заканчивается на самом высоком уровне.

Тестирование потоков осуществляется при тестировании работающих в реальном масштабе времени систем, которые обычно состоят из большого количества взаимодействующих процессов, управляемых с помощью прерываний. Стратегия тестирования потоков направлена на слежение за отдельными процессами.

Стратегия *интенсивного тестирования* часто включает серию тестов с постепенно возрастающей нагрузкой и продолжается до тех пор, пока система не выйдет из строя.

3.1.9. Эксплуатация и сопровождение

Основные действия, связанные с этим этапом, сводятся к наблюдению за созданной системой и поддержке ее нормального функционирования по окончании развертывания.

Поддержка БД предполагает разрешение проблем, возникающих в процессе эксплуатации БД и связанных как с ошибками реализации БД, так и с изменениями в самой предметной области, созданием дополнительных программных компонентов или модернизацией самой БД.

3.2. Методы проектирования реляционных баз данных

3.2.1. Модель «сущность-связь»

Средством моделирования предметной области на этапе концептуального проектирования является модель «сущность-связь». В нем моделирование структуры данных предметной области базируется на использовании графических средств *ER-диаграмм* (диаграмм «сущность-связь»). В наглядном виде они представляют связи между сущностями.

Аббревиатура ER происходит от слов *Entity (сущность) и Relationship (связь)*.

Основные понятия ER-диаграммы. Основные понятия ER-диаграммы – это сущность, атрибут, связь.

Сущность – это некоторый объект реального мира, который может существовать независимо. Сущность имеет *экземпляры*, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию.

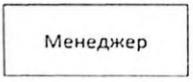
Атрибут – это свойство сущности.

Пример. Сущность КНИГА характеризуется такими атрибутами, как автор, наименование, цена, издательство, тираж, количество страниц. Конкретные книги являются экземплярами сущности КНИГА. Они отличаются значениями указанных атрибутов и однозначно идентифицируются атрибутом «наименование». Атрибут, который уникальным образом идентифицирует экземпляры сущности, называется *ключом*. Может быть *составной* ключ, представляющий комбинацию нескольких атрибутов.

Задача. Предположим, что проектируется база данных, предназначенная для хранения информации о деятельности некоторого банка. Этот банк имеет филиалы. Филиалы управляются менеджерами. Клиенты имеют в филиалах счета разных

типов – текущие, срочные, до востребования, депозитные, карточные. Филиалы обрабатывают эти счета. Описываемую предметную область назовем БАНК. В ней могут быть выделены четыре сущности: филиал, менеджер, счет, клиент.

На ER-диаграмме сущность изображается прямоугольником, в котором указывается ее имя. Например,



В реальном мире существуют связи между сущностями. *Связь* представляет взаимодействие между сущностями. Она характеризуется *мощностью*, которая показывает, сколько сущностей участвует в связи. Связь между двумя сущностями называется *бинарной*, а связь между более чем двумя сущностями *тернарной*. В рассматриваемой предметной области БАНК можно выделить три связи.

1) МЕНЕДЖЕР – УПРАВЛЯЕТ – ФИЛИАЛ
2) ФИЛИАЛ – ОБРАБАТЫВАЕТ – СЧЕТ
3) КЛИЕНТ – ИМЕЕТ – СЧЕТ

На ER-диаграмме связь изображается ромбом. Например,

Важной характеристикой связи является *тип* связи (*кардинальность*). Рассмотрим типы связей 1-3.

Так как менеджер управляет только одним филиалом, то каждый экземпляр сущности МЕНЕДЖЕР может быть связан не более чем с одним экземпляром сущности ФИЛИАЛ. В этом случае связь 1 имеет тип «один-к-одному» (1:1). На рисунке 3.1 представлена ER-диаграмма для связи типа 1:1.

Рис. 3.1. ER-диаграмма связи 1:1

Так как филиал обрабатывает несколько счетов, а счет обрабатывается только одним филиалом, то каждый экземпляр сущности **ФИЛИАЛ** может быть связан более чем с одним экземпляром сущности **СЧЕТ**, а каждый экземпляр сущности **СЧЕТ** может быть связан не более чем с одним экземпляром сущности **ФИЛИАЛ**. В этом случае связь 2 имеет тип «один-ко-многим» (1:M). На рисунке 3.2 представлена ER-диаграмма для связи типа 1:M.

Рис. 3.2. ER-диаграмма связи 1:M

Так как счет может совместно использоваться несколькими клиентами и клиент может иметь несколько счетов, то каждый экземпляр сущности **СЧЕТ** может быть связан с несколькими экземплярами сущности **КЛИЕНТ** и каждый экземпляр сущности **КЛИЕНТ** может быть связан с несколькими экземплярами сущности **СЧЕТ**. В этом случае связь 3 имеет тип «многие-ко-многим» (M:N).

На рисунке 3.3 представлена ER-диаграмма для связи типа M:N.

Рис. 3.3. ER-диаграмма связи M:N

Рассмотрим понятие класс принадлежности сущности.

Если каждый экземпляр сущности **A** связан с экземпляром сущности **B**, то класс принадлежности сущности **A** является *обязательным*. Этот факт отмечается на ER-диаграмме черным кружочком, помещенным в прямоугольник, смежный с прямоугольником сущности **A**.

Если не каждый экземпляр сущности **A** связан с экземпляром сущности **B**, то класс принадлежности сущности **A** является *необязательным*. Этот факт отмечается на ER-диаграмме черным кружочком, помещенным на линии связи возле прямоугольника сущности **A**.

В качестве примера на рисунке 3.4 изображены возможные ER-диаграммы для связи **M:N** с учетом класса принадлежности сущности.

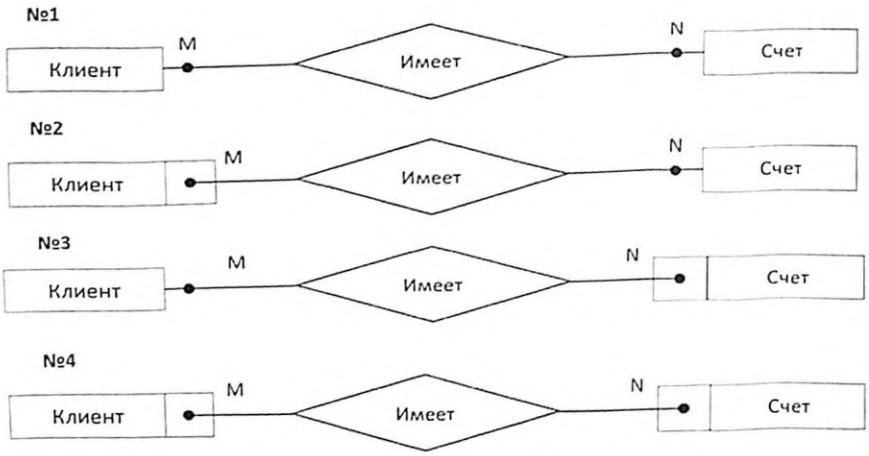


Рис. 3.4. ER-диаграммы связи M:N с учетом класса принадлежности сущности

На ER-диаграмме №1 класс принадлежности обеих сущностей необязательный.

На ER-диаграмме №2 класс принадлежности сущности **КЛИЕНТ** обязательный, а сущности **СЧЕТ** необязательный.

На ER-диаграмме №3 класс принадлежности сущности **КЛИЕНТ** необязательный, а сущности **СЧЕТ** обязательный.

На ER-диаграмме №4 класс принадлежности обеих сущностей обязательный.

Предположим, что в рассматриваемой предметной области **БАНК** класс принадлежности всех четырех сущностей является обязательным. Тогда ER-модель предметной области **БАНК** будет иметь вид, представленный на рисунке 3.5.

Рис. 3.5. Пример ER-модели предметной области БАНК

Каждая из четырех сущностей приведенной ER-модели может быть описана своим набором атрибутов (рис. 3.6).

МЕНЕДЖЕР
Номер менеджера (НМ)
Стаж работы (СТАЖ)
Специальность (СПЕЦ)

СЧЕТ
Номер счета (НС)
Тип счета (ТИП)
Остаток на счете (ОСТ)

ФИЛИАЛ
Номер филиала (НФ)
Адрес филиала (АДР_Ф)

КЛИЕНТ
Номер клиента (НК)
Ф.И.О. клиента (ФИО_К)
Социальное положение (СОЦ)
Адрес клиента (АДР_К)

Рис. 3.6. Наборы атрибутов сущностей предметной области БАНК

Примечание. Ключевые атрибуты выделены жирным шрифтом.

ER-модель в совокупности с наборами атрибутов сущностей может служить примером концептуальной модели предметной области или концептуальной схемы базы данных.

В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в CASE-средствах. Эти средства предназначены для автоматизированного проектирования реляционных баз данных.

CASE-средства позволяют строить ER-диаграммы в реальном масштабе времени, что дает возможность наглядно изучать концептуальную модель данных и перестраивать ее соответственно поставленным целям и имеющимся ограничениям.

Преобразование ER-модели в реляционную модель. Концептуальные модели позволяют более точно представить предметную область, чем реляционные и другие более ранние модели. Но в настоящее время существует немного систем управления базами данных, поддерживающих эти

модели. На практике наиболее распространены системы, реализующие реляционную модель. Поэтому необходим метод перевода концептуальной модели в реляционную. Такой метод основывается на формировании набора предварительных таблиц из ER-диаграмм.

Для каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы. Правила генерации таблиц из ER-диаграмм опираются на два основных фактора: тип связи и класс принадлежности сущности. Изложим их.

Правило 1. Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным, то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

На ER-диаграмме связи 1:1, представленной на рисунке 3.5, класс принадлежности сущностей МЕНЕДЖЕР, ФИЛИАЛ является обязательным. Тогда согласно правилу 1 должна быть сгенерирована одна таблица следующей структуры:

МЕНЕДЖЕР-ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

Первичным ключом этой таблицы может быть и первичный ключ сущности МЕНЕДЖЕР – *НМ*.

Правило 2. Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

Представим, что на ER-диаграмме связи 1:1, изображенной на рисунке 3.5, класс принадлежности сущности МЕНЕДЖЕР будет обязательный, а сущности ФИЛИАЛ – необязательный. Тогда согласно правилу 2 должны быть сгенерированы две таблицы следующей структуры:

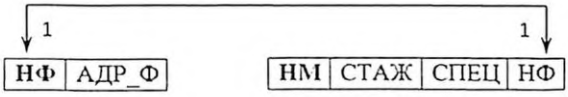
МЕНЕДЖЕР-ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	НФ

ФИЛИАЛ

НФ	АДР_Ф

Сущность с необязательным классом принадлежности (ФИЛИАЛ) именуется *родительской*, а с обязательным (МЕНЕДЖЕР) – *дочерней*. Первичный ключ родительской сущности – *НФ*, помещаемый в таблицу, представляющую дочернюю сущность, называется *внешним* ключом родительской сущности. Связь между указанными таблицами устанавливается путем связи первичного и внешнего ключа и имеет вид:

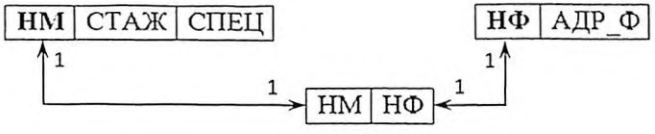


Примечание. Если внешний ключ представляет связь 1:1, то должны быть запрещены его дублирующие значения.

Правило 3. Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Представим, что на ER-диаграмме связи 1:1, изображенной на рисунке 3.5, класс принадлежности сущностей МЕНЕДЖЕР, ФИЛИАЛ будет необязательный. Тогда согласно правилу 3 должны быть сгенерированы три таблицы следующей структуры:

При этом осуществляется декомпозиция связи 1:1 на две связи 1:1 следующим образом:



Итак, для связи типа 1:1 существуют три отдельных правила формирования предварительных таблиц из ER-диаграмм.

Для связи типа 1:M существуют только два правила. Выбор одного из них зависит от класса принадлежности сущности на стороне M. Класс принадлежности сущности на стороне 1 не влияет на выбор.

Правило 4. Если связь типа 1:M и класс принадлежности сущности на стороне M является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне M.

На ER-диаграмме связи 1:M, представленной на рисунке 3.5 класс принадлежности сущности **СЧЕТ** является обязательным. Тогда согласно правилу 4 должны быть сгенерированы две таблицы следующей структуры:

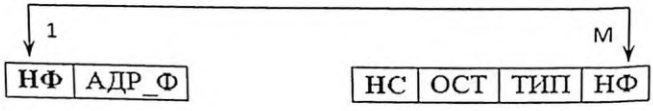
ФИЛИАЛ

НФ	АДР_Ф

СЧЕТ-ФИЛИАЛ

НС	ОСТ	ТИП	НФ

Связь между указанными таблицами будет иметь вид:

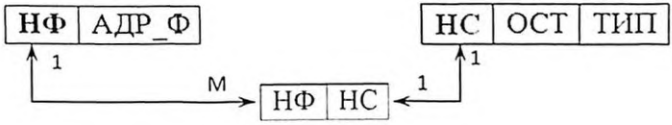


Примечание. Если внешний ключ представляет связь 1:M, то должны быть разрешены его дублирующие значения.

Правило 5. Если связь типа 1:M и класс принадлежности сущности на стороне M является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Представим, что на ER-диаграмме связи 1:M, изображенной на рисунке 3.5, класс принадлежности сущности СЧЕТ является необязательным. Тогда согласно правилу 5 должны быть сгенерированы три таблицы следующей структуры:

При этом осуществляется декомпозиция связи 1:M на две связи 1:M и 1:1 следующим образом:

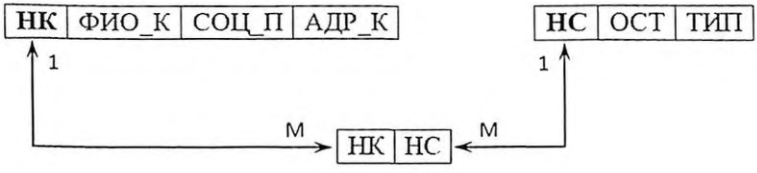


Для связи типа M:N класс принадлежности сущности не имеет значения.

Правило 6. Если связь типа M:N, то необходимо построить три таблицы по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

ER-диаграмма связи M:N имеется на рисунке 3.5. Согласно правилу 6 на основе этой ER-диаграммы должны быть сгенерированы три таблицы следующей структуры:

При этом осуществляется декомпозиция связи M:N на две связи:



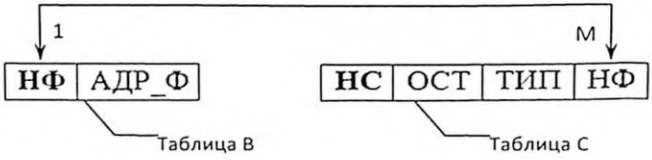
В таблице КЛИЕНТ–СЧЕТ клиенту, имеющему, например, три счета, будут соответствовать три строки с одним и тем же номером клиента. А счет, у которого, например, два владельца, представляется двумя строками с различными номерами клиентов, владеющими этим счетом.

К ER-модели предметной области БАНК, представленной на рисунке 3.5, применимы правила 1, 4, 6. Связь МЕНЕДЖЕР–ФИЛИАЛ представляется (согласно правилу 1) одной таблицей:

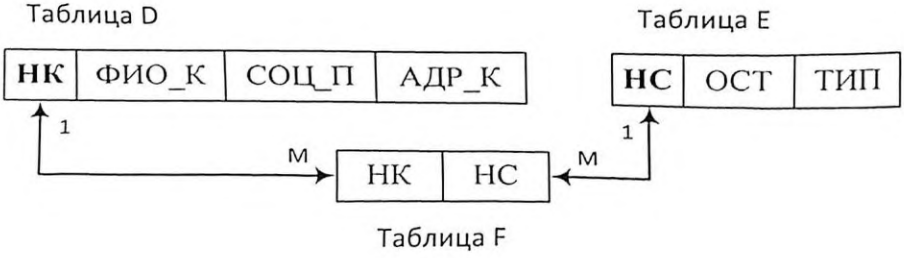
таблица A

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

Связь ФИЛИАЛ–СЧЕТ представляется (согласно правилу 4) связью:



Связь КЛИЕНТ–СЧЕТ представляется (согласно правилу 6) связью:



Анализ состава атрибутов полученных таблиц A, B, C, D, E, F показывает, что таблица B является составной частью таблицы A, таблица E – составной частью таблицы C. Поэтому таблицы B, E можно исключить из рассмотрения. Оставшиеся таблицы A, C, D, F можно связать посредством связи первичных и внешних ключей, как на рисунке 3.7. В результате получим реляционную модель для ER-модели предметной области БАНК.

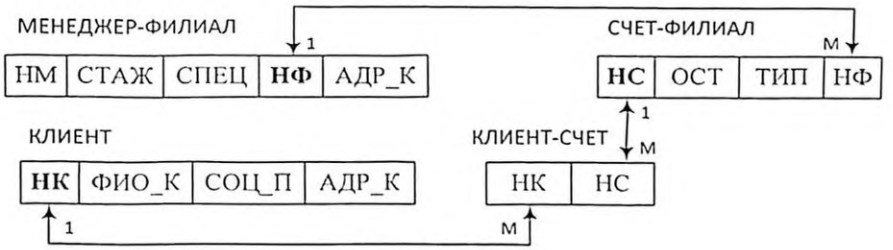


Рис. 3.7. Реляционная модель предметной области **БАНК**

3.2.2. Метод нормализации

Здесь будет рассмотрен классический (исторически первый) подход, при котором весь процесс проектирования производится в терминах реляционной модели данных методом последовательных приближений к удовлетворительному набору схем отношений.

Нормализация – это процесс декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, удовлетворяющих требованиям следующей нормальной формы.

Исходной точкой этого процесса является представление предметной области в виде одного или нескольких отношений, и на каждом шаге проектирования производится некоторый набор схем отношений, обладающих лучшими свойствами.

Методику нормализации отношений разработал американский ученый Э. Кодд в 1970 г. (три первых нормальных формы). В целом в теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- нормальная форма Бойса-Кодда (БКНФ);
- четвертая нормальная форма (4НФ);
- пятая нормальная форма (5НФ).

Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

Основные свойства нормальных форм:

– каждая следующая нормальная форма в некотором смысле лучше предыдущей;

– при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Формирование исходного отношения. Проектирование БД начинается с определения всех объектов, сведения о которых будут включены в базу, и определения их атрибутов. Затем атрибуты сводятся в одну таблицу – *исходное отношение*.

Пример. Предположим, что для учебной части факультета создается БД о преподавателях. На первом этапе проектирования БД в результате общения с заказчиком (учебным отделом) должны быть определены содержащиеся в базе сведения о том, как она должна использоваться и какую информацию заказчик хочет получать в процессе ее эксплуатации. В результате устанавливаются атрибуты, которые должны содержаться в отношениях БД, и связи между ними. Перечислим имена выделенных атрибутов и их краткие характеристики:

ФИО – фамилия и инициалы преподавателя. Исключаем возможность совпадения фамилии и инициалов у преподавателей.

Должн – должность, занимаемая преподавателем.

Оклад – оклад преподавателя.

Стаж – преподавательский стаж.

Н_Стаж – надбавка за стаж.

Каф – номер кафедры, на которой числится преподаватель.

Предм – название предмета (дисциплины), читаемого преподавателем.

Группа – номер группы, в которой преподаватель проводит занятия.

ВидЗан – вид занятий, проводимых преподавателем в учебной группе.

Каждый преподаватель в одной группе по одному предмету может либо читать лекции, либо проводить практические занятия.

Исходное ненормализованное отношение, содержащее сведения о преподавателях может иметь следующий вид.

Первая нормальная форма. Отношение находится в 1НФ, если все его атрибуты являются атомарными (простыми).

Приведем исходное отношение «Преподаватель» к первой нормальной форме путем вставки записей (рис. 3.8). Исходное отношение «Преподаватель» находится в 1НФ, поскольку все его атрибуты простые, и имеет составной ключ ФИО, Предм, Группа.

Вторая нормальная форма. Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от ключа.

В отношении «Преподаватель» в соответствии с рисунком 3.2 можно выделить частичную зависимость атрибутов **Стаж**, **Н_Стаж**, **Каф**, **Должн**, **Оклад** от ключа – указанные атрибуты находятся в функциональной зависимости только от атрибута **ФИО**, являющегося частью составного ключа.

ФИО	Должн	Оклад	Стаж	Н_Стаж	Каф	Предм	Группа	ВидЗан
Иванов И. М.	ассист.	500	5	100	25	СУБД; Программирование	256 123	Практ Практ
Петров М. И.	ст.преп.	800	7	100	25	СУБД; Информатика	256	Лекция; Практ
Сидоров Н. Г.	ассист.	500	10	150	25	Программирование; Информатика	123 256	Лекция Лекция
Егоров В. В.	ассист.	500	5	100	24	Радиотехника	244	Лекция

Рис. 3.1. Исходное отношение «Преподаватель»

ФИО	Должн	Оклад	Стаж	Н_Стаж	Каф	Предм	Группа	ВидЗан
Иванов И. М.	ассист	500	5	100	25	СУБД	256	Практ
Иванов И. М.	ассист	500	5	100	25	Программирование	123	Практ
Петров М. И.	ст.преп.	800	7	100	25	СУБД	256	Лекция
Петров М. И.	ст.преп.	800	7	100	25	Информатика	256	Практ
Сидоров Н. Г.	ассист	500	10	150	25	Программирование	123	Лекция
Сидоров Н. Г.	ассист	500	10	150	25	Информатика	256	Лекция
Егоров В. В.	ассист	500	5	100	24	Радиотехника	244	Лекция

Рис. 3.8. Отношение «Преподаватель» в 1НФ

Эта частичная зависимость от ключа приводит к следующему:

1) в отношении присутствует дублирование данных, например:

– повторение сведений о стаже, должности и окладе преподавателей, проводящих занятия в нескольких группах и/или по разным предметам,

– повторение сведений об окладах для одной и той же должности или о надбавках за одинаковый стаж;

2) следствием избыточного дублирования данных является проблема их редактирования. Например, изменение должности у преподавателя Иванова И. М. потребует просмотра всех кортежей отношения и внесения изменений в те из них, которые содержат сведения о данном преподавателе.

Для устранения частичной зависимости и перевода отношения в 2НФ его необходимо разбить на два отношения, R1 и R2, следующим образом:

R1

ФИО	Предм	Группа	ВидЗан
Иванов И. М.	СУБД	256	Практ
Иванов И. М.	Программирование	123	Практ
Петров М. И.	СУБД	256	Лекция
Петров М. И.	Информатика	256	Практ
Сидоров Н. Г.	Программирование	123	Лекция
Сидоров Н. Г.	Информатика	256	Лекция
Егоров В. В.	Радиотехника	244	Лекция

R2

ФИО	Должн	Оклад	Стаж	Н_Стаж	Каф
Иванов И. М.	ассист	500	5	100	25
Петров М. И.	ст. преп.	800	7	100	25
Сидоров Н. Г.	ассист	500	10	150	25
Егоров В. В.	ассист	500	5	100	24

Рис. 3.9. Отношения R1 и R2, полученные в результате декомпозиции исходного отношения «Преподаватель», в 2НФ

В отношении R1 первичный ключ является составным и состоит из атрибутов ФИО, Предм, Группа. Напомним, что данный ключ в отношении R1 получен в предположении, что каждый преподаватель в одной группе по одному предмету может либо читать лекции, либо проводить практические занятия. В отношении R2 ключ – ФИО.

Третья нормальная форма. Отношение находится в 3НФ, если оно находится в 2НФ и каждый неключевой атрибут *нетранзитивно* зависит от первичного ключа.

Транзитивной называется такая зависимость, при которой какое-либо неключевое поле зависит от другого неключевого поля, а то в свою очередь зависит от ключа.

Если в отношении R1 транзитивные зависимости отсутствуют, то в отношении R2 они есть:

- Оклад→Должн→ФИО;
- Н_Стаж→Стаж→ФИО.

Устраним транзитивные зависимости. Для этого отношение R2 разобьем на три новых отношения R3, R4, R5.

R3

ФИО	Должн	Стаж	Каф
Иванов И. М.	ассист	5	25
Петров М. И.	ст. преп.	7	25
Сидоров Н. Г.	ассист	10	25
Егоров В. В.	ассист	5	24

R4

Должн	Оклад
ассист	500
ст. преп.	800

R5

Стаж	Н_Стаж
5	100
7	100
10	150

Рис. 3.10. Отношения R3, R4 и R5,
полученные в результате декомпозиции отношения R2, в 3НФ

В результате требования 3НФ удовлетворены.

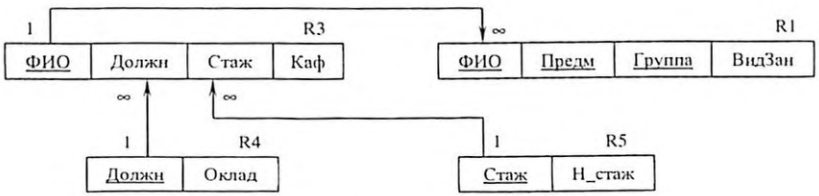


Рис. 3.11. Схема отношения БД «Преподаватель»,
полученная в процессе нормализации отношений

На практике построение 3НФ схем отношений в большинстве случаев является достаточным и приведением к ним процесс проектирования структур реляционных БД заканчивается.

3.3. Семантическая объектная модель

Семантическая объектная модель (semantic object model) так же, как и модель «сущность-связь», используется для моделирования данных. Семантическая объектная модель (СОМ) была впервые представлена Эдгаром Коддом в 1988 году как *модель данных*.

Рассмотрим, как эта модель используется для моделирования данных.

3.3.1. Семантические объекты

Начальная задача проектировщика базы данных состоит в том, чтобы определить зависимость форм, отчетов и запросов от сущностей и объектов, представляющих смысловой интерес для потребителя базы данных. Поэтому целью ранних стадий разработки базы данных является определение того, какие объекты должны быть представлены в базе данных, с какими характеристиками и с какими взаимосвязями.

Возник термин *«семантический объект» (semantic objects)*. Слово «семантический» означает «смысловой», и *семантический объект* – это объект, который в определенной степени моделирует смысл пользовательских данных. Семантические объекты моделируют восприятие пользователя более точно, чем модель «сущность–связь».

3.3.2. Определение семантических объектов

Сущности и объекты в некоторых отношениях схожи, но у них есть и различия. Сходство заключается в том, что семантический объект – это представление некоторой вещи, идентифицируемой в рабочей среде пользователя. Если выражаться более формально, *семантический объект* – это именованная совокупность атрибутов, которая в достаточной степени описывает отдельный экземпляр. Подобно сущностям, семантические объекты группируются в классы. У объектного класса есть имя, которое отличает его от других классов и соответствует именам вещей, представляемых этим классом.

Семантические объекты имеют *атрибуты*, описывающие их характеристики. Есть три типа атрибутов:

1) *простые атрибуты (simple attributes)*. Состоят из одного элемента (например, `Фамилия студента`);

2) *групповые атрибуты (group attributes)*. Являют собой совокупности других атрибутов (пример – атрибут `Адрес`, состоящий из атрибутов `{Улица, Город, Область, Индекс}`);

3) *семантические объектные атрибуты (semantic object attributes)*. Это атрибуты, которые устанавливают связь между двумя семантическими объектами (пример – атрибуты `ФАКУЛЬТЕТ`, `ПРЕПОДАВАТЕЛЬ`, `СТУДЕНТ`, показанные на рисунке 3.12).

Набор атрибутов является достаточным, если он отражает все характеристики, которые требуются пользователям для выполнения их работы. Объектные атрибуты могут быть простыми элементами данных, группами или другими семантическими объектами.

На рисунке 3.12 показана так называемая *семантическая объектная диаграмма (semantic object diagram)*, или просто *объектная диаграмма (object diagram)*. Такие диаграммы используются разработчиками для описания и визуального представления структуры объектов. Объекты изображаются в вертикально ориентированных прямоугольниках. Имена объектов пишутся заглавными буквами в верхней части диаграммы. Необъектные атрибуты записываются строчными буквами, а группы атрибутов заключаются в скобки.

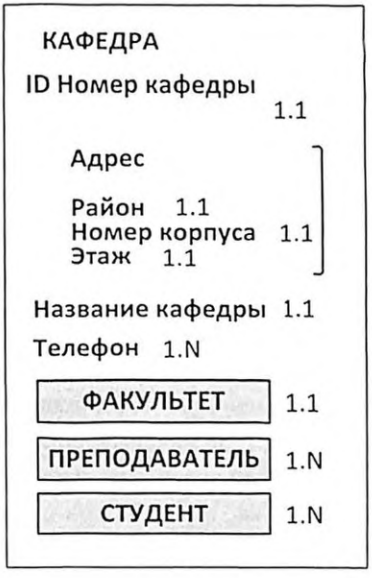


Рис. 3.12. Семантическая объектная диаграмма объекта **КАФЕДРА**

Смысл этих объектных атрибутов, или *объектных ссылок (object links)* состоит в следующем: когда пользователь анализирует информацию о кафедре, он имеет в виду не только название кафедры, ее адрес, номер телефона кафедры, но также и факультет, к которому она относится, преподавателей и студентов, относящихся к ней. Поскольку **ФАКУЛЬТЕТ**, **ПРЕПОДАВАТЕЛЬ** и **СТУДЕНТ** также являются объектами, полная модель данных содержит диаграммы и для них. Объект **ФАКУЛЬТЕТ** несет в себе атрибуты факультета, объект **ПРЕПОДАВАТЕЛЬ** – атрибуты преподавательского состава, а объект **СТУДЕНТ** содержит атрибуты студентов.

Каждый атрибут семантического объекта имеет максимальное и минимальное *кардинальные числа*. Минимальное кардинальное число показывает количество экземпляров атрибута, которые должны существовать, чтобы объект был допустимым. Обычно это число равно 0 или 1. Если оно равно 0, атрибут не обязан иметь значение, а если 1, то атрибут обязан иметь значение. Минимальное кардинальное число иногда может быть больше единицы.

Пример. Атрибут **Студент** в объекте под названием **ГРУППА** может иметь минимальное кардинальное число, равное 10, это означает, что наименьшее число студентов, требуемое для создания группы, равно 10.

Максимальное кардинальное число показывает максимальное количество экземпляров атрибута, которое может иметь объект. Обычно оно равно 1 или N. Если оно равно 1, атрибут может иметь не более одного экземпляра; если оно равно N, атрибут может иметь много экземпляров, и предельное количество не задано. Иногда максимальное кардинальное число равно определенному числу, например 5, – это означает, что объект может иметь не более пяти экземпляров атрибута.

Кардинальность изображается в виде нижнего индекса атрибута в формате **N.M**, где **N** – минимальное кардинальное число, а **M** – максимальное. На рисунке 3.12 минимальное кардинальное число для атрибута **Название кафедры** равно 1, и максимальное также 1. Таким образом, требуется ровно один экземпляр этого атрибута. Кардинальность атрибута **Телефон** равна **1.N**, то есть кафедра обязана иметь минимум один номер телефона, но в принципе номеров у нее может быть много.

Объектный идентификатор (object identifier) – это один или несколько объектных атрибутов, с помощью которых пользователи идентифицируют экземпляры объектов. В семантических объектных диаграммах объектные идентификаторы обозначаются с помощью букв ID перед атрибутом. Обычно, если атрибут должен использоваться в качестве идентификатора, он обязан иметь значение. Кроме того, как правило, для данного объекта имеется не более одного идентифицирующего атрибута. Поэтому в большинстве случаев кардинальность атрибута-идентификатора равна **1.1**.

Домен (domain) атрибута – это описание множества его значений. Характеристики домена зависят от типа атрибута. Домен простого атрибута состоит из *физического* и *семантического описания*.

Физическое описание (physical description) показывает тип данных (например, число или строка), длину данных и другие ограничения (например, требование, чтобы первый символ был буквой или чтобы значение не превышало 10).

Семантическое описание (semantic description) указывает функцию или назначение данного атрибута; оно отличает этот атрибут от других атрибутов с тем же физическим описанием.

Домен объектного атрибута – это набор экземпляров объектов данного типа. На рисунке 3.12, например, доменом объектного атрибута ПРЕПОДАВАТЕЛЬ является совокупность всех экземпляров объектов класса ПРЕПОДАВАТЕЛЬ, представленных в базе данных. В определенном смысле, *домен атрибута* – это динамически нумерованный список, который содержит все экземпляры объектов данного типа.

Существует семь типов объектов:

1) *простые объекты* не имеют многозначных и объектных атрибутов;

2) *композитные объекты* имеют многозначные атрибуты, но не имеют объектных атрибутов;

3) *составные объекты* содержат объектные атрибуты;

4) *гибридные объекты* сочетают в себе композитные и составные объекты;

5) *ассоциативные объекты* связывают два или более других объектов;

6) *объекты подтипов* представляют разновидности объектов;

7) *объекты вида архетип/версия* используются для моделирования объектов, содержащих базовые данные и множественные их вариации, или версии.

3.3.3. Представления семантических объектов

Пользователи получают информацию из базы данных с помощью объектов (в СУБД Microsoft Office Access такими объектами являются формы, запросы и отчеты) или через приложения (например, Visual Basic строит приложение для выбора данных из СУБД SQL Server) базы данных, которые предоставляют формы для ввода данных, отчеты и запросы. В большинстве случаев такие формы, отчеты и запросы не требуют доступа ко всем атрибутам объекта.

Часть объекта, видимая для конкретного приложения, называется *представлением семантического объекта (semantic object view)*, или просто *представлением (view)*. Представление состоит из имени объекта и списка всех атрибутов, видимых в этом представлении.

На рисунке 3.13 показаны два представления семантического объекта КАФЕДРА.

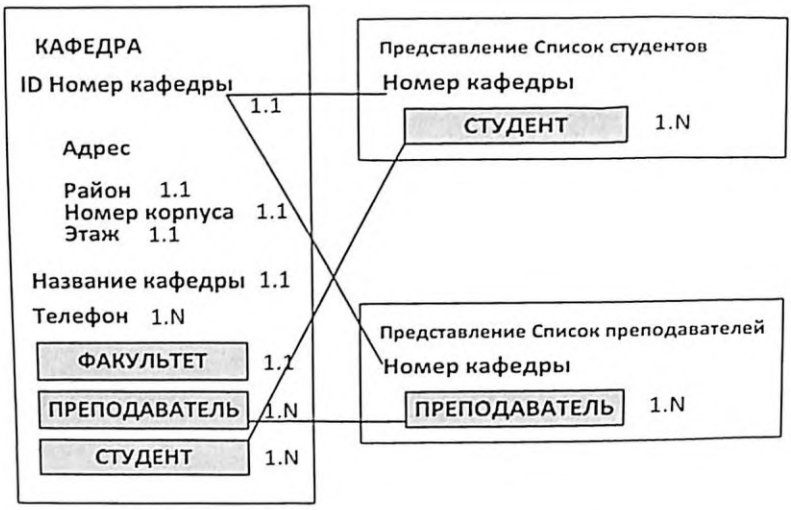


Рис. 3.13 Представления семантического объекта КАФЕДРА – Список студентов и Список преподавателей

3.3.4. Использование представлений

Представления используются двумя способами.

Если известны требования к виду и содержанию выходных информационных объектов, то при разработке модели данных разработчики базы данных и приложений двигаются в обратном направлении. То есть они начинают с форм, отчетов и запросов, которые *пользователи объявляют необходимыми*, и двигаются назад, к структуре базы данных. Для этого разработчик выбирает требуемую форму, отчет или запрос и определяет, какое представление должно существовать, чтобы можно было создать такую форму, отчет или запрос. Затем разработчик переходит к следующей форме, отчету или запросу. Далее получившиеся два представления объединяются. Описанный процесс повторяется до тех пор, пока структура базы данных не будет полностью сформирована.

Второй вариант использования представлений возникает, когда структура базы данных уже создана. В этом случае представления создаются для поддержки новых форм, отчетов и запросов на основе существующей структуры базы данных.

3.4. Этапы проектирования базы данных и их процедуры

Проектирование базы данных осуществляется в три этапа:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

3.4.1. Процедуры концептуального проектирования

Цель этапа концептуального проектирования – *создание концептуальной модели данных исходя из представлений пользователей о предметной области*. Для ее достижения выполняется ряд последовательных процедур.

1. *Определение сущностей и их документирование*. Для идентификации сущностей определяются объекты, которые существуют независимо от других. Такие объекты являются *сущностями*. Каждой сущности присваивается осмысленное имя, понятное пользователям. Имена и описания сущностей заносятся в *словарь данных*. Если возможно, то устанавливается ожидаемое количество экземпляров каждой сущности.

2. *Определение связей между сущностями и их документирование*. Определяются только те связи между сущностями, которые необходимы для удовлетворения требований к проекту базы данных. Устанавливается тип каждой из них. Выявляется класс принадлежности сущностей. Связям присваиваются осмысленные имена, выраженные глаголами. Развернутое описание каждой связи с указанием ее типа и класса принадлежности сущностей, участвующих в связи, заносится в словарь данных.

3. *Создание ER-модели предметной области*. Для представления сущностей и связей между ними используются ER-диаграммы. На их основе создается единый наглядный образ моделируемой предметной области – ER-модель предметной области.

4 Определение атрибутов и их документирование. Выявляются все атрибуты, описывающие сущности созданной ER-модели. Каждому атрибуту присваивается осмысленное имя, понятное пользователям. О каждом атрибуте в словарь данных помещаются следующие сведения:

- имя атрибута и его описание;
- тип и размерность значений;
- значение, принимаемое для атрибута по умолчанию (если такое имеется);
- может ли атрибут иметь Null-значения;

– является ли атрибут составным, и если это так, то из каких простых атрибутов он состоит. Например, атрибут «Ф.И.О. клиента» может состоять из простых атрибутов «Фамилия», «Имя», «Отчество», а может быть простым, содержащим единые значения, как-то «Николаев Сергей Александрович». Если пользователь не нуждается в доступе к отдельным элементам «Ф.И.О.», то атрибут представляется как простой;
– является ли атрибут расчетным, и если это так, то как вычисляются его значения.

5. Определение значений атрибутов и их документирование. Для каждого атрибута сущности, участвующей в ER-модели, определяется набор допустимых значений и ему присваивается имя. Например, атрибут «Тип счета» может иметь только значения «депозитный», «текущий», «до востребования», «карт-счет». Обновляются записи словаря данных, относящиеся к атрибутам, – в них заносятся имена наборов значений атрибутов.

6. Определение первичных ключей для сущностей и их документирование. На этом шаге руководствуются определением первичного ключа – как атрибута или набора атрибутов сущности, позволяющего уникальным образом идентифицировать ее экземпляры. Сведения о первичных ключах помещаются в словарь данных.

7. Обсуждение концептуальной модели данных с конечными пользователями. Концептуальная модель данных представляется ER-моделью с сопроводительной документацией, содержащей описание разработанной модели данных. Если будут обнаружены несоответствия предметной области, то в модель вносятся изменения до тех пор, пока пользователи не подтвердят, что предложенная им модель адекватно отображает их личные представления.

3.4.2. Процедуры логического проектирования

Цель этапа логического проектирования – преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависимую от особенностей используемой в дальнейшем СУБД для физической реализации базы данных. *Для ее достижения выполняются следующие процедуры.*

1. *Выбор модели данных.* Чаще всего выбирается реляционная модель данных в связи с наглядностью табличного представления данных и удобства работы с ними.

2. Определение набора таблиц, исходя из ER-модели, и их документирование. Для каждой сущности ER-модели создается таблица. Имя сущности – имя таблицы. Осуществляется формирование структуры таблиц на основании изложенных в разделе 3.2.1 правил. Устанавливаются связи между таблицами посредством механизма первичных и внешних ключей. Структуры таблиц и установленные связи между ними документируются.

3. Нормализация таблиц. Для правильного выполнения нормализации проектировщик должен глубоко изучить семантику и особенности использования данных. На этом шаге он проверяет корректность структуры таблиц, созданных на предыдущем шаге, посредством применения к ним процедуры нормализации. Эта процедура была описана в параграфе 1.5. Она заключается в приведении каждой из таблиц, по крайней мере, к ЗНФ. В результате нормализации получается очень гибкий проект базы данных, позволяющий легко вносить в нее нужные расширения.

4. Проверка логической модели данных на предмет возможности выполнения всех транзакций, предусмотренных пользователями. Транзакция – это набор действий, выполняемых отдельным пользователем или прикладной программой с целью изменения содержимого базы данных. Так, примером транзакции в проекте **БАНК** может быть передача права распоряжаться счетами некоторого клиента другому клиенту. В этом случае в базу данных потребуется внести сразу несколько изменений. Если во время выполнения транзакции произойдет сбой в работе компьютера, то база данных окажется в противоречивом состоянии, так как некоторые изменения уже будут внесены, а остальные еще нет. Поэтому все частичные изменения должны быть отменены для возвращения базы данных в прежнее непротиворечивое состояние.

Перечень транзакций определяется действиями пользователей в предметной области. Используя ER-модель, словарь данных и установленные связи между первичными и внешними ключами, производится попытка выполнить все необходимые операции доступа к данным вручную. Если какую-либо операцию выполнить вручную не удается, то составленная логическая модель данных является неадекватной и содержит ошибки, которые надо устранить. Возможно, они связаны с пропуском в модели сущности, связи или атрибута.

5. Определение требований поддержки целостности данных и их документирование. Эти требования представляют собой ограничения, которые вводятся с целью предотвратить помещение в базу данных противоречивых

данных. На этом шаге вопросы целостности данных освещаются безотносительно к конкретным аспектам ее реализации. Должны быть рассмотрены следующие типы ограничений:

- обязательные данные. Выясняется, есть ли атрибуты, которые не могут иметь Null-значений;
- ограничения для значений атрибутов. Определяются допустимые значения для атрибутов;
- целостность сущностей. Она достигается, если первичный ключ сущности не содержит Null-значений;
- ссылочная целостность. Она понимается так, что значение внешнего ключа должно обязательно присутствовать в первичном ключе одной из строк таблицы для родительской сущности;
- ограничения, накладываемые бизнес-правилами. Например, в случае с проектом БАНК может быть принято правило, запрещающее клиенту распоряжаться, скажем, более чем тремя счетами.

Сведения обо всех установленных ограничениях целостности данных помещаются в словарь данных.

6. Создание окончательного варианта логической модели данных и обсуждение его с пользователями. На этом шаге подготавливается окончательный вариант ER-модели, представляющей логическую модель данных. Сама модель и обновленная документация, включая словарь данных и реляционную схему связи таблиц, представляется для просмотра и анализа пользователям, которые должны убедиться, что она точно отображает предметную область.

3.4.3. Процедуры физического проектирования

Цель этапа физического проектирования – *описание конкретной реализации базы данных, размещаемой во внешней памяти компьютера.* Это описание структуры хранения данных и эффективных методов доступа к данным базы. При логическом проектировании отвечают на вопрос – *что* надо сделать, а при физическом – выбирается способ, *как* это сделать. Процедуры физического проектирования следующие.

1. *Проектирование таблиц базы данных средствами выбранной СУБД.* Осуществляется выбор реляционной СУБД, которая будет использоваться для создания базы данных, размещаемой на машинных носителях. Глубоко изучаются ее функциональные возможности по проектированию таблиц. Затем выполняется проектирование таблиц и схемы их связи в среде

СУБД. Подготовленный проект базы данных описывается в сопровождаемой документации.

2. *Реализация бизнес-правил в среде выбранной СУБД.* Обновление информации в таблицах может быть ограничено бизнес-правилами. Способ их реализации зависит от выбранной СУБД. Одни системы для реализации требований предметной области предлагают больше возможностей, другие– меньше. В некоторых системах вообще отсутствует поддержка реализации бизнес-правил. В таком случае разрабатываются приложения для реализации их ограничений.

Все решения, принятые в связи с реализацией бизнес-правил предметной области, подробно описываются в сопроводительной документации.

3. Проектирование физической организации базы данных. На этом шаге выбирается наилучшая файловая организация для таблиц. Выявляются транзакции, которые будут выполняться в проектируемой базе данных, и выделяются наиболее важные из них. Анализируется *пропускная способность транзакций* – количество транзакций, которые могут быть обработаны за заданный интервал времени, и *время ответа* – промежуток времени, необходимый для выполнения одной транзакции. Стремятся к повышению пропускной способности транзакций и уменьшению времени ответа. На основании указанных показателей принимаются решения об оптимизации производительности базы данных путем определения индексов в таблицах, ускоряющих выборку данных из базы, или снижения требований к уровню нормализации таблиц. Проводится оценка и, по возможности, минимизация дискового пространства, необходимого для размещения создаваемой базы данных.

Принятые решения по изложенным вопросам документируются.

4. Разработка стратегии защиты базы данных. База данных представляет собой ценный корпоративный ресурс, и организации ее защиты уделяется большое внимание. Для этого проектировщики должны иметь полное и ясное представление обо всех средствах защиты, предоставляемых выбранной СУБД.

5. Организация мониторинга функционирования базы данных и ее настройка. После создания физического проекта базы данных организуется непрерывное слежение за ее функционированием. Полученные сведения об уровне производительности базы данных используются для ее настройки. Для этого привлекаются и средства выбранной СУБД.

Решения о внесении любых изменений в функционирующую базу данных должны быть обдуманными и всесторонне взвешенными.

3.5. CASE-средства для автоматизированного проектирования реляционных баз данных

Для автоматизации проектирования и разработки информационных систем в 70-80-е гг. XX в. широко применялась так называемая *структурная методология* проектирования базы данных. При этом использовались графические средства описания с помощью схем и диаграмм. При ручной разработке баз данных такие графические модели разрабатывать и использовать весьма трудоемко.

Отмеченные обстоятельства послужили одной из причин появления программно-технологических средств, получивших название CASE-средств и реализующих CASE-технологию создания и сопровождения баз данных. Кроме структурной методологии, в ряде современных CASE-средств используется объектно-ориентированная методология проектирования.

Термин *CASE (Computer Aided Software Engineering)* дословно переводится как разработка программного обеспечения с помощью компьютера. В настоящее время этот термин получил более широкий смысл, означающий автоматизацию разработки различных информационных систем, в том числе и баз данных.

CASE-систему можно определить как набор CASE-средств, имеющих определенное функциональное предназначение и выполненных в рамках единого программного продукта.

CASE-технология обычно определяется как методология проектирования информационных систем плюс инструментальные средства, позволяющие наглядно моделировать предметную область, анализировать ее модель на всех этапах разработки и сопровождения информационной системы и разрабатывать приложения для пользователей.

3.5.1. Классификация CASE-средств

При классификации CASE-средств используют следующие признаки:

- ориентацию на этапы жизненного цикла;
- функциональную полноту;
- тип используемой модели;
- степень независимости от СУБД;
- допустимые платформы.

Рассмотрим классификацию CASE-средств по наиболее часто используемым признакам.

Классификация CASE-средств по ориентации на этапы жизненного цикла. По этому признаку выделяют следующие основные типы CASE-средств:

– *средства анализа,* предназначенные для построения и анализа моделей предметной области, например, Design/IDEF (Meta Software) и BPwin (Logic Works);

– *средства анализа и проектирования,* обеспечивающие создание проектных спецификаций, например, Vantage Team Builder (Cayenne), Silverrun (Computer Systems Advisers Inc.), PRO-IV (McDonnell Douglas) и CASE.Аналитик (МакроПроджект);

– *средства проектирования баз данных,* обеспечивающие моделирование данных и разработку схем баз данных для основных СУБД, например: ERwin (Logic Works), S-Designor (SPD), DataBase Designer (ORACLE);

– *средства разработки приложений,* например: Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Centura) и Delphi (Borland).

Классификация CASE-средств по функциональной полноте. По этому признаку CASE-системы и средства можно условно разделить на следующие типы:

– системы, предназначенные для решения частных задач на одном или нескольких этапах жизненного цикла, например, ERwin (Logic Works), S-Designor (SPD), CASE.Аналитик (МакроПроджект) и Silverrun (Silverrun Technologies);

– *интегрированные системы,* поддерживающие весь жизненный цикл ИС, например система Vantage Team Builder (Cayenne) и система Designer/2000 с системой разработки приложений Developer/2000 (ORACLE).

Классификация CASE-средств по типу используемых моделей. По этому признаку CASE-системы условно можно разделить на три основные разновидности:

– структурные;

– объектно-ориентированные;

– комбинированные.

Исторически первые *структурные* CASE-системы основаны на методах структурного и модульного программирования, структурного анализа и синтеза, например, система Vantage Team Builder (Cayenne).

Объектно-ориентированные методы и CASE-системы получили массовое распространение с начала 90-х годов. Они позволяют сократить сроки

разработки, а также повысить надежность и эффективность функционирования ИС. Примерами объектно-ориентированных CASE-систем являются Rational Rose (Rational Software) и Object Team (Cayenne).

Комбинированные инструментальные средства поддерживают одновременно структурные и объектно-ориентированные методы, например Designer/2000 (ORACLE).

Классификация CASE-средств по степени независимости от СУБД. По этому признаку CASE-системы можно разделить на две группы:

- независимые системы;
- системы, встроенные в СУБД.

Независимые CASE-системы поставляются в виде автономных систем, не входящих в состав конкретной СУБД. К числу независимых CASE-систем относятся S-Designor (SDP, Powersoft), ERwin (LogicWorks) и Silverrun (Computer Systems Advisers Inc.).

Встроенные CASE-системы обычно поддерживают главным образом формат баз данных СУБД, в состав которой они входят. При этом возможна поддержка и других форматов баз данных. Примером встроенной системы является Designer/2000, входящая в состав СУБД ORACLE.

3.5.2. Общая характеристика и функциональные возможности CASE-системы ERwin

ERwin – средство разработки структуры базы данных. ERwin сочетает графический интерфейс Windows, инструменты для построения ER-диаграмм, редакторы для создания логического и физического описания модели данных и прозрачную поддержку ведущих реляционных СУБД и настольных баз данных.

Процесс построения информационной модели состоит из следующих шагов:

1) определение сущностей;
2) определение зависимостей между сущностями;
3) задание первичных и альтернативных ключей;
4) определение атрибутов сущностей;
5) приведение модели к требуемому уровню нормальной формы;
6) переход к физическому описанию модели:
   - назначение соответствий *имя сущности – имя таблицы*, *атрибут сущности – атрибут таблицы*;
   - задание триггеров, процедур и ограничений;
7) генерация базы данных.

ERwin создает визуальное представление (модель данных) для решаемой задачи. Это представление может использоваться для детального анализа, уточнения и распространения как части документации, необходимой в цикле разработки. Кроме того, ERwin автоматически создает базу данных (таблицы, индексы, хранимые процедуры, триггеры для обеспечения ссылочной целостности и другие объекты, необходимые для управления данными).

Словарь терминов

CASE-система – набор CASE-средств, имеющих определенное функциональное предназначение и выполненных в рамках единого программного продукта.

CASE-технология – методология проектирования информационных систем и инструментальные средства, позволяющие наглядно моделировать предметную область, анализировать ее модель на всех этапах разработки и сопровождения информационной системы и разрабатывать приложения для пользователей.

ER-диаграммы – средство моделирования структуры данных предметной области модели «сущность-связь».

Жизненный цикл базы данных – процесс проектирования, реализации и поддержки базы данных.

Концептуальное проектирование – создание для анализируемой части предприятия концептуальной модели данных.

Логическое проектирование – перевод концептуальной модели в логическую модель, основанную на структурах, характерных для выбранной модели данных.

Модель «сущность-связь» – средство моделирования предметной области на этапе концептуального проектирования является.

Нормализация – процесс декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, удовлетворяющих требованиям следующей нормальной формы.

Представление семантического объекта – часть объекта, видимая для конкретного приложения.

Проектирование – процесс создания описаний нового или модернизируемого объекта (изделия, процесса), достаточных для изготовления или реализации этого объекта в заданных условиях.

Проектирование базы данных – процесс создания проекта базы данных, предназначенной для поддержки функционирования экономического объекта и способствующей достижению его целей.

Семантическая объектная модель – средство моделирования данных.

Семантический объект – объект, который моделирует смысл пользовательских данных.

Транзакция – некоторое событие реального мира; может состоять из нескольких операций, однако с точки зрения пользователя эти операции представляют собой единое целое, переводящее базу данных из одного непротиворечивого состояния в другое.

Физическое проектирование – расширение логической модели характеристиками, необходимыми для определения способов физического хранения базы данных, типа устройств для хранения, методов доступа к данным базы, требуемого объема памяти, правил сопровождения базы данных и др.

Вопросы и задания для самоконтроля

1. Что собой представляет процесс проектирования как таковой?
2. Что собой представляет проектирования базы данных?
3. Какие требования необходимо учитывать при проектировании базы данных?
4. Что собой представляет жизненный цикл базы данных? Из каких этапов он состоит?
5. Охарактеризуйте этап планирования БД.
6. Охарактеризуйте этап определения требований к системе.
7. Охарактеризуйте этап сбора и анализа требований пользователей к проектируемой БД.
8. Охарактеризуйте этап проектирования БД.
9. Охарактеризуйте этап разработки приложений БД.
10. Охарактеризуйте этап реализации БД.
11. Охарактеризуйте этап загрузки данных в БД.
12. Охарактеризуйте этап тестирования БД.
13. Охарактеризуйте этап эксплуатации и сопровождения БД.
14. Какие методы проектирования баз данных существуют?
15. Какой метод проектирования БД используется на этапе концептуального проектирования.
16. Перечислите и охарактеризуйте основные понятия ER-диаграммы.
17. По каким правилам выполняется преобразование ER-модели в реляционную модель?
18. Что собой представляет процесс нормализации?
19. Каковы требования первой, второй и третьей нормальных форм?
20. Что собой представляет семантическая объектная модель?
21. Что собой представляет семантический объект?

22. Назовите и охарактеризуйте типы атрибутов семантических объектов.

23. Что собой представляют максимальное и минимальное кардинальные числа атрибута семантического объекта?

24. Назовите и охарактеризуйте следующие характеристики семантических объектов: объектный идентификатор, домен, физическое описание, семантическое описание.

25. Семантические объекты каких типов существуют?

26. Что такое «представление семантического объекта»?

27. Какие процедуры выполняются на этапе концептуального проектирования.

28. Какие процедуры выполняются на этапе логического проектирования?

29. Какие процедуры выполняются на этапе физического проектирования?

30. Что собой представляет CASE-система?

31. Что собой представляет CASE-технология?

32. Проведите классификацию CASE-средств.

33. Охарактеризуйте CASE-систему ERwin.

4. Системы управления базами данных

4.1. Понятие СУБД

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение к ней доступа пользователей осуществляются централизованно с помощью специального программного инструментария – *системы управления базами данных (СУБД)*.

Система управления базами данных – это совокупность языковых и программных средств, предназначенная для создания, ведения и совместного использования БД многими пользователями. Современная СУБД включает в себя средства создания БД, средства работы с данными и сервисные средства.

4.2. Языковые средства СУБД

Функциональные возможности СУБД становятся доступными для конечных пользователей, разработчиков приложений, персонала администрирования благодаря наличию у нее комплекса языковых средств, который включает в себя:

- язык описания данных (ЯОД);
- язык манипулирования данными (ЯМД);
- язык запросов (ЯЗ).

Язык описания данных. Описание базы данных средствами ЯОД называется схемой базы данных. Оно включает описание структуры базы данных и налагаемых на нее ограничений целостности данных. Помимо указанных функций, ЯОД некоторых СУБД обеспечивают также возможности определения в схеме ограничений доступа к данным или полномочий пользователей. ЯОД включает язык логического описания данных и язык описания физических данных.

Язык логического описания данных предусматривает:

- идентификацию файлов БД, объектов БД, полей БД и их типов;
- определяет длину записей, полей, порядок полей в записи, диапазоны допустимых значений полей и др.

Используя этот язык, пользователь формирует свой взгляд на БД – создается логическая БД.

Язык описания физических данных определяет способы размещения данных на машинных носителях, способы их адресации и поиска. Посредством этого языка формируется взгляд системы на БД – создается физическая БД.

Язык манипулирования данными. ЯМД дает возможность реорганизации данных в базе (добавления новых, удаления ненужных, обновления существующих).

Язык запросов. С появлением интерфейсов конечных пользователей СУБД возникла потребность в языках, с помощью которых можно было бы формулировать запросы к системе базы данных. Такие языки стали называть *языками запросов*. ЯЗ обеспечивает доступ к данным и их извлечение по запросам пользователей.

Языковые средства могут быть реализованы различными способами:

- синтаксическими конструкциями (командами);
- меню;
- диалоговыми сценариями;
- таблицами.

У многих СУБД возможности описания, манипулирования и чтения данных объединены в единых синтаксических рамках – рамках языка SQL (Structured Query Language), широко используемого в качестве языка баз данных.

Часто пользователю требуется более сложная аналитическая обработка данных базы, недостигаемая через систему меню СУБД. В этом случае приходится разрабатывать прикладные программы. Для их создания СУБД имеют *встроенный язык программирования*.

Благодаря языковым средствам обеспечивается доступ пользователей к БД в абстрактных терминах, не связанных со способами хранения данных в компьютере.

Программные средства СУБД обеспечивают работу с физической БД и выполнение всех ее функций.

4.3. Функциональные возможности и производительность СУБД

Современные СУБД обладают следующими функциональными возможностями:

- создание БД, в которой интегрированы данные многих пользователей с целью удовлетворения их информационных потребностей;
- обновление хранящихся в ней данных;
- быстрое извлечение из БД необходимых данных по запросам пользователей;
- выполнение вычислений над данными;
- создание экранных форм, обеспечивающих удобство работы с данными;
- вывод данных из базы в отчетах в виде, удобном для восприятия;
- разработка приложений;
- экспорт данных в другие БД и импорт данных из них;
- публикации данных в Internet.

Перечисленные средства ориентированы на непрофессиональных пользователей.

СУБД обеспечивают также:

1) *ведение системного каталога*, доступного конечным пользователям.

Системный каталог, или *словарь данных*, является хранилищем информации, описывающей данные в базе данных (по сути, это «данные о данных», или метаданные). Обычно в системном каталоге хранятся следующие сведения:

- имена, типы и размеры элементов данных;
- имена связей;
- накладываемые на данные ограничения поддержки целостности;
- имена санкционированных пользователей, которым предоставлено право доступа к данным;
- внешняя, концептуальная и внутренняя схемы и отображения между ними;
- статистические данные, например частота транзакций и счетчики обращений к объектам базы данных.

Наличие системного каталога позволяет:

- централизовано хранить информацию о данных, что обеспечивает контроль доступа к этим данным и любому другому ресурсу;
- легко обнаружить избыточность и противоречивость описания отдельных элементов данных;
- протоколировать внесение в базу данных изменений и определить их последствия еще до их внесения, поскольку в системном каталоге зафиксированы все существующие элементы данных, установленные между ними связи, а также все их пользователи;
- усилить меры обеспечения безопасности;
- выполнять аудит сохраняемой информации.

2) *поддержку логической целостности БД с помощью механизма транзакций*:

- *целостность БД* есть свойство базы данных, означающее, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область информация. Поддержание целостности БД включает проверку целостности и ее восстановление в случае обнаружения противоречий в базе данных. Целостное состояние БД описывается с помощью *ограничений целостности* в виде условий, которым должны удовлетворять хранимые в базе данные. Примером таких условий может служить ограничение диапазонов возможных значений атрибутов объектов, сведения о которых хранятся в БД, или отсутствие повторяющихся записей в таблицах реляционных БД;

– *транзакция* – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Если транзакция успешно выполняется, то СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти. Если по каким-либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении) транзакция остается незавершенной, то она отменяется. Транзакции присущи три основных свойства: атомарность (выполняются все входящие в транзакцию операции или ни одна), сериализуемость (отсутствует взаимное влияние выполняемых в одно и то же время транзакций), долговечность (даже крах системы не приводит к утрате результатов зафиксированной транзакции).

3) *поддержку параллельной работы*. СУБД должна иметь механизм, который гарантирует корректное обновление базы данных при параллельном выполнении операций обновления многими пользователями. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако, когда два или больше пользователей одновременно получают доступ к базе данных, легко может возникнуть конфликт с нежелательными последствиями;

4) *защиту данных от несанкционированного доступа*;

5) *надежность хранения данных во внешней памяти*:

– под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Очевидно, что в случае для восстановления БД после сбоя нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение *журнала изменений БД*.

Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью. Сюда поступают записи обо всех изменениях основной части БД. Во всех случаях придерживаются стратегии «упреждающей» записи в журнал (так называемого протокола *Write Ahead Log – WAL*). Эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память

журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается эта стратегия, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

6) *поддержку обмена данными*. СУБД должны поддерживать работу в локальной сети, чтобы вместо нескольких разрозненных баз данных для каждого отдельного пользователя можно было бы установить одну централизованную базу данных и использовать ее как общий ресурс для всех существующих пользователей. При этом предполагается, что не база данных должна быть распределена в сети, а удаленные пользователи должны иметь возможность доступа к централизованной базе данных. Такая топология называется *распределенной обработкой*;

7) *вспомогательные функции*. СУБД должна предоставлять некоторый набор различных вспомогательных функций, обычно предназначенных для администрирования базы данных, импорта и экспорта БД, мониторинга характеристик функционирования и использования базы данных, статистического анализа (оценка производительности или степени использования базы данных), реорганизации индексов, перераспределения памяти.

На заре технологии БД было легко установить границу между СУБД и приложением, т. к. приложения были отдельными программами, которые вызывали СУБД. Сегодня, в особенности с появлением СУБД для ПК, эта граница стала несколько размытой. Поэтому будем считать, что все формы, отчеты, меню, как и программный код, содержащийся в них, входят в приложение БД. Автономные программы, вызывающие СУБД, также являются частью приложения. Все структуры, правила и ограничения, касающиеся таблицы, а также определения связей относятся к ведению СУБД и входят в состав БД.

Приложения выполняют пять основных функций:

- создание, чтение, обновление и удаление представлений;
- форматирование представлений;
- реализация ограничений;
- обеспечение механизмов безопасности и контроля;
- реализация логики обработки информации.

Производительность СУБД оценивается:

- временем выполнения запросов;
- скоростью поиска информации в неиндексированных полях;
- временем выполнения операций импортирования базы данных из других форматов;

– скоростью создания индексов и выполнения таких массовых операций, как обновление, вставка, удаление данных;

– максимальным числом параллельных обращений к данным в многопользовательском режиме;

– временем генерации отчета.

На производительность СУБД оказывают влияние два фактора:

– СУБД, которые следят за соблюдением целостности данных, несут дополнительную нагрузку, которую не испытывают другие программы;

– производительность собственных прикладных программ сильно зависит от правильного проектирования и построения базы данных.

Т. о., благодаря своим развитым функциональным возможностям, СУБД используются в качестве мощного инструментального средства для создания и ведения баз данных автоматизированных информационных систем, позволяющего сокращать сроки их разработки, экономить трудовые, материальные и финансовые ресурсы.

4.4. Архитектура СУБД

В среде СУБД можно выделить следующих пять основных компонентов:

1) аппаратное обеспечение;
2) программное обеспечение;
3) данные;
4) процедуры;
5) пользователи.

4.4.1. Аппаратное обеспечение

Для работы СУБД и приложений необходимо некоторое аппаратное обеспечение. Одни СУБД предназначены для работы только с конкретными типами операционных систем или оборудования, другие могут работать с широким кругом аппаратного обеспечения и различными операционными системами. Для работы СУБД обычно требуется некоторый минимум оперативной и дисковой памяти, но такой минимальной конфигурации может оказаться совершенно недостаточно для достижения приемлемой производительности системы.

4.4.2. Программное обеспечение

Этот компонент включает операционную систему, программное обеспечение самой СУБД, прикладные программы, включая и сетевое программное обеспечение, если СУБД используется в сети. Обычно приложения создаются

на языках программирования третьего поколения, таких как *C*, *COBOL*, *Fortran*, *Ada* или *Pascal*, или на языках четвертого поколения, таких как *SQL*, операторы которых внедряются в программы на языках третьего поколения. СУБД может иметь свои собственные инструменты четвертого поколения, предназначенные для быстрой разработки приложений с использованием встроенных непроцедурных языков запросов, генераторов отчетов, форм, графических изображений и даже полномасштабных приложений.

4.4.3. Данные

Данные – наиболее важный компонент с точки зрения конечных пользователей. База данных содержит как рабочие данные, так и метаданные, т. е. «данные о данных».

4.4.4. Процедуры

Процедуры, к которым относят инструкции и правила, которые должны учитываться при проектировании и использовании базы данных:

- регистрация в СУБД;
- использование отдельного инструмента СУБД или приложения;
- запуск и остановка СУБД;
- создание резервных копий СУБД;
- обработка сбоев аппаратного и программного обеспечения, включая процедуры идентификации вышедшего из строя компонента, исправления отказавшего компонента (например, посредством вызова специалиста по ремонту аппаратного обеспечения), а также восстановления базы данных после устранения неисправности;
- изменение структуры таблицы, реорганизация базы данных, размещенной на нескольких дисках, способы улучшения производительности и методы архивирования данных на вторичных устройствах хранения.

4.4.5. Пользователи

К числу пользователей относятся: клиенты БД, администратор БД, программисты. Более подробно этот компонент рассматривается в теме 9 «Администрирование базы данных».

СУБД значительно различаются по характеристикам и функциям. Логически в них можно выделить три компоненты (рис. 4.1).

Подсистема средств проектирования представляет собой набор инструментов, упрощающих проектирование и реализацию баз данных и их приложений. Как правило, этот набор включает в себя средства для создания таблиц, форм, запросов и отчетов. В СУБД имеются также языки программирования и интерфейсы для них. Например, в Microsoft Office Access–

макроязык, не требующий глубокого знания программирования, и версия языка Basic – Visual Basic for Application.

Подсистема обработки обеспечивает обработку компонентов приложений, созданных с помощью средств проектирования. Например, в Access имеется компонент, реализующий построение формы и связывающий элементы формы с данными таблиц.

Третий компонент СУБД – ее *ядро (DBMS Engine)* – выполняет функцию посредника между подсистемой средств проектирования и обработки и данными. Ядро СУБД получает запросы от двух других компонентов, выраженные в терминах таблиц, строк и столбцов, и преобразует эти запросы в команды операционной системы, выполняющие запись и чтение данных с физического устройства.

Кроме того, ядро СУБД участвует в управлении транзакциями, блокировке, резервном копировании и восстановлении.

Microsoft представляет два различных ядра для Microsoft Office Access: Jet Engine и SQL Server. Ядро *Jet Engine* используется для персональных и коллективных баз данных небольшого объема. Ядро *SQL Server* предназначено для крупных баз данных.

Рис. 4.1. Компоненты СУБД

4.5. Классификация СУБД

4.5.1. Общая классификация СУБД

На самом общем уровне все СУБД можно разделить на две группы:

– *профессиональные (промышленные)*, которые представляют собой программную основу для разработки автоматизированных систем управления

крупными экономическими объектами. На их базе создаются комплексы управления и обработки информации крупных предприятий, банков или даже целых отраслей. В настоящее время характерными представителями профессиональных СУБД являются такие программные продукты: Oracle, DB2, Sybase, Informix, Ingres, Progress;

– *персональные (настольные)*. Это программное обеспечение, ориентированное на решение задач локального пользователя или компактной группы пользователей и предназначенное для использования на персональном компьютере, это объясняет их второе название – *настольные*. К ним относятся DBASE, FoxBase, FoxPro, Clipper, Paradox, Access.

В настоящее время среди СУБД выделяют СУБД (условно говоря) промежуточные между профессиональными и персональными. Это SQL Windows/SQL Base, Interbase, Microsoft SQL Server.

4.5.2. Классификация СУБД по поддерживаемому типу модели данных

Важнейшим классификационным признаком СУБД является *тип модели данных*, поддерживаемый СУБД. По этому признаку СУБД делятся на:

– *иерархические*. Первой иерархической СУБД была система *IMS (Information Management System)* компании IBM, коммерческое распространение которой началось в 1968 г.;

– *сетевые*. Первой сетевой СУБД считается система *IDS (Integrated Data Store)*, разработанная компанией General Electric немного позже системы IMS;

– *реляционные*. Первые коммерческие реляционные СУБД от компаний IBM, Oracle Corporation, Relation Technology и др. поставщиков появились в начале 80-х годов. Реляционные СУБД просты в использовании, хорошо приспособлены для работы в архитектуре «клиент-сервер», позволяют параллельную обработку БД, хорошо приспособлены к графическим пользовательским интерфейсам. Реляционные СУБД продолжают совершенствоваться, предоставляя пользователю возможность решать все более сложные задачи;

– *объектно-ориентированные*. В основе объектно-ориентированных СУБД лежит объектно-ориентированная модель обработки данных;

– *объектно-реляционные*. Объектно-реляционные СУБД продолжают использовать стандартный язык запросов для реляционных БД – SQL, но с объектными расширениями;

– *многомерные*, в основе которых лежит многомерная модель данных.

4.5.3. Классификация СУБД по степени универсальности

По степени универсальности различают СУБД общего и специального назначения.

СУБД общего назначения не ориентированы на какую-либо конкретную предметную область или информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторых моделях компьютеров в определенной операционной системе, и поставляется многим пользователям как коммерческий продукт. СУБД общего назначения обладают средствами настройки на работу с конкретной БД в условиях конкретного применения. Развитые функциональные возможности таких СУБД обеспечивают безболезненную эволюцию построенных на их основе автоматизированных информационных систем в рамках их жизненного цикла.

Однако в некоторых случаях существующие СУБД общего назначения не позволяют добиться требуемой производительности и/или удовлетворить заданные ограничения по объему памяти, предоставляемой для хранения БД. Тогда приходится разрабатывать *специализированную СУБД* для данного конкретного применения. Примером специализированной СУБД может быть система IMBASE, используемая для автоматизации проектных и конструкторских разработок. Эта СУБД содержит информацию о стандартных конструктивных элементах, материалах и других элементах, используемых для проектирования сложных механических устройств (узлов, агрегатов).

4.5.4. Дополнительные признаки классификации СУБД

В качестве классификационных признаков СУБД можно рассматривать также:

– среду функционирования СУБД (платформу) – класс компьютеров и операционных систем, под управлением которых работает СУБД;

– наличие диалоговых и инструментальных средств конструирования объектов БД;

– возможности встроенного языка СУБД;

– использование OLE-технологии – взаимодействие объектов БД с объектами других приложений: табличных и текстовых процессоров, графических редакторов и др.;

– возможности работы с нетрадиционными данными (данными, не являющимися текстом, числами и датами);

– обеспечение интеграции данных из баз, созданных в разных СУБД, и др.

4.6. Режимы работы пользователя с СУБД

Все современные СУБД имеют графический пользовательский интерфейс, через который возможна работа пользователя с СУБД в трех режимах:

1) через меню системы;
2) в командном режиме;
3) в программном режиме.

Режим работы *через меню системы* обеспечивает взаимодействие пользователя с БД в интерактивном режиме. Он реализуется чаще всего в виде различных диалоговых окон, с помощью которых пользователь постепенно уточняет, какие действия он хочет выполнить и какую информацию получить из БД. Для этого не надо знать языка СУБД.

Командный режим обеспечивает диалог с БД на уровне синтаксических конструкций языка СУБД. Этот режим требует определенной подготовки пользователя, но обеспечивает более быстрый доступ к ресурсам БД.

Программный режим обеспечивает организацию доступа к данным и управление ими из прикладных программ.

В последние годы широкое распространение получили компьютерные сети. Пользователи компьютерных сетей могут работать с СУБД в *однопользовательском* и *многопользовательском* режимах, обеспечивающих доступ к БД соответственно одного из них и многих одновременно.

Словарь терминов

Журнал базы данных – особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью. Сюда поступают записи обо всех изменениях основной части БД.

Подсистема обработки – обеспечивает обработку компонентов приложений, созданных с помощью средств проектирования.

Подсистема средств проектирования – набор инструментов, упрощающих проектирование и реализацию баз данных и их приложений.

Система управления базами данных – совокупность языковых и программных средств, предназначенная для создания, ведения и совместного использования БД многими пользователями.

Системный каталог (словарь данных) – хранилище информации, описывающей данные в базе.

Ядро СУБД – выполняет функцию посредника между подсистемой средств проектирования и обработки и данными.

Вопросы и задания для самоконтроля

1. Что собой представляет система управления базами данных?
2. Что входит в комплекс языковых средств СУБД?
3. Охарактеризуйте язык описания данных СУБД.
4. Охарактеризуйте язык манипулирования данными СУБД.
5. Охарактеризуйте язык запросов СУБД.
6. Каковы функциональные возможности СУБД?
7. Какие компоненты можно выделить в архитектуре СУБД?
8. Охарактеризуйте компонент «аппаратное обеспечение СУБД».
9. Охарактеризуйте компонент «программное обеспечение СУБД».
10. Охарактеризуйте компонент «данные».
11. Охарактеризуйте компонент «процедуры».
12. Охарактеризуйте компонент «пользователи».
13. Проведите классификацию СУБД по поддерживаемому типу модели данных.
14. Проведите классификацию СУБД по степени универсальности.
15. В каких режимах может работать с СУБД пользователь?

5. СУБД Microsoft Office Access 2013

5.1. Общая характеристика

СУБД Microsoft Access 2013 (в дальнейшем Microsoft Access) предназначена для работы с реляционными базами данных. Эта СУБД входит в программный комплекс Microsoft Office 2013.

Microsoft Access располагает инструментальными средствами для создания:

- локальной БД;
- централизованной БД в локальной сети с файловым сервером;
- проектов (клиентских приложений, работающих с базами данных Microsoft SQL Server).

Microsoft Access поддерживает механизм OLE – связывание и внедрение объектов различных приложений Windows в БД.

Позволяет импортировать данные таблиц:

- других баз данных Microsoft Access;
- баз данных dBase, Microsoft FoxPro, Paradox, Oracle, Microsoft SQL Server;
- табличного процессора Microsoft Excel.

Внешними для Microsoft Access могут быть также: данные почтовой программы Microsoft Exchange; таблицы и списки HTML на сервере локальной, корпоративной сети (сети масштаба предприятия), сети Интернет.

Microsoft Access может экспортировать объекты БД в другие приложения.

Microsoft Access позволяет осуществлять:

- восстановление БД;
- сжатие БД;
- защиту БД.

Microsoft Access дает возможность конвертировать БД из предыдущих версий в текущую и наоборот.

5.2. Объекты БД и их размещение

СУБД Access ориентирована на работу с объектами БД, к которым относятся таблицы, запросы, формы, отчеты, страницы доступа к данным, макросы и модули.

Таблицы – это основные объекты БД, предназначенные для хранения информации. По терминологии СУБД *строки таблицы* – это *записи*, а *столбцы* – *поля* БД. Записи идентифицируются по некоторой уникальной характеристике, включающей одно или несколько полей и называемой *ключом*.

Запросы – служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию. С помощью запросов можно выполнять преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое наполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и др.

Формы – это средства для ввода данных. Смысл их в том, чтобы 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 др.) для автоматизации ввода. Также с помощью форм данные можно просматривать и редактировать.

Отчеты – предназначены для формирования выходного документа, предназначенного для вывода на печать.

Макросы – это последовательности макрокоманд для автоматизации выполнения операций в среде Microsoft Access без программирования.

Модули – это программы для работы с БД, написанные на языке Visual Basic for Applications (VBA).

Объекты БД могут быть объединены в именованные группы объектов по функциональному или иному признаку.

Все объекты можно хранить в одном файле – файле БД с расширением *.accdb*. Это упрощает их перенос с компьютера на компьютер, облегчает создание связанных объектов, проверку целостности данных.

Отчеты можно экспортировать в файлы Microsoft Excel, текстовые файлы, файлы формата *.pdf* и *.xps*.

С целью защиты форм, отчетов и модулей VBA база данных может быть сохранена в файле приложения с расширением *.mde*. При этом БД сжимается, оптимизируется использование памяти и повышается быстродействие БД.

Словарь терминов

Microsoft Access – система управления реляционными базами данных.

Запросы Microsoft Access – служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде.

Макросы Microsoft Access – это последовательности макрокоманд для автоматизации выполнения операций в среде Microsoft Access без программирования.

Модули Microsoft Access – это программы для работы с базой данных, написанные на языке Visual Basic for Applications (VBA).

Отчеты Microsoft Access – предназначены для формирования выходного документа, предназначенного для вывода на печать.

Таблицы Microsoft Access – это основные объекты БД, предназначенные для хранения информации.

Формы Microsoft Access – средства для ввода и просмотра данных.

Вопросы и задания для самоконтроля

1. Какими инструментальными средствами располагает Microsoft Access 2013?

2. Назовите и охарактеризуйте объекты базы данных в Microsoft Access 2013.

6. Технологии работы с базой данных в СУБД Microsoft Office Access 2013

6.1. Пользовательский интерфейс и основные настройки Microsoft Access 2013

В Microsoft Office Access 2013 имеется три основных компонента пользовательского интерфейса, которые формируют среду, в которой создаются и используются базы данных:

1. *Лента.*
2. *Представление Backstage.*
3. *Область навигации.*

6.1.1. Лента

Лента – это область в верхней части окна приложения. Она является основным командным интерфейсом в Access 2013. Лента содержит:

– основные вкладки с группами наиболее часто используемых команд;

– контекстные вкладки, которые появляются только тогда, когда их использование допустимо;

– панель быстрого доступа – небольшую панель инструментов, на которую можно добавить самые нужные команды (рис. 6.3).

Некоторые кнопки на вкладках ленты предоставляют выбор действий (если имеется присоединенная кнопка ▾), а другие позволяют выполнить определенную команду.

Основные вкладки. В Microsoft Access 2013 основные вкладки команд – «Файл», «Главная», «Создание», «Внешние данные» и «Работа с базами данных». Каждая вкладка содержит группу связанных команд, которые могут открывать другие новые элементы интерфейса, например *коллекцию* – новый элемент управления, позволяющий выбирать варианты визуально.

Команды ленты также соответствуют объекту, активному в настоящее время.

Пример. Если открыть таблицу в режиме таблицы и нажать кнопку «Форма» в группе «Формы» на вкладке «Создание», приложение Access создаст форму на основе активной таблицы. Иначе говоря, имя активной таблицы будет указано в свойстве формы «Источник записей».

Более того, некоторые вкладки ленты появляются только в определенном контексте.

Пример. Вкладка «Конструктор» появляется только при открытии объекта в режиме конструктора.

В приведенной ниже таблице представлены вкладки и команды, находящиеся на каждой из них. Набор доступных вкладок и команд изменяется в зависимости от выполняемых действий.

Таблица 6.1

Вкладки и соответствующие им команды

Вкладка	Команды
Главная	Выбор другого представления (режима)
	Копирование и вставка данных из буфера обмена
	Настройка параметров шрифта
	Выбор выравнивания текста
	Работа с записями (обновление, создание, сохранение, удаление, итоги, орфография, дополнительно)
	Сортировка и фильтрация записей
	Поиск записей
Создание	Создание пустой таблицы
	Создание таблицы на основе шаблона
	Создание списка на сайте SharePoint, а также связанной с этим списком таблицы в текущей базе данных
	Создание пустой таблицы в режиме конструктора
	Создание формы на основе активной таблицы или запроса
	Создание сводной таблицы или диаграммы
	Создание отчета на основе активной таблицы или запроса
	Создание запроса, макроса, модуля или модуля класса
Внешние данные	Импорт или связывание внешних данных
	Экспорт данных
	Сбор и обновление данных по электронной почте
	Создание сохраненных операций импорта и экспорта
	Запуск диспетчера связанных таблиц
Работа с базами данных	Перенос некоторых или всех частей базы данных на новый или существующий сайт SharePoint
	Запуск редактора Visual Basic или выполнение макроса
	Создание и просмотр отношений между таблицами
	Показ или скрытие зависимостей объектов
	Запуск архивариуса или анализ производительности
	Перемещение данных в Microsoft SQL Server или базу данных Access (только таблицы)
	Управление надстройками Access
	Создание или изменение модуля VBA

Контекстные вкладки. Помимо стандартных вкладок команд в Access 2013 есть также *контекстные вкладки*. Они отображаются рядом с основными вкладками в зависимости от контекста (т. е. от того, с каким объектом работает пользователь и какие действия он выполняет). Такие вкладки содержат команды и функциональные элементы, необходимые для работы *в определенном контексте*.

Пример. При открытии таблицы в режиме конструктора контекстные вкладки содержат команды, которые используются для работы с таблицей только в этом режиме. Так, при открытии таблицы в режиме конструктора рядом с вкладкой «Работа с базами данных» появляется контекстная вкладка «Конструктор». Если открыть вкладку «Конструктор», на ленте появятся команды, доступные для объекта только в режиме конструктора.

Иногда требуется выделить на экране дополнительное пространство для работы. В этих случаях можно свернуть ленту и оставить только строку с вкладками. Чтобы скрыть ленту, необходимо дважды щелкнуть активную вкладку. Чтобы опять отобразить ленту, повторно выполняется двойной щелчок левой кнопкой мыши по активной вкладке.

Панель быстрого доступа. Панель быстрого доступа обеспечивает доступ к командам одним щелчком мыши. Набор по умолчанию включает команды *Сохранение*, *Отмена* и *Возврат*, при этом можно настроить панель быстрого доступа для добавления в нее наиболее часто используемых команд. Можно также изменить расположение и увеличить размер этой панели инструментов. В стандартном уменьшенном виде она находится рядом с вкладками ленты. Если выбран крупный размер, она располагается под лентой во всю ее длину.

Для настройки панели быстрого доступа нужно выполнить следующие действия:

1) щелкнуть присоединенную кнопку со стрелкой в правой части панели ▾;

2) в появившемся меню выбрать команду, которую необходимо добавить;

3) если нужной команды нет в списке, выбрать пункт *Другие команды* – откроется окно диалога «Параметры Access»;

4) в диалоговом окне «Параметры Access» выбрать команду или команды, которые требуется добавить, и нажать кнопку Добавить >>.

5) для удаления команды выделить ее в списке, расположенном справа, и нажать кнопку << Удалить;

6) по завершении нажать кнопку ОК.

6.1.2. Представление Backstage

Режим *Backstage* содержит команды и сведения, применимые ко всей базе данных, например, *Сжать и восстановить* (рис. 6.1). Переход в представление *Backstage* выполняется путем выбора вкладки «Файл» на ленте. Также представление *Backstage* открывается при запуске приложения Access, если при этом не открывается база данных (например, при запуске приложения Access из меню «Пуск»).

В представлении *Backstage* можно создавать или открывать базы данных, публиковать их в Интернете на сервере SharePoint Server и выполнять многие задачи обслуживания файлов и баз данных.

Рис 6.1. Представление *Backstage* Access 2013

6.1.3. Область навигации

Область навигации – это область в левой части окна Access (рис. 6.2), предназначенная для работы с объектами базы данных. Область навигации позволяет упорядочить объекты базы данных и является основным средством открытия или изменения объектов базы данных.

Объекты в области навигации упорядочены по категориям и группам. Пользователи могут выбрать различные параметры упорядочения, а также создать собственную схему упорядочения. По умолчанию в новой базе данных для упорядочения используется категория типа объекта, которая содержит группы, соответствующие различным типам объектов базы данных.

Область навигации можно уменьшить или скрыть, но она не перекрывается открытыми объектами базы данных.

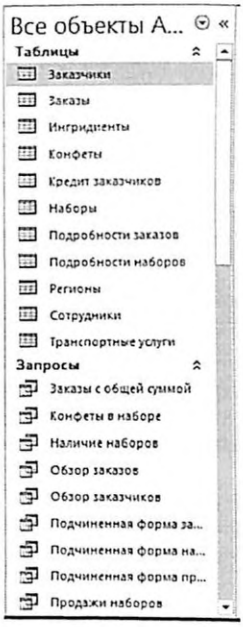


Рис 6.2. Область навигации Microsoft Access 2013

Чтобы открыть объект БД (например, таблицу, форму или отчет) можно воспользоваться одним из перечисленных способов:

– выполнить двойной щелчок левой кнопкой мыши на объекте в области навигации;

– выделить объект в области навигации и нажмите клавишу Enter;

– в области навигации щелкнуть объект правой кнопкой мыши и выберите в контекстном меню команду *Открыть*.

Все объекты базы данных в области навигации делятся на категории, которые содержат группы. Некоторые категории являются встроенными, но можно создавать и пользовательские группы.

6.1.4. Строка состояния

Вдоль нижней границы окна может отображаться строка состояния. Эта строка предназначена для вывода текста сообщения, связанного с текущим режимом, состоянием объекта, выполняющейся программой (рис. 6.3).

Элементы управления в правой части строки состояния позволяют быстро переключать различные режимы активного объекта.

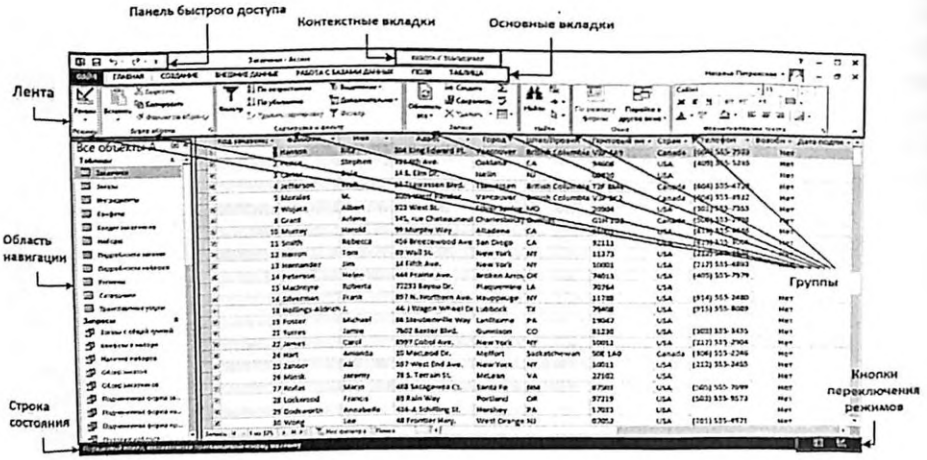


Рис. 6.3. Окно Microsoft Access 2013

6.2. Справочная система Access

Справочная система Access позволяет получить недостающие сведения и ответы на возникающие в ходе работы вопросы. Для этого нужно нажать клавишу F1 или щелкнув вопросительный знак в правой части ленты (рис. 6.4).

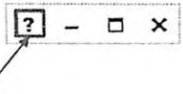


Рис. 6.4. Точка входа в справочную систему

Справочной системой при подключении к Интернету обеспечивается поиск на сайте *Office.com* по ключевым словам и подготовка списка гиперссылок на разделы, отвечающие на введенный запрос. Ввод дополнительных ключевых слов сужает список найденных разделов. Ответы представляются в списке в порядке релевантности, т. е. первым отображается наиболее подходящий ответ на вопрос. Содержимое найденных разделов может открываться непосредственно в окне справки или на страницах браузера. Раздел, открытый в окне справки, может быть распечатан с помощью соответствующей кнопки в окне справки. Окно справки имеет кнопки перехода вперед и назад по просмотренным разделам.

6.3. Данные в Microsoft Access

6.3.1. Именование полей таблиц в базах данных Microsoft Access

Создавая компьютерную БД, пользователь выполняет ввод данных в поля таблиц этой БД. При этом он должен прежде описать эти поля согласно требованиям Access:

– присвоить имена полям;

– указать их тип и свойства (которые позволят управлять сохранением, обработкой и отображением данных поля).

Полям присваиваются имена с учетом следующих требований:

– имя должно содержать не более 64 символов;

– запрещается использование точки (.), восклицательного знака (!), надстрочного знака (`), квадратных скобок ([ ]) и управляющих символов с кодами ASCII от 0 до 31.

6.3.2. Типы данных в Microsoft Access

В поля БД можно вводить данные нижеприведенных типов:

1) *текстовый*. В поля такого типа помещают текст или комбинацию текстовых и числовых значений. Объем текста не должен превышать 255 символов.

2) *поле MEMO*. Специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. В поле хранится указатель на него.

3) *числовой*. Тип данных для хранения действительных чисел.

4) *дата/время*. Тип данных для хранения календарных дат (в диапазоне от 100 по 9999 год) и текущего времени.

5) *денежный*. Тип данных для хранения денежных значений и предотвращения округления во время вычислений. Их максимальная точность составляет 15 знаков слева от десятичной запятой и 4 знака справа от нее.

6) *счетчик*. Уникальные последовательные с шагом 1 номера, автоматически вставляемые при вставке записи в БД.

7) *логический*. Поля, которые могут принимать только одно значение из пары значений, таких как Да/Нет, Истина/Ложь или Вкл/Выкл.

8) *поле объекта OLE*. Специальный тип данных, предназначенный для хранения объектов, вставляемых внедрением или связыванием (OLE). Реально такие данные в таблице не хранятся, в таблице хранятся только указатели на них.

9) *гиперссылка*. В полях этого типа хранятся гиперссылки – путь к файлу на жестком диске либо адрес в сети Internet или интранет (может хранить до 1 Гбайт данных);

10) *вложение* – используется для вложения в поле записи файлов изображений, электронных таблиц, документов, диаграмм и других файлов поддерживаемых типов точно так же, как в сообщения электронной почты. Вложенные файлы можно просматривать и редактировать в соответствии с заданными для поля параметрами. Эти поля более рационально используют место для хранения, чем поля с типом данных *Поле объекта OLE*, поскольку не создают растровые изображения исходного файла. Максимальная длина поля для сжатых вложений – 2 Гбайт, для несжатых – примерно 700 Кбайт в зависимости от степени возможного сжатия вложения;

11) *вычисляемый* – предназначен для создания вычисляемых полей: числовых, текстовых, денежных, дата/время, логических. Значение вычисляемого поля определяется выражением, записанным в поле и использующим другие поля текущей записи, некоторые встроенные функции и константы, связанные арифметическими, логическими или строковыми операторами;

12) *мастер подстановок* – вызывает мастер подстановок, с помощью которого можно создать поле, позволяющее выбрать значения из списка, построенного на основе значений поля другой таблицы, запроса или фиксированного набора значений. Такое поле отображается как поле со списком. Если список построен на основе поля таблицы или запроса, тип данных и размер создаваемого поля определяется типом данных и размером привязанного столбца, если на основе набора значений – размером текстового поля, содержащего значение. Кроме того, мастер подстановок позволяет определить связь таблиц и включить проверку целостности данных.

Основные свойства задаются для каждого поля. Набор допустимых свойств для поля зависит от того, какого типа данные будут храниться в поле. Если открыть таблицу в режиме конструктора, то весь набор свойств выбранного поля будет представлен в нижней части окна на двух вкладках: «Общие» и «Подстановка». Приведем и охарактеризуем свойства полей, наиболее важные на первом этапе изучения Microsoft Office Access 2013.

6.3.3. Свойства вкладки «Общие»

1. *Имя поля*. Определяет, как следует обращаться к данным этого поля при операциях с БД (по умолчанию имена полей используются в качестве заголовков столбцов таблиц).

2. *Размер поля*. Определяет максимальную длину текстового или числового поля. Для числового поля часто используемыми являются значения:

- *байт* – целые числа от 0 до 255 (поле занимает 1 байт);
- *целое* – целые числа от -32 768 до +32 767 (2 байта);

3. *длинное целое* – целые числа от -2 147 483 648 до +2 147 483 647 (4 байта).

Применяются и другие значения: одинарное с плавающей точкой, двойное с плавающей точкой, действительное.

4. *Формат поля*. Определяет способ отображения текста, чисел, дат и значений времени на экране и на печати.

5. *Число десятичных знаков*. Дает возможность указывать для чисел количество дробных знаков.

6. *Маска ввода*. Определяет форму, в которой данные вводятся в поле.

7. *Подпись*. Определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство *имя поля*).

8. *Значение по умолчанию*. Позволяет указать значение, автоматически вводящееся в поле при создании новой записи.

9. *Правило проверки*. Определяет множество значений, которые можно вводить в поле таблицы.

10. *Сообщение об ошибке*. Позволяет указать текст сообщения, выводящегося на экран при вводе недопустимого значения.

11. *Обязательное поле*. Указывает, требует ли поле обязательного ввода значения.

12. *Пустые строки*. Определяет, допускается ли ввод в текстовое поле пустых строк.

13. *Индексированное поле*. Позволяет создать индекс для поля, ускоряющий поиск и сортировку в таблице.

6.3.4. Свойства вкладки «Подстановка»

В окне конструктора таблиц на вкладке «Подстановка» задается свойство *Тип элемента управления*. Это свойство определяет, будет ли отображаться поле в таблице и в элементе управления формы в виде *Поля*, *Списка* или *Поля со списком*.

Если для поля выбран тип элемента управления «Список» или «Поле со списком», на вкладке «Подстановка» появляются дополнительные свойства, которые определяют источник данных для строк списка и ряд других характеристик списка. Источник данных определяет, откуда брать значения для столбца подстановок. В качестве источника данных для поля со списком выбирается таблица или запрос, с которым осуществляется постоянная связь, что обеспечивает актуальное состояние списка. Для списка строится список конкретных значений, разделенных точкой с запятой. Следует заметить, что не все типы данных позволяют использовать поле со списком.

6.4. Выражения в Microsoft Access

Выражения используются при выполнении многих операций Access (например, при определении условий в запросах).

Выражение – комбинация значений и операторов, дающая определенный результат. Например, следующее выражение позволяет вывести в запросе сумму значений полей «СуммаЗаказа» и «СтоимостьДоставки»:

```
= [СуммаЗаказа] + [СтоимостьДоставки]
```

В качестве значений в выражениях используют литералы, константы, функции и идентификаторы.

Литерал – фактическое значение в виде числа, текстовой строки, даты (например, 100, «Минск», #1-янв-99#).

Константа – не изменяющееся значение (например, Yes, No, True, False и Null).

Функция – небольшая программа, которая всегда возвращает значение (число или строку символов), являющееся результатом расчетов или выполнения других операций. К функции обращаются по имени, за которым в круглых скобках указывают ее аргументы. Имеются и безаргументные функции, например, *Date()*. В Access определено множество типов функций: статистические, математические, даты/время, текстовые, финансовые и др.

Идентификатор – ссылка на значение поля, элемента управления или свойства. Например, *Forms![Заказы]![ДатаРазмещения].Default Value* (ссылка на свойство Default Value элемента управления «ДатаРазмещения» в форме «Заказы»). Символ «!» предопределяет ссылку на открытую форму, на открытый отчет или их элемент управления, а символ «.» – на свойства форм, отчетов и элементов управления.

Оператор – операция над значениями. В выражениях используются четыре типа операторов: арифметические операторы, операторы сравнения, логические операторы, текстовый оператор.

Арифметические операторы: + (сложение), – (вычитание), * (умножение), / (деление), % (проценты), ^ (возведение в степень) и др.

Операторы сравнения: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно).

Логические операторы: AND (логическое умножение), OR (логическое сложение), NOT (логическое отрицание).

Текстовый оператор: & (слияние строковых значений).

Пользователь имеет возможность создавать выражения с помощью

Построителя выражений (рис. 6.5) или путем набора их на клавиатуре. *Построитель выражений* доступен во всех случаях, когда возможен ввод выражения.

В верхней части окна построителя выражений находится поле выражения. Под ним располагаются кнопки с часто используемыми операторами. Ниже содержится раздел, предназначенный для создания элементов выражения. Процесс создания выражения сводится к последовательному выбору в разделе необходимых значений (констант, функций, идентификаторов) и вставке их в поле выражения. При этом с помощью соответствующих элементов управления попутно вставляются между значениями необходимые операторы. Литералы, присутствующие в выражении, набираются на клавиатуре.

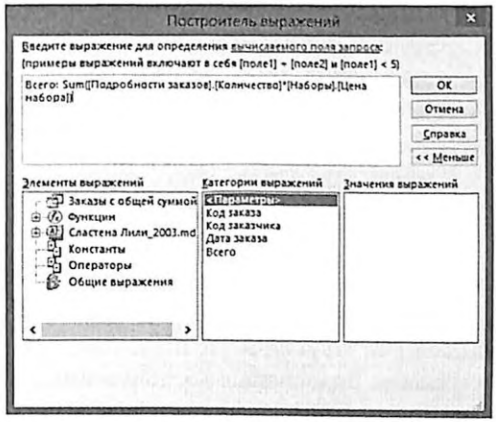


Рис. 6.5. Окно «Построитель выражений» при формировании вычисляемого поля запроса

6.5. Инструментальные средства создания объектов БД

СУБД Microsoft Access 2013 предоставляет несколько средств создания каждого из основных объектов базы. Эти средства можно классифицировать как:

– средства разработки *вручную* (в режиме *Конструктора*). *Конструктор* предоставляет пользователю ряд инструментальных средств, с помощью которых можно создавать и модифицировать объекты БД. Для конструирования макета формы, отчета, страницы доступа к данным используется панель элементов, появляющаяся при вызове конструктора;

– *автоматизированные* (разработка с помощью программ-мастеров). Мастер задает пользователю ряд вопросов и на основе его ответов строит законченный объект БД или осуществляет определенную операцию;

– *автоматические* – средства ускоренной разработки простейших объектов.

Ручные средства наиболее трудоемки, но обеспечивают максимальную гибкость; автоматизированные и автоматические средства являются наиболее производительными, но и наименее гибкими.

6.6. Создание и корректировка БД в СУБД Access

Создание новой базы данных Access осуществляется в соответствии с ее структурой, полученной в результате проектирования. Процесс проектирования реляционной базы данных был рассмотрен в главе 3. Структура реляционной базы данных определяется составом таблиц и их взаимосвязями.

Создание реляционной базы данных начинается с формирования структуры таблиц. При этом определяется состав полей, их имена, тип данных каждого поля, размер поля, ключи, индексы таблицы и другие свойства полей. После определения структуры таблиц создается схема данных, в которой устанавливаются связи между таблицами. Access запоминает и использует эти связи при заполнении таблиц и обработке данных.

При создании базы данных важно задать параметры, в соответствии с которыми Access будет автоматически поддерживать целостность данных. Для этого при определении структуры таблиц должны быть заданы ключевые поля таблиц, указаны ограничения на допустимые значения данных, а при создании схемы данных на основе нормализованных таблиц должны быть заданы параметры поддержания целостности связей базы данных.

Завершается создание базы данных заполнением таблиц конкретными данными.

6.6.1. Создание новой (пустой) базы данных

Для того чтобы в Microsoft Office Access 2013 создать новую БД, необходимо выполнить следующие действия:

1) открыть Microsoft Access – отобразится представление *Backstage* (рис. 6.1);

2) щелкнуть по значку «Пустая база данных рабочего стола» откроется окно как на рисунке 6.6;

3) в этом окне в поле «Имя файла» ввести имя файла базы данных или используйте предлагаемое имя;

4) нажать кнопку Создать – будет создана новая база данных и открыта новая таблица в режиме таблицы.

Рис. 6.6. Окно «Пустая база данных рабочего стола

Access 2013 содержит ряд встроенных шаблонов баз данных. Шаблон Access представляет собой готовую базу данных с профессионально разработанными таблицами, формами и отчетами. Шаблоны позволяют быстро пройти начальные этапы создания базы данных.

Чтобы создать базу данных из образца шаблона, нужно выполнить следующие действия:

1) открыть Microsoft Access – отобразится представление *Backstage*;
2) щелкнуть на значке подходящего образца шаблона;
3) ввести имя файла или использовать предложенное имя;
4) нажать кнопку Создать – приложение Access создаст на основе шаблона новую базу данных и откроет ее.

Загрузить дополнительные шаблоны Access можно с веб-сайта *office.com* непосредственно в представлении *Backstage*.

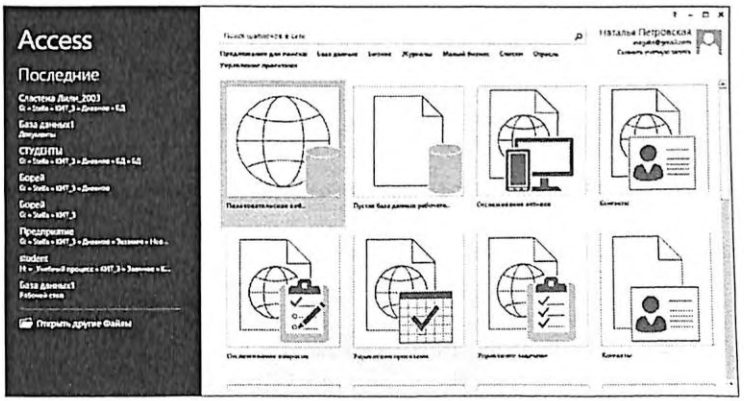


Рис 6.7. Создание новой БД

6.6.2. Открытие недавно использовавшейся базы данных

Чтобы открыть недавно использовавшуюся БД, нужно выполнить следующие действия:

1) открыть Microsoft Access – отобразится представление *Backstage*;

2) в представлении *Backstage* в списке «Последние» щелчком мыши выбрать базу данных, которую необходимо открыть – приложение Access откроет выбранную базу данных.

Примечание. Если необходимо открыть БД, которой нет в списке «Последние», то следует воспользоваться ссылкой *Открыть другие файлы*.

6.7. Создание таблицы

В Access существует два способа создания таблицы:

- в режиме таблицы;
- в режиме конструктора.

6.7.1. Создание таблицы в режиме таблицы

В процессе создания новой базы данных в ней автоматически создается новая пустая таблица с именем *Таблица1*, которая открывается в рабочей области в режиме таблицы. На рисунке 6.8 показана такая таблица с открытым списком поля «Щелкните для добавления», обеспечивающим выбор типа данных для поля таблицы.

Одновременно открывается контекстная вкладка «Работа с таблицами|Поля», обеспечивающая определение полей таблицы с различными свойствами. На рисунке 6.9 представлена эта вкладка с открытым списком «Другие поля», включающая расширенный список для добавления и удаления полей таблицы с различными типами данных.

Для создания следующей таблицы в аналогичном режиме предназначен инструмент *Таблица*  , размещенный на вкладке ленты «Создание» в группе «Таблицы».

В предлагаемой таблице определено ключевое поле с типом данных «Счетчик», и в нее в режиме таблицы можно добавлять новые поля, наделенные рядом характеристик. Столбец «Щелкните для добавления» постоянно отображается в режиме таблицы, за исключением случая, когда в таблице не определен первичный ключ.

Как уже было сказано ранее, для каждого поля обязательно задается имя, однозначно определяющее это поле в таблице, и тип данных, соответствующий сохраняемым в этом поле данным. Тип данных определяет значения, которые можно сохранить в поле, операции, которые можно выполнить

с данными, а также выделяемый объем памяти. С каждым полем также связывается группа параметров, называемых свойствами, которые определяют функциональные характеристики и внешний вид этого поля. В Access 2013 в режиме таблицы для поля может быть задано большинство свойств.

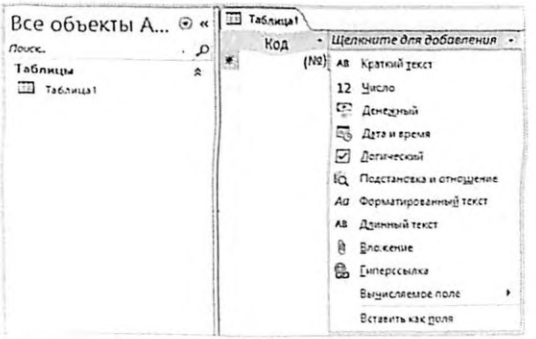


Рис. 6.8. Новая пустая таблица со списком типов данных для добавляемого поля

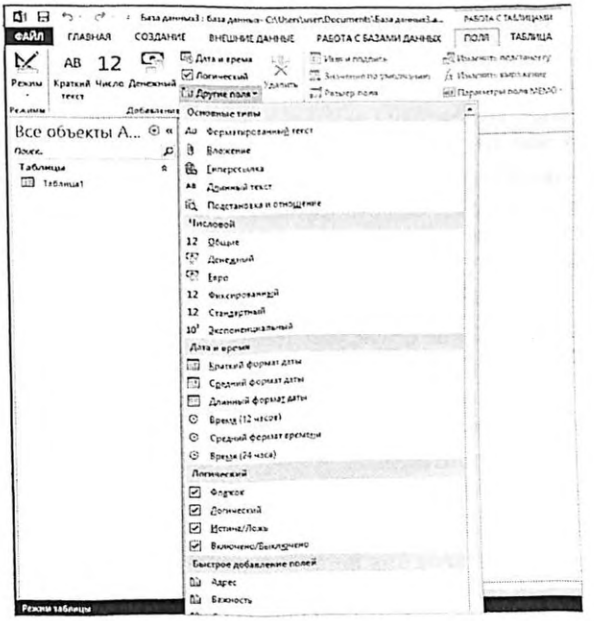


Рис. 6.9 Контекстная вкладка «Работа с таблицами | Поля»

Добавление полей. В режиме таблицы определение полей можно выполнить как выбором необходимых команд на контекстной вкладке «Работа с таблицами | Поля» или в списке столбца «Щелкните для добавления», так и непосредственным вводом данных в столбец «Щелкните для добавления» (рис. 6.8). При вводе первого значения Access автоматически определит тип данных (например, дата, число или текст) и добавит новое поле в таблицу. При добавлении полей этим способом Access автоматически назначит им имена: *Поле1* для первого, *Поле2* для второго и т. д. Чтобы дать полям содержательные имена, их следует изменить:

1) выполнить двойной щелчок на заголовке поля;
2) ввести новое имя.

Access распознает и автоматически определяет следующие типы данных:

– для текстовых значений – *Короткий текст*, *Длинный текст*;

– для целых числовых значений – *Числовой*, *Длинное целое*;

– для числовых значений типа 45,76 или 34,25 % или 12 % – *Числовой*, *Двойное с плавающей точкой*;

– для гиперссылок – *Гиперссылка*. Допускается использование любого префикса протокола Интернета, например `http://`, `mailto:`;

– для даты и времени – *Дата и время*;

– для денежных значений типа 12,50 р. – *Денежный*.

Если на основании введенных данных Access не может точно определить тип данных, задается тип данных *Короткий текст*.

Access, наряду с определением типа данных, может автоматически задавать значение свойства *Формат* в зависимости от вида вводимых данных. Это свойство определяет вид отображения и печати данных поля.

Пример. Если ввести 01-янв-2010, будет задан тип данных *Дата и время* и в свойстве *Формат* значение *Средний формат даты*.

После определения структуры таблицы таким простым способом ее всегда можно доработать, воспользовавшись командами контекстной вкладки «Работа с таблицами | Поля» в группах «Свойства», «Форматирование» и «Проверка поля» (рис. 6.9). При этом будут доступны почти все используемые Access типы данных.

Используя команду *Вставить как поля*, можно добавить в таблицу новое поле, скопированное из другой таблицы. Для этого нужно выполнить следующие действия:

1) открыть таблицу, из которой будет выполняться копирование;
2) подвести курсор к нужному значению поля и при появлении на нем знака белый плюс щелчком мыши выделить его;

3) скопировать выделенное поле. При этом копируются конкретное значение поля, его имя и все свойства;

4) командой *Вставить как поля* добавить поле в создаваемую таблицу.

Наиболее простым способом добавления полей в таблицу можно считать использование команд группы «Добавление и удаление», в которой отдельными кнопками представлены основные типы данных и в раскрывающемся списке «Другие поля» представлены не только недостающие типы данных, но и форматы отображения поля (рис. 6.9).

Определив структуру таблицы в режиме таблицы, можно тут же приступить к заполнению ее данными – формированию записей таблицы. Команды по работе с записями таблицы, а также команды оформления внешнего вида таблицы представлены на вкладке ленты «Главная».

Если таблица была закрыта, открыть ее в режиме таблицы можно двойным щелчком левой кнопки мыши по имени таблицы.

6.7.2. Создание таблицы в режиме конструктора

Для создания таблицы в режиме конструктора на вкладке «Создание» в группе «Таблицы» следует воспользоваться инструментом «Конструктор таблиц». В режиме конструктора открывается таблица *Таблица1*, в котором определяется ее структура (рис. 6.10).

Для перехода из режима конструктора в режим таблицы и наоборот следует использовать инструмент «Режим», доступный на вкладках «Работа с таблицами | Поля» и «Главная».

При переходе в режим конструктора таблиц активизируется контекстная вкладка «Работа с таблицами | Конструктор» (рис. 6.11).

Для определения поля в окне конструктора таблицы (рис. 6.10) задаются *Имя поля*, выбирается *Тип данных*, при необходимости приводится *Описание* – краткий комментарий, в разделе «Свойства поля» задаются свойства, представленные на двух вкладках:

– «Общие». К общим относятся такие свойства поля, как максимальный размер, формат, подпись, которая выводится в заголовке столбца таблицы, значение по умолчанию и др.;

– «Подстановка». На этой вкладке выбирается *Тип элемента управления*: поле, список фиксированных значений или поле со списком.

Свойства поля зависят от выбранного типа данных. Для отображения свойств поля необходимо установить курсор на строке соответствующего поля.

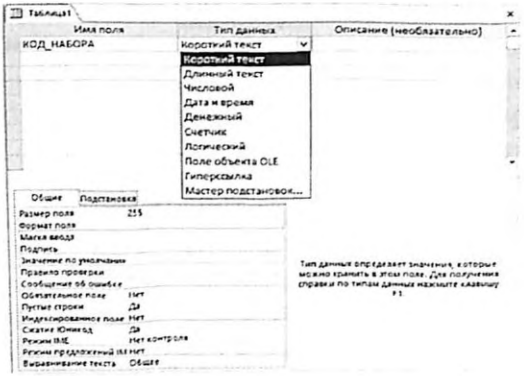


Рис. 6.10. Окно определения структуры таблицы в режиме конструктора

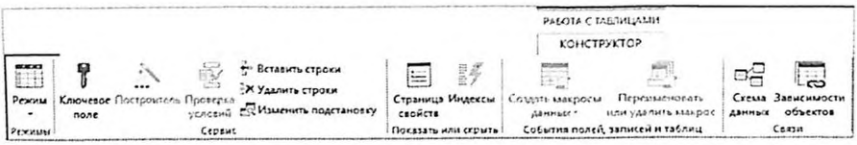


Рис. 6.11. Лента инструментов конструктора таблиц

Если создается главная таблица, то в ней необходимо определить первичный ключ. Для этого выделяется нужное поле путем щелчка на маркере выделения строки и выполняется команда *вкладка «Работа с таблицами | Конструктор» → инструмент «Ключевое поле».*

Примечание. Создание составного ключа выполняется точно также, но предварительно нужно выделить те поля, которые войдут в составной ключ. Групповое выделение полей выполняется при нажатой клавише Shift щелчками на маркерах выделения строки.

В случае создания подчиненной таблицы надо учесть следующее. Если между главной и подчиненной таблицами связь типа 1:1, то для поля связи подчиненной таблицы следует задать свойство *Индексированное поле* со значением *Да (Совпадения не допускаются)*; если связь типа 1:М, то со значением *Да (Допускаются совпадения)*.

Описав структуру таблицы, окно конструктора закрывают, на вопрос о сохранении таблицы дают утвердительный ответ и таблице присваивают имя – с этого момента она доступна в числе прочих таблиц в области навигации.

Для заполнения созданной таблицы конкретными данными она открывается в режиме таблицы. Поочередно заполняются записи таблицы. При этом переход к новой записи вызывает автоматическое сохранение предыдущей записи.

В процессе работы с БД может понадобиться корректировка структуры таблицы. Она выполняется в режиме конструктора. Поля таблицы можно:

– перемещать при помощи мыши;

– удалять и добавлять с помощью инструментов группы «Добавление и удаление» контекстной вкладки «Работа с таблицами | Поля»;

Кроме того, возможно изменение имен, типов, свойств полей.

Примечание. Изменение типа может привести к потере информации, и оно должно выполняться с осторожностью.

6.7.3. Работа с макетом таблицы

Для удобства работы с таблицей можно изменить ее представление на экране. При этом можно менять ширину поля, высоту строки, шрифт данных таблицы, цвет текста, линий сетки и фона, оформление, которое может быть обычным, приподнятым или утопленным. Можно выводить на экран только те поля, которые нужны для текущей работы, можно зафиксировать поле при просмотре широких таблиц. Эти параметры отображения таблицы на экране называются *макетом* таблицы и сохраняются вместе с ней.

Настройка макета выполняется в *режиме таблицы*. При этом могут быть использованы команды вкладки ленты «Главная».

Многие операции настройки макета можно выполнить непосредственно в таблице с помощью мыши.

Таблица 6.2

Операции со значениями полей

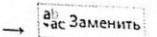
Наименование операции	Выполняемые действия и команды
Редактирование значения поля	Щелчок на значении поля
Поиск значения в поле	Выделение поля, выполнение команды *вкладка ленты «Главная» → группа «Найти» →* 
Замена значения в поле	Выделение поля, выполнение команды *вкладка ленты «Главная» → группа «Найти» →* 

Таблица 6.3

Операции с записями

Наименование операции	Выполняемые действия и команды
Добавление записи	Выполнение команды *вкладка ленты «Главная»* → *группа «Записи»* → Создать или нажатие одноименной кнопки
Удаление записи	Выделение записи, выполнение команды *вкладка ленты «Главная»* → *группа «Записи»* → Удалить или нажатие клавиши Delete на клавиатуре
Сортировка записей	Выделение поля, по которому будут сортироваться записи, выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → → По возрастанию (или По убыванию)
Фильтр по выделенному	Выделение значения поля, выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Выделение
Обычный фильтр	Выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Дополнительно → → Фильтр по форме. Выбор значений, по которым будет осуществляться отбор записей. Выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Фильтр
Расширенный фильтр	Выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Дополнительно → → Расширенный фильтр... Очистка бланка фильтра и формирование условия отбора записей. Выполнение команды *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Фильтр
Отображение подчиненных записей для записи главной таблицы	Выбор подчиненной таблицы по команде *вкладка ленты «Главная»* → *группа «Записи»* → → Дополнительно → Подтаблица → → Подтаблица... Щелчок на значке «+» слева от записи
Отображение всех подчиненных записей в главной таблице	Выполнение команды *вкладка ленты «Главная»* → *группа «Записи»* → Дополнительно → Подтаблица → Развернуть все

Примечание. Фильтр временно разделяет записи таблицы на два подмножества, одно из которых будет показано, а другое – нет. Для отмены фильтра используется команда *вкладка ленты «Главная»* → *группа «Сортировка и фильтр»* → Дополнительно → Очистить все фильтры или нажимается кнопка Фильтр.

Примечание. Для отмены отображения подчиненных записей у записи главной таблицы производится щелчок на значке «–» слева от записи; для отмены отображения всех подчиненных записей применяется команда *вкладка ленты «Главная»* → *группа «Записи»* → Дополнительно → Подтаблица → Свернуть все.

Таблица 6.4

Операции с полями и их выполнение

Наименование операции	Выполняемые действия и команды
Переименование поля	Двойной щелчок на заголовке поля, ввод нового имени или выполнение команды *контекстная вкладка ленты «Работа с таблицами | Поля» → группа «Свойства» →* Имя и подпись – в открывшемся окне диалога «Введите свойства поля» выполнить ввод нового имени
Вставка поля	Выделение поля, после которого осуществляется вставка, использование одного из инструментов группы «Добавление и удаление» контекстной вкладки ленты «Работа с таблицами | Поля»
Удаление поля	Выделение поля, выполнение команды *контекстная вкладка ленты «Работа с таблицами | Поля» → группа «Добавление и удаление» →*
Скрытие полей	Выделение полей, выполнение команды *контекстное меню выделенных полей → Скрыть поля*
Перемещение поля	Выделение поля, перетаскивание мышью поля за заголовок на новое место
Закрепление поля	Выделение поля, который должен быть крайним левым при прокрутке таблицы вправо, выполнение команды *контекстное меню выделенного поля → Закрепить поля*

Примечание. Для отмены скрытия столбца используется команда *контекстное меню выделенного поля → Отобразить поля*; для отмены закрепления столбца – *контекстное меню выделенного поля → Отменить закрепление всех полей.*

Кроме того, для макета таблицы также возможны следующие изменения:

– изменение ширины полей (*контекстное меню выделенного поля → Ширина поля*) и высоты строк (*контекстное меню выделенного поля → Высота строки*);

– специальное оформление таблицы с помощью инструментов группы «Форматирование текста» вкладки ленты «Главная» (рис. 6.12) и инструментов окна диалога «Формат таблицы» (рис. 6.13), которое можно открыть щелчком по кнопке в группе «Форматирование текста». Под специальным форматированием понимается задание линий сетки, их цвета и типа, фона и типа границы таблицы;

– проверка орфографии в текстовых полях по команде *вкладка ленты «Главная» → группа «Записи» →* Орфография.

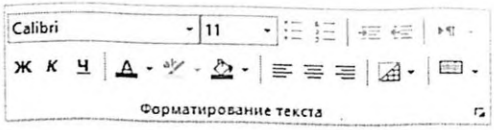


Рис. 6.12. Группа инструментов «Форматирование текста»

Рис. 6.13. Окно диалога «Формат таблицы»

6.8. Создание схемы данных

Схема данных является удобным и наглядным средством отображения логических связей таблиц БД. Связи используются для объединения записей связанных таблиц при любой обработке. Это упрощает процесс конструирования многотабличных запросов, форм и отчетов. Если создается связь с поддержкой целостности данных, то можно предусмотреть в БД следующие каскадные операции (операции по корректировке БД):

– каскадное обновление связанных полей: изменение значений ключевого поля в записи главной таблицы влечет автоматическое изменение значений связанного поля;

– каскадное удаление связанных записей: удаление записи из главной таблицы приводит к автоматическому удалению всех подчиненных записей.

Для создания схемы данных нужно выполнить следующие действия:

1) выполнить команду *вкладка ленты «Работа с базами данных»* → *Схема данных* – откроется окно «Добавление таблицы» (рис 6.14);

Рис. 6.14. Окно диалога «Добавление таблицы»

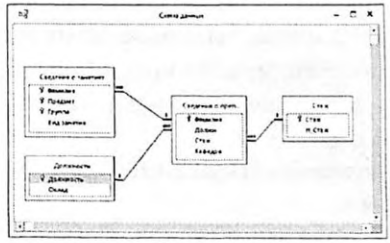
2) в окне «Добавление таблицы» на вкладке «Таблицы» поочередно выделять значки связываемых таблиц и нажимать кнопку Добавить;

3) добавив все таблицы, закрыть окно «Добавление таблицы» – станет доступным окно «Схема данных» (рис. 6.15, а);

4) для установления связи между двумя таблицами следует перетащить имя ключевого поля главной таблицы на имя связываемого поля подчиненной таблицы – откроется окно диалога «Изменение связей» (рис. 6.15, б);

5) в окне «Изменение связей» предусмотреть контроль целостности данных и выполнение каскадных операций;

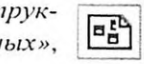
6) нажать кнопку Создать – произойдет возврат в окно «Схема данных». В этом окне между таблицами установится связь, обозначенная на схеме, например, как 1:1 или 1:∞ (рис. 6.15, а).



а)

б)

Рис. 6.15. Установление связей в БД: а) Окно «Схема данных», б) Окно диалога «Изменение связей»

Для печати схемы данных сначала формируется специальный отчет по команде *контекстная вкладка «Работа со связями | Конструктор» → группа «Сервис» → инструмент «Отчет по схеме данных»*,  затем выполняется печать полученного отчета.

Для изменения параметров существующей связи нужно выполнить следующие действия:

1) открыть окно «Схема данных» (рис. 6.15, а);

2) выполнить двойной щелчок по линии связи, подлежащей изменению;

3) изменить параметры связи в открывшемся окне «Изменение связей» (рис. 6.15, б).

6.9. Формирование запросов в СУБД Access

6.9.1. Возможности, типы и средства создания запросов

Запрос – это требование на отбор данных, хранящихся в таблицах, или требование на выполнение определенных действий с данными. Запрос позволяет создать общий набор записей из данных, находящихся в разных таблицах, который будет служить источником данных для формы, отчета или страницы доступа к данным. Итогом выполнения запроса является результирующая таблица, которая может быть просмотрена, проанализирована, а затем сохранена или не сохранена.

Запросы позволяют решать многие задачи, не прибегая к программированию. Например, производить вычисления над полями БД, группировать записи и находить для полей итоговые значения с помощью статистических функций: *Sum*, *Avg* (соответственно *сумма*, *среднее* значений поля); *Max*, *Min* (соответственно *максимальное*, *минимальное* значение поля); *Count* (*число значений поля*) и др.

СУБД Access позволяет создавать запросы трех типов:

- запросы выбора;
- перекрестные запросы;
- запросы действия.

Запрос выбора является наиболее часто используемым типом запроса. Он дает возможность: выбирать записи, удовлетворяющие условиям отбора; включать в результирующую таблицу поля из одной или нескольких таблиц в нужном порядке; осуществлять вычисления над полями БД; выполнять статистические расчеты для групп записей. Разновидностью запроса выбора является запрос с параметром (параметрами). Этот запрос при выполнении отображает в собственном диалоговом окне приглашение ввести интересующее пользователя значение критерия отбора записей.

Перекрестный запрос представляет собой специальный запрос итогового типа. Он отображает результаты итоговых статистических расчетов над значениями некоторого поля в виде перекрестной таблицы. В ней значения одного или нескольких столбцов слева образуют заголовки строк, верхняя строка – заголовки столбцов из значений определенного поля, а на пересечении строк и столбцов – итоговые значения.

Запрос действия – это запрос, который вносит изменения в саму БД. Существует четыре типа запросов действия:

– *запрос на удаление* – удаляет группу записей из одной таблицы или нескольких взаимосвязанных таблиц БД, для которых задано каскадное удаление связанных записей;

– *запрос на обновление* – служит для изменения информации в полях таблицы БД;

– *запрос на добавление* – производит добавление записей из таблицы с результатами запроса в таблицу БД;

– *запрос на создание таблицы* – создает новую таблицу на основе всех или части данных из одной или нескольких таблиц БД.

Запрос на создание таблицы полезен в следующих случаях:

– создание таблицы для экспорта в другую БД Access;

– создание страниц доступа к данным, отображающих данные соответственно указанному моменту времени;

– создание резервной копии таблицы;

– создание архивной таблицы, содержащей старые записи.

СУБД Access позволяет создавать запросы с помощью *мастера* и с помощью *конструктора*. Мастер используется для создания следующих запросов:

– простого запроса на выборку полей из источника запроса и подведения итогов;

– перекрестного запроса;

– запроса на поиск повторяющихся записей в таблице;

– запроса на поиск записей, не имеющих подчиненных им записей в другой таблице.

С помощью конструктора можно создавать запросы любых типов.

6.9.2. Создание запроса выбора

Для создания запроса выбора с помощью конструктора необходимо открыть окно конструктора запросов. Для этого нужно выполнить следующие действия:

1) в группе «Запросы» вкладки «Создание» щелчком мыши выбрать инструмент «Конструктор запросов» – откроется окно конструктора запроса и окно диалога «Добавление таблицы» (рис. 6.16);

Рис. 6.16. Окно диалога «Добавление таблицы»

2) в этом окне выбрать таблицы-источники запроса;

3) закрыть окно «Добавление таблицы» и перейти к работе в окне конструктора запроса (рис. 6.17).

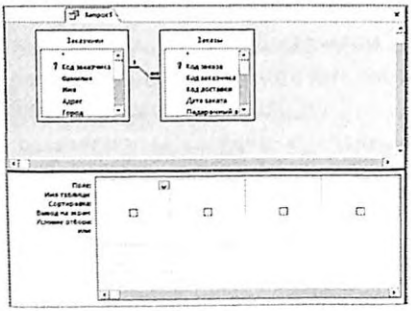


Рис. 6.17. Окно «Конструктор запросов»

Окно состоит из двух областей:

1) верхняя область содержит схему данных запроса. В ней представлены списки полей таблиц и запросов, выбранных в качестве источника запроса. Если ранее была создана связь между этими таблицами, то она отображается на схеме данных. В противном случае может отображаться связь, автоматически созданная СУБД Microsoft Access. Пользователь может сам установить новую связь между таблицами;

2) нижняя область представляет собой бланк запроса по образцу (QBE[1]-запроса). Он представлен в виде таблицы, предназначенной для определения структуры результирующей таблицы запроса и задания условий отбора данных из таблиц. Каждый столбец бланка относится к одному полю таблицы.

Строки бланка имеют следующее назначение:

- *Поле* – указывает имена полей, участвующих в формировании запроса;
- *Имя таблицы* – указывает имена таблиц, которым принадлежат эти поля;
- *Сортировка* – дает возможность отсортировать записи в результирующей таблице запроса;
- *Вывод на экран* – позволяет управлять отображением полей в этой таблице;
- *Условие отбора* – служит для задания условий отбора записей;
- *или* – позволяет объединять условия отбора логической операцией *ИЛИ*. При этом условия отбора могут указываться в нескольких строках бланка запроса.

Включение отдельных полей в бланк запроса можно выполнить одним из следующих способов:

– перетащить поле из списка полей в крайнюю слева свободную ячейку строки «Поле»;

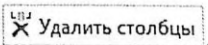
– дважды щелкнуть по имени поля в списке полей;

– щелкнуть в ячейке строки «Поле» и из раскрывающегося списка выбрать нужное поле.

Включение в бланк запроса всех полей таблицы можно выполнить, если:

– дважды щелкнуть по имени таблицы и перетащить все выделенные поля в ячейку строки «Поле»;

– перетащить звездочку, стоящую под заголовком таблицы, в ячейку строки «Поле». При этом в клетке отобразится только имя таблицы со звездочкой, но в результат запроса будут включены все поля таблицы.

Для очистки отдельных столбцов бланка запроса используется команда *контекстная вкладка «Работа с запросами | Конструктор»* → → *группа «Настройка запроса»* → 

.

[1] Query By Example

Формирование запроса на вывод полей из одной или нескольких таблиц. Перетаскиваются из списков полей в строку «Поле» только те поля, которые должны присутствовать в результирующей таблице, или все поля всех таблиц, а затем в строке «Вывод на экран» отмечаются флажками только нужные.

Формирование запроса с применением сортировки. По умолчанию данные отображаются в порядке, установленном для первичного ключа первой выбранной таблицы. Если первичного ключа нет, данные отображаются в порядке отбора записей. Для задания другого порядка сортировки в нижней части бланка имеется специальная строка «Сортировка». При щелчке в этой строке появляется кнопка списка, в котором можно выбрать метод сортировки: *по возрастанию* или *по убыванию*. Текстовые данные сортируются по возрастанию от A до Z (от А до Я), а числовые – от 0 до 9. В результирующей таблице данные будут отсортированы по тому полю, для которого задан порядок сортировки.

Возможна многоуровневая сортировка, т. е. сразу по нескольким полям. В этом случае данные сначала сортируются по тому полю, которое в бланке запроса по образцу находится левее, затем по следующему полю, для которого включена сортировка, и так далее слева направо. Соответственно, при формировании запроса надо располагать поля результирующей таблицы с учетом будущей сортировки.

Формирование запроса с условиями отбора. *Условия отбора* – это ограничения, используемые в запросе для определения записей, включаемых в результирующую таблицу. Они задаются выражениями в строках «Условие отбора», «или».

Если выражения вводятся в несколько ячеек одной строки «Условие отбора», то они автоматически объединяются с помощью логического оператора *And* (*И*). Например (рис. 6.18):

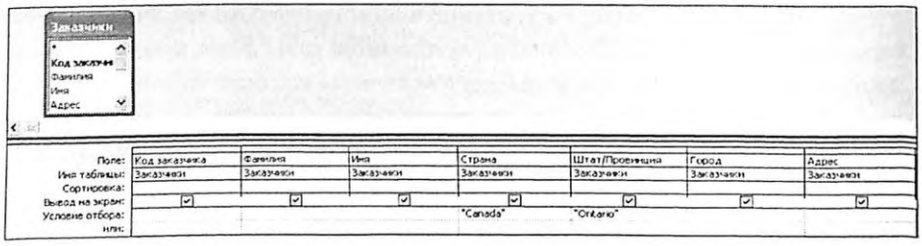


Рис. 6.18. Запрос выбора с использованием логического оператора And

В этом запросе выбираются записи, которые имеют значение *Canada* в поле *Страна* и *Ontario* в поле *Штат/Провинция*.

Если выражения вводятся в разные строки бланка запроса, то Microsoft Access объединяет их логическим оператором *Or* (*ИЛИ*). Например (рис. 6.19):

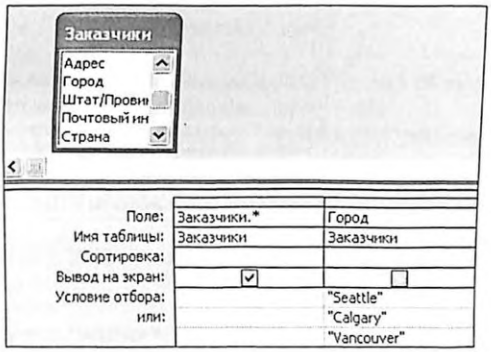


Рис. 6.19. Запрос выбора с использованием логического оператора `Or`

В этом запросе выбираются все записи, содержащие в поле *Город* либо *Seattle*, либо *Calgary*, либо *Vancouver*.

В одном и том же запросе можно использовать комбинацию логических операций *And* и *Or*. Например (рис. 6.20):

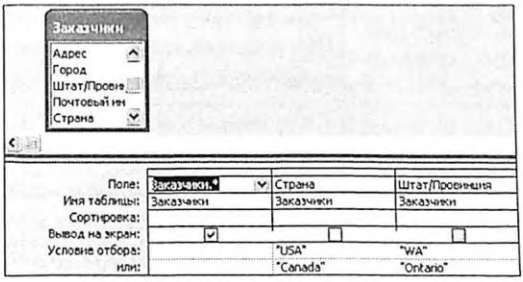


Рис. 6.20. Запрос выбора с использованием логических операторов `And` и `Or`

В этом примере запроса извлекаются данные, содержащие *USA* в поле *Страна* и *WA* в поле *Штат/Провинция*; или *Canada* в поле *Страна* и *Ontario* в поле *Штат/Провинция*.

Таблица 6.5

Наиболее типичные выражения, используемые в запросах

Поле (условное)	Условие отбора	Результат
Количество	>234	Отбор записей, значение поля «Количество» которых больше 234
Количество	Between 50 And 100	Отбор записей, значение поля «Количество» которых находится в диапазоне от 50 до 100
Страна получателя	Not "США"	Отбор записей с некоторыми сведениями, содержащих в поле «Страна получателя» любую страну за исключением США
Фамилия	"Иванов"	Отбор записей, содержащих фамилию «Иванов» в поле «Фамилия»
Имя получателя	Like "С*"	Отбор записей, в которых имена в поле «Имя получателя» начинаются с буквы С
Имя получателя	Like "[А-Д]*"	Отбор записей, в которых имена в поле «Имя получателя», начинаются с букв в диапазоне от А до Д
Дата	#2/2/2000#	Записи, в поле «Дата» которых указана дата 2 февраля 2000
Дата	Date()	Записи, в поле «Дата» которых указана текущая дата
Дата	Between #02-фев-1999# And #01-дек-1999#	Отбор записей, содержащих в поле «Дата» даты в диапазоне от 2-фев-99 до 1-дек-99
Количество	Is Null	Отбор записей, значение поля «Количество» которых содержит значение Null, т. е. является пустым

Формирование запроса с вычисляемым полем. Для создания вычисляемого поля в пустую ячейку строки «Поле» вводится имя вычисляемого поля с двоеточием, после которого – *выражение* (рис. 6.21).

Для удобства набора выражения можно использовать *Область ввода*, которая открывается по нажатию сочетания клавиш Shift+F2 (рис. 6.22).

Если выражение сложное, то для его создания целесообразно использовать *построитель выражений*.

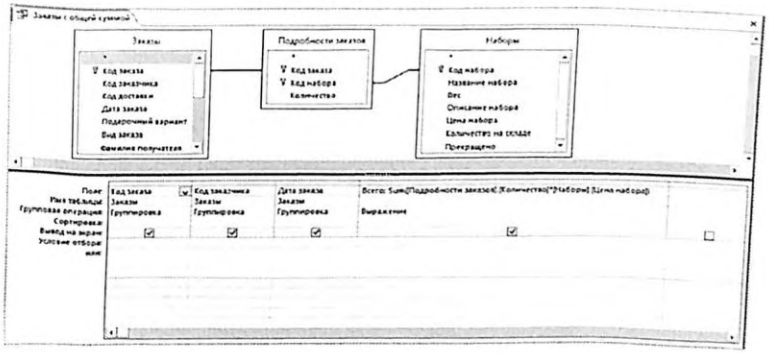


Рис. 6.21. Создание запроса с вычисляемым полем

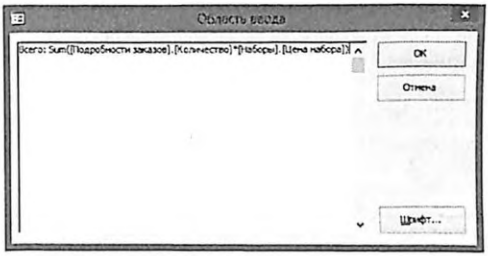


Рис. 6.22. Область ввода

Формирование запроса с группировкой. Часто требуется видеть в результирующей таблице не все записи, а только итоговые значения по группам записей. Расчет итогов для некоторых полей групп производится с помощью статистических функций, которые были описаны выше. Для формирования запроса с группировкой нужно выполнить следующие действия:

1) перетащить в ячейки строки «Поле» необходимые поля (по которым производится группировка и по которым подводятся итоги);

2) выполнить команду *контекстная вкладка «Работа с запросами | Конструктор»* → *группа «Показать или скрыть»* → *Итоги* – в бланке запроса появляется строка «Групповая операция», в которой для всех полей указано *Группировка* (рис. 6.23);

3) в строке «Групповая операция» для полей, по которым подводятся итоги, производится выбор из раскрывающегося списка требуемой статистической функции.

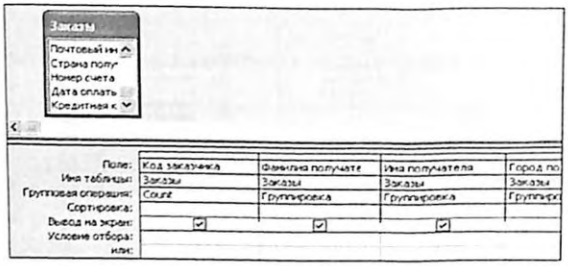


Рис. 6.23. Использование в запросе группировки и статистической функции

Формирование запроса с параметрами. Если необходимо часто выполнять один и тот же запрос выбора, меняя в условиях отбора значения полей, то целесообразно создать *запрос с параметрами*. В строках «Условие отбора», «или» для полей, играющих роль параметров, вводится в квадратных скобках текст приглашения на ввод интересующих пользователя значений этих полей. Этот текст будет выводиться в диалоговом окне «Введите значение параметра» при выполнении запроса (рис. 6.24).

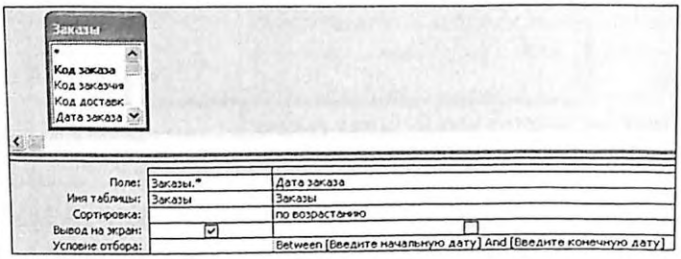


Рис. 6.24. Создание запроса с параметрами

6.9.3. Создание перекрестного запроса

Для создания перекрестного запроса в режиме конструктора (рис. 6.25) нужно выполнить следующие действия:

1) открыть окно конструктора запросов;
2) в бланк запроса последовательно перетащить:
   - поля, значения которых будут заголовками строк перекрестной таблицы;
   - поле, значения которого будут заголовками столбцов перекрестной таблицы;
   - поле, по которому подводится итог с использованием статистической функции.

3) выполнить команду *контекстная вкладка «Работа с запросами | Конструктор» → группа «Тип запроса» → Перекрестный* – в результате в бланке запроса появятся две строки: «Групповая операция» и «Перекрестная таблица»;

4) в строке «Перекрестная таблица» для полей со значениями в роли заголовков строк выбирается из раскрывающегося списка значение *Заголовки строк*, для полей со значениями в роли заголовков столбцов – *Заголовки столбцов*;

5) в строке «Групповая операция» для поля, по которому подводится итог, из раскрывающегося списка выбирается необходимая статистическая функция.

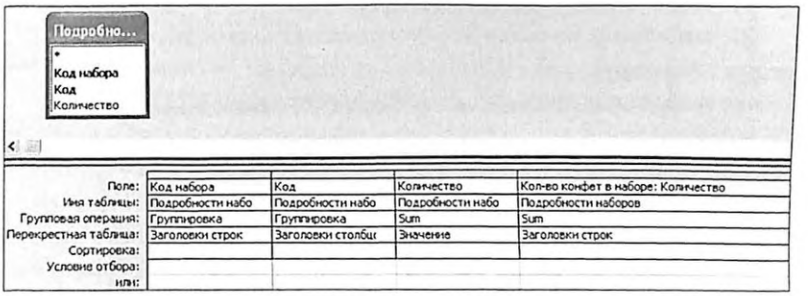


Рис. 6.25. Создание перекрестного запроса в режиме конструктора

6.9.4. Создание запросов действия

Формирование запроса на создание таблицы БД. Процедура формирования запроса на создание таблицы с помощью конструктора запросов:

1) открыть окно конструктора запросов;

2) выполнить команду *контекстная вкладка «Работа с запросами | Конструктор» → группа «Тип запроса» → Создание таблицы* – откроется окно диалога «Создание таблицы» (рис. 6.26);

Рис. 6.26. Окно диалога «Создание таблицы»

3) в этом окне указать имя создаваемой таблицы и куда ее следует поместить – в текущую или в другую БД – откроется бланк запроса;

4) из списков полей перетаскиваются в бланк запроса поля, которые должны быть в этой таблице, и при необходимости задаются условия отбора записей;

5) нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» для выполнения запроса. После выполнения запроса новая таблица будет в списке таблиц в области навигации.

Формирование запроса на обновление полей таблицы БД. Для формирования запроса на обновление полей таблицы с помощью конструктора нужно выполнить следующие действия:

1) открыть окно конструктора запросов;

2) выполнить команду *контекстная вкладка «Работа с запросами | Конструктор» → группа «Тип запроса» → Обновление* – в бланке запроса появится строка «Обновление» (рис. 6.27);

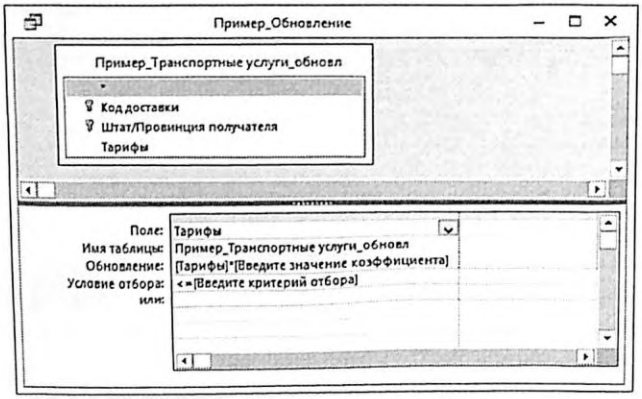


Рис. 6.27. Создание запроса на обновление данных в таблице

3) перетащить в бланк запроса те поля таблицы, которые необходимо обновить или для которых нужно задать условия отбора;

4) для полей, подлежащих обновлению, в строке «Обновление» задаются выражения, значения которых будут новыми значениями обновляемых полей;

5) нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» для выполнения запроса.

Формирование запроса на добавление записей в таблицу БД. Для формирования запроса на добавление записей в таблицу с помощью конструктора нужно выполнить следующие действия:

1) открыть окно конструктора запроса. При этом указать таблицу, из которой добавляются записи в другую таблицу;

2) выполнить команду *контекстная вкладка «Работа с запросами | Конструктор» → группа «Тип запроса» → Добавление* – откроется диалоговое окно «Добавление» (рис. 6.28);

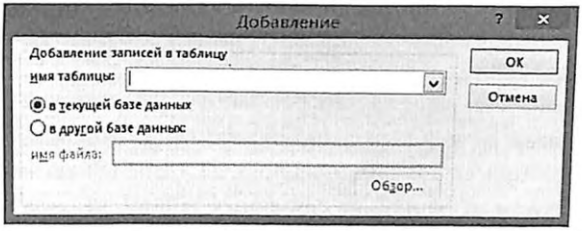


Рис. 6.28. Окно диалога «Добавление»

3) в этом окне указать имя дополняемой таблицы и где эта таблица находится – в текущей или в другой БД. Кроме того, в бланке запроса появится новая строка «Добавление»;

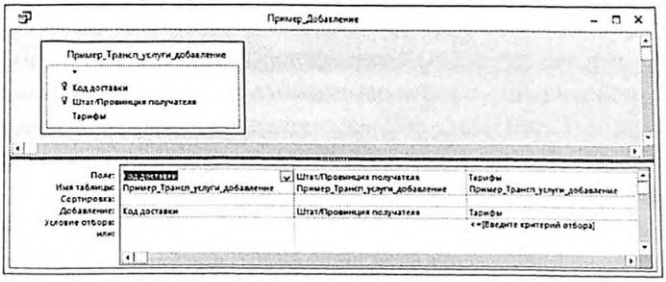


Рис. 6.29. Создание запроса на добавление данных в таблицу

4) перетащить из списка полей в бланк запроса поля, которые необходимо добавить или которые будут использоваться при определении условия отбора;

Примечание. Если в обеих таблицах выбранные поля имеют одинаковые имена, то соответствующие имена автоматически вводятся в строку *Добавление*. Если имена полей двух таблиц отличны друг от друга, нужно указать в строке *Добавление* имена полей таблицы-получателя.

5) нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» для выполнения запроса.

Формирование запроса на удаление записей из таблицы БД. Факторы, которые необходимо учитывать при использовании запроса на удаление:

– при удалении записей с помощью запроса на удаление отменить операцию невозможно. Поэтому, прежде чем выполнить такой запрос, необходимо просмотреть выбранные для удаления данные. Для этого в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» следует нажать кнопку «Режим» и просмотреть запрос в режиме таблицы;

– рекомендуется всегда делать резервные копии данных. Таким образом, если были удалены не те записи, их можно восстановить из резервных копий;

– иногда при выполнении запроса на удаление также могут быть удалены и некоторые записи из связанных таблиц, не включенных в запрос. Это происходит, если в запрос включена только таблица, находящаяся на стороне *один* отношения *один-ко-многим*, и для этого отношения было предусмотрено выполнение каскадных операций. При удалении записей из данной таблицы будут также удаляться записи из таблицы, находящейся на стороне *многие*.

Если между таблицами установлена связь с обеспечением целостности данных, но без каскадного удаления записей, то сначала составляется запрос на удаление записей из подчиненной таблицы, а затем – из главной. Если между таблицами установлена связь без обеспечения целостности данных, то также рекомендуется сначала составить запрос на удаление записей из подчиненной таблицы, а затем – из главной, чтобы после удаления записей БД осталась в непротиворечивом состоянии.

Для формирования запроса на удаление записей с помощью конструктора нужно выполнить следующие действия:

1) создать новый запрос, содержащий таблицы, из которых необходимо удалить записи;

2) в окне конструктора запросов выполнить команду *контекстная вкладка «Работа с запросами | Конструктор»* → *группа «Тип запроса»* → *Удаление* – в результате в бланке запроса появится новая строка «Удаление»;

3) из списка полей подчиненной таблицы перетащить в бланк запроса символ звездочки (*) – в строке *Удаление* для этого поля отобразится значение *Из*;

4) из списка полей главной таблицы перетаскиваются поля, участвующие в условии отбора удаляемых записей – для них в строке *Удаление* появится значение *Условие*;

5) задать условия отбора удаляемых записей (рис. 6.30);

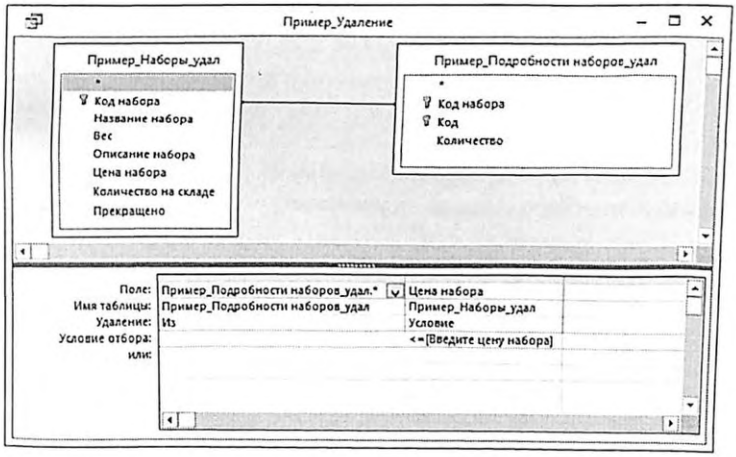


Рис. 6.30. Создание запроса на удаление данных из таблиц, связанных отношением 1:∞ с обеспечением целостности данных, но без выполнения каскадных операций

6) чтобы просмотреть записи, которые будут удалены, в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» следует нажать кнопку «Режим» и просмотреть запрос в режиме таблицы. Чтобы вернуться в режим конструктора запроса, следует снова нажать кнопку «Режим» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор». Если нужно, внести в режиме конструктора необходимые изменения;

7) нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор», чтобы удалить записи из подчиненной таблицы;

8) выделить список полей из подчиненной таблицы и нажать клавишу Delete, чтобы удалить их из запроса;

9) когда в запросе останется лишь главная таблица, а в бланке запроса – поле, для которого установлено условие отбора, снова нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» – Microsoft Access удалит отобранные записи из главной таблицы.

Если между таблицами установлена связь с обеспечением целостности данных и с каскадным удалением записей, то создается запрос на удаление записей только из главной таблицы. Для создания такого запроса выполнить следующие действия:

1) создать новый запрос, содержащий главную таблицу, из которой необходимо удалить записи;

2) в окне конструктора запросов выполнить команду *контекстная вкладка «Работа с запросами | Конструктор» → группа «Тип запроса» → Удаление* – в результате в бланке запроса появится новая строка «Удаление»;

3) в бланк запроса перетащить поля из главной таблицы, участвующие в условиях отбора записей на удаление;

4) задать условия отбора удаляемых записей (рис. 6.31);

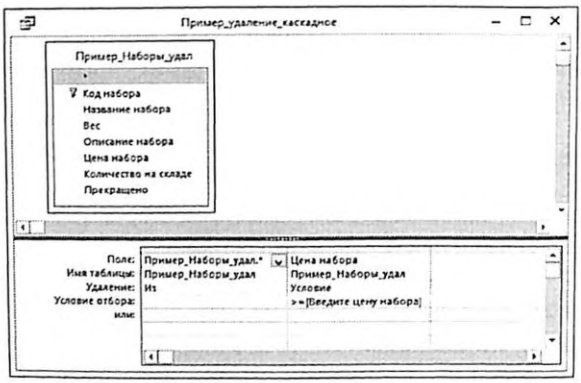


Рис. 6.31. Создание запроса на удаление данных из таблиц, связанных отношением 1:∞ с обеспечением целостности данных и выполнением каскадных операций

5) чтобы просмотреть записи, которое будут удалены из главной таблицы, в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор» следует нажать кнопку «Режим» и просмотреть запрос в режиме таблицы. Чтобы вернуться в режим конструктора запроса, снова нажать кнопку «Режим» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор». Если нужно, внести в режиме конструктора необходимые изменения;

6) нажать кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор», чтобы удалить записи из главной и подчиненной таблиц.

6.9.5. Выполнение и сохранение запроса

После формирования запроса его необходимо выполнить. Из окна конструктора запросов это можно сделать, нажав кнопку «Выполнить» в группе «Результаты» контекстной вкладки «Работа с запросами | Конструктор».

Если результаты выполнения запроса неудовлетворительны, то можно вернуться в окно конструктора запросов для его модификации. Если модификация запроса не требуется, то результат запроса можно сохранить с помощью команды *вкладка Файл → Сохранить* или при закрытии окна конструктора запросов.

6.10. Проектирование форм в СУБД Access

6.10.1. Назначение и способы проектирования форм

Формы служат удобным средством для ввода, просмотра и редактирования информации БД.

Формы, адекватные формам первичных документов, позволяют выполнять загрузку справочных, плановых и оперативно-учетных данных, в любой момент просматривать и редактировать содержимое ранее введенных в базу данных документов, оформлять новые документы.

Формы обеспечивают удобную работу с данными одной или нескольких взаимосвязанных таблиц. Данные из таблиц выводятся на экран с использованием макета формы, разработанного пользователем. Работая с формой, пользователь может добавлять, удалять и изменять записи таблиц, получать расчетные данные. В процессе работы может осуществляться контроль вводимых данных, могут проверяться ограничения на доступ к данным, выводиться необходимые дополнительные сведения.

Существуют *простые* и *составные* (включающие другие формы) формы.

Форма состоит из элементов управления, которые отображают поля таблиц и графические элементы, не связанные с полями таблиц. Элементы управления предназначены, прежде всего, для разработки макета формы: размещения полей таблиц и запросов, надписей, внедряемых объектов (рисунков, диаграмм), вычисляемых полей, а также кнопок, выполняющих печать и открывающих другие объекты или задачи.

Как форма в целом, так и каждый из ее элементов обладают множеством свойств. Посредством их изменения можно настроить внешний вид, размер, местоположение элементов в форме, определить источник данных

формы, режим ввода/вывода, привязать к элементу выражение, макрос или программу. Набор свойств доступен в соответствующем окне, где они разбиты на категории, каждая из которых представлена на своей вкладке. Основными вкладками в окне свойств являются:

– «Макет» – представляет свойства, ориентированные на определение внешнего вида формы или ее элементов;

– «Данные» – представляет свойства для определения источника данных формы или ее элементов, режима использования формы (только ввод, разрешение на изменение, добавление, удаление и т. п.);

– «События» – событиями называют определенные действия, возникающие при работе с конкретным объектом или элементом: нажатие кнопки мыши, изменение данных (до обновления, после обновления), открытие или закрытие формы и т. д. Они могут быть инициированы пользователем или системой. С событием могут связываться макрос или процедура обработки события на языке VBA, выполняющие некоторые действия или рассчитывающие значения.

Пример. В процедуре можно организовать открытие связанной формы, обновление данных таблицы расчетными значениями, печать формы, вывод отчета. Запрограммировав в процедурах вызов различных объектов базы данных, можно автоматизировать выполнение задач приложения.

Способы проектирования форм:

– *«вручную»* – с помощью конструктора форм. Конструктор предоставляет пользователю набор инструментов, с помощью которого пользователь может создать форму соответственно своим вкусам и требованиям;

– *автоматизированным способом* – с помощью мастера форм. Мастера форм предназначены для быстрого создания форм в режиме диалога с пользователем;

– *автоматическим способом* – с помощью инструментов «Форма», «Разделенная форма» или «Несколько элементов» (рис. 6.32). Общим для этих инструментов является то, что они автоматически создают для выбранной таблицы форму, не вступая в диалог с пользователем, и сразу выводят на экран форму в режиме макета.

Однако следует учитывать, что точное формирование макета формы, отвечающего заданным требованиям, дополнение процедурами обработки событий, возникающих в форме, обеспечивается средствами конструирования. Конструктор форм можно использовать как для создания новой формы, так и для редактирования формы, созданной мастером. Кроме того, в Access2007-2013 включены новые функциональные возможности, позволяющие выполнить доработку формы в режиме макета.

Форму можно спроектировать на базе одной или нескольких таблиц и/или запросов. На основе одной таблицы или запроса можно построить несколько форм. В процессе создания формы выбираются поля таблицы, которые должны быть представлены в форме, осуществляется их размещение в форме, создаются вычисляемые поля, графические элементы: кнопки, переключатели, элементы оформления, поясняющий текст и рисунки. Для настройки различных элементов формы используется типовой набор их свойств.

Формы в Access могут быть представлены в трех режимах:

– *режим формы* предназначен для ввода, просмотра и корректировки данных таблиц, на которых основана форма;

– *режим макета* обеспечивает просмотр данных почти в таком виде, в каком они отображаются в режиме формы, и в тоже время позволяет изменять форму. В этом режиме элементы формы становятся выделяемыми, их можно перетаскивать в другие места, редактировать содержимое надписей полей, изменять формат, размер и т. п. Режим макета позволяет удобно настраивать внешний вид формы и может использоваться для внесения большинства структурных изменений. В режиме макета стала доступной вкладка «Конструктор». Однако если некоторую задачу невозможно выполнить в режиме макета, следует переключиться в режим конструктора. В ряде случаев в Access отображается сообщение о том, что для внесения изменений надо переключиться в режим конструктора;

– *конструктор* предназначен для разработки формы с помощью полного набора инструментов, обеспечивающего более детальную проработку структуры формы, использование всех элементов управления. В этом режиме форму можно разработать с нуля или доработать ее после создания мастером. Просмотр данных при внесении изменений в этом режиме не предусматривается.

6.10.2. Однотабличные формы

Однотабличная форма предназначена для загрузки, просмотра и корректировки данных одной таблицы. Источником данных такой формы служит единственная таблица. Она может быть легко создана одним щелчком мыши с помощью инструментов автоматического создания формы: «Форма», «Разделенная форма» или «Несколько элементов», размещенных на вкладке ленты «Создание» в группе «Формы» (рис. 6.32).

Для последующей настройки формы в соответствии с требованиями пользователя ее можно доработать в режиме макета или конструктора. Можно удалить из формы ненужные поля, изменить расположение элементов управления и подобрать их размеры, добавить новые элементы управления, произвести вычисления, задать свойства формы и ее элементов управления.

Рис. 6.32. Команды группы «Формы» на вкладке «Создание»

Формы, созданные с помощью перечисленных команд, отличаются, прежде всего, способом отображения записей базового источника данных. В форме, созданной с помощью инструмента «Форма», одновременно отображается только одна запись, поэтому ее целесообразно использовать, например, в случае, когда таблица включает поля с данными, требующими большого окна для отображения (поле MEMO, диаграмма, фотография). Поля отображаемой записи располагаются внутри одного раздела формы в столбик (макет в столбик) с надписью слева от каждого поля.

В форме, созданной с помощью инструмента «Несколько элементов», отображается сразу несколько записей, и все поля записи размещаются в одной строке, что удобно для ввода данных из простых справочников, имеющих только табличную часть. Этот макет формы аналогичен отображению записей таблицы в режиме таблицы, однако при этом предоставляются многочисленные возможности по настройке отображения и дополнения новыми элементами. Например, в полях такой формы отображаются рисунки.

В форме, созданной с помощью инструмента «Разделенная форма», одновременно отображаются данные в двух представлениях – в одном ее разделе записи отображаются в виде таблицы, в другом выводится единственная выделенная в таблице запись, предназначенная для удобной работы с ее данными.

Создать однотабличную форму можно также с помощью инструмента «Мастер форм», размещенного на вкладке «Создание» в группе «Формы». В диалоговых окнах мастера пользователь выбирает поля, которые надо включать в форму, способ отображения записей, стиль оформления.

6.10.3. Редактирование формы в режиме макета

В режиме макета можно просматривать данные практически так же, как в режиме формы, и в то же время вносить изменения в форму. Это средство функционирует по принципу *WYSIWYG* (что видим, то и получаем), и позволяет вносить изменения в форму и тут же видеть результат внесенных изменений, что очень удобно для уточнения местоположения, размера, шрифта элементов управления, изменения текста надписей и выполнения других задач, связанных с внешним видом и удобством формы. При отображении формы в режиме макета появляются контекстные вкладки ленты «Работа с макетами форм | Конструктор, Упорядочить, Формат» (рис. 6.33-6.36), которые сохраняются на экране, пока активно окно формы и не выполнено переключение в другой режим.

Примечание. При уменьшении размеров окна Access сворачиваются группы ленты. При этом команды группы можно открыть с помощью значка списка ▾.

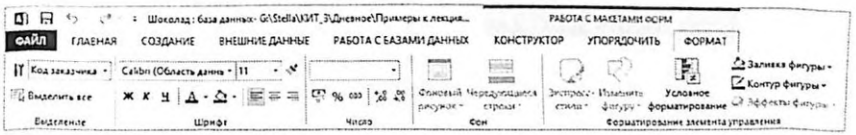


Рис. 6.33. Контекстная вкладка
«Работа с макетами форм | Формат»

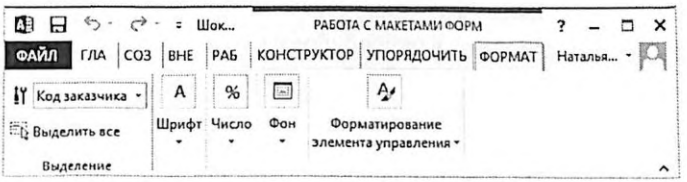


Рис. 6.34. Свернутые группы контекстной вкладки
«Работа с макетами форм | Формат»

Рис. 6.35. Контекстная вкладка
«Работа с макетами форм | Упорядочить»

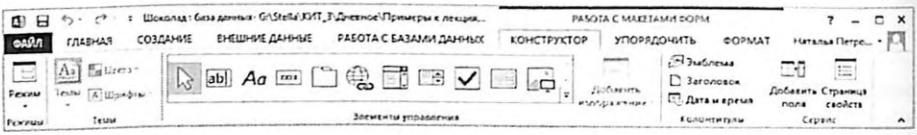


Рис. 6.36. Контекстная вкладка «Работа с макетами форм | Конструктор»

Макеты элементов управления. *Макет элементов управления* – это объединение элементов управления в группу, для которой произведено выравнивание по вертикали и горизонтали для единообразного оформления формы. В форме может быть несколько макетов элементов управления.

Макет можно рассматривать как таблицу, в столбцах которой размещены элементы управления. Естественно в таблице можно менять только ширину всего столбца или всей строки, а не отдельных ее ячеек. В макете, как и в таблице, можно перемещать значение из одной ячейки в другую.

В рамках одного макета невозможно изменение ширины и высоты отдельных элементов, местоположение элемента может меняться только в пределах макета. В тоже время макет при перетаскивании элементов для изменения их местоположения может легко расширяться путем добавления новых строк и столбцов.

Элемент можно удалить из макета командой *контекстное меню элемента → Макет → Удалить макет* и тогда будет обеспечено его свободное перемещение и изменение размеров. Выделив весь макет и выполнив ту же команду, можно полностью разгруппировать элементы управления. Выделив нужную группу элементов, можно создать новый макет в столбик или табличный соответствующими командами группы «Таблица» на контекстной вкладке «Работа с макетами форм | Упорядочить». Именно макет позволяет выравнивать все его элементы.

6.10.4. Свойства формы

Редактирование формы, ее разделов и входящих в нее элементов управления может быть произведено не только графическими средствами, но и путем изменения их свойств. Для изменения свойств предназначена «Страница свойств».

Чтобы открыть свойств следует выполнить команду *контекстная вкладка «Работа с макетами форм | Конструктор» → группа «Сервис» → инструмент «Страница свойств»*.

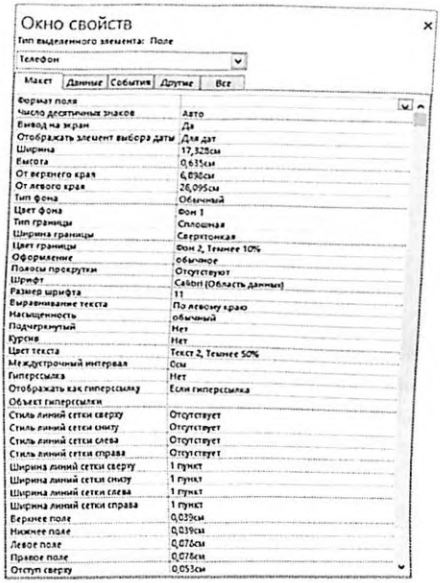


Рис. 6.37. Окно свойств формы

Примечание. Следует заметить, что изменение многих свойств формы и ее элементов управления невозможно в режиме макета.

6.10.5. Добавление полей в форму

В режиме макета возможно удаление и добавление полей в форму. Для удаления поля достаточно выделить его вместе с наименованием и выполнить команду контекстного меню *Удалить*. Можно предварительно поле удалить из макета. Для добавления полей в форму нужно выполнить команду *контекстная вкладка «Работа с макетами форм| Конструктор» → группа «Сервис» → инструмент «Добавить поля»* – отобразится область «Список полей» базовой таблицы или запроса, источника данных формы. Добавить поле в активный макет формы можно двойным щелчком на нем. Для размещения поля в любом макете формы следует перетащить его туда из области «Список полей».

В режиме макета в группе «Элементы управления» на контекстной вкладке «Работа с макетами форм | Конструктор» представлен набор элементов, для включения которых в форму нет необходимости переходить в режим конструктора. Этот набор содержит элементы для создания поля,

надписи, списка, поля со списком, флажка, кнопки и т.п. (рис. 6.38). Набор элементов управления режима макета в сравнении с доступным в режиме конструктора несколько усечен.

Рис. 6.38. Группа «Элементы управления» контекстной вкладки «Работа с макетами форм | Конструктор»

6.10.6. Многотабличные формы

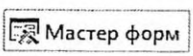
Многотабличная форма создается для работ ы с данными нескольких взаимосвязанных таблиц. Источником данных такой формы является многотабличный запрос. При этом форма также может быть простой, отображающей одну запись в столбик, или ленточной, отображающей все записи в табличном виде с надписями в заголовке формы. Для создания такой формы могут быть использованы инструменты «Форма» или «Несколько элементов». Форма, построенная на многотабличном запросе, может быть названа *одиночной*.

Многотабличная форма может быть *составной*: состоять из главной формы и одной или нескольких подчиненных форм. Подчиненная форма, как правило, строится на основе таблицы, подчиненной таблице-источнику записей главной формы, т. е. находится с ней в отношении 1:М. Подчиненная форма отображает данные из всех записей подчиненной таблицы, которые связаны с записью, отображаемой в главной форме.

Основным средством создания многотабличных форм можно считать мастер форм, который, запросив у пользователя сведения о включаемых в форму полях из нескольких взаимосвязанных таблиц и запросов, создает составную или одиночную форму.

Полученная с помощью мастера форма при необходимости может быть отредактирована в режиме макета или конструктора. Режим конструктора позволяет детально просмотреть структуру формы и настроить любой ее раздел. Некоторые задачи удобнее выполнять в режиме конструктора, а не макета, некоторые могут выполняться только в режиме конструктора.

Для вызова мастера форм нужно выполнить команду *вкладка ленты «Создание» → группа «Формы» → инструмент «Мастер форм»* (рис. 6.39). Дальнейшее создание формы ведется, следуя указаниям мастера форм.



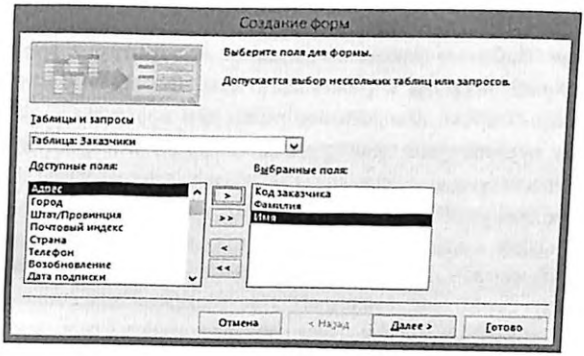


Рис. 6.39. Стартовое окно мастера форм

6.11. Проектирование отчетов в СУБД Access

6.11.1. Назначение и способы проектирования отчетов

Отчет является важным средством визуализации информации, хранящейся в БД. Под визуализацией понимается вывод на экран или на печать в виде, удобном для восприятия и анализа пользователем. В отчете можно сортировать и группировать данные, осуществлять расчеты в строках и проводить итоговые вычисления над группами строк и над всеми строками с использованием статистических функций. Отчет может основываться на множестве таблиц и представлять сложные зависимости между различными наборами данных. Он может быть составным, т. е. включать другие отчеты.

Access предоставляет большие возможности по оформлению отчетов: шрифтовое, фоновое и цветовое оформление, обрамление, рисунки, деловая графика, вставка объектов других приложений. Все это позволяет создавать отчеты высокого качества.

Способы проектирования отчетов:

– *«вручную»* – с помощью конструктора отчетов. Конструктор дает возможность самостоятельного проектирования отчетов;

– *автоматизированным способом* – с помощью мастера отчетов. Мастер отчетов позволяет создать отчет на основе ответов пользователя на вопросы, касающиеся структуры, содержания и оформления отчета;

– *автоматическим способом* – с помощью инструмента «Отчет». Это средство автоматически создает для выбранного объекта базы данных отчет, не вступая в диалог с пользователем, и сразу выводит на экран отчет в режиме макета.

Инструменты разработки отчетов представлены на контекстных вкладках ленты «Работа с макетами отчетов» и «Инструменты конструктора отчетов», которые связаны с режимами изменения и просмотра отчетов, и появляются при переходе из одного режима в другой.

В Access существуют два представления, в которых можно вносить изменения в отчет: *режим макета* и *режим конструктора*. Режим макета является наиболее удобным для внесения изменений в отчет, поскольку пользователь сразу видит данные отчета. В этом режиме доступны большинство инструментов, необходимых для его настройки. В нем можно изменить ширину столбцов, поменять их местами, добавить или изменить уровни группировки и итоги. Можно также разместить в макете отчета новые поля, а также задать свойства отчета и элементов управления.

В режиме конструктора отображаются разделы отчета и доступны дополнительные инструменты и возможности разработки. Следует перейти в режим конструктора, если не удается выполнить изменения в режиме макета. В определенных случаях в Access отображается сообщение о том, что для внесения изменений следует переключиться в режим конструктора.

Просматривать отчет можно в режимах «Представление отчета», «Предварительный просмотр» или «Макет». В режиме «Представление отчета» можно отфильтровать данные для отображения только заданных строк, найти нужные данные, скопировать текст отчета или его часть в буфер обмена. Режим предварительного просмотра предназначен для просмотра отчета перед печатью. В этом режиме можно увеличивать масштаб для просмотра деталей или уменьшать его для проверки размещения данных на странице, изменять параметры страницы. Режим макета позволяет, просматривая данные отчета, изменять его макет.

6.11.2. Однотабличные отчеты

Наиболее простым способом создания отчета на основе таблицы или запроса является использование команды «Отчет». С помощью этого средства отчет формируется без диалога с пользователем и выводит все поля выбранного источника. В дальнейшем созданный таким способом отчет может быть доработан в режиме макета или конструктора.

6.11.3. Многотабличные отчеты

Отчеты в Microsoft Access могут создаваться не только на основе записей одного объекта БД, но и на основе информации, содержащейся в нескольких объектах БД.

Мастер отчетов позволяет в режиме диалога с пользователем создать многотабличный отчет путем выбора необходимых объектов БД и полей, определения полей группировки, итоговых значений для записей. Создание отчета мастером является простой процедурой, а полученный отчет без больших усилий может быть приведен к желаемому виду.

Многотабличные отчеты, так же как формы, могут состоять из главного отчета и включаемого в него подчиненного отчета. Для каждого из этих отчетов в качестве источника данных выбираются своя таблица или запрос, построенный на нескольких таблицах.

Настройка отображения полученного отчета, как и настройка форм, выполняется в режиме макета и конструктора.

6.11.4. Инструментальные средства конструкторов форм и отчетов

Средствами конструктора форм можно создать любую форму (отчет) с нуля. Как уже упоминалось ранее, в режиме конструктора также можно в любой момент отредактировать ранее разработанную любыми средствами форму (отчет).

При создании и редактировании формы (отчета) может выполняться: определение источника данных формы (отчета), добавление новых полей и надписей, включение полей со списком, создание кнопок, добавление подчиненных форм (отчетов), внедрение объектов из других приложений, например, рисунков, диаграмм и т. п.

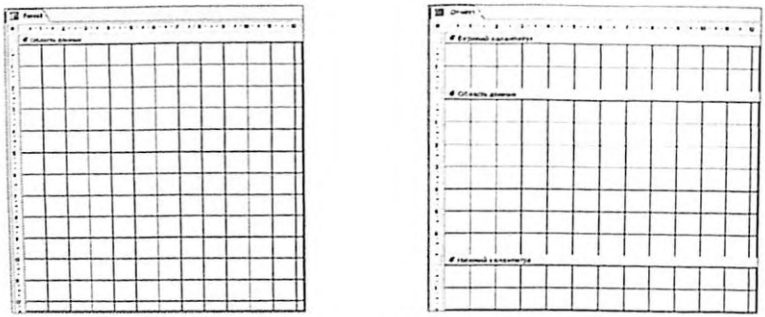
Наиболее точная и полная настройка структуры и внешнего вида всех разделов и элементов формы (отчета) производится в режиме конструктора. Ряд элементов управления доступен только в режиме конструктора.

Для того чтобы приступить к созданию формы (отчета) в режиме конструктора нужно выполнить одно из следующих действий:

– выполнить команду *вкладка ленты «Создание»* → *группа «Формы»* → *инструмент «Конструктор форм»*. В результате выполнения команды открывается окно пустой формы в режиме конструктора с именем по умолчанию – «Форма1». В пустой форме конструктора представлен только один раздел – «Область данных» (рис. 6.40, а);

– выполнить команду *вкладка ленты «Создание»* → *группа «Отчеты»* → *инструмент «Конструктор отчетов»*. В результате выполнения команды открывается окно пустого отчета в режиме конструктора с именем по умолчанию – «Форма1». В пустом отчете конструктора представлены три раздела: «Верхний колонтитул»,

«Область данных» и «Нижний колонтитул» (рис. 6.40, б);



а) б)

Рис. 6.40. Начальный вид при создании в режиме конструктора: а) формы и б) отчета

Начать конструирование формы (отчета) следует с определения источника данных. Источник данных может быть создан различными способами. Можно открыть свойства формы (отчета) и на вкладке «Данные» в строке «Источник записей» открыть список и выбрать таблицу или запрос.

Наиболее простым способом определения источника записей является использование области «Список полей», в которой отображаются поля всех таблиц базы данных. Для отображения списка полей нужно выполнить одну из двух команд:

– *контекстная вкладка «Инструменты конструктора форм | Конструктор» → группа «Сервис» → инструмент «Добавить поля»;*

– *контекстная вкладка «Инструменты конструктора отчетов | Конструктор» → группа «Сервис» →инструмент «Добавить поля».*

Так как методика проектирования формы и отчета одинакова, то и для формы, и для отчета используются сходные инструментальные средства конструирования:

1) для форм – контекстная вкладка «Инструменты конструктора форм | Конструктор. Упорядочить. Форма» (рис. 6.41–6.43);

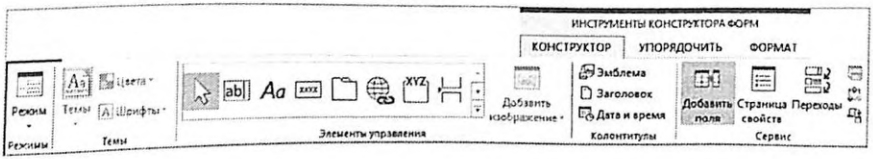


Рис. 6.41. «Инструменты конструктора форм | Конструктор»

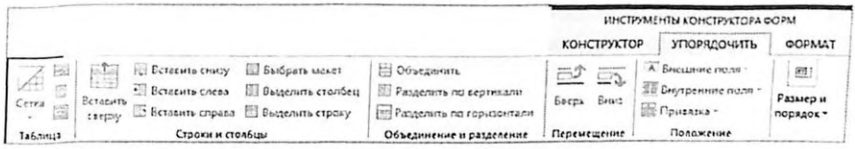


Рис. 6.42. «Инструменты конструктора форм | Упорядочить»

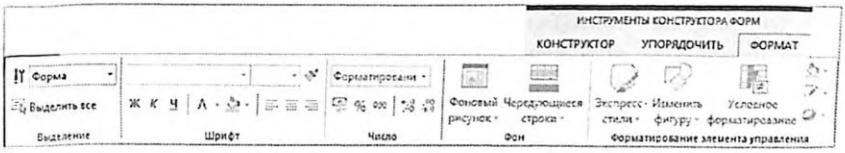


Рис. 6.43. «Инструменты конструктора форм | Форма»

2) для отчетов – контекстная вкладка «Инструменты конструктора отчетов | Конструктор. Упорядочить. Форма. Параметры страницы» (рис. 6.44-6.48);

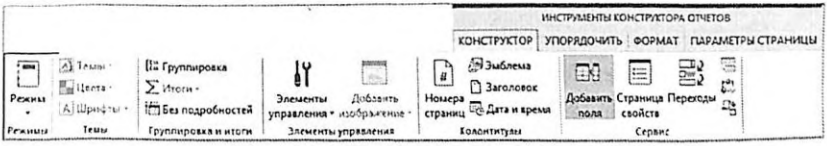


Рис. 6.44. «Инструменты конструктора отчетов | Конструктор»

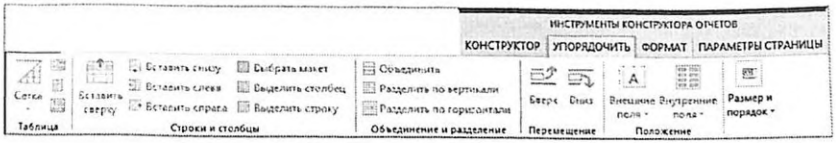


Рис. 6.45. «Инструменты конструктора отчетов | Упорядочить»

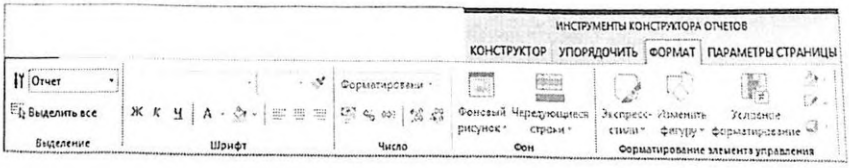


Рис. 6.47. «Инструменты конструктора отчетов | Формат»

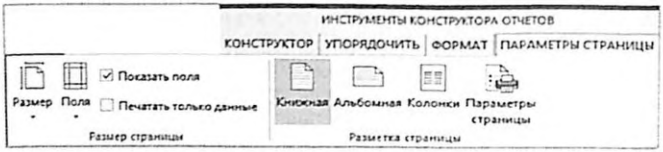


Рис. 6.48. «Инструменты конструктора отчетов | Параметры страницы»

3) разделы формы (отчета), в которых размещают различные элементы управления, определяющие их вид и содержание;

4) элементы управления, которые находятся в одноименной группе контекстной вкладки «Инструменты конструктора форм | Конструктор» или «Инструменты конструктора отчетов | Конструктор» (рис. 6.49);

Рис. 6.49. Элементы управления конструкторов форм и отчетов

5) список полей таблицы или запроса (рис. 6.50), служащих источником данных для формы (отчета). Он включается/выключается по команде *контекстная вкладка «Инструменты конструктора форм (отчетов) | Конструктор» → Добавить поля*;

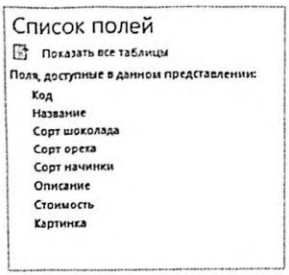
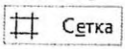
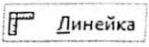


Рис. 6.50. Список полей, служащий источником данных для создания формы или отчета

6) сетка, отображаемая в разделах формы (отчета) для удобства проектирования. Она включается/выключается по команде *контекстное меню формы (отчета) → Сетка*;

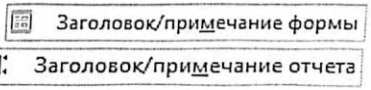


7) горизонтальная и вертикальная линейки, включаемые/выключаемые по команде *менюВид* → *Линейка*. Они также предназначены для удобства проектирования.

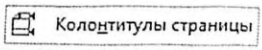


Кроме разделов формы (отчета), выводимых по умолчанию, при конструировании могут быть использованы и другие разделы:

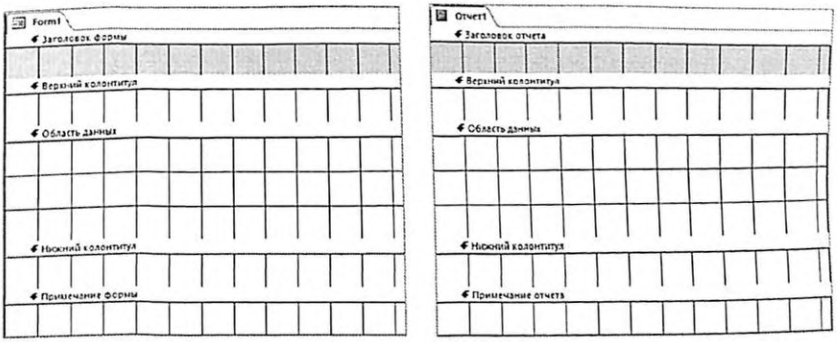
8) *заголовок формы (отчета)* и *примечание формы (отчета)*, включаемые/выключаемые попарно соответственно командами: *контекстное меню формы* → *Заголовок/примечание формы* и *контекстное меню отчета* → *Заголовок/примечание отчета*;



9) *верхний колонтитул* и *нижний колонтитул*, включаемые/выключаемые попарно командой *контекстное меню формы (отчета)* → *Колонтитулы страницы*.



Если в окне конструктора формы (отчета) включены все названные разделы, то они располагаются в рабочей области этого окна в такой последовательности, как показано на рисунке 6.51.



а) б)

Рис. 6.51. Расположение разделов в окне конструктора:
а) формы, б) отчета

Заголовок формы (отчета) содержит сведения, общие для всех записей, такие как название формы (отчета), дата и время. *Примечание формы (отчета)* отображает сведения, общие для всех записей, такие как инструкции по работе с формой, командные кнопки, общие суммы в счетах. *Верхний колонтитул* и *нижний колонтитул* печатаются вверху и внизу на каждой выводящейся на печать странице; они не отображаются в режиме формы. *Верхний колонтитул* используется для вывода заголовков столбцов, даты

или номера страницы сверху на каждой странице формы или отчета. В формах верхние колонтитулы отображаются только при печати. *Нижний колонтитул* используется для вывода итоговых значений по странице, даты или номера страницы снизу на каждой странице формы или отчета. В формах нижние колонтитулы отображаются только при печати.

Примечание. При необходимости группировки строк в отчете в окно конструктора отчета должны быть добавлены еще два раздела: *заголовок группы* и *примечание группы*. Эти разделы можно получить в отчете по команде *контекстная вкладка «Инструменты конструктора отчетов | Конструктор» → группа «Группировка и итоги» → «Группировка»*. В разделе «Заголовок группы» отображается заголовок для групп строк отчета. Данный раздел размещается перед областью данных. Раздел «Примечание группы» служит для итоговых вычислений для групп строк и размещается после области данных.

Изменение высоты и ширины раздела производится путем перетаскивания соответственно нижней и правой границ раздела.

6.11.5. Элементы управления и работа с ними

Элементы управления, которыми может пользоваться разработчик, представлены на *панели элементов*, которая содержит следующие инструменты:

Таблица 6.6

Внешний вид и назначение кнопок панели элементов

Кнопка	Назначение
Выбор объектов	Выделение элемента управления, раздела проекта формы, отчета
Мастера	Кнопка включения/отключения автоматического вызова Мастеров создания элементов управления. Существуют Мастера по созданию группы, поля со списком, списка и командной кнопки
Надпись	Создание элемента управления, содержащего неизменяемый текст. По умолчанию большинство элементов управления содержат присоединенный текстовый элемент
Поле	Используется для отображения, ввода или изменения данных, содержащихся в источнике записей, вывода результатов вычислений, а также приема данных, вводимых пользователем
Группа переключателей	Служит для создания группы элементов (флажков, переключателей или выключателей), представляющих набор альтернативных значений, из которых выбирается одно значение

Кнопка	Назначение
Выключатель  Переключатель  Флажок 	Предназначены для отображения логических значений. Выбор (включение) элемента приводит к вводу в соответствующее логическое поле значения «Да», «Истина» или «Вкл» (определяется значением свойства поля *Формат поля*). Повторный выбор элемента изменяет значение на противоположное: «Нет», «Ложь» или «Выкл»
Поле со списком 	Создает комбинированный список, который позволяет выбирать значение из списка или вводить его вручную
Список 	Предназначен для создания списка возможных значений. Список можно создать, явно вводя данные, либо указав источник данных – таблицу или запрос
Кнопка 	Обычно используется для запуска закрепленного за ней макроса или программы на языке Visual Basic
Рисунок 	Предназначен для размещения в форме неизменяемого рисунка
Свободная рамка объекта 	Используется для размещения объекта из приложения, поддерживающего технологию OLE. Включаемый объект становится частью формы, но не хранится в таблице БД. В качестве объекта может быть электронная таблица, рисунок, диаграмма, звуковой файл и т. д.
Присоединенная рамка объекта 	Используется для включения в форму OLE-объектов в виде ссылок
Разрыв страницы 	Позволяет вставлять разрыв страницы в многостраничной форме
Вкладка 	Позволяет создать в форме несколько вкладок, каждая из которых может содержать другие элементы управления
Подчиненная форма/отчет 	Предназначен для внедрения в форму некоторой другой (подчиненной) формы
Линия  Прямоугольник 	Предназначены для создания соответствующих геометрических фигур
Элементы ActiveX 	Кнопка, после нажатия которой открывается список всех установленных в системе элементов управления ActiveX (Элемент ActiveX – это элемент управления (такой как флажок или кнопка), служащий для выбора параметров, либо для запуска макроса или сценария, автоматизирующего выполнение задачи)

Элементы управления делятся на связанные и несвязанные. *Связанный* элемент управления присоединен к полю базовой таблицы или запроса. Такие элементы управления используются для просмотра, ввода или редактирования значений из полей БД. Ввод связанного элемента управления в проект формы (отчета) осуществляется путем перетаскивания из списка полей нужного поля в раздел «Область данных». В результате появляются два прямоугольника – *подпись поля* и *поле* (рис. 6.52).

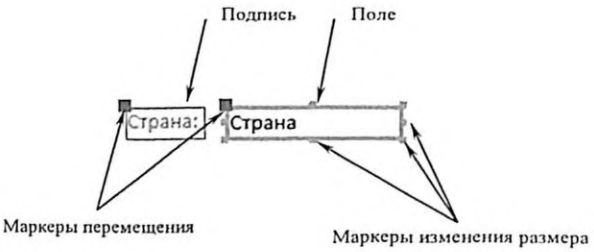


Рис. 6.52. Пример связанного элемента управления

Маркеры перемещения служат для отдельного перемещения подписи и поля. При установке на них указателя мыши появляется изображение двунаправленной стрелки и тогда следует выполнять перемещение.

Для одновременного перемещения подписи и поля указатель мыши устанавливается на границу выделенной подписи или выделенного поля и, когда он принимает вид двунаправленной стрелки, производят перетаскивание. Для удаления подписи осуществляют ее выделение и нажимают клавишу Delete; для удаления поля с подписью выделяют поле и нажимают клавишу Delete.

Для несвязанных элементов управления источника данных не существует. Такие элементы управления используются для отображения линий, прямоугольников, рисунков. Ввод в проект формы (отчета) несвязанного элемента управления осуществляется так: на панели элементов производится щелчок по кнопке нужного элемента управления, а затем щелчок в том разделе проекта, где требуется разместить этот элемент.

Навыки конструирования формы (отчета) предполагают умение выделения группы элементов управления. Оно производится путем щелчков по элементам управления, входящим в группу, при нажатой клавише Shift. Выделить группу смежных элементов можно и путем очерчивания рамки

вокруг этой группы. Чтобы изменить размер элементов группы, произвести их выравнивание, изменить интервал между ними, необходимо воспользоваться контекстной вкладкой «Инструменты конструктора форм (отчетов) | Упорядочить».

Перенос и копирование элементов управления в другой раздел формы (отчета) возможны посредством команд для работы с буфером обмена.

Для придания элементу управления необходимых свойств его выделяют и выполняют команду *контекстная вкладка «Инструменты конструктора форм (отчетов)* | *Конструктор»* → *Страница свойств* или вызывают контекстное меню и выбирают в нем пункт *Свойства*. В результате появляется окно свойств элемента управления (рис. 6.53), в котором, например, вкладка «Макет» позволит произвести его желаемое форматирование.

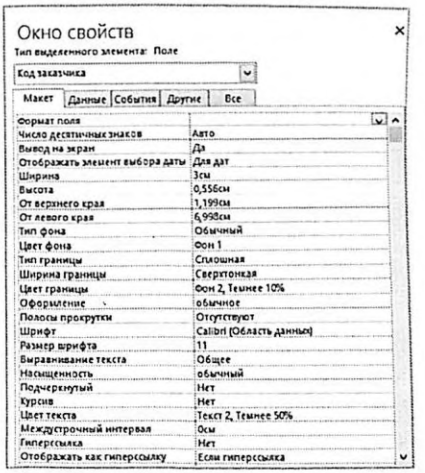


Рис. 6.53. Окно свойств поля «Код заказчика»

6.11.6. Рекомендации по созданию формы

Пусть необходимо спроектировать форму, в которой должны быть определенные заголовок, поля БД, вычисляемое поле и комментарий (на основе таблицы «Наборы»). Для этого нужно выполнить следующие действия:

1) перейти в режим конструктора формы, указав источник формы;

2) настроить это окно для работы, включив, если необходимо линейки, сетку, панель элементов и три раздела: «Заголовок формы», «Область данных», «Примечание формы»;

3) задать заголовок формы, для чего в раздел «Заголовок формы» поместить элемент управления «Надпись» и ввести текст заголовка;

4) из области со списком полей перетащить в раздел «Область данных» те поля БД, значения которых должны просматриваться в форме, и расположить их в нужном порядке;

5) создать вычисляемое поле. Для этого поместить в раздел «Область данных» свободное поле, используя элемент управления «Поле». В само поле ввести выражение для расчета, а в подпись – имя вычисляемого поля;

6) задать необходимый комментарий в разделе «Примечание формы», используя элемент управления «Надпись»;

7) произвести желаемое оформление формы (рис. 6.54, 6.55).

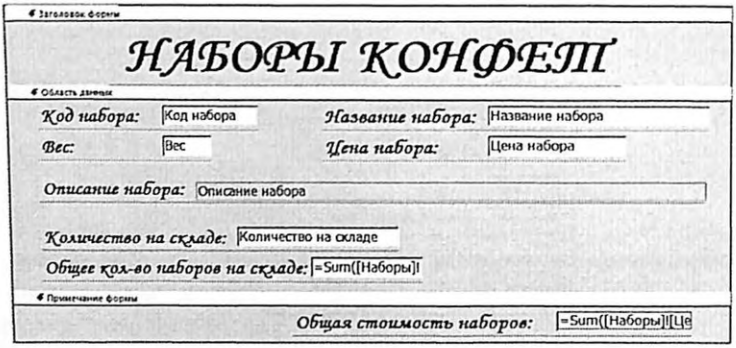


Рис. 6.54. Создание формы в Конструкторе форм

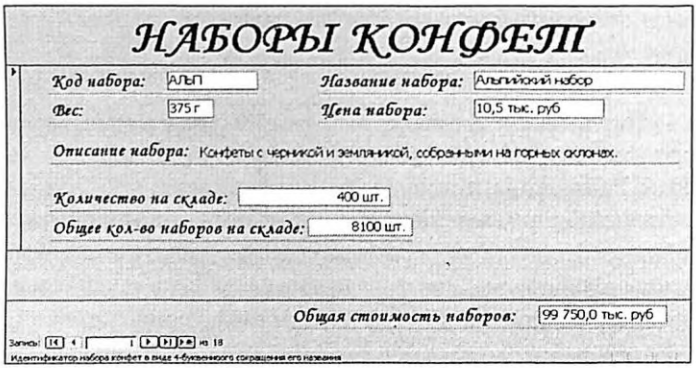
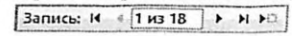


Рис. 6.55. Так выглядит созданная форма в режиме формы

6.11.7. Работа с формой

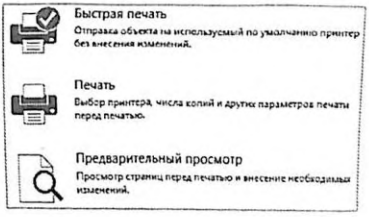
Просмотреть, как выглядит спроектированная форма можно в окне формы, открываемом из области навигации двойным щелчком левой кнопки мыши на имени формы, из режима конструктора или режима макета по команде *контекстная вкладка «Инструменты конструктора форм | Конструктор» → группа «Режимы» → Режим формы (таблицы)*.

В *режиме формы* в форме отображается только одна запись, а в *режиме таблицы* – группа записей. В обоих режимах можно редактировать данные, а также просматривать их с помощью кнопок перехода к записям (к предыдущей, к следующей, к первой, к последней). Кнопка перехода к пустой записи позволяет добавить в базу новые данные.



Форму можно сохранить, либо выполнив однократный щелчок левой кнопкой мыши по кнопке «Сохранить» на панели быстрого доступа, либо при закрытии окна формы.

Форма может быть предварительно просмотрена из области навигации, из режима конструктора, из режима макета и из режима формы по команде *вкладка «Файл» → Печать → Предварительный просмотр* и распечатана по команде *вкладка «Файл» → Печать → Печать* или по команде *вкладка «Файл» → Печать → Быстрая печать*. При печати ее заголовок появляется только в начале первой страницы, а ее примечание – в конце последней. Верхний и нижний колонтитулы видны только при печати, соответственно в начале и в конце каждой страницы.



Изменения в форму вносятся в режиме конструктора или в режиме макета.

6.11.8. Рекомендации по созданию отчета

Пусть требуется создать отчет, в котором имеется несколько расчетных столбцов, должна быть группировка строк по некоторому полю, необходимы итоги по некоторым полям для групп строк (частные итоги) и общие итоги. Для создания такого отчета необходимо:

1) перейти в режим конструктора отчета, указав источник отчета;

2) настроить это окно для работы, включив (если не включены) линейки, сетку, панель элементов и все необходимые разделы;

3) задать заголовок отчета, для чего в раздел «Заголовок отчета» вставить элемент управления «Надпись» и ввести текст заголовка;

4) спроектировать строки отчета:
   - из окна со списком полей перетащить в раздел «Область данных» те поля, значения которых должны выводиться в строках отчета;
   - подписи всех полей переместить из раздела «Область данных» в раздел «Верхний колонтитул»;
   - поля в разделе «Область данных» расположить в строку в нужном порядке, выровнять их по верхнему краю раздела и сделать между ними равные интервалы по горизонтали.
5) спроектировать «шапку» отчета:
   - в разделе «Верхний колонтитул» подписи полей расположить в строку над соответствующими полями;
   - изменить их на полные названия столбцов. Названия столбцов выровнять по верхнему краю раздела.
6) спроектировать частные итоги:
   - в раздел «Заголовок группы» ввести заголовок для групп строк при помощи элемента управления «Надпись»;
   - в раздел «Примечание группы» вставить свободные поля, используя элемент управления «Поле», и в них ввести выражения для подсчета частных итогов. Подписи этих полей удалить;
   - созданные вычисляемые поля расположить в строку, под полями, по которым подводятся частные итоги, и выровнять их по верхнему краю раздела. В этой строке слева ввести название для строк с частными итогами при помощи элемента управления «Надпись».
7) спроектировать общие итоги:
   - в раздел «Примечание отчета» вставить свободные поля, используя элемент управления «Поле», и в них ввести выражения для подсчета общих итогов. Подписи этих полей удалить;
   - созданные вычисляемые поля расположить в строку, под полями, по которым подводятся общие итоги, и выровнять их по верхнему краю раздела. В этой строке слева ввести название для строки с общими итогами при помощи элемента управления «Надпись».
8) предусмотреть нумерацию страниц отчета, воспользовавшись командой *контекстная вкладка «Инструменты конструктора отчетов | Конструктор» → группа «Колонтитулы» → Номера страниц*;

9) выполнить оформление отчета, например:
   - в разделе «Заголовок отчета» провести нижнюю и верхнюю линии шапки отчета, используя элемент управления «Линия»;

– выделить разными шрифтами шапку отчета, вычисляемые поля для частных и общих итогов;
– выполнить предварительный просмотр отчета. Если отчет устраивает, сохранить и напечатать его; в противном случае модифицировать его в конструкторе.

6.11.9. Создание диаграмм в формах и отчетах

Для того чтобы создать диаграмму в форме или отчете, необходимо выполнить следующие действия:

1) открыть форму (отчет) в режиме конструктора;

2) выполнить команду *контекстная вкладка «Инструменты конструктора отчетов | Конструктор» → группа «Элементы управления» → Диаграмма* ;

3) щелкнуть в области данных формы (отчета) – откроется первое окно мастера диаграмм (рис. 6.56);

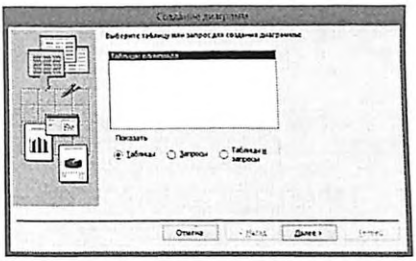


Рис. 6.56. Первое окно диалога мастера диаграмм

4) в этом окне выбрать источник данных для диаграммы – таблицу или запрос;

5) нажать кнопку Далее – откроется второе окно мастера (рис. 6.57);

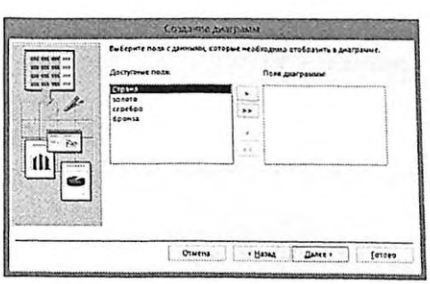


Рис. 6.57. Второе окно диалога мастера диаграмм

6) в этом окне выбрать поля для отображения в диаграмме;

7) нажать кнопку Далее – откроется третье окно диалога мастера создания диаграмм (рис. 6.58);

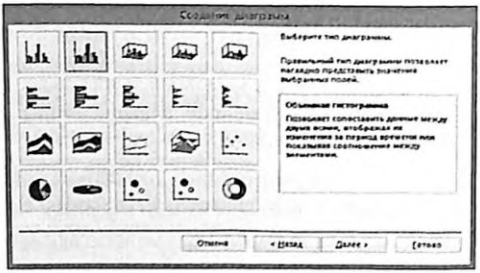


Рис. 6.58. Третье окно диалога мастера диаграмм

8) в этом окне выбрать тип диаграммы;

9) нажать кнопку Далее – откроется четвертое окно диалога мастера диаграмм (рис. 6.59);

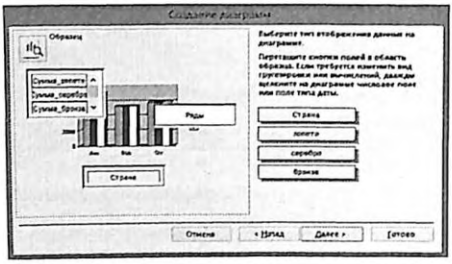


Рис. 6.59. Четвертое окно диалога мастера диаграмм

10) в этом окне выбрать тип отображения данных на диаграмме путем перетаскивания кнопок полей в область образца;

11) если необходимо, изменить вид группировки или вычислений:

- выполнить двойной щелчок на нужной кнопке поля в области образца – откроется окно диалога «Вычисление итоговых значений» (рис. 6.60);
- в этом окне выбрать необходимую статистическую функцию или вариант *Отсутствует*;
- ОК.

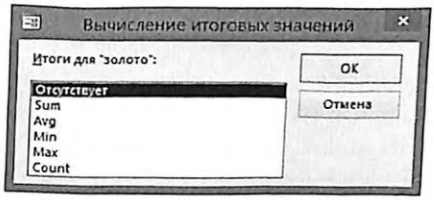


Рис. 6.60. Окно диалога «Вычисление итоговых значений»

12) нажать кнопку Далее – откроется пятое окно диалога мастера диаграмм (рис. 6.61);

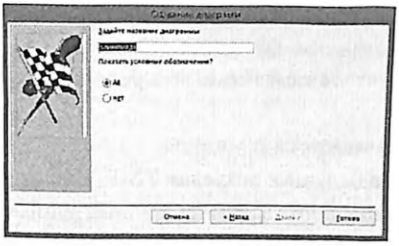


Рис. 6.61. Пятое окно диалога мастера диаграмм

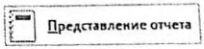
13) в этом окне присвоить диаграмме имя;

14) нажать кнопку Готово – откроется отчет с созданной диаграммой.

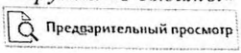
Примечание. При необходимости диаграмму можно отредактировать в режиме конструктора.

6.11.10. Работа с отчетом

Спроектированный отчет может быть предварительно просмотрен в одном из двух представлений: *представление отчета* и *предварительный просмотр*.

В режим представления отчета можно перейти по 

команде *контекстная вкладка «Инструменты конструктора отчетов | Конструктор» → группа «Режимы» → Режим «Представление отчета»*. Он позволяет оценить внешний вид отчета, не вникая в подробности.

Предварительный просмотр дает возможность увидеть отчет таким, каким он будет после печати. В этот режим можно перейти по команде *«Инструменты конструктора отчетов | Конструктор» → группа «Режимы» → Режим «Предварительный просмотр»*, а также из 

окна БД по команде *вкладка «Файл» → Печать → Предварительный просмотр*.

Отчет можно сохранить либо выполнив однократный щелчок левой кнопкой мыши по кнопке «Сохранить» на панели быстрого доступа,  либо при закрытии окна отчета.

Печать отчета можно выполнить из области навигации или из любого режима по команде *вкладка «Файл» → Печать → Печать* или по команде *вкладка «Файл» → Печать → Быстрая печать*. Установка параметров печати осуществляется перед печатью с помощью инструментов контекстной вкладки «Инструменты конструктора отчетов | Параметры страницы».

Изменения в отчет вносятся в окне конструктора.

6.12. Расширение функциональности баз данных с помощью макросов

6.12.1. Понятие макроса и модуля

Как видно из предыдущих разделов УМК, база данных Access – это не только набор таблиц, хранящих взаимосвязанные данные, но и набор средств для работы с данными, таких как запросы, формы, отчеты. При разработке приложения с помощью Access эти объекты составляют его основу, а формы и отчеты называются *объектами приложения*. При этом формы составляют *основу интерфейса*, обеспечивая интерактивный ввод, просмотр и изменение данных. Может быть создана форма для управления приложением, представляющая все его подзадачи и обеспечивающая пользователя простым доступом ко всем его функциям. Отчеты обеспечивают обобщенное представление результатов обработки данных и их вывод на экран и печать.

Для автоматизации выполнения задач, связи различных объектов, создания, редактирования и автоматизации более сложной логики приложений необходимо использовать *средства программирования*: макросы и модули.

Объектно-ориентированный язык программирования VBA (Visual Basic for Applications), являясь стандартом макропрограммирования для всего семейства Microsoft Office, позволяет создавать в среде Access целостные графические диалоговые приложения пользователя с большими возможностями по управлению и контролю за их выполнением, решать и автоматизировать выполнение самых сложных задач. На языке VBA создаются *модули* – объекты БД, содержащие созданные пользователем процедуры. *Процедура* представляет собой совокупность операторов языка VBA, реализующую ряд логических шагов для выполнения конкретного действия.

Однако простейшим способом, позволяющим расширять функциональные возможности и автоматизировать выполнение задач приложения, является *создание макросов*.

Макрос представляет собой набор из одной или нескольких *макрокоманд*, каждая из которых выполняет определенное действие. Использование макросов для автоматизации управления реакцией приложения на действия пользователя в формах или отчетах позволяет создавать полноценные интерактивные приложения без написания кода на VBA. Макросы обеспечивают выполнение части команд, доступных в VBA, а для пользователей создание макроса оказывается проще, нежели написание кода VBA. Конструктор макросов снабжен ясным и понятным интерфейсом и позволяет не только избежать написания кода VBA, но и упростить процесс добавления функциональных возможностей в приложение базы данных.

Пример. С помощью макроса можно найти и отфильтровать записи, необходимые для составления отчета; встроить в форму командную кнопку для поиска нужных данных и др.

6.12.2. Классификация макрокоманд. Типы макросов

Как отмечалось выше макрос представляет собой набор из одной или нескольких макрокоманд. *Макрокоманда* – это инструкция, ориентированная на выполнение определенного действия над объектами Access и их элементами. Макрокоманды можно классифицировать по назначению следующим образом:

– макрокоманды для работы с данными в формах и отчетах (например, *НайтиЗапись*);

– макрокоманды выполнения (например, *ОткрытьЗапрос*);

– макрокоманды импорта/экспорта (например, *ОтправитьОбъект*);

– макрокоманды для работы с объектами БД (например, *КопироватьОбъект*) и др.

Макрос, в котором макрокоманды выполняются последовательно, одна за другой, называется *линейным*. Макрос, в котором отдельные макрокоманды или их наборы выполняются в зависимости от выполнения некоторого условия (задаваемого логическим выражением), называется *макросом с условием*. Для перехода по различным ветвям макроса с условием используется блок управления Если.

Макрос, наряду с другими объектами, может быть представлен как отдельный объект – *изолированный макрос*, который отображается в области навигации в группе «Макросы». Кроме того, макрос, связанный с любым событием в форме, отчете или элементе управления, может быть внедрен

в форму или отчет – *внедренный макрос*. При этом он не отображается как объект в группе «Макросы», а становится компонентом формы или отчета.

Изолированный макрос может выполняться в ответ на многочисленные виды событий, возникающих в формах, отчетах и их элементах управления.

Внедренный макрос всегда связывается с событием и сохраняется в форме или отчете. События наступают, прежде всего, при выполнении определенных действий пользователя с объектами. Примерами событий являются: изменение данных в поле, открытие или закрытие формы или отчета, нажатие кнопки в форме и просто передача фокуса от одного поля к другому. Связь макросов с событиями позволяет автоматизировать приложения, используя макросы для открытия форм, печати отчетов, выполнения последовательности запросов, для выполнения действий, зависящих от значений некоторого поля в базе данных, для вывода пользовательских сообщений или отключения предупреждающих сообщений во время выполнения запросов действия и многого другого. Сохранение внедренных макросов вместе с формами и отчетами упрощает управление объектами приложения.

Access предоставляет возможность создавать в макросе *вложенные макросы*. Вложенный макрос имеет имя и может содержать любые макрокоманды. Работать с объединенными макросами часто оказывается удобнее, чем с несколькими отдельными макросами. Целесообразно объединять несколько макросов в одном, если они связаны с решением одной задачи или используются при работе с одним объектом. В области навигации макрос с вложенными макросами отображается как один объект. Пользователь запускает главный макрос на выполнение и далее все управление выполнением задачи осуществляется изнутри макроса. Макрос сам открывает нужные объекты, выбирает и обрабатывает данные, вызывает другие макросы, следуя алгоритму, приводящему к решению задачи. При необходимости из макроса может быть инициирован диалог с пользователем.

Вызывается вложенный макрос с помощью макрокоманды `ЗапускМакроса` или в ответ на событие. Для ссылки на вложенный макрос используется следующий синтаксис:

```
ИмяГруппыМакросов.ИмяВложенногоМакроса
```

Для включения в макрос вложенного макроса нужно выбрать соответствующую макрокоманду из раскрывающегося списка в поле «Добавить новую макрокоманду» или перетащить «Вложенный макрос» в нужное место из раздела «Управление» каталога макрокоманд. В макросе отобразится блок (рис. 6.66), в котором по умолчанию вложенному макросу присвоено имя `Sub1`, предоставляется возможность добавления макрокоманд и вставлен признак конца вложенного макроса.

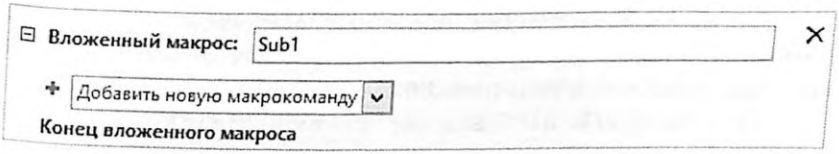


Рис. 6.66. Блок пустого вложенного макроса

Программы, формируемые в макросах, реализуют алгоритмы решения отдельных задач приложения. Механизм связывания макросов с событиями в объектах позволяет объединить разрозненные задачи приложения в единый комплекс, управляемый пользователем. Пользователь, выполняя различные действия, прежде всего в формах, инициирует выполнение макросов, автоматизирующих решение связанных с действиями пользователя задач.

6.12.3. Конструирование макроса

Создание макросов осуществляется в диалоговом режиме и сводится к записи в окне *конструктора макроса* последовательности *макрокоманд*, для которых задаются *аргументы*. Каждому макросу присваивается имя. При выполнении макроса макрокоманды выполняются последовательно в порядке их расположения. При этом используются объекты или данные, указанные в аргументах макрокоманд. Для изменения порядка выполнения макрокоманд может быть использован логический блок управления Если.

Выполнение макросов инициируется простой операцией и может сводиться к его открытию, как это делается и для других объектов базы данных. Помимо этого, Access предоставляет возможность автоматически инициировать выполнение макроса при наступлении некоторого события. Для связи макроса с событием достаточно в окне свойств объекта или его элемента управления внести в строку этого события имя макроса или создать внедренный макрос. События, с которыми можно связать макрос, представлены в свойствах форм и отчетов и их элементов управления.

Типичными событиями в Microsoft Access являются, например, нажатие кнопки мыши, изменение данных, а также открытие/закрытие формы или отчета. Всего существует около 40 событий.

По функциональному назначению события можно разделить на следующие группы:

– *события данных* (Data Events) возникают при вводе, удалении или изменении данных в форме или элементе управления, а также при перемещении фокуса с одной записи на другую;

– *события клавиатуры* (`Keyboard Events`) возникают при вводе с клавиатуры, а также при передаче нажатий клавиш с помощью макрокоманды «КомандыКлавиатуры» (`Send Keys`) или инструкции `SendKeys`;

– *события ошибки и таймера* (`Error and Timing Events`) используются при обработке ошибок и синхронизации данных в формах;

– *события мыши* (`Mouse Events`) возникают при действиях с мышью, например при нажатии кнопки мыши или при удержании кнопки в нажатом положении;

– *события фильтра* (`Filter Events`) возникают при создании или применении фильтра в форме;

– *события печати* (`Print Events`) возникают при печати отчета или при его форматировании для печати;

– *события фокуса* (`Focus Events`) возникают, когда форма или элемент управления теряют/получают фокус, а также в момент, когда они становятся активными/неактивными;

– *события окна* (`Window Events`) возникают при открытии, изменении размеров или закрытии формы или отчета.

Создание изолированного макроса. Создание изолированного макроса, являющегося отдельным объектом базы данных, начинается с выполнения команды *вкладка «Создание» → группа «Макросы и код» → Макрос*. В результате выполнения команды открывается окно макроса и каталог макрокоманд (рис. 6.62).

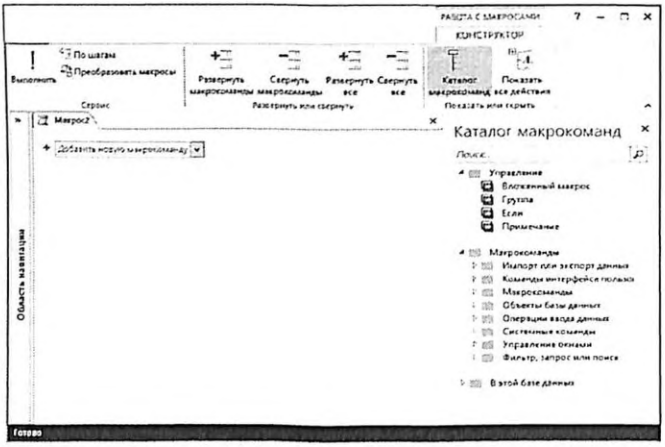


Рис. 6.62. Окно конструирования макроса с каталогом макрокоманд

Создание внедренного макроса. Для создания внедренного макроса нужно выполнить следующие действия:

1) выбрать в области навигации форму (отчет);

2) выбрать в контекстном меню формы (отчета) команду *Конструктор* или *Режим макета*;

3) открыть окно свойств;

4) выбрать элемент управления, раздел или форму (отчет) целиком;

5) на вкладке «События» выбрать событие, с которым нужно связать макрос;

6) нажать кнопку построителя выражений [...] – откроется окно диалога «Построитель» (рис. 6.63);

7) в диалоговом окне «Построитель» выделить пункт «Макросы»;

8) [OK] – откроется окно макроса, такое же, как при создании изолированного макроса.

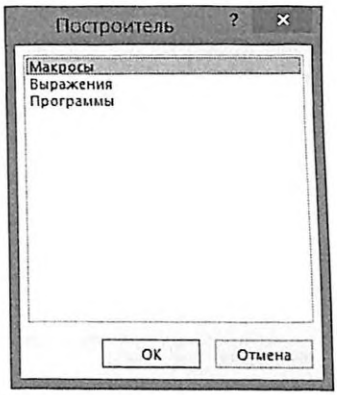


Рис. 6.63. Окно диалога «Построитель»

Внедренные макросы отличаются от изолированных макросов тем, что они хранятся в формах и отчетах. Они не отображаются в виде объектов в группе «Макросы» в области навигации. Это упрощает управление базой данных, поскольку не нужно следить за тем, какие макросы относятся к какой форме или отчету. Внедренные макросы сохраняются в составе формы или отчета и при их копировании, импорте или экспорте.

Примечание. Изолированный макрос, так же как внедренный, может быть связан с любым событием формы (отчета) или их элементов. Для этого нужно выбрать ранее созданный изолированный макрос из списка в строке свойства события.

6.12.4. Формирование макрокоманд в окне макроса

При создании нового макроса в его окне отображается поле «Добавить новую макрокоманду» с раскрывающимся списком (рис. 6.62). В списке представлен весь доступный набор макрокоманд. Этот набор можно изменять, нажимая кнопку «Показать все действия» в группе «Показать или скрыть» на контекстной вкладке «Работа с макросами | Конструктор». Если кнопка не нажата, то в набор не включаются так называемые *небезопасные макрокоманды*. К ним относятся макрокоманды, изменяющие базу данных или получающие доступ к ресурсам вне базы. Это позволяет пользователю при необходимости исключить использование небезопасных макрокоманд в своем приложении и, таким образом, предоставить возможность открывать базу данных с полным набором функций, даже если она не получила статус доверенной. Доверенной база данных становится, например, в случае, если она получена из источника, включенного в список надежных.

Для постоянного отображения в окне конструктора макросов списка всех доступных макрокоманд нужно нажать кнопку «Каталог макрокоманд». В окне каталога для простоты поиска макрокоманды разбиты на функциональные группы и обеспечена очень удобная возможность поиска макрокоманды по имени (рис. 6.62). Кроме того, в каталоге имеется раздел «В этой базе данных», где представлены не только все макросы из области навигации, но и макросы, внедренные в формы и отчеты. Содержимое этих макросов также может быть скопировано в другие конструируемые макросы.

Для ввода макрокоманды в поле можно нажать кнопку раскрытия списка макрокоманд и выбрать нужную. Можно ввести имя макрокоманды с клавиатуры, при этом система помогает сформировать его. Можно в каталоге выбрать макрокоманду и перетащить ее в окно макроса. Место размещения новой макрокоманды отмечается оранжевой полосой.

Окно конструирования внедренного макроса отличается от окна конструирования изолированного макроса только записью в заголовке окна. Для изолированного макроса там записано имя макроса, для внедренного – `имя формы (отчета): имя элемента: свойство события`.

Пример. `ТОВАР: ЦЕНА: После обновления`.

После ввода макрокоманды в макросе отображается блок, содержащий имя макрокоманды и строки ее аргументов (рис. 6.64). Значения аргументов задаются путем выбора их из списка, открывающегося в строке аргумента, с помощью построителя или вручную. Для выбранного аргумента или макрокоманды выводится всплывающая подсказка.

Знак минус слева от имени макрокоманды позволяет скрыть ее аргументы. Зеленые стрелки в правой части блока позволяют перемещать макрокоманду выше или ниже других макрокоманд. Здесь же имеется значок удаления макрокоманды.

Для ввода в макрос комментария используется блок «Примечание» размещенный в окне каталога макрокоманд в разделе «Управление». При перетаскивании этого блока в макрос создается пустой блок, в который и вводится нужный комментарий. После завершения ввода комментарий отображается строкой зеленого цвета, заключенной в знаки `/*` и `*/`.

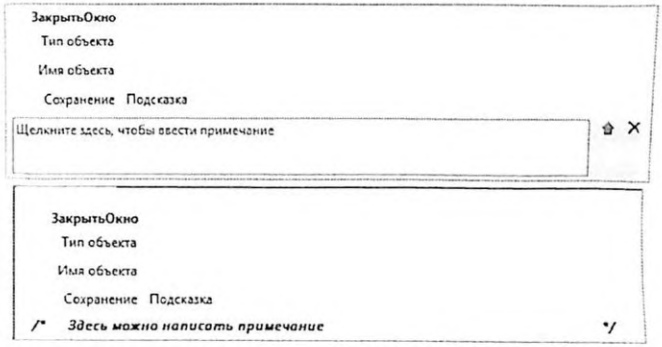


Рис. 6.64. Блок конструирования макрокоманды и ввода комментария

Каждая новая макрокоманда макроса добавляется в нужное место макроса, и проще всего это сделать перетаскиванием ее из каталога макрокоманд. Порядок размещения макрокоманд в бланке определяет последовательность их выполнения.

Примечание. Для просмотра и редактирования существующего изолированного макроса надо выбрать его в области навигации и в контекстном меню выбрать пункт «Конструктор».

Примечание. Для редактирования внедренного макроса нужно так же, как и при его создании, воспользоваться кнопкой построителя в строке свойства события.

6.12.5. Формирование макроса с помощью мыши

Создание некоторых макрокоманд в макросе может быть выполнено путем переноса с помощью мыши объекта базы данных из области навигации в окно макроса. Макрокоманда по умолчанию создается со значениями аргументов, соответствующими выбранному объекту.

Пример. При перетаскивании таблицы создается макрокоманда ОткрытьТаблицу, открывающая таблицу в режиме таблицы со значением Изменение для аргумента Режим данных.

Возможно перемещение любых объектов, представленных в области навигации базы данных. Соответственно перемещаемому объекту могут быть созданы макрокоманды ОткрытьТаблицу, ОткрытьФорму, ОткрытьЗапрос, ОткрытьОтчет, ОткрытьМодуль или ЗапускМакроса.

6.12.6. Использование в макросах ссылок на объекты

В макросах, как и в языке VBA, при обработке данных в базе необходимо уметь правильно ссылаться на эти данные. В Access определен ряд объектов, через которые предоставляется возможность получить доступ к данным базы. К таким объектам относятся формы и отчеты. Ссылки на формы, отчеты, их элементы управления и свойства формируются по определенным правилам. Построитель выражений позволяет сформировать такие ссылки простым выбором объекта, элемента управления и свойства в списках. Ко времени выполнения в макросах выражения со ссылкой на объект этот объект должен быть открыт.

6.12.7. Ссылки на объекты и их элементы управления

Ссылаться на объекты можно по имени, но нужно учесть, что в Access объекты объединяются в *семейства.* Формы объединены в семейство «Формы», отчеты – в семейство «Отчеты». Поэтому ссылка на объект включает имя семейства и – через восклицательный знак – имя объекта. Если имя включает пробелы или специальные символы, его надо брать в квадратные скобки.

Пример. Для ссылки на форму надо записать:
```
Forms![ДОГОВОРЫ С ПОКУПАТЕЛЯМИ].
```

В ссылке на элемент управления вслед за именем объекта через восклицательный знак записывается имя элемента управления, заключенное в квадратные скобки.

Пример. Ссылку на поле в форме надо записать так:
```
Forms![ДОГОВОРЫ С ПОКУПАТЕЛЯМИ]![СУММА_ДОГ]
```
Пример. Ссылку на элемент управления «Надпись» с именем Цена_Надпись надо записать так:
```
Forms![ДОГОВОРЫ С ПОКУПАТЕЛЯМИ]![Цена_Надпись]
```

6.12.8. Ссылки на свойство объекта

В ссылке на свойство объекта вслед за именем объекта через точку записывается имя свойства. Перечень свойств формы или отчета можно посмотреть, открыв объект в режиме конструктора или макета и вызвав окно его свойств. Многие свойства названы несколькими словами с пробелами между ними. Истинное имя свойства таких пробелов не имеет, поэтому в ссылках оно записывается без них.

Пример. Имя свойства «Область выделения» должно записываться так: `ОбластьВыделения`, имя свойства «Полосы прокрутки» – `ПолосыПрокрутки`. Ссылку на это свойство формы надо записать так:

```
Forms! [ДОГОВОРЫ С ПОКУПАТЕЛЯМИ].ScrollBars
```

В русифицированной версии Access в строках окна свойств указаны русские имена свойств. В ссылках следует использовать английский вариант имени свойства.

6.12.9. Ссылки на свойство элемента управления

Для записи ссылки на свойство элемента управления нужно дополнить ссылку на элемент управления через точку именем свойства. Различные типы элементов управления имеют разные свойства.

Пример. Ссылку на свойство `ВыводНаЭкран` (английский вариант имени `Visible`) элемента управления с именем `Цена_Надпись`, которое соответствует строке «Вывод на экран» на вкладке «Макет» (рис. 6.65), надо записать так:

```
Forms![ДОГОВОРЫ С ПОКУПАТЕЛЯМИ]![Цена_Надпись].Visible
```

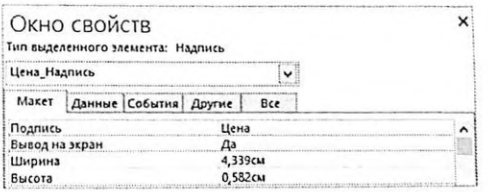


Рис. 6.65. Отображение свойства `Visible` в окне свойств

В общем виде правило записи ссылок может быть сформулировано следующим образом: оператор ! (восклицательный знак) указывает, что следующий за ним элемент является элементом, определяемым пользователем.

Пример. Оператор ! ставится перед ссылкой на открытую форму, на открытый отчет или элемент управления в открытой форме или отчете.

Оператор .(точка) обычно указывает, что следующий за ним элемент определен в Access.

Пример. Оператор . (точка) ставится перед ссылкой на свойства форм, отчетов и элементов управления.

6.12.10. Ссылка на подчиненную форму или отчет

Подчиненная форма (отчет) рассматриваются в Access как элемент управления формы (отчета). Форма после внедрения содержится в элементе управления «Подчиненная форма», отчет – в элементе управления «Подчиненный отчет». Поэтому ссылка на подчиненную форму (отчет) записывается как ссылка на элемент управления формы (отчета).

Элемент управления «Подчиненная форма/отчет» имеет специальное свойство «Форма» или, соответственно, «Отчет». Это свойство позволяет ссылаться на элементы управления подчиненных объектов и их свойства. Полная ссылка на свойство элемента управления в подчиненной форме имеет в общем виде следующую структуру:

```
Forms![Имя_формы]! [Элемент_Подчиненная_форма].Form![Элемент_подчиненной
                                    формы].Имя_свойства
```

Заметим, что при ссылках на элемент управления в подчиненной форме или подчиненном отчете не обязательно указывать свойство `Form` или `Report`.

Пример. Здесь приведена ссылка на элемент управления – поле СУММА_ПОСТ в подчиненной форме ПЛАН ПОСТАВОК, встроенной в форму ДОГОВОРЫ С ПОКУПАТЕЛЯМИ:

```
Forms![ДОГОВОРЫ С ПОКУПАТЕЛЯМИ]![ПЛАН ПОСТАВОК]![СУММА_ПОСТ]
```

Если макрос вызывается при наступлении некоторого события, например, при обновлении пользователем полей КОЛ_ПОСТ или ЦЕНА в форме ПЛАН ПОСТАВОК, то при передаче управления в макрос текущим объектом остается эта форма. Поэтому в макросе для ссылок достаточно использовать только имена полей.

6.12.11. Создание ссылок в построителе выражений

Ссылки на объекты, элементы управления и свойства удобно создавать с помощью построителя выражений.

Чтобы вызвать построитель выражений, надо выбрать в окне макроса строку аргумента макрокоманды, в которую требуется ввести выражение, и нажать появившуюся кнопку построителя  .

В открывшемся окне «Построитель выражений» в поле записи выражений надо сформировать выражение. Для этого:

1) в поле «Элементы выражений» раскройте двойным щелчком кнопки мыши папку, содержащую объекты;

2) выберите папку необходимого объекта;

3) в поле «Категории выражений» выберите нужный элемент управления объекта;

4) в поле «Значения выражений» выберите «Значение», если формируется ссылка на элемент управления, или нужное свойство;

5) двойным щелчком вставьте ссылку в выражение;

6) OK.

Access вставит созданное выражение в ту строку, из которой был вызван построитель выражений.

Если строка аргумента, из которой вызывается построитель выражений, уже содержит выражение, то оно автоматически отображается в поле выражений, где может быть отредактировано.

Заметим, если нужный объект или свойство не появляется в нижней части окна построителя выражений, это означает, что их нельзя использовать в том контексте, в котором был вызван построитель выражений.

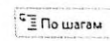
6.12.12. Сохранение, запуск и отладка созданного макроса

После ввода всех макрокоманд в макрос его надо сохранить, воспользовавшись инструментом «Сохранить» на панели быстрого доступа.

В режиме конструктора для выполнения макроса предназначен инструмент «Выполнить» из группы «Сервис» контекстной вкладки «Работа с макросами | конструктор». Если макрос уже закрыт, то для его выполнения надо в области навигации выбрать макро и в контекстном меню выбрать команду «Выполнить». Кроме того, для выполнения макроса можно использовать инструмент «Выполнить макрос» из группы «Макрос» вкладки «Работа с базами данных». Таким образом, реализованная макросом задача может решаться по инициативе пользователя.

Выполнение внедренного макроса происходит автоматически при наступлении *события*, с которым он связан.

Если при выполнении макроса не получен требуемый результат, макрос необходимо отладить. Для этого используется пошаговый режим с остановками после выполнения каждой макрокоманды,



что дает возможность проследить за результатом выполнения каждой отдельной макрокоманды и локализовать ошибку. Для перехода в этот режим выполняется команда *контекстная вкладка «Работа с макросами | Конструктор» → группа «Сервис» → По шагам.*

Словарь терминов

Внедренный макрос – всегда связывается с событием и сохраняется в форме или отчете.

Выражение – комбинация значений и операторов, дающая определенный результат.

Идентификатор – ссылка на значение поля, элемента управления или свойства.

Изолированный макрос – может выполняться в ответ на многочисленные виды событий, возникающих в формах, отчетах и их элементах управления.

Константа – не изменяющееся значение.

Конструктор – предоставляет пользователю ряд инструментальных средств, с помощью которых можно создавать и модифицировать объекты БД.

Лента Microsoft Access 2013 – это область в верхней части окна приложения, основной командный интерфейс.

Литерал – фактическое значение в виде числа, текстовой строки, даты.

Макет элементов управления – это объединение элементов управления в группу, для которой произведено выравнивание по вертикали и горизонтали для единообразного оформления формы.

Макрос – набор из одной или нескольких макрокоманд, каждая из которых выполняет определенное действие.

Многотабличная форма – средство для работы с данными нескольких взаимосвязанных таблиц.

Многотабличный отчет – создается на основе информации, содержащейся в нескольких объектах БД.

Несвязанный элемент управления – элемент управления, для которого источника данных не существует.

Область ввода – окно, используемое для удобства набора выражения.

Область навигации – это область в левой части окна Microsoft Access 2013, предназначенная для работы с объектами базы данных. Область навигации позволяет упорядочить объекты базы данных и является основным средством открытия или изменения объектов базы данных.

Оператор – операция над значениями.

Панель быстрого доступа – обеспечивает доступ к командам одним щелчком мыши.

Построитель выражений – средство автоматизации создания выражений.

Представление Backstage – содержит команды и сведения, применимые ко всей базе данных.

Процедура – совокупность операторов языка VBA, реализующая ряд логических шагов для выполнения конкретного действия.

Связанный элемент управления – присоединен к полю базовой таблицы или запроса.

Составная форма – многотабличная форма, состоящая из главной формы и одной или нескольких подчиненных форм.

Составной отчет – многотабличный отчет, который состоит из главного отчета и включаемого в него подчиненного отчета.

Схема данных – удобное и наглядное средство отображения логических связей таблиц БД.

Функция – небольшая программа, которая всегда возвращает значение (число или строку символов), являющееся результатом расчетов или выполнения других операций.

Вопросы и задания для самоконтроля

1. Назовите основные компоненты пользовательского интерфейса Microsoft Access 2013.
2. Охарактеризуйте компонент интерфейса Microsoft Access 2013 «Лента».
3. Охарактеризуйте компонент интерфейса Microsoft Access 2013 «Представление Backstage».
4. Охарактеризуйте компонент интерфейса Microsoft Access 2013 «Область навигации».
5. Как организована справочная система Microsoft Access 2013?
6. Данные каких типов можно хранить в таблицах Microsoft Access 2013?
7. Назовите и охарактеризуйте основные свойства полей.
8. Что собой представляет выражение? Что используется в качестве значений в выражениях?
9. Назовите и охарактеризуйте инструментальные средства создания объектов БД в Microsoft Access 2013.
10. Как создать новую (пустую) базу данных?
11. Назовите и охарактеризуйте способы создания таблиц в Microsoft Access 2013.
12. Как ведется работа с макетом таблицы?
13. Как создается схема базы данных?
14. Назовите и охарактеризуйте типы запросов, которые можно создавать в Microsoft Access 2013?
15. Какие существуют средства создания запросов?

16. Назовите и охарактеризуйте разновидности запросов выбора. Как они создаются?

17. Как создать перекрестный запрос?

18. Назовите и охарактеризуйте разновидности запросов действия. Как они создаются?

19. Как выполнить и сохранить запрос?

20. Для чего используются формы и из каких элементов они состоят?

21. Что определяют свойства формы?

22. Назовите и охарактеризуйте способы создания форм.

23. Назовите и охарактеризуйте режимы представления форм.

24. Охарактеризуйте однотабличные и многотабличные формы.

25. Как ведется редактирование формы в режиме макета.

26. Что собой представляет макет элементов управления?

27. Для чего используются отчеты?

28. Назовите и охарактеризуйте способы создания отчетов.

29. Охарактеризуйте однотабличные и многотабличные отчеты.

30. Назовите и охарактеризуйте инструментальные средства конструкторов форм и отчетов.

31. Как ведется работа с элементами управления в формах и отчетах?

32. Как создать диаграмму в форме (отчете)?

33. Для чего необходимы макросы и модули?

34. Что такое «процедура»?

35. Что такое «макрокоманда»? Проведите классификацию макрокоманд.

36. Назовите и охарактеризуйте различные виды макросов.

37. Назовите и охарактеризуйте события в Microsoft Access.

38. Как ведется конструирование макросов?

39. Как формируются макрокоманды в окне макроса?

40. Как в макросе задать ссылку: на объект, на свойство объекта, на свойство элемента управления, на подчиненную форму или отчет?

41. Как создать вложенный макрос?

42. Как созданный макрос запустить на выполнение?

43. Для чего используется пошаговый режим выполнения макроса?

7. Введение в язык SQL

7.1. Общая характеристика языка SQL

SQL (Structured Query Language) представляет собой непроцедурный (декларативный) язык, используемый для управления данными в реляционных СУБД. Термин «непроцедурный» означает, что на данном языке можно сформулировать, что нужно сделать с данными, но не требуется указывать, как именно это следует сделать. В этом языке отсутствуют алгоритмические конструкции, такие как метки, условные переходы, операторы цикла и др. Таким образом, язык SQL скрывает от пользователя сложности алгоритмов поиска данных и их реализации, обеспечивая простоту понимания и легкость манипулирования данными.

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language– «структурированный английский язык запросов». Позже по юридическим соображениям язык SEQUEL был переименован в SQL. Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования.

Стоит отметить, что SEQUEL был не единственным языком подобного назначения. В Калифорнийском Университете Беркли была разработана некоммерческая СУБД Ingres (предшественница популярной сейчас некоммерческой СУБД PostgreSQL), которая являлась реляционной СУБД, но использовала свой собственный язык QUEL, который, однако, не выдержал конкуренции по количеству поддерживающих его СУБД с языком SQL.

Первыми СУБД, поддерживающими новый язык, стали в 1979 году Oracle V2 для машин VAX от компании Relational Software Inc. (впоследствии ставшей компанией Oracle) и System/38 от IBM, основанная на System/R.

Название языка SQL (*Structured Query Language – структурированный язык запросов*) только частично отражает его суть. Язык SQL главным образом ориентирован на удобную и понятную пользователям формулировку запросов выбора из БД, однако его функции шире – с самого начала этот язык задумывался как полный язык БД. Под этим понимается, что (по крайней мере, теоретически) для выполнения любых осмысленных действий с базой данных, управляемой SQL-ориентированной СУБД, достаточно знать язык SQL.

Первоначально официального стандарта языка в области БД не существовало, что в определенном смысле сдерживало развитие технологий баз данных. Первый официальный стандарт языка SQL был принят *ANSI (American National Standards)* в 1986 году и *ISO (International Standards Organisation)* в 1987 году (так называемый SQL-86 или SQL1) и несколько уточнен в 1989 году. Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 году нового расширенного стандарта (ANSI SQL-92 или просто SQL2). Следующим стандартом стал SQL:1999 (SQL3). В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесенными позже (SQL:2008). Каждый стандарт представляет собой многостраничный документ сложной структуры. Например, базовая часть стандарта SQL:2003 состоит из более чем 1300 страниц текста.

Таким образом, SQL-стандарт следует рассматривать не как статический, а как постоянно развивающийся объект, что объясняется современным интенсивным развитием технологий баз данных. Функция SQL-стандартов – стать путеводителем на сложном пути для будущих согласованных совершенствований технологий обработки данных.

Язык SQL является первым и пока единственным стандартным языком работы с базами данных, который получил достаточно широкое распространение. В SQL сделаны большие инвестиции как со стороны разработчиков, так и со стороны пользователей. В настоящее время SQL реализован практически во всех коммерческих СУБД, все фирмы провозглашают соответствие своей реализации стандарту SQL. Таким образом, для грамотного использования любой SQL-ориентированной СУБД знание стандартов языка SQL является необходимым. SQL является логичным и достаточно мощным языком для взаимодействия с БД и в тоже время относительно легким для изучения. Освоив один раз стандарт этого языка, пользователь SQL может работать с БД в среде любой SQL-ориентированной СУБД.

7.2. Функциональные возможности языка SQL

С помощью языка SQL можно выполнять следующие действия:

– выполнение операций над таблицами – создание, удаление, изменение структуры;

– выполнение операций над данными таблиц – выборка, изменение, добавление и удаление;

– управление доступом – с помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа;

– совместное использование данных – SQL координирует совместное использование данных пользователями, работающими одновременно;

– целостность данных – SQL позволяет защитить БД от разрушения из-за несогласованных изменений или отказа компьютерной системы.

Язык SQL можно использовать для доступа к БД в двух режимах:

– при интерактивной работе (командный режим);

– прикладных программах (программный режим).

С помощью SQL пользователь может в интерактивном режиме быстро получить ответы на любые, в том числе достаточно сложные запросы, тогда как для программной реализации этих запросов на другом языке пришлось бы разрабатывать соответствующую программу. При написании прикладных программ также используют язык SQL для обращения к БД (встроенный SQL).

7.3. Достоинства и недостатки SQL

7.3.1. Достоинства SQL

Независимость от конкретной СУБД. Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своем тексты SQL-запросов, содержащие операторы определения данных и операторы манипулирования данными, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД (например: система электронного документооборота *Documentum* может работать как с *Oracle*, так и с *Microsoft SQL Server* и *IBM DB2*). Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно.

Наличие стандартов. «Стабилизации» языка способствует наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту.

Декларативность. С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Тем не менее, не стоит думать, что это полностью универсальный принцип– программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

7.3.2. Недостатки SQL

Несоответствие реляционной модели данных. Создатели реляционной модели данных Эдгар Кодд, Кристофер Дейт и их сторонники указывают на то, что SQL не является истинно реляционным языком. В частности, они отмечают следующие проблемы SQL:

- повторяющиеся записи;
- неопределенные значения (nulls);
- явное указание порядка полей слева направо;
- поля без имени и дублирующиеся имена полей;
- высокая избыточность и др.

Сложность. Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста.

Отступления от стандартов. Несмотря на наличие международного стандарта ANSI SQL-92, многие компании, занимающиеся разработкой СУБД (например, Oracle, Sybase, Microsoft, MySQL AB), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД, тем самым отступая от стандарта. Таким образом, появляются специфичные для каждой конкретной СУБД диалекты языка SQL.

Сложность работы с иерархическими структурами. Ранее диалекты SQL большинства СУБД не предлагали способа манипуляции древовидными структурами. Некоторые поставщики СУБД предлагали свои решения (например, Oracle использует выражение `CONNECT BY`). В настоящее время в ANSI стандартизована рекурсивная конструкция `WITH` из диалекта SQL DB2. В MS SQL Server рекурсивные запросы появились лишь в версии MS SQL Server 2005. В версии MS SQL Server 2008 появился новый тип данных – *hierarchyid*, упрощающий манипуляцию древовидными структурами.

7.4. Операторы SQL

Язык SQL представляет собой совокупность операторов, которые делятся на следующие группы:

- операторы определения данных (*Data Definition Language, DDL*);
- операторы манипуляции данными (*Data Manipulation Language, DML*);
- операторы определения доступа к данным (*Data Control Language, DCL*);
- операторы управления транзакциями (*Transaction Control Language, TCL*).

Опишем минимальное множество операторов SQL (табл. 7.1).

Таблица 7.1

Основные операторы SQL

Оператор	Назначение
CREATE TABLE	Создает структуру таблицы
DROP TABLE	Удаляет таблицу
INSERT	Добавляет новые записи в таблицу
DELETE	Удаляет записи из таблицы
UPDATE	Обновляет (изменяет) данные таблицы
SELECT	Извлекает данные из БД
CREATE VIEW	Создает представление*
GRANT	Предоставляет пользователю право доступа
REVOKE	Отменяет право доступа
COMMIT	Завершает текущую транзакцию
ROLLBACK	Отменяет текущую транзакцию
DECLARE	Определяет курсор* для запроса
OPEN	Открывает курсор для чтения результата запроса
FETCH	Считывает курсор из результатов запроса
CLOSE	Закрывает курсор

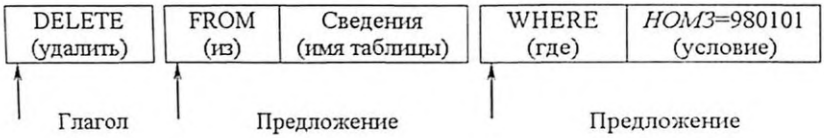
Каждая команда SQL начинается с *глагола* – ключевого слова, описывающего действие, выполняемое командой. Например, CREATE (создать), INSERT (добавить), COMMIT (завершить). После глагола идет одно или несколько *предложений*. Предложение описывает данные, с которыми работает команда, или содержит уточняющую информацию о действии, выполняемом командой. Каждое предложение начинается с *ключевого слова*. Например, WHERE (где), FROM (откуда), INTO (куда), HAVING (имеющий). Одни предложения являются обязательными, а другие – нет. Многие предложения содержат имена таблиц или полей БД; некоторые из них – дополнительные ключевые слова, константы и выражения.

---

* Назначение терминов, помеченных «звездочкой» будет рассмотрено ниже при описании соответствующих возможностей SQL.

Ключевые слова SQL нельзя использовать для идентификации таблиц, полей и пользователей. Имена должны содержать от 1 до 18 символов, начинаться с буквы и не содержать пробелы и специальные символы пунктуации.

Приведем пример команды SQL:



7.5. Данные и выражения в SQL

В языке SQL для представления информации в реляционной БД предусмотрены типы данных, перечисленные в таблице 7.2.

Таблица 7.2

Типы данных языка SQL

Тип данных	Описание
CHAR (длина) CHARACTER (длина)	Строки символов постоянной длины
INTEGER INT	Целые числа
SMALLINT	Малые целые числа
NUMERIC (точность, степень) DECIMAL (точность, степень) DEC (точность, степень)	Числа с фиксированной запятой
FLOAT (точность)	Числа с плавающей запятой
REAL	Числа с плавающей запятой низкой точности
DOUBLE PRECISION	Числа с плавающей запятой высокой точности

Допустимые значения для указанных данных специфичны в каждой СУБД. В некоторых командах SQL необходимо конкретно указывать значения данных посредством констант. Константы с фиксированной запятой представляются в виде обычных десятичных чисел (знак плюс (+) не указывается). Например, `21`, `-375.18`, `62.3`. Константы с плавающей запятой имеют такой же формат, как и в большинстве языков программирования.

Например, `1.5E7, 2.5E-6, -0.783E24` (символ E читается как *«умножить на десять в степени»*). Строковые константы должны быть заключены в кавычки. Например, «Минск», «New York». SQL поддерживает обработку отсутствующих данных с помощью понятия *отсутствующее значение* (`NULL`).

Приведем примеры отсутствующих значений:

Ф.И.О.	Код менеджера	План по сбыту, млн. р.
Петров П. П.	106	NULL[2]
Сидоров С. С.	NULL[1]	127

[1]Сидоров С.С. не менеджер.

[2]План по сбыту для Петрова П.П. не определен.

Выражения в SQL используются для выполнения операций над значениями, которые считаны из БД или используются для поиска в БД. Они представляют собой определенную последовательность полей, констант, функций, соединенных операторами. В них можно использовать следующие операторы:

1) *арифметические*: + (сложение), – (вычитание), * (умножение), / (деление);

2) *реляционные*: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно);

3) *логические*: `AND` (логическое «И»), `OR` (логическое «ИЛИ»), `NOT` («НЕТ»);

4) *специальные*:

- `IN` – определяет множество, которому может принадлежать значение;
- `BETWEEN` – задает границы, в которые должно попадать значение;
- `LIKE` – применяется для поиска по шаблону. В шаблоне используются специальные символы: % (процент) (ANSI-92), * (звездочка) (ANSI-89) – заменяет любую последовательность символов; _(подчеркивание) (ANSI-92), ? (вопросительный знак) (ANSI-89) – заменяет один любой символ;
- `IS NULL` – используется для поиска отсутствующих значений.

Для формирования сложных выражений можно применять скобки. Например, *Цена*(Остаток_на_начало + Приход - Расход)*.

7.6. Формирование запросов на языке SQL

Формирование запросов на языке SQL предполагает ввод команд с клавиатуры в строгом соответствии с их форматами. Для описания форматов команд (иногда усеченных) будем использовать следующие соглашения:

– угловые скобки (< >) – то, что в них указано, определяет пользователь;

– квадратные скобки ([ ]) – выделяют те части команды, которые могут отсутствовать;

– фигурные скобки ({ }) – заключенная в эти скобки часть команды может повторяться несколько раз;

– круглые скобки ( ) – в них заключаются аргументы команды;

– вертикальная черта (|) – означает альтернативный выбор.

Запись команд SQL, реализующих запросы к БД, с целью их лучшего восприятия будем приводить, используя прописные и строчные буквы и их различные начертания. Но для ускорения ввода команд целесообразно набирать их на одном регистре (например, полностью прописными буквами).

7.6.1. Создание таблицы. Обновление данных. Удаление таблицы

Запрос на описание полей таблицы. Запрос на описание полей таблицы, т. е. на создание ее структуры, на языке SQL реализуется с помощью команды CREATE TABLE следующего формата:

```
CREATE TABLE <имя таблицы>
({<имя поля> <тип данных>[<размер>]
[<ограничения на поле>]
[<значение поля по умолчанию>]},...
[,<ограничения на таблицу>]);
```

Некоторые из возможных ограничений на поля: NOT NULL (поле, для которого указано это ограничение, не допускает значений NULL); UNIQUE (поле, для которого указано это ограничение, не допускает повторяющихся значений); PRIMARY KEY (поле, для которого это ограничение задано, должно выступать в роли первичного ключа); CHECK (позволяет определить условие, которому должны удовлетворять вводимые в поле значения).

Ограничения на таблицу применяются к группам, состоящим из нескольких полей, и задаются определенными ключевыми словами. Значение поля по умолчанию указывается при помощи ключевого слова DEFAULT.

Пример. Пусть в БД необходимо создать таблицу с именем «Сведения», поля которой описываются следующим образом:

Имена полей	НОМЗ (№ зачетки)	ФИО	ГОД	ПОЛ
Типы данных	Целые числа	Текст	Целые числа	Текст
Размеры полей	6	15	4	7
Типы данных на языке SQL	INT	CHAR(15)	INT	CHAR(7)

Запрос на создание структуры таблицы «Сведения»:

```
CREATE TABLE Сведения
(НОМЗ INT, ФИО CHAR(15), ГОД INT, ПОЛ CHAR(7));
```

Первичное заполнение таблицы и добавление к ней записей. Эта операция осуществляется с помощью команды INSERT следующего формата:

```
INSERT <имя таблицы>[{<имя поля>},...]
   VALUES ({<значение>},...);
```

Если имена полей не указаны, то значениями заполняются поля, состав и порядок которых был описан командой CREATE TABLE. Значения полей задаются константами (выражения не допустимы).

Пример. Запрос на заполнение полей таблицы «Сведения» конкретными значениями.

```
INSERT INTO Сведения
   VALUES (980101, 'ИВАНОВ И. И.', 1980,
                   'МУЖСКОЙ');
```

Запрос на изменение значений полей таблицы. Такой запрос реализуется с помощью команды UPDATE, имеющей формат:

```
UPDATE <имя таблицы>
   SET {<имя поля>=<выражение>},...
   [WHERE <условие>];
```

Условие задается выражением типа сравнения или логическим выражением. Если предложение WHERE отсутствует, то изменения выполняются во всех записях таблицы.

Пример. Запрос на изменение в таблице «Сведения» фамилии и инициалов студентки, у которой номер зачетной книжки 980201, на Кравцова И.И.

```
UPDATE Сведения
   SET ФИО = 'КРАВЦОВА И. И.'
   WHERE НОМЗ = 980201;
```

Запрос на удаление записей из таблицы. Такой запрос формируется с помощью команды DELETE следующего формата:

```
DELETE FROM <имя таблицы>
   [WHERE <условие>];
```

Если предложение WHERE отсутствует, то удаляются все записи таблицы.

Пример. Запрос на удаление из таблицы «Сведения» данных о студенте, у которого номер зачетной книжки 980201.

```
DELETE FROM Сведения
   WHERE НОМЗ = 980201
```

Запрос на удаление таблицы. Такой запрос формируется с помощью команды DROP TABLE следующего формата:

```
DROP TABLE <имя таблицы>
```

Пример. Запрос на удаление таблицы «Сведения».

```
DROP TABLE Сведения
```

7.6.2. Формирование запросов выбора

Для выбора информации из таблиц используется команда SELECT следующего формата:

```
SELECT [DISTINCT|ALL] {<выражение>},...|*
   FROM {<имя таблицы>},...
   [WHERE <условие>]
   [ORDER BY {<имя поля> [ASC|DESC]},...]
   [GROUP BY {<имя поля>},...]
   [HAVING <условие>];
```

Частным случаем выражения является поле. Команда SELECT позволяет выводить все поля (*) или некоторые по указанному списку из (FROM) заданной таблицы по некоторому условию (WHERE). Выводить можно все значения полей (ALL) или только неповторяющиеся (DISTINCT). При этом возможны сортировка записей (ORDER BY) по возрастанию (ASC) или убыванию (DESC), их группировка (GROUP BY) по некоторым полям, исключение определенных результирующих групп (HAVING).

Команда SELECT дает возможность проведения вычислений над полями, а также итоговых вычислений с использованием статистических функций SUM, AVG, COUNT, MAX, MIN. Она позволяет также создавать вложенные запросы.

Пример. Запрос на выбор из таблицы «Сведения» данных о студентах мужского пола:

```
SELECT *
   FROM Сведения
   WHERE ПОЛ= 'МУЖСКОЙ';
```

Пример. Запрос на выбор из таблицы «Сведения» данных о студентах, у которых год рождения больше или равен 1977 и меньше или равен 1980:

```
SELECT *
   FROM Сведения
   WHERE ГОД >= 1977 AND ГОД <= 1980;
```

Пример. Запрос на выбор из таблицы «Сведения» данных о студентах, фамилии которых начинаются с буквы «И»:

```
SELECT *
   FROM Сведения
   WHERE ФИО LIKE 'И*';
```

7.6.3. Представления и их создание

Представление (view) – это виртуальная таблица, которая выглядит как реальная. Представления позволяют ограничить объем данных, которые пользователь может просматривать и модифицировать. Т. о., с их помощью можно осуществлять контроль за доступом пользователей к данным. Для пользователя представление ничем не отличается от настоящей таблицы: к нему можно формулировать запросы, выполнять обновление, вставку, удаление данных и соединение с другими таблицами и представлениями. По одним и тем же таблицам можно построить несколько представлений. Представление создается с помощью команды CREATE VIEW. Эта команда не поддерживается СУБД Access.

7.6.4. Управление доступом

Каждый пользователь БД имеет определенные права (действия) по отношению к объектам БД. Права могут меняться с течением времени: старые

могут отменяться, новые – добавляться. Пользователь, создавший таблицу, является ее владельцем. Как владелец, пользователь имеет все права на таблицу и может назначить права для работы с ней другим пользователям. Язык SQL предусматривает права: SELECT (право читать таблицу); INSERT (право добавлять данные в таблицу); UPDATE (право изменять данные таблицы); DELETE (право удалять данные из таблицы); REFERENCES (право определять первичный ключ). Назначение прав осуществляется командой GRANT, а лишение прав – командой REVOKE. Эти команды не поддерживаются СУБД Access.

7.6.5. Управление транзакциями

Для обеспечения целостности данных используются средства обработки транзакций – команды COMMIT и ROLLBACK. Команда COMMIT сообщает об успешном окончании транзакции, а команда ROLLBACK – о неуспешном окончании транзакции и необходимости отмены всех изменений, внесенных в БД в результате ее выполнения. Эти команды не поддерживаются СУБД Access.

7.6.6. Встроенный SQL

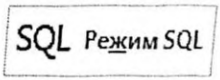
Язык SQL можно использовать при написании прикладных программ на определенных языках программирования исключительно для управления БД. По мере надобности в эти программы «встраиваются» команды SQL.

Для пересылки данных из БД в программу используются специальные команды SQL (операторы) – DECLARE, OPEN, FETCH, CLOSE, предназначенные для работы с курсором. В языке SQL *курсор* понимается нетрадиционно (как мигающий графический символ на экране дисплея). *Курсор SQL* – это переменная, связанная с запросом. Ее значениями являются строки результирующей таблицы запроса.

Оператор DECLARE описывает выполняемый запрос и связывает имя курсора с результатом запроса (определяет курсор для запроса). Оператор OPEN дает команду СУБД начать выполнение запроса и создавать таблицу результатов запроса (открывает курсор для чтения результата запроса). Оператор FETCH считывает данные запроса в переменную прикладной программы (считывает курсор из результатов запроса). Оператор CLOSE прекращает доступ к таблице результатов запроса и ликвидирует связь между курсором и этой таблицей.

7.7. Создание запроса на языке SQL в СУБД Access

Если пользователь желает работать с БД, используя язык SQL, то он должен открыть окно конструктора запроса и перейти в режим SQL по команде *контекстная вкладка «Работа с запросами | Конструктор» → группа «Результаты» → инструмент «Режим» → Режим SQL*. В появившемся окне можно формировать запрос на языке SQL и редактировать его, используя для этого привычную технологию редактирования в текстовом редакторе.



Выполнение и сохранение запроса на языке SQL осуществляется аналогично тому, как это делается для любых других запросов.

7.8. Диалекты языка SQL в СУБД

Несмотря на наличие международного стандарта ANSI SQL, многие компании, занимающиеся разработкой СУБД, вносят изменения в язык SQL, применяемый в разрабатываемой СУБД, тем самым отступая от стандарта. Каждая из реализаций языка SQL в конкретной СУБД называется *диалектом*. Функции, которые добавляются к стандарту языка разработчиками коммерческих реализаций, принято называть *расширениями*. Например, в стандарте языка SQL определены конкретные типы данных, которые могут храниться в базах данных. Во многих реализациях этот список расширяется за счет разнообразных дополнений.

Выделяют три уровня соответствия стандарту ANSI ISO – начальный, промежуточный и полный. В настоящее время не существует ни одного диалекта, полностью соответствующего стандарту. Производители СУБД (например, Oracle, Microsoft, Borland, Informix, Sybase) применяют собственные реализации SQL, отвечающие как минимум начальному уровню соответствия стандарту и содержащие некоторые расширения, специфические для данной СУБД. Не существует двух совершенно идентичных диалектов. Более того, поскольку разработчики баз данных вводят в системы все новые функциональные средства, они постоянно расширяют свои диалекты языка SQL, в результате чего отдельные диалекты все больше и больше отличаются друг от друга. Это имеет свои достоинства и недостатки.

Конкретная реализация языка может включать в себя более широкие возможности по сравнению со стандартом SQL, например, больше типов

данных, большее количество команд, больше дополнительных возможностей у имеющихся команд. Такие возможности делают работу с конкретной СУБД более эффективной. Кроме того, такие нестандартные возможности языка проходят практическую апробацию и со временем могут быть включены в стандарт.

Недостаток в том, что различия в синтаксисе реализаций SQL затрудняют перенос приложений из одной системы в другую. Например, если приложение было написано для базы данных MS SQL Server с использованием своего диалекта SQL – языка Transact-SQL, то при переносе системы в базу данных ORACLE не все конструкции языка будут понятны соответствующему диалекту SQL – языку PL/SQL.

В широко распространенных в настоящее время СУБД используются следующие диалекты языка SQL:

- PL/SQL – в СУБД Oracle;
- Transact-SQL – в СУБД Microsoft SQL;
- Informix-SQL – в СУБД Informix;
- Jet SQL – Microsoft Access.

Язык Jet SQL почти соответствует стандарту ANSI SQL. Основные различия языков Jet SQL и ANSI SQL состоят в следующем:

- они имеют разные наборы зарезервированных слов и типов данных;
- разные правила применимы к оператору `BETWEEN`, используемому для определения условий выборки записей;
- подстановочные знаки ANSI и Microsoft Jet, которые используются в операторе `LIKE`, различны;
- язык Jet SQL обычно предоставляет пользователю большую свободу, например, разрешается группировка и сортировка по значению выражения;
- язык Jet SQL позволяет использовать более сложные выражения.

Словарь терминов

SQL (Structured Query Language) – непроцедурный (декларативный) язык, используемый для управления данными в реляционных СУБД.

Диалект SQL – реализация языка SQL в конкретной СУБД.

Представление (view) – виртуальная таблица, которая выглядит, как реальная.

Расширения SQL – функции, которые добавляются к стандарту языка разработчиками коммерческих реализаций.

Вопросы и задания для самоконтроля

1. Что собой представляет язык SQL?
2. Какие существуют стандарты языка SQL?
3. Охарактеризуйте функциональные возможности языка SQL.
4. Каковы достоинства и недостатки языка SQL?
5. Какие группы операторов существуют в SQL?
6. Какова структура команды SQL?
7. Какие типы данных существуют в SQL?
8. Каково назначение и синтаксис команды `CREATE TABLE`?
9. Каково назначение и синтаксис команды `INSERT`?
10. Каково назначение и синтаксис команды `UPDATE`?
11. Каково назначение и синтаксис команды `DELETE`?
12. Каково назначение и синтаксис команды `DROP TABLE`?
13. Каково назначение и синтаксис команды `SELECT`?
14. С какой целью в SQL создаются представления?
15. Как в SQL осуществляется управление доступом?
16. Как в SQL осуществляется управление транзакциями?
17. Что собой представляет встроенный SQL?
18. Какова технология создания запроса на языке SQL в СУБД Microsoft Access?
19. Чем характеризуются диалекты языка SQL?

8. Системы обработки многопользовательских баз данных

8.1. Эволюция концепций обработки данных

Обработка данных со временем претерпела некоторую эволюцию. В развитии концепций обработки данных можно выделить следующие этапы:

1) обработка БД на мэйнфреймах с помощью СУБД;
2) обработка БД с помощью систем удаленной обработки данных;
3) обработка локальных БД на ПК с помощью настольных СУБД;
4) использование систем совместного использования (работа с централизованной базой данных с помощью сетевых версий настольных СУБД);
5) использование клиент-серверных систем;
6) использование систем обработки распределенных баз данных.

Классической архитектурой обработки многопользовательских БД является удаленная обработка.

Пользователи обрабатывают данные в пакетном режиме. Интерактивный режим доступа осуществляется с помощью терминалов, которые не обладают собственными вычислительными ресурсами. Программы управления коммуникациями (связью), прикладные программы, СУБД и ОС работают на едином центральном компьютере. Поскольку вся обработка производится единственным компьютером, то пользовательский интерфейс систем удаленной обработки обычно достаточно прост. Схема удаленной обработки показана на рисунке 8.1.

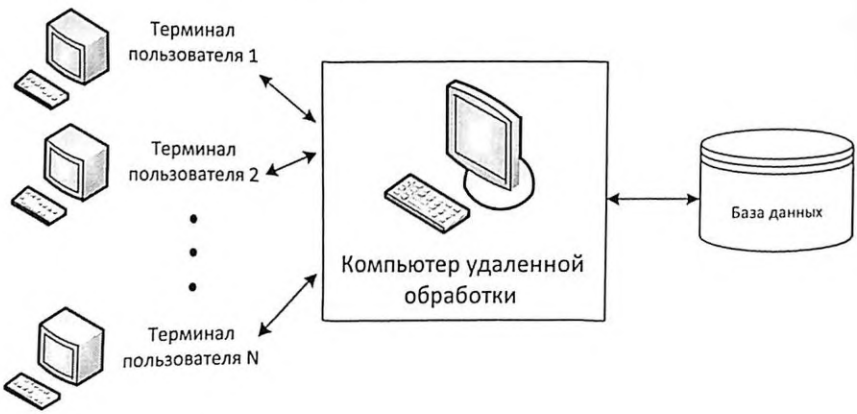


Рис. 8.1. Схема удаленной обработки многопользовательских БД

Пользователи (на рис. 8.1 показано N пользователей) работают с *терминалами*, которые передают данные и сообщения о транзакциях *центральному компьютеру* (компьютер удаленной обработки). Функции управления данными возложены на операционную систему. Часть ОС, отвечающая за управление связью, принимает сообщения и данные и передает их соответствующим прикладным программам. Программы обращаются к СУБД, а СУБД выполняет операции с БД, используя ту часть ОС, которая отвечает за обработку данных. Когда транзакция завершается, подсистема управления связью возвращает результаты пользователям, находящимся у терминалов. Поскольку их пользовательский интерфейс достаточно прост и имеет в основном текстовую ориентацию, все команды форматирования вывода генерируются процессором центрального компьютера и передаются по линии связи. Такие системы, подобные описанной, называются *системами удаленной обработки*, поскольку связь между входами и выходами осуществляется через находящийся на расстоянии центральный компьютер, ведущий обработку данных.

Преимуществом такой обработки является возможность коллективного использования ресурсов и оборудования, централизованное хранение данных, а недостатком – *отсутствие персонализации рабочей среды* (все программное обеспечение хранится централизованно и используется коллективно). Исторически системы удаленной обработки были наиболее распространенной альтернативой многопользовательским системам баз данных. Но по мере того, как персональные компьютеры стали появляться в офисах и выросла их мощь в качестве серверов данных, возникли новые архитектуры многопользовательских систем обработки данных.

8.2. Системы совместного использования файлов

8.2.1. Архитектура «файл-сервер» и роль настольных СУБД в ней

При наличии компьютерной сети открывается возможность хранить и использовать в многопользовательском режиме централизованные БД, размещаемые на одном компьютере – *сервере сети*. В этом случае каждый пользователь своего ПК получает доступ к общей для всех пользователей централизованной БД. Существуют различные концепции сетевой обработки данных.

Рассмотрим архитектуру информационной системы *с совместным использованием файлов*.

ИС с применением *файл-сервера* исторически появились первыми. Почти во всех ИС с совместным использованием файлов применяются локальные сети. В таких ИС БД хранится на *файловом сервере сети*. Файловый

сервер содержит файлы, необходимые для работы приложений и самой СУБД. Он обеспечивает функционирование той части сетевой версии СУБД, которая осуществляет управление данными в БД. Однако пользовательские приложения и сама сетевая СУБД размещены и функционируют на отдельных рабочих станциях.

Рассмотрим механизм функционирования ИС, построенной по архитектуре *файл-сервер* с использованием настольной СУБД.

Сетевые версии настольных СУБД отличаются от локальных версий тем, что они обладают некоторыми специальными механизмами, позволяющими многим пользователям совместно обращаться к общим ресурсам данных из централизованной базы данных. СУБД на каждой рабочей станции посылает запросы файловому серверу по всем необходимым ей данным, которые хранятся на диске файлового сервера. Все данные из БД пересылаются на компьютер пользователя, независимо от того, сколько реально их нужно для выполнения запроса. В результате на компьютере пользователя создается локальная копия БД (время от времени обновляемая из реальной БД на сервере). Затем СУБД пользователя выполняет запрос.

Схема работы с настольной СУБД в многопользовательском режиме показана на рисунке 8.2.

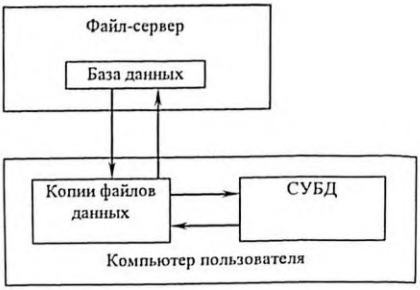


Рис. 8.2. Архитектура «файл-сервер»

Основные недостатки архитектуры «файл-сервер». Архитектура с использованием файлового сервера обладает следующими основными недостатками:

– поскольку файловый сервер не может обрабатывать SQL-запросы, то при совместном использовании файлов по локальной сети передаются большие объемы данных (полные копии БД перемещаются по сети с сервера на компьютер клиента);

– с увеличением объема хранимых данных и числа пользователей, снижается производительность настольных СУБД. Из-за этих проблем ИС с совместным использованием файлов редко используются для обработки больших объемов данных;

– при такой архитектуре вся тяжесть выполнения запроса к БД и управления целостностью БД ложится на СУБД пользователя;

– на каждой рабочей станции должна находиться сама сетевая версия настольной СУБД, что требует наличия больших объемов оперативной памяти на компьютере пользователя;

– доступ к одним и тем же файлам могут осуществлять сразу несколько пользователей, что усложняет управление целостностью, восстановлением БД на сервере;

– невысока безопасность БД, и трудно обеспечить конфиденциальность информации.

Достоинства и недостатки настольных СУБД. Достоинства настольных СУБД:

– они являются простыми для освоения и использования;

– обладают дружественным пользовательским интерфейсом;

– ориентированы на класс ПК, на самую широкую категорию пользователей – непрофессионалов;

– обеспечивают хорошее быстродействие при работе с небольшими БД.

Недостатки настольных СУБД:

– при росте объемов хранимых данных и увеличении числа пользователей снижается их производительность и могут возникать сбои при обработке данных;

– контроль за целостностью совершается внутри пользовательского приложения, что может вызывать нарушение целостности данных;

– очень малая эффективность работы в компьютерной сети.

Известно более десятка настольных СУБД. Наиболее популярными, исходя из числа проданных копий, признаются DBASE, Visual DBASE, Paradox, Microsoft FoxPro, Visual FoxPro, Access.

8.2.2. Клиент-серверные системы

В настоящее время наиболее перспективной является архитектура *клиент-сервер*.

В отличие от системы удаленной обработки, в которой имеется только один компьютер, клиент-серверная система состоит из множества компьютеров, объединенных в сеть.

Серверы и клиенты. *Сервером* определенного ресурса в компьютерной сети называется компьютер (программа), управляющий этим ресурсом, а *клиент* – это компьютер (программа), использующий этот ресурс.

В качестве ресурса компьютерной сети могут выступать, к примеру, базы данных, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Т. о., если управляемым ресурсом является база данных, то соответствующий сервер называется *сервером базы данных*.

Тип компьютеров, используемых в качестве клиентов, может быть разным: это могут быть большие ЭВМ или микрокомпьютеры. Однако, как правило, функции клиентов выполняют почти всегда ПК. В роли сервера может выступать компьютер любого типа, но по экономическим причинам функции сервера чаще всего также выполняют ПК, но имеющие более высокую производительность.

Клиентские приложения, серверы баз данных. На сервере сети размещается БД и устанавливается мощная серверная СУБД – *сервер баз данных*. *Сервер БД* – это программный компонент, обеспечивающий хранение больших объемов информации, ее обработку и представление пользователям в сетевом режиме.

На компьютере-клиенте приложение-клиент формирует запрос к БД в виде инструкций языка SQL. Серверная СУБД обеспечивает интерпретацию запроса, его выполнение, формирование результата запроса и пересылку его по сети на клиентский компьютер. Клиентское приложение интерпретирует его необходимым образом и представляет пользователю. Клиентское приложение может также посылать запрос на обновление БД и серверная СУБД внесет необходимые изменения в БД. Принцип функционирования архитектуры «клиент-сервер» приведен на рисунке 8.3.

В архитектуре «клиент-сервер» функции клиентского приложения и серверной СУБД разделены.

Функции клиентского приложения разбиваются на следующие группы:

– *ввод-вывод данных (презентационная логика)* – это часть кода клиентского приложения, которая определяет, что пользователь видит на экране, когда работает с приложением;

– *бизнес-логика* – это часть кода клиентского приложения, которая определяет алгоритм решения конкретных задач приложения;

– *обработка данных внутри приложения (логика базы данных)* – это часть кода клиентского приложения, которая связывает данные сервера с приложением. Для этой связи используется процедурный язык запросов

SQL, с помощью которого осуществляется выборка и модификация данных в серверных СУБД.

Сервер баз данных в общем случае осуществляет целый комплекс действий по управлению данными. Основными среди них являются следующие:

– выполнение пользовательских запросов на выбор и модификацию данных и метаданных, получаемых от клиентских приложений, функционирующих на ПК локальной сети;

– хранение и резервное копирование данных;

– поддержка ссылочной целостности данных согласно определенным в БД правилам;

– обеспечение авторизованного доступа к данным на основе проверки прав и привилегий пользователя;

– протоколирование операций и ведение журнала транзакций.

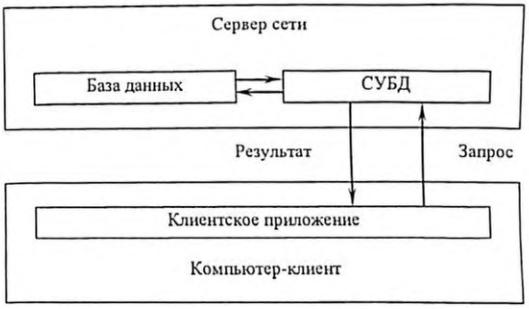


Рис. 8.3. Архитектура «клиент-сервер»

Общие сведения о хранимых процедурах и триггерах. В современной модели «клиент-сервер» бизнес-логика разделена между клиентом и сервером. На сервере бизнес-логика реализована в виде *хранимых процедур* – специальных программных модулей, которые хранятся в БД и управляются непосредственно СУБД.

Хранимая процедура – это специальная процедура, которая выполняется сервером баз данных. Хранимые процедуры пишутся на процедурном языке, который зависит от конкретной СУБД. Для написания хранимых процедур для MS SQL Server используется расширенный стандарт языка SQL – Transact-SQL. Хранимая процедура здесь – это последовательность операторов Transact-SQL, хранящихся в БД. Хранимые процедуры предварительно откомпилированы, поэтому их эффективность выше, чем обычных запросов. Они выполняются непосредственно на сервере.

Существует два вида хранимых процедур:

– системные;

– пользовательские.

Системные хранимые процедуры предназначены для получения информации из системных таблиц и выполнения различных служебных операций и особенно полезны при администрировании базы данных.

Пользовательские хранимые процедуры создаются непосредственно разработчиками или администраторами базы данных. Полезность хранимых процедур определяется в первую очередь высокой (по сравнению с обычными Transact-SQL запросами) скоростью их выполнения. Однако наибольший эффект достигается при выполнении многократно повторяющихся операций. Пользовательские хранимые процедуры применяются при решении практически любых задач. Пользователь может получить право выполнения хранимой процедуры, даже если он не имеет права доступа к объектам, к которым обращается программа.

Хранимая процедура вызывается явно, т. е. при непосредственном обращении к процедуре из клиентского приложения, работающего с базой данных. Хранимые процедуры используются для извлечения или изменения данных в любое время. Хранимые процедуры могут принимать аргументы при запуске и возвращать значения в виде результирующих наборов данных.

Логика БД реализуется с помощью триггеров. *Триггер* – это специальный тип хранимой процедуры, которая автоматически выполняется при каждой попытке изменить данные. Триггер всегда связан с конкретной таблицей и выполняется тогда, когда при редактировании этой таблицы наступает событие, с которым он связан (например, вставка, удаление или обновление записи). Каждая таблица может иметь произвольное количество триггеров любых типов. После операций вставки, обновления, удаления может быть запущен триггер, который в результате приведет к вычислению бизнес-правил или к выполнению определенных действий. При удалении таблицы, имеющей триггеры, все они также удаляются.

Триггеры обеспечивают целостность данных, предотвращая их несанкционированное или неправильное изменение. Триггеры не принимают параметров и не возвращают значений. Они выполняются неявно, то есть триггер запускается только при попытке изменения данных. Триггеры могут иметь несколько уровней вложенности (например, в СУБД MS SQL Server триггеры имеют до 32 уровней вложенности), то есть выполнение одного триггера инициирует выполнение другого триггера. Триггер является частью транзакции, следовательно, если триггер не выполнятся, то отменяется вся транзакция. И наоборот, если какая-то часть транзакции не выполнилась, то и триггер будет отменен.

Достоинства архитектуры «клиент-сервер». Этой архитектуре присущи следующие достоинства:

– при клиент-серверной обработке уменьшается сетевой трафик, так как через сеть передаются только результаты запросов;

– груз файловых операций ложится в основном на сервер, который мощнее компьютеров-клиентов и поэтому способен быстрее обслуживать запросы. Как следствие этого, уменьшается потребность клиентских приложений в оперативной памяти;

– поскольку серверы способны хранить большое количество данных, то на компьютерах-клиентах освобождается значительный объем дискового пространства для других приложений;

– повышается уровень непротиворечивости данных и существенно повышается степень безопасности БД, так как правила целостности данных определяются в серверной СУБД и являются едиными для всех приложений, использующих эту БД;

– имеется возможность хранения бизнес-правил (например, правил ссылочной целостности или ограничений на значения данных) на сервере, что позволяет избежать дублирования кода в различных клиентских приложениях, использующих общую базу данных.

8.2.3. Характеристика серверов баз данных

Современные серверные СУБД обладают следующими характеристиками:

– существуют в нескольких версиях для различных платформ, как правило, для различных коммерческих версий UNIX-Solaris, HP/UX. Многие производители также выпускают версии своих серверов баз данных для Windows, а также версии для Linux;

– в большинстве случаев поставляются с удобными административными утилитами;

– осуществляют резервное копирование и архивацию данных и журналов транзакций;

– поддерживают несколько сценариев репликации;

– позволяют осуществлять параллельную обработку данных в многопроцессорных системах. Серверы, допускающие параллельную обработку, разрешают нескольким процессорам обращаться к одной БД, что обеспечивает высокую скорость обработки транзакций;

– поддерживают создание хранилищ данных и OLAP. Хранилище данных – это совокупность данных, полученных прямо или косвенно из информационных систем, которые содержат текущую и деловую информацию, а также из некоторых внешних источников;

– выполняют распределенные запросы и транзакции;

– дают возможность использовать различные средства проектирования схем данных – универсальные или ориентированные на конкретную СУБД;

– имеют средства разработки клиентских приложений и генераторы отчетов;

– поддерживают публикацию баз данных в Интернет;

– обладают широкими возможностями управления пользовательскими привилегиями и правами доступа к различным объектам БД.

К современным серверам баз данных относятся Oracle (Oracle), MS SQL Server (MS), Informix (Informix), Sybase (Sybase), DB2 (IBM).

8.2.4. Механизмы доступа к базам данных

Все серверные СУБД имеют клиентскую часть, которая обращается к БД посредством СУБД. Между клиентским приложением и СУБД не существует прямой связи и дополнительно встраиваются программные модули, позволяющие клиентскому приложению получать доступ к БД, создаваемым с помощью разных СУБД. Такие модули называются *механизмами доступа к данным*.

Существует два основных способа доступа к данным из клиентских приложений:

– использование прикладного интерфейса;

– использование универсального программного интерфейса.

Прикладной программный интерфейс (API – Application Programming Interface) представляет собой набор функций, вызываемых из клиентского приложения. Он может работать только с СУБД данного производителя и при ее замене придется переписывать значительную часть кода клиентского приложения. Прикладной программный интерфейс различен для разных СУБД.

Универсальный механизм доступа к данным обеспечивает возможность использования одного и того же интерфейса для доступа к разным типам СУБД. Обычно он реализован в виде специальных дополнительных модулей, называемых *драйверами*.

Наиболее распространенным программным интерфейсом, обеспечивающим доступ к данным конкретной базы данных, является *ODBC (Open Database Connectivity)* фирмы Microsoft. В рамках ODBC программное приложение непосредственно взаимодействует с диспетчером драйвером, посылая ему ODBC-вызовы. Диспетчер драйверов отвечает за динамическую загрузку нужного ODBC-драйвера, через который обращается к серверу баз данных. ODBC-драйвер выполняет все вызовы ODBC-функций и «переводит» их на язык источника данных. СУБД хранит и выводит данные в ответ на запросы со стороны ODBC-драйвера.

Задание ODBC-источника данных является действием, которое осуществляется средствами операционной системы, управляющей компьютером. В операционной системе Windows в Панели управления предусмотрен пункт «Источники данных ODBC (32 разр)», из которого вызывается *Администратор источников данных ODBC*. С его помощью могут быть заданы:

– *пользовательский DSN* – источник данных, доступный только текущему пользователю на текущем компьютере;

– *файловый DSN* – источник данных, которые могут применять совместно различные пользователи, у которых установлены одинаковые ODBC-драйверы;

– *системный DSN* – источник данных, доступный всем пользователям и службам текущего компьютера.

8.3. Системы обработки распределенных баз данных

8.3.1. Понятие и архитектура распределенной БД

Распределенная БД (РаБД) – набор логически связанных между собой разделяемых данных и их описаний, которые физически распределены по нескольким компьютерам (узлам) в некоторой компьютерной сети.

Каждая таблица в РаБД может быть разделена на некоторое количество частей, называемых *фрагментами*. Фрагменты могут быть *горизонтальными*, *вертикальными* и *смешанными*. *Горизонтальные* фрагменты представляют собой подмножества строк, а *вертикальные* – подмножества столбцов. Фрагменты распределяются на одном или нескольких узлах.

С целью улучшения доступности данных и повышения производительности системы для отдельных фрагментов может быть организована *репликация* – поддержка актуальной копии некоторого фрагмента на нескольких различных узлах. *Репликаты* – множество различных физических копий некоторого объекта БД, для которых в соответствии с определенными в БД правилами поддерживается синхронизация с некоторой «главной копией».

Существуют несколько альтернативных стратегий размещения данных в системе:

– раздельное (фрагментированное) размещение;

– размещение с полной репликацией;

– размещение с выборочной репликацией.

Раздельное (фрагментированное) размещение. В этом случае БД разбивается на непересекающиеся фрагменты, каждый из которых размещается на одном из узлов системы. При отсутствии репликации стоимость хранения данных будет минимальна, но при этом будет невысок также уровень

надежности и доступности данных в системе. Отказ на любом из узлов вызовет утрату доступа только к той части данных, которая на нем хранилась.

Размещение с полной репликацией. Эта стратегия предусматривает размещение полной копии всей БД на каждом из узлов системы. Следовательно, надежность и доступность данных, а также уровень производительности системы будут максимальными. Однако стоимость хранения данных и уровень затрат на передачу данных в этом случае будут самыми высокими.

Размещение с выборочной репликацией. Данная стратегия представляет собой комбинацию методов фрагментации, репликации и централизации. Одни массивы данных разделяются на фрагменты, тогда как другие подвергаются репликации. Все остальные данные хранятся централизованно. Целью применения данного метода является объединение всех преимуществ, существующих в остальных моделях, с одновременным исключением свойственных им недостатков. Благодаря своей гибкости, именно эта стратегия используется чаще всего.

8.3.2. Распределенная СУБД

Работу с РаБД обеспечивают распределенные СУБД. *Распределенная СУБД* (РаСУБД) – комплекс программ, предназначенный для управления распределенной БД и позволяющий сделать распределенность информации «прозрачной» для конечного пользователя. Из определения РаСУБД следует, что для конечного пользователя должен быть полностью скрыт тот факт, что распределенная БД состоит из нескольких фрагментов, которые могут размещаться на нескольких компьютерах, расположенных в сети и к ней возможен параллельный доступ нескольких пользователей. Назначение обеспечения «прозрачности» состоит в том, чтобы распределенная система внешне вела себя точно так же, как и централизованная. Такое распределение данных позволяет, например, хранить в узле сети те данные, которые наиболее часто используются в этом узле. Такой подход облегчает и ускоряет работу с этими данными и оставляет возможность работать с остальными данными БД, хотя для доступа к ним требуется потратить некоторое время на передачу данных по сети.

Основная задача РаСУБД состоит в обеспечении средств интеграции локальных баз данных, располагающихся в некоторых узлах компьютерной сети, с тем, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим БД как к единой БД. Другими словами, для клиентских приложений РаБД представляется не набором баз, а единым целым. Каждый фрагмент БД сохраняется на одном или нескольких компьютерах, которые соединены между собой линиями связи и каждый из них работает под

управлением отдельной СУБД. Пользователи взаимодействуют с РаБД через приложения. Приложения могут быть классифицированы как те, которые не требуют доступа к данным на других узлах (*локальные приложения*), и те, которые требуют подобного доступа (*глобальные приложения*). В РаСУБД должно существовать хотя бы одно глобальное приложение, поэтому любая РаСУБД имеет следующие особенности:

- набор логически связанных разделяемых данных;
- сохраняемые данные разделены на некоторое количество фрагментов;
- между фрагментами может быть организована репликация данных;
- фрагменты и их реплики распределены по различным узлам;
- узлы связаны между собой сетевыми соединениями;
- работа с данными на каждом узле управляется локальной СУБД.

СУБД на каждом узле способна поддерживать автономную работу локальных приложений.

8.3.3. Гомогенные и гетерогенные распределенные БД

РаБД можно классифицировать на *гомогенные* и *гетерогенные*.

Гомогенной РаБД управляет один и тот же тип СУБД. *Гетерогенной* РаБД управляют различные типы СУБД, использующие разные модели данных – реляционные, сетевые, иерархические или объектно-ориентированные СУБД.

Гомогенные РаБД значительно проще проектировать и сопровождать. Кроме того, подобный подход позволяет поэтапно наращивать размеры РаБД, последовательно добавляя новые узлы к уже существующей РаБД. *Гетерогенные* РаБД обычно возникают в тех случаях, когда независимые узлы, управляемые своей собственной СУБД, интегрируются во вновь создаваемую РаБД.

8.3.4. Двенадцать правил К. Дейта для РаБД и РаСУБД

К. Дейтом были сформулированы двенадцать правил (1987) для типичной РаБД. Основой этих правил является то, что РаБД должна восприниматься пользователем точно так же, как и привычная централизованная БД.

1. *Локальная автономность*. В данном контексте автономность означает следующее:

- локальные данные принадлежат локальным владельцам и сопровождаются локально;
- все локальные процессы остаются чисто локальными;
- все процессы на заданном узле контролируются только этим узлом.

2. Отсутствие опоры на центральный узел. В системе не должно быть ни одного узла, без которого система не сможет функционировать, т. е. никакой конкретный сервис (управление транзакциями, оптимизация запросов и др.) не должен возлагаться на какой-либо специально выделенный центральный узел.

3. Непрерывное функционирование. В идеале в системе не должна возникать потребность в плановом останове ее функционирования.

4. Независимость от расположения. Пользователь должен получать доступ к базе данных с любого узла, причем получать доступ к любым данным, независимо от того, где они физически сохраняются.

5. Независимость от фрагментации. Пользователь должен получать доступ к данным независимо от способа их фрагментации.

6. Независимость от репликации. Пользователь не должен нуждаться в сведениях о наличии репликации данных, т. е. пользователь не будет иметь средств для получения прямого доступа к конкретной копии элемента данных, а также не должен заботиться об обновлении уже имеющейся копии.

7. Обработка распределенных запросов. Система должна поддерживать обработку запросов, ссылающиеся на данные, расположенные более чем на одном узле.

8. Обработка распределенных транзакций. Система должна поддерживать выполнение транзакций.

9. Независимость от типа оборудования. Система должна быть способна функционировать на оборудовании с различными вычислительными платформами.

10. Независимость от сетевой архитектуры. Система должна быть способна функционировать в сетях с различной архитектурой.

11. Независимость от операционной системы. Система должна быть способна функционировать под управлением различных операционных систем.

12. Независимость от типа СУБД.

8.3.5. Обработка распределенных запросов

В распределенной среде работа системы не должна демонстрировать никакого снижения производительности, связанного с его распределенной архитектурой, например, с присутствием медленных сетевых соединений. РаСУБД должна находить наиболее эффективные стратегии выполнения запросов. В распределенной среде обработчик распределенных запросов отображает запрос на доступ к данным в упорядоченную последовательность операций локальных баз данных (в отличие от централизованной, где

обработчик запросов оценивает каждый запрос на доступ к данным, а выполнение его представляет собой упорядоченную последовательность операций с БД). Дополнительная сложность возникает из-за необходимости учитывать наличие фрагментации, репликации и определенной схемы размещения данных. Обработчик распределенных запросов должен выяснить:

- к какому фрагменту следует обратиться;
- какую копию фрагмента использовать, если его данные реплицируются;
- какое из местоположений должно использоваться.

Возможности выполнения распределенного запроса поддерживаются сейчас почти всеми серверными СУБД (по крайней мере в том случае, когда в транзакцию вовлечены серверы от одного производителя). С этой целью используется механизм двухфазного завершения транзакций, когда на первом этапе серверы, вовлеченные в транзакцию, сигнализируют о готовности ее завершить, а на втором этапе происходит реальная фиксация изменений в БД.

8.3.6. Достоинства и недостатки РаСУБД

Системы с распределенными БД имеют дополнительные достоинства перед традиционными централизованными системами баз данных. Перечислим эти достоинства:

- отражение структуры организации;
- разделяемость и локальная автономность;
- повышение доступности данных;
- повышение надежности;
- повышение производительности;
- экономические выгоды;
- модульность системы.

Недостатки РаСУБД:

- *повышение сложности.* РаСУБД являются более сложными программными комплексами, чем централизованные СУБД, что обусловлено распределенной природой используемых ими данных, а также репликацией данных;
- *увеличение стоимости.* Увеличение сложности означает и увеличение затрат на приобретение и сопровождение РаСУБД;
- *проблемы защиты.* В централизованных системах доступ к данным легко контролируется. Однако в распределенных системах требуется организовать контроль доступа не только к данным, реплицируемым на несколько различных узлов, но и защиту сетевых соединений самих по себе;
- *усложнение контроля за целостностью данных.* В РаСУБД повышенная стоимость передачи и обработки данных может препятствовать организации эффективной защиты от нарушений целостности данных;

– *отсутствие стандартов.* Отсутствуют стандарты на каналы связи и протоколы доступа к данным, а также отсутствуют инструментальные средства и методологии, способные помочь пользователям в преобразовании централизованных систем в распределенные;

– *недостаток опыта.* Еще не накоплен необходимый опыт промышленной эксплуатации распределенных систем, сравнимый с опытом эксплуатации централизованных систем;

– *усложнение процедуры разработки БД.* Разработка распределенных БД, помимо обычных трудностей, связанных с процессом проектирования централизованных БД, требует принятия решения о фрагментации данных, распределении фрагментов по отдельным узлам и организации процедур репликации данных;

– *сложность управления* и обусловленная этим потенциальная опасность потери целостности данных.

8.3.7. Обзор распределенных СУБД

В настоящее время наиболее развитыми в теоретическом и практическом отношении являются реляционные распределенные СУБД. К наиболее изученным РаСУБД относятся:

– система SDD-1, созданная в конце 70-х – начале 80-х годов в научно-исследовательском отделении фирмы Computer Corporation of America;

– система R* , которая является распределенной версией системы System R и создана в начале 80-х годов фирмой IBM;

– система Distributed INGRES, которая является распределенной версией системы INGRES и создана в начале 80-х годов в Калифорнийском университете в Беркли.

В настоящее время в большинстве коммерческих реляционных серверных СУБД предусмотрены разные виды поддержки использования распределенных баз данных. Наиболее полно функции распределенной СУБД реализованы в системах:

– INGRES/STAR, разработанная отделением Ingres Division фирмы The ASK Group Inc.;

– ORACLE 7 фирмы ORACLE Corp.;

– модуле распределенной системы DB2 фирмы IBM.

Наиболее близко подошли к реализации функций распределенных такие СУБД, как:

– Informix On-line фирмы Informix Software;

– Sybase System 10 фирмы Sybase Inc.

Словарь терминов

Клиент определенного ресурса в компьютерной сети – это компьютер (программа), использующий этот ресурс.

Прикладной программный интерфейс – представляет собой набор функций, вызываемых из клиентского приложения.

Распределенная база данных – набор логически связанных между собой разделяемых данных и их описаний, которые физически распределены по нескольким узлам в некоторой компьютерной сети.

Распределенная СУБД – комплекс программ, предназначенный для управления распределенной БД и позволяющий сделать распределенность информации «прозрачной» для конечного пользователя.

Репликаты – множество различных физических копий некоторого объекта БД, для которых в соответствии с определенными в базе данных правилами поддерживается синхронизация с некоторой «главной копией».

Репликация – поддержка актуальной копии некоторого фрагмента распределенной базы данных на нескольких различных узлах.

Сервер базы данных – это программный компонент, обеспечивающий хранение больших объемов информации, ее обработку и представление ее пользователям в сетевом режиме.

Сервер определенного ресурса в компьютерной сети – компьютер (программа), управляющий этим ресурсом.

Триггер – специальный тип хранимой процедуры, которая автоматически выполняется при каждой попытке изменить данные.

Универсальный механизм доступа к данным – обеспечивает возможность использования одного и того же интерфейса для доступа к разным типам СУБД.

Файловый сервер – содержит файлы, необходимые для работы приложений и самой СУБД.

Хранимые процедуры – специальных программные модули, которые хранятся в БД и управляются непосредственно СУБД.

Вопросы и задания для самоконтроля

1. Назовите этапы развития концепций обработки данных?
2. Охарактеризуйте архитектуру «удаленная обработка».
3. Охарактеризуйте архитектуру с совместным использованием файлов (архитектура «файл-сервер»).
4. Охарактеризуйте архитектуру «клиент-сервер».
5. Что собой представляют хранимые процедуры? Каких видов они бывают?
6. Что собой представляют триггеры?

7. Каковы характеристики современных серверных СУБД?
8. Охарактеризуйте механизмы доступа к базам данных.
9. Что собой представляет распределенная база данных?
10. Какими способами осуществляется фрагментирование в распределенных базах данных?
11. Для чего выполняется репликация фрагментов?
12. Что собой представляет репликат?
13. Назовите и охарактеризуйте стратегии размещения данных в РаБД.
14. Что собой представляет распределенная СУБД?
15. Охарактеризуйте гомогенные РаБД.
16. Охарактеризуйте гетерогенные РаБД.
17. Приведите правила К. Дейта для РаБД и РаСУБД.
18. Как ведется обработка распределенных запросов?
19. Каковы достоинства и недостатки РаСУБД?

9. Администрирование баз данных

9.1. Пользователи базы данных, администратор базы данных, его функции

Пользователь БД (user) – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. На каждом этапе жизненного цикла базы данных с ней связаны разные категории пользователей.

9.1.1. Конечные пользователи

Конечные пользователи – это основная категория пользователей, в интересах которых создается БД. В зависимости от особенностей создаваемой БД круг конечных пользователей может различаться. Это могут быть случайные пользователи, которые обращаются за информацией к БД время от времени и постоянные пользователи. В качестве случайных пользователей могут рассматриваться, например, клиенты фирмы, просматривающие каталог продукции или услуг. Постоянными пользователями могут быть сотрудники, которые работают со специально разработанными для них программами, которые обеспечивают автоматизацию их деятельности при выполнении служебных обязанностей.

9.1.2. Администратор базы данных

Администратор базы данных (АБД) – это физическое лицо или группа лиц, отвечающих за выработку требований к базе данных, ее проектирование, создание, эффективное использование и сопровождение. В процессе эксплуатации АБД следит за функционированием информационной системы, обеспечивает защиту от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой в базе данных информации. Для однопользовательских информационных систем функции АБД обычно возлагаются на лиц, непосредственно работающих с приложением БД.

В вычислительной сети АБД взаимодействует с администратором сети. В его обязанности входит контроль за функционированием аппаратно-программных средств, реконфигурация сети, восстановление программного обеспечения после сбоев и отказов оборудования, профилактические мероприятия и обеспечение разграничения доступа.

9.1.3. Разработчики и администраторы приложений

Разработчики и администраторы приложений – это группа пользователей, которая функционирует во время проектирования, создания и реорганизации БД. Администраторы приложений координируют деятельность разработчиков при разработке конкретного приложения или группы приложений, объединенных в функциональную подсистему.

Не в каждой БД могут быть выделены все типы пользователей. При разработке информационных систем с использованием настольных СУБД администратор БД, администратор приложений и разработчик часто существовали в одном лице. Однако при построении современных сложных корпоративных баз данных, которые используются для автоматизации бизнес-процессов в крупной фирме или корпорации, могут существовать и группы администраторов приложений, и отделы разработчиков. Наиболее сложные обязанности возложены на группу администратора БД.

База данных взаимодействует в соответствующей среде со множеством пользователей. Пользователи могут предъявлять противоречивые требования к базе данных. Следовательно, возникает проблема координации деятельности пользователей и управления целостностью данных и защитой БД. Необходимость решения этой проблемы вызвала необходимость *администрирования* базы данных.

К основным функциям группы администратора БД относят:

1) *анализ предметной области*: описание предметной области, выявление ограничений целостности, определение статуса (доступности, секретности) данных, определение потребностей пользователей;

2) *проектирование структуры БД*: описание информационного содержания и внутренней структуры БД;

3) *задание ограничений целостности при описании структуры БД*:
   - определение ограничений целостности, вызванных структурой БД,
   - разработка процедур обеспечения целостности БД при вводе и коррекции данных,
   - определение ограничений целостности при параллельной работе пользователей в многопользовательском режиме;

4) *первоначальная загрузка и ведение БД*;

5) *защита данных*:
   - определение системы паролей, принципов регистрации пользователей, создание групп пользователей, обладающих одинаковыми правами доступа к данным,

- тестирование системы защиты,
- исследование случаев нарушения системы защиты,
- разработка средств фиксации доступа к данным и попыток нарушения системы защиты,
- разработка принципов защиты конкретных данных и объектов проектирования;

6) *обеспечение восстановления БД*: разработка организационных средств архивирования и принципов восстановления БД; разработка дополнительных программных средств и технологических процессов восстановления БД после сбоев;

7) *анализ обращений пользователей*: сбор статистики по характеру запросов, времени их выполнения;

8) *анализ эффективности функционирования БД*: анализ показателей функционирования БД, планирование реструктуризации;

9) *работа с конечными пользователями*: сбор информации об изменении предметной области, об оценке работы БД, обучение и консультирование пользователей;

10) *подготовка и поддержание системных средств*: анализ существующих на рынке программных средств и возможность их использования, проверка работоспособности закупаемых программных средств;

11) *организационно-методическая работа по проектированию БД*:

- выбор пли создание методики проектирования БД;
- определение целей и направления развития системы в целом;
- планирование этапов развития БД;
- обеспечение возможностей комплексной отладки множества приложений, взаимодействующих с БД и т. д.

9.2. Защита баз данных

9.2.1. Актуальность защиты базы данных

Практически ни одна современная компания не может обойтись без использования БД. В БД хранится и накапливается в течение длительного времени информация, критически важная для деятельности компании. Утечка данных из базы вследствие компьютерного браконьерства, их искажение из-за сбоев в компьютерной системе, ошибок пользователей и программных средств, потеря в случае различных катастроф (наводнений, землетрясений, пожаров) могут парализовать работу компании и нанести ей огромный материальный ущерб. Все это делает актуальной проблему защиты данных.

Чтобы обеспечить защиту данных в компьютерных системах, необходимо определить соответствующий комплекс мер.

9.2.2. Методы защиты баз данных

Методы защиты баз данных в различных СУБД несколько отличаются друг от друга. Анализ современных СУБД фирм Borland и Microsoft показывает, что они условно делятся на две группы: *основные* и *дополнительные*.

К *основным средствам* защиты относится:

- парольная защита;
- шифрование данных и программ;
- разграничение прав доступа к объектам базы данных;
- защита полей и записей таблиц БД.

Шифрование базы данных. Этот способ объединяет два средства: *кодирование* и *пароли* баз данных. Чтобы зашифровать базу данных, достаточно установить пароль для открытия базы данных. Стойкий алгоритм шифрования баз данных в формате Access 2007–2013 исключает их несанкционированный просмотр. Все данные становятся нечитаемыми в других программных средствах, и для того чтобы использовать эту базу данных, пользователи должны вводить пароль.

Пароли устанавливаются пользователями или администраторами БД. Учет и хранение паролей выполняется самой СУБД. Обычно пароли хранятся в определенных системных файлах СУБД в зашифрованном виде. После ввода пароля пользователю СУБД предоставляются все возможности по работе с БД.

Основной недостаток парольной защиты состоит в том, что все пользователи, использующие одинаковый пароль, с точки зрения вычислительной системы неразличимы. Неудобство парольной защиты для пользователя состоит в том, что пароль надо запоминать пли записывать. При небрежном отношении к записям пароль может стать достоянием других.

Для БД, которая используется небольшой группой пользователей, или на автономном компьютере, обычно достаточно установки пароля. Парольная защита может использоваться в дополнение к защите на уровне пользователя. В этом случае устанавливать парольную защиту может пользователь, обладающий правами администратора БД.

Процедура установки пароля в Microsoft Access:

1) открыть базу данных в режиме монопольного доступа (рис. 9.1);

2) выполнить команду *вкладка «Файл»* → *Сведения* → *Задать пароль базы данных* – откроется окно диалога «Задание пароля базы данных» (рис. 9.2);

3) ввести пароль в поле «Пароль». При вводе пароля соблюдать следующие правила:

- учитывать регистр клавиатуры;
- пароль может содержать от 1 до 20 знаков и включать в себя буквы, цифры, пробелы;
- пароль не должен содержать знаки " \ [ ] : | < > + = ; , . ? *, пробелы в начале имени, управляющие знаки с кодами ASCII от 10 до 31;

4) для подтверждения пароля ввести его еще раз в поле «Подтверждение»;

5) OK – теперь пароль задан. При следующем открытии базы данных появится диалоговое окно, в которое необходимо ввести пароль (рис. 9.3).

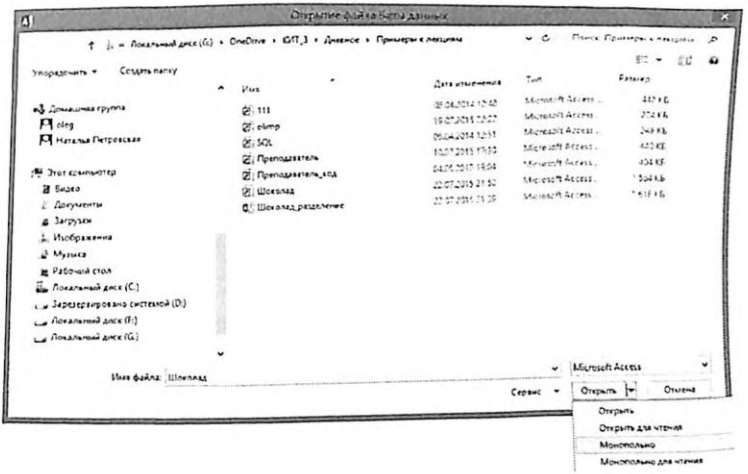


Рис. 9.1. Открытие файла базы данных в монопольном режиме

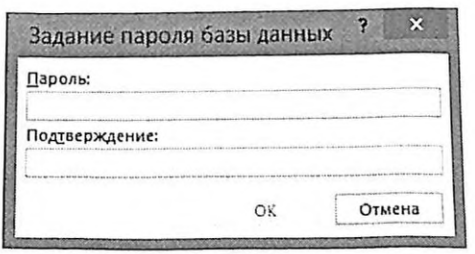


Рис. 9.2. Окно диалога «Задание пароля базы данных»

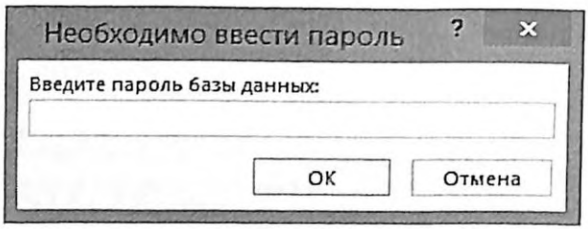


Рис. 9.3. Окно диалога для ввода пароля

Процедура удаления пароля:

1) открыть базу данных в режиме монопольного доступа (рис. 9.1);

2) выполнить команду *вкладка «Файл» → Сведения →* → *Удалить пароль базы данных* – откроется окно диалога «Удаление пароля базы данных» (рис. 9.4);

3) в этом окне ввести текущий пароль;

4) OK – пароль удален.

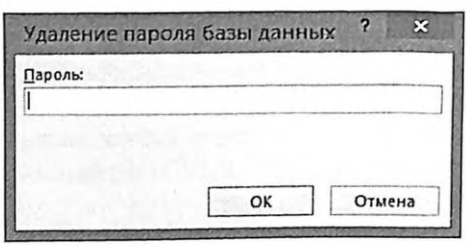


Рис. 9.4. Окно диалога «Удаление пароля базы данных»

Средств изменения пароля БД в Microsoft Access нет, поэтому для изменения пароля следует удалить текущий пароль, а затем определить новый.

При использовании парольной защиты нужно иметь ввиду, что парольная защита не защищает БД от удаления.

Разграничение прав доступа. В целях контроля использования основных ресурсов СУБД во многих системах имеются средства *установления прав доступа к объектам* БД. Права доступа определяют возможные действия над объектами. *Владелец объекта* (пользователь, создавший объект), а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа. Разрешение на доступ к конкретным объектам базы данных сохраняется в *файле рабочей группы*.

Файл рабочей группы содержит данные о пользователях группы и считывается во время запуска. Файл содержит следующую информацию:

- имена учетных записей пользователей;
- пароли пользователей,
- имена групп, в которые входят пользователи.

При запуске СУБД от пользователя требуется идентифицировать себя и ввести пароль.

По отношению к *таблицам* могут быть предоставлены следующие права доступа:

- просмотр (чтение) данных;
- изменение (редактирование) данных;
- добавление новых записей;
- добавление и удаление данных;
- изменение структуры таблицы.

К данным, имеющимся в таблице, могут применяться меры защиты по отношению к отдельным полям и отдельным записям.

Защита данных в *полях* таблиц предусматривает следующие уровни прав доступа:

- полный запрет доступа;
- только чтение;
- разрешение всех операций (просмотр, ввод новых значений, удаление и изменение).

По отношению к *формам* могут предусматриваться две основные операции – *вызов для работы* и *проектирование* (режим конструктора):

- запрет вызова конструктора целесообразно выполнять для экранных форм готовых приложений, чтобы конечный пользователь случайно не изменил приложение;
- защита отдельных элементов. Например, некоторые поля исходной таблицы вообще могут отсутствовать или быть скрыты от пользователя, а некоторые поля могут быть доступны для просмотра.

Отчеты во многом похожи на экранные формы. На отчеты, так же как и на формы, может налагаться запрет на вызов средств их разработки.

Дополнительные средства защиты БД. К дополнительным средствам защиты БД можно отнести такие, которые нельзя прямо отнести к средствам защиты, но которые непосредственно влияют на безопасность данных.

К таким средствам относятся:

- встроенные средства контроля значений данных в соответствии с типами;

– средства повышения достоверности вводимых данных;

– средства обеспечения целостности связей таблиц;

– средства организации совместного использования объектов БД в сети.

9.2.3. Правовая охрана баз данных

В технически развитых странах формируются информационные ресурсы, и происходит переход от индустриальной экономики к экономике, основанной на информации. Сейчас по своему социальному значению информатизация общества сопоставима с ее индустриализацией. Наш век называют веком информации. Информация стала главной ценностью земной цивилизации. Это послужило причиной обращения законодателей многих стран к проблеме правового регулирования производства и распространения информации.

Республика Беларусь стремится к созданию цивилизованного информационного рынка. Об этом свидетельствуют принятые указы, постановления, законы:

– «Об информатизации»;

– «О научно-технической информации»;

– «О национальном архивном фонде и архивах в Республике Беларусь»;

– «О печати и других средствах массовой информации»;

– «О правовой охране программ для ЭВМ и баз данных»;

– «О введении в действие Единой системы классификации и кодирования технико-экономической и социальной информации Республики Беларусь» и др.

Закон «Об информатизации» важен тем, что в нем даются определения основных понятий с правовой точки зрения.

Закон «О научно-технической информации», принятый 5 мая 1999 г. устанавливает правовые основы регулирования правоотношений, связанных с созданием, накоплением, поиском, получением, хранением, обработкой, распространением и использованием научно-технической информации в Республике Беларусь.

Закон «О правовой охране программ для ЭВМ и баз данных» и закон «Об авторском праве и смежных правах» следует рассматривать взаимосвязано, т. к. их положения затрагивают правовую охрану программ для персональных компьютеров и баз данных.

9.3. Оптимизация работы базы данных. Сжатие и восстановление базы данных

Сжатие – это операция уменьшения занимаемого пространства на жестком диске базой данных.

В процессе работы с БД приходится удалять ее объекты. Занимаемое удаленными объектами и записями таблиц дисковое пространство не освобождается, а отмечается как неиспользуемое. При очередном добавлении объектов и записей снова выделяется дисковое пространство под эти объекты и размер файла БД увеличивается. Т. е. файл БД становится фрагментированным и место на диске используется нерационально. Сжатие БД приводит к созданию ее копии, в которой диск используется более экономно, и *оптимизирует* быстродействие баз данных.

Поскольку данные, хранимые компьютерными средствами, подвержены потерям и повреждениям, вызываемым разными событиями, важно обеспечить средства восстановления данных. Приведение базы данных точно в то состояние, которое существовало перед отказом не всегда возможно, но процедуры восстановления базы данных могут привести ее в состояние, существовавшее незадолго до отказа.

Восстановление применяется при повреждениях БД, не позволяющих пользователю нормально работать с базой данных или даже открыть ее (воздействие компьютерных вирусов, наличие дефектов (физических или логических) на диске).

9.3.1. Сжатие и восстановление базы данных в Microsoft Office Access 2013

В большинстве случаев Microsoft Access определяет повреждение БД при попытке открыть, сжать, закодировать или раскодировать ее.

Сжатие и восстановление БД в Microsoft Access может осуществляться как в ручном, так и в автоматическом режимах.

Перед запуском процесса сжатия и восстановления рекомендуется создать резервную копию базы данных (раздел 9.4.2. «Резервное копирование базы данных»).

Автоматическое сжатие и восстановление базы данных при закрытии. Microsoft Access 2013 позволяет настроить автоматическое сжатие и восстановление базы данных при каждом закрытии. Для этого нужно выполнить следующие действия:

1) перейти на вкладку «Файл»;

2) на вкладке «Файл» выбрать пункт «Параметры» – откроется окно диалога «Параметры Access»;

3) в этом окне перейти к разделу «Текущая база данных»;

4) в разделе «Параметры приложений» установить флажок «Сжимать при закрытии» (рис. 9.5).

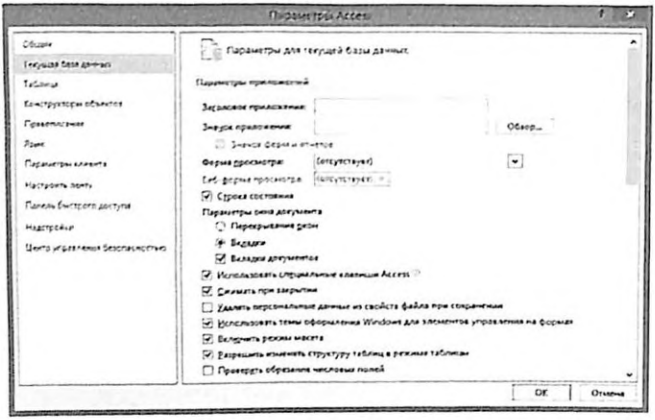


Рис. 9.5. Окно диалога «Параметры Access»

Сжатие и восстановление открытой базы данных вручную. Чтобы провести сжатие и восстановление открытой базы данных вручную, нужно выполнить следующие действия:

1) перейти на вкладку «Файл»;

2) на вкладке «Файл» перейти к разделу «Сведения»;

3) в этом разделе нажать кнопку «Сжать и восстановить базу данных».

9.3.2. Резервное копирование базы данных

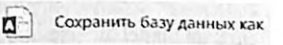
Одним из средств восстановления БД является также регулярное и частое резервное копирование БД. *Резервное копирование БД* – это создание точной копии БД. Если повреждается или теряется файл, являющийся частью БД, то из резервной копии можно извлечь копию этого файла и восстановить его в базе.

Для создания резервной копии БД необходимо выполнить следующие действия:

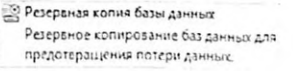
1) перейти на вкладку «Файл»;

2) на вкладке «Файл» нажать кнопку «Сохранить как»;

3) в разделе «Типы файлов» выбрать инструмент «Сохранить базу данных как»;



4) в разделе «Дополнительно» выбрать инструмент «Создать резервную копию базы данных».



Примечание. Резервную копию рекомендуется создавать перед выполнением любого запроса на изменение, особенно если в результате будет изменен или удален большой объем данных.

9.3.3. Разделение базы данных

Разделение базы данных помогает избежать повреждения файла базы данных и потери данных путем сохранения данных в отдельном файле, прямой доступ к которому закрыт для пользователей.

Рекомендуется выполнять разделение любой базы данных, которую несколько пользователей совместно используют в сети. Разделив совместно используемую базу данных, можно повысить производительность и уменьшить вероятность повреждения файла базы данных.

При разделении базы данных выполняется ее реорганизация в два файла – базу данных с таблицами, в которой содержатся таблицы данных, и клиентскую базу данных, включающую все остальные объекты базы данных, например запросы, формы и отчеты. Каждый пользователь взаимодействует с данными, используя локальную копию клиентской базы данных.

Разделение базы данных выполняется с помощью мастера разделения баз данных. После разделения базы данных необходимо распространить клиентскую базу данных среди пользователей.

Достоинства разделенной базы данных:

– *улучшенная производительность*. Производительность базы данных обычно значительно улучшается, так как по сети пересылаются только данные. В совместно используемой неразделенной базе данных по сети пересылаются не только данные, но и сами объекты базы данных – таблицы, запросы, формы, отчеты, макросы и модули;

– *повышенная доступность*. Поскольку по сети пересылаются только данные, транзакции базы данных (например, изменение записи), завершаются быстрее, что приводит к большей доступности данных для внесения изменений;

– *улучшенная безопасность*. Если база данных с таблицами хранится на компьютере с файловой системой NTFS, для лучшей защиты данных можно использовать средства безопасности NTFS. Поскольку пользователи

получают доступ к базе данных с таблицами через связанные таблицы, менее вероятно, что злоумышленник сможет получить несанкционированный доступ к данным, похитив клиентскую базу данных или выбрав себя за полномочного пользователя;

– *улучшенная надежность*. Если пользователь столкнулся с проблемой и произошло неожиданное закрытие базы данных, любые повреждения файла базы данных касаются только копии клиентской базы данных, открытой пользователем. Поскольку пользователь получает доступ к данным, которые содержатся в базе данных с таблицами, через связанные таблицы, повреждение файла базы данных с таблицами намного менее вероятно;

– *гибкая среда разработки*. Поскольку каждый пользователь работает с локальной копией клиентской базы данных, он может разрабатывать запросы, формы, отчеты и другие объекты базы данных независимо, не мешая другим пользователям. Таким же образом можно разработать и распространить новую версию клиентской базы данных без нарушения доступа к данным, которые хранятся в базе данных с таблицами.

Перед разделением базы данных рекомендуется создать ее резервную копию.

Для разделения базы данных нужно выполнить следующие действия:

1) создайте на своем компьютере копию базы данных, которую требуется разделить;

2) открыть копию базы данных;

3) выполнить команду *вкладка «Работа с базами данных» → группа «Перемещение данных» → инструмент «База данных Access»* – будет запущен мастер разделения баз данных (рис. 9.6);

4) нажать кнопку «Разделить» – откроется окно диалога «Создание базы данных с таблицами»;

5) в этом окне задать имя, тип и местоположение для файла данных с таблицами. Рекомендуется использовать имя, предложенное программой Access. Оно представляет собой исходное имя файла, к которому перед расширением файла добавлены символы *_be*, указывающие на то, что этот файл является базой данных с таблицами;

6) подтвердить разделение базы данных. После завершения работы мастера появится сообщение, подтверждающее результат (рис. 9.7).

Разделение базы данных выполнено. *Клиентская база данных* – это файл, с запуска которого началась работа с базой данных (копия исходной базы данных), а база данных с таблицами расположена в сети в месте, указанном на пятом этапе этой процедуры.

После разделения базы данных нужно распространить клиентскую базу данных среди пользователей, чтобы они смогли начать работу с ней. Для этого следует выполнить одно из действий, указанных ниже:

– отправить пользователям базы данных сообщение электронной почты и вложить в него файл клиентской базы данных. Включить все инструкции, которые помогут пользователям немедленно приступить к работе с клиентской базой данных;

– сохранить файл клиентской базы данных в таком месте в сети, к которому разрешен доступ всем пользователям базы данных, а затем отправить пользователям сообщение с указанием расположения файла в сети и другими инструкциями, которые могут им потребоваться для получения доступа к базе данных;

– распространить файл клиентской базы данных, используя съемный носитель, например компакт-диск или флэш-накопитель USB.

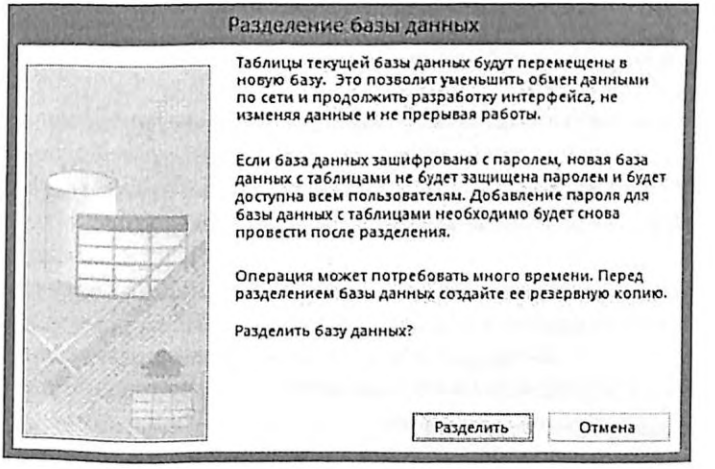


Рис. 9.6. Стартовое окно мастера «Разделение базы данных»

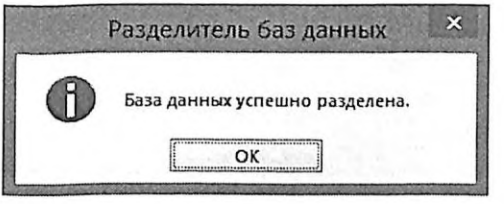


Рис. 9.7. Сообщение об успешном разделении БД

Словарь терминов

Администратор базы данных – это физическое лицо или группа лиц, отвечающих за выработку требований к базе данных, ее проектирование, создание, эффективное использование и сопровождение.

Восстановление базы данных – применяется при повреждениях БД, не позволяющих пользователю нормально работать с базой данных или даже открыть ее.

Конечные пользователи – это основная категория пользователей, в интересах которых создается база данных.

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

Разделение базы данных – помогает избежать повреждения файла базы данных и потери данных путем сохранения данных в отдельном файле, прямой доступ к которому закрыт для пользователей.

Разработчики и администраторы приложений – это группа пользователей, которая функционирует во время проектирования, создания и реорганизации базы данных.

Сжатие базы данных – это операция уменьшения занимаемого пространства на жестком диске базой данных.

Вопросы и задания для самоконтроля

1. Кто такой пользователь базы данных?
2. Кто такие конечные пользователи базы данных?
3. Кто такой администратор базы данных, каковы его функции?
4. Кто такие разработчики и администраторы приложений?
5. Каковы функции группы администратора БД?
6. Чем обусловлена актуальность защиты базы данных?
7. Назовите и охарактеризуйте методы защиты базы данных.
8. Как в Республике Беларусь реализуется правовая охрана баз данных?
9. Какими методами осуществляется оптимизация работы базы данных.
10. С какой целью проводится разделение базы данных?

10. Хранилища данных

Накопление больших объемов данных в последнее время сделали актуальными прикладные задачи, предназначенные для извлечения, сбора и представления конечному пользователю информации, необходимой для анализа текущего состояния дел и прогноза будущего решения. Такие ИС получили название *систем поддержки принятия решений* (СППР). Исторически первыми такими системами стали *ИС руководителя (EIS – Executive Information Systems)*.

Новыми же информационными технологиями, которые обеспечивают аналитикам, управленцам и руководителям высшего звена возможность изучать большие объемы взаимосвязанных данных при помощи быстрого интерактивного отображения информации на разных уровнях детализации с различных точек зрения в соответствии с представлениями пользователя о предметной области, являются *хранилища данных (Datawarehouse)* и *оперативный анализ данных (On-LineAnalyticalProcessing, OLAP)*.

10.1. Понятие хранилища данных

Хранилище данных (ХД) – ориентированная на поддержку управленческих решений автоматизированная система, состоящая из организационной структуры, технических средств, базы или совокупности базы данных (БД) и программного обеспечения, которое выполняет, как правило, следующие функции:

- извлечение данных из разрозненных источников, их трансформация и загрузка в хранилище;
- администрирование данных и хранилища;
- извлечение данных из хранилища, аналитическая обработка и представление данных конечным пользователям.

Ральф Кимбалл (Ralph Kimball), один из авторов концепции хранилищ данных, описывал хранилище данных как «место, где люди могут получить доступ к своим данным». Он же сформулировал и основные требования к хранилищам данных:

- поддержка высокой скорости получения данных из хранилища;
- поддержка внутренней непротиворечивости данных;
- возможность получения и сравнения так называемых *срезов данных (slice and dice)*;
- наличие удобных утилит просмотра данных в хранилище;
- полнота и достоверность хранимых данных;
- поддержка качественного процесса пополнения данных.

10.2. Технология оперативной аналитической обработки данных OLAP

Системы поддержки принятия решений обычно обладают средствами предоставления пользователю агрегатных данных для различных выборок из исходного набора в удобном для восприятия и анализа виде. Как правило, такие агрегатные функции образуют многомерный (и, следовательно, нереляционный) набор данных (нередко называемый *гиперкубом* или *метакубом*), оси которого содержат *параметры*, а *ячейки* – зависящие от них *агрегатные данные*. Вдоль каждой оси данные могут быть организованы в виде *иерархии*, представляющей различные уровни их детализации. Благодаря такой модели данных пользователи могут формулировать сложные запросы, генерировать отчеты, получать подмножества данных (раздел 2.7 «Многомерная модель данных»).

Технология комплексного многомерного анализа данных получила название *OLAP (On-Line Analytical Processing)*. OLAP – это *ключевой компонент организации* хранилищ данных.

Концепция OLAP была описана в 1993 году Эдгаром Коддом, известным исследователем баз данных и автором реляционной модели данных. В 1995 году на основе требований, изложенных Коддом, был сформулирован так называемый *тест FASMI (Fast Analysis of Shared Multidimensional Information – быстрый анализ разделяемой многомерной информации)*, включающий следующие требования к приложениям для многомерного анализа:

– предоставление пользователю результатов анализа за приемлемое время (обычно не более 5 сек), пусть даже ценой менее детального анализа;

– возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;

– многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;

– многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий (это – ключевое требование OLAP);

– возможность обращения к любой нужной информации независимо от ее объема и места хранения.

Следует отметить, что OLAP-функциональность может быть реализована различными способами, начиная с простейших средств анализа данных в офисных приложениях и заканчивая распределенными аналитическими системами, основанными на серверных продуктах.

10.3. Отличия хранилища данных от базы данных. Классификация и технологические решения ХД

Типичное хранилище данных, как правило, отличается от обычной реляционной базы данных.

1. Обычные базы данных предназначены для того, чтобы помочь пользователям выполнять повседневную работу, тогда как хранилища данных предназначены для принятия решений.

Пример. Продажа товара и выписка счета производятся с использованием базы данных, предназначенной для обработки транзакций, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, с помощью хранилища данных.

Обычные базы данных подвержены постоянным изменениям в процессе работы пользователей, а хранилище данных относительно стабильно: данные в нем обычно обновляются согласно расписанию (например, еженедельно, ежедневно или ежечасно – в зависимости от потребностей). В идеале процесс пополнения представляет собой просто добавление новых данных за определенный период времени без изменения прежней информации, уже находящейся в хранилище.

Обычные базы данных чаще всего являются источником данных, попадающих в хранилище. Кроме того, хранилище может пополняться за счет внешних источников, например статистических отчетов.

Несмотря на то, что хранилища обладают общими свойствами, разные типы хранилищ имеют свои индивидуальные особенности.

10.3.1. Финансовые хранилища данных

В большинстве случаев финансовые хранилища данных – это хранилища, которые организации строят в первую очередь. Создание финансового хранилища – очень привлекательное решение, поскольку:

– финансовые данные *всегда* находятся в центре внимания организации. Поэтому привлечь внимание к хорошо построенному финансовому хранилищу данных очень легко;

– в большинстве организаций (но не во всех) финансовые данные представляют самые маленькие объемы данных из имеющихся;

– финансы охватывают все аспекты функционирования корпорации и имеют один общий знаменатель – деньги;

– финансовые данные по своей природе имеют структуру, на которую напрямую влияет повседневная практика обработки финансовой информации, и т. д.

По этим причинам финансы становятся предпочтительной областью построения корпоративного хранилища данных.

10.3.2. Хранилища данных в области страхования

Хранилища данных в области страхования за некоторыми небольшими исключениями похожи на другие хранилища. Первое исключение (и это особенно справедливо в отношении страхования жизни) заключается в том, что продолжительность существования имеющихся хранилищ очень велика. Такие хранилища содержат данные, которые являются старыми, очень старыми. В этом случае вполне разумно обращаться к данным начала столетия – к началу XX века. Причина, по которой страховые компании вынуждены интересоваться столь «древней историй», – актуарная обработка данных. Практически для каждого дела приводится довод, что деятельность, которой организация занималась в 1950 году, практически не связана с сегодняшним занятием. И часто этот довод звучит правдоподобно.

Второе отличие этих хранилищ определяется датами, которые хранятся в этом бизнесе. Среда страхования отличается наличием большего числа дат, связанных с бизнесом, чем какой-либо другой вид деятельности. Так, в сфере розничной торговли имеется несколько важных дат: дата продажи, дата появления на складе, возможно, дата производства. В банковском деле существенна дата транзакции. В телекоммуникации – это дата телефонного звонка. В страховании же присутствуют даты всевозможных типов.

Наконец, третье отличие заключается в том, что эти хранилища данных используют свой рабочий цикл деловой активности. Большинство организаций имеет весьма ограниченный и короткий экономический цикл. Так, в банках это обналичивание чека, в торговле – покупка изделия, в телефонной компании – звонок. Однако, в страховании им может быть заявка на страховое возмещение, которая может быть удовлетворена спустя пять лет. Или закрытие полиса может сопровождаться двухмесячной отсрочкой. Резюмируя, можно сказать, что скорость, с которой функционирует страхование, отличается от скорости, характерной для других отраслей.

10.3.3. Хранилища данных для управления людскими ресурсами

Хранилища данных для управления людскими ресурсами имеют весьма существенные отличия от других хранилищ:

– *число предметных областей*. Такое хранилище данных неизбежно имеет одну важную предметную область – это работник. Практически все остальное подчинено этой области или занимает второстепенное положение. Большинство же других хранилищ данных имеют несколько базовых предметных областей;

– такие хранилища используют *очень мало транзакций*.

Пример. Имеется дата, когда субъект становится работником. Дата, когда человек увольняется. Годовые прибавки и повышения. Но, кроме транзакций фонда заработной

платы и прочих редких, сгенерированных работником, транзакций, в таком хранилище практически больше ничего нет. Сравните сферу управления людскими ресурсами с коммуникацией или банковской средой, и разница в числе транзакций станет очевидной.

10.3.4. Глобальные хранилища данных

Глобальные хранилища данных предназначены для глобального представления корпорации. Различают три типа таких хранилищ:

– с географически превалирующей обработкой данных;

Пример. Необходимо интегрировать бизнес в Гонконге с бизнесом в Париже, который в свою очередь следует интегрировать с Рио-де-Жанейро, а тот – с Нью-Йорком.

– с функционально превалирующей обработкой данных. Производственная деятельность должна быть интегрирована с поставками, которые необходимо интегрировать с продажами, а те – с исследованиями и т. д.;

– с отраслевой превалирующей обработкой данных.

Пример. Требуется интегрировать печатное дело с консалтингом, который подлежит интеграции с бизнесом в сфере медицинского оборудования, а тот со специализацией в области программного обеспечения.

Особенность глобального хранилища данных заключается в том, что на глобальном уровне зачастую очень мало общих измерений. Единственное общее измерение – это *деньги*. И интеграция бизнеса может быть достигнута только с помощью этого измерения. Другие же измерения могут иметь или не иметь смысл на глобальном уровне. Так, *клиент*, *продукт*, *поставщик*, *транзакция* – все эти классические предметные области могут как присутствовать, так и отсутствовать в глобальной интегрированной сфере – глобальном хранилище данных.

Помимо этого, глобальное хранилище данных подвержено тому, от чего другие хранилища «благополучно избавлены» – от «разрушительного» воздействия перемен. Если в прочих хранилищах изменения базовых данных случаются нечасто, то для этого типа хранилищ они происходят постоянно и в самом основании. Так, в любой момент может быть открыто новое месторождение нефти, например, в Венесуэле. В следующую минуту в Перу вспыхнет революция. А затем, благодаря развитию технологии, станут доступными залежи нефти в Луизиане. Вслед за этим последуют санкции ОПЕК. В Мексике будет изменено законодательство. И так далее. Если рассматривать ситуацию в глобальном аспекте, то видно, что изменения носят постоянный характер. Поэтому структура и технология, используемая для размещения и обслуживания глобального хранилища данных, должна позволять поддерживать эти непрерывные перемены.

10.3.5. Хранилища данных с возможностями Data Mining/Data Mining и Exploration

Хранилища данных, поддерживающие технологию *Data Mining и Exploration* (методы «добычи» и исследования данных), являются гибридом классических хранилищ. Такие хранилища используются для выполнения мощной статистической обработки данных. Эти хранилища являются:

- очень детальными;
- глубоко историческими;
- оптимизированными для статистического анализа.

Кроме того, для таких хранилищ характерна ориентация на какой-либо проект. Это означает, что, в отличие от всех других типов хранилищ данных, их перестают использовать сразу по завершении анализа, ради которого они создавались.

Еще одно важное отличие хранилищ данных с возможностями *Data Mining/Data Mining и Exploration* заключается в том, что эти хранилища очень часто включают внешние данные. Такие данные очень полезны с точки зрения обеспечения бизнес-перспективы, которую не так легко увидеть без их участия.

10.3.6. Хранилища данных в области телекоммуникаций

Отличительная особенность этих хранилищ состоит в том, что они в значительной степени определяются данными, сгенерированными в деталях *на уровне звонка*. Разумеется, в отрасли телекоммуникации присутствует множество других типов данных. Но ни одна другая область хранилищ данных не предопределяется в такой степени размером одной предметной области – *деталями на уровне звонка*.

Существует много способов хранения деталей на уровне звонка:

- хранение деталей на уровне звонка только за несколько месяцев;
- хранение множества деталей на уровне звонка, размещенных на различных носителях;
- резюмирование или агрегирование деталей на уровне звонка;
- хранение только отобранных деталей на уровне звонка, и так далее.

К сожалению, несмотря на разнообразие методов обработки, для данного хранилища данных обработка может быть выполнена только над деталями на уровне звонка. А работа на итоговом или агрегированном уровне просто невозможна.

10.4. Обзор программного обеспечения для разработки хранилищ данных

В настоящем разделе дается краткий обзор программного обеспечения для разработки хранилищ данных основных производителей программного обеспечения.

10.4.1. Data Warehouse Plus

Data Warehouse Plus – это решение компании IBM.

Несущая СУБД для хранилища данных – семейство объектно-реляционных СУБД *DB2*. Язык манипулирования данными – SQL.

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

10.4.2. Warehouse Technology Initiative

Warehouse Technology Initiative – это решение компании Oracle.

Несущая СУБД для хранилища данных – семейство объектно-реляционных СУБД *Oracle 11g/10g*. Язык манипулирования данными – SQL.

Компания выпускает специальный CASE-инструментарий для проектирования ХД.

10.4.3. Enterprise Information Factory

Enterprise Information Factory – это решение компании NCR.

Несущая СУБД для хранилища данных – реляционная СУБД *Teradata*.

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

10.4.4. Rapid Data Warehousing

Rapid Data Warehousing – это решение компании SAS Institute.

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

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

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

– наличие у компании сервера многомерных БД;

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

10.4.5. Warehouse WORKS

Warehouse WORKS – это решение компании Sybase.

Несущая СУБД для хранилища данных – реляционная СУБД Sybase *System 9*. Язык манипулирования данными – SQL. Компания выпускает специальный CASE-инструментарий для проектирования ХД.

10.4.6. Microsoft Data Warehousing Framework

Microsoft Data Warehousing Framework – это решение компании Microsoft.

Компания сравнительно недавно стала активно предлагать комплексные решения в области ХД. Целью корпорации Microsoft является создание инструментальной и технологической среды, которая позволила бы минимизировать затраты на создание ХД и сделала бы этот процесс доступным для массового пользователя. Открытость среды Microsoft Data Warehousing Framework обеспечила ее поддержку многими производителями программного обеспечения.

Цель Microsoft Data Warehousing Framework состоит в том, чтобы упростить разработку, внедрение и администрирование решений на основе

ХД. Эта спецификация призвана обеспечить:

– открытую архитектуру, которая интегрируется и расширяется третьими фирмами;

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

– доступ к разделяемым метаданным со стороны процессов разработки ХД.

Несущая СУБД для хранилища данных – реляционная СУБД *MS SQL Server 2005/2008*. Язык манипулирования данными – SQL со встроенными средствами обработки многомерных кубов.

10.4.7. Open Data Warehouse Initiative

Open Data Warehouse Initiative – это решение компании Software AG.

Несущая СУБД для хранилища данных – сетевая СУБД *ADABAS*. Язык манипулирования данными – Natural 4GL.

У компании имеются собственные средства извлечения и анализа данных, а также программный продукт управления хранилищем данных *SourcePoint*.

Словарь терминов

OLAP (On-Line Analytical Processing) – технология комплексного многомерного анализа данных, ключевой компонент организации хранилищ данных.

Система поддержки принятия решений – информационная система для извлечения, сбора и представления конечному пользователю информации, необходимой для анализа текущего состояния дел и прогноза будущего решения.

Хранилище данных – ориентированная на поддержку управленческих решений автоматизированная система, состоящая из организационной структуры, технических средств, базы или совокупности базы данных (БД) и программного обеспечения.

Вопросы и задания для самоконтроля

1. Что собой представляет система поддержки принятия решений?
2. Что собой представляет хранилище данных, каковы его функции?
3. В чем состоит сущность технологии оперативной аналитической обработки данных OLAP?
4. Чем хранилище данных отличается от базы данных?
5. Охарактеризуйте финансовые хранилища данных.
6. Каковы отличительные особенности хранилища данных с возможностями Data Mining/Data Mining и Exploration?
7. Назовите и охарактеризуйте программные средства для разработки хранилищ данных.

11. Базы знаний и модели представления знаний

11.1. Данные и знания. Основные понятия

Совершенствование средств вычислительной техники позволяет значительно расширить сферу их применения в народном хозяйстве. В настоящее время при помощи ЭВМ решаются различные типы задач:

– *вычислительные задачи*, в которых в соответствии с определенным алгоритмом и множеством входных данных получают множество результатов. Алгоритм в таких задачах выступает в качестве строгой последовательности операций.

– *информационные задачи* – это нахождение части базы данных, соответствующей внешнему запросу. Алгоритм здесь – это последовательность информационно-поисковых процедур, а база данных – набор декларативных знаний.

– *задачи принятия решений*, когда на основании определенного набора критериев из множества альтернатив выбирается наиболее подходящая для достижения поставленных целей. Цели и критерии могут быть как постоянными, так и изменяться в процессе решения задачи.

– *логические задачи*, в которых по описанию начальной и целевой ситуаций из имеющегося набора действий синтезируется алгоритм достижения цели.

Для решения двух последних типов задач могут применяться методы искусственного интеллекта, основанные на знаниях. Одной из разновидностей систем, использующих эти методы, являются экспертные системы (ЭС). Они представляют собой попытку создания человеко-машинных комплексов для решения слабо формализуемых задач или задач, не имеющих алгоритмического решения.

При изучении интеллектуальных систем традиционно возникает вопрос: что же такое знания и чем они отличаются от данных, обрабатываемых на компьютере.

Данные представляют собой отдельные факты, характеризующие объекты, процессы и явления, а также их свойства. При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

– данные как результат измерений и наблюдений;

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

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

– данные в компьютере на языке описания данных;

– базы данных на машинных носителях информации.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Совокупность знаний, нужных *для принятия решений*, принято называть *знаниями о предметной области*. В любой предметной области есть свои понятия и связи между ними, своя терминология, свои законы, связывающие между собой объекты данных предметной области, свои процессы и события. Кроме того, каждая предметная область имеет свои методы решения задач.

Таким образом, *знания* – это закономерности предметной области: принципы, связи, законы, – полученные в результате практической деятельности, профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области. При обработке на ЭВМ знания трансформируются аналогично данным:

– знания в памяти человека как результат мышления;

– материальные носители знаний (учебники, методические пособия);

– поле знаний – условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

– знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы);

– база знаний на машинных носителях информации.

Рассмотрим особенности знаний, которые отличают их от данных.

1. *Интерпретация*. Данные, хранимые в памяти ЭВМ, могут интерпретироваться только соответствующей программой. Данные без программы не несут никакой информации, в то время как знания имеют интерпретацию, поскольку они содержат одновременно и данные, и описание данных.

2. *Наличие классифицирующих связей*. Ни одна из разнообразных форм представления данных в отличие от знаний не обеспечивает эффективного описания связей между различными типами данных.

3. *Наличие ситуативных связей*. Совместимость отдельных событий или факторов в некоторой ситуации определяется *ситуативными связями*, а также такими отношениями, как одновременность, расположение в одной области пространства и т. д. Ситуативные связи позволяют строить процедуры анализа знаний на совместимость, противоречивость и другие, которые трудно реализовать при хранении традиционных наборов данных.

Знание обладает важным свойством – *ценностью*. Ценность знания в различные периоды различна в зависимости от того, какие задачи выдвигаются на передний план в определенный отрезок времени. При этом выделяют два вида ценности знания: непосредственную и потенциальную.

11.2. Классификация знаний

11.2.1. Классификация знаний по глубине

По глубине выделяют знания:

– *поверхностные* – это знания о видимых взаимосвязях между отдельными событиями и фактами предметной области;

– *глубинные* – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Современные системы работают в основном с поверхностными знаниями. Это связано с тем, что не всегда имеются универсальные методики, позволяющие выявлять глубинные структуры знаний и работать с ними.

11.2.2. Классификация знаний по выполняемым функциям

По выполняемым функциям различают знания:

– *декларативные знания*. Такие знания не содержат в явной форме процедур, которые нужно выполнить. Такие знания представляются множеством утверждений, не зависящих от того, где они применяются. Их использование предполагает полное описание пространства состояний моделируемого объекта, которое носит синтаксический характер. Вывод и поиск решений базируются на процедурах поиска в пространстве состояний. Эти процедуры учитывают специфику конкретной предметной области, т. е. ее *семантику*. Следовательно, при декларативной форме представления семантические и синтаксические знания в определенной мере отделены друг от друга, что придает названной форме большую универсальность и общность;

– *процедурные знания*. Такие знания содержат описание некоторых процедур. Состояние объекта представляется в виде набора процедур, с помощью которых обрабатывается определенный участок базы знаний. Процедурные знания обеспечивают более быстрый поиск решения по сравнению с декларативными знаниями, однако уступают им в возможности накопления и актуализации знаний.

Существует и более детализированный подход к классификации знаний. В частности, различают:

– *понятийные знания* – это набор понятий, которыми пользуются при решении конкретной задачи. Этот тип знаний вырабатывается в фундаментальных науках и теоретических разделах прикладных наук;

– *конструктивные знания* – это знания о наборах возможных структур объектов и взаимодействии между их частями. Получение этого типа знаний характерно для техники и большей части прикладных наук;

– *процедурные знания* – это используемые в выбранной предметной области методы, алгоритмы и программы, полезные для данного конкретного приложения, которые можно использовать, передавать и объединять в библиотеки;

– *фактографические знания* – количественные и качественные характеристики объектов и явлений;

– *метазнание* – это знание о знаниях, возникающее в результате сопоставления знаний. Если сопоставлять метазнания, то результатом будет *метаметазнание*, то есть метазнания – это знания о порядке и правилах применения знаний. На основании знаний получают метазнания, на основании метазнаний – метаметазнания и т. д.

11.2.3. Альтернативная классификация знаний

Ряд ведущих специалистов предлагает иной подход в классификации знаний. Согласно ему выделяются восемь типов знаний:

1) *базовые элементы, объекты реального мира*. Они связаны с непосредственным восприятием, не требуют обсуждения и добавляются к базе фактов в том виде, в котором они получены;

2) *утверждения и определения* основаны на базовых элементах и заранее рассматриваются как достоверные;

3) *концепции* представляют собой перегруппировки или обобщения базовых объектов. Для построения каждой концепции используются свои приемы. Например, в системах MECHO (Bundy, 1979) или AM (Lenat, 1977), – они построены на основе примеров, контрпримеров, частных случаев, более общих или аналогичных концепций;

4) *отношения* выражают как элементарные свойства базовых элементов, так и отношения между концепциями. Кроме того, к свойствам отношений относятся их большее или меньшее правдоподобие, большая или меньшая связь с данной ситуацией;

5) *теоремы и правила перезаписи*. Теоремы не представляют никакой пользы без экспертных правил их использования. Явное присутствие теорем в экспертных системах представляет их главное отличие от систем управления базами данных, в которых они либо отсутствуют, либо программируются. Модификация или добавление новых теорем является весьма трудоемкой, но необходимой процедурой, так как нужно обеспечить хорошее структурированное управление базой данных и оптимизировать получение ответов (Gallaire, 1987). Теоремы и правила перезаписи являются частным случаем продукционных правил с вполне определенными свойствами;

6) *алгоритмы решений* необходимы для выполнения определенных задач. Во всех случаях они связанны со знанием особого типа, поскольку определяемая ими последовательность действий оказывается оформленной в блок в строго необходимом порядке в отличие от других типов знания, где элементы информации могут появляться и располагаться без связи друг с другом;

7) *стратегии и эвристика* представляют собой врожденные или приобретенные правила поведения, которые позволяют в данной конкретной ситуации принять решение о необходимых действиях. Этот тип знаний использует информацию в порядке, обратном тому, в котором она была получена.

Пример. Можно привести рассуждение типа: «Я знаю, что это действие приводит к такому-то результату (информация типа 4), поэтому, если я хочу получить именно этот результат, я могу рассмотреть это действие».

Человек постоянно пользуется этим типом знаний при восприятии, формировании концепций, решении задач и формальных рассуждениях. Появление экспертных систем связанно с необходимостью принятия в расчет именно этого фундаментального типа человеческих знаний;

8) *метазнание* представляет собой любое знание о знании. Оно является фундаментальным понятием для систем, которые не только используют свою базу знаний такой, как она есть, но и умеют на ее основе делать выводы, структурировать ее, абстрагировать, обобщать, а также решать, в каких случаях она может быть полезна. Ввод знаний в систему является задачей не только специалистов данной области, но и самой системы, которая должна управлять этим процессом.

Метазнание присутствует на многих уровнях и представляет собой знание того, что известно и определяет значение коэффициента доверия к этому знанию, важность элементарной информации по отношению ко всему множеству знаний. Кроме того, сюда же относятся вопросы организации каждого типа знаний и указаний, когда и как они могут быть использованы.

Выделяют следующие разновидности метазнания:

– *метазнание объектов окружающего мира*. Решение проблем, охватывающих большие объемы знаний, требует умения обращаться с очень сложными и различными типами данных, как по содержанию, так и по взаимосвязи с другими элементами системы. Требуемая информация должна не только присутствовать в программных комментариях и в мыслях разработчика, но и быть доступной самой системе. Таким образом, с каждым элементарным объектом может быть связано смысловое значение – *концепт*;

Пример. В системах MECHO и MYCIN каждый из концептов содержит следующую информацию:

- описательную часть, которая уточняет характеристики и структуру данных;
- указатель на все известные примеры концепта;
- связи с другими концептами;
- указатель в иерархии схем на группу концептов того же семейства.

– *метазнание стратегий*. В продукционных системах стратегии также представлены в форме метаправил, поскольку они затрагивают сами правила. Они являются изолированными и доступными для системы. Эвристические законы управления поиском решения используют эти метаправила в качественном и декларативном виде, вместо того чтобы использовать количественные оценки. С применением числовых оценочных функций связан риск ошибки: они плохо читаемы, отражаемый ими частичный порядок приводит к сложным вычислениям, любая модификация приводит к возможности появления ошибки.

Стратегии, заданные в форме метаправил, являются более четкими и определенными. Выводы метаправил указывают на действия, которые необходимо предпринять в рассматриваемой ситуации. Таким образом они реализуют полезный потенциал, заключенный в множестве правил, и дают двойной эффект. Правила сначала группируются в подмножества. Каждое подмножество предназначено для определенной обработки и используется при выполнении соответствующих условий. Соответствие между конечными классами и подмножествами правил устанавливается с помощью правил заданий, которые составляют второй уровень знаний. Эти правила определяют, как следует выполнить данное задание наилучшим образом.

11.3. Модели представления знаний

Знания в базе знаний представляются в определенной форме. Форма представления знаний оказывает существенное влияние на характеристики и свойства системы, поэтому представление знаний является одной из важных проблем, характерных для систем, основанных на знаниях. Поскольку логический вывод и действия над знаниями производятся программным путем, знания не могут быть представлены непосредственно в том виде, в котором они используются человеком. В связи с этим для представления знаний используются формальные модели представления знаний. Язык, используемый для разработки систем, спроектированных на основе этих моделей, называется *языком представления знаний*.

При разработке конкретной модели преставления знаний стараются учесть следующие требования:

1) представление знаний должно быть *однородным*. Однородное представление приводит к упрощению механизма управления логическим выводом и упрощению управления знаниями;

2) представление знаний должно быть *понятным* экспертам и пользователям системы. В противном случае затрудняются приобретение знаний и их оценка.

Существует много моделей (и языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим:

- продукционные модели;
- семантические сети;
- фреймы;
- формальные логические модели.

11.3.1. Продукционная модель

Продукционная модель – это модель, основанная на правилах, которая позволяет представить знания в виде предложений типа:

Если *(условие)*, **То** *(действие)*

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

Знания в продукционных моделях представляются в следующей форме: Если A То B. Вместо A и B могут стоять некоторые утверждения, факты, приказы и т. д.

Примеры:

– Если диагонали четырехугольника пересекаются под прямым углом, То этот четырехугольник ромб;

– Если надо работать с текстом, То загрузите текстовый процессор.

Из примеров видно, что правило состоит из двух частей: *посылки (условия)* и *следствия (заключения)*.

Если A*(посылка)* имеет место, **То B** *(следствие)* также реализуется или может быть реализовано

```
Если все посылки от A1 до AN истинны, То следствие B также истинно
```

Посылки A1 . . . AN есть простые посылки, они соединяются с помощью союзов: и, или и могут содержать отрицание не. При реализации правил такого вида из одной или нескольких посылок (знаний) могут быть получены новые знания, поэтому они называются *продукционными*.

Пример. Если ПК включен и операционная система загружена, То можно приступать к работе на ПК.

Условия называют еще фактами. С помощью фактов описывается текущее состояние предметной области. Действие B трактуется как добавление нового факта. При использовании продукционной модели исходные понятия и факты хранятся в базе фактов, а набор правил и алгоритмов хранится в базе знаний. На основе базы фактов запускается машина вывода, которая выполняет перебор правил из базы знаний.

К преимуществам продукционных систем относят следующие:

1. *Модульность*. Системы устроены так, что каждая единица информации может быть удалена, изменена или добавлена независимо от всех остальных. Знания вводятся неупорядоченно, как в словаре или энциклопедии. Опыт показывает, что это является естественным способом пополнения своих знаний для эксперта. На практике ряд американских авторов (Дэвис, Ньюэлл, Куинлан, Стэфик) нарушают это правило, т. е. проблема реорганизации решается у них довольно слабо.

2. *Модифицируемость*. Если добавляется или модифицируется какое-либо правило, то все, что было уже сделано ранее, остается в силе и к новому правилу не относится. В отличие от этого в процедурных программах подобные взаимодействия ограничены, любая модификация является не существенной и может привести к непредсказуемым последствиям.

3. *Доступность чтения*. Наше собственное знание является модульным, поэтому продукционные системы нам кажутся более близкими и более легкими для чтения. Подобные системы в первую очередь предназначены для врачей, биологов, архитекторов и т. д. Во многих областях реально существуют добавляемые знания и соответствующие базы правил, например, автоматизированные экспертные обучающие системы.

Для машины более простой проверке поддается база знаний, представленная в форме правил, а не в форме процедур. Противоречия или избыточность в действительности легко определяются с помощью синтаксической обработки.

Способность к самообъяснению. Это свойство связано одновременно и с правилами, и с их структурами внешнего управления. Система легко прослеживает цепочку правил, которую она использовала для получения вывода, так же как и использовавшиеся метаправила.

Однородное представление знания, определяемое установленным форматом, разрешает создавать модели правил, которые позволяют получить ответы на некоторые вопросы и предусмотреть большую часть изменений, в случае обнаружения ошибки в базе данных.

Эффективность. Практика доказала гибкость и компетентность таких систем. Например, системы Mycin, Prospector и R1 являются уже не опытными образцами, а высокопроизводительными системами. Они оказались достаточно эффективными и выдерживают сравнение с процедурными системами во многих областях. Их структура управления позволяет принимать в расчет многочисленные параметры, характеризующие ситуацию. Причина эффективности продукционных правил заключается в том, что эти правила учитывают конкретные данные в каждом случае.

Недостатки продукционных систем. Среди них выделяют три главных, с которыми связаны определенные ограничения, лежащие в основе используемого формализма. Они относятся к концепции, формулировке и использованию правил.

1. *Трудность составления продукционного правила,* соответствующего элементу знания. В этом случае нужно, чтобы рассматриваемая область уже была достаточно изучена и установлены хорошие примитивы, и чтобы уровень детализации не был излишне подробным, иначе потребуется иметь по одному правилу на каждую ситуацию и человеку трудно разобраться в таком обилии информации.

2. *Трудность записи правила.* Единый формат записи `Если...То` приводит к громоздким выражениям и повторению тех же посылок в схожих ситуациях, с его помощью трудно выразить сложные правила. Но иногда жесткий синтаксис и громоздкие записи представляют определенные преимущества. Во всех случаях проблема «поймать знания» остается главной при написании экспертной системы.

3. *Трудности использования* связаны не с самими правилами, а с единой системой их связи. При этом база фактов играет роль кратковременной памяти, которая реально смоделирована, например, в разработанной Ньюэллом (Newell, 1975) системе PSC – познавательной психологии. В ней запрещен взаимный прямой вызов одного правила из другого, и она неудобна для выполнения алгоритмов в обычном смысле слова. Однако именно благодаря этой базе система в каждый момент времени точно знает, что она делает и что она знает.

Имеется много программных средств, реализующих продукционный подход:

– язык OPS 5;

– оболочки, или «пустые» экспертные системы – это Exsys Professional, КАРРА, Эксперт, ЭКО;

– инструментальные системы ПИЭС и СПИЭС и др., а также промышленных системы на его основе, например, системы, созданные средствами G2) и др.

Продукционная модель чаще всего используется в промышленных экспертных системах. Она отличается наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

11.3.2. Семантические сети

Семантическая модель представления знаний была предложена американским психологом Россом Квиллианом. *Семантика* – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая *смысл знаков*. Термин «семантическая» означает «смысловая», а *семантическая сеть* представляет собой *ориентированный граф*, вершины которого есть *понятия*, а дуги – *отношения* между ними. В качестве понятий обычно выступают *абстрактные или конкретные объекты*, а отношения представляют собой связи типа: *АКО-связи* (A-Kind-Of = это), *«имеет частью»* («has part»), *«принадлежит»* (рис. 13.1).

В основе сетевых моделей представления знаний лежит идея о том, что любые знания можно представить в виде совокупности *объектов* (понятий) и *связей* (отношений) между ними. В отличие от продукционных эти модели более наглядны, поскольку любой пример можно представить в виде ориентированного (направленного) графа.

Наиболее часто в семантических сетях используются следующие отношения:

– *связи типа «часть – целое»* («класс – подкласс», «элемент – множество» и т. п.);

– *функциональные связи* (определяемые обычно глаголами «производит», «влияет»...);

– *количественные* (больше, меньше, равно...);

– *пространственные* (далеко от..., близко от..., за..., под..., над...);

– *временные* (раньше, позже, в течение...);

– *атрибутивные связи* (иметь свойство, иметь значение);

– *логические связи* (И, Или, Не);

– *лингвистические связи* и др.

Однако обычно обязательным является наличие трех типов отношений:

- *класс – элемент класса* (цветок – «роза»);
- *свойство – значение* (цвет – желтый);
- *пример элемента класса* (роза – чайная).

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

Известно несколько подходов к классификации семантических сетей, связанных с типами отношений между понятиями. Так, по типам отношений выделяют:

- *бинарные*, в которых отношения связывают два объекта;
- *N-арные*, в которых есть специальные отношения, связывающие более двух понятий.

По количеству типов отношений выделяют семантические сети:

- *однородные* (с единственным типом отношений);
- *неоднородные* (с различными типами отношений).

Основным достоинством этой модели является то, что она более других соответствует современным представлениям об организации долговременной памяти человека. *Недостаток* – это сложность организации процедуры поиска вывода на семантической сети.

Изображения и графы эффективно используются для доказательства теорем. Они оказывают большую помощь в проверке отсутствия зацикливания этапов, в полном и эффективном хранении задачи в памяти, в организации ввода новых элементов.

Для реализации семантических сетей существуют специальные сетевые языки, например *Net*, язык реализации систем *Simer+Mir* и др. Известны экспертные системы, использующие семантические сети в качестве языка представления знаний: *Prospector*, *Casnet*, *Torus*.

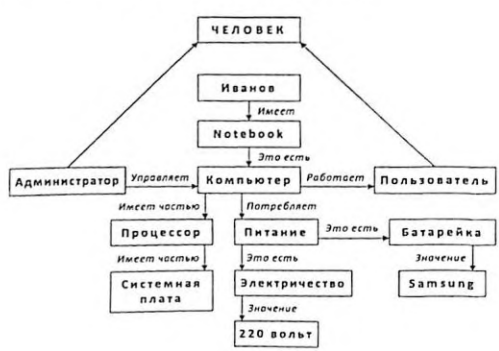


Рис. 13.1. Семантическая сеть

11.3.3. Фреймы

Термин *фрейм* (от англ. *frame*, что означает «каркас», или «рамка») был предложен в 70-е годы XX-го века Марвином Минским, одним из родоначальников систем искусственного интеллекта – для обозначения структуры знаний и восприятия пространственных сцен. Эта модель имеет психологическое обоснование.

Фрейм – это абстрактный образ для представления некоего стереотипа восприятия. Фреймы представляют собой сложные структуры данных, описывающих какую-либо типичную ситуацию, например экзамен, лекция, раунд переговоров. Фрейм состоит из *позиций* для размещения объектов, характеризующих данную ситуацию. Позиция может быть передана другому фрейму. Кроме того, он содержит информацию о выполняемых действиях, о том, как следует поступать в типичных и нетипичных случаях, о влиянии соседних фреймов.

В психологии и философии известно понятие абстрактного образа.

Пример. Произнесение вслух слова «лекция» порождает образ лекции: аудитория, лектор, слушатели. Из этого описания ничего нельзя убрать, например, убрав лектора, мы получим уже собрание студентов, а не лекцию. Но в нем есть «дырки», называемые *слотами*, – это незаполненные значения некоторых атрибутов, например, количество студентов, ФИО лектора и др. В теории фреймов такой образ лекции называется *фреймом лекции*. Фреймом также называется и формализованная модель для отображения образа.

Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний, используя:

- *фреймы-структуры*, использующиеся для обозначения объектов и понятий (заем, залог, вексель);
- *фреймы-роли* (менеджер, кассир, клиент);
- *фреймы-сценарии* (банкротство, собрание акционеров, празднование именин) и др.

Различают *фреймы-образцы (прототипы)*, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных ситуаций на основе поступающих данных. Рассмотрим пример (табл. 13.1):

Таблица 13.1

Продукты

Продукт	Код	Количество
Мед	111	10
Соль	222	20

Фрейм-образец:
Имя: Продукты
Продукт: (Значение слота 1)
Код: (Значение слота 2)
Количество: (Значение слота 3)
Фрейм-экземпляр:
Имя: Продукты
Продукт: (Мед-Соль)
Код: (111-222)
Количество: (10-20)

Фрейм-образец содержит набор *атрибутов*, называемых *слотами*. В *фрейме-экземпляре* слотам присваиваются *конкретные значения*. Традиционно структура фрейма может быть представлена как список свойств:

Имя фрейма:
(Имя 1-го слота: значение 1-го слота),
(Имя 2-го слота: значение 2-го слота), ...
(Имя N-го слота: значение N-го слота).

Эту же запись можно представить в виде таблицы (табл. 13.2).

Таблица 13.2

Структура фрейма

Имя фрейма			
Имя слота	Значение слота	Способ присоединения получения значения	Присоединенная процедура

В таблице 11.2 дополнительные два столбца (справа) предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур, что допускается в теории фреймов.

Существует несколько способов получения слотом значений во фрейме-экземпляре:

– по умолчанию от фрейма-образца (Default-значение);
– через наследование свойств от фрейма, указанного в слоте АКО (A-Kind-Of = это);
– по формуле, указанной в слоте;
– через присоединенную процедуру;
– явно из диалога с пользователем;
– из базы данных.

В качестве значения слота может выступать имя другого фрейма – тогда образуется *сеть фреймов*. В сетях фреймов происходит наследование свойств по *АКО-связям*.

Важнейшей характеристикой теории фреймов является *наследование свойств*, заимствование из теории семантических сетей. И во фреймах, и в семантических сетях наследование происходит по АКО-связям. Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

Основным преимуществом фреймов как модели представления знаний является ее гибкость и наглядность, а также то, что она отражает концептуальную основу организации памяти человека.

Разработаны специальные языки представления знаний в сетях фреймов:

– FRL (Frame Representation Language);

– Фреймовая «оболочка» KAPPA и другие программные средства, которые позволяют эффективно строить промышленные экспертные системы.

Широко известны такие фрейм-ориентированные экспертные системы, как: *Analyst*, *Модис*, *Tristan*, *Alterid*.

11.3.4. Формальные логические модели

Формальная логика – наука об общих структурах и законах правильного мышления, образования и сочетания понятий и высказываний, о правилах умозаключений независимо от их конкретного содержания.

Система искусственного интеллекта в определенном смысле моделирует интеллектуальную деятельность человека и, в частности, логику его рассуждений. Логические построения при этом сводятся к следующей схеме: из одной или нескольких посылок (которые считаются истинными) следует сделать «логически верное заключение» (вывод, следствие). Для этого необходимо, чтобы и посылки, и заключение были представлены на понятном языке, адекватно отображающем предметную область, в которой проводится вывод.

В обычной жизни это наш естественный язык общения. В математике, например, это язык определенных формул. Наличие же языка предполагает, во-первых, наличие алфавита (словаря), отображающего в символьной форме весь набор базовых понятий (элементов), с которыми придется иметь дело и, во-вторых, набор синтаксических правил, на основе которых, пользуясь алфавитом, можно построить определенные выражения.

Логические выражения, построенные в данном языке, могут быть *истинными* или *ложными*. Некоторые из этих выражений, являющиеся всегда истинными, объявляются *аксиомами* (или *постулатами*). Они составляют

ту базовую систему посылок, исходя из которой и пользуясь определенными правилами вывода, можно получить заключения в виде новых выражений, также являющихся истинными.

Если перечисленные условия выполняются, то говорят, что система удовлетворяет требованиям *формальной теории* и такую систему называют *формальной* или *аксиоматической*.

Всякая формальная теория, определяющая некоторую *аксиоматическую систему*, характеризуются:

- наличием алфавита;
- множеством синтаксических правил;
- множеством аксиом, лежащих в основе теории;
- множеством правил вывода.

Примеры *аксиоматических систем*:

- исчисление высказываний;
- исчисление предикатов.

Логика (исчисление) высказываний – это раздел математической логики, лежащий в основе всех ее разделов. Основными объектами в ней являются *высказывания* – повествовательные предложения, о каждом из которых можно сказать одно из двух: *истинно* оно или *ложно*.

Пример. Студент сдает экзамен. Истина обозначается символом 1, ложь – 0.

Логика предикатов начинается с анализа строения высказываний, которые выражают тот факт, что объекты обладают некоторыми *свойствами* или находятся между собой в некоторых *отношениях*. Понятие свойства и отношения рассматриваются как частный случай понятия предиката: соответственно *предикат-свойство* и *предикат-отношение*.

Предикатам могут быть приписаны *кванторы*. В естественном языке имеется большое количество кванторов: «каждому», «для всех» и др. С помощью кванторов создается *предикатная модель базы знаний* и делаются *логические выводы*.

В представлении знаний традиционно выделяют *формальные логические модели*, основанные на классическом исчислении предикатов 1-го порядка, когда предметная область или задача описывается в виде набора аксиом.

Достоинство аксиоматических систем в том, что они хорошо исследованы, а *недостаток* – это их закрытость и негибкость: модификация и расширение здесь всегда связаны с перестройкой всей системы.

Формальная логическая модель предъявляет высокие требования и ограничения к предметной области, поэтому применима в основном в исследовательских системах. Понять эти модели, не имея специальной подготовки,

достаточно сложно. Описание формальных логических моделей в промышленных экспертных системах практически не используется. Чаще используются различные ее модификации и расширения.

Словарь терминов

Данные – представляют собой отдельные факты, характеризующие объекты, процессы и явления, а также их свойства.

Знания – закономерности предметной области: принципы, связи, законы, полученные в результате практической деятельности, профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

Продукционная модель – это модель, основанная на правилах, которая позволяет представить знания в виде предложений типа `Если (условие), То (действие).`

Семантика – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков.

Семантическая сеть – ориентированный граф, вершины которого есть понятия, а дуги – отношения между ними.

Формальная логика – наука об общих структурах и законах правильного мышления, образования и сочетания понятий и высказываний, о правилах умозаключений независимо от их конкретного содержания.

Фрейм – абстрактный образ для представления некоего стереотипа восприятия.

Вопросы и задания для самоконтроля

1. Какие типы задач решаются в настоящее время при помощи ЭВМ?
2. Что собой представляют данные?
3. Что собой представляют знания?
4. Чем отличаются знания от данных?
5. Проведите классификацию знаний по глубине.
6. Проведите классификацию знаний по выполняемым функциям.
7. Проведите альтернативную классификацию знаний.
8. Что собой представляют модели представления знаний?
9. Какие требования учитываются при разработке конкретной модели представления знаний?
10. Что собой представляет продукционная модель, каковы ее преимущества и недостатки?
11. Что собой представляет семантическая сеть?
12. Что такое фрейм?
13. Какие модели фреймов могут использоваться?
14. Что собой представляет формальная логическая модель?

ВОПРОСЫ И ЗАДАНИЯ К ЭКЗАМЕНУ

Тема «Экономическая информация в автоматизированных информационных системах»

1. Экономическая информация, ее виды и особенности.
2. Структурные единицы экономической информации.
3. Экономические информационные системы, их классификация, информационное обеспечение.
4. Автоматизированные информационные системы.
5. Внемашинная организация экономической информации.
6. Классификация информации и кодирование информации.
7. Файловая организация информации и ее недостатки.
8. Понятие базы данных. Требования, предъявляемые к базе данных.
9. Приложения и компоненты базы данных.
10. Сверхбольшие базы данных.

Тема «Модели данных»

1. Понятие модели данных.
2. Иерархическая и сетевая модели данных.
3. Реляционная модель данных. Достоинства и недостатки реляционной модели данных.
4. Логические связи между отношениями. Контроль целостности связей.
5. Постреляционная модель данных.
6. Объектно-ориентированная и объектно-реляционная модели данных.
7. Многомерная модель данных.

Тема «Проектирование базы данных»

1. Жизненный цикл базы данных. Этапы жизненного цикла БД.
2. Модель «сущность-связь».
3. Нормализация.
4. Семантическая объектная модель.
5. Этапы проектирования базы данных и их процедуры.
6. CASE-средства для автоматизированного проектирования реляционных баз данных.

7. Общая характеристика и функциональные возможности CASE-системы ERwin.

8. Понятие СУБД. Языковые средства СУБД.

9. Функциональные возможности СУБД.

10. Архитектура СУБД.

11. Классификация СУБД.

12. Режимы работы пользователя с СУБД.

Темы «СУБД Microsoft Office Access 2013» и «Технологии работы с базой данных в СУБД Microsoft Office Access 2013»

1. Общая характеристика Microsoft Access.

2. Объекты БД и их размещение.

3. Пользовательский интерфейс и основные настройки Microsoft Access 2013.

4. Данные в Microsoft Access.

5. Выражения в Microsoft Access.

6. Инструментальные средства создания объектов БД.

7. Создание и корректировка БД в СУБД Access.

8. Создание таблиц: в режиме таблицы, в режиме конструктора.

9. Работа с макетом таблицы.

10. Создание схемы данных.

11. Возможности, типы и средства создания запросов.

12. Создание запросов выбора различных видов.

13. Создание запросов действия.

14. Создание перекрестных запросов.

15. Назначение и способы проектирования форм различных видов. Редактирование существующих форм.

16. Назначение и способы проектирования отчетов различных видов.

17. Инструментальные средства конструкторов форм и отчетов.

18. Элементы управления форм и отчетов и работа с ними.

19. Создание диаграмм в формах и отчетах.

20. Вычисления в запросах, формах и отчетах.

21. Понятие макроса и модуля.

22. Классификация макрокоманд. Типы макросов.

23. Конструирование макроса.

24. Формирование макрокоманд в окне макроса и с помощью мыши.

25. Использование в макросах ссылок на объекты и их элементы управления.

26. Ссылки на свойство объекта и элемента управления.
27. Ссылка на подчиненную форму или отчет.
28. Создание ссылок в построителе выражений.
29. Вложенные макросы.
30. Сохранение, запуск и отладка созданного макроса.

Тема «Введение в язык SQL»

1. Язык SQL: общая характеристика, функциональные возможности.
2. Достоинства и недостатки SQL.
3. Операторы, данные и выражения в SQL.
4. Формирование запросов на языке SQL, структура команды SQL.
5. Синтаксис команд CREATE TABLE, INSERT, UPDATE, DELETE, SELECT.
6. Создание запроса на языке SQL в СУБД Access.
7. Диалекты языка SQL в СУБД.

Тема «Системы обработки многопользовательских баз данных»

1. Эволюция концепций обработки данных.
2. Архитектура файл-сервер и роль настольных СУБД в ней.
3. Клиент-серверные системы.
4. Хранимые процедуры и триггеры.
5. Достоинства архитектуры «клиент-сервер».
6. Характеристика серверов баз данных.
7. Механизмы доступа к базам данных.
8. Понятие и архитектура распределенной БД.
9. Распределенная СУБД.
10. Гомогенные и гетерогенные распределенные БД.
11. Двенадцать правил К. Дейта для РаБД и РаСУБД.
12. Обработка распределенных запросов.
13. Достоинства и недостатки РаСУБД.

Тема «Администрирование баз данных»

1. Пользователи БД; администратор БД, его функции.
2. Разработчики и администраторы приложений.
3. Методы защиты баз данных Microsoft Access: шифрование, разграничение прав доступа.

4. Правовая охрана баз данных.

5. Шифрование базы данных. Защита БД на уровне пользователя.

6. Восстановление БД. Сжатие БД. Резервное копирование базы данных в Microsoft Access.

7. Разделение базы данных в Microsoft Access.

Тема «Хранилища данных»

1. Понятие хранилища данных.

2. Технология оперативной аналитической обработки данных OLAP.

3. Отличия хранилища данных от базы данных. Классификация и технологические решения ХД.

4. Виды хранилищ данных: финансовые, в области страхования, для управления людскими ресурсами, глобальные, с возможностями Data Mining/Data Mining и Exploration.

Тема «Базы знаний и модели представления знаний»

1. Данные и знания. Основные понятия.

2. Классификация знаний: по глубине, по выполняемым функциям, альтернативная.

3. Модели представления знаний, общая характеристика.

4. Продукционная модель представления знаний, ее достоинства и недостатки.

5. Семантические сети.

6. Фреймы.

7. Формальные логические модели представления знаний.

НЕКОТОРЫЕ ПРИМЕРЫ ТЕСТОВЫХ ЭКЗАМЕНАЦИОННЫХ ЗАДАНИЙ

1. Совокупность различных сведений экономического характера, используемых для планирования, учета, контроля, анализа и управления народным хозяйством и его звеньями – это...

- информационная система;
- экономическая информация;
- экономическая информационная система;
- информационная база.

2. Совокупность документов, объединенных по определенному признаку, – это...

- массив;
- набор;
- коллекция;
- подборка.

3. Что включает в себя информационное обеспечение автоматизированной информационной системы?

- реквизиты;
- системы классификации и кодирования информации;
- информационную надстройку;
- документацию для отражения показателей;
- информационную базу;
- системы показателей, описывающих деятельность экономического объекта.

4. Иерархическая модель данных представляет собой…

- набор отношений, изменяющихся во времени;
- структуру, у которой один или несколько порожденных узлов имеют более одного исходного узла;
- перевернутое дерево, из корня и узлов которого исходят ветви.

5. Форма представления «Описание свойств объекта» соответствует элементу реляционной модели данных…

- кортеж;
- тип данных;

– сущность;
– атрибут.

6. Что является средством моделирования предметной области на этапе концептуального проектирования?
– нормализация отношений;
– модель «сущность-связь»;
– объектно-ориентированное программирование;
– продукционная модель.

7. На ER-диаграмме между двумя сущностями установлена связь типа 1:1, класс принадлежности обеих сущностей является необязательным. Сколько таблиц необходимо создать при преобразовании такой ER-модели в реляционную модель?
– одну;
– две;
– три;
– четыре.

8. В соответствии с поддерживаемым типом модели данных СУБД делятся на…
– иерархические, сетевые, реляционные, постреляционные, многомерные и объектно-ориентированные;
– иерархические, сетевые, реляционные, постреляционные, многомерные и гибридные;
– иерархические, сетевые, реляционные;
– многомерные и объектно-ориентированные.

9. Имя поля в Microsoft Access содержит…
– не менее 64 символов;
– не более 64 символов;
– не менее 8 символов;
– не более 8 символов.

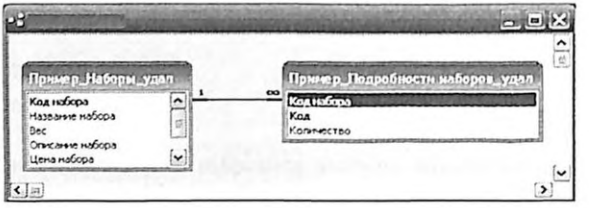
10. ^, &, OR, %, < > – это…
– константы;
– функции;

– литералы;
– операторы;
– идентификаторы.

11. Какая инструкция SQL используется для создания таблицы базы данных?

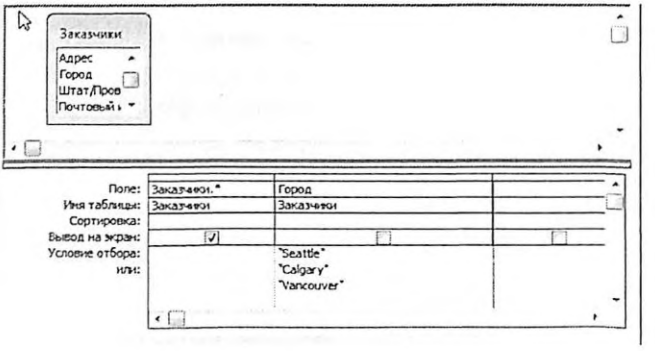
– `CREATE TABLE;`
– `DROP TABLE;`
– `NEW TABLE;`
– `BUILD TABLE.`

12. Что изображено на рисунке?



– окно «База данных»;
– окно конструктора таблиц;
– окно «Схема данных»;
– окно формы.

13. Что произойдет при выполнении запроса, изображенного на рисунке в режиме конструктора?

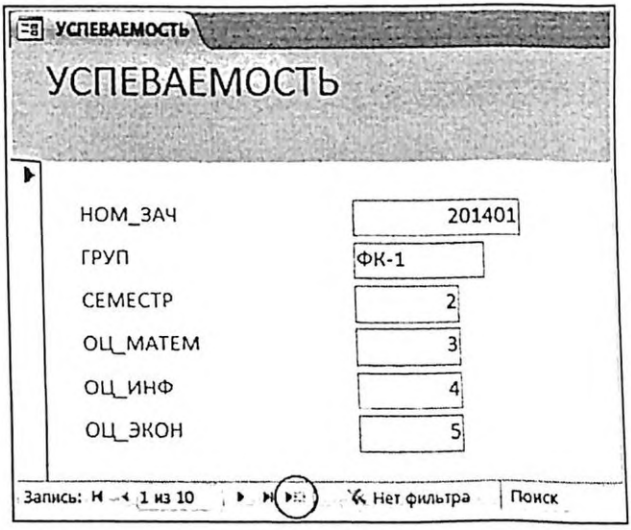


– из БД будут отобраны записи, содержащие в поле «Город» значение «Seattle», и «Calgary», и «Vancouver»;
– из БД будут отобраны записи, содержащие в поле «Город» значение «Seattle», или «Calgary», но не «Vancouver»;
– из БД будут отобраны записи, содержащие в поле «Город» значение «Seattle», или «Calgary», или «Vancouver»;
– из БД будут отобраны записи, содержащие в поле «Город» значение «Seattle», но не «Calgary» или «Vancouver».

14. Какое условие отбора нужно задать, чтобы выбрать все товары с ценой не менее 350 руб.?

– <=350;
– =<350;
– >=350;
– =>350.

15. Для чего используется в форме выделенная на рисунке кнопка?

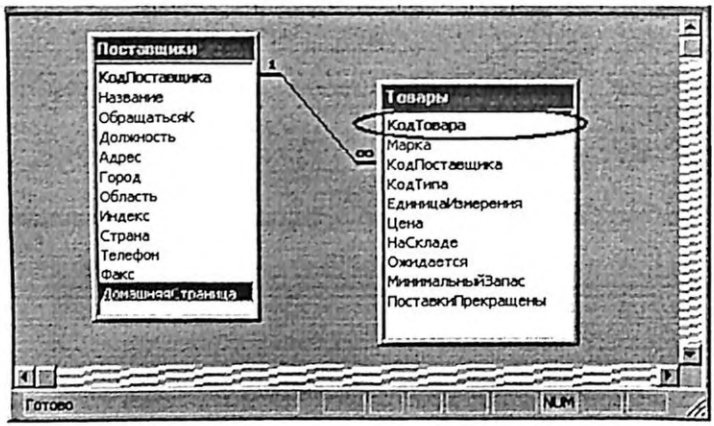


– для удаления записи;
– для перехода на последнюю запись;
– для обновления формы;
– для добавления новой записи.

16. Выберите верное утверждение:

– все таблицы всех баз данных Microsoft Access хранятся в одном файле;
– все таблицы одной базы данных Microsoft Access хранятся в одном файле;
– каждая таблица одной базы данных Microsoft Access обычно хранится в отдельном файле.

17. Почему в окне «Схема данных» поле «КодТовара» в таблице «Товары» помечено полужирным шрифтом?



– поле «Код товара» – первичный ключ;
– поле «Код товара» – внешний ключ;
– поле «Код товара» – составной индекс;
– поле «Код товара» – первое по счету поле в таблице «Товары»

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная

1. Базы данных : учебник / А. Д. Хомоненко [и др.] ; под ред. А. Д. Хомоненко. – СПб. : КОРОНА Принт, 2006. – 736 с.

2. Болотова, Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях : учебник / Л. С. Болотова. – М. : Финансы и статистика, 2012. – 663 с.

3. Малыхина, М. П. Базы данных : учеб. пособие / М. П. Малыхина. – СПб. : БХВ-Петербург, 2004. – 512 с.

4. Оскерко, В. С. Технологии баз данных : учеб. пособие / В. С. Оскерко, З. В. Пунчик, О. А. Сосновский. – Минск : БГЭУ, 2007. – 171 с.

5. Паклин, Н. Б. Бизнес-аналитика: от данных к знаниям : учеб. пособие / Н. Б. Паклин, В. И. Орешков [и др.]. – 2-е изд. – СПб. : Питер, 2010. – 701 с.

Дополнительная

1. Автоматизированные информационные технологии в экономике : учебник / В. В. Брага, Н. Г. Бубнова, Л. А. Вдовенко [и др.]; под ред. Г. А. Титоренко. – М. : ЮНИТИ, 2002. – 399 с.

2. Грофф, Дж. Р. Энциклопедия SQL / Дж. Р. Грофф, П. Н. Вайнберг. – СПб. : Питер, 2003. – 896 с.

3. Дунаев, В. В. Базы данных. Язык SQL / В. В. Дунаев. – СПб. : БХВ-Петербург, 2006. – 288 с.

4. Козадаев, К. В. Организация баз данных и экспертных систем : курс лекций / К. В. Козадаев. – Минск : БГУ, 2012. – 198 с.

5. Кренке, К. Теория и практика построения баз данных / К. Кренке. – СПб. : Питер, 2003. – 800 с.

6. Оскерко, В. С. Компьютерные информационные технологии : в 3 ч. / В. С. Оскерко, З. В. Пунчик. – Минск : БГЭУ, 2011. – Ч. 2 : Базы данных и знаний : учеб. пособие.– 226 с.

7. Роб, П. Системы баз данных: проектирование, реализация и управление / П. Роб, К. Карлос. – СПб. : БХВ-Петербург, 2004. – 1024 с.

8. Сидоркина, И. Г. Системы искусственного интеллекта : учеб. пособие. – М. : КноРус, 2014. – 245 с.

9. Советов, Б. Я. Представление знаний в информационных системах : учебник / Б. Я. Советов. – 2-е изд. – М. : Академия, 2012. – 141 с.

10. Стоцкий, Ю. Microsoft Office 2010: самоучитель. / Ю. Стоцкий [и др.]. – СПб. : Питер, 2011. – 425 с.

11. Туманов, В. Е. Проектирование реляционных хранилищ данных / В. Е. Туманов, С. В. Маклаков. – М. : Диалог-МИФИ, 2007. – 333 с.

12. Харрингтон, Дж. Л. Проектирование реляционных баз данных : пер. с англ. / Дж. Л. Харрингтон. – М. : Лори, 2006. – 230 с.

Учебное издание

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методический комплекс
для студентов экономических специальностей

В трех частях

Часть 3

РЯСОВА Стелла Евгеньевна

Технологии баз данных и знаний

В двух книгах

Книга первая

Редактор *Д. В. Палынская*
Дизайн обложки *М. С. Мухоморова*

Подписано в печать 07.12.2017. Формат 60×84 1/16. Бумага офсетная.
Ризография. Усл. печ. л. 14,85. Уч.-изд. л. 12,75 Тираж 30 экз. Заказ 1640

Издатель и полиграфическое исполнение:
учреждение образования «Полоцкий государственный университет».
Свидетельство о государственной регистрации
издателя, изготовителя, распространителя печатных изданий
№ 1/305 от 22.04.2014.

ЛП № 02330/278 от 08.05.14.

Ул. Блохина, 29, 211440, г. Новополоцк.

004
Р 99
1214011232291 НБ УО "ПГУ