

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ РАСТИТЕЛЬНЫХ
ПОЛИМЕРОВ»**

Г.А. Петров, С.В. Тихов, В.П. Яковлев

БАЗЫ ДАННЫХ

Учебное пособие

**Санкт-Петербург
2015**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ РАСТИТЕЛЬНЫХ
ПОЛИМЕРОВ»**

Г.А. Петров, С.В. Тихов, В.П. Яковлев

БАЗЫ ДАННЫХ

Учебное пособие

**Санкт-Петербург
2015**

УДК 681.3(075)

ББК 32.9я7

П 305

Петров Г.А., Тихов С.В., Яковлев В.П. Базы данных: учебное пособие / СПбГТУ РП. - СПб. 2015. – 74 с.

В пособии изложены теоретические и практические основы организации баз данных, являющиеся важнейшей подсистемой различных типов автоматизированных информационных систем. Целью данного пособия является формирование у студентов теоретических знаний и практических навыков по проектированию реляционных баз данных для различных предметных областей в соответствии с современными методами и подходами. Рассмотрены все основные этапы разработки базы данных: нормализация таблиц, схемы данных, объекты и средства среды Access.

Пособие предназначено для студентов направления подготовки 01.03.02 “Прикладная математика и информатика”, обучающихся по дисциплине “Базы данных”.

Рецензенты:

доцент кафедры «Информационные технологии и математика» СПбУУиЭ, кандидат технических наук Курлов В.В.;

доцент кафедры прикладной математики и информатики СПбГТУРП, кандидат технических наук, доцент Антонюк П.Е.

Рекомендовано к изданию Редакционно-издательским советом университета в качестве учебного пособия.

Редактор и корректор Т.А. Смирнова

Техн. редактор Л.Я. Титова

План 2015., поз. 10

Подп. к печати 16.02.15 г.

Бум. тип. № 1. Печать офсетная.

Тираж 30 экз. Изд. № 10.

Заказ №

Формат 60х84/16

Печ.л. 4,75. Уч.-изд. л. 4,75.

Цена «С».

Ризограф Санкт-Петербургского государственного технологического университета растительных полимеров, СПб., 198095, ул. И. Черных, 4.

© Санкт-Петербургский
государственный технологический
университет растительных полимеров, 2015

© Петров Г.А., Тихов С.В., Яковлев В.П., 2015

ВВЕДЕНИЕ

В настоящее время деятельность человека в различных областях науки, техники обычной жизни настолько насыщена различного рода информацией, что для ее хранения и эффективной обработки требуется создание огромного количества хранилищ данных различного назначения. Современные информационные системы характеризуются не только огромными объемами хранимых данных, но также и высокими требованиями к скорости обработки и передачи данных, их сложной структурной организации, необходимостью удовлетворять разнообразным требованиям многочисленных пользователей. Поэтому эффективно организованное хранение информации и ее обработка требуют применения наиболее перспективных компьютерных и информационных технологий.

В истории развития средств компьютерной техники можно выделить ряд основных направлений. Первое направление – это применение компьютерной техники для решения различных задач: инженерных, математических, физических и многих других, обеспечивающих интенсивное развитие многих областей науки и техники. Характерной особенностью данных областей применения компьютерной техники является наличие сложных алгоритмов обработки, которые применяются к различным по структуре данным, объем которых сравнительно невелик. Другое направление, которое непосредственно относится к теме нашего учебного пособия - это использование компьютерных технологий в автоматических и автоматизированных информационных системах, обеспечивающих выполнение следующих основных функций:

- надежное хранение больших объемов информации в памяти компьютера;
- выполнение специфической для данной области обработки и передачи данных;
- предоставление пользователям удобного интерфейса.

В литературе выделяют ряд этапов в развитии информационных систем. Вначале данные обрабатывались с использованием оборудования с перфокартами и электромеханических машин для сортировки и табулирования миллионов записей. На следующем этапе данные хранились на магнитных лентах, и программы выполняли пакетную обработку последовательных файлов. Затем было введено понятие базы данных и обеспечен автоматический доступ к базам, были внедрены распределенная и клиент-серверная обработка данных. В настоящее время используются информационные системы, которые хранят сложные структуры мультимедийных данных документы, графику, аудио- и видеоданные. Эти системы представляют собой базовые средства хранения для приложений Интернет и интранет. Мультимедийные базы данных и средства доступа к ним будут краеугольным камнем в нашем движении к киберпространству. Значительные расширения в применении баз данных требуют решения сложных проблем, которые можно сгруппировать в следующие задачи:

- поддержка мультимедийных объектов;
- распределенное хранение информации;
- новые виды приложений баз данных;
- управление транзакциями и потоками работ;
- эффективность управления базами данных и их использования.

В связи с техническими достижениями последних лет, такими как значительный рост емкости систем памяти, мощности компьютерных систем, развитие коммуникаций (Internet, мобильные коммуникации и т.п.) возникают новые области исследований в развитии систем управления баз данных.

В настоящем учебном пособии изложены теоретические и практические основы разработки баз данных, являющихся важнейшей подсистемой различных видов автоматизированных информационных систем. Рассмотрены все основные этапы разработки реляционных баз данных: логические модели баз данных, нормализация таблиц, схемы данных, создание баз данных из множества таблиц, применение форм, создание и использование запросов, отчетов.

1.ОСНОВНЫЕ ПОНЯТИЯ БАЗ ДАННЫХ

Базы данных (БД) используются в различных областях и сферах человеческой деятельности. В качестве примеров можно привести БД, содержащие информацию о клиентах, товарах, предоставляемых услугах, коммерческих операциях и многое другое. БД являются основой любой информационной системы и в широком смысле слова - это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области. БД представляет собой хранилище огромных массивов информации в связанном виде для совместного использования. Однако сама по себе БД не может обслужить запросы пользователей на поиск и обработку информации, так как это просто "склад информации". Чтобы эффективно использовать подобный склад, необходима система управления базой данных.

В литературе предлагается множество определений БД, которые отражают те или иные аспекты субъективного мнения различных авторов. Мы будем понимать под базой данных организованную в соответствии с определёнными правилами и хранимую в памяти компьютера совокупность логически связанных данных, отражающих состояние объектов в определенной предметной области и используемую для обеспечения информационных потребностей пользователей. В определениях БД наиболее часто присутствуют следующие характерные признаки:

1.БД хранятся и обрабатываются в компьютерной системе и любые некомпьютерные хранилища информации, например: библиотеки, картотеки и т.п. базами данных не являются;

2.Данные в БД логически связаны и структурированы с целью обеспечения возможности их эффективного поиска и обработки в системе. Структурированность подразумевает явное выделение составных элементов и

связей между ними, типизацию элементов и связей, для которых имеет место определённая семантика и допустимые операции;

3.БД включает схему описания логической структуры БД в формальном виде.

БД является основой для построения информационных систем, под которыми понимают совокупность базы данных и комплекса аппаратно-программных средств для хранения, модификации и поиска информации, взаимодействия пользователя с системой. Для управления базами данных необходима система управления, представляющая собой совокупность программных и языковых средств, предназначенных для создания, ведения и совместного использования базы данных коллективом пользователей. Эффективное управление системой памяти является основной функцией системы управления базами данных (СУБД). Эти обычно специализированные средства настолько важны с точки зрения эффективности, что при их отсутствии система просто не сможет выполнять некоторые задачи уже потому, что их выполнение будет занимать слишком много времени. При этом ни одна из таких специализированных функций не является видимой для пользователя. Они обеспечивают независимость между логическим и физическим уровнями системы. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

История развития систем управления базами данных насчитывает десятки лет. Первая промышленная СУБД фирмы IBM была введена в эксплуатацию в 1968 году, а в 1975 году появился первый стандарт, который определил ряд основных понятий в теории систем баз данных. Развитие вычислительной техники, появление персональных компьютеров, мощных рабочих станций и компьютерных сетей обусловило развитие технологии баз данных. С появлением локальных сетей возникла задача согласования данных, хранящихся и обрабатываемых в разных местах, но связанных логически. Решение этой задачи привело к появлению распределённых баз данных, позволяющих организовать параллельную обработку информации и сохранять целостность баз данных.

Лицом, отвечающим за выработку требований к базе данных, её проектирование, реализацию, эффективное использование и сопровождение, защиту от несанкционированного доступа является администратор базы данных. Важной функцией администратора баз данных является также поддержка целостности базы данных.

Классификация баз данных. По технологии обработки данных БД можно подразделить на централизованные и распределённые. Централизованная БД хранится в памяти одной компьютерной системы и может использоваться в локальных сетях персональных компьютеров. Централизованные БД могут быть с сетевым доступом. Архитектуры систем централизованных БД с сетевым доступом подразделяются на файл-сервер и клиент-сервер. Архитектура систем БД с сетевым доступом (файл-сервер)

предполагает выделение одного из компьютеров сети в качестве центрального, выполняющего функцию сервера файлов. В нем хранится совместно используемая централизованная БД. Все другие компьютеры сети являются рабочими станциями. Файлы БД в соответствии с пользовательскими запросами передаются на рабочие станции, где и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность системы падает.

В архитектуре Клиент-сервер подразумевается, что помимо хранения централизованной БД центральный компьютер- сервер базы данных должен выполнять основной объём обработки данных. Запрос на данные клиента порождает поиск данных на сервере. Извлечённые данные передаются по сети от сервера к клиенту. Примером подобной БД является БД сотрудников организации, в которой, например, указаны: Ф.И.О., должность, дата рождения, адрес, телефон и т.п.

Распределённая БД состоит из нескольких частей, хранимых в различных компьютерах сети, и работа такой БД происходит с помощью СУБД. По способу доступа к данным БД подразделяются на БД с локальным и удалённым доступом. БД называют с локальным доступом, если компьютер является компонентом сети и к такой БД возможен распределённый доступ. Для распределённого хранения данных и доступа к базе компьютеры объединяют в локальные, региональные и даже глобальные сети. Для построения подобных сетей также используется технология клиент-сервер. Система клиент-сервер - это обычная локальная вычислительная сеть, которая содержит группу компьютеров-клиентов и один специальный компьютер – сервер. Компьютеры-клиенты обращаются к серверу за различными услугами. Компьютер-сервер может пересылать им различные программы, например, обработку текстов, работы с таблицами, выполнение запросов к базе данных и возвращать результаты. Основная идея состоит в том, что каждый компьютер выполняет то, что он делает наиболее эффективно. Сервер извлекает и обновляет данные, клиент выполняет специальные расчеты и предоставляет результаты конечному пользователю. Вначале серверы выполняли простейшие функции: серверы печати, файловые серверы, по запросу клиента на доступ к какому-нибудь файлу сервер пересылал данный файл компьютеру-клиенту. Сервер базы данных – это программа, которая запускается на компьютере-сервере и обслуживает доступ клиентов к базе данных. Таким образом, в основе системы клиент-сервер лежит принцип разделения труда. Клиент-это компьютер, с которым работает пользователь, а компьютер-сервер выполняет обслуживание группы клиентов: доступ к базе данных, обновление базы данных и т.п. Прогрессивным путем коллективного доступа к БД в последние 20 лет является использование всемирной сети Интернет с группой ее служб. Примерами серверов могут служить:

- сервер телекоммуникаций, обеспечивающий сервис по связи локальной сети с другими сетями и серверами;

- вычислительный сервер, дающий возможность производить вычисления, которые невозможно выполнить на рабочих станциях;
- дисковый сервер, обладающий расширенными ресурсами внешней памяти и предоставляющий их в использование компьютерам-клиентам и, возможно, другим серверам;
- файловый сервер, поддерживающий общее хранение файлов для всех рабочих станций;
- сервер баз данных - фактически обычная СУБД, принимающая и обслуживающая запросы по локальной сети.

Хотя обычно одна база данных целиком хранится в одном узле сети и поддерживается одним сервером, серверы баз данных представляют собой простое и дешевое приближение к распределенным базам данных, поскольку общая база данных доступна для всех пользователей локальной сети. Одним из перспективных направлений СУБД является гибкое конфигурирование системы, при котором распределение функций между клиентской и пользовательской частями СУБД определяется при установке системы. СУБД должны обеспечивать логическую целостность данных, подразумевающую поддержание непротиворечивой и полной информации, адекватно отражающей предметную область.

С требованием логической целостности данных связано понятие транзакции. Транзакция – это группа логически объединённых последовательных операций по работе с данными, обрабатываемая или отменяемая целиком. Например, если оформлять заказ на определенный товар, нужно выполнить ряд операций: регистрация заявки на товар, резервирование товара, уменьшение этого товара на складе. При нарушении на любом из этапов произойдет сбой, и логическая целостность БД будет нарушена. С целью предотвращения подобных случаев вводится транзакция «Оформление заказа», в которой над БД должны произвестись все необходимые операции, т.е. товар продается, количество его на складе уменьшается или происходит возврат к исходному состоянию (товар не продан и его количество на складе осталось прежним).

Основные функции СУБД. СУБД представляют собой оболочку, с помощью которой после организации структуры таблиц и заполнения их данными формируется определенная БД. Программные средства включают систему управления вводом-выводом, хранение и обработку информации, создание и тестирование БД, трансляторы. В качестве языков программирования могут использоваться объектно-ориентированные языки C++, Java или язык структурированных запросов SQL (StructuredQueryLanguage). Появление данного языка обусловлено ростом количества данных, необходимостью их хранения и обработки. С помощью SQL пользователи могут манипулировать данными независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции или универсальном компьютере. Данный язык в настоящее время получил очень широкое распространение и фактически превратился в стандартный язык

реляционных баз данных. Стандарт на язык SQL был разработан Американским национальным институтом стандартов (ANSI) в 1986 г., а в 1987 г. Международная организация стандартов ISO приняла его в качестве международного. Язык запросов выполняет поиск совокупностей параметров в файле в соответствии с заданным набором критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД.

К основным функциям СУБД можно отнести следующие:

- **Управление данными во внешней памяти.** Данная функция обеспечивает поддержку требуемых структур внешней памяти как для хранения данных в БД, так и для выполнения служебных функций.
- **Управление буферной памятью.** Размер БД обычно значительно превосходит емкость оперативной памяти, и при обращении к данным система будет работать со скоростью внешней памяти. Одним из способов повышения скорости является буферизация данных в оперативной памяти. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с индивидуальным протоколом управления буферной памятью.
- **Управление транзакциями.** Применение СУБД для работы с интегрированными БД выявило особую важность проблемы ее целостности, под которой понимают правильность и непротиворечивость содержимого БД. Нарушение целостности может быть вызвано, например, ошибками или сбоями в системе, так как в этом случае она не в состоянии обеспечить нормальную обработку или выдачу правильных данных. Аспект целостности обеспечивается на уровне структур данных и отдельных операторов языковых средств СУБД и при нарушении такой целостности оператор отвергается. Для обеспечения целостности БД и служит аппарат транзакций. Таким образом, под транзакцией понимают неделимую с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации). При этом для поддержания ограничений целостности на уровне БД допускается их нарушение внутри транзакции так, чтобы к моменту завершения транзакции условия целостности были соблюдены.
- **Журнализация.** Одним из основных требований к развитым СУБД является надежность хранения баз данных. Это требование предполагает, в частности, возможность восстановления согласованного состояния БД после любого рода аппаратных и программных сбоев. Очевидно, что для выполнения восстановлений необходима некоторая дополнительная информация. В подавляющем большинстве современных реляционных СУБД такая избыточная дополнительная информация поддерживается в виде журнала изменений базы данных. Итак, общей целью журнализации изменений баз данных является обеспечение возможности восстановления согласованного состояния БД после любого сбоя. Поскольку основой поддержания целостного состояния базы данных является механизм транзакций, журнализация и восстановление тесно связаны. Общим принципом

восстановления является сохранение результатов зафиксированных транзакций в восстановленном состоянии БД. Примерами ситуаций, при которых требуется производить восстановление состояния БД, являются следующие. Восстановление после внезапной потери содержимого оперативной памяти - мягкий сбой. Такая ситуация может возникнуть при аварийном выключении электропитания, при возникновении неустранимого сбоя процессора и т.д. Ситуация характеризуется потерей той части БД, которая к моменту сбоя содержалась в буферах оперативной памяти. Восстановление после отказа основного внешнего носителя базы данных - жесткий сбой. Эта ситуация при достаточно высокой надежности современных устройств внешней памяти может возникать сравнительно редко, но тем не менее, СУБД должна быть в состоянии восстановить БД даже и в этом случае. Основой восстановления является архивная копия и журнал изменений БД. Во всех случаях основой восстановления является избыточное хранение данных, которые хранятся в журнале, содержащем последовательность записей об изменении базы данных. Журнал - это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, иногда поддерживаются две копии журнала, располагаемые на разных физических дисках. В журнал поступают записи обо всех изменениях в БД. В разных СУБД изменения БД отображаются в журнале на разных уровнях. Иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда - минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

- **Языки БД.** Для работы с БД используются специальные языки, в целом называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка: язык определения схемы БД и язык манипулирования данными. В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания и завершая разработкой пользовательского интерфейса с БД. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык запросов SQL.

Язык SQL содержит специальные средства определения ограничений целостности БД. Ограничения целостности хранятся в специальных таблицах-каталогах, и обеспечение контроля целостности БД производится на языковом уровне. При компиляции операторов модификации БД компилятор SQL на основании имеющихся в БД ограничений целостности генерирует соответствующий программный код. Операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми в БД запросами (результатом любого запроса к реляционной БД является таблица) с именованными столбцами. Авторизация доступа к объектам БД производится также на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL

разного вида пользователь должен обладать различными полномочиями. Пользователь, создавший таблицу БД, обладает полным набором полномочий для работы с этой таблицей. В число этих полномочий входит разрешение на передачу всех или части полномочий другим пользователям. Полномочия пользователей описываются в специальных таблицах-каталогах, контроль полномочий поддерживается на языковом уровне.

Функциональные возможности СУБД. По степени универсальности различают СУБД двух видов:

- СУБД общего назначения, реализованные как программный продукт, способный функционировать на компьютере с определённой операционной системой и поставляемый пользователям в качестве законченного коммерческого изделия;
- специализированные СУБД, создаваемые в случаях невозможности или нецелесообразности разработки СУБД общего назначения.

СУБД общего назначения представляют собой сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией БД информационной системы. Рынок программного обеспечения персональных компьютеров располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем СУБД общего назначения. Производительность СУБД оценивается следующими параметрами:

- временем выполнения запросов;
- временем выполнения операций импортирования данных, имеющих отличные форматы;
- скоростью выполнения таких операций как обновление, вставка, удаление данных и других;
- степенью параллелизма при обращении к данным в многопользовательском режиме;
- временем генерации отчёта.

СУБД, поддерживающие целостность данных, предполагают наличие средств, позволяющих определять, что информация в БД всегда остаётся корректной и полной. Примерами операций, обеспечивающих безопасность СУБД, являются шифрование прикладных программ и данных, защита паролем, ограничение доступа к БД и другие. Например, хороший уровень безопасности имеют СУБД Oracle, Access. Целостность должна обеспечиваться независимо от того, каким образом данные заносят в память, от действий пользователей, сбоев сети и т.п. Может предусматриваться назначение отдельных паролей для пользователей, присвоение различных прав доступа для таблиц, запросов, отчётов и т.п.

2. ЛОГИЧЕСКИЕ МОДЕЛИ БАЗ ДАННЫХ

Прежде чем создавать базу данных необходимо выбрать наиболее эффективную модель данных для решения поставленной задачи. Модель данных – это совокупность структур данных и операций их обработки. С

помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

В классической теории баз данных под моделью данных понимают формальный метод представления и обработки данных в системе управления базами данных. Следует учитывать, по крайней мере, три следующих аспекта при выборе модели:

- выбор методов описания типов и логических структур данных в БД;
- определение способов обработки данных в БД;
- обеспечение метода поддержки целостности БД.

Первый аспект определяет, что из себя логически представляет база данных, способ обработки определяет способы перехода между состояниями базы данных, то есть способы модификации данных и извлечения данных из базы данных, аспект целостности определяет средства описаний корректных состояний базы данных.

Модель данных - это абстрактное логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную структуру доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы - поведение данных. Каждая БД и СУБД строится на основе некоторой явной модели данных и построенные на одной и той же модели данных, относятся к одному типу. Длительное время термин «модель данных» не имел формального определения. Одним из первых специалистов, который ввел достаточно формальное понятие в статье «Модели данных в управлении базами данных» был Э. Кодд. Он определил модель данных как комбинацию трех компонентов:

- множество типов объектов данных, образующих базовые строительные блоки для любой базы данных соответствующей модели;
- набор общих правил целостности, ограничивающих набор экземпляров тех типов объектов, которые законным образом могут появиться в любой такой базе данных;
- множество операций, применимых к таким типам объектов для выборки и других функций.

Модель данных представляет сочетание следующих компонентов: структурную часть, т.е. набор правил, по которым может быть построена база данных, и управляющую часть, определяющую типы дополнительных операций над данными (обновление, изменение связей, удаление). Выбор модели данных возлагается на разработчика и зависит от технического и программного обеспечения, определяется сложностью задач и объемом информации. Организация способов хранения данных в компьютерных системах задается на логическом и на физическом уровнях. Физический уровень определяет способ размещения данных непосредственно на машинном носителе. Этот уровень обеспечивается автоматически прикладными программами без вмешательства пользователя. Пользователь в прикладных программах оперирует представлениями логической

организации данных. Для размещения одной и той же информации в компьютере могут быть использованы различные структуры и модели данных. Логическая модель данных должна отражать предметное содержание, структуру связей, методы доступа к данным и таким образом саму организацию, для которой создается база данных. Различают следующие основные четыре модели данных:

- иерархическая;
- сетевая;
- объектно-ориентированная;
- реляционная.

Эти модели отличаются между собой по способу установления связями между данными.

1. Иерархическая модель данных. Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагает жесткое подчинение между данными в системе иерархий. Подобные структуры удовлетворяют требованиям многих, но далеко не всех реальных задач.

2. Сетевая модель данных. В сетевых базах данных наряду с вертикальными связями допустимы и горизонтальные связи. Однако в данной модели имеют место многие недостатки иерархической модели и главный из них - это необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе.

3. Объектно-ориентированная модель. Новые области использования средств вычислительной техники, такие как научные исследования, автоматизированное проектирование и другие потребовали от баз данных способности хранить и обрабатывать новые объекты - текст, аудио- и видеoinформацию, а также документы. Основные трудности объектно-ориентированного моделирования данных проистекают из того, что такого развитого математического аппарата, на который могла бы опираться общая объектно-ориентированная модель данных, не существует. В большой степени, поэтому до сих пор нет базовой объектно-ориентированной модели.

4. Реляционная модель. Реляционная модель появилась вследствие стремления сделать базу данных как можно более гибкой. Данная модель предоставила простой и эффективный механизм поддержания связей данных. Все данные в модели представляются только в виде таблиц и реляционная модель - единственная из всех, обеспечивающих единообразие представления данных. И сущности, и связи этих самых сущностей представляются в модели совершенно одинаково - таблицами. Рассмотрим более подробно указанные выше модели данных.

2.1. Иерархическая модель данных

Первые иерархические модели данных появились в конце 50-х годов. Они представляли собой древовидную структуру, где данные были распределенные по уровням от главного - к подчиненному и представляли

собой неориентированный граф. Пример иерархической модели данных приведен на рис.1.

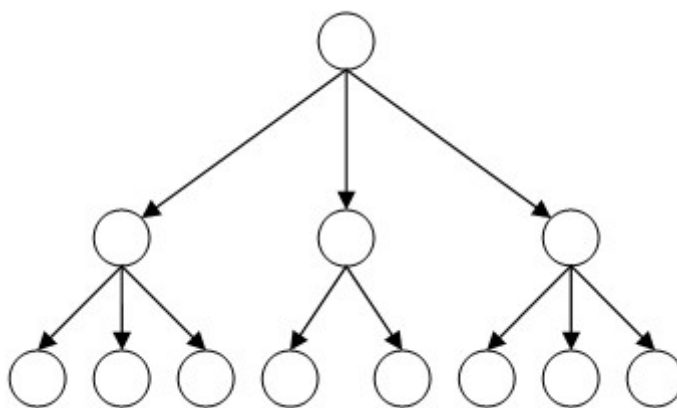


Рис 1. Иерархическая модель данных

Данная модель характеризуется количеством уровней и узлов. Каждый уровень представляет собой один или несколько объектов (данных) и может иметь несколько узлов подчиненных уровней, причем связи между всеми объектами жестко закреплены и один потомок может иметь не более одного предка. Основные типы структур данных рассматриваемой модели – поле, запись, файл. Запись является основной структурной единицей обработки данных и единицей обмена между оперативной и внешней памятью. В модели на основе записей база данных состоит из записей фиксированного формата, которые могут быть разного типа. Каждый тип записи определяет фиксированное количество полей, каждое из которых имеет фиксированную длину. Организация данных в СУБД иерархического типа определяется в следующих терминах:

Поле (атрибут) – это элементарная единица логической организации данных, которая соответствует отдельной, неделимой единице информации – реквизиту.

Запись – это совокупность полей, соответствующих логически связанным реквизитам. Структура записи определяется составом и последовательностью входящих в нее полей, каждое из которых содержит элементарное данное.

Файл – это множество одинаковых по структуре записей со значениями в отдельных полях, причем поля имеют единственное значение.

Групповое отношение – иерархическое отношение между записями двух типов. Родительская запись называется исходной записью, а дочерние записи – подчиненными. Иерархическая база данных может хранить только такие древовидные структуры.

Корневая запись каждого дерева обязательно должна содержать ключ с уникальным значением. Ключи некорневых записей должны иметь уникальное значение только в рамках группового отношения. Каждая запись

идентифицируется полным ключом, под которым понимается совокупность ключей всех записей от корневой по иерархическому пути.

При графическом изображении групповые отношения изображают дугами ориентированного графа, а типы записей - вершинами (диаграмма Бахмана). Для групповых отношений в иерархической модели обеспечивается автоматический режим включения записи. Это означает, что для запоминания любой некорневой записи в БД должна существовать ее родительская запись.

Примером иерархической базы данных является Каталог папок Windows, с которым можно работать, запустив Проводник. Верхний уровень занимает папка Рабочий стол. На втором уровне находятся папки Мой компьютер, Мои документы, Сетевое окружение и Корзина, которые являются потомками папки Рабочий стол, а между собой является близнецами. В свою очередь, папка **Мой компьютер** является предком по отношению к папкам третьего уровня - папкам дисков.

Типичным и наиболее известным примером с иерархической моделью данных является СУБД IMS (InformationManagementSystem) компании IBM.

2.2.Сетевая модель данных

На разработку данной модели большое влияние оказал американский ученый Ч. Бахман. Основные принципы сетевой модели данных были разработаны в середине 60-х годов. Под сетевой моделью понимается модель данных, подобная иерархической, но допускающая свободную систему связей между узлами различных уровней. Таким образом, она является расширением иерархической модели данных (рис.2). В отличие от иерархической модели у потомка сетевой модели может быть более одного предка и один объект может быть одновременно главным и подчиненным. Таким образом, в данной модели отношения между данными такие, что каждая запись может быть подчинена записям более, чем из одного файла. В сетевых моделях можно по ключу иметь непосредственный доступ к любому объекту независимо от уровня, на котором он находится в модели.

К достоинству сетевой модели можно отнести эффективность реализации по степени затрат памяти и быстроты доступа. Недостатком является повышенная сложность схемы данных, построенной на её основе. Типичным представителем систем, основанных на сетевой модели данных, является СУБД IDMS (IntegratedDatabaseManagementSystem), разработанная компанией CullinetSoftwareInc. и изначально ориентированная на использование мэйнфреймов (компьютеров общего назначения) компании IBM. Архитектура системы основана на предложениях DataBaseTaskGroup (DBTG) организации CODASYL (ConferenceonDataSystemsLanguages), которая отвечала за определение языка программирования COBOL. Отчет DBTG был опубликован в 1971 г., и вскоре после этого появилось несколько систем, поддерживающих архитектуру CODASYL, среди которых

присутствовала и СУБД IDMS. В настоящее время IDMS принадлежит компании ComputerAssociates.

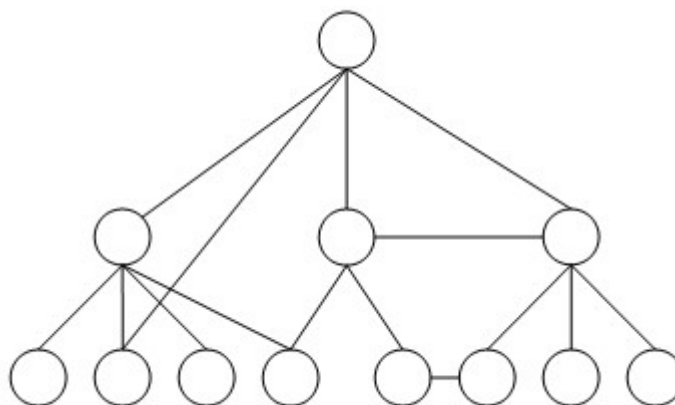


Рис.2. Сетевая модель данных

2.3. Объектно-ориентированная модель данных

В настоящее время общепринятого определения объектно-ориентированной модели данных не существует и поэтому можно только говорить лишь о некотором "объектном" подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации. Объектно-ориентированная модель позволяет создавать, хранить и использовать информацию в форме объектов (рис. 3).

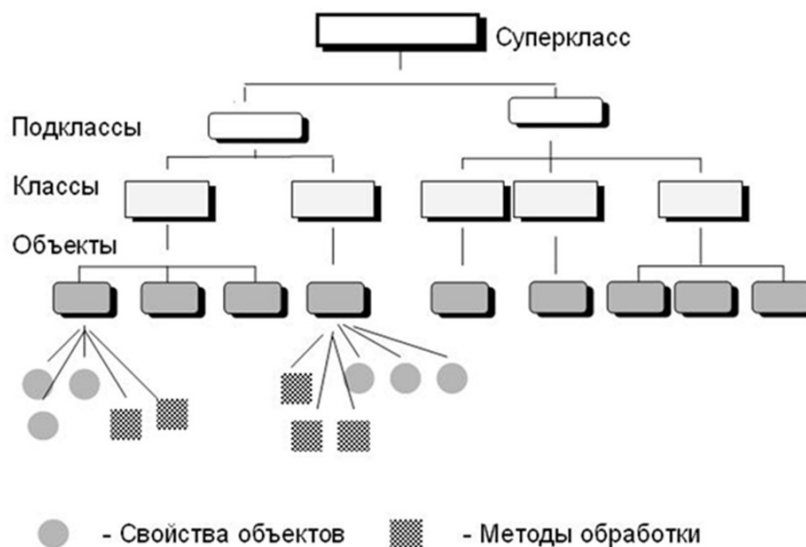


Рис.3. Объектно-ориентированная модель данных

Объект при своем создании получает генерируемый системой уникальный идентификатор, который связан с объектом все время его существования и не меняется при изменении состояния объекта. Каждый объект имеет состояние и поведение. Состояние объекта - набор значений его атрибутов, а поведение объекта – это набор методов (программный код), оперирующих над состоянием объекта. Значение атрибута объекта - это тоже некоторый объект или множество объектов. Состояние и поведение объекта

инкапсулированы в объекте. Взаимодействие объектов производится на основе передачи сообщений и выполнении соответствующих методов.

Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов. Объект должен принадлежать только одному классу, если не учитывать возможности наследования. Допускается наличие примитивных предопределенных классов, объекты которых не имеют атрибутов: целые, строки и т.д. Класс, объекты которого могут служить значениями атрибута объектов другого класса, называется доменом этого атрибута.

Допускается порождение нового класса на основе уже существующего класса - наследование. В этом случае новый класс, называемый подклассом существующего класса, наследует все атрибуты и методы суперкласса. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы. Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько. Если в языке или системе поддерживается единичное наследование классов, набор классов образует древовидную иерархию. При поддержании множественного наследования классы связаны в ориентированный граф с корнем, называемый решеткой классов. Объект подкласса считается принадлежащим любому суперклассу этого класса.

Наиболее широкое применение объектно-ориентированные базы данных нашли в таких областях, как системы автоматизированного конструирования/производства (CAD/CAM), системы автоматизированной разработки программного обеспечения (CASE), системы управления составными документами, т.е. в областях не традиционных для баз данных. Примерами объектно-ориентированных СУБД являются POET, Jasmine, Versant, Iris, Orion.

2.4. Реляционная модель данных

В 1970 году американский математик Кодд Е.Ф. опубликовал революционную по своему содержанию статью, в которой предложил использовать для обработки данных теорию множеств. Он утверждал, что данные нужно связывать в соответствии с их логическими взаимоотношениями (например, объединение, пересечение), а не физическими указателями. Он предложил простую модель данных, в которой все данные сведены в таблицы, состоящие из строк и столбцов, имеющих уникальные имена. Эти таблицы получили название реляций (relatio - отношение), а модель – реляционной моделью данных, построенной на понятии математических отношений, и иногда ее называют моделью Кодда. Предложения Кодда были настолько эффективны для баз данных, что за эту модель он был удостоен престижной премии Тьюринга в области теоретических основ вычислительной техники.

В реляционных базах все данные хранятся в простых таблицах, разбитых на строки (их называют записями) и столбцы (их называют полями), на пересечении которых расположена информация о данных. В общем виде это может быть представлено на рис.4.

	поле 1	поле 2	...	поле N
запись 1				
запись 2				
...	...	...	...	...
запись M				

Рис.4. Реляционная БД

Каждый столбец имеет индивидуальное имя и таблица «Товар на складе» иллюстрирует (рис. 5) имена полей: Идентификатор, Товар, Название группы, Единица измерения, Закупочная цена, Цена реализации, Наличие на складе.

Идентификатор	Товар	Название группы	Единица измерения	Закупочная цена	Цена реализации	Наличие на складе
1	OLYMPUS PT80	Цифровые фотокамеры	Шт.	2 580,00 р	6 750,00 р.	3
2	Карта памяти	Аксессуары	Шт.	890,00 р.	2 100,00 р.	10
3	Фотопленка KodakGoldUltra 400	Аксессуары	Шт.	45,50 р.	120,00 р.	20

Рис. 5. Таблица «Товар на складе»

Все значения в одном столбце имеют один тип и, таким образом, поля являются различными характеристиками объекта, иногда их называют атрибутами. Значения полей в одной строке относятся к одному объекту, а различные поля отличаются именами.

Каждая запись может идентифицироваться уникальным ключом записи, которые бывают двух типов: первичный и вторичный.

Первичный ключ – это одно или несколько полей, однозначно идентифицирующих запись. Если первичный ключ состоит из одного поля, он называется простым, если из нескольких полей – составным ключом.

Вторичный ключ – это поле, значение которого может повторяться в нескольких записях файла, т.е. он не является уникальным.

Внешний ключ подчиненной таблицы - это вторичный ключ данного отношения, который, в то же время, выполняет функции первичного ключа в главной таблице. Если по значению первичного ключа может быть найден один единственный экземпляр записи, то по значению внешнего ключа несколько (рис.6). Как правило, реляционная база данных состоит из нескольких таблиц, т.к. объединить в одной таблице все сведения,

необходимые сотрудникам (пользователям БД) какой-либо организации для решения задач, не представляется возможным. Средством эффективного доступа по ключу к записи файла является индексирование. При индексировании создается дополнительный файл, который содержит в упорядоченном виде все значения ключа файла данных. Для каждого ключа в индексном файле содержится указатель на соответствующую запись файла данных. С помощью указателя на запись в файле данных осуществляется прямой доступ к этой записи.

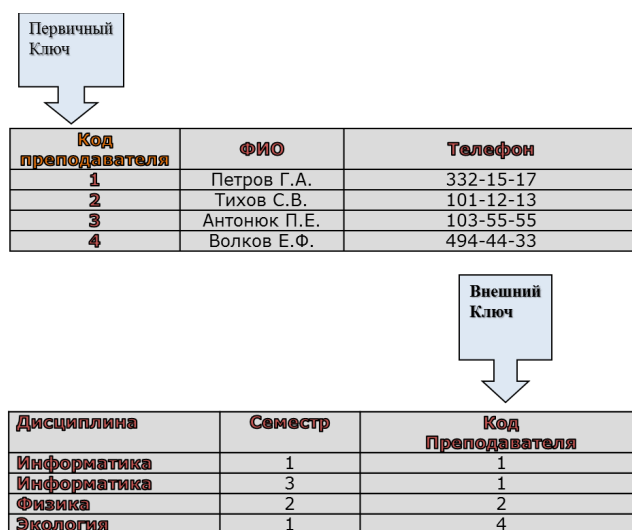


Рис.6. Пример использования внешнего ключа

Для работы с реляционными базами данных в настоящее время обычно используется язык структурированных запросов SQL, применяемый для создания, модификации и управления данными. Язык SQL не является алгоритмическим языком программирования - это информационно-логический язык и основывается на реляционной алгебре и подразделяется на три части:

- операторы определения данных;
- операторы манипуляции данными;
- операторы определения доступа к данным.

Таким образом, таблица является основным типом структуры данных реляционной модели и определяется совокупностью столбцов. В каждой строке таблицы содержится по одному значению в соответствующем столбце. В таблице не может быть двух одинаковых строк, общее число строк не ограничено. Столбец – это элемент данных, каждый столбец имеет имя. Один или несколько атрибутов, значения которых однозначно идентифицируют строку таблицы, являются ключом таблицы.

Достоинствами реляционной модели являются:

- простота и доступность понимания конечным пользователем - единственной информационной конструкцией является таблица;

- при проектировании реляционной БД применяются строгие правила, базирующиеся на математическом аппарате;
- полная независимость данных, при изменении структуры таблицы в прикладных программах модификации минимальны;
- для построения запросов и написания прикладных программ нет необходимости знания конкретной организации БД во внешней памяти.

Недостатками реляционной модели являются:

- относительно низкая скорость доступа и большой объем внешней памяти;
- трудность понимания структуры данных из-за появления большого количества таблиц в результате логического проектирования;
- не всегда предметную область можно представить в виде совокупности таблиц.

Реляционные базы данных в настоящее время получили наибольшее распространение. Сетевые и иерархические модели считаются устаревшими, объектно-ориентированные модели пока не стандартизированы и не получили значительного распространения.

3. ОСНОВНЫЕ ЭТАПЫ РАЗРАБОТКИ БАЗ ДАННЫХ

Можно выделить следующие этапы разработки баз данных:

- проектирование;
- программная реализация;
- заполнение и эксплуатация.

Этап проектирования – это теоретическое построение исходной информационной модели базы данных и включает в себя:

- сбор информации о предметной области, ее структуре, входных и выходных информационных потоках данных, изучение задач автоматизации, анализ и выделение объектов исходной системы и определение связей между ними;
- определение свойств и характеристик для каждого объекта в БД, которым назначаются поля (атрибуты), разрабатывают исходные таблицы и отношения между ними, выполняется определение элементов данных, включаемых в базу данных, ограничения на значения данных и т.п.
- назначение первичных ключей (полей) для каждого объекта и нормализация (разбиение) исходных таблиц;
- проверку корректности проекта, который должен отображать все выделенные объекты, их атрибуты и описываемые процессы на требуемом уровне детализации, отображать предметную область решаемой задачи;
 - определение логической структуры базы данных;

- решение вопросов защиты и поддержки целостности базы данных. Под обеспечением целостности данных понимается система мер, направленных на поддержание правильности данных в базе в любой момент времени.

Этап программной реализации связан с разработкой приложений на компьютере, для чего необходимо выполнить следующие действия:

- описать полученные таблицы средствами СУБД и ввести их в компьютер;
- для пользователей информационной системы разработать интерфейсы работы с БД, т.е. экранные формы для ввода и отображения данных, отчеты для печати сводных данных, запросы для получения данных;
- выработать порядок ведения и поддержания БД в рабочем состоянии, работы конечных пользователей;
- провести тестирование системы, составить инструкции по работе с ней и обучить персонал.

Этап эксплуатации и заполнения начинается с наполнения базы данных конкретными данными. Он включает в себя непосредственное ведение базы данных и её сопровождение. При разработке БД для крупных предприятий и корпораций анализ и моделирование выполняется с использованием специальных программных средств, например CASE-средств, которые позволяют промоделировать потоки данных, процессы и функции предприятия, выявить узкие места и дать рекомендации по эффективной организации структуры и бизнес-процессов на предприятии.

Кроме построения моделей текущего состояния предприятия и анализа программные средства моделирования позволяют сформировать спецификации и построить проект будущей системы, более того, может быть получен программный код для наиболее распространенных СУБД. Таким образом, стадия моделирования может захватывать этап проектирования и часть этапа реализации информационной системы.

3.1. Концептуальное проектирование базы данных

Первая фаза процесса проектирования базы данных называется концептуальным проектированием базы данных. Она заключается в создании концептуальной модели данных для анализируемой части объектов исследуемой системы. Эта модель данных создается на основе информации, записанной в спецификациях требований пользователей. Концептуальное проектирование базы данных абсолютно не зависит от таких подробностей ее реализации, как тип выбранной СУБД, набор создаваемых прикладных программ, используемые языки программирования, тип выбранной вычислительной платформы, а также от любых других особенностей физической реализации. Созданная концептуальная модель данных является источником информации для фазы логического проектирования базы данных.

3.2. Логическое проектирование базы данных

Вторая фаза проектирования базы данных называется логическим проектированием базы данных. Ее цель состоит в создании логической модели данных. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в СУБД, например, реляционная или сетевая модель.

Если концептуальная модель данных не зависит от любых физических аспектов реализации, то логическая модель данных создается на основе выбранной модели организации данных в СУБД. Иначе говоря, на этом этапе уже должно быть известно, какая СУБД будет использоваться - реляционная, сетевая, иерархическая или объектно-ориентированная. Однако на этом этапе игнорируются все остальные аспекты выбранной СУБД - например, любые особенности физической организации ее структур хранения данных и построения индексов.

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей. Для проверки корректности логической модели данных используется метод нормализации. Нормализация гарантирует, что выведенные из существующей модели данных отношения не будут обладать избыточностью данных, способной вызвать аномалии обновления после их физической реализации. Помимо всего прочего логическая модель данных должна обеспечивать поддержку всех необходимых пользователям транзакций.

Построенная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами нахождения компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. Логическая модель данных также играет важную роль на этапе эксплуатации и сопровождения уже готовой системы. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения и производить оценку их влияния на прикладные программы.

3.3. Нормализация базы данных

При проектировании баз данных наиболее важным является определение структур таблиц и связей между ними. Ошибки в структуре данных трудно, а чаще вообще невозможно исправить программным путем. Чем лучше структура данных, тем легче программировать БД. Теория проектирования БД содержит концепцию нормальных форм, предназначенных для оптимизации структуры БД. Нормальные формы - это линейная последовательность правил, применяемых к БД, причем, чем выше номер нормальной формы, тем совершеннее структура БД.

Нормализация отношений- формальный аппарат ограничений на формирование отношений (таблиц), который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение (ввод, редактирование и т.п.) базы данных.

Нормализация - это многоступенчатый процесс, при котором таблицы БД организуются, разъединяются и данные приводятся в порядок. Задача нормализации - устранить из БД некоторую избыточную информацию. В частности, ставится задача устранить некоторые виды избыточности данных и благодаря этому избежать аномалий при изменении данных. Аномалии изменения данных - это сложности при операциях вставки, изменения и удаления данных, возникающие из-за структуры БД. Хотя существует много уровней, обычно достаточно выполнить нормализацию до третьей нормальной формы. Е. Коддом выделены три **нормальных формы отношений** и предложен механизм, позволяющий любое отношение преобразовать к третьей (самой совершенной) нормальной форме.

Рассмотрим пример нормализации БД управления доставкой заказов. Неупорядоченная БД «Продажи» состояла бы из одной таблицы (рис.7).

Клиент	Код товара	Наименование товара	Количество	Цена	Всего
Иванов	222,333,444	Телефон, Стол, Стул	3,5,7	8,10,4	160,100,900

Рис.7. БД «Продажи»

В таблице каждая запись содержит сведения о нескольких заказах одного клиента. Поскольку столбец со сведениями о товаре содержит слишком много данных получить упорядоченную информацию из этой таблицы сложно, например, составить отчет о суммарных закупках по различным видам товаров.

3.3.1. Первая нормальная форма

Первая нормальная форма предопределяет атомарность всех данных, содержащихся в столбцах. Слово "атом" происходит от латинского "atomis", что буквально означает "не подлежащий разделению". Первая нормальная форма задает существование в каждой позиции, определяемой строкой и столбцом, только одного значения, а не массива или списка значений. Преимущества этого требования очевидны: если в одном столбце хранятся списки значений, то не существует простого способа манипулировать этими значениями. Конечно, при этом увеличивается количество записей в таблице.

Выполним нормализацию БД " Продажи" до первой нормальной формы (рис.8).

Клиент	Код товара	Наименование товара	Количество	Цена	Всего
Иванов	222	Телефон	8	20	160

Иванов	333	Стол	10	10	100
Иванов	444	Стул	3	300	900

Рис.8. Первая нормальная форма

3.3.2. Вторая нормальная форма

Ко второй нормальной форме можно перейти от таблицы, которая уже соответствует первой нормальной форме. Дополнительно должно выполняться следующее условие: каждое не ключевое поле должно полностью зависеть от первичного ключа.

Выполним нормализацию БД "Продажи" до второй нормальной формы. Все сведения, не связанные с отдельными заказами, выделим в отдельную таблицу. В итоге, вместо одной таблицы "Продажи" получим две: таблицу "Заказы" (рис.9) и таблицу "Товары" (рис.10).

Клиент	Код товара	Количество	Всего
Иванов	222	8	160
Иванов	333	10	100
Иванов	444	3	900

Рис.9. Таблица "Заказы"

Код товара	Наименование товара	Цена
222	Телефон	20
333	Стол	10
444	Стул	300

Рис.10. Таблица "Товары"

Таким образом, вид товара хранится только в одной таблице. Следует обратить внимание, что при нормализации информация не теряется.

3.3.3. Третья нормальная форма

Считается, что таблица соответствует третьей нормальной форме, если она соответствует второй нормальной форме и все не ключевые столбцы взаимно независимы. Столбец, значения которого получаются вычислением на основе данных из других столбцов, представляет собой один из примеров зависимости.

Выполним нормализацию БД "Продажи" до третьей нормальной формы. Для этого следует удалить из таблицы "Заказы" столбец "Всего". Значения в этом столбце не зависят ни от одного ключа и могут быть вычислены по формуле ("Цена")*("Количество"). Таким образом, получена БД "Продажи" с оптимальной структурой, которая состоит из двух таблиц (рис.11).

Клиент	Код товара	Количество
Иванов	222	8
Иванов	333	10
Иванов	444	3

Код товара	Наименование товара	Цена
222	Телефон	20
333	Стол	10
444	Стул	300

Рис. 11. Нормализованная БД "Продажи"

3.4. Типы связей и свойства отношений

Реляционные базы данных могут состоять из множества таблиц, связь между которыми устанавливается с помощью совпадающих полей. Каждая запись в таблицах идентифицирует один объект. Отношение между объектами определяет отношение между таблицами. Существует четыре типа отношений:

- отношение "один-к-одному" (1:1) означает, что каждая запись в одной таблице соответствует только одной записи в другой таблице;
- отношение "один-ко-многим" (1:M) означает, что каждой записи в одной таблице соответствует одна или несколько записей в другой таблице;
- отношение "многие-к-одному" (M:1) аналогично рассмотренному ранее типу "один-ко-многим". Тип отношения между объектами зависит от вашей точки зрения;
- отношение "многие-ко-многим" (M:M) возникает между двумя таблицами в тех случаях, когда каждой записи в одной таблице соответствует 0, 1, 2 и более записей в другой таблице и наоборот.

В большинстве случаев любые две таблицы связаны отношением "один-ко-многим". Это означает, что любая запись в первой таблице может быть связана с несколькими записями во второй, однако любая запись второй таблицы связана только с одной записью в первой.

Связь один - к - одному (1:1) предполагает, что в каждый момент времени одному объекту А соответствует не более одного информационного объекта В и наоборот. Рис.12 иллюстрирует указанный тип отношения.

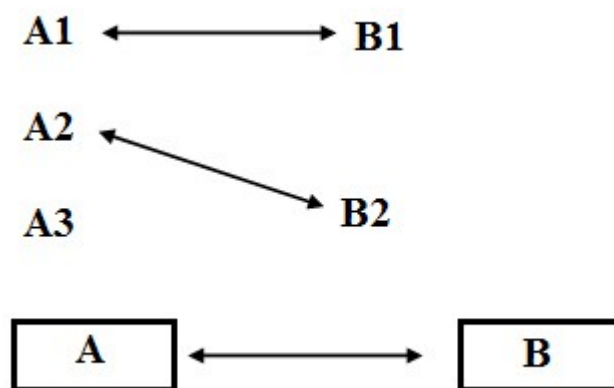


Рис. 12. Отношения 1:1

Примером связи данного типа является пользователь компьютера и его пароль.

При связи один ко многим (1:M) одному информационному объекту А соответствует некоторое множество М объектов В, но каждый экземпляр объекта В связан не более чем с 1 экземпляром объекта А. Графически данное соответствие имеет вид, представленный на [рис. 13](#).

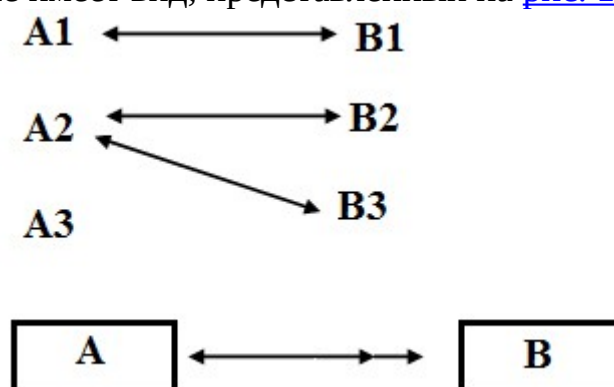


Рис. 13. Отношения 1:M

Связь многие - ко -многим (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует несколько экземпляров объекта В и наоборот. На [рис. 14](#) графически представлено указанное соответствие.

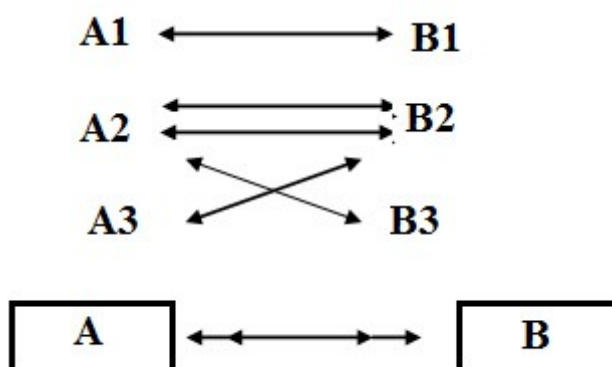


Рис. 14. отношения М:М

Например, один студент обучается у группы преподавателей, а один преподаватель, в свою очередь, может обучать многих студентов.

3.5. Программная реализация базы данных

Программная реализация базы данных осуществляется посредством создания целевой СУБД на языке определения данных (DDL). Команды DDL-языка компилируются и используются для создания схем и пустых файлов базы данных. На этом же этапе определяются и все специфические пользовательские представления.

Прикладные программы реализуются с помощью языков третьего или четвертого поколения. Некоторые элементы этих прикладных программ будут представлять собой транзакции обработки базы данных, записываемые на языке манипулирования данными (DML) целевой СУБД и вызываемые из программ на базовом языке программирования - например, на VisualBasic, C++, Java. Кроме того, на этом этапе создаются другие компоненты проекта приложения - например, экраны меню, формы ввода данных и отчеты. Следует учитывать, что многие существующие СУБД имеют свои собственные инструменты разработки, позволяющие быстро создавать приложения с помощью непроцедурных языков запросов, разнообразных генераторов отчетов, генераторов форм, генераторов графических изображений и генераторов приложений.

На этом этапе также реализуются используемые приложением средства защиты базы данных и поддержки ее целостности. Одни из них описываются с помощью языка DDL, а другие, возможно, потребуется определить иными средствами - например, с помощью дополнительных утилит СУБД или посредством создания прикладных программ, реализующих требуемые функции.

3.5.1. Разработка приложений

Разработка приложений – это проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных. В большинстве случаев проектирование приложений нельзя завершить до окончания проектирования базы данных. С другой стороны, база данных предназначена для поддержки приложений, а потому между фазами проектирования базы данных и проектирования приложений для этой базы данных должен постоянно происходить обмен информацией.

Необходимо убедиться, что все функциональные возможности, предусмотренные в спецификациях требований пользователей, обеспечиваются интерфейсом пользователя соответствующих приложений. Это относится как к проектированию прикладных программ доступа к информации в базе данных, так и к проектированию транзакций, т.е. проектированию методов доступа к базе данных.

Помимо проектирования способов, с помощью которых пользователь сможет получить доступ к необходимым ему функциональным возможностям, следует также разработать соответствующий пользовательский интерфейс приложений базы данных. Этот интерфейс должен предоставлять необходимую пользователю информацию самым удобным для него образом.

3.5.2. Тестирование базы данных

Тестирование - процесс выполнения прикладных программ с целью поиска ошибок. Прежде чем использовать новую систему на практике, ее следует тщательно проверить. Этого можно добиться путем разработки алгоритма тестирования с использованием реальных данных, который должен быть построен таким образом, чтобы весь процесс тестирования выполнялся строго последовательно и методически правильно. Задачей тестирования не является процесс демонстрации отсутствия ошибок, оно вряд ли сможет продемонстрировать отсутствие ошибок в программном обеспечении - скорее, наоборот, оно способно лишь показать их наличие. Если тестирование проведено успешно, то обязательно вскроются имеющиеся в прикладных программах и структурах базы данных ошибки. В качестве побочного результата тестирование может лишь показать, что база данных и прикладные программы работают в соответствии с их спецификациями и удовлетворяют при этом существующим требованиям, предъявляемым к производительности. Кроме того, сбор статистических данных на стадии тестирования позволяет установить показатели надежности и качества созданного программного обеспечения.

Как и при проектировании баз данных, пользователи новой системы должны быть вовлечены в процесс ее тестирования. В идеале, тестирование системы должно проводиться на отдельном комплекте оборудования, но зачастую это просто невозможно. При использовании реальных данных важно предварительно создать их резервные копии на случай их повреждения в результате ошибок. По завершении тестирования процесс создания прикладной системы считается законченным, и она может быть передана в промышленную эксплуатацию.

3.6. Эксплуатация и сопровождение базы данных

Под эксплуатацией и сопровождением понимают поддержку нормального функционирования БД. На предыдущих этапах приложение базы данных было полностью реализовано и протестировано. Теперь система входит в последний этап своего жизненного цикла, называемый эксплуатацией и сопровождением. Он включает выполнение таких действий, как:

- контроль производительности системы. Если производительность падает ниже приемлемого уровня, то может потребоваться дополнительная реорганизация базы данных;
- сопровождение и модернизация (в случае необходимости) приложений баз данных. Новые требования включаются в приложение базы данных при повторном выполнении предыдущих этапов жизненного цикла.

Как только база данных будет введена в эксплуатацию, следует постоянно контролировать процесс ее функционирования. Это позволит убедиться, что производительность и другие показатели соответствуют предъявляемым требованиям. Типичная СУБД обычно предоставляет различные утилиты администрирования базы данных, включая утилиты загрузки данных и контроля за функционированием системы. Подобные утилиты способны отслеживать работу системы и предоставлять информацию о различных показателях, таких как уровень использования базы данных, эффективность системы блокировок (включая сведения о количестве имевших место взаимных блокировок), а также выбираемые стратегии выполнения запросов. Администратор базы данных может использовать эту информацию для настройки системы с целью повышения ее производительности (например, за счет создания дополнительных индексов), ускорения выполнения запросов, изменения структур хранения, объединения или разбиения отдельных таблиц.

Процесс мониторинга должен поддерживаться на протяжении всего процесса эксплуатации приложений, что позволит в любой момент времени провести эффективную реорганизацию базы данных с целью удовлетворения изменяющихся требований. Подобные изменения предоставляют информацию о наиболее вероятном совершенствовании БД и ресурсах, которые могут потребоваться в будущем. Если в используемой СУБД нет некоторых нужных утилит, то администратору придется либо разработать их самостоятельно, либо приобрести требуемые дополнительные инструменты у сторонних разработчиков.

4. ОРГАНИЗАЦИЯ И ПРИМЕНЕНИЕ СУБД ACCESS

4.1. Основные понятия СУБД Access

Система MS Access является системой управления БД, использует реляционную модель данных и входит в состав пакета прикладных программ MSOffice. Она предназначена для хранения, ввода, поиска и редактирования данных, а также вывода их в удобном виде.

К областям применения Access можно отнести следующие:

- в малом бизнесе (бухгалтерский учет, ввод заказов, ведение информации о клиентах, ведение информации о деловых контактах);
- в крупных корпорациях (приложения для рабочих групп, системы обработки информации);

- в качестве персональной СУБД (справочник по адресам, ведение инвестиционного портфеля, поваренная книга, каталоги книг, пластинок, видеофильмов и т. п.).

Access является одной из самых мощных, удобных и простых систем управления базами данных. Поскольку Access входит в состав Microsoft Office, она обладает многими чертами, характерными для приложений данного пакета и может обмениваться с ними информацией. Например, работая в Access, можно открывать и редактировать файлы, а также использовать буфер обмена для копирования данных из других приложений.

Средствами разработки объектов в Access являются «мастера» и «конструкторы». Это специальные программы, которые служат для создания и редактирования таблиц, запросов, различных типов форм и отчетов. Как правило, «мастер» используется для создания, а «конструктор» - для редактирования объектов. Процесс редактирования предполагает изменение вида некоторого объекта с целью его улучшения. При редактировании формы можно изменить названия и порядок расположения полей, увеличить или уменьшить размер области ввода данных, и т.д. Можно использовать «конструктор» и для создания форм, но это очень трудоемкая работа. В Access включены специальные программные средства, помогающие производить анализ структуры данных, импортировать электронные таблицы и текстовые данные, повышать быстродействие приложений, создавать и настраивать приложения с использованием встроенных шаблонов. Чтобы полностью автоматизировать работу приложений, можно использовать макросы для связывания данных с формами и отчетами.

В Access реализовано управление реляционными базами данных. Система поддерживает первичные и внешние ключи, обеспечивает целостность данных на уровне ядра, что не разрешает несовместимые операции обновления или удаления данных. Таблицы в Access снабжены средствами проверки допустимости данных, т.е. не разрешается некорректный ввод. Каждое поле таблицы имеет свой формат и стандартные описания, что облегчает ввод данных. Access поддерживает следующие типы полей, в том числе: вкладка, текстовый, числовой, счетчик, денежный, дата/время, МЕМО, логический, гиперссылка, поля объектов OLE, вложение и вычисляемый. Если в полях не оказывается никаких значений, система обеспечивает полную поддержку пустых значений.

В Access можно использовать графические средства, как и в MSWord, Excel, PowerPoint и других приложениях, позволяющие создавать различные виды графиков и диаграмм. Можно создавать гистограммы, двухмерные и трехмерные диаграммы. В формы и отчеты Access можно добавлять всевозможные объекты: рисунки, диаграммы, аудио- и видеоклипы. Связывая эти объекты с разработанной базой данных, можно создавать динамические формы и отчеты. Также в Access можно использовать макросы, позволяющие автоматизировать выполнение некоторых задач. Они позволяют открывать и закрывать формы и отчеты, создавать меню и

диалоговые окна с целью автоматизации создания различных прикладных задач.

В Access можно получить контекстно-зависимую справку, для получения которой надо нажать <F1>, и на экране появится справочная информация по тому вопросу, который интересует пользователя в текущий момент. При этом можно легко перейти к оглавлению справочной системы, конкретной информации, журналу предыдущих обращений и закладкам. Информация базы данных хранится в файле с расширением accdb.

4.2. Объекты СУБД Access

При запуске СУБД Access появляется окно для создания новой базы данных или для работы с ранее созданными БД, или уже имеющимися шаблонами (рис.15). Шаблоны представляют собой пустые структуры баз данных, в которых определены типы полей, созданы основные объекты, осуществлена связь между таблицами и т.п.

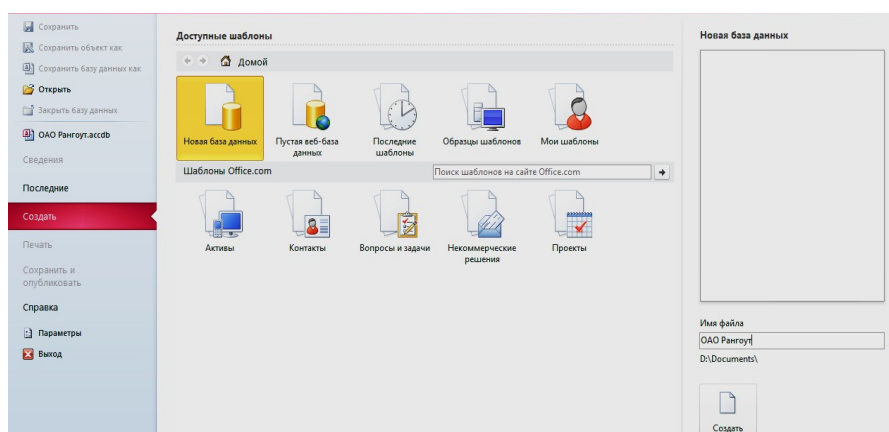


Рис. 15. Запуск Access

При создании новой базы данных Access откроет пустую таблицу, содержащую одну строку и два столбца (рис 16).

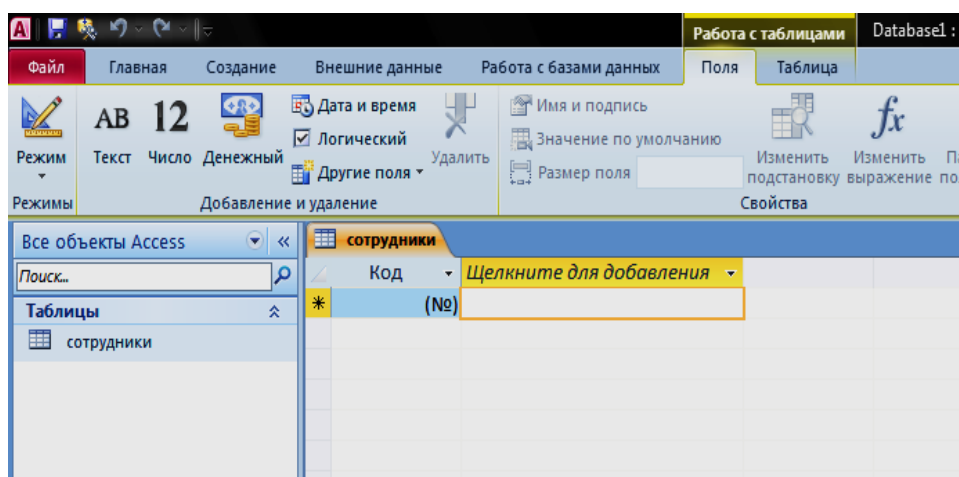


Рис.16. Окно создания новой базы данных

В левой части окна (область переходов) показаны все созданные объекты БД, пока мы лишь видим, пустую таблицу, так как созданных объектов в новой базе данных больше нет (рис.16). К основным объектам СУБД Access относятся следующие.

Таблицы. Таблицы являются основными объектами баз данных, так как в них хранятся все данные, и они определяют структуру базы данных. База данных может содержать тысячи таблиц, размеры которых ограничиваются только доступным пространством на жестком диске компьютера. Количество записей в таблицах определяется объемом жесткого диска, а количество полей не более 255.

Таблицы в Access могут быть созданы следующим образом:

- в режиме «конструктора»;
- в режиме ввода данных в таблицу.

Создать таблицу можно путем импорта данных, хранящихся в другом месте, или создания связи с ними. Это можно сделать, например, с данными, хранящимися в файле Excel, в списке Windows SharePointServices, XML-файле, другой базе данных MS ACCESS. Список SharePoint позволяет предоставить доступ к данным пользователям, у которых не установлено приложение MS ACCESS. При импорте данных создается их копия в новой таблице текущей базы данных. Последующие изменения, вносимые в исходные данные, не будут влиять на импортированные данные, и наоборот. Если осуществляется связывание с данными, в текущей базе данных создается связанная таблица, обеспечивающая динамическое подключение к данным, хранящимся в другом месте. Изменения данных в связанной таблице отражаются в источнике, а изменения в источнике - в связанной таблице.

В режиме таблицы отображаются данные, которые хранятся в таблице, а в режиме «конструктора» отображается структура таблицы. Если таблицы имеют общие поля, можно воспользоваться подчиненной таблицей, чтобы вставить в одну таблицу записи из другой. Такой подход позволяет одновременно просматривать данные из нескольких таблиц.

Запросы. Запросы - это специальные средства, предназначенные для поиска и анализа информации в таблицах базы данных, отвечающей определенным критериям. Найденные записи, называемые результатами запроса, можно просматривать, редактировать и анализировать различными способами. Кроме того, результаты запроса могут использоваться в качестве основы для создания других объектов Access. Существуют различные типы запросов, наиболее распространенными из которых являются запросы на выборку, параметрические и перекрестные запросы, запросы на удаление записи, изменение и другие. Реже используются запросы на действие и запросы SQL (StructuredQueryLanguage). Если нужного запроса нет, то его можно создать дополнительно.

Запросы формируются различными способами, например, с помощью «мастера», также можно создать запрос вручную в режиме «конструктора». Простейшим и наиболее часто используемым видом запросов является

запрос на выборку. Эти запросы выбирают данные из одной или нескольких таблиц и формируют из них новую таблицу, записи в которой можно изменять. Запросы на выборку нужны для вычисления сумм, средних значений и нахождения других итоговых значений. Таким образом, запросы используют данные из основных таблиц и создают временные таблицы.

Формы. Формы используются для ввода и редактирования записей в таблицах базы данных. Формы можно отображать в трех режимах: в режиме, предназначенном для ввода данных, в режиме таблицы, где данные представлены в табличном формате, и в режимах «макета» и «конструктора», позволяющих вносить изменения и дополнения в формы.

Основными элементами формы являются надписи, в которых указан текст, непосредственно отображающийся в форме, и поля, содержащие значения полей таблицы. Хотя режим «конструктора» позволяет создать форму с нуля, обычно он используется для доработки и совершенствования форм, созданных с помощью «мастера». Помимо вышеперечисленных средств формы также можно создавать с помощью следующих инструментов:

- «форма»;
- «разделенная форма»;
- «несколько элементов»;
- «пустая форма».

Наиболее эффективно использовать формы для ввода данных в виде специальных бланков, так как форма может иметь вид бланка. Применение форм позволяет вводить данные в удобном для пользователя виде привычных документов. Формы ввода-вывода позволяют вводить данные в базу, просматривать их, изменять значения полей, добавлять и удалять записи. Форма может содержать кнопку, используемую для печати отчета, открытия других объектов или автоматического выполнения других задач.

Отчеты. Отчеты используются для отображения информации в таблицах в отформатированном виде, который наглядно представляется как на экране монитора, так и на бумаге. Отчет является эффективным средством для вывода данных на печать из базы данных в форме, требуемой для пользователя (в виде справок, экзаменационных ведомостей, таблиц и т.д.). Помимо данных, извлеченных из нескольких таблиц и запросов, отчеты могут включать элементы оформления, свойственные печатным документам, как, например, названия, заголовки и колонтитулы.

Отчет можно отобразить в четырех режимах:

- в режиме «конструктора», позволяющем изменить внешний вид отчета;
- в режиме просмотра образца, в котором можно отобразить все элементы готового отчета, но в сокращенном виде;
- в режиме «макета», позволяющем более наглядно отображать (по сравнению с режимом конструктора) и форматировать отчет;
- в режиме предварительного просмотра, где отчет отображается в том виде, в каком будет напечатан.

Таблицы, запросы, формы и отчеты представляют собой объекты, которые наиболее широко используются при разработке баз данных Access. Однако возможности базы данных можно существенно расширить, если воспользоваться страницами доступа, макросами и модулями.

Страницы. Чтобы предоставить пользователям Интернета доступ к информации, в базе данных можно создать специальные страницы доступа к данным. С помощью страниц доступа к данным можно просматривать, добавлять, изменять и обрабатывать данные, хранящиеся в базе данных. Страницы доступа к данным могут также содержать данные из других источников, например, из Excel. Для публикации информации из базы данных в Web Access используют «мастер», который обеспечивает создание страницы доступа.

Макросы. Макросы представляют собой небольшие программы из одной или более макрокоманд, выполняющих определенные операции, с помощью которых обеспечивается, например, открытие формы, печать отчетов, щелчок кнопки и т.п. Это особенно удобно, если предполагается передать базу данных неквалифицированным пользователям. Например, можно написать макросы, содержащие последовательность команд, выполняющих рутинные задачи, или связать такие действия, как открытие формы или печать отчета, с кнопками кнопочной формы.

Модули. Модуль - объект базы данных, который позволяет создавать библиотеки подпрограмм и функций, используемых во всем приложении. Используя коды модулей можно решать такие задачи, как обработка ошибок ввода, объявление и применение переменных, организация циклов и т.п.

4.3. Создание объектов СУБД Access

4.3.1. Создание таблиц

При вводе данных в Access полям присваиваются имена: Поле1, Поле2 и так далее. Можно использовать предложенные имена или изменить их. Название полей в таблице можно задавать двумя способами. После выбора способа создания таблицы выполняется команда «**Создать**» и вызывается соответствующее окно. На рис.17. показано создание таблицы в режиме «конструктора». Создаются требуемые поля таблицы с заданным типом данных, который выбирается посредством кнопки выбора – «галочка», в нижней части окна находится раздел выбора свойств поля, которые предлагаются вначале по умолчанию.

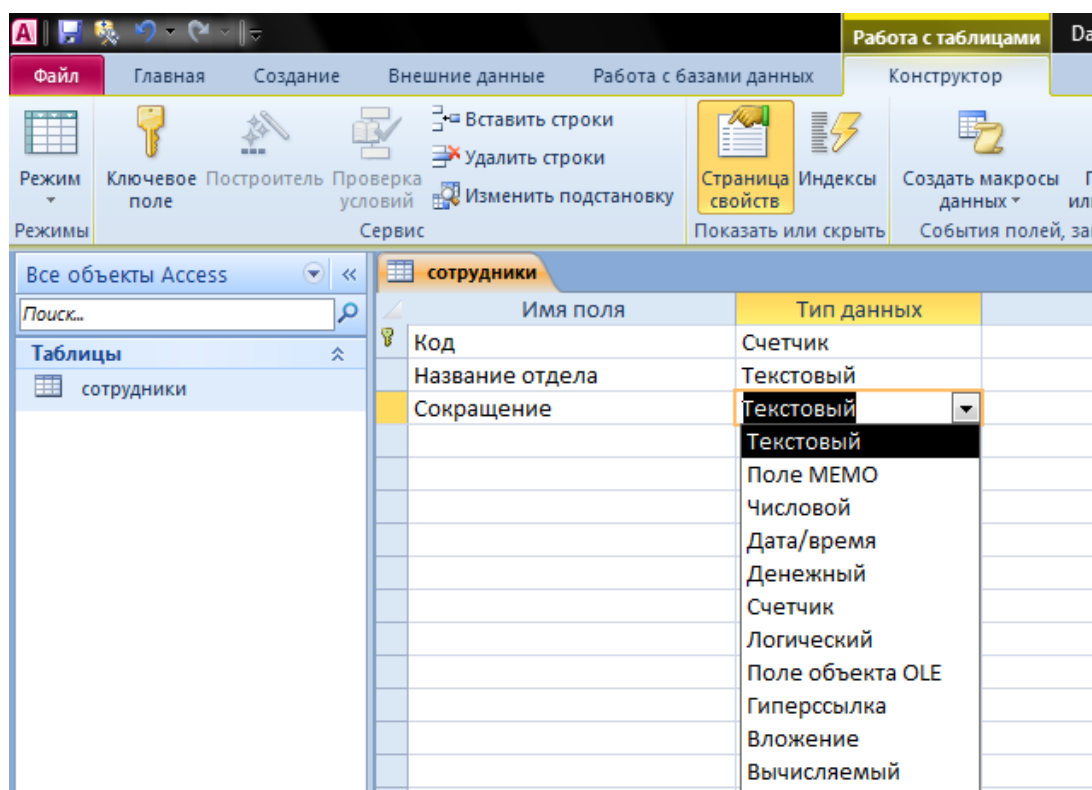


Рис. 17.Создание таблицы в режиме конструктора

Свойства полей таблицы базы данных Access указаны в нижней половине таблицы (рис.17).

Можно создать таблицу в режиме «конструктора», изменяя, добавляя или удаляя поля таблицы. Для введения нового поля в верхней части окна таблицы указывается имя поля и определяется его тип. Для переименования поля, надо изменить его имя в столбце «Имя поля».

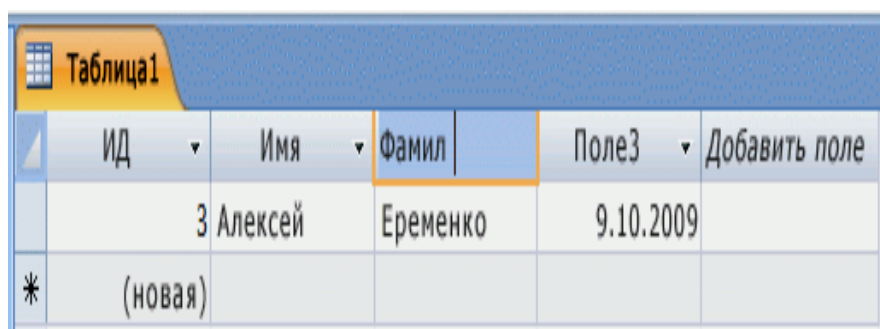
При создании таблиц используются следующие основные типы данных (рис.18).

Тип данных	Применение	Размер
Текстовый	Используется для буквенно-цифровых символов, включая текст, а также текст и числа, не применяемые в вычислениях.	До 255 символов
Числовой	Применяется для хранения числовых значений (целых или дробных), которые используются в вычислениях (за исключением денежных сумм).	1, 2, 4, 8, 12 байтов
Счетчик	Используется для формирования уникальных значений, которые могут применяться в качестве первичного ключа.	4 байта
Логический	Используется для логических значений: Да/Нет, Истина/Ложь	1 бит (0,125 байта)

Тип данных	Применение	Размер
	или Выкл./Выкл.	

Рис.18. Типы данных Access

После создания таблицы ее сохраняют, присвоив имя, и, таким образом, она становится частью базы данных для дальнейшей работы с ней. При необходимости можно затем добавлять новые поля, выполнять изменение типов данных и т.п. Для заполнения таблицы данными ее открывают (она имеет только названия полей) и заполняют стандартным образом, например, как в электронных таблицах (рис.19).



ИД	Имя	Фамил	Поле3	Добавить поле
3	Алексей	Еременко	9.10.2009	
*	(новая)			

Рис.19. Ввод данных в таблицу

Определение связей между таблицами. Перед тем как приступить к созданию базы данных, необходимо разработать схему данных, т.е. организовать связи между таблицами. Схема данных – это структура основных таблиц, в Access используется два типа связей между таблицами:

- «один» - ко - «многим»;
- «один» - к - «одному»;
- много - ко - «многим».

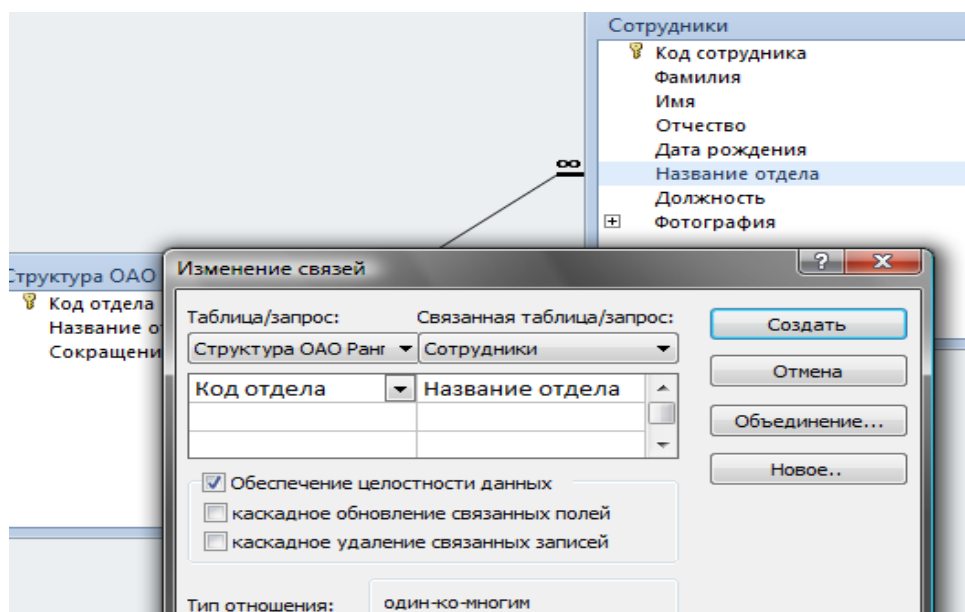
Один – к – одному: каждой записи первой сущности соответствует только одна запись второй сущности, а каждой записи второй сущности – только одна запись первой сущности. Пример - автор, у которого в данный момент имеется лишь одна незавершенная книга.

Один – ко многим: каждой записи первой сущности могут соответствовать несколько записей второй сущности, однако каждой записи второй сущности соответствует только одна запись первой сущности. Пример - издательство, выпустившее несколько книг.

Много – ко – многим: каждой записи первой сущности могут соответствовать несколько записей второй сущности, а каждой записи второй сущности соответствуют несколько записей первой сущности. Пример - один автор может написать несколько книг, а у одной книги может быть несколько авторов. Такие отношения не реализуются в реляционной модели, они должны быть преобразованы через отношения один – ко – многим.

Связь между двумя таблицами определяется наличием общего поля, причем в одной таблице оно должно быть первичным ключом, т.е. на стороне «один» - первичный ключ, на стороне «многим» - вторичный ключ. При создании связи между таблицами связываемые поля могут иметь разные имена, однако они должны иметь одинаковый тип. Если в базе данных не определены еще связи, то, после нажатия кнопки **«Схема данных»** на панели инструментов, автоматически открывается диалоговое окно **«Добавление таблицы»**. Чтобы задать связи между полями, надо выбрать поле в одной таблице и перетащить его на соответствующее поле во второй таблице. Обычно связывают поле первичного ключа (выделенное в списке полей полужирным шрифтом) одной таблицы с соответствующим ему полем (иногда имеющим то же имя) второй таблицы, называемое полем внешнего ключа. Внешний ключ – это одно или несколько полей, содержащих ссылку на поле или поля первичного ключа в другой таблице. Внешний ключ определяет способ объединения таблиц.

Далее необходимо настроить свойства связи. Для этого мышью выделяют в схеме данных нужную связь, открывают контекстное меню связи, а далее выбирают команду **«Изменить связь»**. В открывшемся окне устанавливают флажок **«Обеспечение целостности данных»** (тогда удалять в ключевом поле данные нельзя). Чтобы преодолеть ограничения на удаление или изменение связанных записей, сохраняя при этом целостность данных, следует установить флажки **«Каскадное обновление связанных полей»** и **«Каскадное удаление связанных полей»**. Если установлен флажок **«Каскадное обновление связанных полей»**, то при изменении ключевого поля главной таблицы автоматически изменяются и соответствующие значения связанных записей. Если установлен флажок **«Каскадное удаление связанных полей»**, то при удалении записи в главной таблице удаляются и все связанные записи в подчиненной таблице». На рис. 20 показана схема данных со связью «один» – ко – «многим».



4.3.2. Создание запросов

В режиме таблицы возможны самые различные операции с данными: просмотр, сортировка, фильтрация, редактирование, печать. При необходимости выполнения обработки или просмотра данных из группы таблиц можно выбрать требуемые данные с помощью запросов.

Запросы могут являться источником записей для форм и отчетов. После выполнения запроса на выборку, Access формирует набор записей, содержащих отобранные данные. Как правило, с набором записей можно работать также, как и с таблицей: выполнять просмотр и выборку данных, обновлять данные и выводить на печать. Этот набор записей физически не существует, он создается только во время выполнения запроса. Если изменить данные в наборе записей, Access внесет соответствующие изменения в таблицы, на базе которых построен запрос. В Access имеется несколько основных типов запросов.

Запросы на выборку. Этот запрос используется наиболее часто. Результатом этого запроса являются данные из одной или нескольких таблиц в виде таблицы, запись в которой можно обновлять. Эти данные можно также использовать для выполнения каких-либо действий, например, для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений.

Перекрестные запросы. Запросы, которые используются для расчетов и представления данных в структуре, облегчающей их анализ. Перекрестный запрос подсчитывает сумму, среднее, число значений или выполняет другие статистические расчеты, после чего результаты группируются в виде таблицы по двум наборам данных, один из которых определяет заголовки столбцов, а другой заголовки строк.

Запросы на изменение. Это запрос, который за одну операцию меняет или перемещает несколько записей. Существует четыре типа запросов на изменение:

- На удаление записи - удаляет группу записей из одной или нескольких таблиц. С помощью запроса на удаление можно удалять только всю запись, а не отдельные поля внутри нее.
- На обновление записи - вносит общие изменения в группу записей одной или нескольких таблиц. Запрос на обновление записей позволяет изменять данные в существующих таблицах.
- На добавление записей - добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц.
- На создание таблицы - создает новую таблицу на основе всех или части данных из одной или нескольких таблиц. Запрос на создание таблицы полезен при создании таблицы для экспорта в другие базы данных Access или при создании архивной таблицы, содержащей старые записи.

- На объединение – позволяет объединить данные из двух таблиц с аналогичными структурами.

Запрос на создание таблицы. Этот запрос создает новую таблицу из данных одной или нескольких таблиц. Он нужен при создании таблицы для экспорта в другие базы данных Access.

Запросы с параметрами. Это запросы, при выполнении отображающие в собственном диалоговом окне приглашение ввести данные, например условие для возвращения записей или значение, которое требуется вставить в поле. Можно разработать запрос, выводящий приглашение на ввод нескольких единиц данных, например двух временных промежутков. Затем MSAccess может выбрать все записи, приходящиеся на интервал времени между этими временными интервалами. Запросы с параметрами также удобно использовать в качестве основы для форм, отчетов и страниц доступа к данным.

Запросы SQL. Это запрос, создаваемый при помощи команд языка SQL (StructuredQueryLanguage), и используемый для обновления и управления реляционными базами данных. Когда пользователь создает запрос в режиме «конструктора», автоматически создается эквивалентная конструкция SQL.

Выбор данных из таблиц. Одним из преимуществ запросов является то, что они позволяют достаточно быстро отобрать необходимые данные из нескольких связанных таблиц, но запросы полезны и при работе с одной таблицей. Запрос можно создавать с помощью «конструктора» или с помощью «мастера» запросов. Все приемы, используемые при работе с одной таблицей, подходят и для сложных многотабличных запросов, поэтому рассмотрим формирование запроса на выборку данных из двух таблиц.

1. На вкладке Создание в группе Другие кликните по кнопке Мастер запросов.

2. Запустите режим создания запроса с помощью мастера (Простой запрос).

3. Создайте запрос, следуя шагам мастера (рис.21):

3.1. Выбор из списка Таблицы и запросы таблицу Структура ОАО РАНГОУТ.

3.2. Перенесите в список выбранных полей (справа) поле Сокращение.

3.3. Выберите из списка Таблицы и Запросы таблицу Сотрудники.

3.4. Перенесите в список выбранных полей (справа) поле Фамилия (рис.21).

3.5. Перейдите на следующий шаг, нажав кнопку Далее.

4. Введите имя запроса. Для этого:

4.1. Выберите тип запроса Подробный.

4.2. Введите имя запроса Исполнитель.

4.3. Завершите работу мастера, нажав кнопку Готово.

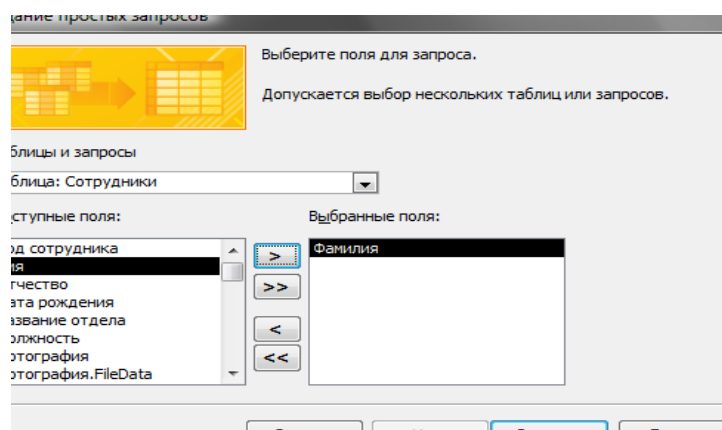


Рис. 21. Создание простых запросов

5. Просмотрите результаты запроса в режиме Таблицы.

6. Выберите режим Конструктора. Откроется бланк запроса, он состоит из двух областей: в верхней части бланка отображается структура таблицы или нескольких таблиц, к которым адресован запрос, в нижней части – пустая таблица будущего запроса, куда необходимо будет перенести название полей. Достаточно щелкнуть на нужном поле таблицы базы данных и в таблице-запросе появится это поле. Сформировав запрос, его закрывают, дают имя и в дальнейшем его запускают из окна базы данных.

Данные в запросе можно упорядочивать по возрастанию или убыванию по выбранному полю. Если необходимо сделать выбор по нескольким полям одновременно, то сортировка будет происходить по столбцам слева направо.

По умолчанию все поля запроса выводятся на экран, но в бланке запроса можно сбросить флажок «Вывод на экран» и столбец на экране будет не виден (рис. 22).

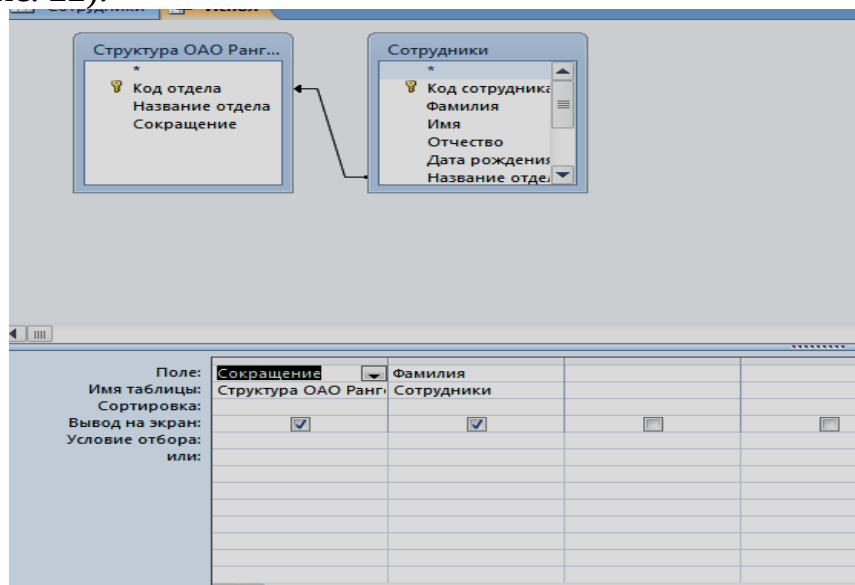


Рис.22. Бланк запроса

7. Установите в обоих полях сортировку по возрастанию. При наличии сортировки в нескольких полях программа выполняет сначала первую, а

потом вторую и т.д. Таким образом, запрос предоставляет возможность задать сортировку по нескольким полям.

8. Перейдите в режим просмотра Таблицы. Просмотрите, как изменилось расположение данных и не забудьте сохранить изменения.

Рассмотрим пример простейшего запроса. Пусть имеется база данных, состоящая из двух таблиц – «Структура ОАО» и «Сотрудники», таблицы связаны между собой, как показано на рис. 20. Эта связь означает, что в одном отделе может числиться много сотрудников, но ни один сотрудник не может числиться сразу в нескольких отделах. Необходимо создать запрос «**Исполнитель**», в котором по данным таблицы Сотрудники будут получены новые данные со следующими назначениями (рис.23).

Имя поля запроса	Назначение
Код Сотрудника	Устанавливает связь получаемых в других полях данных с конкретным сотрудником по ключевому полю
ФИО	Содержит фамилию, имя и отчество как одну строку
Возраст	Вычисляет количество полных лет по дате рождения
Месяц	Определяет номер месяца рождения по дате
День	Определяет порядковый день месяца рождения по дате

Рис.23. Назначение полей запроса

Порядок действий следующий:

1. На вкладке «**Создание**» в группе «**Запросы**» кликните по кнопке «**Мастер запросов**».
2. Запустите режим создания запроса с помощью «**Мастера**» простой запрос (рис. 24).
3. Создайте запрос, следуя шагам «**Мастера**»:
 - 3.1. Выбор из списка «**Таблицы и запросы**» таблицу «**Структура ОАО**».
 - 3.2. Перенесите в список выбранных полей (справа) поле «**Сокращение**».
 - 3.3. Выберите из списка «**Таблицы и Запросы**» таблицу «**Сотрудники**».
 - 3.4. Перенесите в список выбранных полей (справа) поле «**Фамилия**».
 - 3.5. Перейдите на следующий шаг, нажав кнопку «**Далее**».
4. Введите имя запроса. Для этого:

- 4.1. Выберите тип запроса **«Подробный»**.
- 4.2. Введите имя запроса **«Исполнитель»**.
- 4.3. Завершите работу мастера, нажав кнопку **«Готово»**.

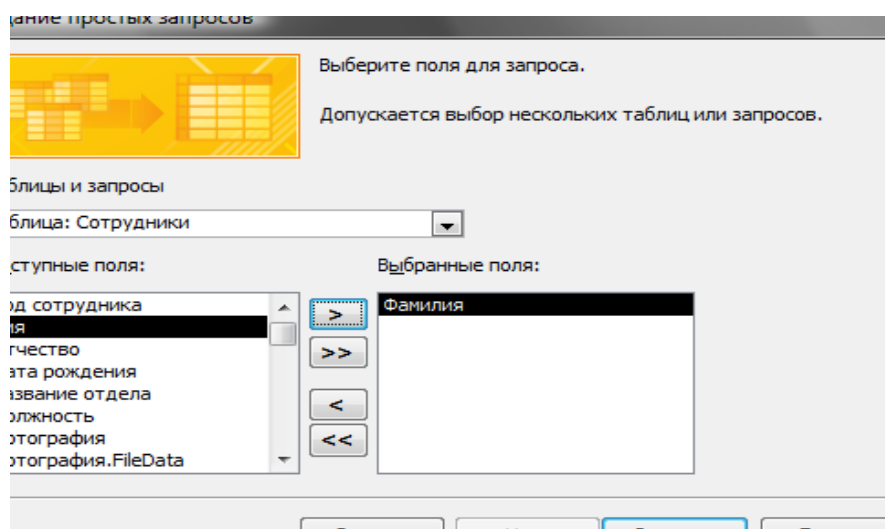


Рис. 24. Создание простых запросов

5. Просмотрите результаты запроса в режиме **«Таблицы»**.
6. Выберите режим **«Конструктора»**.

Откроется бланк запроса, в верхней части бланка расположена схема связи таблиц, используемых в запросе. В нижней части расположена таблица описания полей запроса. В первой строке перечислены поля запроса. Во второй строке указано имя таблицы, из которой взято поле. В третьей строке можно задать сортировку полей (рис. 25). В бланке для каждого поля можно задать индивидуальные условия отбора с параметрами. В строке «условия отбора» можно задавать условие, например, >20 , >50 или <120 , в диапазоне и т.п. В итоге получите только отобранные записи по одному из этих условий.

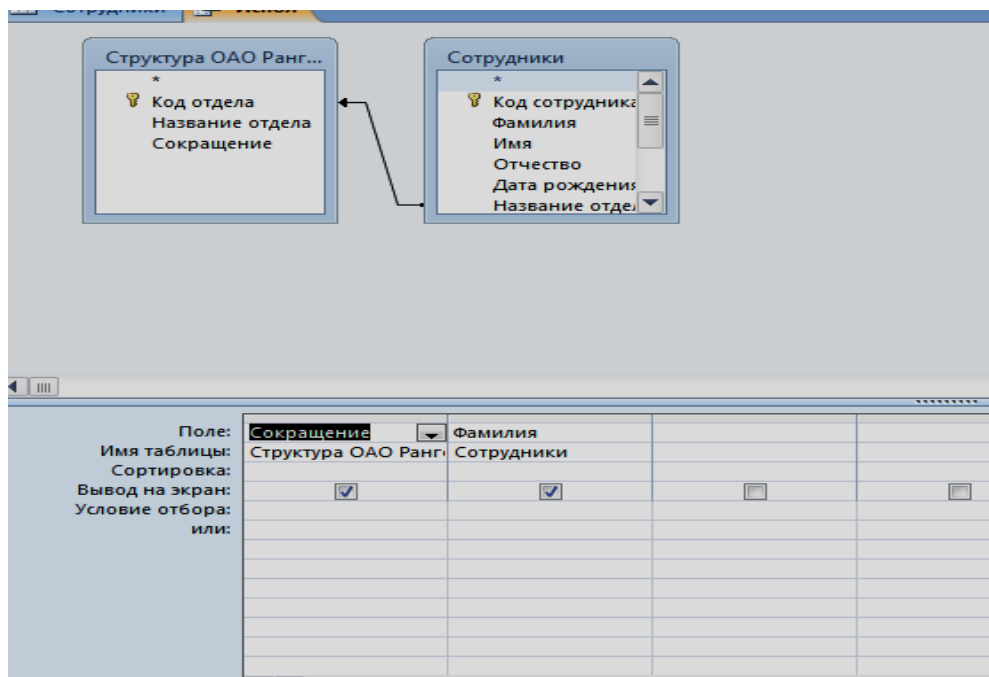


Рис. 25. Запрос на выборку

7. Для создания вычисляемых полей используется «Построитель выражений». На рис. 26 представлен пример его использования для получения поля ФИО, содержащее фамилию, имя и отчество как одну строку.

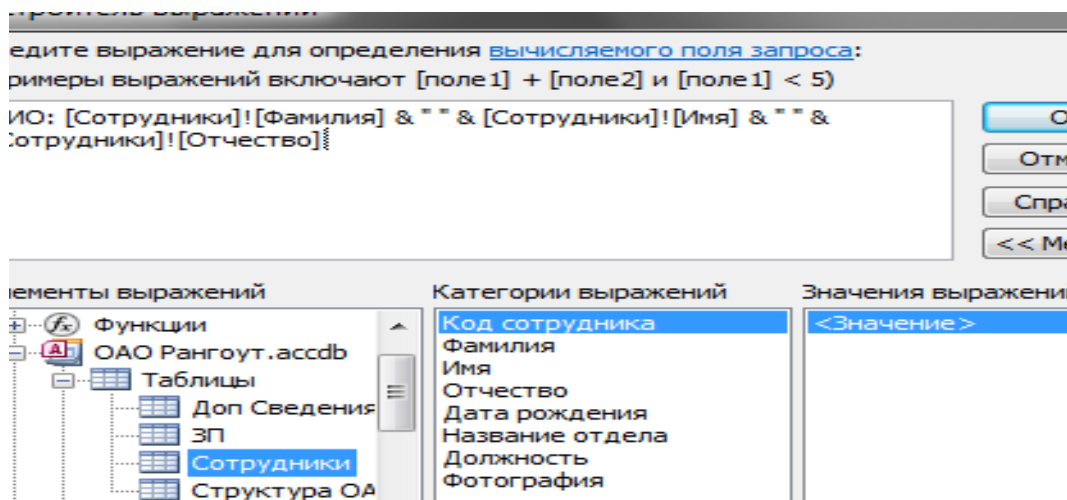


Рис.26. Построитель выражений

8. Поле *Возраст* постройте выражение для поля (рис. 23), в котором производится вычисление количества полных лет по дате рождения:

Возраст: Year(Now()) - Year([Сотрудники]![Дата рождения]).

Примечание. Эта формула содержит встроенные функции Year(), которая вычисляет год по дате, и Now(), которая вычисляет текущую дату. Встроенные функции можно найти, открыв в построителе выражений в окне обзора папки Функции папку Встроенные функции. Возраст получается как разность между годом, отсчитанным от текущей даты, и годом, отсчитанным от даты рождения.

Сотрудники					Сотрудники
			по возрастанию	по возрастанию	
					

Рис. 27. Бланк запроса

9. В следующем столбце постройте выражение для поля Месяц, в котором производится вычисление по дате рождения порядкового номера месяца. В формуле используется встроенная функция Month():

Месяц: *Month([Сотрудники]![Дата Рождения]).*

10. В следующем столбце постройте выражение для поля День, в котором производится вычисление по дате рождения порядкового дня месяца. В формуле используется встроенная функция Day():

День: *Day([Сотрудники]![Дата Рождения]).*

11. Задайте в бланке запроса сортировку по полю Месяц, затем по полю День.

12. Включите в бланк запроса поле Дата рождения из таблицы Сотрудники (рис.27).

13. Перейдите в режим таблицы и просмотрите результаты работы запроса.

Созданные в запросе поля «Возраст, Месяц и День» позволяют провести фильтрацию сотрудников по месяцам и дням даты рождения и в результате составить список, в котором сотрудники будут указаны в порядке дат рождения от начала года.

4.3.3. Создание форм

Access предоставляет пользователю наглядный интерфейс для работы с данными с помощью форм, под которыми понимаются настраиваемые диалоговые окна, сохраняемые в базе данных в виде объектов специального типа. Формы являются такими же объектами базы данных Access, как таблицы, запросы и используются для ввода и отображения данных. Формы предоставляют удобный способ просмотра и редактирования данных в таблицах, существенно упрощают процесс ввода данных и содержат элементы управления для эффективного доступа к данным в таблицах. Под элементами управления понимаются текстовые поля для ввода и правки данных, кнопки, флажки ит.п.

Формы Access предоставляют функциональные возможности для выполнения многих задач, которые нельзя выполнить другими средствами. Формы позволяют проверять правильность ввода данных, проводить вычисления, и обеспечивают доступ к данным в связанных таблицах с помощью подчиненных форм.

Access позволяет создавать формы несколькими способами, самым простым из которых является использование средств автоматического создания форм на основе таблицы или запроса. Автоматически создаваемые формы бывают нескольких видов, каждый из которых отличается способом отображения данных. Форма для ввода и модификации данных:

1. Форма с одним элементом.
 - 1.1. Разделенная форма.
 - 1.2. Форма для нескольких элементов (ленточная форма).
 - 1.3. Составная форма (главная и подчиненная).
2. Сводная таблица (сводная диаграмма).
3. Форма навигации.
4. Пользовательское окно диалога.

Форму можно создавать с помощью множества средств, находящихся на вкладке **Создание** в группе **Формы** (в зависимости от типа и сложности создаваемой формы) (рис.28).

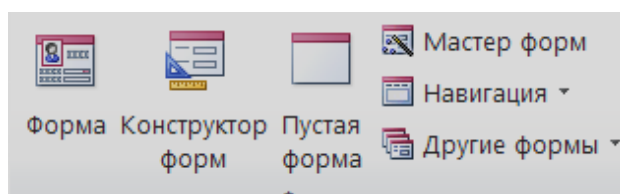


Рис.28. Средства для создания форм

Форма с одним элементом. При работе с данным видом формы одновременно выводится только одна запись.

Разделенная форма. Данный вид формы позволяет одновременно отображать данные в двух представлениях — в режиме формы и в режиме таблицы (рис.29).

Код сотрудн	Фамилия	Имя	Отчество	Фотография	Дата рождени	Название отд	Должность
1	Соколов	Александр	Васильевич		07.09.1975	Технический отдел	Замдиректора

Ид сотру	Фамилия	Имя	Отчество	Э	Дата рождени	Название отдела	Должность
1	Соколов	Александр	Васильевич	Э(2)	07.09.1975	Технический отдел	Замдиректора
2	Белых	Алексей	Иванович	Э(1)	23.03.1965	Отдел снабжения	Специалист
3	Мухин	Антон	Павлович	Э(1)	24.05.1961	Технический отдел	Консультант
4	Ильина	Анна	Васильевна	Э(1)	15.05.1956	Администрация	Секретарь
5	Плужников	Дмитрий	Андреевич	Э(1)	15.05.1956	Дирекция	Замдиректора

Рис.29. Разделенная форма

Ленточная форма. Представляет собой форму для нескольких элементов и позволяет отображать сведения более чем об одной записи одновременно.

Составная форма (главная и подчиненная). Используется для просмотра в одной форме данных из нескольких таблиц со связью «Один-ко-

многим». Сводная таблица/сводная диаграмма. Сводная таблица и сводная диаграмма дают возможность выполнять комплексный анализ данных (по аналогии с MS Excel).

Окно диалоговой формы. Окна диалоговой формы, как правило, используются для сбора информации, предоставляемой пользователем. Они всегда модальные - т.е. пользователь не может перейти к выполнению других этапов работы с приложением, пока не закончит обработку формы. Окна разрабатываются чаще всего с использованием средств программирования.

Рассмотрим в качестве примера создание формы «Сотрудники» с помощью «Мастера форм», который позволяет создавать формы на базе одной или нескольких таблиц. Источником данных для создаваемой формы является таблица «Сотрудники» и таблица «Личные данные». Для создания данной формы необходимо выполнить следующее:

1. На вкладке «Создание» в группе команд «**Формы**» выберете элемент управления «**Другие формы**» – «**Мастер форм**».
2. Запустите режим создания формы с помощью «мастера» (рис. 30).

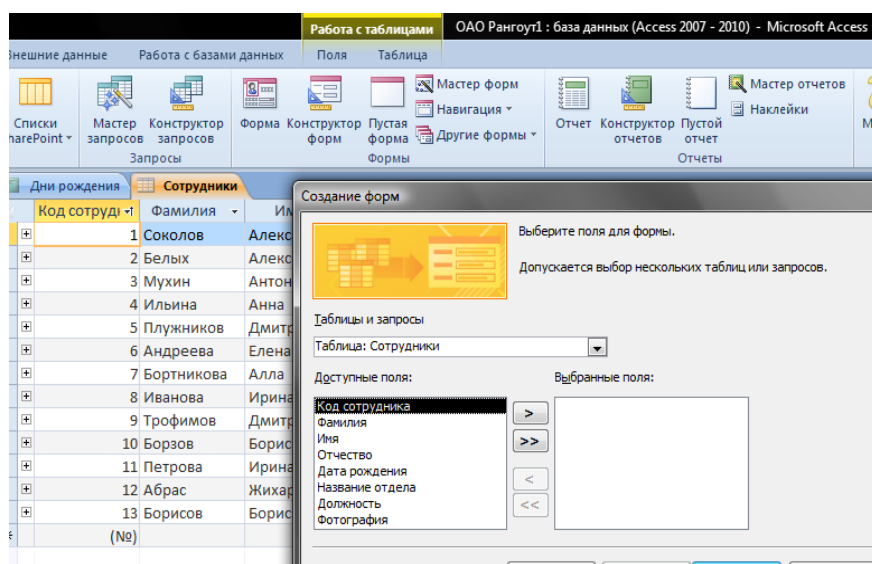


Рис. 30. «Мастер» форм

3. Создайте форму, следуя шагам «мастера», для чего:
 - 3.1. Выберите из таблицы поля для создания формы.
 - 3.2. В списке «**Таблицы и запросы**» выберите таблицу «**Сотрудники**». Перенесите из списка «**Доступные поля**» в список «**Выбранные поля**» все поля таблицы, щелкнув на кнопке «**Добавить все**» «>>>» (рис. 31);

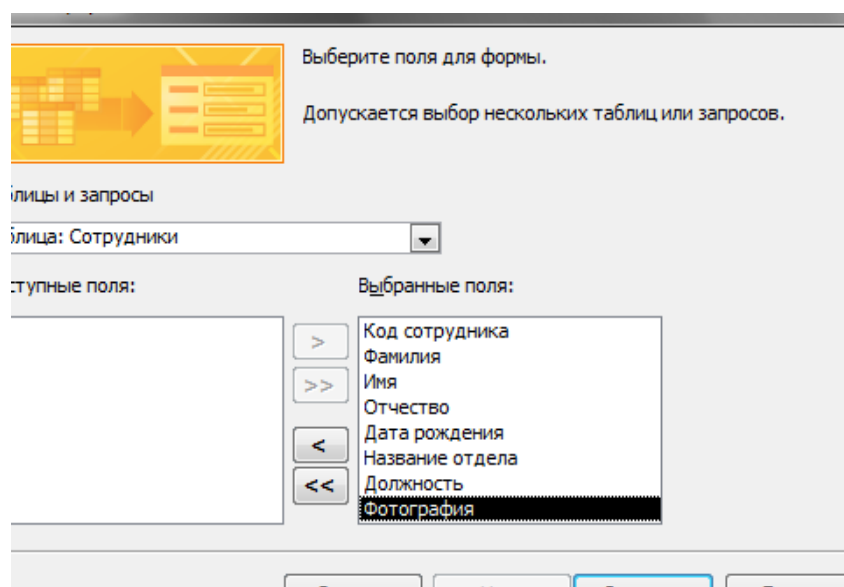


Рис. 31. Выбор полей формы

3.3. Выберите таблицу **«Личные данные»**.

3.4. Перенесите из нее все поля (кроме первого поля **«Код Сотрудника»**, так как оно уже есть в списке).

3.5. Перейдите на следующий шаг, нажав кнопку **«Далее»**.

3.6. Выбор вида формы. Установите переключатель **«В один столбец»** - это наиболее удобный вид формы.

3.7. Перейдите на следующий шаг, нажав кнопку **«Далее»**.

3.8. Выберите стиль оформления. Просмотрите список представленных стилей, попеременно щелкая на их названиях и выберите понравившийся.

3.9. Перейдите на следующий шаг, нажав кнопку **«Далее»**.

3.10. Введите имя формы **«Сотрудники»**.

3.11. Установите переключатель **«Открыть форму для просмотра и ввода данных»**.

3.12. Завершите работу с «мастером» щелчком на кнопке **«Готово»**.

4. Просмотрите имеющиеся записи, щелкая по кнопкам:

- переход к первой записи ;
- переход на предыдущую запись ;
- переход на следующую запись ;
- переход к последней записи ;
- переход на новую запись .

Чтобы улучшить наглядность и дизайн формы, её нужно отредактировать. Процесс редактирования предполагает изменение вида некоторого объекта с целью его улучшения. При редактировании формы можно изменить названия полей для ввода, увеличить или уменьшить размер области ввода данных, изменить порядок расположения полей в карточке и многое другое. Для редактирования формы надо перейти в режим «конструктора» (рис.32).

Код сотруд	Код сотрудника	Улица	Улица
Фамилия	Фамилия	Дом	Дом
Имя	Имя	Квартира	Квартира
Отчество	Отчество	Домашний телефон	Домашний телефон
Фотографии	Фотографии	Мобильный телефон	Мобильный телефон
Дата рождения	Дата рождения		
Название отдела	Название отдела		
Должность	Должность		
Паспорт	Паспорт		
Дата поступления на работу	Дата поступления на работу		
Индекс	Индекс		
№ приказа	№ приказа		

Рис.32. Неотредактированная форма «Сотрудники»

Можно использовать «конструктор» и для создания форм, но это очень трудоемкая работа. Поэтому лучше поручить ее «мастеру», а «конструктор» - только для редактирования формы. Порядок действий при редактировании следующий:

1. Откройте созданную форму «Сотрудники» (рис 33).
2. Перейдите в режим «Конструктора».

Примечание. В режиме «конструктора» рабочее поле представляет собой сетку с выделенными областями для расположения объектов: «Заголовок формы», «Область данных», «Примечание формы». Если навести указатель мыши на границу области, ее можно увеличить или уменьшить. Основные объекты расположены в «Области данных». Это – надписи полей и поля. Обратите внимание, что надпись и название поля могут не совпадать. Информация, расположенная в рамке надписи, не меняется при просмотре записей. Это аналог неизменяемой части карточки. Поле предназначено для ввода данных. В окне «конструктора» оно представляет собой белый прямоугольник с одной или несколькими строками. Первоначально в режиме «Конструктора» и для надписи, и для поля указано имя соответствующего поля. Вы можете отредактировать надпись. Название поля менять не рекомендуется, так как это может привести к ошибке.

3. Измените размеры рамки для фотографии, для этого:

- 3.1. Щелчком выделите рамку, по границе рамки появятся маркеры.
- 3.2. Подведите курсор к угловому маркеру так, чтобы он принял форму двусторонней стрелки.

- 3.3. Потяните границу за маркер и увеличьте ее; размер рамки контролируйте по сетке, в которой одна клетка соответствует 1 см.
4. Измените свойства рамки так, чтобы фотография полностью помещалась внутри рамки независимо от ее исходного размера. Для этого:
 - 4.1. Правой кнопкой мыши щелкните внутри рамки.
 - 4.2. В контекстном меню выберите пункт «Свойства».
 - 4.3. В открывшемся окне свойств рамки выберите вкладку «Макет».
 - 4.4. Щелкните на строке «Масштабы рисунка» и пункт списка «По размеру рамки» (рис. 33).
5. Перейдите в режим формы и просмотрите все записи; убедитесь, что теперь все фотографии полностью размещаются внутри рамки.
6. Аналогично измените размеры рамок надписей.
7. Переместите рамки некоторых надписей в правую часть формы (рис. 28).
8. Измените шрифт названия надписей. Для этого:
 - 8.1. Выделите щелчком надпись поля «Код Сотрудника», при выделении по границе надписи появляются маркеры выделения.

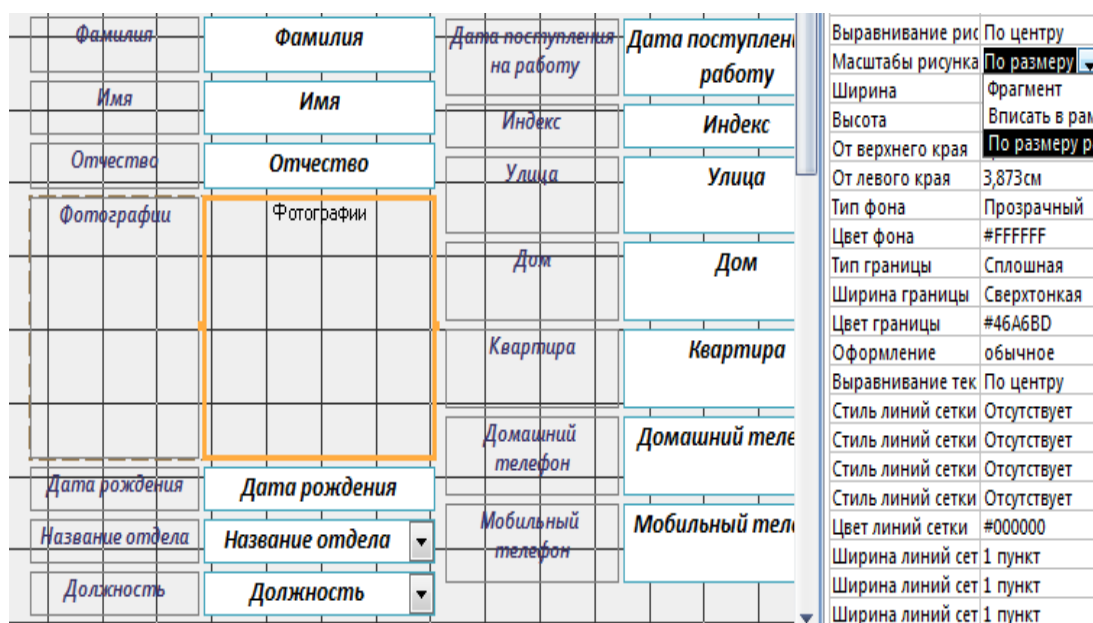


Рис. 33. Редактирование формы в режиме конструктор

- 8.2. Кликните внутри надписи и появится мигающий текстовый курсор, измените шрифт и его размеры.
- 8.3. Аналогичным образом измените вид других надписей.
9. Измените, если необходимо, размеры формы. Для этого:
 - 9.1. Подведите курсор к нижней границе так, чтобы он принял форму двусторонней стрелки.
 - 9.2. Захватите границу формы (не путать с границей окна) и потяните вниз.
 - 9.3. Аналогичным образом измените горизонтальный размер формы.

На рис.34 представлена форма, полученная после работы «конструктора».

Сотрудники			
Код сотрудника	1	№ Приказа	56
Фамилия	Петров	Дата поступления на работу	12.11.2006
Имя	Сидр	Индекс	193301
Отчество	Иванович	Улица	Куйбышева
Фотографии		Дом	19
		Квартира	54
		Домашний телефон	2363348
		Мобильный телефон	8-911-98778899
Дата рождения	07.09.1975		
Название отдела	Дирекция		
Должность	директор		
Паспорт	4003-601234		

Рис 34. Форма «Сотрудники»

4.3.4. Создание и печать отчетов

Одной из основных задач создания и использования баз данных является предоставление пользователям необходимой информации на основе существующих данных. В Access для этих целей предназначены отчеты, позволяющие выбрать из базы данных требуемую пользователем информацию, представить выводимую информацию на печать в желаемом пользователю формате. Отчет можно дополнить рисунками, графиками, колонтитулами и т.д. Источником данных для отчета могут быть таблицы или запросы. Отчеты и формы Access имеют много общего, но в отличие от форм, отчеты предназначены только для просмотра и печати данных. Отчет может быть создан также, как и форма с помощью «мастера», его разделы могут включать заголовок отчета, область данных, верхний и нижний колонтитулы и т.д. Access позволяет создавать отчет на основе таблицы или запроса, для чего необходимо выбрать объект «Отчет» и таблицу для его построения. Наиболее просто это реализуется с помощью «Мастера отчетов», который предлагает за ряд шагов для выбранной таблицы «Сотрудники» сформировать отчет.

- Во вкладке Создание в группе Отчеты выберете «Мастер отчетов» и запустите его.
 - Выберите из таблицы Структура ОАО поле Название отдела, из таблицы Сотрудники – поля Фамилия, Имя, Отчество, Должность (рис. 35).

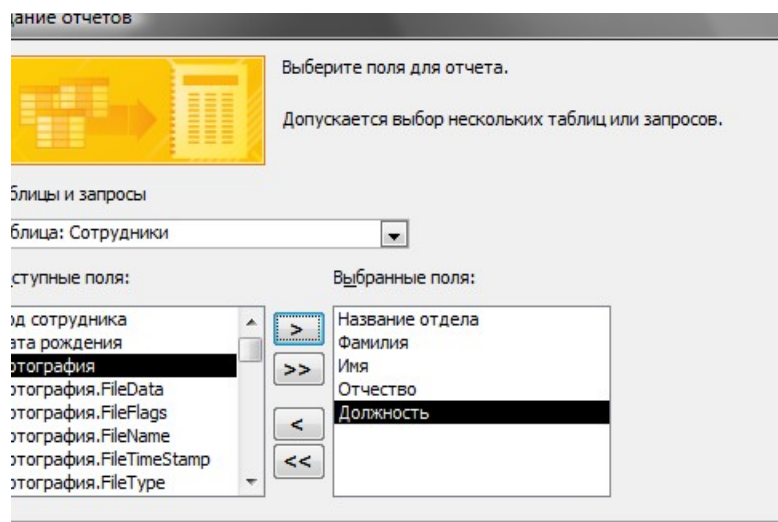


Рис. 35. Создание отчета с помощью мастера

1. Выберите группировку данных по названиям отделов. Группировка позволяет вывести названия отделов в виде оформленных подзаголовков (рис. 36).

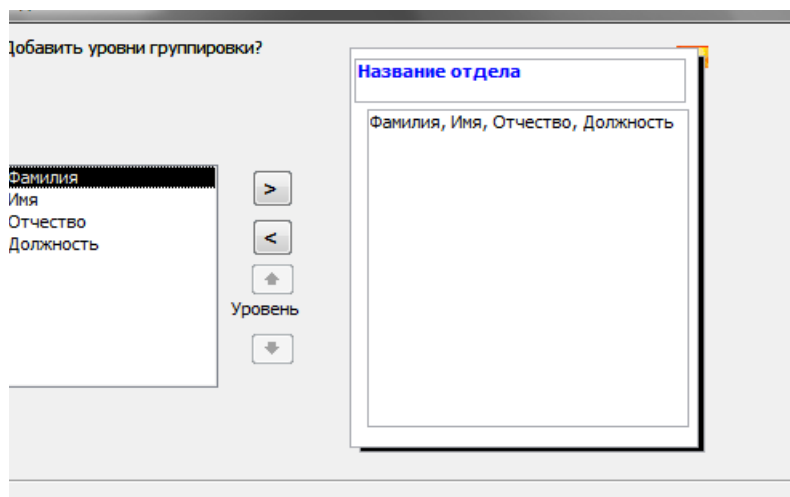


Рис. 36. Группировка данных

4. Задайте сортировку по полю Фамилия для того, чтобы фамилии в отчете были расположены в алфавитном порядке.
5. Выберите вид макета для отчета, например, Ступенчатый.
6. Введите заголовок отчета: Список сотрудников и завершите работу мастера, щелкнув на кнопке **Готово**.
7. После завершения работы мастера включается Режим предварительного просмотра отчета. Фрагмент отчета представлен на рис. 37.

Сотрудники				
Название отдела	Фамилия	Имя	Отчество	Должность
Отдел рекламы	Борзов	Борис	Александрович	Специалист
Отдел снабжения	Андреева	Елена	Федоровна	Менеджер
	Белых	Алексей	Иванович	Специалист
	Бортникова	Алла	Александровна	Менеджер
Дирекция	Борисов	Борис	Борисович	Менеджер
	Ильина	Анна	Васильевна	Секретарь
	Плужников	Дмитрий	Андреевич	Замдиректора

Рис.37. Отчет «Сотрудники»

Однако, при просмотре можно заметить некоторые недочеты, которые требуют исправления и корректировки, например, название столбцов расположены не по центру, и т.д. Для исправления отчета (изменение свойства объектов – шрифт, цвет, расположение элементов) перейдите в режим «Конструктора». Отчет имеет несколько областей, в которых расположены объекты отчета: надписи, поля и другие элементы (рис.38).

The screenshot shows the design view of a report titled 'Сотрудники'. It displays several regions: 'Заголовок отчета' (Report Header) containing the title; 'Верхний колонтитул' (Upper Footer) with fields for department name, surname, name, patronymic, and position; 'Заголовок группы' (Group Header) for department names; 'Область данных' (Data Area) with fields for employee details; and 'Нижний колонтитул' (Lower Footer) with page numbering formulas like '=Now()' and '= "Стр. " & [Page] & " из " & [Pages]'.

Рис. 38. Вид областей отчета

Также как и в форме, исправлять можно только надписи. Поля базы данных исправлять нельзя, так как это может привести к ошибке.

Примечание. Чтобы узнать с каким объектом вы работаете, наведите курсор на объект и кликните правой кнопкой. В появившемся контекстном меню выберите пункт Свойства. В титульной строке окна свойств будет написано название объекта.

1. Кликните мышкой поочередно на тех полях (эти поля будут выделяться маркерами), которые вы хотите откорректировать.
2. Кликните внутри надписи: появится текстовый курсор. Задайте для названий таблицы и столбцов форматирование по центру.
3. Во вкладке **Конструктор** выберите **Элементы управления**.

4. На панели элементов выделите кнопку **Надпись** и нарисуйте небольшую рамку для надписи в области заголовка. В рамке напишите «По состоянию дел на: « »».

5. Перенесите из области нижнего колонтитула в область заголовка объект с функцией Now(), который выводит в отчет текущую дату.

6. Установите начертание поля группы: полужирный, курсив.

Фрагмент откорректированного отчета представлен на рис. 39.

Список сотрудников				
По состоянию дел на: 2 ноября 2014 г.				
Название отдела	Фамилия	Имя	Отчество	Должность
Отдел рекламы	Борзов	Борис	Александрович	Специалист
Отдел снабжения	Андреева	Елена	Федоровна	Менеджер
	Белых	Алексей	Иванович	Специалист
	Бортникова	Алла	Александровна	Менеджер
Дирекция	Борисов	Борис	Борисович	Менеджер

Рис.39. Фрагмент отчета «Сотрудники»

4.4. Основные этапы разработки базы данных

Процесс проектирования базы данных представляет собой последовательность переходов от неформального словесного описания информационной структуры определенной предметной области к формализованному описанию объектов предметной области в терминах реляционной модели. В общем случае можно выделить следующие этапы проектирования:

- **Определение цели создания базы данных.** На данном этапе выполняется системный анализ и словесное описание информационных объектов предметной области. Определяется назначение и использование базы данных, выбираются фактические данные, которые нужно хранить в базе данных и по каким темам они должны быть распределены. Темам должны соответствовать таблицы, а данным - поля (столбцы) в этих таблицах.
- **Определение полей в базе данных.** Каждое поле содержит требуемые фактические данные, например, фамилия, имя, отчество, адрес, должность, номер телефона и т.п. Для каждого типа сведений следует выделить отдельное поле в таблице. При составлении схемы полей учитывайте следующее: включать все необходимые сведения и разбивать информацию на минимальные логические компоненты. Например, имена сотрудников удобно разбить на отдельные поля: Фамилия, Имя, Отчество, что облегчит сортировку по фамилиям. Например, в таблице, содержащей поля «Цена» и

«Количество» не следует создавать поле, содержащее произведение значений этих полей.

- Определение таблиц и полей таблиц базы данных. Каждая таблица должна содержать данные только на одну тему. Список нужных полей покажет, какие требуются таблицы. Каждое поле необходимо включать только в одну таблицу, не надо включать поле в таблицу, если в результате его добавления одни и те же данные будут появляться в нескольких записях этой таблицы.
- Определение полей с уникальными значениями в записи. Для связывания данных, хранящихся в разных таблицах, каждая таблица должна содержать поля, однозначно определяющие каждую запись. Такое поле называют первичным ключом. Первичный ключ всегда имеет уникальный индекс и используется для связывания таблицы с внешними ключами в других таблицах.
- Определение связей между таблицами. После разбиения сведений на таблицы и определения полей первичного ключа необходимо выбрать способ, которым будут связаны сведения. Для этого следует определить связи между таблицами базы данных.
- Усовершенствование структуры базы данных. После определения и создания требуемых таблиц, полей и связей, пока таблицы не заполнены данными, необходимо проанализировать полученную структуру базы данных и выявить возможные недостатки. Для этого следует ввести в таблицы необходимый объем данных для проверки структуры, создать между ними связи. Для проверки связей в базе данных создайте запросы для получения нужных данных, также рекомендуется создать пробные формы, отчеты и посмотреть, отображаются ли в них нужные данные. Проанализируйте избыточные повторы данных в таблицах и исключите их.
- Ввод данных и создание требуемых объектов базы данных. После коррекции структуры таблиц в соответствии с заданными требованиями выполняется ввод всех данных. Затем создаются все необходимые объекты базы данных: запросы, формы, отчеты, страницы доступа к данным, макросы и модули.

4.5. Разработки базы данных средствами СУБД Access

Рассмотрим пример, который позволит дать представление об этапах разработки базы данных для решения прикладных задач. Предположим, что транспортная компания создает систему, помогающую менеджеру-логисту автоматизировать процессы работы с клиентами и контроль исполнения заказов на доставку товара. Система должна обеспечить решение следующих функциональных задач:

- получение полной информации о потребителе и его реквизитах;
- учет, обработку и оформление заказов, поступивших от менеджеров;
- учет сведений о заказах;
- расчет перегруза и вместимости товара;

- ведение отчетности.

Можно выделить следующие массивы данных:

- Потребители.
- Сотрудники.
- Перевозчики.
- Товар.
- Ж/д станции.
- Тарифы ж/д.
- Заказано.
- Заказы.

Определим атрибуты объектов БД и связи между ними:

Потребители	
Код потребителя (КП)	Число
Краткое наименование	Текст
Полное наименование	Текст
Юридический адрес	Текст
Код сотрудника (КСТР)	Число
Код ж/д станции (КС)	Число
Ж/д реквизиты адр. получателя	Текст
Почтовый адрес	Текст

Сотрудники	
Код сотрудника (КСТР)	Число
ФИО	Текст
Должность	Текст
Телефон	Число
Доп. информация	Текст

Перевозчики	
Код транспортной кампании (КТК)	Число
Транспортная компания	Текст
Телефон	Текст
Контактное лицо	Текст
Номер договора	Текст
Договор действует до	Дата

Товар	
Код товара (КТ)	Число
Наименование товара	Текст
Литраж	Текст
Кол. бут в коробе	Число

Вес короба с гот. продукцией	Число
------------------------------	-------

Ж/Д станции	
Код станции (КС)	Число
Ж/Д станция	Текст
Доп. данные	Текст

Тарифы. Ж/Д	
Код транспортной компании (КТК)	Число
Код станции (КС)	Число
Цена	Число

Заказано	
Код заказа (КЗ)	Число
Код товара (КТ)	Число
Количество	Число
Цена	Число

Заказы	
Код заказа (КЗ)	Число
Код покупателя (КП)	Число
Код транспортной компании (КТК)	Число
Код сотрудника (КСТР)	Число
Дата исполнения	Дата
Стоимость утепления	Число
Номер контейнера	Текст
Дата прибытия	Дата
Номер ТТН (товарно-транспортной накладной)	Текст

В рассматриваемом примере объекты вступают во взаимоотношения «один-ко-многим». Это означает, например, что один сотрудник может оформить несколько заказов и работать со многими потребителями, один товар может входить во многие заказы, один перевозчик может осуществлять доставку многих заказов, но один и тот же заказ не могут выполнять несколько перевозчиков. Связи между объектами проектируемой БД представлены на рис. 40.

На основании модели, изображенной на рис. 40, построим логическую модель данных. Для реляционной модели данных каждый объект

преобразуется в набор отношений (таблиц), построенных по определенным, строго заданным правилам. Таблица состоит из столбцов (полей) и строк (записей).

Для этого требуется выполнить следующие действия:

- создать по одной таблице для каждого объекта.
- для каждого объекта, вступающего во взаимоотношения с другими объектами как «один-ко-многим», указать один столбец в качестве первичного ключа.
- задать первичный ключ для каждого объекта, выступающего во взаимоотношениях как «многие-к-одному».

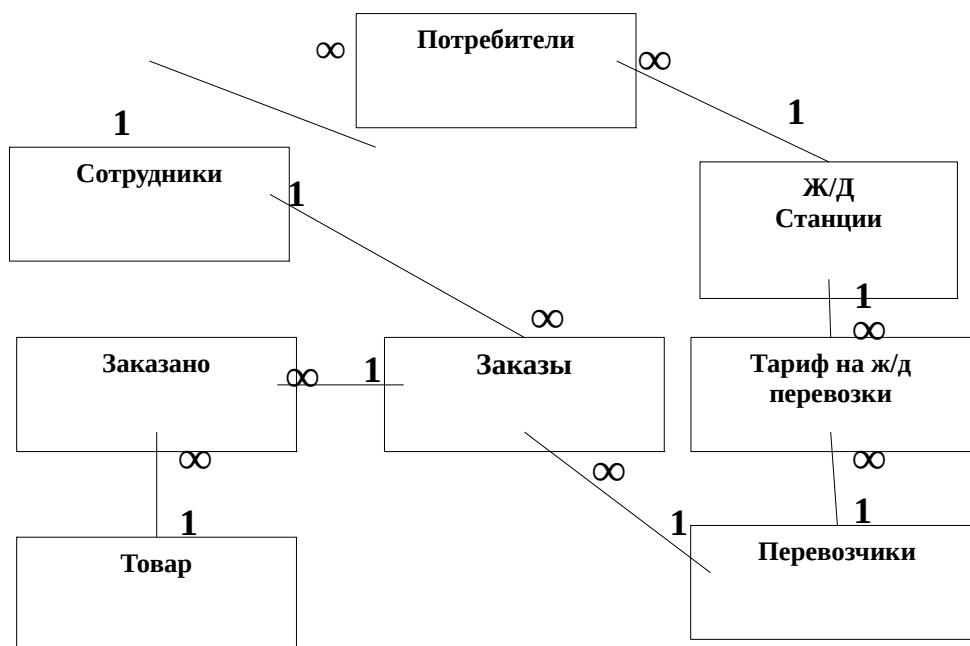


Рис. 40. Связи между объектами БД

Выполним это преобразование для нашего примера. На основании модели, изображенной на рис. 40, можно построить 8 таблиц: Потребители, Сотрудники, Перевозчики, Товар, Ж/Д станции, Тарифы Ж/Д, Заказано,

Заказы. Зададим первичные ключи для этих таблиц, выступающих в связях как «один-ко-многим». Таблица «Заказы» имеет в своем составе уникальное для каждой строки поле — это «**Код заказа**». Таблицам «Потребители», «Сотрудники», «Перевозчики», «Товар», «Ж/Д станции», назначим первичные ключи:

- Код потребителя (КП)
- Код сотрудника (КСТР)
- Код транспортной кампании (КТК)
- Код товара (КТ)
- Код станции (КС)
- Код товара (КТ).

Логическая модель для связи отношений представлена на рис.41.

Рассмотрим более подробно структуру таблиц.

Таблица «Товар» содержит поля - характеризующие тип товара и имеет связь типа «один ко многим» с таблицей «Заказано». Это означает, что в одной заявке на заказ может числиться товар различных типов.

Таблица «Заказано» содержит поля наименование товара, количество и цену. Помимо связи с таблицей «Товары» имеет связь «один ко многим» с таблицей «Заказы». Это означает, что в одном заказе на перевозку может быть много заявок.

Таблица «Заказы» содержит реквизиты товара, данные перевозчика, пункт назначения, стоимость услуги. Помимо связи с таблицей «Заказано» имеет связь типа «многие к одному» с таблицей «Сотрудники». Это означает, что один сотрудник может оформить несколько заказов.

Таблица «Сотрудники» содержит информацию о менеджерах компании и помимо связи с таблицей «Заказы» имеет связь «один ко многим» с таблицей «Потребители». Это означает, что один менеджер может иметь контакт со многими заказчиками.

Таблица «Потребители» содержит информацию о контрагентах и их реквизитах. Помимо связи с таблицей «Сотрудники» имеется связь типа «многие к одному» с таблицей «Ж/Д станции». Это означает, что один контрагент может заказывать товар только по одному.

Таблица «Ж/Д станции» помимо связи с таблицей «Потребители» имеет связь типа «один ко многим» с таблицей «Тарифы на ж/д». Это означает, что стоимость доставки для одного и того же пункта назначения может быть различной.

Таблица «Перевозчики» имеет связь типа «один ко многим» с таблицами «Заказы» и «Тарифы на ж/д». Это означает, что одна транспортная фирма может осуществлять доставку различных заказов и по различным тарифам.

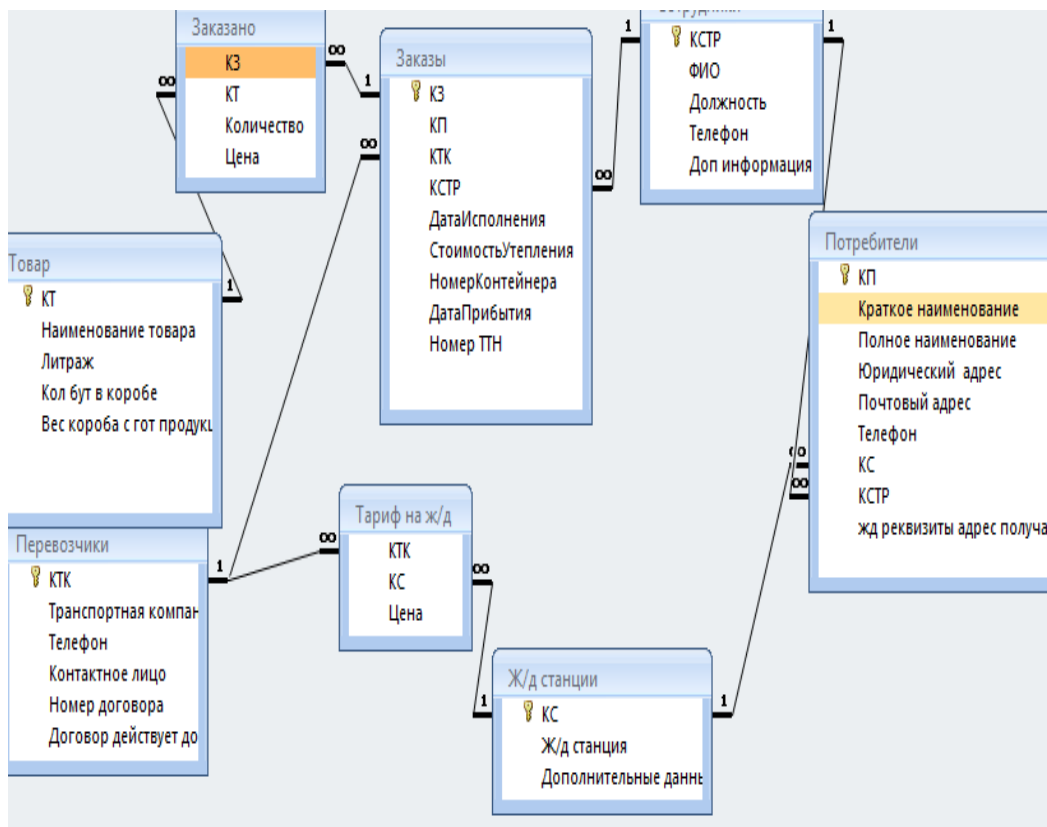


Рис.41. Логическая схема данных

Для рациональной и удобной эксплуатации БД необходимо создание других объектов, которые применяются в Access и, в первую очередь, форм, запросов и отчетов.

Формы предназначены для создания пользователю комфортного интерфейса и дают возможность выводить на экран данные в виде электронного бланка. Для рассматриваемого примера создадим следующие формы: «Главная», «Заказы», «Поиск по дате исполнения», «Поиск по № контейнера», «Отгрузки за период», «Сверка по перевозчикам», «Сотрудники», «Потребители», «Товар», «Ж/д станции», «Тарифы по ж/д». В качестве иллюстрации рассмотрим создание формы «Товар», предназначенной для ввода данных о перевозимых товарах. Для этого перейдем на вкладку «Создание» и выберем «мастер» форм. В открывшемся окне выберем таблицу «Товар» и соответствующие поля (рис.42). Нажимаем «Далее», выбираем внешний вид, требуемый стиль и задаем название формы «Товар».

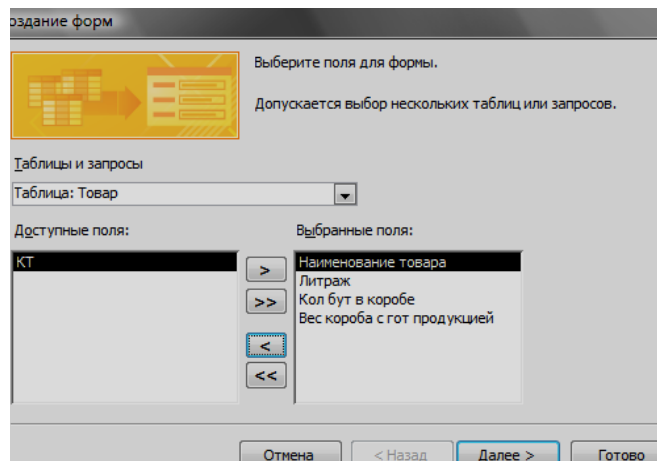


Рис.42. Создание формы «Товар»

Закрываем форму и выбираем режим «Конструктор». Раздвигаем область данных, регулируем размер полей, выбираем тип оформления, создаем кнопку «найти запись» и т.д. (рис. 43).

Запрос представляет собой выраженное информационное требование и позволяет производить поиск и анализ информации в таблицах базы данных, отвечающей определенным критериям. Для нашей БД создадим следующие запросы:

- «Заявка на контейнер» (необходимо ввести № заказа), содержит поля: код заказа, реквизиты покупателя, дату исполнения, стоимость специальных услуг.
- «Отгрузки», содержит поля: полное наименование заказчика, пункт назначения, доп. данные пункта назначения, название компании перевозчика, дата исполнения, цена контейнера.
- «Отгрузки за период» (необходимо ввести начальную и конечную даты), содержит поля: полное наименование заказчика, ж/д станция, транспортная компания, номер контейнера, дата исполнения, дата прибытия, сумма за услуги.
- «Поиск номера контейнера» (необходимо ввести № контейнера), содержит поля: полное наименование (табл. «Потребители»), Ж/Д станция, транспортная компания (табл. «Перевозчики»), № контейнера (табл. «Заказы»), Дата исполнения (табл. «Заказы»), Дата прибытия (табл. «Заказы»).
- «Поиск по дате исполнения» (введите дату) содержит поля: полное наименование (табл. «Потребители»), Ж/Д станция, транспортная компания (табл. «Перевозчики»), № контейнера (табл. «Заказы»), Дата исполнения (табл. «Заказы»), Дата прибытия (табл. «Заказы»).
- «Промежуточные данные заказа» содержит поля: код заказа (табл. «Заказано»), Дата исполнения (табл. «Заказы»), промежуточная стоимость (количество*заказано), всего мест (количество*товар), тоннаж (количество*товар).

Наименование товара	Литраж	Кол бут в коробе	Вес короба с гот продукцией
Морс вишневый	0,5	20	17
Морс вишневый	0,7	20	18
Морс вишневый	0,1	40	16
Вермут Монелиано Лайт	0,5	12	15
Сок гранатовый	0,5	20	19
Лимонад Ситро	0,5	20	20

Рис.43. Форма «Товар»

- «Услуги перекрестный» содержит поля: транспортная компания, итого за год, по месяцам.

Для примера рассмотрим создание запроса «Отгрузки». Запустим режим создания запроса с помощью «мастера» (простой запрос) и произведем выборку полей как показано на рис. 44. Нажимаем «Далее», выбираем тип запроса и задаем имя запроса «Отгрузки». Завершаем работу «мастера», нажав кнопку «Готово».

Выбираем режим «Конструктор». Откроется бланк запроса, в верхней части бланка расположена схема связи таблиц, используемых в запросе. В нижней части расположена таблица описания полей запроса.

Выберите поля для запроса.
Допускается выбор нескольких таблиц или запросов.

Таблицы и запросы
Таблица: Заказы

Доступные поля:

- КП
- КТК
- КСТР
- СтоимостьУтепления
- НомерКонтейнера
- ДатаПрибытия
- Номер ТТН

Выбранные поля:

- Ж/д станция
- Дополнительные данные
- Полное наименование
- Транспортная компания
- ДатаИсполнения
- КЗ

Отмена < Назад Далее > Готово

Рис.44. Создание запроса «Отгрузки» с помощью «мастера»

В первой строке перечислены поля запроса. Во второй строке указано имя таблицы, из которой взято поле. В третьей строке можно задать сортировку полей (рис. 45).

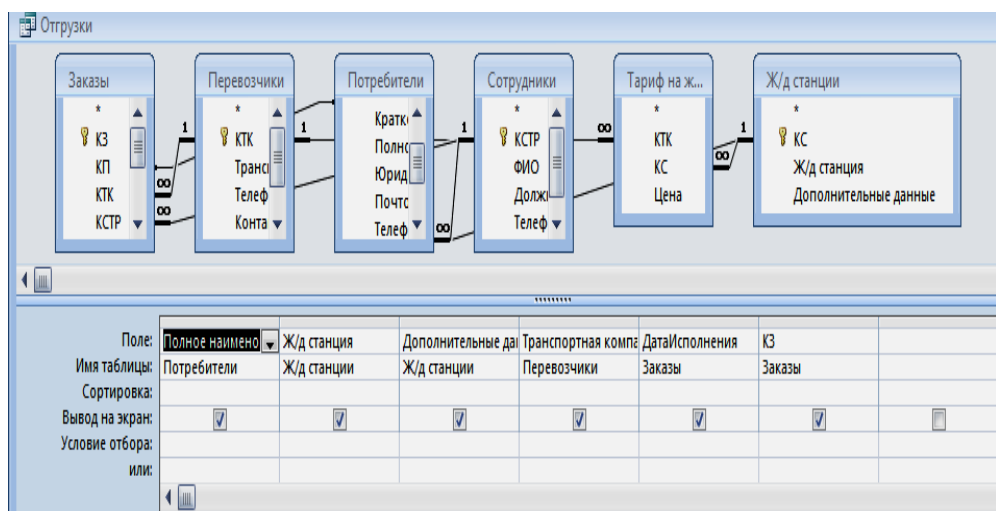


Рис.45. Бланк запроса

В седьмом столбце создадим поле «Стоимость контейнера». Наберем название поля, правой кнопкой мыши кликнем на седьмой графе строки «Поля», откроется окно «Построитель выражений». В левом окне обзора раскроем папку «Таблицы» и в ней последовательно вложенную папку «Тариф на ж/д» (поля таблицы будут представлены в среднем окне построителя запросов) и папку «Заказы». Введем формулу для вычисляемого поля согласно схеме рис. 46 и завершаем работу нажатием кнопки «ОК».

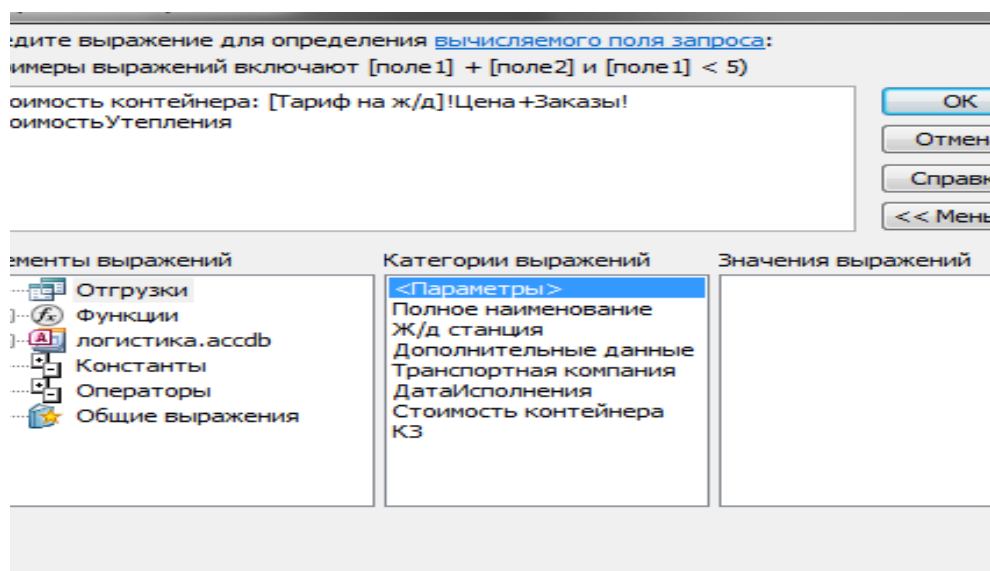


Рис.46. Использование построителя выражений

Отчет позволяет представить информацию на печать в удобном виде. Его можно дополнить диаграммами и рисунками, которые сделают данные более наглядными. Для рассматриваемого примера необходимо разработать два отчета «Заявка на контейнер» с полями: покупатель; перевозчик; дата

исполнения; всего мест; тоннаж; ж/д назначения; дополнительные данные; полное наименование заказчика; ж/д реквизиты и «Отчет по услугам», содержащий поля: транспортная компания; итого за год; по месяцам.

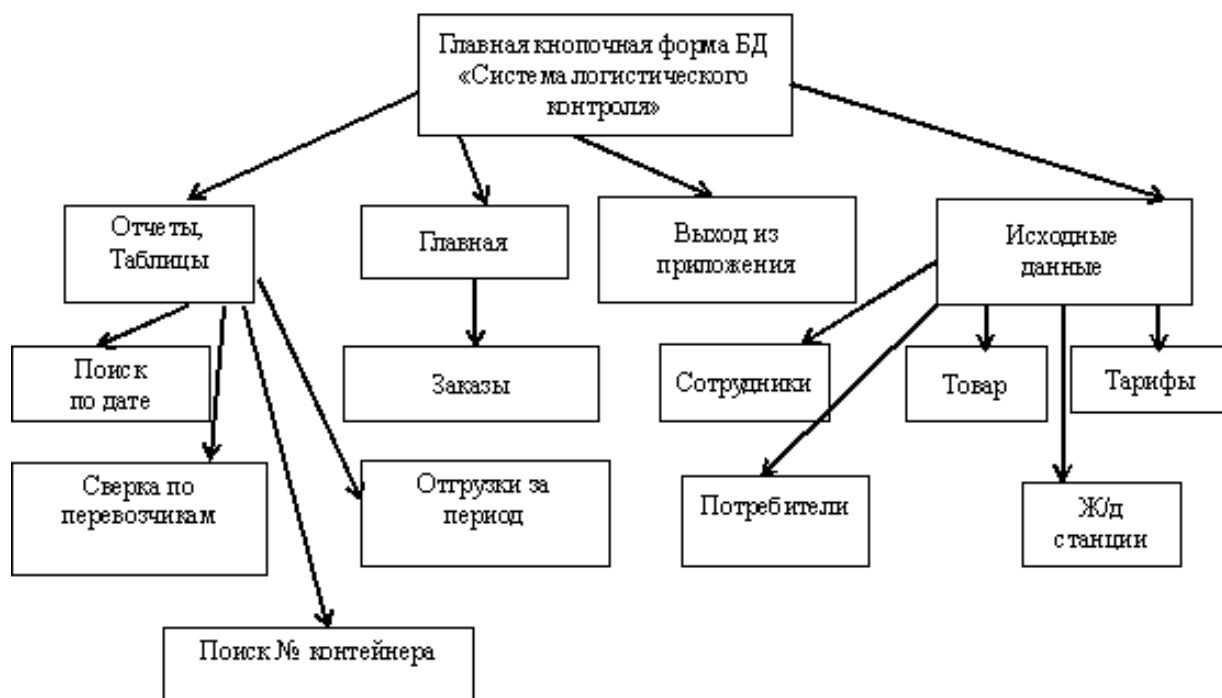


Рис. 47. Структура главной кнопочной формы

Для комфортной эксплуатации БД и учитывая, что не все сотрудники компании владеют необходимыми навыками работы в Access, полезно создавать главную кнопочную форму. Она позволяет, не вникая в детали построения БД, выводить нужные объекты на экран монитора, производить корректировку данных, печатать отчеты и т.д. Перед ее созданием целесообразно разработать структуру, в которую включить все необходимые управляющие элементы (рис.47).

Для её создания на вкладке «Работа с базами данных» выберем команду «Диспетчер кнопочных форм», если эта команда отсутствует, то она может быть получена следующими действиями.

На вкладке **Файл** выбираем **Параметры**. Откроется окно **Параметры Access** (рис. 48). Далее выбираем **Настройка ленты**. В списке справа выбираем **Работа с базами данных** и нажимаем на кнопку **Создать группу**. В левом списке выбираем **Диспетчер кнопочных форм** и нажимаем кнопку **Добавить**.

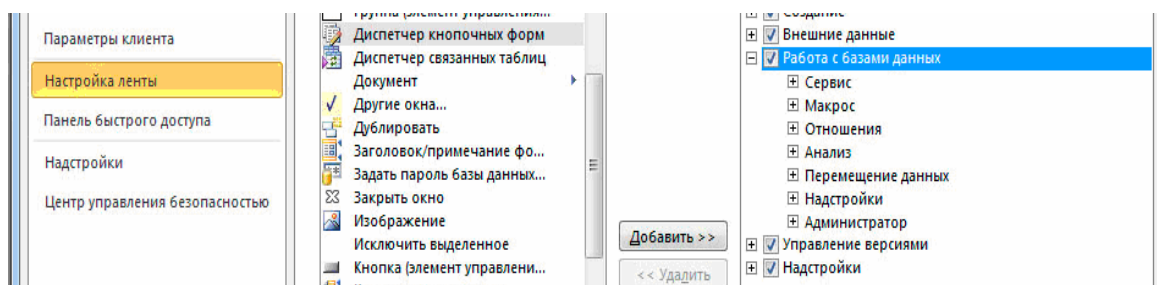


Рис. 48. Окно параметры Access

В результате окно диалога должно измениться так, как показано на рис.49.

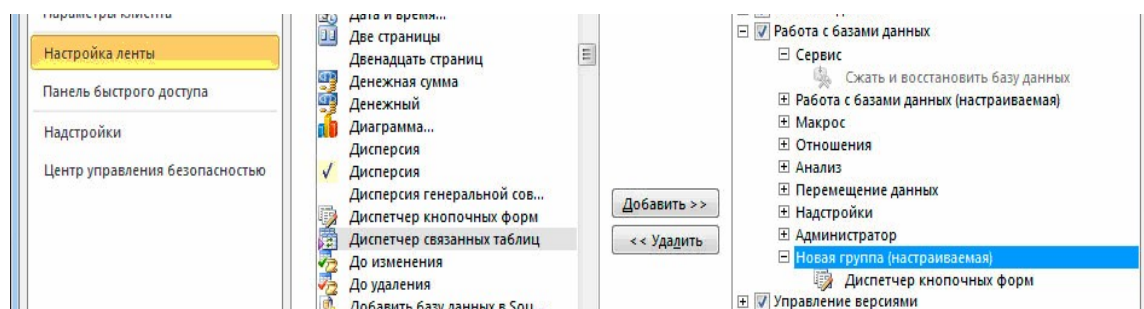


Рис.49. Работа с параметрами

Теперь на вкладке **Работа с базами данных** в группе **Работа с базами данных** нажмите кнопку **Диспетчер кнопочных форм** (рис. 50).

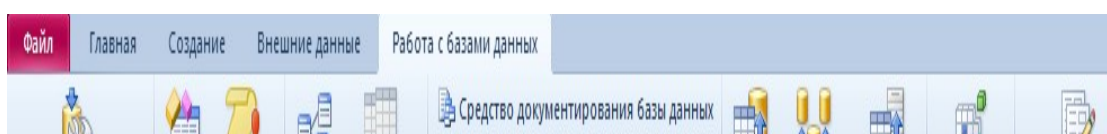


Рис.50.Вкладка Работа с базами данных, группа Кнопочная форма.

Нажмем кнопку **Диспетчер кнопочных форм**. Access сообщит о том, что не может найти кнопочную форму и предложит ее создать. Кликнем кнопку «Да» для продолжения работы «Диспетчера кнопочных форм». Далее нужно создать еще три страницы кнопочной формы – «Главная», «Отчеты, таблицы», «Исходные данные». Для создания новой команды меню кликнем мышью кнопку «Создать». Введем текст «Главная», будет создана страница кнопочной формы «Главная». Повторяем последовательность действий и создаем кнопочные формы «Отчеты и таблицы» и «Исходные данные» (рис.51).

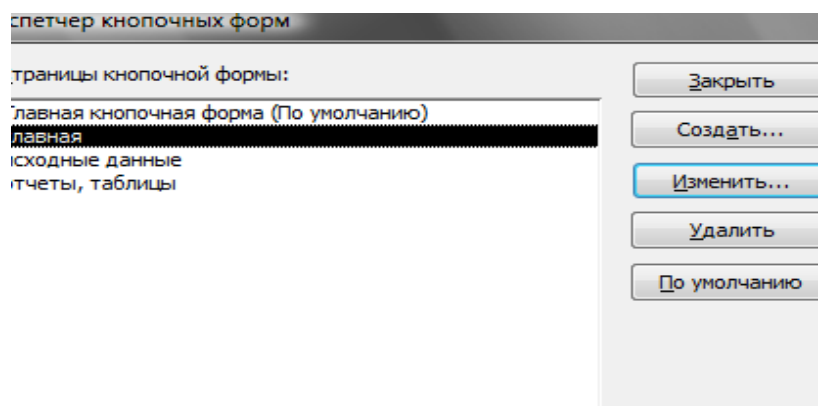


Рис.51. Страницы главной кнопочной формы

Теперь создадим элементы «Главной кнопочной формы». Выделяем страницу «Главная кнопочная форма» и нажимаем «Изменить», откроется новое окно «Изменение страниц кнопочной формы». В нем набираем текст «Главная», выбираем из раскрывающегося списка команду «Перейти к кнопочной форме» и из списка выбираем кнопочную форму «Главная». Нажимаем «ОК». Повторяем действия для создания элементов «Исходные данные», «Таблицы и отчеты» (рис.52).

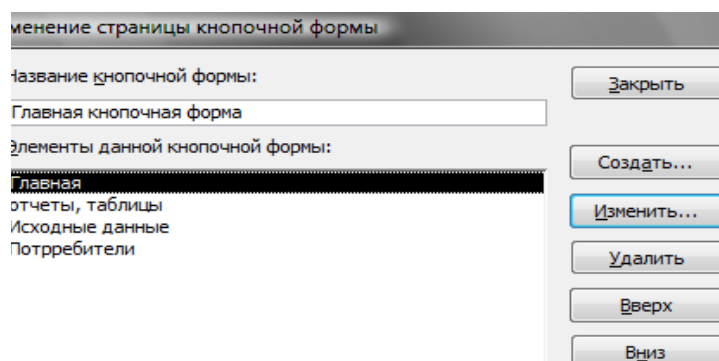


Рис.52. Элементы главной кнопочной формы

Теперь необходимо создать подчиненные формы для кнопочных форм «Главная», «Отчеты, таблицы» и «Исходные данные», которые бы позволяли выводить отчеты, просматривать и добавлять данные и т. п. Для создания элементов кнопочной формы "Главная" нужно выделить ее в окне «Диспетчер кнопочных форм» и щелкнуть на кнопке «Изменить», откроется окно диалога «Изменение страницы кнопочной формы». В этом окне щелкаем на кнопку «Создать», откроется новое окно «Изменение страницы кнопочной формы», набираем текст «Заказы», выбираем из раскрывающегося списка команду «Открыть форму для добавления» и выбираем из списка форму «Заказы». Аналогично создаем подчиненные элементы для кнопочных форм «Отчеты, таблицы» и «Исходные данные». Закрываем диспетчер кнопочных форм и открываем «Главную форму» в режиме «Конструктор» (рис.53).

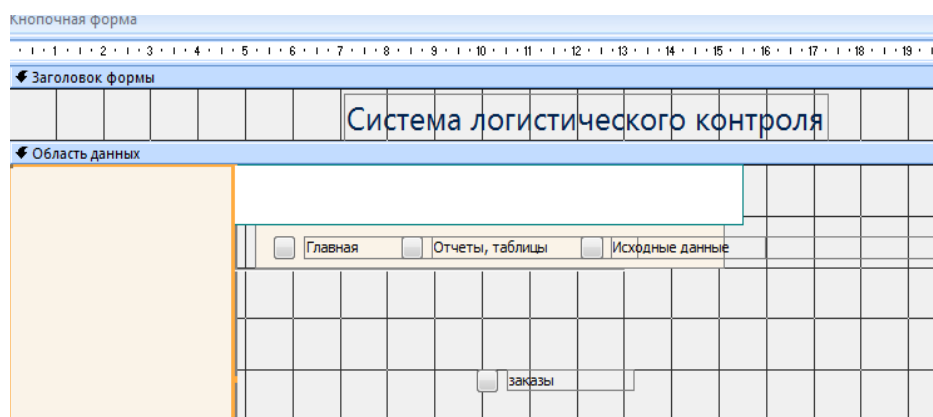


Рис.53. Главная кнопочная форма

Выполняем редактирование формы, добавляем управляющие кнопки, оформляем дизайн и создаем необходимые макросы для открытия нужных объектов.

В меню параметры Access выбираем команду «Текущая база данных» и в окне «Форма просмотра» указываем главную форму. Теперь после запуска БД на экране отобразится форма, показанная на рис. 54.



Рис.54. Главная кнопочная форма

На главной кнопочной форме расположено три вкладки:

- **Главная**, где расположена кнопка, которая открывает форму, содержащую данные о заказах.
- **Отчеты, таблицы** – в этой вкладке расположены кнопки, которые позволяют формировать отчеты или таблицы по заданным параметрам.
- **Исходные данные** – в этой вкладке расположены кнопки, где содержится информация о товаре, сотрудниках, перевозчиках, потребителях, ж/д станциях и их тарифах.
- **Выход** – это кнопка, с помощью которой осуществляется закрытие приложения Access.

При нажатии кнопки «Заказы» открывается форма «Заказы», представленная на рис.55.

Товар	Количество	Цена	Сумма
Морс вишневы 0,7	10	78,00р.	780,00р.
Сумма товара:			2260

Рис. 55. Форма «Заказы»

Данная форма отображает все данные о заказе: № заказа, покупателя, перевозчика, менеджера, дату исполнения, стоимость специальных услуг, № контейнера, дату прибытия, номер товарно-транспортной накладной, наименование товара, количество, цену за единицу продукции и сумму партии. С помощью данной формы можно распечатать и посмотреть отчет «Заявка на контейнер», используя соответствующие кнопки.

При нажатии кнопки **«Поиск по дате исполнения»** на экран выводится окно (рис. 56).

Рис. 56. Окно ввода параметра

При вводе параметра необходимо придерживаться заданного числового формата (ДД-ММ-ГГГГ). После ввода параметра на экран выводится форма (в случае, если в это время была отправка), представленная на рис. 57.

Полное наименование	Ж/д станция	Транспортная компания
ООО 'ВСК'	Абакан	Айхал

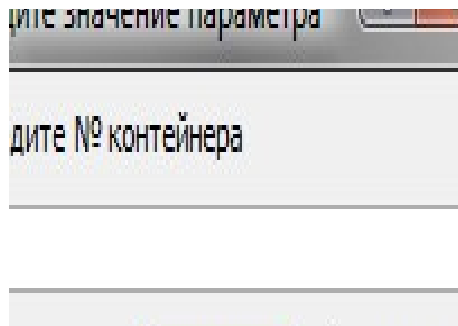
Номер контейнера	Дата исполнения	Дата прибытия на ст. назначения	Стоимость контейнера
RZDU 348902345	20-авг-2013	16-авг-2013	62 000,00р.

Номер ТТН
3548, 3550

Рис. 57. Отчет Поиск по дате исполнения

Форма содержит все данные о поставщиках и потребителях, контейнере и дате исполнения.

При нажатии кнопки «Поиск номера контейнера» на экран выводится окно (рис. 58).



Введите значение параметра

Введите № контейнера

Рис. 58. Окно ввод параметра

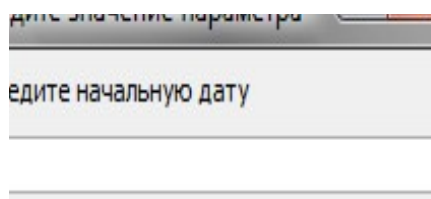
При вводе параметра необходимо придерживаться заданного числового формата, например, RZDU 348902345. Данные о номерах контейнера можно уточнить в таблице «Заказы».

После ввода параметра на экран выводится форма (в случае, если такой контейнер существует), представленная на рис.59.

Полное наименование		Ж/д станция	Транспортная компания	
ООО "ВСК"		Абакан	Айхал	
Номер Контейнера	Дата Исполнения	Дата прибытия на ст. назначения		Стоимость контейнера
RZDU 348902345	20-авг-2013	16-авг-2013		62 000,00р.
Номер ТТН				
3548, 3550				

Рис.59.Форма «Поиск по № контейнера»

Форма содержит данные о пункте назначения, перевозчике, стоимости и т.д. При нажатии кнопки «Отгрузки за период» на экран выводится первый параметр (рис.60).



Введите значение параметра

Введите начальную дату

Рис.60. Окно ввода начальной даты

После ввода начальной даты (ДД-ММ-ГГГГ) на экран выводится окно для ввода конечной даты (рис.61).

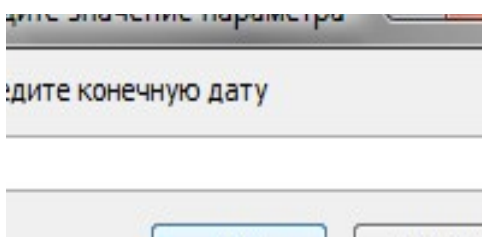


Рис.61. Окно ввода конечной даты

После ввода конечной даты (в формате ДД-ММ-ГГГГ) на экран выводится форма «Отгрузки за период» содержащая поля, представленные на рис.62.

Рис. 62.Форма «Отгрузки за период»

При нажатии кнопки «Сверка по перевозчикам» на экран выводится перекрестный запрос, содержащий поля, представленные на рис.63.

Рис. 63. Запрос «Сверка по перевозчикам»

С его помощью можно проследить совместную работу с транспортной компанией как за весь год, так и отдельно по каждому месяцу. При нажатии кнопки на вкладке «Отчеты и таблицы», запрос «Сверка по перевозчикам» выводится на печать (рис.64).

Услуги_перекрестный			
транспортная компания	Итого за год	янв	фев
кап	87 000,00р.	39 000,00р.	
апр	апр	май	июн
июл	авг	сен	окт
я	дек		

Рис. 64. Отчет «Услуги»

Вкладка «Исходные данные» предназначена для просмотра, добавления, изменения и удаления данных. Она содержит следующие кнопки:

- сотрудники;
- потребители;
- товар;
- ж/д станции;
- тарифы по ж/д.

При нажатии кнопки «Сотрудники» на экран выводится форма «Сотрудники» (рис. 65).

ФИО сотрудника	Должность	Местный телефон
Пуха Владимир Владимирович	Менеджер по продажам	202

Доп информация

Рис. 65. Форма «Сотрудники»

При нажатии кнопки «Потребители» на экран выводится форма, содержащая все данные о заказчике товара (рис. 66).

Потребители

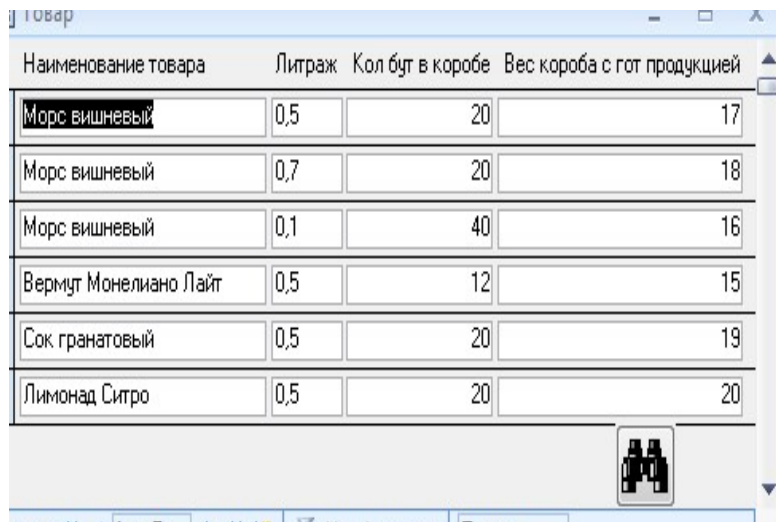
ФИО менеджера: Максимов Александр Владимирович

Краткое наименование	Полное наименование	Юридический адрес	ОКПО	Почтовый адрес	Телефон
ВСК	ООО 'ВСК'	655004, г. Абакан, ул. Советск	44438793	655004, г. Абакан, ул. Советск	

Записи: 1 из 1

Рис. 66. Форма «Потребители»

При нажатии кнопки «Товар» на экран выводятся данные о товаре, подлежащем транспортировке (рис. 67).



Наименование товара	Литраж	Кол бут в коробе	Вес короба с гот продукцией
Морс вишневый	0,5	20	17
Морс вишневый	0,7	20	18
Морс вишневый	0,1	40	16
Вермут Монелиано Лайт	0,5	12	15
Сок гранатовый	0,5	20	19
Лимонад Ситро	0,5	20	20

Рис. 67. Форма «Товар»

При нажатии кнопки «Ж/д станции» выводится информация о пункте назначения товара (рис. 68).

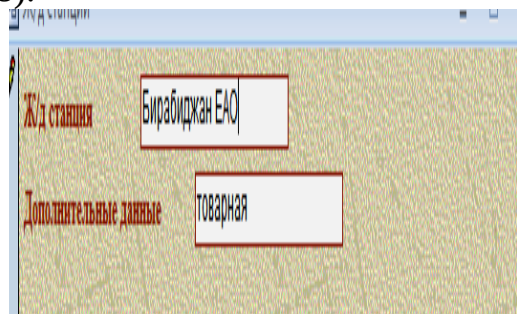
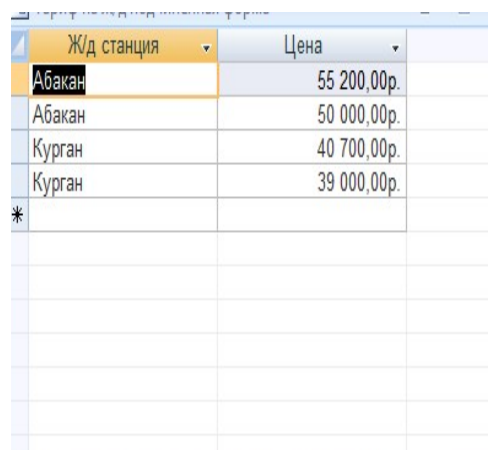


Рис. 68. Форма «Ж/Д станции»

При нажатии кнопки «Тарифы по ж/д» на экран выводится форма, информирующая о ценах за доставку товара (рис.69).



Ж/д станция	Цена
Абакан	55 200,00р.
Абакан	50 000,00р.
Курган	40 700,00р.
Курган	39 000,00р.
*	

Рис. 69. Форма «Тарифы»

В заключение рассмотрения данного примера проектирования базы данных следует отметить, что для самостоятельной разработки подобных или более сложных реляционных баз данных в среде Access следует расширять и углублять свои знания. Для этого можно использовать справочную систему Access, где можно найти ряд полезных примеров. Также много полезной информации можно найти в книгах по данной тематике и в Интернет.

4.6. Перспективы развития баз данных

Современные базы данных являются основой многочисленных информационных систем. В настоящее время продолжается изучение методов эффективной обработки баз данных с точки зрения извлечения из них дополнительных знаний. Базы данных выступают как хранилища информации, это направление называется Data Warehouse (хранилища данных). Существенной особенностью хранилищ данных является их очень большой объем и для работы с ними наиболее значимым становится так называемый интеллектуальный анализ данных (datamining). В качестве средств анализа данных хранилища используется SQL Server Analysis Services, применяемый для построения многомерных моделей данных. Это средство содержит ряд эффективных алгоритмов анализа данных с целью поддержки принятия управленческих решений, в том числе анализ тенденций и статистический анализ данных. Интеллектуальный анализ данных является быстро прогрессирующей областью исследований. Интеллектуальный анализ данных вошел в группу важнейших информационных технологий. Особенно широко данные методы применяются в экономике и бизнесе. Для этих областей интенсивно разрабатываются инструментальные средства высокого уровня, позволяющие решать достаточно сложные практические задачи без специальной математической подготовки пользователей. Актуальность использования в бизнесе интеллектуального анализа данных связана с жесткой конкуренцией, возникшей вследствие перехода от «рынка продавца» к «рынку покупателя». В этих условиях особенно важно качество и обоснованность принимаемых решений, что требует строгого количественного анализа имеющихся данных. При работе с большими объемами накапливаемой информации необходимо постоянно оперативно отслеживать динамику рынка, а это практически невозможно без автоматизации аналитической деятельности.

В бизнес-приложениях большой интерес представляет интеграция методов интеллектуального анализа данных с технологией оперативной аналитической обработки данных OLAP (On-Line Analytical Processing). OLAP использует многомерное представление агрегированных данных для быстрого доступа к важной информации и дальнейшего ее анализа. Системы OLAP обеспечивают аналитикам и руководителям фирм быстрый последовательный интерактивный доступ к внутренней структуре данных и возможность преобразования исходных данных для отражения структуры

системы нужным для пользователя способом. Кроме того, OLAP-системы позволяют просматривать данные и выявлять имеющиеся в них закономерности или визуально, или простейшими методами типа линейной регрессии, а включение в их арсенал нейросетевых методов обеспечивает существенное расширение аналитических возможностей. Термин OLAP ввел Э. Кодд в 1993 году. В своей работе он рассмотрел недостатки реляционной модели, в первую очередь невозможность «объединять, просматривать и анализировать данные с точки зрения множественности измерений, то есть самым понятным для корпоративных аналитиков способом» и определил общие требования к системам OLAP, расширяющим функциональность реляционных СУБД и включающим многомерный анализ как одну из своих характеристик.

Актуальным в развитии систем управления базами данных является отказ от нормализации отношений. Нормализация отношений нарушает естественные иерархические связи между объектами, которые достаточно распространены в нашем мире. Возможность сохранять их на концептуальном уровне позволяет пользователям более естественно отражать смысл предметной области. В настоящее время уже существует теоретическое обоснование работы с ненормализованными отношениями и практические реализации подобных систем.

Дальнейшим развитием СУБД следует отметить объектно-ориентированные базы данных, в которых предметная область моделируется как множество классов взаимодействующих объектов. Каждый объект характеризуется набором свойств, которые являются как бы его пассивными характеристиками и набором методов работы с этим объектом. Можно считать, что наиболее важным качеством объектно-ориентированного подхода к созданию базы данных является учет поведенческого аспекта объектов. В среде объектно-ориентированных баз данных проектирование, разработка и сопровождение системы становятся процессами, в которых интегрируются структурный и поведенческий аспекты. Конечно, для этого нужны специальные языки, позволяющие определять объекты и создавать на их основе прикладную систему.

Очень важным направлением развития баз данных является перспектива взаимодействия Web-технологий и баз данных. Простота и доступность Web-технологий, возможность свободной публикации информации в Интернете завоевали признание у большого числа пользователей. Однако процесс накопления слабоструктурированной информации быстро проходит и далее наступает момент необходимости разработки эффективного управления этой разнообразной информацией, что является серьезной задачей. Авторы Web-публикаций нуждаются в инструментах для быстрого и экономичного построения хранилищ данных, рассчитанных на сложные приложения. Это, в свою очередь, формирует требования к технологии баз данных для создания, управления, поиска и обеспечения безопасности содержимого Web-сайтов.

ПРИЛОЖЕНИЕ

Приведены часто используемые в литературных источниках термины и понятия при изучении баз данных и СУБД. Приведенные термины будут полезными для начального изучения данной дисциплины. Это также связано еще и с тем, что терминология в данной области часто использует англоязычную лексику, что затрудняет изучение литературы и усложняет понимание проблем.

Аппаратный сервер - специализированное устройство со встроенным программным обеспечением, определяющим специализацию и предоставляемый сервис. Аппаратные серверы, как правило, более просты и надежны в эксплуатации, потребляют меньше электроэнергии.

Атрибут- это свойство, которое описывает некоторый аспект объекта.

База данных (database) - совокупность структурированных данных, относящихся к определенной предметной области.

Внешний ключ - ключевой элемент подчиненной (внешней) таблицы, значение которого совпадает со значением первичного ключа главной (родительской) таблицы.

Запрос- процесс обращения пользователя к БД с целью ввода, получения или изменения информации в БД.

Логическая структура БД- определение БД на физически независимом уровне, ближе всего соответствует концептуальной модели БД.

Объект (object) - элемент информационной системы, обладающий определенными свойствами и определенным образом реагирующий на внешние события.

Отношение(relation) - в теории реляционных баз данных синоним таблицы, в которой строка называется кортежем, а столбец называется атрибутом.

Первичный ключ - минимальный набор атрибутов сущности, однозначно идентифицирующий экземпляр сущности. В реляционной теории первичный ключ - минимальный набор атрибутов, однозначно идентифицирующий кортеж.

Первичный ключ (primary key) - главный ключевой элемент, однозначно идентифицирующий запись в таблице. Могут также существовать альтернативный и уникальный ключи, служащие также для идентификации записей в таблице.

Пользователь БД – человек или программа, использующие БД.

Распределенный запрос- запрос, при обработке которого используются данные из БД, расположенные в разных узлах сети.

Реляционная БД - основной тип современных баз данных. Состоит из таблиц, между которыми могут существовать связи по ключевым значениям.

Репликация базы данных - создание копий базы данных (реплик), которые могут обмениваться обновляемыми данными или реплицированными

формами, отчетами или другими объектами в результате выполнения процесса синхронизации.

Связь (relation) - функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (родительской) таблице - первичный, второй - внешний ключ - во внешней (дочерней) таблице.

Сервер БД - программное обеспечение, обслуживающее базу данных и отвечающее за целостность и сохранность данных, а также обеспечивающее операции ввода-вывода при доступе клиента к информации.

Сервер удаленного доступа - программа, обеспечивающая пользователя доступом к удаленной системе с терминала через соответствующую клиентскую программу. Стандартный сервер удаленного доступа к графическому интерфейсу MS Windows называется терминальным сервером.

Сервер-компьютер - компьютер, выполняющий только серверные задачи, или компьютер, специализированный для использования в качестве аппаратного средства для программных серверов.

Система - совокупность взаимодействующих между собой и с внешним окружением объектов.

Система управления базами данных (СУБД) - комплекс программных и языковых средств, необходимых для создания и модификации базы данных, добавления, удаления, поиска информации, отображения информации на экране и в печатном виде, разграничения прав доступа к информации, выполнения других операций с базой данных.

Ссылочная целостность данных - набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

Сущность- это отдельный объект деятельности организации (сотрудник, вещь, понятие и т.п.), который должен быть представлен в базе данных.

Таблица базы данных (table) - регулярная структура, которая состоит из однотипных строк-записей (records), разбитых на столбцы-поля (fields).

Транзакция - совокупность логически взаимосвязанных запросов, направленных на согласованное изменение некоторого множества записей в одной или нескольких таблицах БД.

Файл-сервер - программный сервер для обеспечения доступа к файлам на диске сервера. Прежде всего, это серверы передачи файлов по запросам по определенным протоколам.

Хранимые процедуры - программные модули, сохраняемые в базе данных для выполнения определенных операций с информацией базы.

Язык SQL (Structured Query Language) - универсальный язык работы с базами данных, включающий возможности ее создания, модификации структуры, отбора данных по запросам, модификации информации в базе и прочие операции.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

- Грошев А.С. Основы работы с БД. - М.: ИНТУИТ, 2008.
- Карпова Т.С. Базы данных. Модели, разработка, реализация. – М.: ИНТУИТ, 2008.
- Коваленко Т.А. , Сирант О.В. Работа с базами данных. – М.: ИНТУИТ, 2012.
- Кузнецов С.Д. Основы современных баз данных. – М.: Центр ИТ, 2009.
- Хайдаров К.А. Введение в системы управления базами данных: учеб. пособие/ Казахский университет Алатау, Алатау, 2004.
- Швецов В.И. Базы данных. – М.: ИНТУИТ, 2009.

ОГЛАВЛЕНИЕ

Введение	3
1.Основные понятия баз данных	4
2.Логические модели баз данных	10
2.1. Иерархическая модель данных.....	12
2.2.Сетевая модель данных.....	14
2.3. Объектно-ориентированная модель данных.....	15
2.4.Реляционная модель данных.....	16
3. Основные этапы разработки баз данных	19
3.1. Концептуальное проектирование базы данных.....	20
3.2. Логическое проектирование базы данных.....	20
3.3. Нормализация базы данных.....	21
3.3.1. Первая нормальная форма.....	22
3.3.2. Вторая нормальная форма.....	22
3.3.3. Третья нормальная форма.....	23
3.4.Типы связей и свойства отношений.....	24
3.5. Программная реализация базы данных.....	25
3.5.1. Разработка приложений.....	26
3.5.2.Тестирование базы данных.....	26
3.6. Эксплуатация и сопровождение базы данных.....	27
4. Организация и применение СУБД Access	28
4.1. Основные понятия СУБД Access.....	28
4.2. Объекты СУБД Access.....	29
4.3. Создание объектов СУБД Access.....	33
4.3.1.Создание таблиц.....	33
4.3.2.Создание запросов.....	36
4.3.3.Создание форм.....	42
4.3.4.Создание и печать отчетов.....	49
4.4.Основные этапы разработки базы данных.....	52
4.5. Разработка базы данных средствами СУБД Access.....	53
4.6. Перспективы развития баз данных.....	70
Приложение	72
Библиографический список	75