Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Т.П. Злыднева

БАЗЫ ДАННЫХ:
основы теории и проектирования

*Утверждено Редакционно-издательским советом университета
в качестве учебного пособия*

Издание 2-е

Магнитогорск
2021

УДК 004.43(075.8)

Рецензенты:
Кандидат технических наук,
доцент кафедры вычислительной техники и программирования
ФГБОУ ВО «Магнитогорский государственный технический
университет им. Г.И. Носова»
Л.Г. Егорова

Кандидат педагогических наук,
и.о. зав. кафедрой прикладной математики и информационных
технологий Сибайского института (филиала)
ФГБОУ ВО «Башкирский государственный университет»
И.С. Гумеров

Злыднева Т.П.
Базы данных: основы теории и проектирования [Электронный ресурс] : учебное пособие / Татьяна Павловна Злыднева ; ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова». – Изд. 2-е, подгот. по печ. изд. 2019 г. – Электрон. текстовые дан. (1,59 Мб). – Магнитогорск : ФГБОУ ВО «МГТУ им. Г.И. Носова», 2021. – 1 электрон. опт. диск (CD-R). – Систем. требования : IBM PC, любой, более l GHz ; 512 Мб RAM ; 10 Мб HDD ; MS Windows XP и выше ; Adobe Reader 8.0 и выше ; CD/DVD-ROM дисковод ; мышь. – Загл. с титул. экрана.
ISBN 978-5-9967-2082-8

В учебном пособии раскрываются основные теоретические аспекты по базам данных, методы их проектирования, функциональные возможности систем управления базами данных, основы языка SQL. В работу включен практикум, способствующий получению навыков проектирования баз данных. Учебное пособие разработано в соответствии с программой дисциплины «Базы данных», может быть использовано при прохождении учебной практики – практики по получению первичных профессиональных умений и навыков.

Предназначено для обучающихся по направлению 01.03.02 «Прикладная математика и информатика», может быть полезно студентам других направлений подготовки, интересующимся вопросами проектирования и использования баз данных.

УДК 004.43(075.8)

ISBN 978-5-9967-2082-8

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Главные идеи современных информационных технологий базируются на концепции баз данных, согласно которой основой информационных технологий являются данные, определенным образом организованные в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Изучение баз данных – объективная необходимость для формирования профессиональной компетентности обучающихся в технических вузах.

Учебное пособие «Базы данных: основы теории и проектирования» предназначено для студентов направления «Прикладная математика и информатика». В учебном плане данного направления подготовки существует дисциплина «Базы данных», изучая которую, студенты приобретают новые профессиональные знания, используя современные образовательные и информационные технологии, и овладевают практическими навыками разработки прикладных баз данных.

Раскрытию основных понятий «информация» и «данные», способов обработки данных, а также вопросам организации баз данных посвящена первая глава учебного пособия. Далее рассматриваются основные методы проектирования баз данных. На конкретных примерах, в доступной форме излагается метод декомпозиции отношений и метод «сущность-связь».

Отдельная глава учебного пособия посвящена системам управления базами данных, их архитектуре и функциональным возможностям. Создание основных объектов базы данных – таблиц, запросов, форм, отчетов, макросов и модулей, рассмотрено на примере СУБД Access. В данной главе помимо теоретического материала представлен лабораторный практикум, задача которого – обеспечить студентов конкретным инструментарием для практической реализации методов проектирования баз данных в среде Microsoft Access.

В настоящее время технологии баз данных существенно развиваются. В частности, широко используются информационные системы с архитектурой «клиент-сервер», основой которых являются SQL-ориентированные СУБД. Они стали популярны благодаря высокой производительности, надежности, безопасности, большой информационной емкости. В последней главе раскрываются теоретические основы языка SQL, демонстрируются его возможности, в частности создание запроса на выборку и использование агрегатных (статических) функций. Овладеть практическими навыками разработки баз данных в SQL-ориентированных СУБД обучающимся позволяет выполнение лабораторных работ в СУБД Microsoft SQL Server.

Теоретический материал, излагаемый в данном учебном пособии, составляет лишь часть информации, необходимой для освоения, тот минимум, который нужен для формирования у обучающихся профессиональных компетенций. В список литературы (см. Библиографический список) включены основные публикации, с помощью которых студент может осваивать курс самостоятельно. Фактически все рекомендуемые издания снабжены библиографическими указателями, использование которых позволяет глубже изучить материал. Некоторые рекомендуемые издания, приведенные в списке, существуют в электронном виде.

Учебное пособие «Базы данных: основы теории и проектирования» нацелено на освоение курса «Базы данных» с использованием проблемного обучения и организации учебно-исследовательской деятельности в рамках данной дисциплины. Проблемный подход и организация исследовательской деятельности обучающихся в процессе профессиональной подготовки рассматривается нами во многих предыдущих публикациях [22; 26; 28; 29].

Автором разработана собственная методика организации учебно-исследовательской деятельности студентов направления «Прикладная математика и информатика» на конкретном предметном содержании. Проблемный подход, представленный в ней, используется при изучении дисциплин: «Основы информатики», «Системное и прикладное программное обеспечение» [25], «Практикум на ЭВМ» [23; 33; 20], «Операционные системы» [11]. Для магистрантов данного направления подготовки нами разработано учебное пособие в двух частях для изучения дисциплины «История и методология прикладной математики и информатики» [13; 14], а также учебное пособие «Математика и информатика: от истоков до современности» [16]. В них также предлагается использовать элементы учебно-исследовательской деятельности.

Учебное пособие «Базы данных: основы теории и проектирования» должно стать важной составляющей в методике организации учебно-исследовательской деятельности студентов на конкретном содержании цикла дисциплин по информационным технологиям. Проектирование и разработку базы данных можно назвать творческим процессом, ведь понимание пользовательских требований и перевод их в эффективный проект базы данных требуют и искусства, и умения. Поскольку в настоящее время существует большая необходимость в развитии технологии баз данных, знания, полученные в процессе изучения этого курса, и приобретенные навыки, несомненно, будут востребованы.

1. БАЗЫ ДАННЫХ

1.1. Информация и данные: основные понятия

Информация – это понятие, подразумевающее знание определённых сведений, используемых в различных областях человеческой деятельности. На основе информации углубляются познания законов развития материального мира, взаимосвязываются и координируются работы, контролируются процессы и принимаются решения.

Информацией называются любые сведения о каких-либо явлениях, событиях, процессах, являющиеся объектами восприятия, передачи, преобразования, хранения и использования.

Информация – это ценнейший интеллектуальный ресурс в системе жизнеобеспечения общества, важнейшая часть его интеллектуальной собственности, доля которой все возрастает в современном мире.

Характерными чертами информации являются следующие:

- это наиболее важный ресурс современного производства – он снижает потребность в земле, труде, капитале, уменьшает расход сырья и энергии;
- постоянное накопление информации в увеличивающихся объемах;
- благодаря информации появляются новые производства;
- информация является товаром, причем продавец информации ее не теряет после продажи;
- информация придает дополнительную ценность другим ресурсам, в частности трудовым.

С информацией всегда связывают три понятия:

• источник информации – тот элемент окружающего мира (объект, процесс, явление, событие), сведения о котором являются объектом преобразования;

• потребитель информации – тот элемент окружающего мира, который использует информацию (для выработки поведения, принятия решения, управления или обучения);

• сигнал – материальный носитель, который фиксирует информацию для переноса ее от источника к потребителю.

Обычно потребителей информации интересует какой-то конкретный вопрос, область знаний или какая-то определённая совокупность объектов. В соответствии с этим, в области информационной деятельности введено понятие предметной области.

Предметной областью (ПО) называется определённая часть реального мира, представляющая интерес для конкретного исследования

или планируемых действий и, соответственно, для использования и отображения в информационной системе (в банке данных или знаний).

В процессе исследования объекта наблюдатель фиксирует состояние системы в определённой форме без выполнения над ним каких-либо операций.

Информация, фиксируемая в определённой форме и пригодная для последующей обработки, хранения и передачи, называется ***данными.***

Процесс восприятия состояния системы в виде данных, описывающих состояние системы, называется ***фиксацией данных.***

Информация, представляемая в виде зарегистрированных фактов, называется ***фактографической.***

При изучении предметной области в соответствии с понятиями «информация» и «данные» рассматривается два аспекта: инфологический и датологический.

Инфологический аспект предусматривает рассмотрение вопросов смыслового содержания информации независимо от способа формирования и организации данных в памяти ЭВМ.

На этом этапе осуществляется:

- описание вводимых в информационную систему понятий об объектах информации, их характеристиках, взаимосвязях;
- выявление объектов или явлений реального мира, информацию о которых требуется накапливать и обрабатывать;
- перечень основных учитываемых характеристик и их взаимосвязи.

Датологический аспект охватывает вопросы представления данных в памяти информационной системы.

На этом этапе:

- формулируются правила смысловой интерпретации данных;
- определяются формы представления информации посредствам данных в
  информационной системе;
- определяются модели и методы представления и преобразования данных.

Определение смыслового содержания зарегистрированных данных называется ***семантической информацией*** (или ***семантикой***). Она необходима для дальнейшего использования в производственных операциях. Основное средство представления семантики данных – это естественный язык. В общем случае, работа с семантикой – это работа со знаниями. Благодаря семантической информации машинные системы способны «понимать» задачу в формулировке пользователя, то есть реализуются «интеллектуальные» возможности или способности ЭВМ.

В информационных системах сложноорганизованные данные, содержащие одновременно как фактографическую, так и семантическую

информацию, необходимую пользователю для машинного преобразования исходных фактов в соответствии с определёнными правилами, то есть для работы с данными, называются ***знаниями.***

Наиболее совершенной и прогрессивной формой организации информации и знаний на ЭВМ являются банки данных и банки знаний. Главная их задача – обеспечение пользователей требуемой информацией.

Банк данных (БнД) – это автоматизированная система, включающая базу данных, лингвистические, программные, технические, организационно-методические средства, обеспечивающие централизованное накопление и многоцелевое использование информации в различных областях деятельности пользователей.

В банке данных содержатся совокупности фактов о качественных и количественных характеристиках конкретных объектов предметной области. С банками данных в процессе их создания и эксплуатации взаимодействуют пользователи различных категорий, основными из которых являются конечные пользователи. Ими являются специалисты предметных областей, для удовлетворения информационных потребностей которых и создаются банки данных. На рисунке 1 приведена структурная схема банка данных.

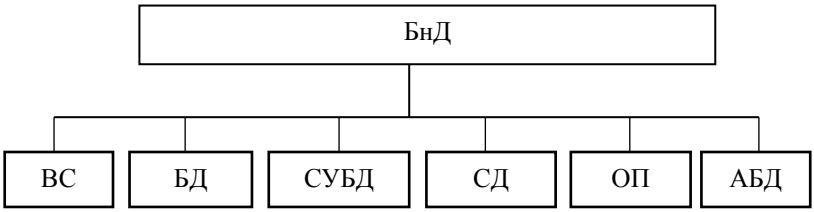


Рис.1. Структура банка данных

Здесь введены следующие обозначения:

ВС – вычислительная система, включающая технические средства (компьютер, устройство ввода-вывода), операционную систему, программные средства общего назначения и программы пользователей;

БД – база данных;

СУБД – система управления базами данных;

СД – словарь данных;

ОП – обслуживающий персонал, то есть группа сотрудников, обеспечивающих операции по сопровождению технических и программных средств БнД, по вводу и выводу текущей информации;

АБД – администратор базы данных.

Базы данных и системы управления базами данных – предмет дальнейшего изучения, более подробно их рассмотрим позднее.

Словарь данных (СД) – это специальная система для хранения единообразной и централизованной информации о всех ресурсах и данных. Он должен обеспечивать пользователей единой терминологией при обслуживании запросов по данной предметной области.

Словарь данных содержит информацию: об объектах, их свойствах и отношениях для данной ПО; о данных, хранимых в БД: наименованиях данных, их структурах; связях с другими данными; возможных значениях, форматах представления; источниках возникновения; кодах защиты и разграничения доступа к данным со стороны пользователей.

В банке данных в зависимости от типа СУБД могут использоваться два вида СД: словарь данных, *интегрированный* с СУБД; *независимый* СД.

В первом случае в СУБД имеются программные средства ведения словаря. Описание данных хранятся в СД в единственном экземпляре и используются при работе системы.Во втором случае для СУБД должен разрабатываться специальный пакет программ для ведения СД. Здесь имеет место избыточность описания данных (дублирование экземпляров данных в Базе данных).

Администратор базы данных (АБД) – это управляющий орган банка данных, состоящий из одного или группы специалистов в области теории систем обработки данных, знающих специфику предметной области данной информационной системы и реализующих управление БД посредствам СУБД.

1.2. Моделирование данных для представления в базе данных

База данных – это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

База данных – это поименованная совокупность данных, относящихся к некоторой области приложения, обладающая определенной структурой, компонентами которой могут быть любые структурные единицы данных (элементы, группы, записи, файлы), связанные между собой определенным образом. Причем эти данные могут использоваться как в одной, так и в нескольких задачах, одним или несколькими пользователями.

Существует множество других определений, отражающих скорее субъективное мнение тех или иных авторов о том, что означает база данных.

Наиболее часто используются следующие отличительные признаки баз данных:

- модельность – БД должна моделировать некоторую часть объектов реального мира;
- актуальность – БД должна отражать текущее состояние объектов реального мира и динамически обновляться в соответствии с изменениями в состоянии объектов;
- непротиворечивость – данные в БД не должны противоречить друг другу и выбранной модели предметной области.
- целостность – БД должна по возможности наиболее полно моделировать объекты реального мира в рамках выбранной предметной области;
- интегрированность – данные, хранящиеся в БД, должны быть направлены на решение общих задач, поставленных при ее разработке;
- надежность – данные должны быть защищены от потери либо искажения.

Центральным понятием в области баз данных является понятие модели.

Модель данных это некоторая абстракция, которая, будучи приложима к конкретным данным, позволяет пользователям и разработчикам трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними.

1.2.1. Информационное моделирование предметной области для базы данных

Информационные модели данных разрабатываются проектировщиками БД с целью обеспечения решения задач обработки данных, формулируемых конкретными пользователями. При информационном моделировании рассматривается: явления реального мира, информация об этих явлениях, представление этой информации посредствам данных. Схематично это отображено на рисунке 2.

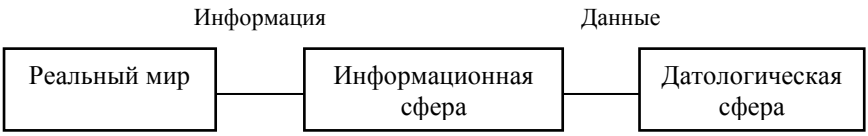


Рис. 2. Информационное моделирование

Для представления информации о конкретной предметной области необходимо:

- выделить в рассматриваемой предметной области реального мира объекты для
  информационного отображения в БД;
- для каждого объекта выявить свойства, достаточные для его описания;
- определить виды взаимосвязей (отношений) между объектами;
- установить совокупности данных о выделенных объектах, необходимые и
  достаточные для их представления в БД.

Существует три уровня представления информационных объектов: концептуальный; внешний; внутренний.

Концептуальное представление информационного объекта (называемое также ***инфологическим***) определяет состав типов его данных, их свойства и отношения между данными объектами.

Внешнее представление информационного объекта (или ***пользовательская модель*** предметной области) – это адаптированное к планируемому комплексу задач конкретного пользователя концептуальное представление информационного объекта.

Внутреннее представление информационного объекта, называемое также ***датологической моделью*** (или схемой) БД, ориентировано на конкретную СУБД и определяет принятую технологию хранения и пути доступа к данным, соответствующие избранной СУБД.

Применение моделей различных уровней позволяет обеспечить логическую и физическую независимость данных.

Логической независимостью данных называется свойства базы данных, обеспечивающее возможность изменения общей логической структуры данных без изменения прикладных программ пользователей. ***Физической независимостью*** данных называется свойство базы данных, обеспечивающее неизменность общей логической структуры данных и прикладных программ при изменениях физического расположения и организации данных в памяти компьютера.

1.2.2. Инфологическое моделирование

Инфологический подход к построению информационных систем – это установление соответствия между составлением предметной области и его восприятием и отображением в БД. При инфологическом моделировании основным составным элементом предметной области является «сущность».

Сущностью называется некоторая принятая в конкретной постановке задачи абстракция реального объекта, процесса или явления, о котором необходимо хранить информацию в системе. В качестве синонима термина «сущность» используется также термин «информационный объект», либо просто «объект». Характеристика, описывающая какое-либо свойство сущности, которое можно сформулировать и записать, называется *атрибутом.* Например: количество, цвет, цена и т.п. Атрибут, который однозначно определяет сущность, называется идентифицирующим атрибутом или *идентификатором.*

Все информационные объекты делятся на *материальные* (вид товара, населённый пункт и т.п.) и *нематериальные* (счёт в банке, событие, адрес клиента и т.п.).

По структуре объекты разделяются на *атомарные и составные*. Составные объекты имеют структуру, определяющую композицию внутренних составляющих, которые, в свою очередь, могут быть атомарными или составными.

По взаимосвязи с другими объектами объекты могут быть локальными и реляционными. Объект, свойства которого не зависят от его отношений с другими объектами, называется *локальным.* Объект, свойства которого зависят от его отношений с другими объектами, называется *реляционным.*

Каждое ***отношение*** (или ***связь)*** между информационными объектами по числу входящих в него объектов характеризуется степенью n=1,2,…,n. Соответственно, связи сущностей могут быть бинарные (между двумя сущностями), тернарные (между тремя сущностями) и т.д. Чаще всего в информационных объектах связи бинарные.

На основе инфологического подхода формируется инфологическая модель (ИЛМ) ПО. Она основывается на знаниях пользователя, администратора базы данных и использует естественный язык для фиксации, а также описания выделенных сведений о предметной области. Инфологическая модель является исходной моделью при описании ПО. Основной моделью, которая используется на этапе инфологического проектирования, является модель типа «сущность-связь».

Моделью типа «сущность-связь» называется модель, представляющая информационные объекты предметной области, называемые сущностями, а также их взаимоотношения. Для построения модели «сущность-связь» применяются три конструктивных элемента: сущность, атрибут, связь.

Наибольшее распространение в качестве ***структурных элементов для моделирования данных*** получили термины «поле», «запись», «файл», «база данных».

Полем называется наименьшее поименованное данное, к которому в БД можно непосредственно адресоваться и с помощью которого выполняется построение всех остальных структур данных. В БД с позиций моделирования рассматривают: тип поля (или тип данного) и экземпляр (или значение) поля, то есть само данное. Например: тип поля: ФИО, значение поля: Иванов.

Записью называется поименованная совокупность полей. Для записей, как и для поля, рассматривают тип записи и экземпляр записи. Например: Тип записи: Служащий банка с типами полей: ФИО, Дата рождения, Образование, Должность; экземпляр записи: Петров А.А., 10.05.90, Высшее, Инженер. *Файлом* называется поименованная совокупность взаимосвязанных записей одного типа, т.е. хранящихся вместе данных. *Базой данных* называется поименованная совокупность экземпляров записей разного типа, содержащая связи между этими записями.

Вопросы проектирования базы данных методом «сущность-связь» рассмотрены в разделе 2 «Проектирование баз данных».

1.2.3. Обобщенная структура модели данных

В целом представление информационных объектов отображаются совокупностью моделей данных, составляющих многоуровневую структуру. Модель каждого последующего уровня строится на основе фиксированных характеристик моделей предыдущих уровней. Обобщенная структура модели данных представлена на рис. 3.

Предметная область (ПО) на первом уровне структуры моделируется инфологической моделью (ИЛМ). ИЛМ ПО вместе с описаниями информационных потребностей пользователей (ОИПП) образует глобальную модель предметной области (ГМПО), которая обеспечивает интегрированное представление о ПО и является исходной для администратора базы данных (АБД), представляя сведения об объектах ПО, их количествах, о фиксируемых свойствах этих объектов, связях и динамике изменения объектов.

Индивидуальные представления отдельных пользователей П1, П2, … , ПN об информационных объектах ПО отображают пользовательские модели (или внешние модели) ПО: ПМПОП1, … , ПМПППОN. Модель, в среде которой функционирует СУБД, называется *информационной* моделью предметной области (ИМ ПО). Она находится в распоряжении и под управлением АБД и включает в себя: информационную модель БД (ИМ БД), БД и прикладные программы пользователей БД (ПППП1, … , ППППN).

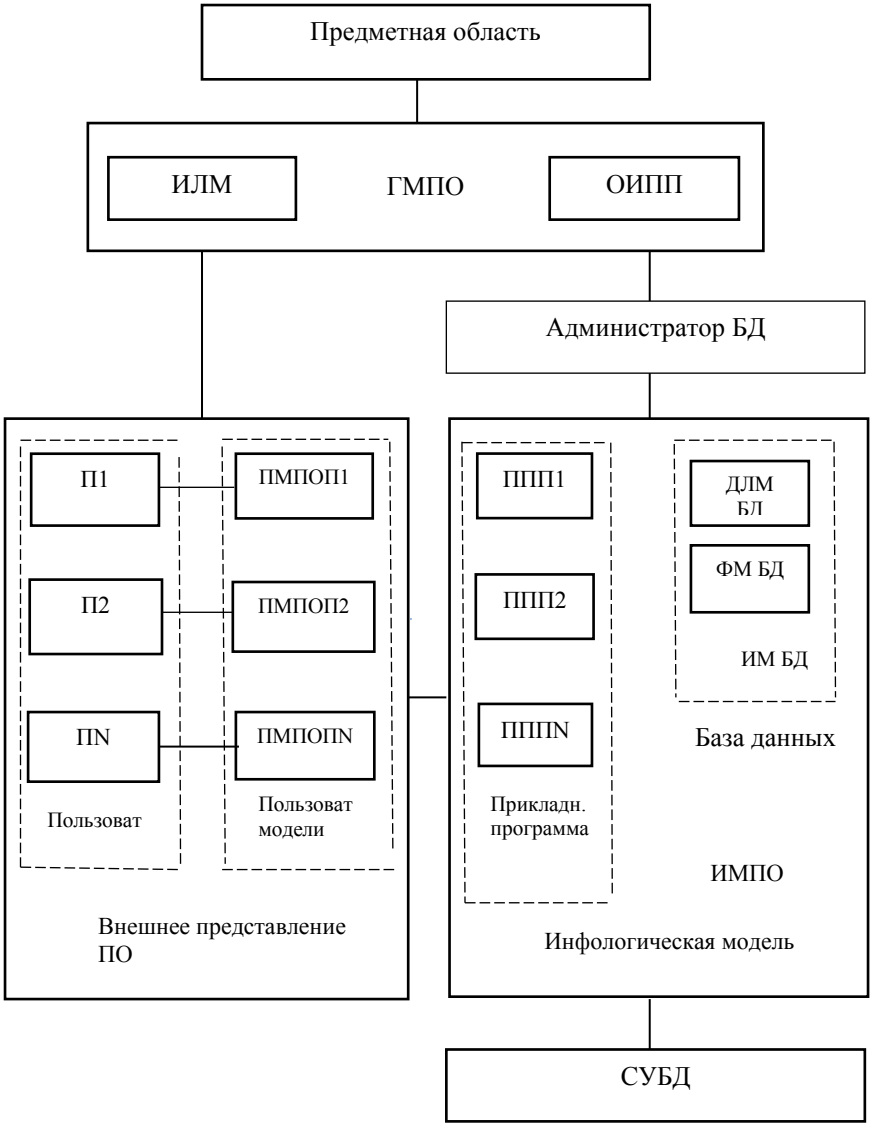


Рис. 3. Структура модели данных

Информационная модель БД (ИМ БД) включает в себя датологическую модель БД (ДЛМ) (или схему БД) и физическую модель БД (ФМ БД) (или схему хранения БД), полное название которой – модель данных физического уровня.

Датологическая модель БД, называемая также моделью данных логического уровня, поддерживается средствами СУБД и отображает логические связи между элементами данных, обеспечивая интегрирование и взаимосвязанное хранения данных безотносительно к их содержанию и среде хранения. Датологическая модель учитывает возможности конкретной СУБД и особенности предметной области, отображаемые в инфологической модели.

Физическая модель БД используется для привязки датологической модели БД к среде физического хранения данных, т.е. к техническим средствам.

1.3. Формы представления структур данных

Формы представления структур данных стандартизированы. В настоящее время широко используются следующие три формы представления структур данных в БД.

Табличная форма структур данных

В таблице строки представляют значения (или экземпляры) записей, а столбцы – это поле базы данных.

Поле – столбец таблицы, содержащий значение определенного свойства.

Запись – строка таблицы, содержащая набор значений свойств, размещенный в полях базы данных.

Ключевое поле – поле, значение которого однозначно определяет запись в таблице идентифицирует каждую запись).

Обобщенный вид табличной формы представлен на рис.4.

Тип поля 1	Тип поля 2	…	Тип поля N
Значение 1 поля 1	Значение 1 поля 2	…	Значение 1 поля N
Значение 2 поля 1	Значение 2 поля 2	…	Значение 2 поля N
…	…	…	…

Рис. 4. Пример табличной формы структуры данных

Графовая форма структуры данных

Графовая форма облегчает понимание и интерпретацию данных. На графе сущности, моделируемые записями, отображаются вершинами графа – прямоугольниками, а связи между сущностями отображаются соответствующими дугами (см. рис.5). Над вершинами указываются типы записей, в вершинах приводятся типы полей. Типы связей записываются вдоль дуг. Типы полей-идентификаторов подчеркиваются.

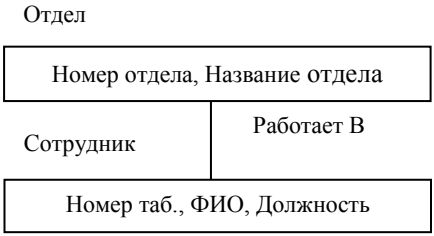


Рис. 5. Пример графовой формы структуры данных

Графическая диаграмма структуры данных

Цель диаграммы (см. рис.6) состоит в более выразительной детализации элементов.

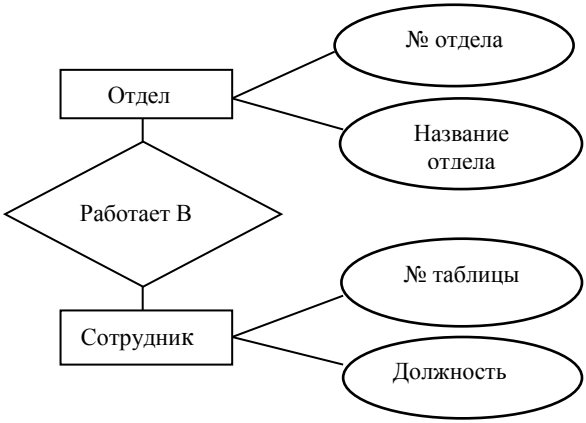


Рис. 6. Пример графической диаграммы структуры данных

На диаграмме используются следующие обозначения:

- типы сущностей обозначаются прямоугольниками;
- атрибуты обозначаются овалами, соединенными с соответствующими типами сущностей дугами; ключи (идентификаторы) подчеркиваются;

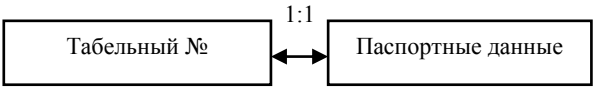
- отношения (или связи) отображаются ромбами, которые соединяются с соответствующими типами сущностей при бинарных связях направленными дугами, а при других связях – направленными дугами.

1.4. Бинарные отношения

Рассмотрим два типа сущностей: А и Б. Между ними возможно четыре вида бинарных отношений (связей).

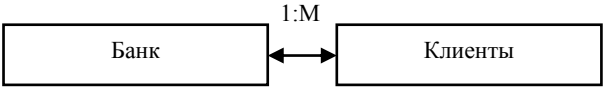
Отношение «один к одному» (1:1) – это связь, при которой каждому экземпляру сущности А соответствует один и только один экземпляр сущности Б и наоборот.

Например, схема структуры данных:

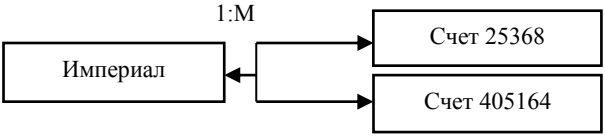


Отношение «один ко многим» (1:М) – это связь, при которой одному экземпляру сущности А может соответствовать 1, 2 или более экземпляров сущности Б. Однако каждому экземпляру сущности Б соответствует только один экземпляр сущности А.

Например, схема структуры данных:

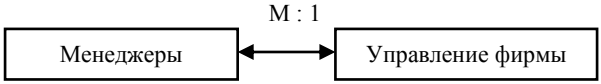


экземпляр схемы структуры данных:

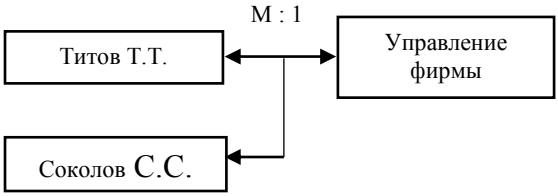


Отношение «многие к одному» (М:1) – это связь с характеристиками, противоположными типу 1:М.

Например, схема структуры данных:

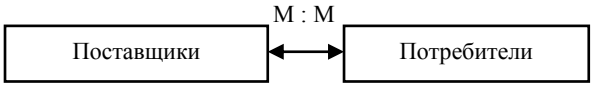


экземпляр схемы структуры данных:



Отношение «многие ко многим» (M:M) – это связь, при которой каждому экземпляру сущности А может соответствовать 1, 2 или более экземпляров сущности Б и наоборот.

Например, схема структуры данных:



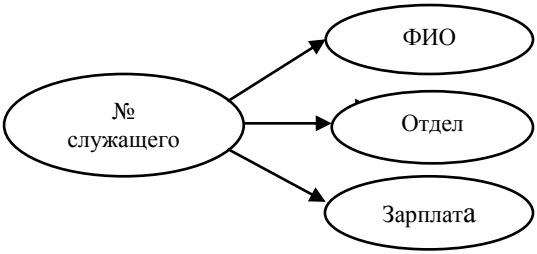
1.5. Интеграция полей базы данных в отношения

В больших базах данных, которые содержат сотни и тысячи различных типов полей, может быть более миллиона связей, что серьезно усложняет и резко замедляет быстродействие информационных систем и оперативность взаимодействия пользователей с такими БД.

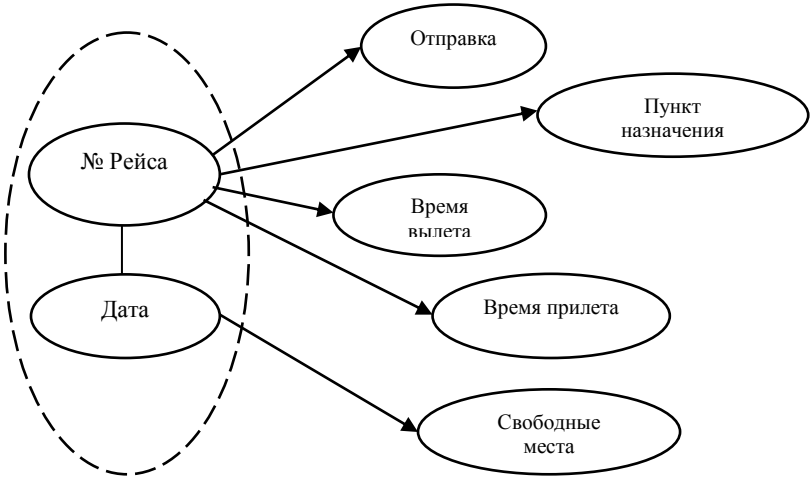
С целью повышения эффективности функционирование банков данных и для уменьшения количества связей поля в БД *интегрируется*, т.е. объединяются в записи с выделением *ключевых полей.*

Для обеспечения поиска конкретных записей объекта среди множества типов полей выделяется уникальное поле, являющееся идентифицирующим, которое называется *первичным ключом* таблицы (пример представлен на рис.7.а).

При формировании ключа возможно использование нескольких полей. Ключ, сформированный из совокупности полей, называется *сцепленным ключом*. Он обозначается в виде суммы и подчеркивается. На рис.7.б. представлен пример, где ключ состоит из двух полей: № Рейса и Дата. Такой ключ обрабатывается как одно поле. Соответствующие записи называются записями *сложной структуры,* в отличие от записей с ключами из одного поля, называемых записями *простой структуры.*



а) Простой ключ (первичный)



б) Сцепленный ключ

Рис. 7. Примеры ключей

В больших БД записи дополнительно группируются в отдельные небольшие структуры, между которыми должно быть некоторое число связей. При отсутствии заранее фиксированного набора отношений группировка атрибутов в отношения допускает большое количество различных вариантов, среди которых необходимо выбрать наилучший.

Рациональные варианты группировки атрибутов в отношения должны удовлетворять следующим требованиям:

1) каждая запись должна иметь простую структуру, т.е. иметь простой ключ, и лишь некоторые записи – сложную структуру;

2) максимально использовать отношение типа 1:1 или 1:M;
3) обеспечение свободы выполнение операций включения, удаления и модификации данных в БД;
4) минимальность реструктурирования отношений при введении новых типов данных;

Неправильное проектирование БД приводит к аномалиям манипулирования данными, т.е. к затруднениям выполнения операций включения, удаления и модификации данных. Чтобы избежать этих проблем, в теории баз данных разработаны соответствующие методы преобразования (или нормализации) исходных схем отношений проекта БД (см. раздел 2. Проектирование баз данных). Они позволяют обеспечить целостность БД и уменьшить вероятность получения ошибок и неверных программных структур.

1.6. Виды моделей данных для БД

При разработке прикладных программ для пользователей БД удобно ориентироваться на заранее проработанные и рекомендованные типовые модели данных, т.е. некоторые стандартные модели, структуры которых удовлетворяют заранее определенным требованиям. В современных информационных системах наиболее распространены три вида моделей данных: иерархическая, сетевая, реляционная.

1.6.1. Иерархическая модель данных

Иерархическая модель данных – это модель, имеющая древовидную графовую структуру, представляющую собой иерархию элементов, называемых вершинами (или узлами), соединенных между собой дугами (или ветвями). На верхнем уровне иерархии, называемом первым (см. рис. 8), находится единственный узел, называемый *корнем* дерева. Узлы следующего более низкого уровня порождаются предыдущими узлами. Каждый узел более высокого уровня может породить один или несколько узлов следующего уровня. Узлы, не имеющие порожденных, называются *листьями*.

Применительно к БД узел носит название объект. Объекты находятся в отношении предка (объект, более близкий к корню) к потомку (объект более низкого уровня). Объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются *близнецами*.

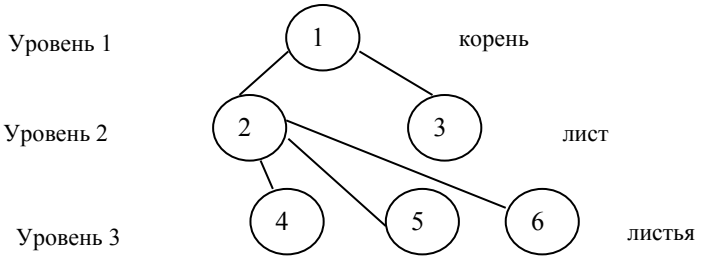
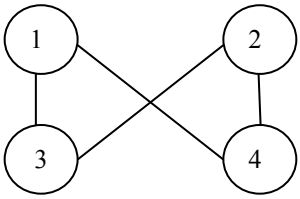


Рис. 8. Иерархическая модель данных

Иерархическая структура используется как для логического, так и для физического описания данных. Файлы с записями, связанными древовидной структурой, называются иерархическими. Примером иерархической базы данных является Каталог папок Windows, с которым можно работать, запустив *Проводник*. Папки *Мой компьютер, Мои документы* и др. являются потомками корневой папки *Рабочий стол*. В свою очередь, папка *Мой компьютер* является предком по отношению к папкам третьего уровня: *Диск С), (D:)* и т.д.

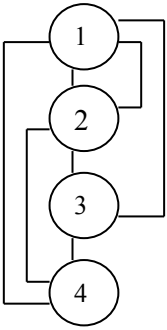
1.6.2. Сетевая модель данных

Сетевой моделью данных называется структура, в которой порожденный элемент (потомок) может иметь больше одного исходного элемента (предка). На связи между объектами в сетевых моделях не накладывается никаких ограничений (см. рис.9).



Каждый из порожденных элементов (3 и 4) имеет по два исходных (1 и 2).

а)



Элемент 4 имеет три исходных (1,2,3)

б)

Рис.9. Сетевая модель данных

Для сетевых моделей данных, как и для иерархических, рассматривается уровневость. Так, структура на рис. 7:

а) двухуровневая; б) четырехуровневая.

В зависимости от уровней связи сетевые модели разделяются на два вида структур:

- сетевые модели простой структуры (при наличии связей типа 1:1 и 1:М);
- сетевые модели сложной структуры (при наличии связей типа М:М).

Сетевой базой данных фактически является Всемирная паутина глобальной компьютерной сети Интернет. Гиперссылки связывают между собой сотни миллионов документов в единую распределенную сетевую базу данных.

1.6.3. Реляционная модель данных

Реляционной моделью данных называется модель, представляемая в виде двухмерной таблицы, информационными единицами которой являются поля, домены, кортежи.

Доменом называется совокупность значений какого-либо типа поля. Из домена извлекаются требуемые значения этого поля.

Кортежем называется запись, или строка, реляционной таблицы. По количеству столбцов в кортеже определяется размерной реляционной модели данных, или отношения. Так, при размерности *n* реляционная модель будет степени *n* или *n-мерной*.

Для различных пользователей из БД интересны различные наборы данных и связи между ними. Для одних требуется извлекать подмножества полей таблицы с целью создания таблиц меньшей размерности, а для других пользователей, наоборот, объединять поля в таблицы большей размерности. Следовательно, *реляционная модель данных предметной области* – это *множество двухмерных таблиц*, называемых отношениями с операторами извлечения и объединения полей.

1.7. Операции над данными

Операции над данными реализуются на основе *языка манипулирования данными*. Язык манипулирования данными СУБД – это набор команд, обеспечивающий:

- передачу команд от прикладных программ (ПП) пользователей в СУБД;
- интерпретацию сообщений СУБД о результатах обработки запросов ПП в СУБД.

В БД предусмотрены следующие виды операций, поддерживаемые средствами СУБД: идентификация данного и определение его позиции в БД; выборка (чтение) данного из БД; включение (запись) данного в БД; удаление данного из БД; модификация данного в БД.

В целом реализация любой операции над данными включает два этапа: селекция данных (выбор); обработка данных, т.е. действие над данными, определяемое характером задания.

Селекция данных

При выборе данных из БД возможно использование различных критериев:

1) по логической позиции данных;
2) по значениям данных;
3) по взаимосвязям данных.

Первый способ базируется на упорядочении данных в памяти системы. Это позволяет селектировать данное, находящееся на первой, следующей, предыдущей, n-ой или последней позиции. При *втором* способе критерий селекции может формироваться на основе условий отбора или на основе операторов алгебры логики.

Третий способ поясним на примере:

В БД «Отдел» с записями: Отдел (№ отдела, № Корпуса, Начальник отдела); Служащий (№ таб., ФИО, Год рождения, Образование); Работает в (№ таб., № отдела) между сущностями *Отдел* и *Служащий* реализована связь *Работает в.* Тогда допускается селекция с использованием этой связи. В результате можно, например, селектировать *места* расположения тех *отделов*, в которых *работают* служащие, возраст которых менее 30 лет.

Обработка данных

Обработка данных осуществляется на основе процедур БД.

Процедурой БД называется последовательность операций, позволяющая реализовать определенный алгоритм обработки данных для получения искомого результата. Процедура является единой, неделимой макрооперацией. Возможность применения процедур существенно расширяет динамические свойства моделей данных. Например: в БД «Отдел» (пример рассмотрен выше) определить количество служащих в возрасте менее 30 лет. Для решения этой задачи необходимо сначала выбрать требуемый контингент, а затем выполнить операцию сложения.

Запросы к БД

Запросы бывают двух видов: простые и комбинированные (или сложные).

Простые запросы определяются формулой A(E) = V, где A – атрибут; E – объект предметной области; V – значение атрибута.

Комбинированные, или сложные запросы – это запросы, в которых используется несколько различных условий, сформированных из совокупностей простых запросов, соединенных логическими операторами: OR (или), AND (и), NOT (не).

1.8. Способы обработки данных

В информационных системах используются три варианта обработки данных:

- централизованная;
- распределенная;
- комбинированная.

1.8.1. Централизованная обработка данных

Централизованная обработка реализует операции по манипулированию данными на базе одного центра, называемого файл-сервером.

Централизация обработки данных устраняет ряд проблем информационных систем, к которым относятся: избыточность данных, противоречивость данных, не связанность данных.

Важными факторами централизации являются:

1) данные могут эффективно использоваться централизованными приложениями: например, в системе снабжения, производственном управлении, бухгалтерском учете и т.п.;
2) пользователям во всех подразделениях необходимы одни и те же данные, причем они часто обновляются;
3) система обрабатывает запросы, для которых данные, возникающие в различных подразделениях, рассматриваются в логическом плане как единое целое;
4) большой объем данных общего назначения и защита данных.

1.8.2. Распределенная обработка данных

Распределенная (или децентрализованная) обработка данных обеспечивает распределение нагрузки по нескольким узлам обработки информации. Это улучшает использование информации на местах, уменьшает затраты по обработке данных. Децентрализация удобна для периферийных подразделений, данные которых не используются в других подразделениях. Если на некоторых рабочих станциях данные очень часто обновляются, то в этом случае также выгоднее обрабатывать и хранить данные децентрализовано, т.е. месте, а не в центральном узле.

Распределенная обработка данных ставит и новые задачи:

1) обеспечение логической прозрачности данных по всей БД;
2) автоматическая декомпозиция запроса пользователя на отдельные составные части, которые для выполнения могут пересылаться в различные узлы сети по месту хранения данных на текущей момент обработки запроса;
3) обеспечение координации процессов обработки;
4) синхронизация обновления и обработки копий данных;
5) защита данных и их восстановление при сбоях.

При использовании децентрализованной обработки данных следует учитывать определенные затруднения по обеспечению целостности и непротиворечивости данных, а также их безопасности.

Комбинированная обработка данных предусматривает сбалансированное использование централизованной и децентрализованной схем обработки данных в зависимости от ситуации. В этом случае можно использовать централизованные и децентрализованные данные.

1.9. Классификация баз данных

1.9.1. Классификация по модели данных

Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней (см. п. *1.6.1. Иерархическая модель данных*). Например, если иерархическая база данных содержит информацию о покупателях и их заказах, то существует объект «покупатель» (предок) и объект «заказ» (потомок). Объект «покупатель» имеет указатели от каждого заказчика к физическому расположению заказов покупателя в объект «заказ». В этой модели запрос, направленный вниз по иерархии, прост (например: какие заказы принадлежат этому покупателю); однако запрос, направленный вверх по иерархии, более сложен (например, какой покупатель поместил этот заказ).

К основным понятиям ***сетевой модели базы данных*** относятся: уровень, элемент (узел), связь (см. п. *1.6.2. Сетевая модель данных)*. Узел – это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. В сетевой структуре каждый элемент может быть связан с любым другим элементом.

Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию. Эта модель решает некоторые проблемы, связанные с иерархической моделью, но выполнение простых

запросов остается достаточно сложным процессом. Кроме того, поскольку логика процедуры выборки данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения, то есть, если необходимо изменить структуру данных, то нужно изменить и приложение.

Реляционная база данных – база данных, основанная на реляционной модели данных (см. п. *1.6.3. Реляционная модель данных)*. Слово «реляционный» происходит от англ. relation (отношение). Для работы с реляционными БД применяют реляционные СУБД. Целью нормализации реляционной базы данных является устранение недостатков структуры базы данных, приводящих к вредной избыточности в данных, которая в свою очередь потенциально приводит к различным аномалиям и нарушениям целостности данных.

Объектные базы данных – это модель работы с объектными данными.

Использование объектной модели баз данных легко воспринимается пользователем, так как создается высокий уровень абстракции. Объектная модель идеально подходит для трактовки такого рода объектных данных как изображение, музыка, видео, разного вида текст.

Объектно-ориентированная база данных (ООБД) – база данных, в которой данные моделируются в виде объектов, их атрибутов, методов и классов. Объектно-ориентированные базы данных обычно рекомендованы для тех случаев, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру.

Характеристики, которым должна отвечать любая ООБД:

1. Поддержка сложных объектов. В системе должна быть предусмотрена возможность создания составных объектов за счет применения конструкторов составных объектов. Необходимо, чтобы конструкторы объектов были ортогональны, то есть любой конструктор можно было применять к любому объекту.

2. Поддержка индивидуальности объектов. Все объекты должны иметь уникальный идентификатор, который не зависит от значений их атрибутов.

3. Поддержка типов и классов. Требуется, чтобы в ООБД поддерживалась хотя бы одна концепция различия между типами и классами. Термин «тип» более соответствует понятию абстрактного типа данных. В языках программирования переменная объявляется с указанием ее типа. Компилятор может использовать эту информацию для проверки выполняемых с переменной операций на совместимость с ее типом, что позволяет гарантировать корректность программного обеспечения. С другой стороны класс является неким шаблоном для

создания объектов и предоставляет методы, которые могут применяться к этим объектам. Таким образом, понятие «класс» в большей степени относится ко времени исполнения, чем ко времени компиляции.

4. Поддержка наследования типов и классов от их предков. Подтип, или подкласс, должен наследовать атрибуты и методы от его супертипа, или суперкласса, соответственно.

5. Перегрузка в сочетании с полным связыванием. Методы должны применяться к объектам разных типов. Реализация метода должна зависеть от типа объектов, к которым данный метод применяется. Для обеспечения этой функциональности связывание имен методов в системе не должно выполняться до времени выполнения программы.

6. Вычислительная полнота. Язык манипулирования данными должен быть языком программирования общего назначения.

7. Набор типов данных должен быть расширяемым. Пользователь должен иметь средства создания новых типов данных на основе набора предопределенных системных типов. Более того, между способами использования системных и пользовательских типов данных не должно быть никаких различий.

1.9.2. Классификация по среде физического хранения

По среде физического хранения различают:

1) БД во вторичной памяти (традиционные): средой постоянного хранения является периферийная энергонезависимая память (вторичная память) – как правило, жёсткий диск. В оперативную память СУБД помещает лишь кэш и данные для текущей обработки.
2) БД в оперативной памяти (in-memory databases): все данные находятся в оперативной памяти.
3) БД в третичной памяти (tertiary databases): средой постоянного хранения является отсоединяемое от сервера устройство массового хранения (третичная память), как правило, на основе магнитных лент или оптических дисков. Во вторичной памяти сервера хранится лишь каталог данных третичной памяти, файловый кэш и данные для текущей обработки; загрузка же самих данных требует специальной процедуры.

1.9.3. Классификация по степени распределённости

Централизованная база данных, или сосредоточенная – БД, полностью поддерживаемая на одном компьютере.

Распределённая база данных – БД, составные части которой размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием.

Неоднородная БД – фрагменты распределённой БД в разных узлах сети поддерживаются средствами более одной СУБД

Однородная БД – фрагменты распределённой БД в разных узлах сети поддерживаются средствами одной и той же СУБД.

Фрагментированная, или секционированная *БД* – методом распределения данных является фрагментирование или секционирование, вертикальное или горизонтальное.

Тиражированная БД – методом распределения данных является тиражирование.

1.9.4. Классификация по типу хранимой информации

По типу хранимой информации БД могут быть:

- документальные;
- фактографические;
- лексикографические.

В ***документальных БД*** единицей хранения является какой-либо документ (например, текст закона или статьи), и пользователю в ответ на его запрос выдается либо ссылка на документ, либо сам документ, в котором он может найти интересующую его информацию. БД документального типа могут быть организованы по-разному – без хранения и с хранением самого исходного документа на машинных носителях. К системам первого типа можно отнести *библиографические* и *реферативные* БД, а также БД-указатели, отсылающие к источнику информации. Системы, в которых предусмотрено хранение полного текста документа, называются *полнотекстовыми***.**

В системах документального типа целью поиска может быть не только какая-то информация, хранящаяся в документах, но и сами документы. Так, возможны запросы типа «сколько документов было создано за определенный период времени» и т. п. Часто в критерий поиска в качестве признаков включаются «дата принятия документа», «кем принят» и другие «выходные данные» документов.

Специфической разновидностью баз данных являются базы данных форм документов. Они обладают некоторыми чертами документальных систем (ищется документ, а не информация о конкретном объекте, форма документа имеет название, по которому обычно и осуществляется его поиск), и специфическими особенностями (документ ищется не с целью извлечь из него информацию, а с целью использовать его в качестве шаблона).

В системах ***фактографического*** типа в БД хранится информация об интересующих пользователя объектах предметной области в виде «фактов» (например, биографические данные о сотрудниках, данные о выпуске продукции производителями и т.п.). В ответ на запрос пользователя выдается требуемая информация об интересующем его объекте (объектах) или сообщение о том, что искомая информация отсутствует в БД.

К ***лексикографическим базам данных*** относятся различные словари (классификаторы, многоязычные словари, словари основ слов и т. п.).

1.9.5. Некоторые другие виды баз данных

Пространственная база данных – БД, оптимизированная для хранения и выполнения запросов к данным о пространственных объектах, представленных некоторыми абстракциями: точка, линия и др. В то время, как традиционные БД могут хранить и обрабатывать числовую и символьную информацию, пространственные обладают расширенной функциональностью, позволяющей хранить целостный пространственный объект, объединяющий как традиционные виды данных (описательная часть или атрибутивная), так и геометрические (данные о положении объекта в пространстве).

Временная, или темпоральная БД – это базы, хранящие данные, привязанные ко времени и имеющие средства управления такой информацией. Главное отличие темпоральных систем управления базами данных от обычных реляционных СУБД заключается в том, что для любого объекта, который был создан в момент времени t1 и был удален в момент времени t2, сохраняются все его состояния в этом временном интервале [t1, t2], тогда как в обычной СУБД существует только текущее на конкретный момент времени состояние объекта. Таким образом, в темпоральной БД хранится история изменений состояний объекта, и пользователь может получить информацию о состоянии записи в БД в любой момент времени из указанного промежутка.

Пространственно-временная база данных – БД, в которой одновременно поддерживается одно или более измерений в аспектах как пространства, так и времени.

Сверхбольшая база данных *(Very Large Database, VLDB)* – это база данных, которая занимает чрезвычайно большой объём на устройстве физического хранения. Термин подразумевает максимально возможные объёмы БД, которые определяются последними достижениями в технологиях физического хранения данных и в технологиях программного оперирования данными. Количественное определение понятия «чрезвычайно большой объём» меняется во

времени; в настоящее время считается, что это объём, измеряемый по меньшей мере петабайтами. Для сравнения, в 2005 г. самыми крупными в мире считались базы данных с объёмом 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 объёмами данных. Как правило, необходимы специальные решения для дисковой подсистемы, специальные версии операционной среды и специальные механизмы обращения СУБД к данным.

2. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

2.1. Основные положения

Система управления базами данных имеет два режима работы: проектировочный и пользовательский. Первый предназначен для создания или изменения структуры базы и создания ее объектов. Во втором режиме происходит использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

Проектирование базы данных является основополагающим режимом и состоит из двух этапов: предварительное проектирование и непосредственная разработка. Предварительное проектирование включает в себя: разработку технического задания (выяснение круга задач и потребностей заказчика; составление списка исходных данных; составление списка выходных данных, необходимых заказчику; взаимодействие со всеми службами и подразделениями заказчика), разработку структуры базы данных (составление списка полей БД; определение типа данных для каждого поля; распределение полей по базовым таблицам; выбор в каждой таблице ключевого поля; определение связей между таблицами).

На этапе предварительного проектирования учитываются все пожелания заказчика. Возможность гибкого исполнения этих пожеланий во многом определяется квалификацией разработчика базы данных. При переходе к непосредственной разработке БД начинают работать с системой управления базами данных (СУБД).

При проектировании базы данных решаются две основные проблемы.

1) Каким образом отобразить объекты предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было наиболее удобным

и эффективным? Эту проблему обычно называют проблемой логического проектирования баз данных.

2) Как обеспечить эффективность выполнения запросов к базе данных, т. е. каким образом, имея в виду особенности конкретной СУБД, расположить данные во внешней памяти, создания каких дополнительных структур потребовать и т. д.? Эту проблему обычно называют проблемой физического проектирования баз данных.

В случае реляционных баз данных в плане физического проектирования слишком многое зависит от используемой СУБД. Поэтому в данном разделе рассмотрим вопросы логического проектирования реляционных баз данных, которые существенны при использовании любой реляционной СУБД. С этой точки зрения, проблема проектирования базы данных состоит в обоснованном принятии решений о том, из каких отношений должна состоять БД и какие атрибуты должны быть у этих отношений.

Существует два основных метода проектирования баз данных: метод декомпозиции отношений и метод «сущность-связь».

Классический подход к проектированию базы данных состоит в том, что он осуществляется в терминах реляционной модели данных методом последовательных приближений к удовлетворительному набору схем отношений. Исходной точкой является представление предметной области в виде одного или нескольких отношений, и на каждом шаге проектирования производится некоторый набор схем отношений, обладающих «улучшенными» свойствами. Процесс проектирования представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает лучшими свойствами, чем предыдущая.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений. Примером может служить ограничение первой нормальной формы – значения всех атрибутов отношения должны быть атомарны.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса-Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма (5NF) или

нормальная форма проекции-соединения (PJ/NF).

Основные свойства нормальных форм состоят в следующем: каждая следующая нормальная форма в некотором смысле лучше

предыдущей нормальной формы; при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, которые удовлетворяют требованиям следующей нормальной формы.

Наиболее важные на практике нормальные формы отношений основываются на фундаментальном в теории реляционных баз данных понятии *функциональной зависимости*. Дадим несколько определений.

Функциональная зависимость: в отношении R атрибут Y функционально зависит от атрибута X (X и Y могут быть составными) в том и только в том случае, если каждому значению X соответствует в точности одно значение Y: R.X (r) R.Y.

Полная функциональная зависимость: функциональная зависимость R.X (r) R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.

Транзитивная функциональная зависимость: функциональная зависимость R.X -> R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости R.X -> R.Z и R.Z -> R.Y и отсутствует функциональная зависимость R.Z -> R.X.

Неключевой атрибут: неключевым атрибутом называется любой атрибут отношения, не входящий в состав первичного ключа (в частности, первичного).

Взаимно независимые атрибуты: два или более атрибута взаимно независимы, если ни один из этих атрибутов не является функционально зависимым от других.

2.2. Проектирование баз данных методом декомпозиции отношений

2.2.1. Вторая и третья нормальная форма

Метод декомпозиции отношений рассмотрим на конкретном примере. Пусть имеется следующая схема отношения:

Сотрудники-Отделы-Проекты (СОТР_НОМЕР, СОТР_ЗАРП, ОТД_НОМЕР, ПРО_НОМЕР, СОТР_ЗАДАН)

Первичный ключ: СОТР_НОМЕР, ПРО_НОМЕР

Функциональные зависимости:

СОТР_НОМЕР -> СОТР_ЗАРП
СОТР_НОМЕР -> ОТД_НОМЕР
ОТД_НОМЕР -> СОТР_ЗАРП
СОТР_НОМЕР, ПРО_НОМЕР -> СОТР_ЗАДАН

Хотя первичным ключом является составной атрибут СОТР_НОМЕР, ПРО_НОМЕР, атрибуты СОТР_ЗАРП и ОТД_НОМЕР

функционально зависят от части первичного ключа, атрибута СОТР_НОМЕР. В результате мы не сможем вставить в отношение *Сотрудники-Отделы-Проекты* кортеж, описывающий сотрудника, который еще не выполняет никакого проекта (первичный ключ не может содержать неопределенное значение). При удалении кортежа мы не только разрушаем связь данного сотрудника с данным проектом, но утрачиваем информацию о том, что он работает в некотором отделе. При переводе сотрудника в другой отдел мы будем вынуждены модифицировать все кортежи, описывающие этого сотрудника, или получим несогласованный результат. Такие неприятные явления называются аномалиями схемы отношения. Они устраняются путем нормализации.

Вторая нормальная форма (определение):

Отношение R находится во второй нормальной форме (2NF) в том и только в том случае, когда находится в 1NF, и каждый неключевой атрибут полностью зависит от первичного ключа.

В этом определении предполагается, что единственным ключом отношения является первичный ключ.

Можно произвести следующую декомпозицию отношения *Сотрудники-Отделы-Проекты* в два отношения *Сотрудники-Отделы* и *Сотрудники-Проекты*:

Сотрудники-Отделы (СОТР_НОМЕР, СОТР_ЗАРП, ОТД_НОМЕР)

Первичный ключ: СОТР_НОМЕР

Функциональные зависимости:

СОТР_НОМЕР -> СОТР_ЗАРП

СОТР_НОМЕР -> ОТД_НОМЕР

ОТД_НОМЕР -> СОТР_ЗАРП

Сотрудники-Проекты (СОТР_НОМЕР, ПРО_НОМЕР, СОТР_ЗАДАН)

Первичный ключ: СОТР_НОМЕР, ПРО_НОМЕР

Функциональные зависимости:

СОТР_НОМЕР, ПРО_НОМЕР -> СОТР_ЗАДАН

Каждое из этих двух отношений находится в 2NF, и в них устранены отмеченные выше аномалии.

Вторая нормальная форма (определение для случая наличия нескольких ключей):

Отношение R находится во второй нормальной форме (2NF) в том и только в том случае, когда оно находится в 1NF, и каждый неключевой атрибут полностью зависит от каждого ключа R.

Отношения с несколькими ключами слишком громоздки и относятся к ситуациям, редко встречающимся на практике. Поэтому данные примеры рассматривать не будем.

Рассмотрим еще раз отношение *Сотрудники-Отделы*, находящееся в 2NF. Функциональная зависимость СОТР_НОМЕР -> СОТР_ЗАРП является транзитивной, она является следствием функциональных зависимостей СОТР_НОМЕР -> ОТД_НОМЕР и ОТД_НОМЕР -> СОТР_ЗАРП. То есть, заработная плата сотрудника на самом деле является характеристикой не сотрудника, а отдела, в котором он работает.

В результате мы не сможем занести в базу данных информацию, характеризующую заработную плату отдела, до тех пор, пока в этом отделе не появится хотя бы один сотрудник (первичный ключ не может содержать неопределенное значение). При удалении кортежа, описывающего последнего сотрудника данного отдела, мы лишимся информации о заработной плате отдела. Чтобы согласованным образом изменить заработную плату отдела, мы будем вынуждены предварительно найти все кортежи, описывающие сотрудников этого отдела. Т.е. в отношении *Сотрудники-Отделы* по-прежнему существуют аномалии. Их можно устранить путем дальнейшей нормализации.

Третья нормальная форма (определение):

Отношение R находится в третьей нормальной форме (3NF) в том и только в том случае, если находится в 2NF и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

В этом определении снова предполагается существование единственного ключа.

Можно произвести декомпозицию отношения *Сотрудники-Отделы* в два отношения *Сотрудники* и *Отделы*:

Сотрудники (СОТР_НОМЕР, ОТД_НОМЕР)

Первичный ключ: СОТР_НОМЕР

Функциональные зависимости:

СОТР_НОМЕР -> ОТД_НОМЕР

Отделы (ОТД_НОМЕР, СОТР_ЗАРП)

Первичный ключ: ОТД_НОМЕР

Функциональные зависимости:

ОТД_НОМЕР -> СОТР_ЗАРП

Каждое из этих двух отношений находится в 3NF и свободно от отмеченных аномалий.

Третья нормальная форма (определение для случая наличия нескольких ключей):

Отношение R находится в третьей нормальной форме (3NF) в том и только в том случае, если находится в 2NF, и каждый неключевой атрибут не является транзитивно зависимым от какого-либо ключа R.

В большинстве случаев на практике третья нормальная форма схем отношений достаточна, и приведением к третьей нормальной форме

процесс проектирования реляционной базы данных обычно заканчивается. Но существуют и другие варианты, когда полезно продолжить процесс нормализации.

2.2.2. Нормальная форма Бойса-Кодда

Рассмотрим конкретный пример. Пусть имеется следующая схема отношения:

Сотрудники-Проекты (СОТР_НОМЕР, СОТР_ИМЯ, ПРО_НОМЕР, СОТР_ЗАДАН)

Возможные ключи:
СОТР_НОМЕР, ПРО_НОМЕР
СОТР_ИМЯ, ПРО_НОМЕР

Функциональные зависимости:
СОТР_НОМЕР -> СОТР_ИМЯ
СОТР_НОМЕР -> ПРО_НОМЕР
СОТР_ИМЯ -> СОТР_НОМЕР
СОТР_ИМЯ -> ПРО_НОМЕР
СОТР_НОМЕР, ПРО_НОМЕР -> СОТР_ЗАДАН
СОТР_ИМЯ, ПРО_НОМЕР -> СОТР_ЗАДАН

В этом примере мы предполагаем, что личность сотрудника полностью определяется как его номером, так и именем.

В соответствии с определением третьей нормальной формы отношение *Сотрудники-Проекты* находится в 3NF. Но здесь имеются функциональные зависимости атрибутов отношения от атрибута, являющегося частью первичного ключа, что приводит к аномалиям. Например, для того, чтобы согласованно изменить имя сотрудника с данным номером, потребуется модифицировать все кортежи, включающие его номер.

Детерминант – это любой атрибут, от которого полностью функционально зависит некоторый другой атрибут.

Нормальная форма Бойса-Кодда (определение):

Отношение R находится в нормальной форме Бойса-Кодда (BCNF) в том и только в том случае, если каждый детерминант является возможным ключом.

Для рассматриваемого нами примера отношений *Сотрудники-Проекты* это требование не выполнено. Можно произвести его декомпозицию к отношениям *Сотрудники* и *Сотрудники-Проекты*:

Сотрудники (СОТР_НОМЕР, СОТР_ИМЯ)

Возможные ключи:
СОТР_НОМЕР
СОТР_ИМЯ

Функциональные зависимости:
СОТР_НОМЕР -> СОТР_ИМЯ
СОТР_ИМЯ -> СОТР_НОМЕР

Сотрудники-Проекты (СОТР_НОМЕР, ПРО_НОМЕР, СОТР_ЗАДАН)

Возможный ключ: СОТР_НОМЕР, ПРО_НОМЕР

Функциональные зависимости: СОТР_НОМЕР, ПРО_НОМЕР -> СОТР_ЗАДАН

Для данного примера возможен другой вариант декомпозиции – если выбрать за основу СОТР_ИМЯ. И в том, и в другом случае получаемые отношения *Сотрудники* и *Сотрудники-Проекты* находятся в BCNF, и им не свойственны отмеченные аномалии.

2.2.3. Четвертая и пятая нормальная форма

Пусть имеется следующая схема отношения:

Проекты (ПРО_НОМЕР, ПРО_СОТР, ПРО_ЗАДАН)

Отношение *Проекты* содержит номера проектов, для каждого проекта список сотрудников, которые могут выполнять проект, и список заданий, предусматриваемых проектом. Сотрудники могут участвовать в нескольких проектах, и разные проекты могут включать одинаковые задания. Каждый кортеж отношения связывает некоторый проект с сотрудником, участвующим в этом проекте, и заданием, которое сотрудник выполняет в рамках данного проекта. В соответствии с приведенными выше условиями единственным возможным ключом отношения является составной атрибут ПРО_НОМЕР, ПРО_СОТР, ПРО_ЗАДАН, и никаких других детерминантов нет. Следовательно, отношение *Проекты* находится в BCNF. Но при этом оно обладает недостатками: если, например, некоторый сотрудник присоединяется к данному проекту, необходимо вставить в отношение *Проекты* столько кортежей, сколько заданий в нем предусмотрено.

Многозначные зависимости (MVD – multi-valued dependency)

В отношении R (A,B,C) существует многозначная зависимость R.A -> -> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

В отношении *Проекты* существуют две многозначные зависимости:
ПРО_НОМЕР -> -> ПРО_СОТР
ПРО_НОМЕР -> -> ПРО_ЗАДАН

Дальнейшая нормализация отношений, аналогичных отношению *Проекты*, основывается на *теореме Фейджина:*

Отношение R (A,B,C) можно спроецировать без потерь в отношения R1 (A, B) и R2 (A, C) в том и только в том случае, когда существует MVD A -> -> B|C.

Под проецированием без потерь понимается такой способ декомпозиции отношения, при котором исходное отношение полностью и без избыточности восстанавливается путем естественного соединения полученных отношений.

Четвертая нормальная форма (определение):

Отношение R находится в четвертой нормальной форме (4NF) в том и только в том случае, если в случае существования многозначной зависимости A -> -> B все остальные атрибуты R функционально зависят от A.

В нашем примере можно произвести декомпозицию отношения *Проекты* в два отношения *Проекты-Сотрудники* и *Проекты-Задания*:

Проекты-Сотрудники (ПРО_НОМЕР, ПРО_СОТР)

Проекты-Задания (ПРО_НОМЕР, ПРО_ЗАДАН)

Оба эти отношения находятся в 4NF и свободны от отмеченных аномалий.

Во всех рассмотренных ранее нормализациях производилась декомпозиция одного отношения в два. Иногда это сделать не удается, но возможна декомпозиция в большее число отношений, каждое из которых обладает лучшими свойствами.

Рассмотрим пример. Пусть имеется следующая схема отношения:

Сотрудники-Отделы-Проекты (СОТР_НОМЕР, ОТД_НОМЕР, ПРО_НОМЕР)

Предположим, что один и тот же сотрудник может работать в нескольких отделах и работать в каждом отделе над несколькими проектами. Первичным ключом этого отношения является полная совокупность его атрибутов, отсутствуют функциональные и многозначные зависимости. Поэтому отношение находится в 4NF. Однако в нем могут существовать аномалии, которые можно устранить путем декомпозиции в три отношения.

Зависимость соединения: Отношение R (X, Y, ..., Z) удовлетворяет зависимости соединения * (X, Y, ..., Z) в том и только в том случае, когда R восстанавливается без потерь путем соединения своих проекций на X, Y, ..., Z.

Пятая нормальная форма (определение):

Отношение R находится в пятой нормальной форме (нормальной форме проекции-соединения - PJ/NF) в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.

Введем следующие имена составных атрибутов:

CO = {СОТР_НОМЕР, ОТД_НОМЕР}

СП = {СОТР_НОМЕР, ПРО_НОМЕР}
ОП = {ОТД_НОМЕР, ПРО_НОМЕР}

Предположим, что в отношении *Сотрудники-Отделы-Проекты* существует зависимость соединения: * (СО, СП, ОП)

При вставках и удалениях кортежей могут возникнуть проблемы, которые устраняются путем декомпозиции исходного отношения в три новых отношения:

Сотрудники-Отделы (СОТР_НОМЕР, ОТД_НОМЕР)
Сотрудники-Проекты (СОТР_НОМЕР, ПРО_НОМЕР)
Отделы-Проекты (ОТД_НОМЕР, ПРО_НОМЕР)

Пятая нормальная форма – это последняя нормальная форма, которую можно получить путем декомпозиции. Ее условия достаточно сложны, и на практике 5NF не используется.

2.3. Проектирование баз данных методом «сущность-связь»

2.3.1. ER-диаграммы

Применение метода декомпозиции при проектировании баз данных является пригодным, если количество атрибутов каждого отношения не превышает 20-30. В том случае, когда число атрибутов отношения переусложнит применение методов декомпозиции необходимо использовать метод «сущность-связь», или ER-диаграмм. Он отличается от метода декомпозиции тем, что функциональные зависимости привлекаются не на начальном, а на конечном этапе проектирования.

«Сущность-связь» – это модель предметной области, которая позволяет моделировать объекты предметной области и их взаимоотношения. В основе модели лежит три конструктивных элемента: сущность, атрибут, связь.

Сущность – это некоторая абстракция реально существующего объекта, процесса и явления, о котором необходимо хранить информацию в системе.

Атрибут – это характеристика сущности, описание свойств сущности.

Например: сущность – Книга; атрибуты – Название, Фамилия автора, Год издания.

Связи – это средства, с помощью которых представляются отношения между сущностями. Пример, представленный на рис. 10 и есть диаграмма ER-типа.

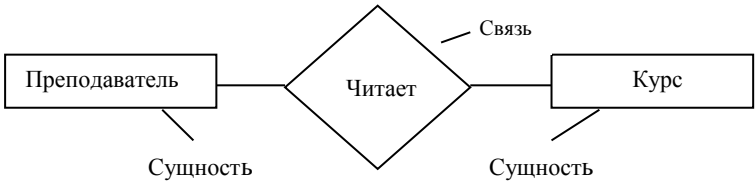


Рис. 10. Пример диаграммы ER-типа

Если эту диаграмму рассмотреть глубже, то получим другую диаграмму, где представлены все преподаватели, все курсы и все связи (рис. 11).

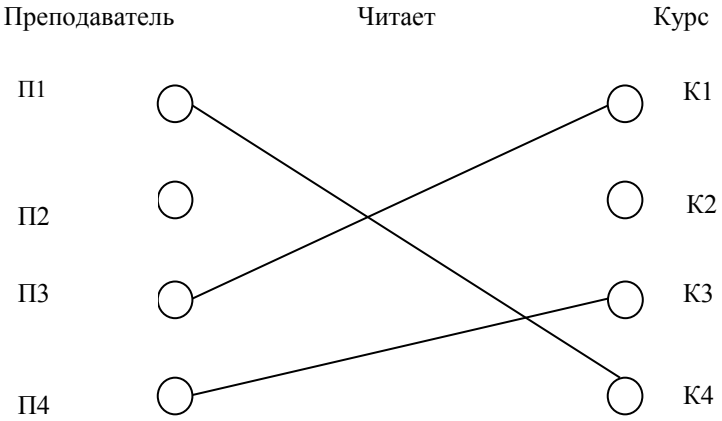


Рис. 11. Диаграмма типа «сущность-связь»

Важными характеристиками связи между сущностями является *степень связи* и *класс принадлежности*. С понятием степень связи мы познакомились ранее (п. 1.4).

Для приведенного выше примера существует также 4 типа степени связи: 1:1– преподаватель читает один курс, 1:M – преподаватель читает много курсов, M:1 – много преподавателей читают один курс, M:M – много преподавателей читают много курсов (рис. 12).

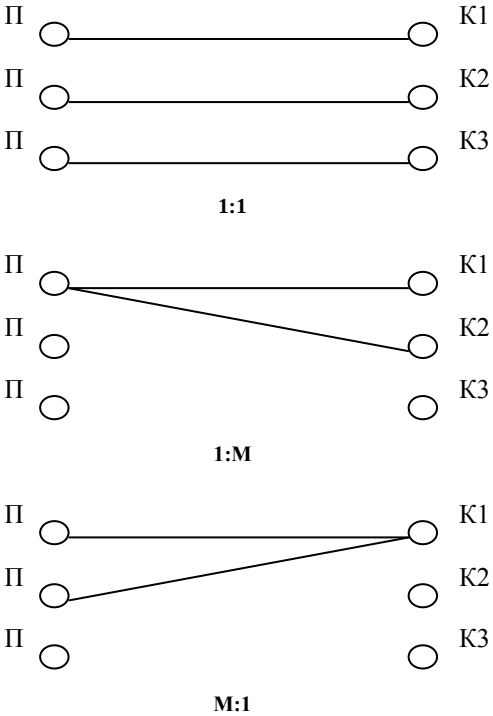


Рис. 12. Пример степеней связи между сущностями

Класс принадлежности сущности определяется правилами, регламентирующими деятельность организации. Он может быть обязательным или необязательным. Если экземпляры данной сущности должны участвовать в связи, класс принадлежности обязательный, в противном случае не обязательный. Эти характеристики – степень связи и класс принадлежности отражаются на ER-диаграммах в виде условных обозначений (рис. 13).

Для данного примера степень связи 1:1; класс принадлежности – ни одна из сущности не является обязательной. Это означает, что не каждый преподаватель читает курсы (например, он занимается методической работой) и не каждый курс читается преподавателем (некоторые курсы студенты должны проработать самостоятельно).

Для каждой сущности вводиться понятие – ключ сущности. Это первичный ключ, однозначно определяющий данные о сущности. В

нашем примере сущность *Преподаватель* имеет первичный ключ – номер преподавателя (НП), сущность *Курс* – номер курса (НК).

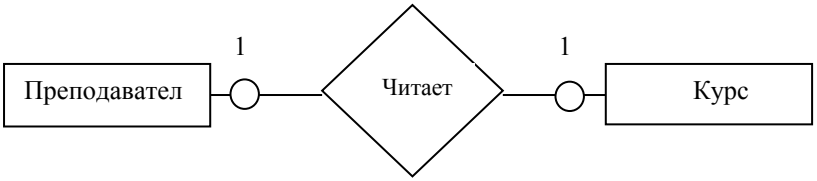


Рис. 13. Диаграмма ER-типа с отражением степени связи 1:1

Другой пример представлен на рис.14. Здесь степень связи 1:M. Класс принадлежности сущности *Преподаватель* – необязательный, *Курс* – обязательный. Это означает, что не каждый преподаватель читает курс, т.е. не все преподаватели загружены работой, но каждый курс читается преподавателем.

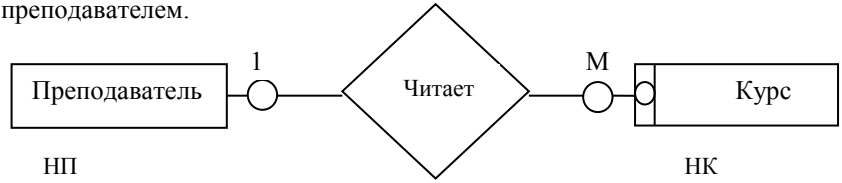


Рис. 14. Диаграмма ER-типа с отражением степени связи 1:M

2.3.2. Получение отношений из ER- диаграмм для бинарных связей

Построение предварительных отношений для бинарных связей степени 1:1

Правило 1. Если степень бинарной связи равна 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей. Отношение может быть наполнено необходимыми атрибутами.

Рассмотрим ER-диаграмму на рис.15.

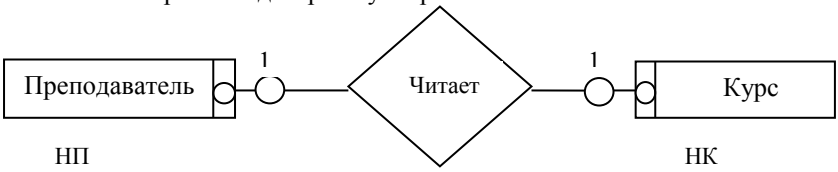


Рис. 15. ER-диаграмма 1

Пусть проектируемое отношение будет называться Преподаватели с ключом НП. Наполним его атрибутами: ПФАМ – фамилия преподавателя, ПТЕЛ – телефон преподавателя. К сущности курс добавим один атрибут: ИМЯ_КУРСА – наименование курса. Тогда схема отношения будет иметь вид: *Преподаватель* (НП, ПФАМ, ПТЕЛ, НК).

Правило 2. Если степень бинарной связи равна 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность необходимо выделение одного отношения при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которой класс принадлежности является необязательным, добавляется в классы атрибута.

Рассмотрим ER-диаграммы на рис.16. Для случая а) класс принадлежности сущности *Преподаватель* – необязательный, класс принадлежности сущности *Курс* – обязательный. Для случая б) класс принадлежности сущности *Преподаватель* – обязательный, класс принадлежности сущности *Курс* – необязательный.

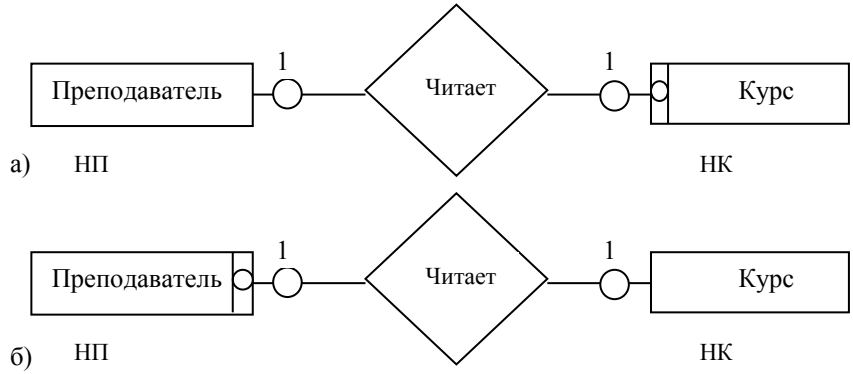


Рис. 16. ER-диаграмма 2

По правилу 2 получаем следующие схемы отношения:

а) *Преподаватель* (НП, ПФАМ, ПТЕЛ), *Курс* (НК, ИМЯ_КУРСА, НП);

б) *Преподаватель* (НП, ПФАМ, ПТЕЛ, НК), *Курс* (НК, ИМЯ_КУРСА).

Правило3. Если степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях и одного для связи. Среди своих

атрибутов отношение, выделяемое для связи, будет иметь по одному ключу от каждой сущности. Ключом отношения может быть любой из ключей сущностей. Для примера на рис. 17 по правилу 3 получаем следующие схемы отношения: *Преподаватель* (НП, ПФАМ, ПТЕЛ), *Курс* (НК, ИМЯ_КУРСА), *Читает* (НП, НК).

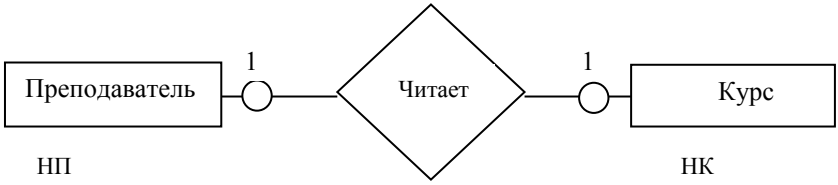


Рис. 17. ER-диаграмма 3

Построение предварительных отношений для бинарных связей степени 1:M

Правило 4. Если степень бинарной связи равна 1:M и класс принадлежности M-связной сущности является обязательным , то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ 1-связной сущности должен быть добавлен как атрибут в отношение отводимое M-связной сущности.

Для примера на рис. 14 по правилу 4 получаем следующие схемы отношения:

Курс (НК, ИМЯ_КУРСА, НП), *Преподаватель* (НП, ПФАМ, ПТЕЛ).

Правило 5. Если степень бинарной связи равна 1:M и класс принадлежности M-связной сущности является необязательным, то необходимо формирование трех отношений: по одному для каждой сущности, причем ключ каждой сущности служит ключом соответствующего отношения, и одного отношения для связи. Связь должна иметь среди своих атрибутов ключ от каждой сущности. В качестве примера рассмотрим ER-диаграмму на рис.18.

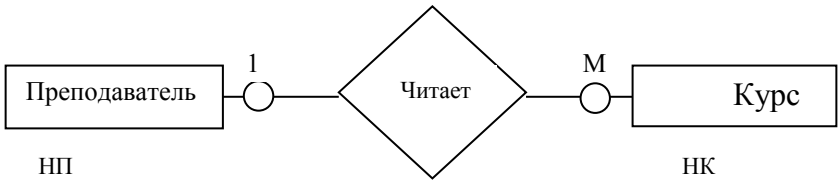


Рис. 18. ER-диаграмма 4

По правилу 5 получаем следующие схемы отношения:
Преподаватель (НП, ПФАМ, ПТЕЛ), *Курс* (НК, ИМЯ_КУРСА), *Читает* (НК, НП).

Построение предварительных отношений для бинарных связей степени M:M

Правило 6. Если степень бинарной связи равна M:M, то вне зависимости от класса принадлежности сущности, для хранения данных необходимо три отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее отношение должно иметь сложный ключ, соответствующий ключу каждой сущности.

Пример ER-диаграммы на рис.19.

По правилу 6 получаем следующие схемы отношения:
Преподаватель (НП, ПФАМ, ПТЕЛ), *Курс* (НК, ИМЯ_КУРСА), *Читает* (НП, НК), НП и НК – сложный ключ

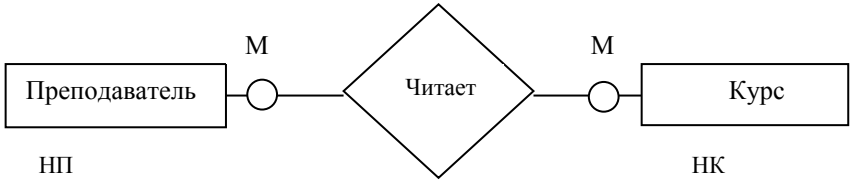


Рис. 19. ER-диаграмма 5

3. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Система управления базами данных (СУБД) представляет собой программный комплекс, предназначенный для выполнения операций по обработке данных с целью обеспечения пользователей информацией.

Необходимо различать собственно ***базы данных*** (БД), которые являются упорядоченными наборами данных, и ***системы управления базами данных*** (СУБД) – программы, управляющие хранением и обработкой данных.

Программные средства, с которыми работают пользователи при решении различных задач, называются приложениями. Именно СУБД призвана обеспечить параллельную и независимую работу множества приложений с единой базой данных. СУБД, как пакет программ, обычно имеет отдельные компоненты для определения и модификации структуры БД, администрирования БД, интерфейса конечного пользователя, для просмотра, ввода и редактирования данных, генерации

отчетов, обмена данными с другими программами, создания приложений и т.д. Эффективность конкретной СУБД определяется наличием и удобством использования средств выполнения этих операций.

3.1. Функциональные возможности СУБД

Средствами СУБД любой пользователь может создать файлы БД, просматривать их, изменять, выполнять поиск, формировать отчеты произвольной формы. Кроме того, он может открыть, просмотреть, выбрать данные и из чужого файла, созданного кем-то программно или средствами СУБД. В настоящее время существует большое количество СУБД, имеющих приблизительно одинаковые возможности.

СУБД выполняет большое количество важнейших функций, многие из которых скрыты от пользователей. Некоторые их них представлены ниже.

Управление хранением данных.

Данная функция предоставляет пользователям возможности выполнения таких операций с данными, как сохранение, извлечение и обновление информации, а также обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным.

Управление буферами оперативной памяти.

СУБД обычно работают с БД значительного размера, этот размер обычно существенно превышает доступный объем оперативной памяти. Если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Единственным же способом реального увеличения этой скорости является буферизация данных в оперативной памяти. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов. При управлении буферами основной памяти приходится разрабатывать и применять согласованные алгоритмы буферизации, журнализации и синхронизации.

Управление словарем данных.

СУБД использует специальный системный каталог, который называют также словарем данных, для поиска необходимых структур данных и их отношений, помогая избежать кодирования таких сложных взаимосвязей в каждой программе, ведь любые программы получают доступ к данным посредством СУБД.

Словарь данных является хранилищем информации, описывающей данные в базе данных. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных

содержится следующая информация: имена, типы и размеры элементов данных; имена связей; накладываемые на данные ограничения поддержки целостности; имена 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. СУБД обеспечивает абстракцию данных, тем самым устраняя в системе структурную зависимость и зависимость по данным.

Журнализация.

Одно из основных требований к СУБД – надежное хранение 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. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.

Поддержание надежного хранения данных в БД требует избыточности хранения данных, причем та их часть, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенный метод поддержания такой избыточной информации – ведение журнала изменений БД. Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая особо тщательно (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД.

Во всех случаях придерживаются стратегии «упреждающей» записи в журнал (протокола Write Ahead Log – WAL). Этот способ 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, если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Управление транзакциями.

Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует изменения БД, произведенные ею, во внешней памяти, либо ни одно из этих изменений никак не отражается в состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.

Примерами простых транзакций может служить добавление, обновление или удаление в базе данных сведений о некоем объекте.

Сложная же транзакция образуется в том случае, когда в базу данных требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

Понятие транзакции обязательное условие даже однопользовательских СУБД, хотя гораздо существеннее во многопользовательских СУБД. То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД.

Управление безопасностью.

СУБД создает систему безопасности, которая обеспечивает защиту пользователя и конфиденциальность данных внутри БД. Правила безопасности устанавливают, какие пользователи могут получить доступ к БД, к каким элементам данных пользователь может получить доступ, какие операции с данными доступны пользователю. В многопользовательских системах данная функция имеет большое значение, так как несколько пользователей могут одновременно получить доступ к данным.

Поддержка языков баз данных.

Для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language – язык структурированных запросов). Так как в состав SQL входят два основных компонента: язык определения данных (DDL) и язык манипулирования данными (DML), SQL позволяет определять схему реляционной БД и манипулировать данными.

3.2. Уровневая архитектура СУБД

Основная цель системы управления базами данных заключается в том, чтобы предложить пользователю абстрактное представление данных, скрыв конкретные особенности хранения и управления ими. Следовательно, отправной точкой при проектировании БД должно быть общее описание информационных потребностей пользователей, которые должны найти свое отражение в создаваемой базе данных.

Более того, поскольку база данных является общим ресурсом, то каждому пользователю может потребоваться свое, отличное от других

представление о характеристиках информации, сохраняемой в базе данных. Для удовлетворения этих потребностей архитектура большинства современных СУБД в той или иной степени строится на базе, так называемой архитектуры ANSI-SPARC, которую мы рассмотрим ниже.

Концепции многоуровневой информационной архитектуры стали основой современной технологии баз данных. Эти идеи связывают с опубликованным в 1975 году отчетом рабочей группы по базам данных ANSI/X3/SPARC (Комитет по планированию стандартов Американского национального института стандартов). В данном отчете была предложена обобщенная трехуровневая модель информационной архитектуры системы базы данных, включающая концептуальный, внутренний и внешний уровни. Такая модель описывает архитектуру любой системы базы данных, но какие-либо ее компоненты или функции в конкретной СУБД могут отсутствовать.

Концептуальный уровень архитектуры ANSI/X3/SPARC служит для поддержки единого «взгляда» на базу данных, общего для всех ее приложений и в этом смысле независимого от них. Именно в среду концептуального уровня при проектировании базы данных отображается концептуальная модель предметной области (см. п. *1.2.1. Информационное моделирование предметной области для БД*). Представление базы данных на концептуальном уровне системы описывается концептуальной схемой базы данных.

Механизмы СУБД, поддерживающие *внутренний уровень* архитектуры, служат для поддержки представления базы данных в среде хранения. Это единственный уровень информационной архитектуры, где база данных в действительности представлена полностью в «материализованном» виде. Описание представления базы данных на внутреннем уровне архитектуры называется внутренней схемой или схемой хранения. На внутреннем уровне хранится следующая информация: сведения о распределении дискового пространства для хранения данных и индексов; описание подробностей сохранения записей; сведения о размещении записей; сведения о сжатии данных и выбранных методах их шифрования.

Уровень, на котором воспринимают данные пользователи, называется *внешним уровнем*. Описания таких представлений называются внешними схемами. В системе базы данных может одновременно поддерживаться несколько внешних схем для различных групп пользователей и приложений.

Необходимо отметить, что в предлагаемой архитектурной модели необходимо поддерживать соответствие между представлениями базы данных на смежных уровнях архитектуры системы базы данных. В модели ANSI/X3/SPARC для этой цели служат механизмы

междууровневого отображения данных «внешний – концептуальный» и «концептуальный – внутренний». Именно эти механизмы обеспечивают абстракцию данных в системе, определяют достижимую в системе степень независимости данных.

Основным назначением трехуровневой архитектуры является обеспечение независимости от данных, которая означает, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

Логическая независимость от данных означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему. Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без внесения изменений в уже существующие внешние схемы или переписывания прикладных программ.

Физическая независимость от данных означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему. Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы. Пользователем могут быть замечены изменения только в общей производительности системы.

3.3. Основные классы СУБД

Основная классификация СУБД основывается на используемой модели баз данных. По этому критерию выделяют несколько классов СУБД: иерархические, сетевые, реляционные, объектные и другие. Некоторые СУБД могут одновременно поддерживать несколько моделей данных. Более ранние СУБД, такие как иерархические и сетевые, имеют древовидную структуру и построены по принципу «предок – потомок». В середине 80-х годов прошлого столетия реляционные системы практически вытеснили с мирового рынка ранние СУБД.

3.3.1. Реляционные СУБД

Реляционный подход организации СУБД предполагает наличие набора отношений (двумерных таблиц), связанных между собой. Связь в данном случае – это ассоциирование двух или более отношений (таблиц). База данных, не имеющая связей между отношениями, имеет очень ограниченную структуру и реляционной называться не может. Запросы к таким базам данных возвращает таблицу, которая повторно может

участвовать в следующем запросе. Данные в одних таблицах, как мы говорили, связаны с данными других таблиц, откуда и произошло название «реляционные».

Реляционный подход в построении СУБД имеет ряд достоинств:

- наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;

- наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации баз данных;

- возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

Реляционная модель имеет строгое теоретическое обоснование. Эта теория способствовала созданию декларативного языка SQL, который в настоящее время стал стандартным в отношении определения и манипулирования реляционными базами данных. Другие сильные стороны реляционной модели – простота, пригодность для систем интерактивной обработки транзакций (OLTP), обеспечение независимости от данных. Однако реляционная модель данных и реляционная СУБД, в частности, имеют и определенные недостатки.

Главным недостатком реляционных СУБД считается присущая этим системам ограниченность использования в областях, в которых требуются достаточно сложные структуры данных. Одним из основных аспектов традиционной реляционной модели данных является атомарность (единственность и неделимость) данных, которые хранятся на пересечении строк и столбцов таблицы. Такое правило было заложено в основу реляционной алгебры при ее разработке как математической модели данных. Кроме того, специфика реализации реляционной модели не позволяет адекватно отражать реальные связи между объектами в описываемой предметной области. Данные ограничения существенно мешают эффективной реализации современных приложений, которые требуют уже несколько иных подходов к организации данных.

Основной принцип реляционной модели – устранять повторяющиеся поля и группы с помощью процесса, который называется нормализацией. Плоские нормализованные таблицы универсальны, просты в понимании и теоретически достаточны для представления данных любой предметной области. Они хорошо подходят для приложений, связанных с хранением и отображением данных в традиционных отраслях, таких как банковские или учетные системы, но их применение в системах, основанных на более сложных структурах

данных, часто является затруднительным. В основном, это связано с примитивностью механизмов хранения данных, лежащих в основе реляционной модели.

Фирмы производители реляционных СУБД: ORACLE, Informix, IBM (DB2), Sybase, Microsoft (MS SQL Server), Progress и другие. В своих продуктах производители СУБД ориентируются на работу на различных типах компьютеров (от майнфреймов до портативных) и на различных операционных системах (ОС). Также производители СУБД не обошли вниманием продукты, работающие на настольных компьютерах, такие как dBase, FoxPro, Access и им подобные. Данные СУБД предназначены для работы на PC и решают локальные задачи на одном PC или небольшой группе PC. Часто данные СУБД используются, как зеркальное отображения небольшой части общей корпоративной СУБД, для минимизации требуемых аппаратных и ресурсных затрат для решения небольших задач.

Различные СУБД работают под управлением разных ОС и аппаратной части. Наиболее известные среди таких ОС: UNIX, VAX, Solaris, Windows. В зависимости от объема хранения данных, количества пользователей, осуществляющих одновременный доступ к данным, сложности задач используются различные СУБД на различных платформах. Например, СУБД Oracle на Unix, инсталлированная на многопроцессорный сервер позволяет решать задачи по обеспечению данными сотни тысяч пользователей.

В настоящее время наибольший интерес представляют СУБД ориентированные на операционную систему Windows использующие платформу Intel.

3.3.2. Объектные СУБД

Возникновение объектных баз данных было обусловлено насущной необходимостью решать задачи, связанные с обработкой и хранением сложных многосвязных данных, а также слабоструктурированной и неструктурированной информации: текстом, изображениями, музыкой и т.д. Объектная СУБД (ОСУБД) идеально подходит для интерпретации такого рода данных, в отличие от реляционных СУБД, где добавление нового типа данных достигается ценой потери производительности или за счет резкого увеличения сроков и стоимости разработки приложений.

Объектная технология – это опыт отражения того, как в действительности человек думает об информации и использует ее. Теоретически, сущности реального мира трактуются как объекты, имеющие состояние (что представляется текущими значениями данных

объектов) и поведение (которое можно наблюдать и на которое можно влиять через программный код объектов).

Объекты, в отличие от реляционных таблиц, тесно увязывают данные и программный код. Объект представляет собой пакет, включающий значения всех данных этого объекта («свойства») и копию всех его кодов («методы»). Методы объекта направляют сообщения для взаимодействия с другими методами этого же или других объектов.

В объектной технологии свойства данных не сводятся к простым «компьютерным» типам данных. Объекты могут содержать внутри себя другие объекты или ссылки на них. Это облегчает построение точных и удобных моделей данных.

Объектные СУБД реализуют весь набор функций, присущих системам управления базами данных плюс возможности объектного программирования. Таким образом, мы получаем все преимущества СУБД наряду с мощным объектным языком программирования (среди них C++, Java, Smalltalk) объектов базы.

В объектной технологии все сложности структур данных скрываются внутри объектов, а доступ к информации осуществляется через простой унифицированный интерфейс. Так как объекты позволяют моделировать комплексные данные очень просто, объектное программирование лучше всего подходит для разработки сложных приложений. Точно так же пополнение и изменение базы данных (например, при обработке транзакций) наиболее эффективно осуществляется через объектный доступ.

Объектная база данных обеспечивает доступ к различным источникам данных, в том числе и к данным реляционных СУБД и разнообразные средства манипуляции с объектами базы данных. Как правило, это и интерфейсы СУБД с объектными языками программирования C++, Java, Smalltalk и набор ActiveX-элементов (модулей, воспринимающих высокоуровневые команды от приложений VisualBasic, Delphi и т.д.), которые разработчик может использовать в своей программе для работы с СУБД.

Основными понятиями, с которыми оперирует эта модель:

1) *Наследование* – это способность порождать один класс объектов из другого. Новый класс (подкласс) сохраняет все свойства и методы своего «родителя» (или «предка»), кроме того, он может иметь дополнительные свойства и методы, характерные только для него. Множественное наследование подразумевает, что подкласс может иметь более одного «родителя».

2) *Инкапсуляция* дает возможность трактовать объект как своеобразный «черный ящик». Независимо от уровня сложности, определенный класс того или иного объекта имеет определенное число общедоступных свойств и методов. Приложению не обязательно знать,

как объект устроен и действует изнутри. Оно взаимодействует только со свойствами и методами объекта.

3) *Полиморфизм* означает, что методы, принадлежащие различным классам, могут использовать один и тот же интерфейс вне зависимости от конкретной реализации этих методов. При этом каждый объект исполняет метод так, как это определено для данного класса объектов.

Дополнительно существует еще две особенности объектного подхода – типизация и сохраняемость. Типизация защищает разработчика от некорректного использования в прикладных программах объектов одного класса вместо другого. Сохраняемость (или хранимость) позволяет объекту существовать в системе после завершения выполнения породившего его процесса. Принцип сохраняемости является чрезвычайно важным для концепции объектных СУБД.

Если сравнивать реляционный подход с объектным, можно выявить, что в реляционных БД существуют только два принципиально разных класса объектов – реляционная таблица с конечным набором операций, которые допустимы для отношений, и встроенные процедуры, работающие с отношениями. Но из этих двух классов объектов нельзя создавать совершенно новые типы ввиду того, что в реляционных БД отсутствуют полноценные механизмы, характерные для объектного подхода.

Таким образом, можно выделить преимущества объектных СУБД:

- обеспечивают инкапсуляцию логики и данных в одном объекте;
- поддерживают сложные типы данных и работу на более высоком уровне абстракции, что позволяет с одной стороны создавать сложные структуры данных, в том числе мультимедийные, а с другой - обеспечить простоту их сопровождения и развития.

Однако ОСУБД также имеют и недостатки, среди которых в первую очередь следует отметить отсутствие развитых средств выборки и анализа данных и единой методологии проектирования объектной БД.

Со времен СССР давно и активно развивались объектные СУБД. В этой области известны такие разработки как: GoodBase, ODB-Jupiter, Dss. Данные разработки совершенно различны, выполнялись в разное время и применялись для различных задач (GoodBase - для решения задач в металлургии, ODB-Jupiter – для создания систем хранения и поиска документов, Dss – для создания систем контроля и управления технологическими процессами). Позже лидерами направления объектных СУБД стали: VERSANT (Versant, Inc), ObjectStore (ObjectDesign, Inc), POET (POET Software, Inc), Jasmine (Computer Associates, Inc).

Популярность объектных СУБД в настоящее время неуклонно растет, что объясняется широкими возможностями по их применению для построения информационных систем корпоративного уровня.

3.3.3. Распределенные СУБД

Еще одна классификация СУБД основывается на методах организации хранения и обработки данных. По данному критерию СУБД делят на централизованные и распределённые. Первые работают с БД, которая физически хранится в одном месте (на одном компьютере). Это не означает, что пользователь может работать с БД только за этим же компьютером: доступ может быть удалённым (в режиме клиент-сервер). Большинство централизованных СУБД перекладывает задачу организации удалённого доступа к данным на сетевое обеспечение, выполняя только свои стандартные функции, которые усложняются за счёт одновременности доступа многих пользователей к данным.

Распределенная СУБД – комплекс программ, предназначенный для управления распределенной БД, причем, 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.

Основное предназначение распределенных СУБД состоит в обеспечении средств интеграции локальных баз данных, располагающихся в некоторых узлах компьютерной сети, для того, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим БД как к единой БД.

Преимущества распределенных СУБД перед централизованными:

- отражают структуру организации;
- обладают локальной автономностью;
- обеспечивают высокую доступность данных;
- обладают высокой надежностью и повышенной производительностью.

Распределенные СУБД обладают и недостатками:

- они являются более сложными программными комплексами, чем централизованные СУБД, что обусловлено распределенной природой используемых ими данных;

- увеличение сложности означает и увеличение затрат на приобретение и сопровождение распределенных СУБД;

- в распределенных системах требуется организовать контроль доступа не только к данным, но и защиту сетевых соединений самих по себе;

- повышенная стоимость передачи и обработки данных может препятствовать организации эффективной защиты от нарушений целостности данных;

- 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.

- распределенные системы сложны в управлении, что обусловливает потенциальную опасность потери целостности данных.

Наиболее полно функции распределенной СУБД реализованы в системах: INGRES/STAR, разработанной отделением Ingres Division фирмы The ASK Group Inc.; ORACLE фирмы ORACLE Corp.; модуле распределенной системы DB2 фирмы IBM. Наиболее близко подошли к реализации функций распределенных СУБД такие как: Informix On-line фирмы Informix Software; Sybase System 10 фирмы Sybase Inc.

3.4. Система управления базами данных Access

В СУБД Microsoft Access используется стандартный многооконный интерфейс, но в отличие от других приложений, не многодокументальный. Единовременно может быть открыта только одна база данных, содержащая обязательное окно базы данных и окна для работы с объектами базы данных. В каждый момент времени одно из окон является активным.

Окно базы данных – один из главных элементов интерфейса. Здесь систематизированы все объекты БД: таблицы, запросы, формы, отчеты, макросы и модули.

Таблица: в базах данных вся информация хранится в двумерных таблицах. Это базовый объект БД, остальные объекты создаются на основе существующих таблиц (производные объекты). Каждая строка в таблице – запись БД, а столбец – поле. Запись содержит набор данных об одном объекте, а поле однородные данные обо всех объектах.

Запросы: в СУБД запросы являются важнейшим инструментом. Главное предназначение запросов – отбор данных на основании заданных условий. С помощью запросов выполняют также сортировку данных и фильтрацию

Формы: формы позволяют отобрать данные, содержащиеся в таблицах или запросах, в более удобном для восприятия виде. При помощи форм можно добавлять в таблицы новые данные, а также редактировать или удалять существующие. Форма может содержать рисунки, графики и др. внедренные объекты.

Отчеты: отчеты предназначены для печати данных, содержащихся в таблицах или запросах, в красиво оформленном виде

Макросы: макросы служат для автоматически повторяющихся операций. Запись макроса производится так же, как в других приложениях, например как в приложении Word.

Модули: модули также служат для автоматизации работы БД. Другое назначение модулей – создание новых функций путем программирования. Модули еще называют процедурами обработки событий и пишутся на языке VBA (Visual Basic for Applications). Это одно из средств, с помощью которых разработчик базы может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, а также уровень ее защищенности.

СУБД Microsoft Access предоставляет несколько средств создания каждого из основных объектов:

- ручные (разработка объектов в режиме Конструктор);
- автоматизированные (разработка с помощью программ мастеров);
- автоматические (средства ускоренной разработки простейших объектов).

Соотношение между этими средствами таковы: ручные средства являются наиболее трудоемкими, но обеспечивают максимальную гибкость; автоматизированные и автоматические средства являются наиболее производительными, но и наименее гибкими.

Основные этапы разработки базы данных в среде MS Access.

Процесс разработки конкретного программного приложения в среде Access в первую очередь определяется спецификой предметной области. Однако можно выделить ряд типичных этапов:

- разработка и описание структур таблиц данных;
- разработка схемы данных и задание системы взаимосвязей между таблицам;
- разработка системы запросов к таблицам базы данных и при необходимости их интеграция в схему данных;
- разработка экранных форм ввода/вывода данных;
- разработка системы отчетов по данным;
- разработка программных расширений для базы данных, решающих специфические задачи по обработке содержащейся в ней информации, с помощью иструментария макросов и модулей;
- разработка системы защиты данных, прав и ограничений по доступу.

Это условная последовательность этапов разработки баз данных в MS Access. Между перечисленными этапами существует большое количеств обратных связей, возврат к более ранним шагам, исходя из вновь открывшихся обстоятельств, которые невозможно было учесть заранее.

3.4.1. Создание таблиц

Таблицы – основа базы данных. Все другие объекты (запросы, формы и отчеты) зависят от таблиц. Создание таблиц начинается с выбора элемента управления Таблицы.

При разработке учебных таблиц рекомендуется использовать ручные средства, работать в режиме *Конструктора*. Использование мастеров ускоряет работу, но не способствует освоению понятий и методов.

При создании таблицы в режиме Конструктора пользователь создает и редактирует структуру таблицу: вводит имена полей и тип данных для каждого поля. Структуру таблицы сохраняют.

Базы данных Microsoft Access работают со следующими типами данных:

- текстовый (тексты, содержащие до 255 символов);
- числовой (действительные числа);
- дата/время (календарные даты и текущее время);
- счетчик (целые числа, которые задаются автоматически при вводе записей – эти числа не могут быть изменены пользователем);
- логический (логические данные: значения Истина (да) или Ложь (нет));
- гиперссылка (ссылка на информационный ресурс в Интернете, например, Web-сайт);
- поле Мемо (специальный тип данных для хранения больших объемов текста, до 65535 символов, – физически текст хранится не в поле, он хранится в другом месте БД, а в поле хранится указатель на него);
- поле объекта OLE (специальный тип данных для хранения объектов OLE, например мультимедийных, ресунков или графиков – реально такие объекты в таблице не хранятся (аналогично Поле Мемо), иначе работа с таблицами была бы чрезвычайно замедленной).

Заполнение таблицы данными производится обычным образом. Если данные не умещаются в ячейках таблицы, то ширину столбцов можно изменять методом перетаскивания их границ, либо использовать автоматическое форматирование столбцов «по содержимому».

После заполнения таблицы данными сохранять не надо – все сохраняется автоматически. Если же произошло редактирование ее макета (например, меняем ширину столбцов), СУБД попросит подтвердить сохранение этих изменений.

Заполнение таблиц данными возможно как непосредственным вводом данных, так и в результате выполнения программ и запросов.

3.4.2. Создание межтабличных связей

Если структура базы данных продумана заранее, а связи между таблицами намечены, то создание решающих отношений между таблицами происходит в специальном окне *Схема данных*. Образовавшаяся межтабличная связь отображается в виде линии, соединяющей два поля разных таблиц. При этом одна из таблиц считается главной, а другая связанной. Главная-это та таблица, которая участвует в связи своим ключевым полем. Реальные базы данных содержат множество связанных таблиц. Группа связанных таблиц называется схемой базы данных. Важнейшей задачей, которую позволяет решать схема, является обеспечение логической целостности данных в базе.

Связь между таблицами позволяет:

1) исключить возможность удаления или изменения данных в ключевом поле главной таблицы, если с этим полем связаны какие- либо поля других таблиц;
2) сделать так, что при удалении (или изменении) данных в ключевом поле главной таблицы автоматически (и абсолютно корректно) произойдет удаление или изменение соответствующих данных в полях связанных таблиц;
3) создавать сложные отчеты.

3.4.3. Работа с запросами

Если исполнителю надо получить данные из базы, он должен использовать специальные объекты – *запросы*. Запрос – это выражение, определяющее, какую информацию вам нужно отыскать в одной или нескольких таблицах. Настраивается запрос с помощью *Конструктора запросов*. Полученные сведения находятся в *результирующей таблице.* С помощью запроса можно также выполнить некоторые действия с данными таблицы (таблиц) и обобщить данные таблицы. Запросы могут использоваться как источники информации для форм и отчетов. В этом случае в запросе используются данные из нескольких таблиц. СУБД Access выполняет запрос каждый раз, когда пользователь открывает форму или отчет, и, следовательно, информация, которую он видит на экране, всегда самая актуальная. При необходимости записи в результирующей таблице можно упорядочить (отсортировать по возрастанию или убыванию). Сортировать можно по одному полю или сразу по нескольким полям (многоуровневая сортировка).

Хотя запросы к таблицам баз данных пишутся на специальном языке программирования и запросов – SQL, пользователям Microsoft

Access изучать его не обязательно, большинство операций можно выполнить щелчками кнопок мыши и приемами перетаскивания.

Виды запросов:

1) запросы на выборку (позволяет выбрать данные из полей таблиц на основе которых запрос сформирован);
2) запрос с параметром (критерий отбора может задавать сам пользователь, введя нужный параметр при вызове запроса);
3) итоговые запросы (отдаленно напоминают итоговые функции электронных таблиц - производят математические вычисления по заданному полю и выдают результат);
4) запросы на изменение (позволяют автоматизировать заполнение полей таблицы);
5) перекрестные запросы (позволяют создавать результирующие таблицы на основе результатов, полученных при анализе группы таблиц);
6) специфические запросы SQL – запросы к серверу базы данных, написанные на языке запросов SQL.

3.4.4. Работа с формами

Формы дают альтернативный способ отображения табличных данных. СУБД Microsoft Access позволяет создавать формы, которые можно использовать для ввода, управления, просмотра и печати данных. СУБД позволяет вводить в созданные экранные формы рисунки, узоры, кнопки. Возможно построение форм, наиболее удобных для работы пользователя, включающих записи различных связанных таблиц базы данных. Формы разрабатываются для интерактивной работы с данными, например, ввода новых данных, изменения имеющихся, удаления данных. Можно также назначить форму для выполнения поисковых процедур по получению данных, отвечающих определенным критериям.

В связи с тем, что формы позволяют с одной стороны вводить данные в таблицы БД, а с другой – выводить результаты запросов в виде красиво оформленных форм, существует два вида формирования структуры форм: на основе таблицы и на основе запроса, возможен и комбинированный подход.

Формы удобнее готовить с помощью средств автоматизации. Полностью автоматическими являются средства, называемые *автоформами*. Существует три вида автоформ: «в столбец» (отображает все поля одной записи, удобна для ввода и редактирования данных); ленточная (отображает одновременно группу записей, ее удобно использовать для вывода данных); табличная (по внешнему виду ничем не отличается от таблицы, на которой она основана). Автоформа основывается только на одном объекте. Другие средства создания форм

позволяют закладывать в основу структуры формы поля нескольких таблиц или запросов. Если форма основывается только на одном объекте, она называется *простой* формой. Если форма основывается на полях из нескольких связанных таблиц, то она называется *сложной* и представляет собой композицию из нескольких форм.

Автоматизированные средства предоставляет *Мастер форм* – специальное программное средство, создающее структуру формы в режиме диалога с разработчиком.

Этапы работы Мастера форм:

1) выбор таблиц и полей, которые войдут в будущую форму;
2) выбор внешнего вида формы;
3) выбор стиля оформления формы;
4) сохранение формы под заданным именем.

Форма имеет три основных раздела: *область заголовка*, *область данных* и *область примечаний.* Область заголовка и область примечаний имеют чисто оформительское значение, а область данных – содержательное.

Элементы управления формы, дизайн форм – изучаем на практических занятиях.

3.4.5. Работа со страницами доступа к данным

Объект служит для обеспечения доступа к данным, содержащимся в БД через Интернет или корпоративную сеть Intranet. Обычно БД имеют очень большие размеры, и напрямую передавать их через каналы связи непрактично.

Страницы доступа выполняют как бы посредническую функцию. Они имеют небольшой размер, содержат удобные элементы управления для навигации в базе данных, могут быть записаны в формате HTML. В связи с тем, что по формату они являются Web-документами, их нетрудно встроить в любой Web-документ, например, разместить на Web-странице.

От других объектов БД страницы доступа отличаются тем, что имеют двойную природу. Все ранее рассмотренные объекты являются *внутренними* (мы не можем выделить ни таблицу, ни запрос, ни форму в виде самостоятельного файла). Эти объекты размещаются где-то внутри файла базы данных, с ним работает лишь сама система УБД, но никак не операционная система. Страница же представлена двумя объектами – внутренним объектом базы (его можно редактировать) и внешним объектом – файлом в формате HTML. Запись этого файла происходит при сохранении спроектированной страницы доступа.

Для страниц доступа, как и для форм, важную роль играет внешний вид, поэтому создавать их удобно с помощью *Мастера страниц*.

3.4.6. Работа с отчетами

Отчеты во многом похожи на формы и страницы доступа к данным, но имеют иное функциональное назначение – они служат для форматированного вывода данных на печатное устройство. Отчеты дают возможность распечатать данные на бумаге или сгруппировать их в виде, удобном для анализа. Это более статичная категория представления данных. Многое, что было сказано о формах, относится и к отчетам. Здесь также существуют средства автоматического и ручного проектирования. Средства автоматического проектирования реализованы автоотчетами (существуют «ленточные» автоотчеты и «в столбец»). Средством автоматизированного создания отчетов является *Мастер отчетов.*

Структура готового отчета отличается от структуры формы только увеличенным количеством разделов. Кроме разделов заголовка, примечания и данных, отчет может содержать разделы верхнего и нижнего колонтитула (здесь можно размещать служебную информацию или номера страниц).

Используя специальные средства создания отчетов, пользователь, работающий с СУБД Access, получает следующие дополнительные возможности вывода данных: добавлять информацию, не содержащуюся в базе данных; включать в отчет информацию из разных связанных таблиц базы данных; располагать выводимую в отчете информацию в любом, удобном для пользователя виде (вертикальное или горизонтальное расположение полей).

3.4.7. Работа с макросами и модулями

При обработке больших объемов данных часто приходится выполнять длинные последовательности действий. Они могут быть оформлены как макросы.

В СУБД Access макрос – определенная последовательность операций. Они хранятся в окне, напоминающем таблицу, в таком порядке, в котором их необходимо выполнять. При разработке макроса необходимо задать действия (ввести макрокоманды), которые он должен выполнить (например, открыть форму, распечатать отчет, выполнить запрос или экспортировать содержимое таблицы в файл электронной таблицы). При запуске макроса Access выполняет эти действия.

Макрос может быть как собственно макросом, состоящим из последовательности макрокоманд, так и группой макросов. Группой макросов называют их набор, сохраняемый под общим именем. В некоторых случаях для решения, должна ли в запущенном макросе выполняться определенная макрокоманда, может применяться условное выражение.

Макрос может существенно облегчить использование СУБД Access. Одной из полезных особенностей макросов является возможность их привязки к кнопкам, которые помещаются в формы. Эти кнопки могут выполнять часто повторяющиеся операции (например, открывать диалоговые окна для поиска, изменять порядок сортировки данных или печатать отчет). Пользователи Access могут использовать определенные кнопки для выполнения этих действий, даже не зная всех подробностей операций, выполняемых макросом. Используя макросы, можно создавать завершенные приложения с пользовательским меню и диалоговыми окнами.

Для вызова макросов в Access существует две возможности:

- вызов макроса по команде пользователя (либо непосредственно из раздела Макросы главного окна базы данных, либо с помощью меню или панели инструментов, с которыми он также может быть ассоциирован);
- вызов макроса по некоторому системному событию (открытие или закрытие формы, изменение управляющего элемента и т. п.).

3.4.8. Защита данных в БД

Система защиты данных в БД должна обеспечивать: блокирование несанкционированного считывания; защиту от несанкционированной модификации каким-либо пользователем; защиту данных от искажения какой-либо программой.

Системы защит данных могут быть реализованы в виде: подпрограмм внутри прикладных программ; спецпрограмм внутри СУБД; средств управления логическими или аппаратными ключами защиты в ОС (паролями).

Базы данных – это особые структуры. Информация, которая в них содержится, очень часто имеет общественную ценность. Нередко одной и той же базой (например, с базой регистрации автомобилей в ГИБДД) работают по всей России. От некоторой информации может зависеть благополучие множества людей. Поэтому целостность содержимого базы не может и не должна зависеть ни от конкретных действий пользователя, ни от перебоев в электросети, ни от чего другого.

Проблема безопасности баз данных решается тем, что в СУБД для сохранения информации используется двойной подход. В части

операций, как обычно с другими приложениями, участвует операционная система компьютера, но некоторые операции сохранения происходят в обход ОС.

Операции изменения структуры БД, создания новых таблиц или иных объектов происходят при сохранении файла базы данных. Об этих операциях СУБД предупреждает пользователя (это глобальные операции). Их никогда не проводят с БД, находящейся в коммерческой эксплуатации, – только с ее копией. В этом случае любые сбои в работе вычислительной системы не страшны.

С другой стороны, операции по изменению содержания данных, не затрагивающие структуру базы, максимально автоматизированы и выполняются без предупреждения. Если, работая с таблицей данных, мы что-то меняем в составе данных, то изменения сохраняются немедленно и автоматически. Попытка закрыть базу «без сохранения» (как это возможно во всех других приложениях) ничего не даст, т.к. все уже сохранено.

3.5. Практическая реализация методов проектирования баз данных в среде Microsoft Access

Данный блок практических заданий состоит из 7 лабораторных работ, описание каждой из которых начинается с разделов «Цель работы» и «Темы для предварительного изучения». Указанные в этих разделах темы обязательно должны быть изучены по материалам лекционного курса или по литературным источникам до начала выполнения практической работы.

Задание к выполнению каждой лабораторной работы сформулировано в разделе «Задание». Варианты индивидуальных заданий представлены в Приложении 1.

Порядок выполнения задания подробно описан в каждой лабораторной работе. Здесь для наглядности приведены примеры выполнения, рассматривается некоторая определенная предметная область. Это способствуют более ясному представлению реализации теоретических вопросов разработки баз данных и работы с ними. Изучение и детальный анализ выполнения задания на конкретном примере приводит к пониманию алгоритма решения поставленной задачи. Предметная область для лабораторных работ понятна студентам и не требует специальных знаний.

Обязательно нужно сохранить файл с базой данных на внешнем носителе. Практикум построен таким образом, что результаты первых работ могут быть необходимы при выполнении последующих.

3.5.1. Лабораторная работа №1
Основы работы с таблицами

Цель работы

Получение навыков работы по созданию структуры таблиц, модификации структуры таблиц, заполнению таблиц. Создание ключевых полей, индексированных полей, установка связей между таблицами. Удаление информации из связанных таблиц и восстановление этой информации.

Темы для предварительного изучения

- Создание новой базы данных. Создание таблиц. Типы полей, используемые СУБД ACCESS. Сохранение таблиц. Создание ключевого поля. Изменение структуры таблицы, добавление данных в таблицу. Редактирование и удаление данных. Поиск данных в таблице. Установка связей между таблицами

Задание

Создать структуры таблиц, ключевые и индексные поля. Заполнить таблицы данными, установить связи, удалить данные, восстановить их.

Каждый студент выполняет индивидуальное задание (своя предметная область). Варианты индивидуальных заданий представлены в Приложении (*Приложение 1*).

Выполнение задания

1. Разработка структуры БД

Выполнение начинается с разработки структуры БД. На этом этапе должны быть детально проанализированы условия задания и, на их основе, определено количество таблиц, необходимых для описания всех характеристик анализируемой предметной области. Кроме того, необходимо определить какие поля в таблицах будут использованы в качестве ключевых, а также определить каким образом будет осуществляться связь между таблицами. Если невозможно установить связи посредством использования ключевых полей, определить таблицы, которые будут использоваться только для связи между другими таблицами.

2. Создание таблиц

Для каждого поля конкретной таблицы необходимо определить его тип и размер и тщательно проверить, удовлетворяет ли диапазон значений выбранного типа тем значениям, которые может реально принимать данное поле. При необходимости, для некоторых полей можно установить Условие на значение и задать сообщение, выдаваемое

на экран в случае несоответствия введенного значения заданному условию или присвоить значения, принимаемые по умолчанию. Можно также определить формат вводимой информации для конкретных полей.

Заполнить соответствующей информацией каждый из разделов создаваемой структуры таблицы: *Имя поля, Тип данных* и *Описание*. Раздел описаний необязателен для заполнения, но информация, введенная в данный раздел отображается в строке состояния при вводе данных для конкретного поля, облегчая процесс ввода.

3. Создание индексов и ключевых полей

Информацию в таблицах можно упорядочить, создав индекс для конкретного поля или нескольких полей. Желательно, чтобы для таблиц были созданы ключевые поля. Для установления связей между таблицами наличие таких полей обязательно. Ключевое поле может быть простым или составным, т.е. состоять из нескольких полей для однозначной идентификации каждой записи в таблице.

4. Сохранение таблиц

По окончании создания структуры таблицы ее необходимо сохранить. Для сохранения выполнить: ***Файл→Сохранить как/экспорт***. В окне Сохранение объекта должен быть выбран параметр в текущей базе данных. Ввести имя созданной таблицы. Выполнить щелчок по кнопке **«ОК»**. После сохранения закрыть окно Конструктора таблиц.

5. Заполнение таблиц

Открыть таблицу в режиме таблицы. Заполнить необходимой информацией, подготовив для заполнения не менее десяти записей для основной таблицы. Сохранение не требуется, т.к. сохранение производиться сразу при переходе к следующей записи. Закрыть заполненную таблицу. Аналогично поступить с остальными таблицами.

6. Установка связей между таблицами

Выполнить команду ***Сервис → Схема данных***.

1. Появится окно **Схема данных**. Если связи устанавливаются впервые, оно будет содержать диалоговое окно **Добавление таблицы**. Если окно **Добавление таблицы** отсутствует, его можно открыть, выбрав ***Связи→Добавить таблицу*** или выбрать пиктограмму **Добавить таблицу**.
2. Выбрать таблицу, которая будет использоваться для установки связей, затем выполнить щелчок на кнопке **«Добавить»**, для добавления таблицы в окно **Схема данных**.
3. Повторить действия, описанные в п.2 для каждой таблицы, участвующей в установке связи.
4. Для создания связей между таблицами переместить поле (или поля), которое необходимо связать на соответствующее поле другой

таблицы. В большинстве связей ключевое поле первой таблицы связывается с аналогичным полем второй таблицы. После перемещения поля появится диалоговое окно **Связи**.

5. В диалоговом окне представлены названия таблиц, между которыми устанавливаются связи и имена полей для связи. Полям, на основе которых создаются связи между таблицами, не обязательно иметь одинаковые имена, однако они должны быть одного типа. Исключение составляют поля счетчиков, которые можно связывать с числовыми полями.
6. Для автоматической поддержки целостности БД установить флажок **Обеспечение целостности данных**. Кроме этого флажка в окне представлены и другие:
   - *Каскадное обновление связанных полей*. При включении данного режима изменения, сделанные в связанном поле первой таблицы, автоматически вносятся в поля связанной таблицы, содержащей те же данные.
   - *Каскадное удаление связанных полей*. При включении данного режима удаление записей в первой таблице приводит к удалению соответствующих записей связанной таблицы.
7. Выполнить щелчок на кнопке **«Создать».** Затем закрыть окно **Связи**. При запросе о сохранении связи выполнить щелчок на кнопке **«Да».**

Пример выполнения лабораторной работы

Предметная область: Отдел кадров

Минимальный список характеристик:

- Фамилия, имя, отчество, домашний адрес, телефон, дата рождения, должность, дата зачисления, стаж работы, образование;
- Фамилия, имя, отчество, и даты рождения членов семьи каждого сотрудника, оклад;
- Наименование подразделения, количество штатных единиц, фонд заработной платы за месяц и за год.

ТЕМА: Создание БД. Создание таблиц.

Цель работы:

Приобретение навыков в работе с ACCESS по созданию таблиц.

Постановка задачи:

Создать базу данных ОТДЕЛ КАДРОВ, поместив в нее три таблицы: СОТРУДНИК, СОСТАВ СЕМЬИ и ШТАТНОЕ РАСПИСАНИЕ, содержащие информацию о сотрудниках предприятия.

Описание прикладной области Отдел кадров предприятия.

Анализ предметной области показывает, что для автоматизации работы Отдела кадров целесообразно создать БД ОТДЕЛ КАДРОВ, состоящую из трех таблиц: СОТРУДНИК, СОСТАВ СЕМЬИ, ШТАТНОЕ РАСПИСАНИЕ. Таблицы будут связаны между собой следующим образом:

Таблица СОТРУДНИК с таблицей СОСТАВ СЕМЬИ связываются по полю **Идент код**, а с таблицей ШТАТНОЕ РАСПИСАНИЕ - по полю **Должн**.

Характеристики таблицы-объекта СОТРУДНИК:

- идентификационный код **Идент код** (10 символов - тип текстовый);
- фамилия **Фамилия** (20 символов - тип текстовый);
- имя **Имя** (15 символов - тип текстовый);
- отчество **Отчество** (15 символов - тип текстовый);
- пол **Пол** (1 символ - тип текстовый);
- дата рождения **Дата рожд**(поле типа дата);
- место рождения **Место рожд** (15 символов - тип текстовый);
- образование **Образов** (15 символов - тип текстовый);
- должность **Должн** (15 символов - тип Мастер подстановок);
- стаж работы **Стаж работы** (длинное целое - тип числовой);
- семейное положение **Сем полож** (7 символов - тип текстовый);
- дата зачисления на работу **Дата зач** (поле типа дата/время);
- телефон **Тел** (8 символов - тип текстовый);
- домашний адрес **Адрес** (поле-МЕМО).

Характеристики таблицы-объекта СОСТАВ СЕМЬИ:

- идентификационный код **Идент код** (10 символов - тип Мастер подстановок);
- отношение **Отношение** (10 символов - тип текстовый);
- фамилия **Фамилия** (20 символов - тип текстовый);
- имя **Имя** (15 символов - тип текстовый);
- отчество **Отчество** (15 символов - тип текстовый);
- год рождения **Дата рожд** (поле типа дата/время).

Характеристики таблицы-объекта ШТАТНОЕ РАСПИСАНИЕ:

- № п/п НПП (длинное целое - тип числовой);
- название подразделения Назвподр (30 символов - тип текстовый);
- должность Должн (15 символов - тип текстовый);
- количество штатных единиц Кол ед (длинное целое - тип числовой);

– должностной оклад Оклад (длинное целое - тип числовой);
– фонд заработной платы за месяц ФЗПМ (длинное целое - тип числовой);
– фонд заработной платы на год ФЗПГ (длинное целое - тип числовой)

Описание работы

Загрузить Microsoft Access, выполнив действия:
Пуск→ Программы → Microsoft Access (или выполнив щелчок на соответствующей пиктограмме на панели Microsoft Office).

Для создания новой базы данных выполнить следующее:

- В окне Microsoft Access выбрать переключатель **Новая база данных**.
- В появившемся окне **Новая База данных** выбрать диск и открыть папку в которой будет создаваться новая БД. Затем в разделе **Имя файла** ввести имя создаваемой БД ОТДЕЛ КАДРОВ, выполнить щелчок по кнопке **«Создать».**
- В появившемся на экране окне БД ОТДЕЛ КАДРОВ выбрать вкладку **Таблицы**, затем – кнопку **«Создать»**.
- В окне **Новая таблица** выбрать режим создания таблицы **Конструктор**.
- Создать структуру таблицы СОТРУДНИК: В окне **Конструктора таблиц** заполнить соответствующие разделы: Имя поля, Тип данных, Описание. Для перехода от раздела к разделу использовать клавишу.
- В разделе **Тип данных** для изменения типа раскрыть окно выбора типа, выполнив щелчок по кнопке раскрытия списка, затем выполнить щелчок в строке, содержащей соответствующий тип.
- При этом в нижней части экрана в разделе **Свойства поля** появляется информация о данном типе поля. При необходимости туда можно вносить изменения, выполнив щелчок в соответствующей строке, удалив предыдущее значение, введя новое. Дополнительно можно задать формат поля, условие на значение и т.д.
- После создания структуры таблицы необходимо задать ключевое поле. Обычно, поле, используемое в качестве ключевого, располагается в таблице первым. Для создания ключевого поля выделить поле, выполнив щелчок слева от имени поля на полосе выделения. Выполнить ***Правка → Ключевое поле*** или выполнить щелчок по пиктограмме **Ключевое поле**. Слева от имени поля появится изображение ключа.
- Поле **Должн** выбрать в качестве индексированного. Для этого в разделе **Свойство поля** выбрать строку **Индексированное поле**.

Выполнить щелчок по кнопке раскрытия списка и выбрать строку *Да (Допускаются совпадения)*.

- После создания структуры таблицы сохранить ее. Выбрать ***Файл→ Сохранить*** или **Сохранить как...** В окне **Сохранение объекта** выбрать **В текущей базе данных**, затем ввести имя для сохранения созданной таблицы: СОТРУДНИК.

Аналогичным образом создать структуру таблицы СОСТАВ СЕМЬИ.

- Для поля **Идент код** целесообразно выбрать тип Мастер подстановок. Это позволит облегчить заполнение данными этого поля. После выбора типа Мастер подстановок откроется первое диалоговое окно **Создание подстановки**. В этом окне выбирается способ, которым столбец подстановки получит свои значения: из таблицы или запроса. Затем щелчок по кнопке **«Далее»**. В следующем диалоговом окне выбирается таблица, содержащая столбец подстановки. Затем щелчок по кнопке **«Далее»**.В следующем окне выбирается поле, используемое в качестве столбца подстановки и щелчком по кнопке « > »переносится в окно **Выбранные поля**. Щелчок по кнопке**«Далее»**. Следующее окно содержит сообщения о том, какие действия выполнить со столбцом в случае необходимости. Затем щелчок по кнопке**«Далее».** В следующем окне выполнить щелчок по кнопке **«Готово»**. Появится сообщение о том, что перед созданием связи необходимо сохранить таблицу. Для этого выполнить щелчок по кнопке **«Да»**.
- В разделе **Тип данных** будет указан тип **Текстовый**, т.е. тип, соответствующий типу поля подстановки из таблицы СОТРУДНИК.
- В таблице СОСТАВ СЕМЬИ необходимо выбрать поле **Идент код** в качестве индексированного поля. Для этого в разделе **Свойство поля** выбрать строку **Индексированное поле**. Выполнить щелчок по кнопке раскрытия списка и выбрать строку*Да (Допускаются совпадения)*.
- При сохранении таблицы отказаться от создания ключевого поля.

Создать структуру таблицы ШТАТНОЕ РАСПИСАНИЕ, включив в нее указанные поля и выбрав для них соответствующие типы.

- В качестве ключевого поля выбрать поле **Должн**.

Заполнение таблиц

Заполнение таблиц целесообразно начинать с таблицы ШТАТНОЕ РАСПИСАНИЕ, так как поле **Должн** этой таблицы используется в

качестве столбца подстановки для заполнения соответствующего поля таблицы СОТРУДНИК.

- В окне **Базы данных** выбрать нужную таблицу
- Выполнить щелчок по кнопке **«Открыть»**
- На экране появится структура БД в табличном виде
- Заполнение производится по записям, т.е. вводится информация для всей строки целиком
- Переход к следующему полю осуществляется нажатием клавишиTab.
- При заполнении первой строки следом за ней появится новая пустая строка.

Для заполнения поля МЕМО в таблице СОТРУДНИК нажать комбинацию клавиш Shif+F2, предварительно установив курсор в поле МЕМО. После ввода или редактирования данных в этом окне щелкнуть по кнопке **«ОК»**.

Для заполнения данными поля **Должн** в таблице СОТРУДНИК использовать список поля подстановки, раскрывая его щелчком мыши по кнопке раскрытия списка. Выбор нужной должности производится щелчком мыши в соответствующей строке.

Аналогично заполняется данными поле **Идент код** в таблице СОСТАВ СЕМЬИ.

После заполнения таблиц данными установить связь между таблицами

- Выбрать команду ***Сервис*** → ***Схема данных*** или выбрать пиктограмму **Схема данных**. Появится окно **Схема данных**, содержащее диалоговое окно **Добавление таблицы**. Выбрать таблицу СОТРУДНИК, затем выполнить щелчок на кнопке **«Добавить»**, для добавления таблицы в окно **Схема данных**. Повторить действие для каждой таблицы, участвующей в установке связи. Для создания связей между таблицами СОТРУДНИК и СОСТАВ СЕМЬИ поместить поле **Идент код** из таблицы СОТРУДНИК, на соответствующее поле таблицы СОСТАВ СЕМЬИ, появится диалоговое окно **Связи**. Для автоматической поддержки целостности БД установить флажок **Обеспечение целостности данных**. Установить также флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей**. Выполнить щелчок на кнопке **«Создать»**. В окне **Тип отношений** будет указан тип **один-ко-многим**. Соответствующим образом связать таблицы СОТРУДНИК и ШТАТНОЕ РАСПИСАНИЕ, выбрав для связи поле **Должн** и установив те же флажки. Затем

закрыть окно **Связи**. При запросе на сохранение связи выполнить щелчок на кнопке **«Да»**.

В окне БД выбрать таблицу ШТАТНОЕ РАСПИСАНИЕ, открыть ее в режиме **Таблицы**. Удалить должность «специалист». Открыть таблицы СОТРУДНИК и СОСТАВ СЕМЬИ и убедиться в том, что из таблиц исчез сотрудник, занимавший эту должность, а также сведения о его семье.

Восстановить удаленную информацию во всех таблицах.

В качестве отчета представить заполненные таблицы: Таблица-объект СОТРУДНИК; Таблица-объект СОСТАВ СЕМЬИ; Таблица-объект ШТАТНОЕ РАСПИСАНИЕ

Выводы

При выполнении лабораторной работы были изучены следующие вопросы: создание БД, создание структуры таблиц, создание индексных и ключевых полей, заполнение таблиц, установка связей между таблицами, каскадное удаление

3.5.2. Лабораторная работа №2
Работа с запросами

Цель работы

Получение навыков работы по созданию запросов.

Темы для предварительного изучения

- Назначение запросов, типы запросов. Создание запросов на выборку из одной или нескольких таблиц. Типы запросов на изменение и технология их разработки

Задание

Создать запрос на выборку информации из основной таблицы, из связанных таблиц, создать параметрический запрос, запрос для выбора информации для создания сложного отчета.

Используется база данных, созданная по индивидуальному заданию в работе №1.

Пример выполнения лабораторной работы

Создание запроса-выборки

Создать запрос, содержащий поля: Идент. код, Фамилия, Имя, Отчество, Дата рождения, включающий только тех сотрудников, фамилии которых начинаются с буквы «С». Список должен быть отсортирован по дате рождения по возрастанию.

Для этого необходимо выполнить следующую последовательность действий:

- При выбранной вкладке **Запросы** выполнить щелчок по кнопке **«Создать».** Открывается окно **Новый запрос**, в котором выбрать режим создания запроса **Конструктор**;
- Открывается окно **Запрос1: запрос на выборку**, а затем активизируется окно **Добавление таблицы**, в котором выбрать из списка таблиц таблицу **Сотрудник** щелчком мыши по имени таблицы, а затем выполнить щелчок по кнопке **«Добавить»**, после чего закрыть окно **Добавление таблицы**;
- Окно **Запрос1: запрос на выборку**, состоит из 2-х частей: В верхней части размещаются выбранные таблицы или запросы, на базе которых строится запрос;
  В нижней части расположен бланк построения запроса **QBE** (Query by example - запрос по образцу);
- При помощи мыши переместить нужные поля из выбранной таблицы и задать способы сортировки и условия отбора из таблицы. Для этого:

– выделить поля для запроса при помощи мыши в комбинации с клавишами Shift или Ctrl и отбуксировать на бланк построения запроса **QBE**. Поля можно перемещать в бланк **QBE** и по одному;
– в строке *Поле* поля размещаются по столбцам слева направо;
– в строке *Имя таблицы* отображается имя таблицы, из которой выбрано поле;
– в строке *Сортировка* в столбце поля **Дата рождения** установить сортировку по возрастанию. Для этого выполнить щелчок мышью в строке *Сортировка* в столбце поля **Дата рождения**, при этом появляется кнопка со стрелкой, нажатие на которую раскрывает окно выбора типа сортировки. Выбрать тип сортировки *по возрастанию*;
– В строке *Вывод на экран* можно отключить вывод поля на экран, убрав флажок для соответствующего поля;
– В строке *Условие отбора* в столбце **Фамилия** ввести условие *Like "С*"*. Перед вводом буквы*С* перейти на русский шрифт;
– Окончательный вид окна конструктора запросов будет иметь вид, представленный на рисунке 2.1.

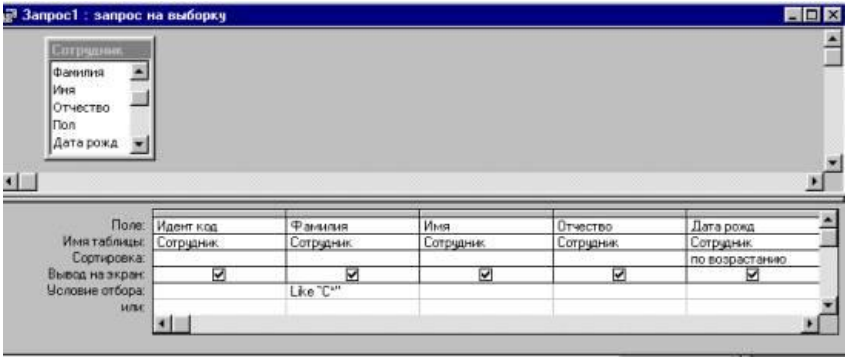


Рис. 2.1.Пример создания запроса на выборку

- Закрыть окно конструктора запроса и ввести имя запроса **fam_c** в ответ на вопрос сохранить изменения или нет. В окне базы данных при выбранной вкладке **Запросы** появится запрос с именем **fam_c**.

Выполнение запроса на выборку

- Выполнить щелчок мышью по запросу **fam_c**, затем по кнопке **«Открыть»**. На экран выводится таблица, в которой должны отображаться 3 записи с фамилиями, начинающимися на букву*С*, записи отсортированы по дате рождения по возрастанию.
- В случае, если не получен ожидаемый результат, внести изменения в запрос **fam_c**. Выполнить щелчок мышью по запросу **fam_c**, затем по кнопке **«Конструктор»**. Внести необходимые изменения, сохранить запрос, повторить его выполнение.

Создание параметрического запроса

Параметрическими называются запросы, представляющие собой варианты базового запроса и незначительно отличающиеся друг от друга.

Создать запрос, в результате выполнения которого будет выводиться Фамилия, Имя, Отчество и Идентификационный код определенного сотрудника.

При выбранной вкладке **Запрос** выполнить щелчок по кнопке **«Создать»**

- Открывается окно **Новый запрос**, в котором выбрать режим создания запроса **Конструктор**;

- Открывается окно **Запрос2: запрос на выборку**, а затем активизируется окно **Добавление таблицы**, в котором выбрать из списка таблиц таблицу **Сотрудник** щелчком мыши по имени таблицы, а затем выполнить щелчок по кнопке **«Добавить»**, после чего закрыть окно **Добавление таблицы**;
- При помощи мыши переместить нужные поля из выбранной таблицы;
- В столбце **Фамилия** в строке *Условие отбора* ввести в квадратных скобках *[Введите фамилию]* (сообщение, которое будет выводиться на экран при выполнении запроса);
- Выбрать из меню **Запрос** подпункт **Параметры**. В появившемся окне **Параметры запроса** в столбце **Параметр** ввести то же сообщение без квадратных скобок. В столбце **Тип данных** выбрать тип **Текстовый**;
- Закрыть запрос, на вопрос о сохранении ответить положительно, сохранить запрос с именем **Идент код;**
- Созданный запрос будет иметь вид, представленный на рисунке 2.2.

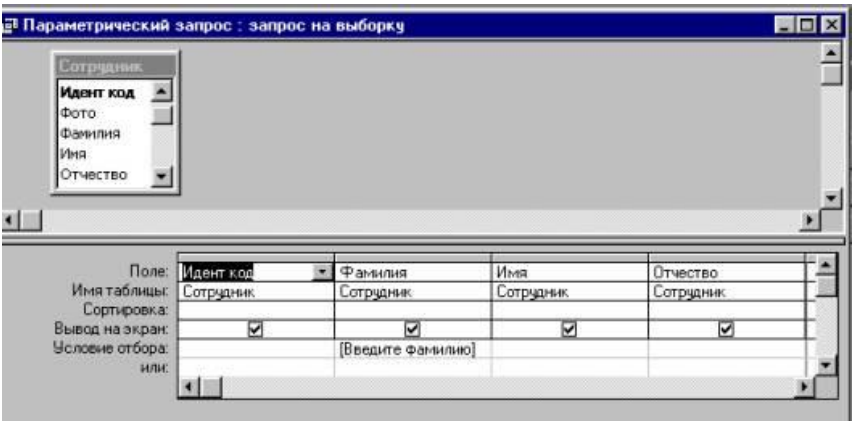


Рис. 2.2. Пример создания параметрического запроса

- Выполнить запрос, выполнив щелчок по кнопке **«Открыть».** В появившемся на экране окне **Введите значение параметра** ввести фамилию сотрудника, информацию о котором необходимо получить;
- На экране появится таблица с данными о выбранном сотруднике. Завершив просмотр, закрыть окно.

Создание запроса, используемого для создания сложного отчета

При выбранной вкладке Запрос выполнить щелчок по кнопке **«Создать»**

- Открывается окно **Новый запрос**, в котором выбрать режим создания запроса **Конструктор**;
- Открывается окно **Запрос3: запрос на выборку**, а затем активизируется окно **Добавление таблицы**, в котором выбрать из списка таблиц таблицу **Сотрудник** щелчком мыши по имени таблицы, а затем выполнить щелчок по кнопке **«Добавить»**, аналогично добавить таблицы **Состав семьи** и **Штатное расписание**, после чего закрыть окно **Добавление таблицы**;
- Из первой таблицы выбрать поля **Фамилия**, **Имя**, **Отчество**, **Должность**. При помощи мыши переместить на бланк запроса;
- Из второй таблицы выбрать поле **Идент код**, поместить его на бланк запроса;
- Из третьей таблицы выбрать поля **НазвПодр** и **Оклад** и также поместить их на бланк
- Добавить в бланк раздел *Групповая операция* (***Вид→Групповые операции***) или выполнив щелчок по соответствующей пиктограмме  .
- В столбце Идент код выполнить щелчок мышью в строке *Групповая операция*, появится кнопка раскрытия списка. Выполнить по ней щелчок мышью и выбрать из списка функцию ***Count*** (количество записей с одинаковыми значениями для данного поля). Окно запроса будет иметь вид, представленный на рисунке 2.3.

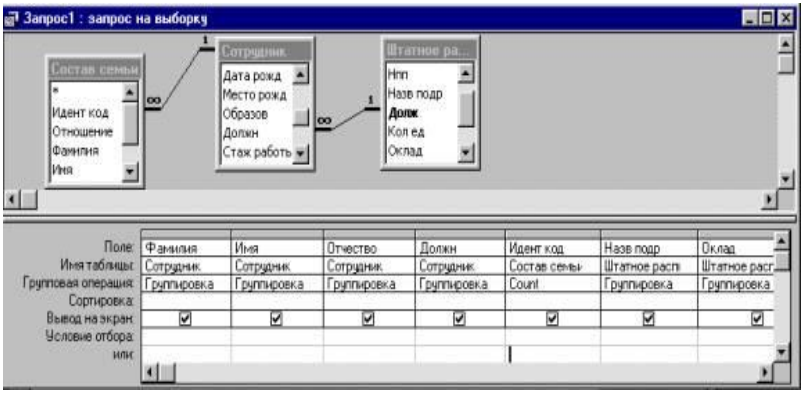


Рис. 2.3. Пример создания запроса для сложного отчета

- Закрыть запрос, выполнив его сохранение под именем **Запрос для отчета**.
- Выполнить запрос.
- На экране появится таблица с данными по выбранным критериям. Завершив просмотр, закрыть окно.

3.5.3. Лабораторная работа №3
Создание отчетов

Цель работы

Получение навыков работы по созданию отчетов.

Темы для предварительного изучения

- Порядок создания отчетов. Разделы бланка отчетов. Элементы управления. Форматирование элементов управления. Сортировка и группировка. Сохранение и печать отчета.

Задание

Создать простой отчет, отображающий результаты обработки информации для *Предметной области*, выбранной в соответствии с вариантом задания.

Создать сложный отчет на основе ранее созданного запроса (в лабораторной работе №2).

Пример выполнения лабораторной работы

Создание простого отчета

Создать простой отчет, содержащий: список сотрудников предприятия с полями: **Идент код**, **Фамилия**, **Имя**, **Отчество**, **Телефон.** Сгруппировать данные по 1-ой букве фамилии и вычислить количество фамилий в каждой группе.

1. Открыть базу данных, для которой создается отчет;
2. Начать создание отчета в режиме **Конструктор**:

Выбрать вкладку Отчеты, нажать кнопку **«Создать»**.Появляется окно **Новый отчет**, в котором выбрать **Конструктор** (для самостоятельного создания отчета), указать источник данных - таблицу **Сотрудник** (выбрать таблицу из раскрывающегося списка с перечнем таблиц и запросов). Нажать **«ОК»**.

На экран выводятся окна, представленные на рисунке 3.1:

a) *Окно отчета в режиме конструктора* с заголовком **Отчет1: отчет** и со следующими областями: **Верхний колонтитул**; **Область данных**; **Нижний колонтитул**.

b) *Панель элементов*, содержащая кнопки для создания элементов управления, которые можно включить в отчет. Панель элементов можно закрыть или вывести, выполнив щелчок по кнопке **«Панель элементов»**на панели инструментов или выполнив команду ***Вид→Панель элементов;***

c) *Список полей* базовой таблицы или запроса (список полей таблицы **Сотрудник**). Список полей можно вывести или закрыть, выполнив команду ***Вид→Список полей*** или выполнив щелчок по кнопке **«Список полей»** на панели инструментов.

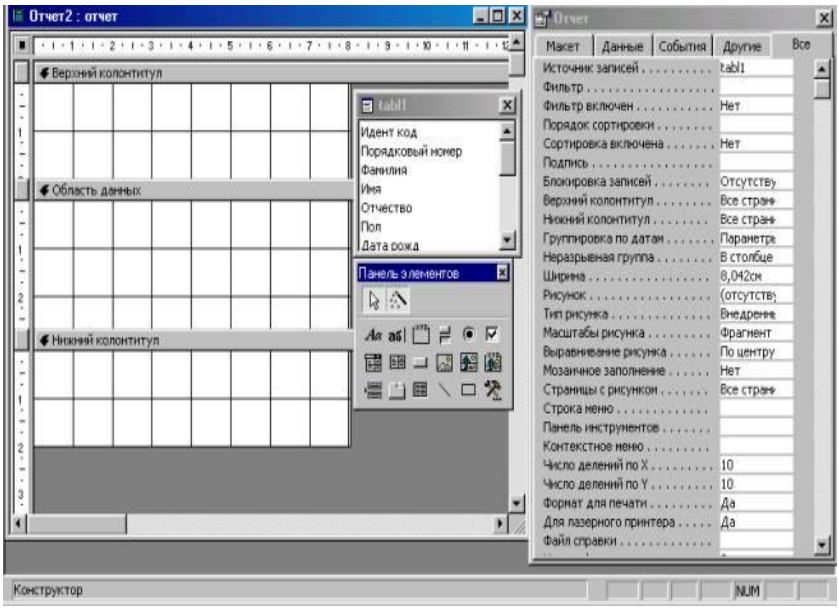


Рис. 3.1 Пример создания отчета

Перемещая окна, можно расположить их в удобном для работы порядке, например: **Окно отчета** - слева, **Список полей** в правой области экрана, ниже -**Панель элементов**. Дополнительно можно вывести окно свойств создаваемого отчета (***Вид→Свойства***) или выполнив щелчок по пиктограмме **Свойства**.

3. Установить размеры отчета

- Переместить правую границу окна создания отчета с помощью указателя мыши так, чтобы на верхней линейке было видно число 19 (размер отчета 18 см);
- Выполнить ***Файл→Параметры страницы***:
- При выбранной вкладке **Страница** установить книжную ориентацию листа и размер А4 (210х297);
- При выбранной вкладке **Поля** установить размеры левого и правого поля по 10 мм;
- При выбранной вкладке **Столбцы** установить: количество столбцов-1; ширина столбца-18 см; высота - 3 см; **«ОК»**;
- Переместить правую границу области данных отчета до значения 18 на верхней линейке;

4. Добавить в бланк отчета области **Заголовок отчета** и **Примечание отчета**:

- Для этого выполнить ***Вид→Заголовок/Примечание отчета***.

5. Переместить из таблицы в **Область данных** список нужных полей.

- В окне таблицы **Сотрудник** выделить в комбинации с клавишей ***Ctrl*** поля **Идент код**, **Фамилия**, **Имя**, **Отчество**, **Телефон** и отбуксировать их в **Область данных**. В **Области данных** появятся связанные элементы управления, т.е. элементы, связанные с полями таблицы **Сотрудник** (слева - подпись, справа - значение поля). Выполнить щелчок мышью на свободном пространстве в области данных, чтобы убрать выделение вставленных элементов управления;
- Можно просмотреть содержимое отчета на данном этапе, выбрав ***Файл→Предварительный просмотр***. В дальнейшем можно использовать эту команду для просмотра содержимого отчета после внесения каких-либо изменений;
- Переместить заголовки столбцов в область **Верхний колонтитул** для этого:
- Выделить подписи элементов управления (слева) в **Области данных**, для чего нажать клавишу ***Shift*** и выполнить щелчок на каждой подписи (или обвести их слева направо с нажатой левой кнопкой мыши). Выполнить команду **Вырезать**;

- Активизировать **Верхний колонтитул** щелчком мыши по заголовку и выполнить команду **Вставить**. Подписи будут вставлены в область **Верхнего колонтитула**;
- Расставить заголовки столбцов следующим образом: подпись **Идент код** переместить в левый верхний угол области. Остальные подписи расставить так, чтобы расстояние между левыми границами подписей было равно 3см;
- Выполнить редактирование и форматирование заголовков столбцов (в соответствии с рисунком 3.1). Для этого выделить все подписи в строке (поместить курсор мыши слева от строки, чтобы курсор принял форму стрелки, направленной вправо и выполнить щелчок мышью), щелкнуть правой кнопкой мыши на выделении, в появившемся окне выбрать команду **Свойства** и установить во вкладке **Макет** следующие значения: Ширина 3см, Высота 1см, Тип границы Сплошная, Размер шрифта 12см. Для редактирования подписи выделить элемент, выполнить щелчок мышью на подписи и внести нужные изменения. Замечания: Форматировать можно и отдельный элемент, выделив только его. При форматировании можно использовать пиктограммы панели форматирования на панели инструментов, или команды меню ***Формат→Выровнять***, ***Формат→Размер***, или установить соответствующие свойства для данного элемента.
- Уменьшить размер области **Верхний колонтитул** по размеру высоты заголовков столбцов, добавив приблизительно 0,5 см, переместив границу следующей области Область данных вверх;
- Разместить поля в **Области данных** в ряд под заголовками соответствующих столбцов. Уменьшить размер **Области данных** мышкой переместив нижнюю границу области вверх. Отформатировать каждое поле в соответствии с рисунком 3.1;
- Просмотреть содержимое отчета на данном этапе, выбрав **Предварительный просмотр**. Внести изменения, если есть несоответствия рисунку 3.1 или ошибки.

6. Определить поля, по которым будет производиться группировка и сортировка данных:

- Выполнить щелчок по пиктограмме **Сортировка и группировка** панели инструментов или выбрать ***Вид→Сортировка и группировка***. Открывается окно **Сортировка и группировка**;
- в столбце **Поле/выражение** (левый столбец) открыть список полей и выбрать поле **Фамилия**;

- в столбце **Порядок сортировки** установить порядок сортировки (по возрастанию)
- Выполнить установку свойств в области **Свойства группы**:
  - Для **Заголовок группы** и **Примечание группы** установить значение **Да**. При этом в окне отчета появляется дополнительная область **Заголовок группы "Фамилия"** перед **Областью данных** и **Примечание группы** после **Области данных**;
  - Для группировки по первому символу установить в строке **Группировка** значение **По первым символам**;
  - В строке **Интервал** установить число начальных символов, по которым хотим образовывать группы, для группировки по одному первому символу это значение равно 1;
  - В строке **Не разрывать** установить значение **Полную группу**;
  - Закрыть окно **Сортировка и группировка**.

7. Вставить в область **Заголовок группы** бланка отчета текст "*Группа фамилий, начинающихся на букву* ", а затем должна выводиться буква, по которой создавалась очередная группа:

- Выделить область **Заголовок группы** в бланке отчета (щелкнуть на заголовке области);
- Из окна **Список полей** перетащить поле **Фамилия** в бланк отчета в область **Заголовок группы**.. Поместить указатель с изображением руки с вытянутым указательным пальцем на маркер, расположенный в левом верхнем углу левого поля (подпись) и отбуксировать это поле влевый верхний угол области **Заголовка группы**, отступив 0,25см слева. Аналогично переместить правое поле вправо на 8см от левой границы области.В левом поле набрать текст "*Группа фамилий, начинающихся на букву*" и нажать клавишу ***Enter***. Установить параметры форматирования: курсив, размер 10. Затем выполнить действия: ***Формат→Размер→По размеру данных***. Изменить размеры правого поля, перемещая маркер правой границы влево так, чтобы видна была одна буква Ф и немного следующая.
- В окне **Панель элементов** щелкнуть по кнопке **«Линия»**, переместить курсор в область **Заголовок группы** под набранный текст и провести линию, подчеркнув оба поля;

8. Просмотреть отчет, выбрав ***Файл→Просмотр***. Если видны 2 буквы фамилии, то уменьшить поле **Фамилия**, если где-либо не видно буквы, то поле **Фамилия** увеличить (как описано в предыдущем пункте).

9. Вставить в область **Примечание группы** бланка отчета текст "*Количество в группе* ", а затем должно выводиться количество фамилий, относящихся к данной группе:

- Создать элемент управления. Для этого выполнить щелчок на **Панели элементов** по кнопке **«аб|»**, а затем в области **Примечание группы** в месте расположения элемента. Появляется элемент, состоящий из 2-х частей. Переместить правую часть элемента вправо. В левую часть поля (подпись) ввести текст "*Количество в группе*". В правую часть элемента ввести формулу ***=Count ([Фамилия])***. Произвести форматирование.

10. Вставить рисунок в заголовок отчета:

- ***Вставка →Рисунок→Из файла.*** В открывшемся окне выбрать подходящий графический файл, выполнить щелчок по кнопке **«ОК».**

11. Вставить текст заголовка отчета и рядом дату создания отчета:

- На панели элементов выбрать кнопку **«аб|»**, переместить курсор в область **Заголовок отчета** справа от рисунка. Отодвинуть правое поле ближе к правой границе области. В левое поле ввести текст заголовка. В правом поле с надписью **Свободный** ввести формулу ***=Date()***. Отформатировать поля. Вставить номер страницы в области **Нижний колонтитул**:
- Аналогично добавить элемент **Номер страницы**: в левое поле ввести текст "*Страница*", а в правое ***=Page***. Сохранить отчет.

Создание сложного отчета

Создать сложный отчет, содержащий:список сотрудников по подразделениям, имеющих в составе семьи более 1 человека; суммарный оклад и средний по подразделениям; суммарный оклад по всему предприятию. Вид отчета представлен на рисунке 3.2.

В связи с тем, что создаваемый отчет использует информацию из всех трех таблиц базы данных, будем строить отчет на базовом запросе **Запрос для отчета**, созданном в предыдущей лабораторной работе. Действия аналогичны действиям, описанным в п.1 данной лабораторной работы при создании простого отчета.

1. Начать создание отчета в режиме **Конструктор**:

Выбрать вкладку **Отчеты**, нажать кнопку **«Создать»**. Появляется окно **Новый отчет**, в котором выбрать **Конструктор** (для самостоятельного создания отчета), указать источник данных - запрос **Запрос для отчета**. Нажать **«ОК».** Вид экрана - как и при создании отчета (рис.3.1)

2. Установить размеры отчета.

3. Добавить в бланк отчета области **Заголовок отчета** и **Примечание отчета**.

4. Определить поля, по которым будем группировать и сортировать данные: Установить группировку по полю **Назвподр** (название подразделения), порядок сортировки по возрастанию.

5. Добавить области **Заголовок группы** и **Примечание группы**;

6. Выбрать из окна запроса **Запрос для отчета** в **Область данных** список всех полей, кроме **Назвподр** и перенести заголовки колонок таблицы в **Верхний колонтитул**. Выполнить размещение и форматирование всех полей в областях **Верхний колонтитул** и **Область данных** в соответствии с рисунком 3.2.

7. Заполнить область **Заголовок группы.**

8. Выбрать из окна запроса **Запрос для отчета** в **Заголовок группы** поле **Назвподр** и изменить содержимое поля Подпись на **Название подразделения**. Выполнить форматирование полей в соответствии с рисунком 3.1.

9. Заполнить область **Примечание группы**:

- Переместить из списка полей запроса **Запрос для отчета** в **Примечание группы** поле **Назвподр** 2 раза, расположив связанные поля друг под другом. Изменить название полей подпись (левое поле) на **Суммарный оклад по подразделению** для первой строки и **Средний оклад по подразделению** для второй строки;
- Рядом с 1-ой и 2-ой строками создать по одному не заполненному полю, щелкнув сначала на Панели элементов по кнопке **«аб|»**, а затем в области **Примечание группы** в соответствующей строке. Убрать левые поля, выделив их и затем, щелкнув на кнопке **«Вырезать»** на панели элементов. В верхней строке в поле с надписью **Свободный** набрать формулу ***=Sum([Оклад])*** (для вычисления суммарного оклада по подразделению),в нижней - ***=Avg([Оклад])*** (для вычисления среднего оклада по подразделению).Разместить поля с формулами под заголовками соответствующих столбцов;
- Аналогично создать строку для вывода в отчет **Суммарного оклада по предприятию** в разделе **Примечание отчета**, создав сначала связанное поле и введя соответствующую подпись и формулу. Для поля с формулой в качестве значения свойства **Сумма с накоплением** установить значение, отличное **Отсутствует**.

10. Вставить в раздел **Заголовок отчета** текст заголовка отчета, а рядом дату и время создания отчета, вставить номер страницы. Для вывода даты и времени использовать в формуле функцию ***Now()***.
11. Выполнить форматирование полей в соответствии с рисунком 3.2. Просмотреть отчет. Сохранить отчет.

Список по подразделениям сотрудников, имеющих в составе семьи более 1 чел.:				28.10.18 13:06:00	
Фамилия:	*Имя:*	*Отчество:*	*Должность:*	*Оклад:*	*Сост семьи*
Название подразделения:			**Дирекция**		
Смирнова	Наталья	Николаевна	гл.бухгалтер	430	1
Козлова	Елена	Дмитриевна	начальник ОК	150	1
Безухов	Владимир	Петрович	зам.директора	500	1
Артюхов	Сергей	Иванович	директор	530	1
Суммарный оклад по подразделению:			**Дирекция**	**1610**	
Средний оклад по подразделению:			**Дирекция**	**402,5**	
Суммарный оклад по предприятию:				**1610**	
Название подразделения:			**уч. кафедра**		
Самороков	Дмитрий	Викторович	преподаватель	350	2
Тарченко	Нина	Борисовна	статистик	100	2
Садчиков	Аркадий	Викторович	диспетчер	100	1
Масловенко	Юрий	Николаевич	специалист	150	2
Броневой	Андрей	Иванович	преподаватель	350	1
Суммарный оклад по подразделению:			**уч. кафедра**	**1050**	
Средний оклад по подразделению:			**уч. кафедра**	**210**	
Суммарный оклад по предприятию:				**2660**	

Рис. 3.2 Пример сложного отчета

3.5.4. Лабораторная работа №4
Работа с формами

Цель работы

Получение навыков работы по созданию экранных форм.

Темы для предварительного изучения

- Назначение форм. Создание формы с помощью мастера форм. Создание простой формы в режиме Конструктора. Работа с элементами управления. Создание сложной формы на основе ранее созданных простых форм.

Задание

Создать форму для ввода информации в таблицы в удобном для пользователя формате.

Создать сложную форму, объединив формы, созданные для разных таблиц.

Пример выполнения лабораторной работы

Создать формы для заполнения таблиц: **Сотрудник**, **Состав семьи**, **Штатное расписание** (Предметная область из лабораторной работы №1).

1. Для создания формы **Сотрудник** выполнить следующие действия:

- В окне БД выбрать вкладку **Формы**. Выполнить щелчок по кнопке **«Открыть»**. Появится диалоговое окно **Новая форма**.
- Выбрать из списка пункт **Конструктор**. Затем в списке **Выберите в качестве источника данных таблицу или запрос** выбрать имя таблицы **Сотрудник**. Выполнить щелчок по кнопке **«ОК».**
- Если на экране отсутствует список полей выбранной для построения формы таблицы, выбрать пункт меню **Вид → Список полей**.
- Поля из списка переместить на форму (по одному или предварительно выделив с использованием клавиши ***Shift*** и мыши, для выделения всех полей выполнить двойной щелчок мышью на заголовке окна **Список полей**.
- Разместить поля на форме в нужных местах по прилагаемому образцу (рисунок 4.1).
- Перемещение полей и их имен по форме производиться следующим образом: Выделяется объект (поле с именем) щелчком мыши. Вокруг него появятся маркеры перемещения и изменения размеров. Перемещать поле можно вместе с привязанным к нему именем или отдельно от него. Для перемещения поместить указатель мыши на

квадратик, находящийся в левом верхнем углу элемента. Указатель мыши в виде ладони позволяет перемещать объект вместе с привязанным к нему именем, в виде ладони с вытянутым указательным пальцем - перемещает один объект. Для изменения надписи, связанной с полем необходимо выполнить на ней двойной щелчок мышью. В открывшемся диалоговом окне Надпись выбрать вкладку Макет, изменить значение у свойства Подпись. Затем закрыть окно. Для изменения размеров поместить курсор на размерные маркеры, при этом курсор примет вид двунаправленной стрелки. Нажать кнопку мыши, буксировать в нужном направлении, затем отпустить кнопку мыши. Для удаления поля выделить его, нажать клавишу **Delete** или выбрать команду ***Правка→Удалить***.

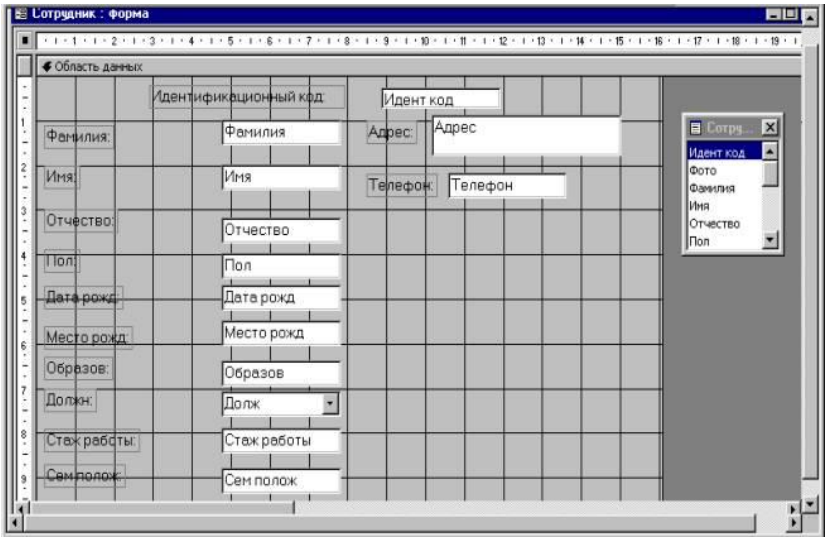


Рис. 4.1 Пример создания формы

- Сохранить форму, выбрав из меню ***Файл→Сохранить как*** в открывшемся окне выбрать режим сохранения **в текущей базе данных**, затем щелчок по кнопке **«ОК»**.
- Просмотреть форму в режиме **Формы**, выполнив щелчок по кнопке **«Открыть»**.
- Если вид формы не удовлетворяет, открыть форму в режиме **Конструктор** и внести необходимые изменения, затем сохранить

форму ***Файл→Сохранить*** или выполнить щелчок по пиктограмме **Сохранить**.

2. Аналогичным образом создать формы **Состав семьи** и **Штатное расписание**.

3. Создать объединенную форму, включающую две ранее созданные: **Сотрудник** и **Состав семьи**.

- В окне БД, при выбранной вкладке **Формы**, выбрать форму **Сотрудник**. Она будет основной. Выполнить щелчок по кнопке **«Конструктор».**
- Расположить окна БД и **Конструктора** с открытой формой **Сотрудник** таким образом, чтобы они не перекрывали друг друга.
- В окне БД выбрать форму **Состав семьи**. Переместить ее в окно формы **Сотрудник** на свободное место в правой части формы.
- Закрыть форму **Сотрудник**. Подтвердить сохранение в появившемся на экране окне.

4. Перейти к вкладке **Таблицы**, выбрать таблицу **Состав семьи**, удалить все записи из таблицы.

5. Заполнить таблицу **Состав семьи**, используя для заполнения созданную форму **Сотрудник**.

- В окне БД выбрать вкладку Формы, выбрать форму **Сотрудник**, выполнить щелчок по кнопке **«Открыть».**
- В открывшемся окне появится форма **Сотрудник**, содержащая первую запись таблицы **Сотрудник** (рисунок 4.2)
- Раздел формы **Состав семьи** будет незаполненным. Для заполнения выполнить щелчок в окне ввода первого элемента формы **Состав семьи**. Для перехода к следующему полю используется клавиша ***Enter***. Ввести информацию в следующее поле ит. д.
- Если в составе семьи сотрудника несколько человек, для перехода к следующей записи **Состав семьи** использовать кнопки, расположенные в нижней части формы. Кнопка со стрелкой вправо - движение вперед, со стрелкой влево - в обратном направлении. Заполнив информацию о всех членах семьи сотрудника, перейти к следующей записи таблицы Сотрудник, используя аналогичные кнопки в окне формы **Сотрудник**. Заполнить для него информацию о составе семьи и т.д., пока не будут внесена информация о составе семьи для всех сотрудников. Закрыть форму **Сотрудник**.

6. Просмотреть заполненную таблицу Состав семьи в режиме Таблица. Убедиться, что все записи, помещенные в таблицу верны. При необходимости внести изменения в данные таблицы. Закрыть таблицу. Подтвердить сохранение произведенных изменений.

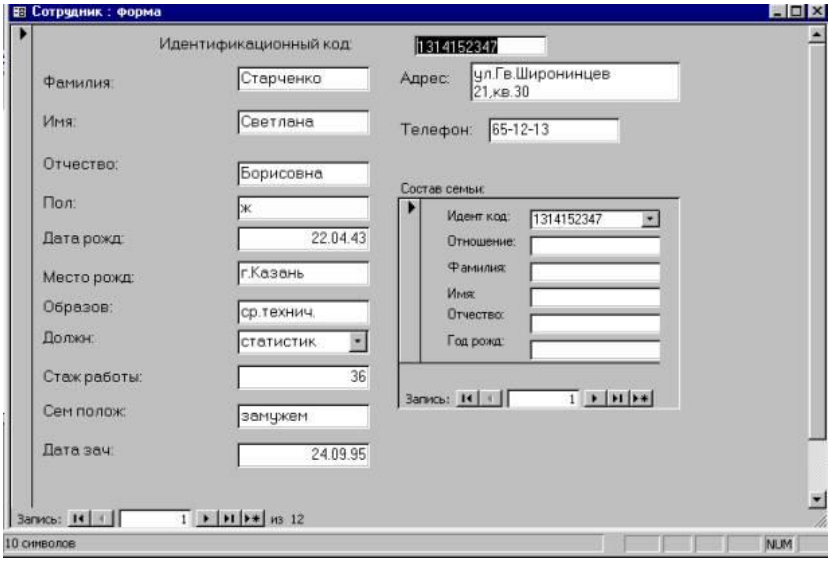


Рис. 4.2 Пример работы с формой

3.5.5. Лабораторная работа №5
Создание кнопочных форм

Цель работы

Получение навыков работы по созданию кнопочных форм.

Темы для предварительного изучения

- Создание кнопочной формы с помощью диспетчера кнопочных форм. Изменение существующей кнопочной формы.

Задание

Создать кнопочную форму для работы с созданными объектами базы данных (таблицы, отчеты, формы). Предусмотреть выход из БД.

Пример выполнения лабораторной работы

Выбрать в меню **Сервис** команду **Надстройки** и подкоманду **Диспетчер кнопочных форм**. Если выводится приглашение подтвердить создание кнопочной формы, выполнить щелчок по кнопке **«Да»** (рисунок 5.1).

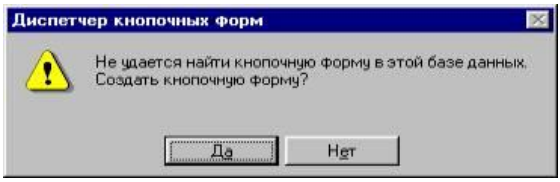


Рис. 5.1 Подтверждение создания кнопочной формы

В диалоговом окне **Диспетчер кнопочных форм** выполнить щелчок по кнопке **«Создать»** (рисунок 5.2).

В диалоговом окне **Создание** (рисунок 5.3) в поле **Имя страницы кнопочной формы** ввести имя кнопочной формы для ввода информации (например, Ввод и редактирование). Выполнить щелчок по кнопке **«ОК»**.

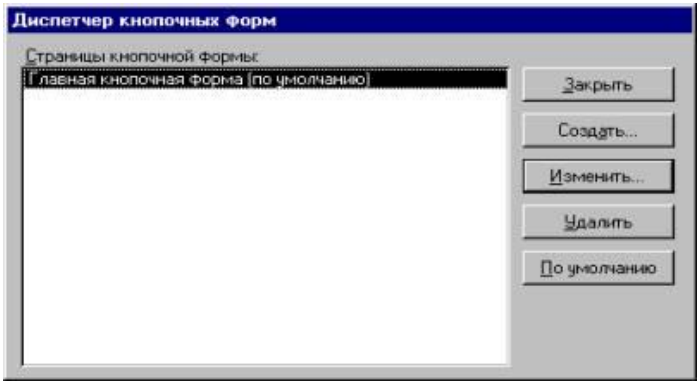


Рис. 5.2 Пример работы с Диспетчером кнопочных форм

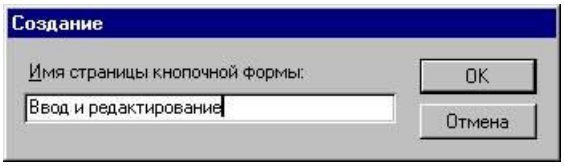


Рис. 5.3 Пример создания кнопочной формы

Аналогично создать кнопочные формы для просмотра документов и кнопку для выхода из БД. Выбрать строку **Ввод и редактирование** выполнить щелчок по кнопке **«Да»**.

Для изменения существующей кнопочной формы в диалоговом окне **Изменение страницы кнопочной формы** (рисунок 5.4) выполнить щелчок по кнопке **«Создать»**.

В диалоговом окне **Изменение элемента кнопочной формы** (рисунок 5.5) ввести текст подписи для первой кнопки - *Ввод информации* выбрать команду в поле со списком **Команда:** *Открытие формы в режиме добавления*. Если выбрана одна из команд работы с формами, то в поле со списком **Форма** следует выбрать имя открываемой формы (Ввод информации). Выполнить щелчок по кнопке **«ОК»**.

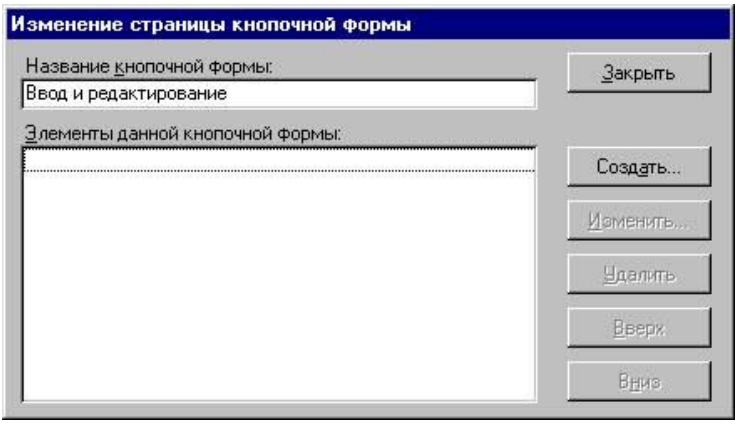


Рис. 5.4 Пример изменения страницы кнопочной формы

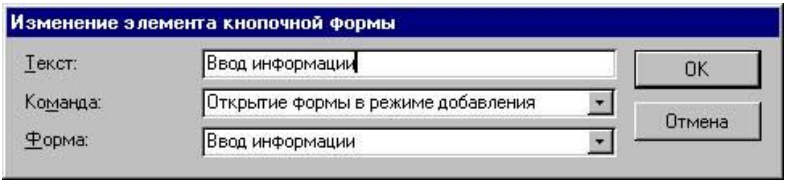


Рис. 5.5 Пример изменения элемента кнопочной формы

Повторить эти действия для создания всех нужных кнопок формы. Имена всех созданных элементов (кнопок) появятся в разделе **Элементы данной кнопочной формы**. Чтобы изменить или удалить какую-либо из созданных кнопок, выбрать имя этой кнопки в списке и нажать кнопку **«Изменить»** или **«Удалить».** При необходимости изменить порядок кнопок в списке выбрать элемент и нажать кнопку **«Вверх»** или **«Вниз»**.

3.5.6. Лабораторная работа №6
Работа с макросами

Цель работы

Получение навыков работы по созданию и выполнению макросов.

Темы для предварительного изучения

- Сущность макросов. Создание макросов. Выполнение макросов. Редактирование макросов

Задание

Создать макросы для индивидуального варианта базы данных (лабораторные работы №1-№5). Выполнить макросы.

Выполнение задания

Макрос - это набор инструкций, которые сообщают программе, какие действия следует выполнить, чтобы достичь определенной цели. Макрос объединяет все эти инструкции в одном сценарии, который затем можно вызвать с помощью команды меню, кнопки панели инструментов или комбинации клавиш.

В СУБД Microsoft Access макрос - определенная последовательность операций. Они хранятся в окне, напоминающем таблицу, в таком порядке, в котором их необходимо выполнять. При запуске макроса СУБД выполняет эти действия. Макрос может существенно облегчить использование Access.

Одной из полезных особенностей макросов является возможность их привязки к кнопкам, которые помещаются в формы. Эти кнопки могут выполнять часто повторяющиеся операции (например, открывать диалоговые окна для поиска, изменять порядок сортировки данных или печатать отчет). Пользователи СУБД Microsoft Access могут использовать определенные кнопки для выполнения этих действий, даже не зная всех подробностей операций, выполняемых макросом.

Используя макросы, можно создавать завершенные приложения с пользовательским меню и диалоговыми окнами.

Создание макросов

Для создания макроса необходимо открыть вкладку **Макросы** в окне БД и выполнить щелчок по кнопке **«Создать»**. Это же действие можно выполнить с использованием пунктов меню ***Вставка →Макрос*** Окно макросов делится на две части: верхнюю и нижнюю. В верхней части находится список макрокоманд, которые необходимо выполнить, и

необязательные примечания к этим командам. В нижней части окна находятся аргументы макрокоманды.

При разработке макроса необходимо задать действия (ввести макрокоманды), которые он должен выполнить (например, открыть форму, распечатать отчет, выполнить запрос и др.). В СУБД Microsoft Access такие действия можно определить двумя способами:

1. Выбрать из списка в столбце ***Макрокоманда*** (или ввести их вручную)
2. Переместить объекты из окна БД в столбец ***Макрокоманда*** окна макросов.

1-й способ:

- В окне макросов выполнить щелчок мышью на первой пустой ячейке в столбце ***Макрокоманда***. Затем выполнить щелчок по кнопке раскрытия списка (раскрывается список допустимых макрокоманд).
- Выбрать из списка команду, которую должен выполнить макрос или набрать эту команду вручную.
- Выполнить щелчок в нижней части окна или нажать клавишу `F6` и указать аргументы действия.
- При необходимости добавить комментарий в столбце ***Примечание.***

2-й способ:

- Переместить окно макросов и изменить его размеры таким образом, чтобы одновременно были видны окна макросов и базы данных.
- Выбрать вкладку объекта, который будет открываться макросом
- Выполнить щелчок на нужном объекте и переместить его в пустую строку столбца ***Макрокоманда*** окна макросов. После этого в столбце ***Макрокоманда*** появится соответствующая команда.
- В разделе ***Аргументы*** макрокоманды появятся аргументы. При необходимости их можно изменить.
- В столбце ***Примечание*** можно ввести любые комментарии, которые помогут проследить, какое действие выполняет макрос.

Сохранение макросов

Для сохранения макроса необходимо выполнить следующие действия: Выбрать команду ***Файл→Сохранить*** (или выполнить щелчок по пиктограмме ***Сохранить*** на панели инструментов). Если макрос сохраняется впервые, СУБД Access запросит для него имя. Ввести имя созданного макроса, выполнить щелчок по кнопке **«ОК»** и закрыть окно макросов, нажав комбинацию клавиш Ctrl+F4.

Выполнение макроса

После завершения разработки макроса можно проверить его работу, запустив макрос на выполнение.

Самый простой способ запуска:

Открыть вкладку ***Макросы*** в окне БД, выделить макрос и щелкнуть по кнопке «**Запуск»** или выполнить двойной щелчок на макросе в окне БД.

Другой способ запуска следующий: Выбрать команду ***Сервис→Макрос→Запуск макроса***. В появившемся диалоговом окне **Запуск макроса** выбрать или ввести имя макроса.

Если макрос открыт в режиме конструктора, его можно выполнить, щелкнув по кнопке «**Запуск»** на панели инструментов.

Кроме того, макросы можно выполнять при помощи кнопок, которые добавляются в формы. Кнопки макросов можно создать методом перетаскивания:

- Открыть нужную форму в режиме конструктора, переместить форму и установить ее размеры таким образом, чтобы можно было видеть и окно БД.
- Открыть вкладку **Макросы** в окне БД для отображения макросов.
- Перетащить нужный макрос в то место формы, где необходимо поместить кнопку. Созданная кнопка появится в окне формы.

Макрос можно настроить таким образом, чтобы он выполнялся автоматически при запуске БД. Для этого достаточно при сохранении созданного макроса присвоить ему имя **Autoexec**. Макросы **Autoexec** обычно используются для открытия форм, с которыми чаще всего работают пользователи, или для размещения на экране нескольких часто используемых форм и/или отчетов. Удерживая клавишу Shift во время открытия БД, можно отменить запуск макроса **Autoexec.**

Редактирование макросов

Структура таблицы в окне макросов напоминает структуру обычной таблицы БД. Команды редактирования текста, используемые для удаления, переноса и копирования содержимого ячеек, могут применяться и в рамках таблицы макроса. Редактирование макроса осуществляется в режиме Конструктора. Аргументы и краткое описание макрокоманды отображаются в окне макросов только при ее маркировке.

Например, для вставки дополнительной макрокоманды в существующий макрос необходимо выполнить следующие действия:

- Выбрать вкладку **Макросы** в окне БД;
- Выбрать макрос для редактирования, выполнив на нем щелчок мышью;
- Открыть макрос в режиме конструктора, выбрав кнопку **Конструктор**;

- Выбрать макрокоманду перед которой необходимо произвести вставку новой, выполнив щелчок мышью в одном из полей этой макрокоманды или маркировав всю строку целиком;
- Выбрать пункт меню **Вставка**, подпункт **Строки**. Перед маркированной строкой будет вставлена пустая;
- Поместить в эту строку новую макрокоманду;
- Сохранить макрос (***Файл→ Сохранить***).

Макросы можно копировать из одной БД в другую или в одну и туже БД под разными именами. Это экономит время при создании макросов, выполняющих похожие задачи. Существующий макрос можно скопировать, выполнив следующие действия:

- В окне БД выбрать нужный макрос.
- Выбрать команду ***Правка→ Копировать***.
- Для копирования макроса в другую БД закрыть текущую и открыть ту, в которую будет копироваться макрос. Выбрать в окне БД вкладку **Макросы**.
- Выбрать команду ***Правка→ Вставить.***
- В появившемся диалоговом окне ввести имя макроса. Если макрос копируется в ту же БД, в которой находится существующий макрос, то копируемому макросу необходимо присвоить другое имя.

3.5.7. Лабораторная работа №7
Создание SQL-запросов в среде MS Access

Цель работы

Получение навыков работы по созданию SQL-запросов в среде MS Access.

Темы для предварительного изучения

- Запросы SQL

Задание

Создать SQL-запрос для индивидуального варианта базы данных (лабораторная работа№1).

Выполнение задания

Для создания запроса необходимо перейти на вкладку «запросы» в вашей БД и выполнить «Создание запроса в режиме конструктора». В появившемся диалоге добавления таблиц в конструкторе запросов нажать

«Закрыть», а затем на панели «Конструктор запросов» выбрать «Вид» запроса – SQL (рис. 7.1). Переключаться между видом конструирования запроса можно, используя контекстное меню.

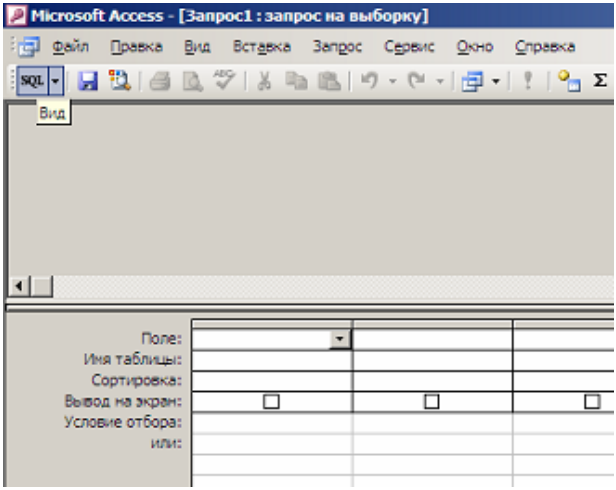


Рис.7.1 Переключение в режим SQL

После выполнения этих действий вам будет предложено окно для ввода текста запроса. По окончанию ввода можете выполнить запрос для просмотра результатов, используя кнопку на панели «Конструктор запросов» (рисунок 7.2).

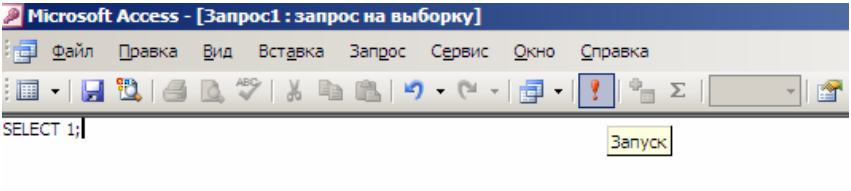


Рис. 7.2 Запуск запроса SQL

Запросу можно передать параметр. Для этого надо в тексте запроса написать [param] (вместо param может быть любое слово, записанное на кириллице или латинице, кроме служебных слов SQL и имен полей используемых отношений) – при выполнении запроса будет запрошен ввод параметра. Несколько параметров с одинаковым именем считаются одним.

4. ЯЗЫК SQL

4.1. Введение в SQL

SQL – это язык запросов к базам данных, он применяется для осуществления доступа к данным, для запросов к реляционным СУБД, для управления базами данных и их обновления. Стандарт SQL утвержден Американским Национальным Институтом Стандартов (ANSI, American National Standards Institute) и Международной организацией по стандартизации (ISO, International Organization for Standardization). ANSI – это организация промышленных и деловых групп, разрабатывающая стандарты для торговли и коммуникации в Соединенных Штатах. ANSI также является членом ISO и IEC (Международной электротехнической комиссии, International Electrotechnical Commission). ANSI публикует стандарты США, которые соответствуют международным стандартам. В 1992 году ISO и IEC опубликовали международный стандарт на SQL, который принято называть SQL-92. ANSI опубликовал в Соединенных Штатах соответствующий стандарт, ANSI SQL-92, который иногда называют ANSI SQL. Хотя разные реляционные СУБД применяют несколько различающиеся версии SQL, но большинство из них соответствуют стандарту SQL-92, определенному ANSI.

В SQL Server 2000 применяется диалект Transact SQL (T-SQL). T-SQL – это усовершенствование стандартного языка программирования SQL. Первоначальный, основной SQL применяется для взаимодействия между приложениями и SQL Server. В T-SQL имеются вес возможности языков DDL и DML стандартного SQL, а кроме этого имеются также расширенные функции, системные хранимые процедуры и конструкции для программирования (такие, как IF and WHILE), обеспечивающие гораздо большую гибкость программирования.

Язык SQL содержит операторы; относящиеся к одному из двух основных языков программирования в составе SQL: DDL и DML. ***Язык DDL*** (data definition language, *язык определения данных*) применяется для определения объектов баз данных (таких как базы данных, таблиц и представлении) и для управления этими объектами. Операторы языка DDL обычно включают в себя команды CREATE, ALTEB и DROP для каждого из объектов, с которым производится работа. ***DML*** (data manipulation language, *язык манипулирования данными*) применяется для манипулирования данными, содержащимися в объектах базы данных, для чего применяются такие операторы, как SELECT, INSERT, UPDATE и DELETE. При помощи этих операторов можно соответственно выбирать строки с данными, вставлять новые строки, изменять имеющиеся и удалять ненужные.

Все ключевые слова SQL можно разделить на следующие категории.

Команды. Представляют собой глаголы, определяющие действие, которое следует
выполнить. Например, SELECT, CREATE И ALTER.

Условия, или *квалификаторы*. Ограничивают диапазон значений элементов, входящих в запрос, например, WHERE.

Модификаторы, или *предложения*. Модифицируют выполнение инструкции, например, ORDER BY.

Предикаты. Представляют собой выражения, такие как IN, ALL, ANY, SOME, LIKE и UNIQUE. Предикаты могут возвращать в качестве результата значения TRUE, FALSE и в некоторых случаях NULL (неизвестный результат). Эти три значения являются ключевыми словами SQL.

Операторы. Такие операторы, как =, < или >, сравнивают значения и применяются для создания объединений в синтаксисе предложений WHERE или JOIN. Операторы также называют предикатами сравнения.

Статистические функции (также называемые агрегатными). Возвращают одно результирующее значение, вычисленное на основании набора данных например COUNT(), MAX() и MIN().

Функции преобразования типа данных. Изменяют тип данных значения с одного на другой. Наиболее часто используемые функции преобразования – это CAST() и CONVERT().

Другие ключевые слова (или зарезервированные слова), изменяющие действие команд или управляющие курсором (указателем текущей записи в наборе результатов запроса), с помощью которого выбираются отдельные строки запроса. Например, модификатор FOR XML [AUTO | RAW | EXPLICIT] языка T-SQL возвращает XML документ или подчиненный XML-документ вместо традиционного набора данных запроса на выборку (SELECT). Модификатор FOR XML не включен в ANSI SQL.

Синтаксис языка SQL использует специальные символы и знаки пунктуации.

Запятые используются для разделения компонентов списка параметров. Так, например, перечень имен нескольких полей будет выглядеть следующим образом: Название, Адрес, Город, Индекс.

Квадратные скобки, в которые заключаются имена полей, необходимы только в том случае, когда в имени поля имеются пробелы или другие знаки пунктуации, не разрешенные в ANSI SQL, например, [Название Компании]. Квадратные скобки могут также применяться для помещения в них имен, которые присваивают в бланке запроса вводным

параметрам для функций и сохраненных (хранимых) процедур SQL Server.

Точки используются для отделения имен объектов зависимого класса. Например, если в запрос включены поля из нескольких таблиц, точка используется для отделения имен таблиц от имен полей – Клиенты. [Название Компании]. Состоящие из четырех частей имена связанных таблиц в инструкциях FROM используют формат: Сервер. БазаДанных. Схема. Таблица.

Идентификаторы строки (также называемые *разделителями*) указывают значения символьных констант.

В ANSI SQL требуется заключать значения строковых литералов в одинарные кавычки ('). Для обратной совместимости с SQL Server 7.0 и более ранними версиями язык T-SQL интерпретирует двойные кавычки как квадратные скобки.

Символы подстановки. В ANSI SQL символы подстановки % и _ (подчеркивание) используются в инструкции LIKE.

Идентификатор даты/времени. ANSI SQL принимает заключенные в одинарные кавычки значения даты в различных символьных форматах. Символ # в ANSI SQL не используется.

Идентификаторы : и @ для переменных. ANSI SQL использует двоеточие (:) в качестве префикса для идентификации переменных, принимающих значения параметра. T-SQL использует символ @ для традиционных переменных (включая переменные, принимающие значения параметров) и @@ для переменных, чьи значения предоставляются SQL Server, таких как @@IDENTITY, которая возвращает текущее значение идентификации столбца таблицы.

Символ ! (восклицательный знак) используется как синоним NOT в ANSI SQL. В качестве оператора "не равно" в ANSI SQL используется комбинация.

4.2. Создание запроса на выборку на языке SQL в проекте

Структура запроса:

```
SELECT    [ALL | DISTINCT]   [TOP n [PERCENT]  ]  список_полей,
FROM      имена_таблиц
[WHERE    критерий_отбора]
[GROUP BY   группируемые_поля]
[HAVING   условие_для_результата ]
[ORDER BY   столбцы_сортировки   [ASC | DESC]  ]
```

Данный запрос возвращает таблицу, содержащую выбранные столбцы (поля) из указанных строк (записей) таблицы-источника. В квадратных скобках указаны необязательные параметры.

Назначение всех элементов этой структуры запроса на выборку:

SELECT. Основная команда, определяющая запрос на выборку. В параметре *список_полей* указываются поля (столбцы), которые должны быть включены в результирующую таблицу запроса. При указании нескольких имен полей они разделяются запятыми.

Необязательные предикаты ALL и DISTINCT определяют способ отбора строк. При использовании первого из них в таблицу включаются все строки, соответствующие указанным далее условиям. Предикат DISTINCT исключает строки с повторяющимися данными.

Необязательный модификатор TOP n [PERCENT] ограничивает результирующий набор записей запроса, возвращая только первые n записей или n процентов набора записей, удовлетворяющих запросу. TOP n PERCENT являются зарезервированными словами T-SQL, а не ANSI SQL. Модификатор TOP можно также использовать для ускорения вывода в том случае, когда необходимо отобразить только наиболее важные строки результирующего набора.

FROM имена_таблиц. Определяет имена таблиц, из которых запрос должен отобрать данные. Имена таблиц разделяются запятыми. Перед именами полей, представленными в нескольких таблицах, необходимо добавлять имя таблицы.

WHERE критерий_отбора. Определяет условие для отбора записей указанных таблиц.

Компонент WHERE не обязательный, поэтому, если его не добавить, запрос возвратит все строки из таблицы, определенной элементом FROM имена_таблиц.

GROUP BY условие_группировки. Предложение GROUP BY применяется после предложения WHERE и означает, что строки набора результатов должны быть сгруппированы в соответствии с данными в колонке группировки. Если в предложении SELECT используется агрегатная функция, то для каждой группы вычисляется и отображается в выводе итоговое агрегатное значение. Агрегатная функция выполняет вычисления и возвращает значение. В предложении GROUP BY в качестве колонок группировки должны быть заданы все колонки из списка выборки (кроме колонок, применяемых для агрегатных функций), в противном случае SQL Server выдаст сообщение об ошибке. Если бы это правило не соблюдалось, результаты нельзя было бы выдать в разумном виде, поскольку колонка, заданная в GROUP BY, должна группировать каждую колонку в списке выборки.

HAVING условие_для_результата. Одно или несколько условий, налагаемых на конечный результат, полученный после выполнения группировки, вычисления или применения агрегатной функции. Условие HAVING является необязательным.

ORDER BY столбцы_сортировки. Определяет порядок сортировки записей. Так же, как и предложение WHERE, ORDER BY, не является обязательным компонентом инструкции. Используя ключевые слова ASC или DESC, можно определить сортировку по возрастанию и по убыванию соответственно. Если порядок не указан явно, по умолчанию используется сортировка по возрастанию.

Например:

```
SELECT  TOP 100 PERCENT авторы.автор, авторы.[год рождения], авторы.[место рождения], авторы.язык, авторы.[число произведений]
FROM    авторы INNER JOIN книги ON авторы.автор = книги.автор
        INNER JOIN издательства ON книги.издательство = издательства.издательство
WHERE   (авторы.[год рождения] LIKE '12.%.%') AND (авторы.[число произведений] BETWEEN 1  AND 500)
ORDER BY авторы.автор
```

Запрос возвратит указанные в SELECT поля из указанных в FROM таблиц с критериями для отбора результатов по дате рождения (12 число) и числу произведений (от 1 до 500), и сортировкой по полю авторы.автор.

4.2.1. Критерии отбора

Чтобы ограничить количество строк, выдаваемых в ответ на запрос, можно задать условия поиска и критерии отбора.

Операции сравнения:

= проверяется равенство двух выражений

!= проверяется неравенство двух выражений

> проверяется, что первое выражение больше второго

>= проверяется, что первое выражение больше второго или равно ему

< проверяется, что первое выражение меньше второго

<= проверяется, что первое выражение меньше второго или равно ему

Например:

```
SELECT  автор, [год рождения]
FROM    авторы
WHERE   автор <> 'Пушкин А.С.'
```

Запрос возвратит ФИО и год рождения всех авторов кроме Пушкина А.С.

Логические операции: AND, OR, NOT

Другие ключевые слова:

LIKE. *Выражение LIKE шаблон*. Применяется для поиска по соответствующему шаблону. Шаблоны являются строковыми выражениями, состоящими из символов и метасимволов. Метасимволы –

это символы, имеющие общий смысл при использовании внутри строковых выражений.

Метасимволы:

% – символ процента соответствует строке из нескольких символов (в том числе пустой строке и строке из одного символа);

_ – символ подчёркивания соответствует любому одному символу;

[ ] – метасимвол диапазона соответствует любому одному символу;

[^] – метасимвол «не в диапазоне» соответствует любому одному символу, не входящему в диапазон или набор символов. Например, [^ m-p] или [^mnop] соответствует любому из символов, кроме символов m, n, o или p.

Например:

```
SELECT    автор, [год рождения],  [место рождения]
FROM      авторы
WHERE     автор LIKE ‘П%’
```

Запрос возвратит ФИО, год рождения и место рождения авторов, фамилия которых начинается с буквы «П».

Например:

```
SELECT    автор, [год рождения],  [место рождения]
FROM      авторы
WHERE     автор LIKE ‘[А-П]%’
```

Запрос возвратит ФИО, год рождения и место рождения авторов, фамилия которых начинается с буквы от «А» до «П».

Например:

```
SELECT    автор, [год рождения],  [место рождения]
FROM      авторы
WHERE     автор NOT LIKE ‘П%’
```

Запрос возвратит ФИО, год рождения и место рождения авторов, фамилия которых начинается с любой буквы, кроме «П».

BETWEEN.

Проверяемое_выражение BETWEEN начальное_значение AND конечное_значение. Применяется всегда в сочетании с ключевым словом AND и задаёт диапазон для поиска.

Например:

```
SELECT    автор, [год рождения], [место рождения], [число произведений]
FROM      авторы
WHERE     [число произведений] BETWEEN 1 AND 50
```

Запрос возвратит ФИО, год рождения, место рождения и число произведений авторов, написавших не более 50 произведений.

*Выражение **IS NULL.***

Применяется для поиска строк, содержащих null значения в заданной колонке. Чтобы найти значения не являющиеся null следует использовать конструкцию IS NOT NULL.

Например:

```
SELECT    автор, [год рождения], [место рождения]
FROM      авторы
WHERE     [место рождения] IS NOT NULL
```

Запрос возвратит ФИО, год рождения и место рождения авторов, у которых известно место рождения.

EXISTS *подзапрос*. Используется для проверки существования строк в выводе подзапроса, указанного после него.

Например:

```
SELECT    авторы.автор
FROM      авторы
WHERE     EXISTS (SELECT книги.издательство
                  FROM книги
    WHERE книги.издательство='Питер' AND
авторы.автор=книги.автор)
```

Запрос вернёт ФИО авторов, книги которых были выпушены издательством «Питер».

4.2.2. Подзапрос

Подзапрос позволяет строить сложные иерархии запросов, многократно выполняемые в процессе построения результирующего набора или выполнения одного из операторов изменения данных (DELETE, INSERT, UPDATE).

Подзапрос позволяет решать следующие задачи:

- определять набор строк, добавляемый в таблицу на одно выполнение оператора INSERT;
- определять данные, включаемые в представление, создаваемое оператором CREATE VIEW;
- определять значения, модифицируемые оператором UPDATE;
- указывать одно или несколько значений во фразах WHERE и HAVING оператора SELECT;
- определять во фразе FROM таблицу как результат выполнения подзапроса;
- применять коррелированные подзапросы. Подзапрос называется коррелированным, если запрос, содержащийся в предикате, имеет ссылку на значение из таблицы (внешней к данному запросу), которая проверяется посредством данного предиката.

Структура подзапроса:

```
SELECT    [ALL | DISTINCT]   [TOP n [PERCENT]  ]  список_полей,
```

FROM имена_таблиц
WHERE поле [= > < IN] (SELECT [ALL | DISTINCT] [TOP n [PERCENT]]
список_полей,
FROM имена_таблиц
[WHERE критерий_отбора])

В круглых скобках обозначен подзапрос. Для установления соответствия между запросом и подзапросом могут применяться операции сравнения (< = >), либо оператор IN. В первом случае оператор подзапрос всегда должен возвращать единственное значение, которое будет проверяться в предикате. Если подзапрос вернет более одного значения, то СУБД выдаст сообщение об ошибке выполнения SQL-оператора. Если результатом подзапроса становится группа строк (это случается всегда, когда условие не гарантирует уникальности значения проверяемого предикатом внутреннего запроса), то следует использовать оператор IN, осуществляющий выбор одного значения из указываемого множества.

IN. *Проверяемое_выражение* IN *подзапрос* или
Проверяемое_выражение IN *список_значений*

Используется как условие поиска, проверяющее, не соответствует проверяемое выражение какому-либо из значений в подзапросе или в списке значений. Если соответствие найдено, то возвращается значение TRUE. NOT IN возвращает отрицание того, что получилось бы при применении IN, поэтому NOT IN возвратит значение TRUE, если проверяемое выражение не найдено в подзапросе или в списке значений.

Например:

SELECT авторы.автор, книги.название
FROM авторы, книги
WHERE книги.издательство IN (SELECT издательства.издательство
FROM издательства
WHERE DATEPART (year, [дата основания]) < 1900)

Запрос вернёт ФИО авторов и их книги, которые были выпущены издательствами, основанными до начала 20 века.

4.3. Использование агрегатных (статических) функций

Агрегатные функции выполняют вычисления над набором значений и возвращают одно значение. Агрегатные функции могут быть заданы в списке выборки и чаще всего применяются в случаях, когда оператор содержит предложение GROUP BY.

Структура запроса:

```
SELECT    статическая_функция (имя_поля) AS заголовок
FROM      имя_таблицы
```

Статические функции:

AVG – возвращает среднее арифметическое для значений выражения.

COUNT – возвращает количество элементов в выражении (количество строк).

COUNT_BIG – то же самое, что и COUNT, но результат имеет тип данных bigint, а не int.

GROUPING – возвращает специальную дополнительную колонку; применяется, только когда предложение GROUP BY содержит операцию CUBE или ROLLUP.

MAX – возвращает максимальное значение из значений выражения.

MIN – возвращает минимальное значение из значений выражения.

STDEV – возвращает статистическое стандартное отклонение (statistical standard deviation) для всех величин выражения. Эта функция предполагает, что выражения, используемые в расчете, являются образцом всей совокупности данных.

STDEVP – возвращает статистическое стандартное отклонение для всех величин выражения. Эта функция предполагает, что выражения, используемые в расчете, являются всей совокупностью данных.

SUM – возвращает сумму всех значений выражения.

VAR – возвращает статистическое отклонение (statistical variance) для всех значений из выражения. Эта функция предполагает, что выражения, используемые в расчете, являются образцом всей совокупности данных.

VARP – возвращает статистическое отклонение для всех значений из выражения. Эта функция предполагает, что выражения, используемые в расчете, являются всей совокупностью данных.

Функция COUNT применяется специальным образом: она подсчитывает все строки таблицы. Для этого нужно после COUNT поместить символ-звёздочку в скобках.

Например:

```
SELECT COUNT (*) AS количество
FROM издательства
```

Функции AVG, COUNT, MAX, MIN и SUM могут применяться с необязательными ключевыми словами ALL или DISTINCT. Для каждой из этих функций ALL означает, что функция должна применяться ко всем значениям выражения, а DISTINCT – повторяющиеся значения должны участвовать в расчете только по одному разу. По умолчанию применяется ALL.

Например:

```
SELECT    MAX (рейтинг) - MIN(рейтинг) AS Разница
FROM      издательства
```

Запрос возвратит разницу между высшим и низшим рейтингами.

4.4. Определение связей между таблицами

Существуют два способа определения связей между таблицами.

Способ 1. Применение команд *JOIN*.

Внутреннее соединение (inner join) является типом соединений, принятым по умолчанию. Внутреннее соединение задает набор результатов, в который будут включены лишь те строки таблиц, которые соответствуют условию ON, а все несоответствующие строки будут отброшены. Чтобы задать соединение, применяйте ключевое слово JOIN. Для задания условия поиска, на котором основывается соединение, применяется ключевое слово ON.

Структура запроса:

```
SELECT [ALL | DISTINCT]  список_полей
FROM  имя_таблицы1  INNER JOIN  имя_таблицы2  ON  условие_обьединения
[INNER JOIN имя_таблицы3 ON условие_обьединения]
 [INNER JOIN имя_таблицы4 ON условие_обьединения]
……....
[WHERE  критерий_отбора]
[ORDER BY  столбцы_сортировки  [ASC | DESC] ]
```

Полное внешнее соединение (full outer join) задает набор результатов, состоящий как из строк, соответствующих условию ON, так и из строк, не соответствующих условию ON. Для строк, не соответствующих условию ON, значением колонки, несоответствующей условию, станет NULL. Структура запроса схожа с представленным выше, но команда INNER JOIN заменяется на FULL OUTER JOIN.

Левое внешнее соединение (left outer join) возвращает строки, в которых произошло соответствие условию поиска, плюс все строки из таблицы, заданной слева от ключевого слова JOIN. Структура запроса схожа с представленным выше, но команда INNER JOIN заменяется на LEFT OUTER JOIN.

Правое внешнее соединение (right outer join) противоположно левому внешнему соединению: в него войдут строки, соответствующие условию поиска, плюс все строки из таблицы, заданной справа от ключевого слова JOIN. Структура запроса схожа с представленным выше, но команда INNER JOIN заменяется на RIGHT OUTER JOIN.

Перекрестное соединение (cross join) – это произведение двух таблиц, в котором не задано предложение WHERE, т.е. не задано условие объединения. Без предложения WHERE будет возвращаться такой результат: каждая строка из первой таблицы сопоставляется с каждой строкой из второй таблицы, поэтому размер набора результатов будет

равен числу строк первой таблицы, умноженному на число строк второй таблицы.

Структура запроса:

```
SELECT    список_полей
FROM      таблица1 CROSS JOIN таблица2
```

Способ 2. Применением команды *WHERE*.

Структура запроса:

```
SELECT    [ALL | DISTINCT]  список_полей
FROM      имена_таблиц
WHERE (условие_объединения1)AND(условие_объединения2)AND (усл
овие_объединения3)
```

Условия объединения должны иметь вид таблица1.поле1 = таблица2.поле2.

Структура с применением команды WHERE поддерживает только внутреннее соединение (аналогичное INNER JOIN).

4.5. Объединение результатов двух или нескольких запросов

Для объединения результатов двух или нескольких запросов в один набор результатов используется операция UNION. При этом необходимо соблюдать два правила:

1. Все запросы должны иметь одинаковое количество колонок.

Колонки, перечисленные в операторах SELECT, объединенные при помощи UNION, сопоставляются друг с другом следующим образом: первая колонка из первого оператора SELECT будет соответствовать первым колонкам из всех последующих операторов SELECT, вторая колонка будет соответствовать вторым колонками всех последующих операторов SELECT, и т.д. Поэтому во всех операторах SELECT объединенных при помощи UNION, должно быть одинаковое количество колонок, что гарантирует однозначное сопоставление.

2. Типы данных соответственных колонок из запросов должны быть совместимыми.

Это значит, что соответственные колонки должны иметь либо одинаковые типы данных, либо SQL Server сможет выполнить неявное преобразование одного типа данных в другой.

Структура запроса:

```
SELECT    [ALL]  список_полей
FROM   имя_таблицы1
[GROUP BY   группируемые_поля]
[HAVING   условие_для_результата ]
UNION SELECT [ALL]  список_полей
```

FROM *имя_таблицы2*
[GROUP BY *группируемые_поля*]
[HAVING *условие_для_результата*]
UNION SELECT [ALL] *список_полей*
FROM *имя_таблицы3*
[GROUP BY *группируемые_поля*]
[HAVING *условие_для_результата*]
……..
[ORDER BY *столбцы_сортировки* [ASC | DESC]]

Единственное ключевое слово, которое можно применять вместе с UNION - это необязательное ключевое слово ALL. Если вы примените его, то в набор результатов будут включены также все повторяющиеся строки (другими словами, в набор результатов будут включены полностью все строки). Если ключевое слово ALL не применяется, то по умолчанию из набора результатов будут исключены все дублирующиеся строки.

ORDER BY можно применять не в каждом операторе SELECT объединения, а только в самом последнем. Благодаря этому ограничению, итоговый набор результатов будет отсортирован только один раз сразу для всех результатов. С другой стороны, вы можете применять GROUP BY и HAVING в отдельных операторах, так как они влияют только на отдельные наборы результатов, а не на итоговый набор результатов.

Например:

```
SELECT Издательство, 'GOOD' As [Рейтинг издательства]
FROM Издательства
WHERE Рейтинг > 50
UNION
SELECT Издательство, 'BAD' As [Рейтинг издательства]
FROM Издательства
WHERE Рейтинг < 50
```

Запрос выведет название издательства и оценку в зависимости от рейтинга – GOOD если рейтинг больше 50, BAD если рейтинг меньше 50.

4.6. Практическая реализация разработки баз данных в SQL-ориентированных СУБД

Данный блок практических заданий выполняется в системе управления базами данных Microsoft SQL Server 2000 Developer Edition либо в Microsoft SQL Server 20XX\Среда SQL Server Management Studio. В лабораторной работе №8 изучаются возможности создания баз данных

в SQL Server Management Studio. Лабораторная работа №9 посвящена созданию запросов на выборку на языке SQL. В отличие от предыдущих работ, здесь не раскрывается алгоритм выполнения задания, предполагается использовать элементы учебно-исследовательской деятельности, проблемный подход. В этом блоке заданий каждый вариант выполняется двумя студентами, с целью научить их работать в команде.

Варианты индивидуальных заданий представлены в Приложении: для лабораторной работы №8 - Приложение 2, для лабораторной работы №9 - Приложение 3.

4.6.1. Лабораторная работа №8
Создание баз данных в SQL Server Management Studio

Цель работы

Получение навыков работы по созданию таблиц в SQL Server Management Studio

Темы для предварительного изучения

- Утилита SQL Server Management Studio. Создание новой базы данных. Отсоединение и присоединение базы данных. Создание таблиц

Задание

Создать базы данных в SQL Server Management Studio в соответствии с вариантами индивидуальных заданий, представленными в Приложении (*Приложение 2*).

Выполнение задания

Использование утилиты SQL Server Management Studio является самым простым способом создания базы данных. В качестве примера создадим БД Продажи (Sales), которую позже заполним таблицами.

1. В окне **Обозреватель объектов** найти папку «Базы данных». Щелкнуть на ней правой кнопкой мыши и выбрать команду **«Создать базу данных»**.

2. В открывшемся диалоговом окне **Создание базы данных** (рис.8.1) на странице **Общие** ввести следующую информацию: Имя базы данных: Sales, Владелец: sa

- В таблице «Файлы базы данных» изменить путь к файлам данных и журнала на ваш каталог.
- Для всех остальных параметров оставьте значения по умолчанию.

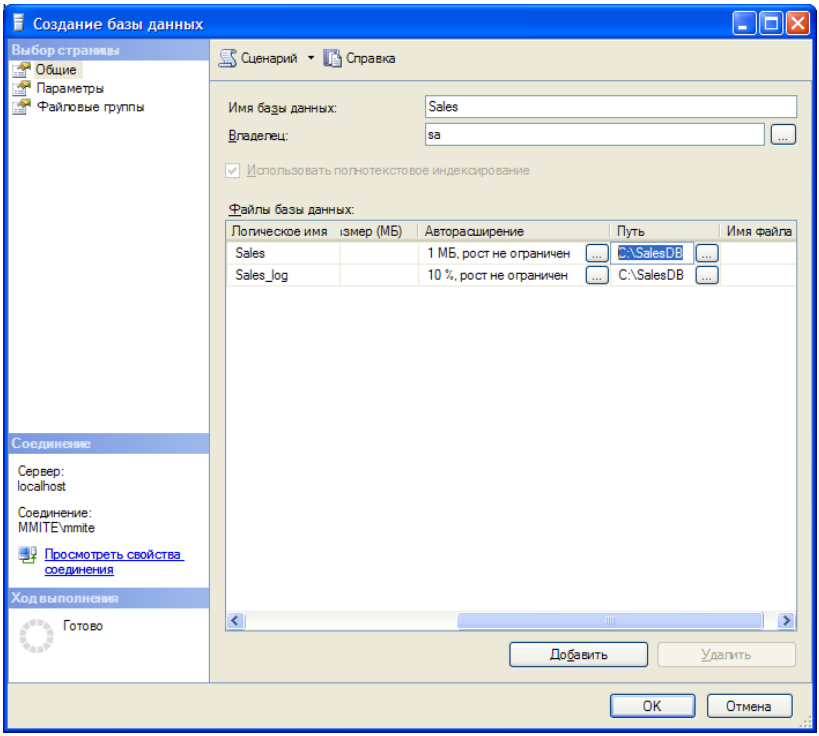


Рис.8.1 Пример создания базы данных

3. Для создания базы данных щелкнуть **«OK»**. Вы должны увидеть свою новую базу данных в окне «Обозреватель объектов».

Отсоединение и присоединение базы данных

Для переноса базы данных на другой сервер необходимо отсоединить ее от текущего сервера. Для этого в контекстном меню базы данных Sales выбрать команду ***Задачи → Отсоединить.*** В диалоговом окне **Отсоединение базы данных** нажать кнопку «OK». Убедитесь, что Sales исчезла из списка баз данных в дереве обозревателя объектов. Теперь файлы базы данных могут быть перенесены на другой сервер.

Для присоединения базы данных к серверу выбрать в контекстном меню узла «Базы данных» команду ***Присоединить***. В диалоговом окне **Присоединение базы данных** с помощью кнопки **«Добавить»** выбрать созданный на предыдущих этапах файл Sales.mdf (ldf файл будет определен системой автоматически), изменить владельца на sa и нажмите

кнопку **«OK»**. База данных Sales должна появиться в списке дерева обозревателя объектов.

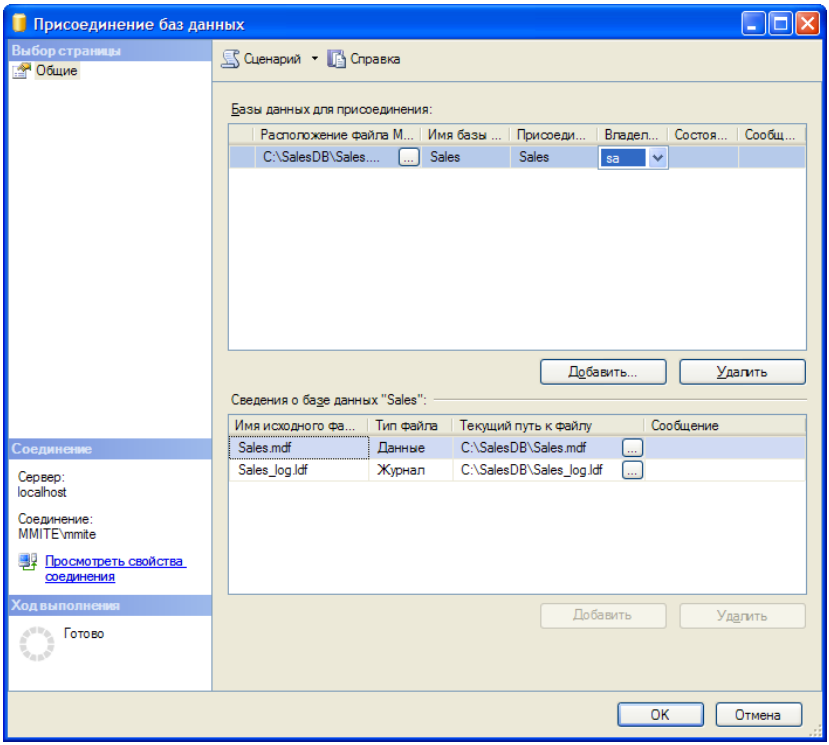


Рис.8.2 Присоединение базы данных

Создание таблиц

Таблицы представляют собой объекты базы данных, используемые непосредственно для хранения всех данных. Одним из самых главных правил организации баз данных является то, что в одной таблице должны храниться данные лишь об одном конкретном типе сущности (например, клиенты, товары, заказы и т. п.).

Данные в таблицах организованы по полям и записям. Поля (или столбцы таблицы) содержат определенный тип информации, например, фамилию, адрес, телефонный номер. Запись (или строка таблицы) - группа связанных полей, содержащих информацию об отдельном экземпляре сущности.

Любое поле таблицы характеризуется как минимум двумя обязательными свойствами: *Имя столбца* (реализует способ обращения к конкретному полю в таблице; рекомендуется всегда присваивать полям смысловые имена) и *Тип данных* (определяет, информация какого типа может храниться в данном поле).

При выборе типа данных для столбца следует отдавать предпочтение типу, который позволит хранить любые возможные для этого столбца значения и занимать при этом минимальное место на диске. Стандартные типы данных, используемые в MS SQL Server, представлены в Приложении (*Приложение 4*).

SQL Server позволяет на основе системных типов данных создавать пользовательские типы со всеми предварительно заданными параметрами. В качестве примера создадим тип данных phone, который будет использоваться в таблице Customer (информация о клиентах) для хранения телефонного номера клиента. Для его создания воспользуемся графическим интерфейсом утилиты Management Studio (рисунок 8.3), для этого необходимо выполнить следующие действия:

- В дереве обозревателя объектов раскрыть ***папки Базы данных*** → ***Sales*** → ***Программирование*** →***- Типы.*** В контекстном меню узла **Определяемые пользователем типы данных** выбрать команду ***Создать определяемый пользователем тип данных***.
- В появившемся окне в текстовом поле **Имя** ввести phone. В раскрывающемся списке **Тип данных** выбрать nchar. В качестве длины введите 10. Отметить параметр **Разрешить значения nul**l, чтобы иметь возможность не указывать телефонный номер при добавлении нового клиента.
- В секции **Привязки** оставить пустыми значения и щелкнуть на кнопке **«ОК»**. Созданный пользовательский тип данных должен появиться в дереве обозревателя объектов.

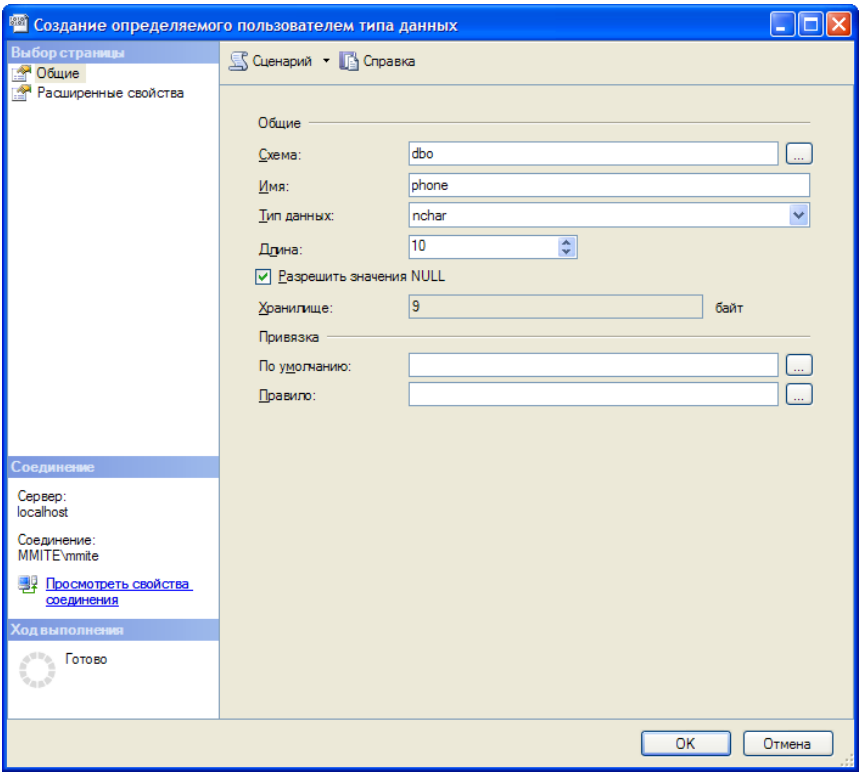


Рис.8.3 Пример создания пользовательского типа данных

Заполним базу данных Sales, для этого создадим в ней пять таблиц:
– первая таблица Customer (информация о клиентах),
– вторая таблица City (справочник городов),
– третья таблица Product (информация о товарах),
– четвертая таблица Order (подробная информация о заказах),
– пятая таблица OrdItem (перечень товаров, входящих в заказ).

Таблицы в БД можно создавать как в графическом интерфейсе (в утилите Management Studio), так и с помощью кода T-SQL. Воспользуемся графическим способом.

В качестве примера создадим таблицу Customer. Вначале определим структуру создаваемой таблицы. В табл. 8.1 представлены все поля таблицы Customer и их основные свойства.

Таблица 8.1

Характеристика полей таблицы Customer

Имя столбца	Тип данных	Разрешить null	Описание
IdCust	int, identity	нет	Уникальный идентификационный номер клиента, на который можно ссылаться в других таблицах
FName	nvarchar(20)	нет	Имя клиента
LName	nvarchar(20)	нет	Фамилия клиента
IdCity	int	нет	Ссылка на номер города
Address	nvarchar(50)	нет	Адрес клиента
Zip	nchar(5)	нет	Почтовый индекс клиента
Phone	phone	да	Телефонный номер клиента

Для создания таблицы Customer (рисунок 8.4) необходимо выполнить следующие действия:

- В дереве обозревателя объектов в базе данных Sales в контекстном меню узла **Таблицы** выбрать команду ***Создать таблицу***.
  В рабочей области должна появиться вкладка с конструктором таблиц.
- В первую строку в столбце *Имя столбца* ввести IdCust, в столбце *Тип данных* выбрать int.
  Убедитесь что параметр *Разрешить значения null* отключен.
- В нижней половине экрана в разделе **Свойства столбцов** ввести описание поля и измените значение параметра *Спецификация идентификатора / (Идентификатор)* на **Да** для того чтобы значения номера клиента формировались автоматически. Свойство *Идентифицирующий столбец (Identity*), обычно используемое совместно с типом данных int, предназначено для автоматического приращения значения на единицу при добавлении каждой новой записи. К примеру, клиент, добавленный в таблицу первым, будет иметь значение идентификатора 1, вторым – 2, третьим – 3, и т.д.
- Аналогичным образом ввести описания всех остальных полей и закрыть окно конструктора таблиц. Ввести в качестве имени таблицы Customer.

Вновь созданная таблица должна появиться в дереве обозревателя объектов в папке **Таблицы.**

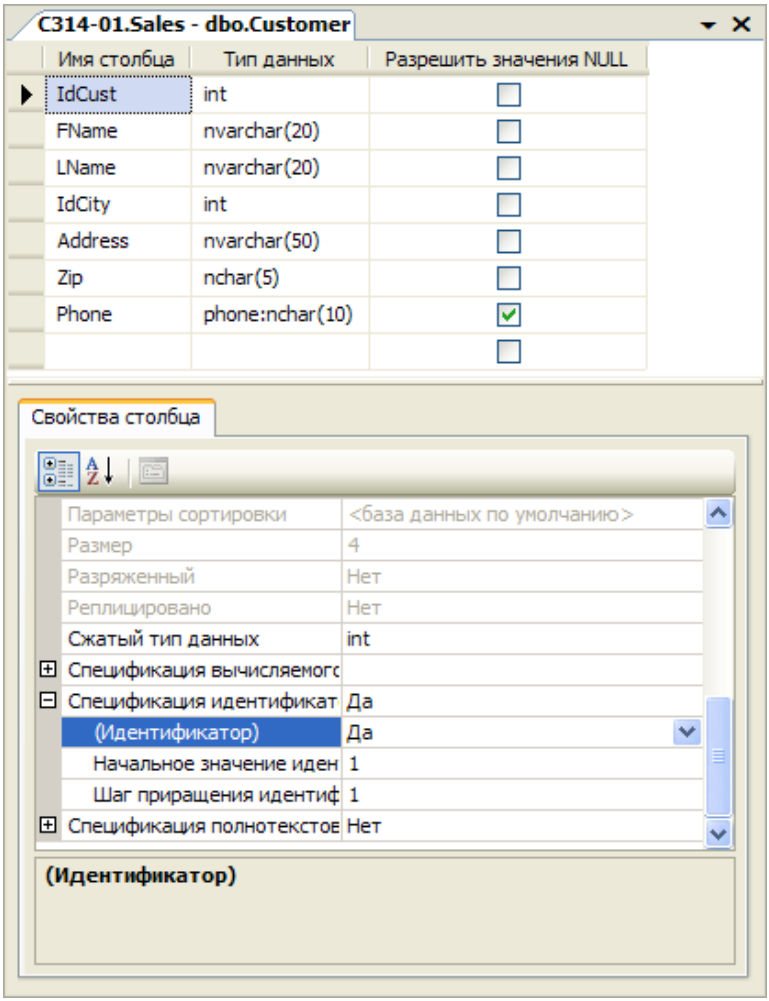


Рис.8.4 Пример создания таблицы

Аналогичным образом создаются все остальные таблицы базы данных Sales.

4.6.2. Лабораторная работа №9
Запросы на выборку на языке SQL

Цель работы

Изучение основ языка структурированных запросов SQL. Получение навыков работы по созданию запросов на выборку на языке SQL.

Темы для предварительного изучения

- Ключевые слова SQL. Структура запроса на языке SQL. Основные команды SQL, применяемые для построения запроса. Использование агрегатных (статических) функций. Объединение двух или нескольких запросов

Задание

1. Создать запросы на выборку из нескольких таблиц на языке SQL заданными критериями отбора.

2. Создать запрос на выборку на языке SQL, содержащий статические (агрегатные функции);

Варианты индивидуальных заданий представлены в Приложении (*Приложение 3*).

Выполнение задания

Структура запроса на выборку:

```
SELECT    [ALL | DISTINCT]   [TOP n [PERCENT]  ]  список_полей,
FROM      имена_таблиц
[WHERE    критерий_отбора]
[GROUP BY   группируемые_поля]
[HAVING   условие_для_результата ]
[ORDER BY   столбцы_сортировки   [ASC | DESC]  ]
```

Данный запрос возвращает таблицу, содержащую выбранные столбцы (поля) из указанных строк (записей) таблицы-источника. В квадратных скобках указаны необязательные параметры.

Назначение всех элементов этой структуры запроса на выборку: см. п.4.2.

Агрегатные (статические) функции выполняют вычисления над набором значений и возвращают одно значение (см. п.4.3).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Агальцов, В.П. Базы данных. В 2-х т. Т. 1. Локальные базы данных: Учебник / В.П. Агальцов. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. – 352 с.
2. Агальцов, В.П. Базы данных. В 2-х т. Т. 2. Распределенные и удаленные базы данных: Учебник / В.П. Агальцов. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. – 272 с.
3. Базы данных: учебник для вузов / Под ред. А. Д. Хомоненко. – 4-е изд., доп. и перераб. - СПб.: КОРОНА принт, 2004. – 736 с. – Библиогр. в конце глав. – Рек. УМО. – ISBN 5-7931-0284-1.
4. Бьюли А. Изучаем SQL. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 312 с.
5. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. Пер. с англ.: – М.: Изд. дом «Вильямс», 2004. – 1088 с.
6. Голицына, О.Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: Форум, 2004. – 352 с.
7. Голицына, О.Л. Базы данных: учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: Форум, 2012. – 400 с.
8. Дейт К. Введение в системы баз данных: проектирование. Реализация и управление. Пер. с англ. – СПб.: БХВ-Петербург, 2004. – 324 с.
9. Дж. Грофф, П. ВайнбергSQL: Полное руководство: Пер. с англ. – 3-е изд., перераб. и доп. – К.: Издательская группа BHV, 2015. – 960 с.
10. Злыднева Т.П. Введение в базы данных: учеб. пособие. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2017. – 71с.
11. Злыднева Т.П. Введение в операционные системы. Проблемно-информационный курс: учеб. пособие / Т.П. Злыднева. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2015. – 99с.
12. Злыднева Т.П. Возможные пути успешной реализации ФГОС ВПО третьего поколения // Педагогические аспекты математического образования : сб. науч. тр. – Магнитогорск, 2012. – Вып. 9. – С. 33–36.
13. Злыднева Т.П. История прикладной математики и информатики. Часть 1. История математики: учеб. пособие / Т.П. Злыднева. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2014. – 89 с.
14. Злыднева Т.П. История прикладной математики и информатики. Часть 2. История информатики: учеб. пособие / Т.П. Злыднева. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2014. – 71 с.
15. Злыднева Т.П. К вопросу об организации учебно-исследовательской деятельности магистрантов // Развитие науки и образования в современном

мире: сб. науч. трудов Международной научно-практической конференции, часть 4. – М.: «АР-Консалт», 2015. – С. 85–88.
16. Злыднева Т.П. Математика и информатика: от истоков до современности: учебное пособие. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2018. – 150 с.
17. Злыднева Т.П. Методика организации исследовательской деятельности студентов в процессе изучения дисциплин информатики // Университетское образование : материалы междунар. Научно-методической конференции – Пенза, 2007. – С.241–243.
18. Злыднева Т.П. Методика формирования компетенций при реализации ФГОС ВПО третьего поколения // Наука и образование в современном обществе: вектор развития: сб. науч. трудов Международной научно-практической конференции, часть 5. – М.: «АР-Консалт», 2014. – С. 91–94.
19. Злыднева Т.П. Моделирование системы организации исследовательской деятельности студентов вузов в процессе профессиональной подготовки // Вестник Челябинского государственного педагогического университета. – 2006. – № 5. – С. 22–30.
20. Злыднева Т.П. Основы работы с базами данных: практикум. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2018. – 66 с.
21. Злыднева Т.П. Проблема качественной подготовки специалистов и возможные пути её решения // Педагогические аспекты математического образования : сб. науч. тр. – Магнитогорск, 2008. – Вып. 5. – С. 58–62.
22. Злыднева Т.П. Проблемный подход в изучении дисциплин информатики // Актуальные проблемы информатики и информационных технологий : сб. науч. трудов XIII Междунар. научно-практической конференции. – Тамбов, 2009. – С. 37–41.
23. Злыднева Т.П. Программирование на языке С: Лабораторный практикум для студентов специальности «Прикладная математика и информатика» / Т.П. Злыднева. – Магнитогорск: МаГУ, 2008. – 72 с.
24. Злыднева Т.П. Обучение студентов исследовательской деятельности в рамках дисциплин информатики // Фундаментальные науки и образование : материалы Всероссийской научно-практической конференции. – Бийск, 2006. – С. 289–293.
25. Злыднева Т.П. Операционные системы. Проблемно-информационный курс : методические рекомендации по изучению курса «Системное и прикладное программное обеспечение» / Т.П. Злыднева. – Магнитогорск: МаГУ, 2007. – 48 с.
26. Злыднева Т.П. Опытно-экспериментальная работа по организации исследовательской деятельности студентов в процессе изучения дисциплин информатики // Психология и педагогика: пути и методы развития : сб. статей III Междунар. научно-практической конференции – Пенза, 2011. – С. 53–57.

27. Организация исследовательской деятельности в процессе обучения естественнонаучным дисциплинам в школе и вузе: монография / П.Ю. Романов, Т.П. Злыднева, Т.Е. Романова и др. – М. : ИНФРА-М, 2017. – 260 с.
28. Злыднева Т.П. Организация исследовательской деятельности студентов университета в процессе профессиональной подготовки: дис… канд. пед. наук: 13.00.08 – Магнитогорск, 2006. – 204 с.
29. Злыднева Т.П. Организация исследовательской деятельности студентов университета в процессе профессиональной подготовки: автореф. дис… канд. пед. наук – Магнитогорск: Изд-во Магнитогорского государственного университета, 2006. – 24 с.
30. Злыднева Т.П. Роль информационных технологий в формировании профессиональных компетенций. // Информационные технологии в науке, управлении, социальной сфере и медицине: сборник научных трудов III Международной научной конференции: в 2 частях. Под редакцией: О.Г. Берестневой, О.М. Гергет, Т.А. Гладковой; Национальный исследовательский Томский политехнический университет. – 2016. – С. 348–350.
31. Злыднева Т.П. Роль учебно-исследовательской деятельности студентов в реализации ФГОС нового поколения // Физико-математические науки и образование: сборник трудов участников Всероссийской научно-практической конференции. – Магнитогорск, 2012. – С. 22–24.
32. Злыднева Т.П. Учебно-исследовательская деятельность студентов как необходимая составляющая процесса формирования профессиональных компетенций // Современные проблемы науки и образования : материалы L внутривузовской научной конференции преподавателей МаГУ. – Магнитогорск, 2012. – С. 233–234.
33. Злыднева Т.П. Язык С: лабораторный практикум по программированию для студентов специальности «Прикладная математика и информатика» / Т.П. Злыднева. – Магнитогорск: МаГУ, 2005. – 74 с.
34. Информационные системы: учеб. пособие для вузов / Ю. Избачков [и др.]. – 3-е изд. – СПб.: Питер, 2011. – 539 с.: ил. – Доп. Мин. обр. РФ. – ISBN 978-5-49807-158-9.
35. Ипатова, Э. Р. Введение в технологии баз данных: учебное пособие для вузов / Э. Р. Ипатова ; МаГУ. – Магнитогорск : Изд-во МаГУ, 2007. – 301 с. – Библиогр.: с. 284-291.
36. К. Дж. Дейт, SQL и реляционная теория. Как грамотно писать код на SQL. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 480 с., ил.
37. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. – СПб.: Питер, 2013. – 240 с.

38. Кириллов В., Громов Г. Введение в реляционные базы данных [Электронный ресурс]. – СПб.: БХВ-Петербург, 2010. – 464 с. – Режим доступа: http://ibooks.ru/reading.php?productid=18462 – Загл. с экрана. - ISBN 978-5-94157-770-5.
39. Кошелев, В.Е. Базы данных в ACCESS 2007: Эффективное использование / В.Е. Кошелев. – М.: Бином-Пресс, 2009. – 592 с.
40. Кузин, А. В. Базы данных: учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 3-е изд., стер. – М.: Академия, 2008. – 315 с. – Библиогр.: с. 313. – Доп. УМО. – ISBN 978-5-7695-5775-0.
41. Кузин, А.В. Базы данных: Учебное пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. – М.: ИЦ Академия, 2012. – 320 с.
42. Кузнецов С. Д. Основы баз данных. – 2-е изд. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. – 484 с.
43. Малыхина М. П. Базы данных: основы, проектирование, использование, 2-е изд. перераб. и доп. – СПб.: БХВ-Петербург, 2007. – 528 с.
44. Мартин Грабер. Введение в SQL, БХВ-Петербург, 2010. – 228 с.
45. Назарова О. Б. Разработка реляционных баз данных с использованием СУБД MS ACCESS: Лаб. практикум по дисциплине «Информационные системы» / О. Б. Назарова, О. Е. Киселева; МаГУ. – Магнитогорск: Изд-во МаГУ, 2003. – 58 с. – Библиогр.: с. 51.
46. Пирогов В. Информационные системы и базы данных: организация и проектирование [Электронный ресурс]. – СПб. : БХВ-Петербург, 2010. – 528 с. – Режим доступа: http://ibooks.ru/reading.php?productid=18485 – Загл. с экрана. – Гриф УМО. – ISBN 978-5-9775-0399-0.
47. Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. – СПб.: БХВ-Петербург, 2009. – 528 с.
48. Ржеуцкая С. Ю. Базы данных. Язык SQL: учеб. пособие / С. Ю. Ржеуцкая. – Вологда: ВоГТУ, 2010. –159 с.
49. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М.: Юрайт, 2013. – 463 с.
50. Фиайли К. SQL. Руководство по изучению языка: Пер. с англ. – М.: ДМК Пресс, 2013. – 456 с.
51. Харрингтон Д. Проектирование объектноориентированных баз данных [Электронный ресурс]: пер. с англ. – М.: ДМК Пресс, 2001. – 272 с.: ил. (Серия «Для программистов»). – Режим доступа: http://e.lanbook.com/, электронная библиотечная система «Лань». – ISBN 5940740979.

Варианты индивидуальных заданий
(лабораторные работы №1-№7)

Вариант 1

Предметная область: Университет
Минимальный список характеристик:
- Номер, ФИО, адрес и должность преподавателя;
- Код, название предмета, количество часов, тип контроля;
- Код, название, номер кафедры, ФИО заведующего кафедрой;
- Номер аудитории, где преподаватель читает свой предмет.

Вариант 2

Предметная область: Деканат
Минимальный список характеристик:
- Направление подготовки, код группы, ФИО студента, дата рождения, домашний адрес, телефон, примечания – например, автобиография студента;
- Код, название, количество часов и вид контроля предметов, код сессии и оценки каждого студента по каждому предмету в каждую сессию.

Вариант 3

Предметная область: Библиотека
Минимальный список характеристик:
- Автор книги, название, год издания, цена, является ли новым изданием, краткая аннотация;
- Номер читательского билета, ФИО, адрес и телефон читателя, дата выдачи книги читателю и дата сдачи книги читателем.

Вариант 4

Предметная область: Поликлиника
Минимальный список характеристик:
- Номер, фамилия, имя, отчество, дата рождения пациента;
- ФИО, должность и специализация лечащего врача;
- Диагноз, поставленный данным врачом данному пациенту, необходимо ли амбулаторное лечение, срок потери трудоспособности, состоит ли на диспансерном учете, примечание.

Вариант 5

Предметная область: Производство
Минимальный список характеристик:

- Код изделия, название изделия, является ли типовым, примечание - для каких целей предназначено;
- Код, название, адрес и телефон предприятий, выпускающих изделия;
- Год выпуска и объем выпуска данного изделия предприятием.

Вариант 6

Предметная область: Личные данные о студентах.
Минимальный список характеристик:

- Фамилия и инициалы студента, курс, факультет, направление подготовки, дата рождения студента, домашний адрес, телефон,
- Семейное положение, сведения о семье.

Вариант 7

Предметная область: Аэропорт
Минимальный список характеристик:

- Номер рейса, пункт назначения, дата рейса, тип самолета, время вылета, время в пути, является ли маршрут международным;
- Сведения о пассажире, примечание.

Вариант 8

Предметная область: Автотранспортное предприятие
Минимальный список характеристик:

- Номерной знак автомобиля, марка автомобиля, его техническое состояние, местонахождение автомобиля, средняя скорость , грузоподъемность, расход топлива;
- Табельный номер водителя, ФИО, дата рождения, стаж работы, оклад;
- Дата выезда, дата прибытия, место назначения, расстояние, расход горючего, масса.

Вариант 9

Предметная область: Сеть магазинов
Минимальный список характеристик:

- Номер, ФИО, адрес, телефон и капитал владельцев магазинов;
- Номер, название, адрес и телефон магазина;
- Номер, ФИО, адрес, телефон поставщика, а также стоимость поставки данного поставщика в данный магазин.

Вариант 10

Предметная область: Авторемонтные мастерские
Минимальный список характеристик:

- Номер водительских прав, ФИО, адрес и телефон владельца автомобиля;
- Номер, ФИО, адрес, телефон и квалификация механика;
- Номер, марка, мощность и цвет автомобиля;
- Номер, название, адрес и телефон ремонтной мастерской.

Вариант 11

Предметная область: Домоуправление
Минимальный список характеристик:

- Номер подъезда, номер квартиры, общая площадь, полезная площадь, количество комнат;
- Фамилия квартиросъемщика, дата прописки, количество членов семьи, количество детей в семье, есть ли задолженность по квартплате, примечание.

Вариант 12

Предметная область: Оптовая база
Минимальный список характеристик:

- Код товара, название товара, количество на складе, стоимость единицы товара, примечания - описание товара;
- Номер и ФИО поставщика товара, срок поставки и количество товаров в поставке.

Вариант 13

Предметная область: Персональные ЭВМ
Минимальный список характеристик:

- Фирма-изготовитель, тип процессора, тактовая частота, объем ОЗУ, объем жесткого диска, дата выпуска;
- Сведения о фирмах-реализаторах: наименование, адрес, телефон, примечание.

Вариант 14

Предметная область: Заказы
Минимальный список характеристик:

- Фамилия, имя, отчество клиента, номер счета, адрес, телефон;
- Номер заказа, дата исполнения, стоимость заказа, название товара, его цена и количество.

Варианты индивидуальных заданий (лабораторная работа №8)

Вариант 1

Создать БД, содержащую информацию о студентах ВУЗа: ФИО студента, год рождения, институт, кафедра, группа, номер зачетки, число специальностей (направлений подготовки) в институте, год основания института, число студентов в институте, директор института, название специальности (направления подготовки), код специальности, год основания специальности, число групп данной специальности, заведующий кафедрой.

Вариант 2

Создать БД, содержащую информацию о книгах в библиотеке: название, автор, издательство, год издания, жанр, количество страниц, язык, на котором пишет автор, число произведений, созданных автором, тираж книги, адрес и дата основания издательства, рейтинг издательства (100 балльный).

Вариант 3

Создать БД, содержащую информацию о компьютерах: наименование, фирма, страна, оборот фирмы, служба поддержки и рейтинг фирмы, стоимость компьютера, модель процессора, объем ОЗУ, тип НЖМД, покупатель компьютера, место жительства и телефон покупателя.

Вариант 4

Создать БД, содержащую информацию о записях в фонотеке: композиция, исполнитель, альбом, автор текста, продолжительность, страна исполнителя, жанр, количество альбомов, год выпуска альбома, тираж альбома, выпускающая компания, продолжительность и число композиций в альбоме.

Вариант 5

Создать БД, содержащую информацию о фильмах в фильмотеке: название фильма, жанр, кинокомпания, режиссер, число фильмов режиссера, длительность, год основания кинокомпании, страна и численность работающих, рейтинг фильм (10 балльный).

Вариант 6

Создать БД, содержащую информацию о напитках в супермаркете: название, тип, тара, цена, код страны, крепость, срок хранения, температура хранения, название страны, валюта, курс по отношению к рублю, стоимость перевозки одной тонны.

Вариант 7

Создать БД, содержащую информацию о продуктах в магазинах: наименование, цена за 1 кг, дата изготовления, производитель, магазин, адрес, телефон и директор магазина, поставщик, адрес и телефон поставщика, количество обслуживаемых магазинов, репутация.

Вариант 8

Создать БД, содержащую информацию о косметических изделиях: наименование, цена, качество, магазин, фирма, адрес и телефон магазина, номер лицензии, рейтинг, страна и директор фирмы, телефон службы поддержки, рейтинг фирмы.

Вариант 9

Создать БД, содержащую информацию об автомобилях на авторынке: модель, цвет, цена, фирма-изготовитель, год выпуска, двигатель, тип кузова, марка бензина, максимальная скорость, страна, год основания фирмы, телефон службы поддержки, рейтинг.

Вариант 10

Создать БД, содержащую информацию о лекарственных средствах: название препарата, группа, форма выпуска, состав, способ введения, заболевание, дозировка, длительность приема, средняя длительность течения заболевания, ФИО больного, возраст, вес, рост, дата заболевания, сопутствующие заболевания.

Варианты индивидуальных заданий
(лабораторная работа №9)

Вариант 1

- Получить выборку всех студентов института естествознания и стандартизации, специальности прикладной математики и информатики с годом рождения после 1997.
- Получить выборку специальностей, основанных в период с 1980 и по 2000 г. Указать факультет и кафедру, которым данная специальность принадлежит.
- Получить список студентов, сгруппированный по институтам. У института указать директора, а у кафедры – заведующего.
- Получить выборку специальностей с указанием директора, института и кафедры.

Вариант 2

- Получить выборку книг с тиражом, превышающим 2000 экземпляров и рейтингом издательства книги выше 50.
- Получить список авторов, публиковавшихся с 1990 по 2005 гг. включительно, но издававшихся не в издательстве «Мир».
- Получить список книг, сгруппированных по издательствам и с тиражом, не ниже среднего.
- Получить выборку издательств с указанием адреса, рейтинга и авторов, которые в них публиковались.

Вариант 3

- Сделать выборку компьютеров, проданных в фирме «Red Comp», заказанных в МГТУ и ценной не ниже 50000. Значение типа процессора не должно быть нулевым.
- Сделать выборку компьютеров в ценовом диапазоне от 20000 до 50000 рублей с указанием фирмы-поставщика, ее телефона службы поддержки, покупателя, и его телефона.
- Получить список фирм, сгруппированных по странам и с годовым оборотом выше среднего.
- Получить выборку данных с указанием заказчика, модели заказанного им компьютера, и его характеристик.

Вариант 4

- Получить список альбомов, записанных в США и выпущенных тиражом выше 1000 экземпляров.
- Получить список композиций, с указанием исполнителя, страны, альбома, в который данная композиция входит, года выпуска и

компании издателя. Продолжительность альбома, на котором присутствует композиция, не должна превышать 50 минут.

- Получить список композиций, сгруппированных по исполнителям и тираж альбомов, в которых они входят, должен быть больше (но не равен) минимального тиража.
- Получить выборку исполнителей США с указанием композиции, альбома и автора текста.

Вариант 5

- Получить выборку фильмов с рейтингом выше 5 и снимавшихся не в Голливуде.
- Получить выборку режиссеров с количеством отснятых фильмов меньше 15 и снятых в жанре «Боевик».
- Получить список фильмов, сгруппированных по жанрам и с рейтингом выше среднего.
- Получить выборку фильмов в жанре «Боевик» с указанием режиссера, количеством снятых им фильмов, кинокомпании и датой ее основания, в которой была снята картина.

Вариант 6

- Получите выборку всех напитков, произведенных во Франции срок годности которых неограничен, а температура хранения которых не превышает 10°C. Тип напитка не должен быть неопределён.
- Получите выборку стран, стоимость перевозки напитков из которых лежит в пределах от 5000 до 8000 включительно, и укажите напитки, производимые в данной стране. Тип напитка не должен быть неопределён.
- Получите выборку напитков сгруппированных по тарам, и выведите на экран только те напитки с максимальной крепостью.
- Получить выборку названий напитков, их типов, крепости, страны изготовителя и стоимостью перевозки одной тонны напитка из страны.

Вариант 7

- Получите выборку продукции, поставляемой фирмой АО «Продторг» с датой изготовления не позднее 10.10. 2017 г. Цена за кг не должна быть неопределенна.
- Создайте запрос, результатом которого будет наименование продукции, производимой Мясокомбинатом и наименование которой должно начинаться на «К». Цена за кг не должна быть неопределенна.
- Выведите на экран наименования товаров, сгруппированных по дате изготовления, поставляемых АО «Продторг» и цена за кг на которые ниже среднего значения.

- Получить выборку наименований товаров, их производителем, магазином, директором магазина, поставщиком и его репутацией.

Вариант 8

- Получите выборку продукции с ценой не ниже 85 руб. и укажите магазин, где можно приобрести данную продукцию, и телефон службы поддержки фирмы-изготовителя. Качество продукции не должно быть неопределенно.
- Создайте запрос, результатом которого будет список продукции фирмы Avon, ее телефон службы поддержки, и телефон магазина, в котором данную продукцию можно приобрести. Рейтинг магазина должен лежать в пределах от 20 до 80. Качество продукции не должно быть неопределенно.
- Выведите список товаров, сгруппированных по качеству изготовления и цена которых не превышает среднего значения цен.
- Получить выборку продукции с указанием ее качества, фирмы изготовителя и рейтинга фирмы.

Вариант 9

- Получить выборку автомобилей, произведенных в Германии не позднее 2010 года. Цвет модели не должен быть неопределён.
- Создайте запрос, результатом которого будет список автомобилей, произведенных с 2000 и по 2015 год включительно. Указать также фирму-изготовитель, страну, и телефон службы поддержки. Цвет модели не должен быть неопределён.
- Вывести список моделей автомобилей, сгруппированный по типу кузовов и с рейтингом фирмы-изготовителя не ниже среднего значения рейтинга.
- Получить выборку моделей автомобилей с указанием их года выпуска, мощности двигателя, типа бензина, фирмы изготовителя, года основания фирмы и страны.

Вариант 10

- Получить выборку ФИО больных с сопутствующим заболеванием «простуда» и названием препарата, который прописаны больному. Состав препарата не должен быть неопределён.
- Создать запрос, результатом которого будет список препаратов, принадлежащих группе 1 и название которых начинается на «М». Состав препарата не должен быть неопределён.
- Выведите на экран список препаратов, сгруппированных по группам и длительность приема которых не превышает среднего значения.
- Получить выборку сопутствующих заболеваний, которым соответствует препарат, принимаемый больным, заболевание и средняя его длительность.

Типы данных в MS SQL Server

Целочисленные данные

- **bit** (1 байт). Может хранить только значения 0, 1 или null (пустое значение, сообщающее об отсутствии данных). Его удобно использовать в качестве индикатора состояния – включено/выключено, да/нет, истина/ложь.
- **tinyint** (1 байт).Целые значения от 0 до 255.
- smallint (2 байта). Диапазон значений от -2^{15} (-32768) до 2^{15} (3767).
- **int** (4 байта). Может содержать целочисленные данные от -2^{31} (-2147483648) до 2^{31} (21474833647).
- **bigint** (8 байт). Включает в себя данные от -2^{63} (9223372036854775808) до 2^{63} (9223372036854775807). Удобен для хранения очень больших чисел, не помещающихся в типе данных int.

Текстовые данные

- **char**. Содержит символьные не Unicode-данные фиксированной длины до 8000 знаков.
- **varchar**. Содержит символьные не Unicode-данные переменной длины до 8000 знаков.
- **nchar**. Содержит данные Unicode фиксированной длины до 4000 символов. Подобно всем типам данных Unicode его удобно использовать для хранения небольших фрагментов текста, которые будут считываться разноязычными клиентами.
- **nvarchar.** Содержит данные Unicode переменной длины до 4000 символов.

Десятичные данные

- **decimal**. Содержит числа с фиксированной точностью от -10^{38}-1 до 10^{38}-1. Он использует два параметра: точность и степень. Точностью называется общее количество знаков, хранящееся в поле, а степень – это количество знаков справа от десятичной запятой.
- **numeric**. Это синоним типа данных decimal – они идентичны.

Денежные типы данных

- **money** (8 байт). Содержит денежные значения от -2^{63} до 2^{63} с десятичной точностью от денежной единицы. Удобен для хранения денежных сумм, превышающих 214768,3647.
- **smallmoney** (4 байта). Содержит значения от -214748,3648 до 214748,3647 с десятичной точностью.

Данные с плавающей точкой

- **float**. Содержит числа с плавающей запятой от -1,79E+38 до 1,79E+38.
- **real**. Содержит числа с плавающей запятой от -3,40E+38 до 3,40E+38.

Типы данных даты и времени

- **datetime** (8 байт). Содержит дату и время в диапазоне от 1 января 1753 года до 31 декабря 9999 года с точностью 3,33 мс.
- **smalldatetime** (4 байта). Содержит дату и время, начиная от 1 января 1900 года и заканчивая 6 июнем 2079, с точностью до 1 минуты.

Двоичные типы данных

- **binary**. Содержит двоичные данные фиксированной длины до 8000 байт.
- **varbinary**. Содержит двоичные данные переменной длины до 8000 байт.

Специализированные типы данных

- **sql_variant**. Используется для хранения значения с различными типами данных.
- **timestamp**. Используется для установки временных меток записей при вставке, которые соответствующим образом обновляются. Удобен для отслеживания изменений в данных.
- **uniqueidentifier**. Глобальный уникальный идентификатор.
- **xml.** Используется для хранения целых документов или фрагментов XML.

Учебное текстовое электронное издание

Злыднева Татьяна Павловна

БАЗЫ ДАННЫХ:
основы теории и проектирования

Учебное пособие

Издание 2-е

1,59 Мб
1 электрон. опт. диск

г. Магнитогорск, 2021 год
ФГБОУ ВО «МГТУ им. Г.И. Носова»
Адрес: 455000, Россия, Челябинская область, г. Магнитогорск,
пр. Ленина 38

ФГБОУ ВО «Магнитогорский государственный
технический университет им. Г.И. Носова»
Кафедра прикладной математики и информатики
Центр электронных образовательных ресурсов и
дистанционных образовательных технологий
e-mail: ceor_dot@mail.ru