О.А. Ильичева, П.В. Федотов

БАЗЫ ДАННЫХ.
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ И ЛАБОРАТОРНЫЙ ПРАКТИКУМ
В СУБД MS ACCESS

Ростов-на-Дону

2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«РОСТОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

О.А. Ильичева, П.В. Федотов

БАЗЫ ДАННЫХ.
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ И ЛАБОРАТОРНЫЙ ПРАКТИКУМ
В СУБД MS ACCESS

Допущено учебно-методическим советом РГСУ в качестве учебного пособия для обучающихся по направлению подготовки 09.03.03 «Прикладная информатика»

Ростов-на-Дону
2015

УДК 004.4
И 46

Рецензент: А.Н. Литвиненко, доцент кафедры
«Информатика и вычислительный эксперимент»
факультета математики, механики и компьютерных наук
Южного федерального университета

Ильичева О.А., Федотов П.В.
И 46 Базы данных. Теоретические основы и лабораторный практикум в СУБД MS Access: учебное пособие. – Ростов-н/Д: Рост. гос. строит. ун-т, 2015. – 176 с.

В пособии представлены основные концепции реляционных баз данных. Обсуждаются понятия и свойства реляционной модели данных, стратегии целостности, нормальные формы, реляционная алгебра. В лабораторном практикуме рассматриваются возможности и инструменты пользовательской работы разного уровня с реляционной базой данных: создание простых и сложных запросов, форм. Для проведения лабораторных работ используется реляционная СУБД MS Access 2013. Пособие содержит множество примеров с комментариями и задания по ключевым темам.

Предназначено для обучающихся по направлению подготовки 09.03.03 «Прикладная информатика».

Данное учебное пособие издано при непосредственной помощи и поддержке Ростовского государственного строительного университета в рамках гранта 2014 г.

УДК 004.4

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Базы данных являются одним из главных достижений информатики, основой современных информационных систем. Многочисленные исследования в этой области за последние десятилетия привели к созданию весьма мощных, понятных, многообразных систем, обслуживающих как небольшие фирмы, так и гигантские корпорации. Базы данных непрерывно эволюционируют, от простых хранилищ числовой и текстовой информации до многомерных и аналитических систем, включающих мультимедийные данные разных типов, размеров и назначения.

Не будет преувеличением сказать, что базы данных проникают во все сферы человеческой деятельности: в море знаний, обрушившихся на современного человека, невозможно разобраться без тщательного учета, классификации и сохранения информации. Эту функцию берут на себя базы данных. Администратор баз данных – одна из наиболее востребованных профессий, в которой нуждаются как столпы мировой индустрии, так и приюты, библиотеки, и даже монастыри.

Совершенствование теории и практики в области баз данных, увеличение возможностей систем управления базами данных сделало их доступными широкому кругу пользователей и разработчиков. К сожалению, в результате этой доступности к созданию баз данных устремились люди, не обладающие достаточными знаниями о методах проектирования эффективно и корректно работающих систем. Трудно переоценить ущерб, наносимый полученными в результате этой деятельности «продуктами». Неверные ответы на неточные вопросы к кое-как «состряпанной» базе могут привести не только к «потерянным миллионам», «зверским налогам», но и к личным трагедиям.

Предлагаемое пособие является введением в теорию и практику реляционных баз данных, завоевавших в последние годы лидирующие позиции в области информационных систем. Очевидно, в рамках одного небольшого

пособия невозможно изложить даже минимум знаний для профессионального разработчика; пособие предназначено для тех, кто делает в этой области первые шаги. Данная публикация – первая часть общего курса «Базы данных» - содержит необходимые теоретические сведения и комплекс лабораторных работ в среде MS Access-2013.

В пособии использованы определения и некоторые примеры, взятые из книг одного из основоположников этой отрасли знаний – К. Дж. Дейта [1], а также из известной книги Т. Конноли, К. Бегг, А. Страчан [2]. Эти книги являются в определенном смысле энциклопедией баз данных и поэтому рекомендуются в первую очередь для всех, кто всерьез намерен изучить этот предмет.

Ссылки на эти источники, а также другие, в том числе и опубликованные в Интернете, содержатся в библиографическом списке, представленном в конце пособия.

1. ФУНКЦИИ СУБД

Исторически развитие вычислительной техники привело к образованию двух областей ее применения: численные расчеты (программирование, языки для записи алгоритмов) и информационные системы (хранение данных и их преобразование в соответствии с конкретным приложением). Современное «нашествие» информационных систем началось с перерождения файловых систем в базы данных (БД) и далее, в базы знаний, «интеллектуальные» БД, использующие средства логического вывода и анализа хранимой информации. Во всем многообразии систем, организующих хранение данных, наиболее востребованными остаются пока файловые системы и базы данных. Рассмотрим некоторые понятия и отличия этих систем.

Файловые системы разрабатывались как средство централизованного управления файлами. С точки зрения прикладной программы ***Файл*** – это именованная область внешней памяти, в которую можно записывать и из

которой можно считывать данные. Правила именования файлов, способ доступа к данным, их структура зависят от конкретной системы управления файлами (СУФ). Основные функции современных СУФ:

- распределение внешней памяти;
- отображение имен файлов в адреса;
- обеспечение доступа к данным;
- поддержка защиты файлов от несанкционированного доступа к данным.

На нижнем, базовом, уровне СУФ делят файл на логические блоки, размер которых обычно равен размеру страницы виртуальной памяти, поддерживаемой аппаратурой компьютера совместно с операционной системой. На уровне пользователя файл рассматривается как набор записей - байтовая последовательность постоянного или переменного размера.

Для обеспечения доступа к данным СУФ может поддерживать невидимые пользователю служебные структуры, позволяющие реализовать хеширование, B-деревья и другие средства быстрого доступа. На уровне пользователя практически все СУФ поддерживают многоуровневое именование файлов: путь с указанием имен вложенных друг в друга каталогов.

Для защиты информации при каждом файле может храниться информация о правах пользователя (пароль, приоритет и т.п.). Многие СУФ разрешают многопользовательский («параллельный») доступ к файлу, используя для синхронизации режим блокировок: действия одного пользователя блокируются, пока другой не закроет требуемый файл.

Файловые системы удобны для хранения слабо структурированной информации, например, текстовых данных, входов и выходов компиляторов, редакторов связей – компоновщиков пакетов прикладных программ. Однако если данные сложно устроены и связаны между собой, файловые системы малопригодны. Они не следят за согласованностью информации в файлах, дублированием данных, не имеют языков манипулирования данными, не

обеспечивают восстановление информации после сбоев.

Согласованность данных – ключевое свойство БД, позволяющее следить за непротиворечивостью хранимой информации и отвечать на запросы пользователей. Например, любая система управления базами данных (СУБД) с легкостью ответит на запрос, непосильный для СУФ: «выдать общую численность отдела, в котором работает Иван Петрович Сидоров».

Понятие «База данных» может быть определено как структурированный набор связанных, согласованных между собой, постоянных («перманентных») данных. Работу с ним осуществляет специальный комплекс программ – **СУБД**, которая реализует следующие **основные функции** [3]:

- непосредственное управление данными во внешней памяти (поддержка собственных системы именования, структур внешней памяти для хранения данных и убыстрения доступа к ним – индексов);
- управление буферами оперативной памяти (собственный набор буферов в оперативной памяти и собственная дисциплина их замены);
- управление *транзакциями* (транзакция – это последовательность операций над БД, рассматриваемая системой как единое целое, управление ими – механизм поддержки непротиворечивости и согласованности информации при параллельной работе нескольких пользователей, при восстановлении информации после сбоя);
- журнализация операций и восстановление после сбоев (ведение журнала изменений в БД: запись об изменении данных должна попасть в журнал раньше, чем измененный объект попадет во внешнюю память; это средство позволяет восстановить утраченную при сбое информацию);
- поддержка языков БД (к ним относятся *язык манипулирования данными* – язык запросов и *язык определения данных* – язык описания данных, их связей, схемы БД);
- поддержка целостности данных, обеспечивающая непротиворечивое состояние хранимых данных;

▪ обеспечение безопасности, предотвращающее несанкционированный доступ к БД со стороны пользователей;

▪ поддержка механизма *представлений* (view), позволяющего любому пользователю иметь свой собственный «взгляд» на базу данных.

2. АРХИТЕКТУРА СИСТЕМЫ БАЗ ДАННЫХ

В семидесятых годах прошлого столетия Комитетом планирования стандартов и норм SPARC (Standards Planning and Requirements Committee) Национального института Стандартизации США (American National Standard Institute – ANSI) была предложена трехуровневая модель архитектуры систем баз данных, соответствующая трем уровням абстракции в описании элементов данных: *внешний уровень* (пользовательский), *концептуальный* (логический) и *внутренний* (физический) (рис.1). Назначение такой архитектуры – отделение пользовательского представления базы данных от ее физического представления. Модель ANSI/SPARC отвечает следующим целям:

▪ каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя собственное представление о них. Он может изменять это представление, причем это изменение не должно оказывать влияния на других пользователей;

▪ пользователи не должны непосредственно иметь дело с подробностями физического хранения данных в базе, их взаимодействие с базой не должно зависеть от особенностей хранения в ней данных;

▪ администратор БД должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские представления;

▪ администратор БД должен иметь возможность изменять концептуальную или глобальную структуру базы данных без какого-либо влияния на всех пользователей;

▪ внутренняя структура БД не должна зависеть от таких изменений физических аспектов хранения данных, как переключение на новое устройство хранения.

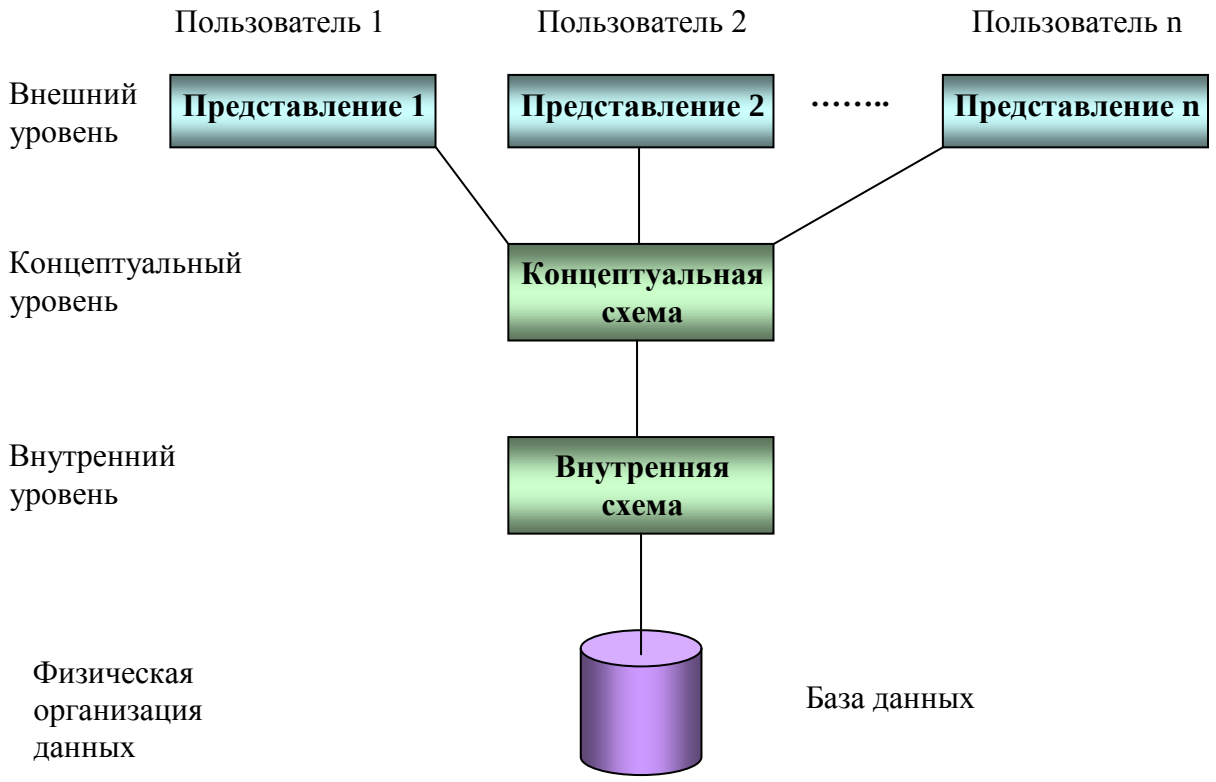


Рис.1. Трехуровневая архитектура ANSI / SPARC

Уровень, на котором воспринимают данные пользователи, называется *внешним*, тогда как СУБД и операционная система воспринимают данные на *внутреннем* уровне. Концептуальный уровень представления данных предназначен для отображения внешнего уровня на внутренний и обеспечения необходимой их независимости друг от друга.

Внешний уровень: представление БД с точки зрения пользователей. Описывает ту часть базы данных, которая относится к каждому пользователю.

Этот уровень состоит из нескольких различных внешних представлений базы данных. Внешнее представление содержит только те сущности, атрибуты и связи реального мира, которые интересны конкретному пользователю. Другие сущности, атрибуты или связи, которые ему неинтересны, также могут быть

представлены в базе данных, но пользователь может даже и не подозревать об их существовании. Например, отдел кадров, службу организации досуга, бухгалтерию организации могут интересовать совершенно разные аспекты информации о сотрудниках.

Помимо этого, различные представления могут по-разному отображать одни и те же данные, в разных форматах. Какие-то представления могут включать в себя производные или вычисляемые данные, которые не хранятся в базе данных, а создаются временно, по мере надобности.

Концептуальный уровень: обобщающее представление БД. Описывает, какие данные хранятся в базе данных, а также связи, существующие между ними.

Этот уровень содержит логическую структуру всей базы данных, это полное представление требований к данным со стороны организации, не зависящее от возможных способов их хранения. На нем определены компоненты:

- все сущности, их атрибуты и связи;
- накладываемые на данные ограничения;
- семантическая информация о данных;
- информация о мерах обеспечения безопасности и поддержки целостности данных.

Концептуальный уровень поддерживает каждое внешнее представление: любые доступные пользователю данные должны содержаться (или вычисляться) на этом уровне. Однако он не содержит никаких сведений о методах хранения данных. Например, описание сущности должно включать описание типов данных, но не их объем в байтах.

Внутренний уровень: физическое представление БД в компьютере. Этот уровень описывает как информация хранится в базе данных.

Этот уровень предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового

пространства. На этом уровне СУБД взаимодействует с операционной системой, создаются индексы (специальные структуры для убыстрения доступа к данным). Внутренний уровень содержит информацию:

- распределение дискового пространства для хранения данных и индексов;
- описание подробностей сохранения записей с указанием реальных размеров элементов данных;
- сведения о размещении записей: сведения о сжатии данных и методах их шифрования.

Ниже внутреннего находится **физический уровень**, который контролируется операционной системой, но под руководством СУБД.

На каждом уровне архитектуры БД данные описываются в рамках определенной модели данных.

Модель данных: интегрированный набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные (структурная часть), типе допустимых операций с данными – обновлении, извлечении данных, изменении структуры БД (управляющая часть).

Структурная часть определяет структуру данных, а управляющая – поведение данных. Модель данных – это абстрактное, логическое определение объектов и операторов.

Применительно к моделям данных используется другая терминология трехуровневого представления БД, соответствующая этапам проектирования базы данных.

Описание предметной области, выполненное с использованием естественного языка, математических формул, таблиц, графиков, диаграмм «сущность-связь», называют **инфологической моделью данных.**

Описание данных на языке конкретной выбранной СУБД называют **даталогической моделью данных**. В этой модели выбирается тип СУБД – иерархическая, сетевая, реляционная, объектная.

Описание хранимых данных в их компьютерном представлении называют **физической моделью данных**.

При создании БД этому представлению соответствуют этапы концептуального, логического и физического проектирования базы данных.

За управлением базой данных и самой СУБД как за корпоративным ресурсом отвечает **администратор БД** – человек или группа лиц. В ведение администратора входят:

- определение информационного содержания БД (что нужно хранить для данного предприятия, концептуальная схема данных);
- определение структуры хранения и стратегии доступа (логическая схема данных);
- взаимодействие с пользователями (внешняя схема);
- определение контроля полномочий и процедур проверки их достоверности;
- управление эффективностью и реакцией на изменение требований.

3. ТИПЫ БАЗ ДАННЫХ

Соответственно используемой модели данных системы баз данных разделяют на категории, отражающие эволюцию СУБД: иерархические системы, сетевые, реляционные, объектно-ориентированные, объектно-реляционные. Наибольшее развитие и применение получили реляционные системы, для которых определены международные стандарты описания данных и операторов манипулирования данными.

Иерархическая БД состоит из упорядоченного набора древовидных структур (иерархий); операции работы с такими структурами включают перемещение по иерархическим указателям вверх и вниз по ветвям деревьев. Пример схемы иерархической БД (рис. 2):

Рис. 2. Иерархическая структура данных

Экземпляр такого дерева представлен на рис. 3.

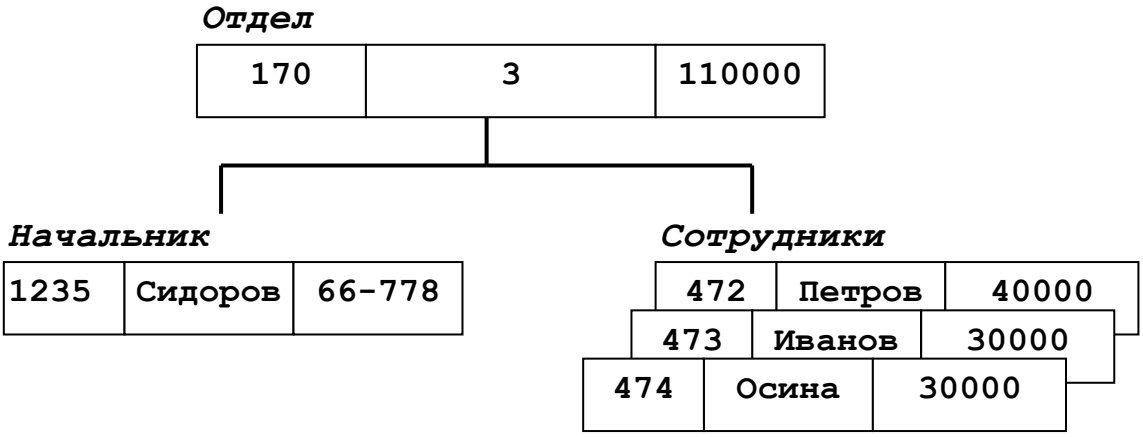


Рис. 3. Экземпляр иерархической структуры

Достоинства иерархических систем – простая структура связей, известные и эффективные алгоритмы обхода и формирования деревьев.

Недостатки – потери производительности и сложность разработки, если информационные зависимости между данными не имеют иерархической структуры. Непросто бывает «затолкать» в иерархию даже довольно очевидные связи; попробуйте, например, представить, не дублируя информацию, отношение между объектами *Поликлиника*, *Врачи*, *Пациенты*, учитывая, что каждый пациент может лечиться у разных врачей (и у одного по разным поводам), и каждый врач может обслуживать множество пациентов.

Необходимость представления более широкого спектра связей между данными привела к появлению сетевых систем баз данных.

Сетевая БД состоит из набора записей и набора связей между этими

записями. При разработке сетевой модели данных определяются типы записей и типы связей между ними.

Пример сетевой схемы БД дан на рис.4.

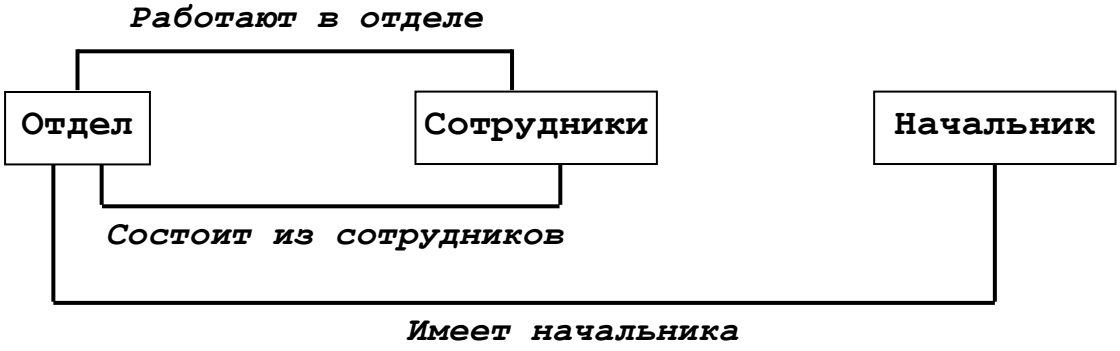


Рис. 4. Сетевая структура данных

Достоинство сетевых систем – возможность отразить в модели данных многообразие связей между ними, создать более адекватное представление о предметной области, для которой и создана БД.

Недостатки – тяжелая и для реализации, и для понимания разработчиком навигация, как при внесении, так и при извлечении данных. Разработчику-программисту необходимо, как правило, изучить несколько внутренних языков базы данных, детально представлять логическую структуру конкретной БД. Поиск в такой базе может быть настолько сложен, что оперативная работа с данными практически невозможна. По словам одного из разработчиков ОС UNIX «сетевая база – это самый верный способ потерять данные».

Реляционная БД – состоит из совокупности связанных между собой «плоских» таблиц. Данные хранятся в ячейках таблиц, термин «плоская» подчеркивает, что значения в ячейках не могут быть структурированными, составными. Например, текстовые значения рассматриваются как единое целое – «часть» значения не может быть извлечена. Все операции над базой данных сводятся к манипуляциям с таблицами. Пример реляционной схемы показан на рис. 5, таблицы представлены именами столбцов.

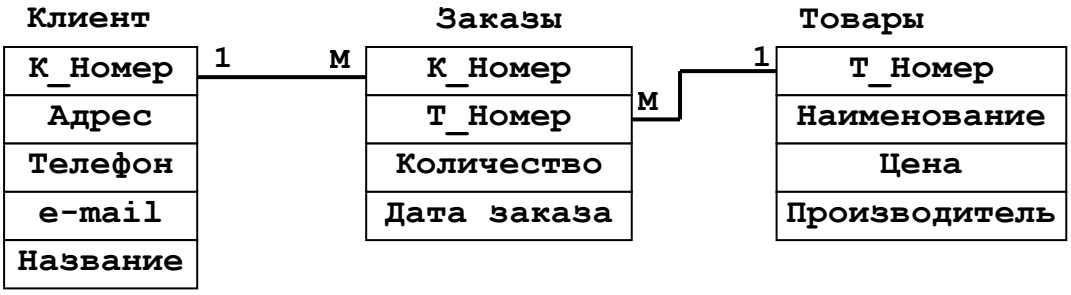


Рис. 5. Реляционная структура данных

Обозначения в связи **1** и **М** указывают, что одна (каждая) строка одной таблицы информационно связана, возможно, с несколькими (многими) строками в другой таблице. Например, один и тот же клиент может сделать несколько заказов. Таблица обычно представляет множество однотипных объектов, «сущность» предметной области, строка таблицы – экземпляр сущности. Таблица может содержать тысячи и сотни тысяч экземпляров.

Достоинства реляционных систем – простота структуры данных и связей, достаточно эффективные алгоритмы навигации по данным, развитая математическая база, позволяющая формально проверять корректность преобразований БД, операций над таблицами.

Недостатки реляционного подхода проявляются в приложениях, характеризуемых большим количеством различных сущностей, каждая из которых обладает небольшим количеством экземпляров. К таким предметным областям относятся автоматизированное проектирование и производство, мультимедийные, геоинформационные системы, медицинские приложения. Такие системы могут содержать миллионы разнотипных элементов с большим количеством разнообразных связей и требовать поддержки координируемой, возможно, долгосрочной коллективной разработки. Однородная структура данных в реляционном подходе не позволяет представить сложно устроенные объекты иначе, чем через множество малоэффективных соединений, связанных с добавлением «лишних» таблиц, моделирующих зависимости элементов. К

недостаткам относится и слабая поддержка целостности: подход обеспечивает, в основном, ссылочную целостность (отсутствие ссылок на несуществующие объекты), для обеспечения же семантической целостности, пользовательских ограничений модель не предоставляет адекватных и эффективных средств. Практически не поддерживаются долговременные и распределенные транзакции, характерные для выполнения сложных проектов.

Объектно-ориентированная (объектная) БД – система, построенная на принципах объектно-ориентированной технологии программирования, завоевавшей лидирующие позиции в разработке современного программного обеспечения.

В объектной модели мир представляется совокупностью взаимодействующих объектов. Каждый *объект* принадлежит некоторому классу (типу данных). *Класс* определяется как множество объектов, имеющих одинаковые структуру (свойства) и поведение. Поведение описывается совокупностью процедур («*методов*», операций), которые можно применять к объекту. Взаимодействие объектов реализуется *сообщениями* между ними - вызовами таких процедур. Классы и, соответственно, объекты могут образовывать иерархию, в которой «младшие чины» - подклассы, помимо наличия собственных, *наследуют* свойства и поведение «старших» по иерархии классов. Классы являются также механизмом *инкапсуляции* объектов, позволяющей разделить их полномочия и скрыть от пользователя реализацию – внутреннюю, физическую структуру объектов и программы методов. При манипуляциях с инкапсулированными объектами доступен только интерфейс: имена и типы объектов, описания операций (имена операций, типы их параметров и тип результата).

Определение данных в объектной БД включает в себя определения классов – свойств (атрибутов) объектов и методов работы с ними. Методы могут реализовывать и пользовательские ограничения, а атрибуты – иметь тип, являющийся ссылкой на объекты других классов. Заполнение БД состоит в

порождении и сохранении объектов (экземпляров) этих классов. Создание объекта сопровождается присвоением ему уникального внутреннего имени – «объектного идентификатора» (OID), который в системе преобразуется в указатель на объект в оперативной памяти. Так как одни объекты могу ссылаться на другие объекты с помощью идентификаторов, то один объект может совместно использоваться несколькими другими объектами. Навигация по базе по запросу осуществляется с использованием ссылок между объектами и допустимых методов работы с объектами.

Объектные базы данных представляют собой пока еще «сырой продукт», нуждающийся, прежде всего, в концептуальном развитии.

Достоинства объектных систем – широкие возможности в определении того, что будет храниться в базе, т.е. любых структур данных, которые можно создать с помощью обычного языка программирования.

Недостатки вытекают из этой «вседозволенности». Постулат «в объектной БД можно сохранять все» оборачивается невозможностью автоматизировать управление и стандартизовать структуру базы и операции манипулирования данными, стратегии поддержки целостности – все ложится на плечи программиста конкретной БД. Такие системы не обеспечивают переносимости СУБД. В отличие от реляционных СУБД, после установки полностью готовых к созданию конкретных приложений, объектные предоставляют «лишь некоторый набор средств построения СУБД», подлежащий порой весьма трудоемкой настройке квалифицированными специалистами, определяющими необходимые классы, методы, процедуры проверки целостности и т.п. Реляционные системы проводят четкую границу между логическим и физическим представлением данных, объектные этого не делают, и поэтому обеспечивают значительно меньшую независимость данных.

Желание совместить хорошо отлаженный механизм функционирования реляционных систем с широкими возможностями определения данных в объектных системах привело к созданию объектно-реляционных БД.

Объектно-реляционная БД является «конгломератом» реляционных и объектных возможностей, концепция и языки таких баз только развиваются. Такие БД представляются, как и реляционные, совокупностью связанных таблиц, однако в определении данных – содержания таблиц – могут присутствовать объекты – составные структуры (например, подтаблицы), являющиеся экземплярами классов (пользовательских типов данных) с возможностью наследования, инкапсуляции и других объектных преимуществ.

Пробные версии подобных систем подвергаются серьезной критике из-за концептуальных противоречий, приводящих порой к непредсказуемости результатов запросов [1].

4. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Реляционную модель данных предложил Э. Кодд (IBM) в 1969-70 гг., и с тех пор она получила мощное развитие и применение в разработке реляционных баз данных. Представление и обработка данных в этой модели поддерживается аппаратом теории множеств и математической логики. Сегодня реляционные СУБД стали фактическим промышленным стандартом.

Реляционная модель данных состоит из трех частей, соответствующих трем аспектам реляционного подхода: структурной части, манипуляционной части и целостной части.

В структурной части фиксируется, что *единственной структурой данных любой реляционной БД является нормализованное n-арное отношение*. Такое отношение представлено в БД двумерной таблицей, состоящей из n именованных столбцов.

Механизмом *манипулирования данными являются реляционная алгебра и реляционное исчисление*. Операции этой алгебры и логические формулы исчисления описывают манипуляции с таблицами. Ответ на запрос информации из БД или изменение данных могут быть представлены как

результат реляционных операций над таблицами, также являющийся таблицей.

Целостная часть направлена на поддержку непротиворечивости и не избыточности хранимой информации и включает в себя два базовых требования: *целостность сущностей и целостность по ссылкам*. Связь таблиц в базе должна обеспечивать доступ к хранимым данным; операции над данными не должны приводить к возникновению ссылок на несуществующие объекты или к получению неоднозначных результатов.

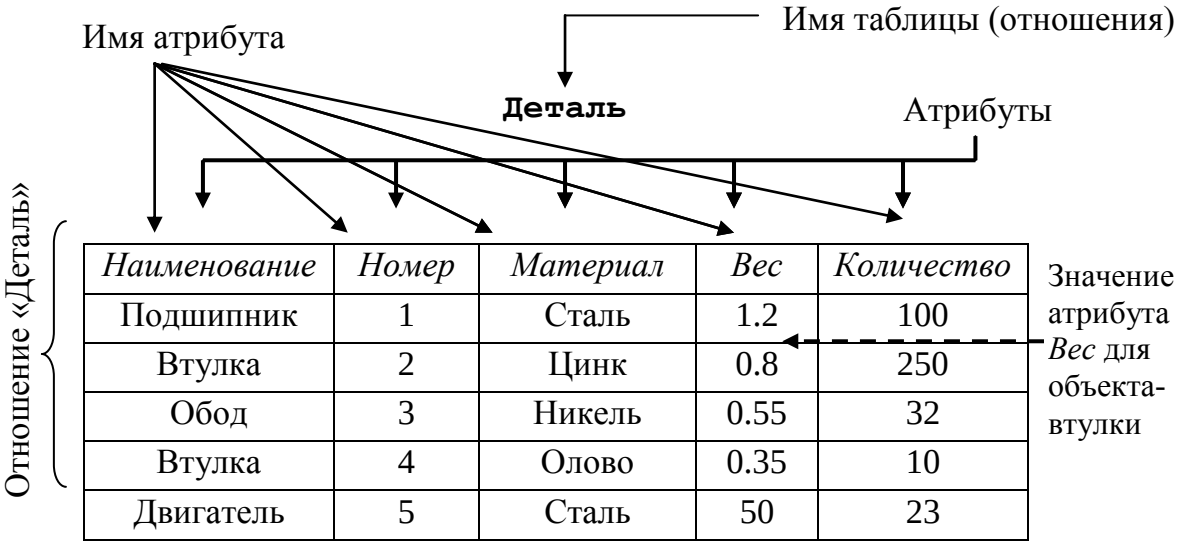
Таким образом, *реляционной считается такая база данных, в которой все данные представлены для пользователя в виде «плоских» таблиц значений данных, и все операции над базой данных сводятся к манипуляциям с таблицами*.

Основные понятия реляционной БД:

- таблица (отношение);
- атрибут;
- домен;
- кортеж;
- первичный ключ;
- внешний ключ.

Таблица состоит из строк и столбцов и имеет имя, уникальное внутри базы данных. **Таблица** отражает *тип* объекта реального мира (*сущность*), а каждая ее строка - конкретный объект (экземпляр сущности).

Каждый столбец таблицы имеет имя, уникальное внутри таблицы, и отражает определенное свойство (**атрибут**) сущности; значение в столбце является значением этого атрибута для объекта-строки (рис. 6).



Наименование	*Номер*	*Материал*	*Вес*	*Количество*
Подшипник	1	Сталь	1.2	100
Втулка	2	Цинк	0.8	250
Обод	3	Никель	0.55	32
Втулка	4	Олово	0.35	10
Двигатель	5	Сталь	50	23

Рис. 6. Реляционная таблица «Деталь»

Каждый столбец таблицы – это совокупность значений конкретного атрибута объекта. Например, значения атрибута *Материал* - это Сталь, Цинк, … , значения атрибута *Количество* – это 100, 250, … .

Атрибут называют также **столбцом** таблицы, **полем** таблицы. Значения любого атрибута выбираются из определенного множества значений – домена.

Домен атрибута – допустимое множество значений атрибута, его область определения. Домен может рассматриваться как некоторый подтип (подмножество) типа данных атрибута. **Тип данных** определяется как множество значений и набор возможных операций над этими значениями. Так, например, тип данных атрибутов *Материал* и *Наименование* – тексты, атрибута *Вес* – действительные числа, атрибута *Количество* – целые числа. Домены могут «вырезать» из этих типов подмножества, удовлетворяющие некоторым дополнительным свойствам. Например, *Количество* должно быть не меньше 2 и не больше 1000 (для данной организации), *Материалом* не могут служить текстовые значения «колбаса» или «Мария Ивановна», *Вес* не может превышать 10 тонн и т.п.

☑ Домен – это не то, что *сейчас* находится в столбце таблицы, а то, что там *может быть* в принципе. Имя атрибута – это, фактически, имя домена.

Кортеж – это строка таблицы, определяющая конкретный объект (в примере – конкретную деталь). Например, <Обод, 3, Никель, 0.55, 32>. Кортеж называют также **записью** в таблице.

☑ С формальной точки зрения кортеж (как и запись) представляется как совокупность пар <*имя атрибута, значение атрибута*>, т.е., например, как {<Наименование, Обод>, <Номер, 3>, <Материал, Никель>, <Вес, 0.55>, <Количество, 32>}. В этом случае достигается соответствие с математическим понятием отношения, элементы которого не переставляются, т.е. кортеж упорядочен.

Степень (арность) отношения – количество его атрибутов (число столбцов таблицы).

Мощность отношения – количество его кортежей (число строк таблицы, исключая заголовки столбцов).

Схема (заголовок) отношения – именованное множество (A_1, A_2, …, A_n) имен атрибутов, которому соответствует множество определяющих их доменов (D_1, D_2, … , D_n).

Тело отношения – множество его кортежей (меняющих во времени свои значения).

Свойства отношений:

- Отсутствие кортежей–дубликатов: в таблице не может быть одинаковых строк.
- Отсутствие требования упорядоченности кортежей: строки могут располагаться в любом порядке, отношение останется тем же самым.
- Отсутствие требования упорядоченности атрибутов: столбцы (вместе с заголовками) могут располагаться в любом порядке, отношение останется тем же самым.
- Отсутствие в одном отношении одинаковых имен атрибутов. Одно и то же имя атрибута может присутствовать в *разных таблицах*. В таблице могут быть одинаковые столбцы, но они должны иметь *разные заголовки*.

• Атомарность значений атрибутов – на пересечении любой строки и любого столбца должно находиться ровно одно значение. Любое составное значение рассматривается как одно целое, нельзя запросить часть этого значения. Например, атрибут *Адрес* может содержать множество названий – страны, города, улицы, дома и т.п., однако все эти названия воспринимаются системой как *единое* текстовое значение.

Ключи отношения – атрибуты, играющие главную роль при поиске записей в реляционной базе данных; они позволяют сделать выбор записей однозначным. Существует несколько видов ключей, рассмотрим основные.

Из того, что в таблице не должно быть повторяющихся строк, следует возможность *однозначного* выбора строки по какому-либо значению - *ключу*.

Среди всех атрибутов отношения можно выбрать один или несколько, по значениям которого (или которых) однозначно определяется любая запись в таблице. Такие атрибуты называют **ключевыми**. Например, в таблице «Деталь» (рис. 6) – это атрибут *Номер*. Назвав номер детали, вы однозначно выберите всю деталь. Например, номеру 5 соответствует только информация о двигателе. Другой атрибут, например, *Материал*, не подходит для этой роли: указав, например, значение Сталь, мы не сможем определить, какую запись – первую или последнюю выбрать. Очевидно, что для роли ключевого атрибута (атрибутов) подходит тот (те), значения которых *не повторяются* в столбце (в совокупности столбцов). Однако, такое определение не совсем верно. Например, значения атрибута *Вес* в данной таблице также не повторяются, однако нет никакой гарантии, что в следующий момент времени в таблицу не попадет деталь, вес которой будет также, например, 1.2. Поэтому в качестве ключевых можно выбирать атрибуты, значения которых в принципе **не могут повториться**. Такое свойство ключевых атрибутов называют **уникальностью.**

Не всегда может найтись *один* атрибут, однозначно определяющий (*идентифицирующий*) каждую запись в таблице. Выбросим, например, атрибут *Номер* из таблицы «Деталь». Очевидно, что ни один из оставшихся не подойдет

для роли ключа (названия могут повторяться для изделий из разных материалов и разных весов, например, разнообразных двигателей и подшипников может быть много). Попробуем тогда выбрать несколько атрибутов, значения которых *в совокупности* не повторятся. Пара <*Наименование*, *Материал*> не подойдет, т.к., например, подшипник из стали может иметь и другой вес. Например, вполне возможно появление записи вида <Подшипник, Сталь, 0.5, 20>. Назвав тогда в качестве ключевого значение-пару <Подшипник, Сталь> мы не сможем однозначно выбрать запись. А вот тройка атрибутов <*Наименование*, *Материал*, *Вес*> подходит для роли ключа, т.к. в данной таблице повтор этих значений с разным значением *Количества* был бы нелепым: глупо хранить, например, кортежи <Подшипник, Сталь, 1.2, 100> и <Подшипник, Сталь, 1.2, 200>, можно сразу их заменить кортежем <Подшипник, Сталь, 1.2, 300>.

Не всегда может найтись и подмножество атрибутов отношения, обладающее свойством уникальности. Например, выбросим из таблицы «Деталь» атрибуты *Номер* и *Количество*. Тогда никакой атрибут или пара атрибутов из оставшихся не может играть роль ключа из-за возможных повторений значений. Но зато тройка (<*Наименование*, *Материал*, *Вес*>), т.е. **все** атрибуты **вместе** могут рассматриваться как ключ. Действительно, если этот набор в какой-либо записи повторится, то нарушится свойство об отсутствии кортежей-дубликатов.

Таким образом, **в любом отношении ключевой набор атрибутов всегда есть**, в крайнем случае, это все атрибуты отношения.

В таблице может быть несколько претендентов на ключ. Например, любая пара, тройка, четверка и пятерка атрибутов таблицы «Деталь», включающая в свой состав атрибут *Номер*, будет обладать свойством уникальности, за счет уникальности самого номера. В этом случае реальный ключ – это *Номер*, а все остальные атрибуты в таком составном ключе – «нахлебники», хранить и осуществлять поиск записей по такому избыточному ключу означает потерять эффективность - время и память. Поэтому в

реляционных БД при поиске данных рассматриваются в качестве основных ключей атрибуты, обладающие еще и свойством **минимальности**.

Потенциальный (возможный) ключ отношения – атрибут или набор атрибутов, обладающий свойствами *уникальности* и *минимальности*.

Более точно, пусть $\mathcal{K} = \{A_i, A_j, \ldots, A_k\}$ – потенциальный ключ некоторого отношения $\mathcal{R}$.

Уникальность означает, что в произвольный заданный момент времени никакие два различных кортежа в $\mathcal{R}$ не имеют одного и того же значения для совокупности $A_i, A_j, \ldots, A_k$.

Минимальность означает, что ни один из атрибутов $A_i, A_j, \ldots, A_k$ не может быть исключен из $\mathcal{K}$ без потери уникальности.

☑ Любое отношение обладает хотя бы одним возможным ключом, т.к., по крайней мере, совокупность всех его атрибутов может обладать этими свойствами (уникальностью – всегда, а минимальностью – если нет других претендентов).

Потенциальных ключей может быть несколько в одном отношении. Например, если в таблицу «Деталь» добавить столбец *Паспорт детали*, то возможных ключей будет уже два, так как паспорт, как и номер, уникален и минимален. Чтобы выделить один, наилучшим (по какому-либо критерию) образом характеризующий сущность-отношение, определяют понятие *первичного ключа*.

Первичный ключ – один из возможных (потенциальных) ключей.

Выбирается разработчиком базы. Остальные потенциальные ключи называют тогда **альтернативными**. В современных реляционных СУБД и CASE-системах автоматизации проектирования часто в качестве первичного ключа «назначается» порядковый номер (счетчик) записей. Несмотря на то, что это – избыточный столбец, счетчик прост и эффективен в употреблении.

Кроме того, **автоматически нельзя определить ключ** вообще, т.к. это понятие *семантическое* – никто, кроме разработчика не может знать, могут ли повторяться значения в столбце. Например, возможно, что в той предметной области, для которой определено отношение «Деталь», наименования деталей не могут повторяться, такова (вдруг) ее специфика.

Между таблицами (сущностями) в реляционной БД могут существовать информационные связи. Пусть, например, в нашей БД хранится также информация о поставщиках деталей в таблице «Поставщики» (рис. 7).

Поставщики

НомерПоставщика	*Организация*	*Адрес*	*Телефон*
1	Металлург	Томск	2233445
2	Моторстрой	Уфа	6677558
3	Северсталь	Тюмень	9874531

Рис. 7. Таблица «Поставщики»

Чтобы отразить информацию о том, какие детали они поставляют, добавим в таблицу «Деталь» столбец с номером соответствующего поставщика (рис. 8).

Деталь

Наименование	*НомерДетали*	*Материал*	*Вес*	*Количество*	*НомерПоставщика*
Подшипник	1	Сталь	1,2	100	1
Втулка	2	Цинк	0,8	250	1
Обод	3	Никель	0,55	32	2
Втулка	4	Олово	0,35	10	3
Двигатель	5	Сталь	50	23	2

Рис. 8. Расширенная таблица «Деталь»

В базе данных такие таблицы *связываются* по общему полю, здесь – это поле *НомерПоставщика*. Обратите внимание, что в таблице «Поставщики», этот атрибут является *первичным ключом*, в таблице «Деталь» его значения могут повторяться. Таким образом, одному (каждому) значению первичного ключа в таблице «Поставщики» соответствует несколько (много) таких же значений этого атрибута в таблице «Деталь». Эта связь в схеме данных БД

называется связью типа «Один-ко-многим» (**1:М**) и обозначается так (рис. 9):

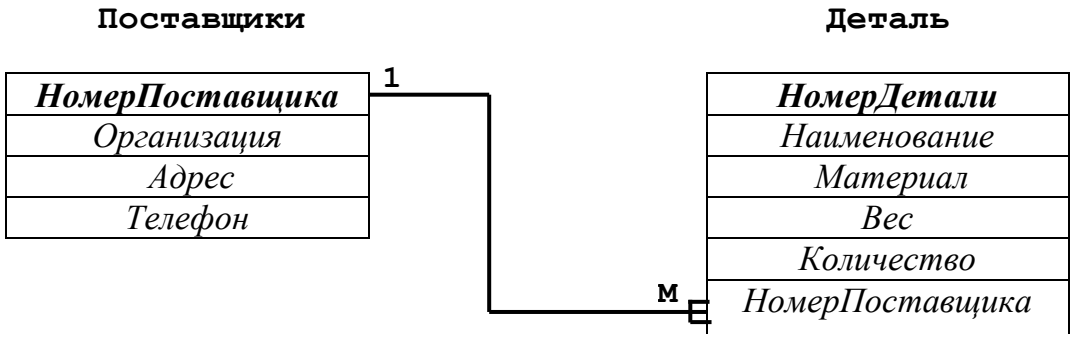


Рис. 9. Связь таблиц в схеме данных БД

В этом случае атрибут *НомерПоставщика* в таблице «Деталь» является **внешним ключом.**

Внешний ключ отношения $\mathcal{R}$ – один или несколько его атрибутов, значения которых обязательно совпадают со значениями первичного или потенциального ключа некоторого другого отношения $\mathcal{R}1$, хранящегося в БД.

В нашей схеме (рис. 10):

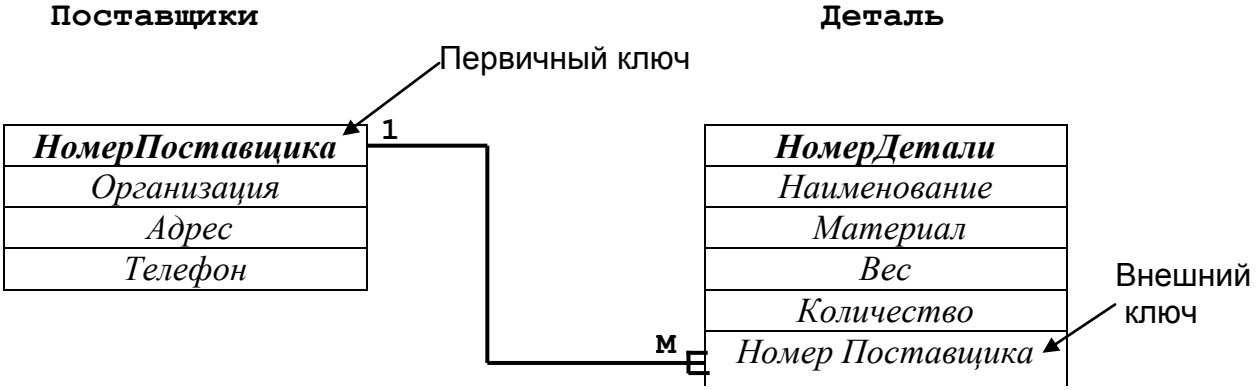


Рис. 10. Связь ключей в реляционной схеме данных

Таблица, которая в связи 1:М выступает на стороне *Один*, называется **родительской**, а таблица, выступающая на стороне *Многие* – **дочерней.**

В приведенной связи таблица «Деталь» является дочерней, а таблица «Поставщики» - родительской. Одна и та же таблица в разных связях может играть разные роли. Например, если в нашей БД имеется таблица «Заказы», в

которой одна и та же деталь присутствует в разных (многих) заказах, то в связи «Деталь» (1) – «Заказы» (М) таблица «Деталь» будет родительской (рис. 11).

Рис. 11. Таблица «Деталь» - дочерняя в связи с таблицей «Поставщики», и родительская в связи с таблицей «Заказы»

В физической модели БД в качестве значений внешнего ключа в таблице хранятся **ссылки** на значения первичного ключа родительской таблицы. Требование для любого значения внешнего ключа *обязательности* совпадения с каким-либо значением первичного ключа родительской таблицы направлено на запрет так называемых **висячих ссылок**, ссылок «никуда».

Например, если в таблице «Деталь» в качестве номера поставщика для детали Двигатель поставить 4, то это будет означать наличие в таблице висячей ссылки на несуществующий объект (рис. 12).

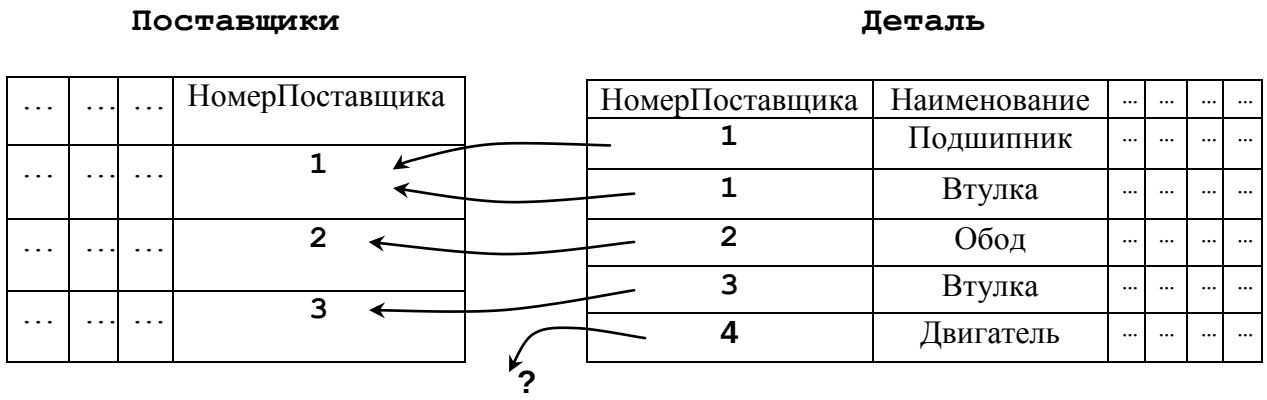


Рис. 12. Образование висячих ссылок при некорректном задании внешних ключей

☑ Внешние ключи – это механизм организации связей объектов в реляционной БД. При проектировании базы выделяют сущности предметной области, их атрибуты, потенциальные ключи, затем определяют связи между сущностями. Для «овеществления» связи в дочерние таблицы «вставляют» в качестве внешних ключей копии потенциальных (или первичных) ключей родительских таблиц.

5. РЕЛЯЦИОННАЯ АЛГЕБРА

Операторы манипулирования данными в реляционной модели основаны на операциях реляционной алгебры. Любой запрос на выборку данных в реляционной БД можно представить композицией этих операций.

Реляционная алгебра – набор операций, использующих отношения в качестве операндов (аргументов) и возвращающих отношение в качестве результата.

Результат любой операции над отношениями *также является отношением*. Это свойство называется **реляционной замкнутостью**. Оно позволяет результат выполнения одной реляционной операции использовать в качестве исходных данных (операндов) для другой. Другими словами, можно записывать *вложенные реляционные выражения*, т.е. выражения, в которых операнды сами представлены реляционными выражениями (как в арифметике).

Базовый набор операций, предложенный Коддом, состоит из восьми операторов:

1. Традиционные операции над множествами:

- объединение;
- пересечение;
- разность;
- декартово произведение.

2. Специальные реляционные операции:

- выборка;
- проекция;
- соединение;
- деление.

Некоторые из этих операций невозможно произвести над совершенно разными таблицами, имеющими неодинаковые заголовки: разные количество или имена атрибутов, домены. Например, простое теоретико-множественное объединение двух таблиц с разными заголовками может не быть отношением, поскольку в отношении не может быть кортежей разных типов. Поэтому, для соблюдения реляционной замкнутости, от отношений-операндов иногда требуется совместимость по типу.

Совместимость по типу отношений означает наличие у них идентичных заголовков, т.е. отношения имеют *одно и то же множество имен атрибутов*, причем атрибуты с одинаковыми именами *определены на одних и тех же доменах*.

Бывает так, что отношения имеют одинаковое количество атрибутов, атрибуты определены на одних и тех же доменах, но имена у них разные. Тогда, чтобы отношения стали совместимыми по типу, над ними (или одним их них) выполняется операция переименования атрибутов (RENAME).

Рассмотрим операции реляционной алгебры подробнее.

1) Объединение (UNION). Объединением двух *совместимых по типу* отношений A и B называется новое отношение, имеющее тот же тип (заголовок), что и у отношений A и B, и тело, состоящее из всех кортежей, которые принадлежат либо A, либо B, либо обоим отношениям одновременно (рис. 13)

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

B

NT	*Товар*	*Цена*
1	Сыр	180
2	Рыба	100
4	Хлеб	20

A UNION B

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20
2	Рыба	100
4	Хлеб	20

Рис. 13. Объединение таблиц A и B

Обратите внимание, что объединение – это не приписывание к кортежам одного отношения *всех* кортежей другого (это привело бы к дублированию кортежей, что нарушает свойства отношений), а добавление тех кортежей, которых не было у первого отношения. Кортежи добавляются целиком, поэтому записей с товаром Хлеб – две, из-за разного значения атрибута *NT*.

☑ Реляционные операции не приводят к автоматическому наследованию ключей! Первичный у отношений A и B ключ *NT* перестал быть таковым в новом отношении - результате объединения.

2) Пересечение (INTERSECT). Пересечением двух *совместимых по типу* отношений A и B называется новое отношение, имеющее тот же тип (заголовок), что и у отношений A и B, и тело, состоящее из всех кортежей, которые принадлежат одновременно обоим исходным отношениям (рис. 14).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

B

NT	*Товар*	*Цена*
1	Сыр	180
2	Рыба	100
4	Хлеб	20

A INTERSECT B

NT	*Товар*	*Цена*
1	Сыр	180

Рис. 14. Пересечение таблиц A и B

3) Вычитание (MINUS). Вычитанием двух *совместимых по типу* отношений A и B называется новое отношение, имеющее тот же тип (заголовок), что и у отношений A и B, и тело, состоящее из всех кортежей, которые принадлежат отношению A и **не** принадлежат отношению B (рис. 15).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

B

NT	*Товар*	*Цена*
1	Сыр	180
2	Рыба	100
4	Хлеб	20

A MINUS B

NT	*Товар*	*Цена*
2	Мясо	300
3	Хлеб	20

Рис. 15. Вычитание таблиц A и B

4) Декартово (или прямое) произведение (TIMES , ×). Декартовым произведением двух отношений A и B называется новое отношение, заголовок которого является сцеплением заголовков отношений A и B, а тело – сцеплением кортежей этих отношений, в котором *каждый* кортеж из A сцепляется *с каждым* кортежем из B.

Совместимости по типу здесь не требуется, A и B *не должны* иметь одинаковых имен атрибутов; если таковые имелись в исходных отношениях, то перед операцией произведения они *переименовываются* (рис. 16).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

B

NП	*Поставщик*	*Адрес*
1	Иванов	Москва
2	Петров	Ростов
3	Сидоров	Сочи

A TIMES B или **A × B**

NT	*Товар*	*Цена*	*NП*	*Поставщик*	*Адрес*
1	Сыр	180	1	Иванов	Москва
1	Сыр	180	2	Петров	Ростов
1	Сыр	180	3	Сидоров	Сочи
2	Мясо	300	1	Иванов	Москва
2	Мясо	300	2	Петров	Ростов
2	Мясо	300	3	Сидоров	Сочи
3	Хлеб	20	1	Иванов	Москва
3	Хлеб	20	2	Петров	Ростов
3	Хлеб	20	3	Сидоров	Сочи

Рис. 16. Декартово произведение отношений A и B

Если бы таблица A имела заголовок | *N* | *Товар* | *Цена* |, а таблица B – заголовок | *N* | *Поставщик* | *Адрес* |, то потребовалось бы переименование атрибута *N* в одной из таблиц; тогда заголовок декартова произведения мог,

например, выглядеть так: | *N* | *Товар* | *Цена* | *N1* | *Поставщик* | *Адрес* |.

Декартово произведение часто используется как основа - вспомогательная операция для определения других реляционных операций.

5) Выборка (или ограничение, селекция) (WHERE q или σ_q, где q – условие, предикат). Выборкой из отношения A по условию q называется новое отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее только те кортежи из A, которые удовлетворяют условию q (т.е. проверка условия q над атрибутами этих кортежей дает значение истина).

Условие q – это логическое выражение, в простейшем случае содержащее операцию сравнения Θ (=, ≠, <, ≤, >, ≥) над атрибутами исходного отношения. Тогда оператор A WHERE q называют **Θ-выборкой** (рис. 17).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

A WHERE Цена < 200

NT	*Товар*	*Цена*
1	Сыр	180
3	Хлеб	20

Рис. 17. Операция Θ-выборки по условию Цена < 200

Другое обозначение рассмотренной выборки: $\sigma_{\text{Цена<200}}(A)$.

Условие в выборке общего вида может быть и более сложным, содержащим логические связки AND (И), OR (ИЛИ), NOT (НЕ) (рис. 18).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

A WHERE Цена = 180 OR Товар = ‘Мясо’

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300

Рис. 18. Операция выборки по условию Цена = 180 OR Товар = ‘Мясо’

6) Проекция. Проекцией отношения A по атрибутам a_i, a_j, …, a_k, принадлежащим отношению A, называется новое отношение, заголовок которого получен из заголовка отношения A посредством удаления из него всех

атрибутов, не входящих во множество a_i, a_j, …, a_k, а тело состоит из множества таких кортежей $<t_i, t_j, \ldots, t_k>$, для которых в исходном отношении A были кортежи со значением атрибута a_i равным t_i, атрибута a_j – равным t_j, …, атрибута a_k – равным t_k.

Проще говоря, проекция получается из A удалением всех столбцов, кроме перечисленных в операции проекции, и последующем удалении дубликатов строк, если таковые образуются. Если операция выборки «вырезает» ненужные строки, то операция проекции «вырезает» столбцы.

Операция проекции обозначается по-разному:

$A[a_i, a_j, \ldots, a_k]$, или $A\{a_i, a_j, \ldots, a_k\}$, или $\Pi_{ai, aj, \ldots, ak}(A)$.

Пример проекций дан на рис. 19.

A

Наименование	*Номер*	*Материал*	*Вес*	*Количество*
Подшипник	1	Сталь	1,2	100
Втулка	2	Цинк	0,8	250
Обод	3	Никель	0,55	32
Втулка	4	Олово	0,35	10
Двигатель	5	Сталь	50	23

A [Материал**]**

Материал
Сталь
Цинк
Никель
Олово

A [Наименование, Количество**]**

Наименование	*Количество*
Подшипник	100
Втулка	250
Обод	32
Втулка	10
Двигатель	23

A [Наименование**]**

Наименование
Подшипник
Втулка
Обод
Двигатель

Рис. 19. Три проекции отношения A

7) Соединение (JOIN, JOIN … ON). Соединением отношений A и B по условию q называется новое отношение (A JOIN B ON q), полученное в результате последовательного выполнения двух операций: декартового произведения отношений A и B и выборки из него по условию q, т.е.

A JOIN B ON q = (A TIMES B) WHERE q.

Различают следующие частные виды соединений:

- Θ-соединение;
- эквисоединение;
- естественное соединение;
- внешнее соединение (левое внешнее, правое и полное внешнее);
- полусоединение.

Θ-соединение ($\bowtie_q$) – соединение, в котором условие q является операцией простого сравнения над атрибутами исходных отношений A и B (рис. 20).

A

NT	*Товар*	*Цена*
1	Сыр	180
2	Мясо	300
3	Хлеб	20

B

NП	*Поставщик*	*Адрес*
1	Иванов	Москва
2	Петров	Ростов
3	Сидоров	Сочи

A JOIN B ON A.NT > B.NП

NT	*Товар*	*Цена*	*NП*	*Поставщик*	*Адрес*
2	Мясо	300	1	Иванов	Москва
3	Хлеб	20	1	Иванов	Москва
3	Хлеб	20	2	Петров	Ростов

Рис. 20. Θ-соединение отношений A и B

Эквисоединение – соединение, в котором условие q является равенством. Это частный вид Θ-соединения.

Рассмотрим для примера (рис. 21) в качестве отношений A и B «урезанные» таблицы «Поставщики» и «Деталь», без некоторых атрибутов, и заменим для краткости имена атрибутов *НомерДетали* на *НД* и *НомерПоставщика* – на *НП*.

A

NП	*Организация*	*Адрес*
1	Металлург	Томск
2	Моторстрой	Уфа
3	Северсталь	Тюмень

B

Наименование	*NД*	*Количество*	*NП*
Подшипник	1	100	1
Втулка	2	250	1
Обод	3	32	2
Втулка	4	10	3
Двигатель	5	23	2

Построение эквисоединения по условию равенства атрибутов NП в этих таблицах потребует переименования одного столбца:

A JOIN B ON A.NП = B.NП

NП	*Организация*	*Адрес*	*Наименование*	*NД*	*Количество*	***NП1***
1	Металлург	Томск	Подшипник	1	100	**1**
1	Металлург	Томск	Втулка	2	250	**1**
2	Моторстрой	Уфа	Обод	3	32	**2**
2	Моторстрой	Уфа	Двигатель	5	23	**2**
3	Северсталь	Тюмень	Втулка	4	10	**3**

Рис. 21. Эквисоединение отношений A и B по атрибуту NП

Условие равенства атрибутов – одно из основных в соединениях, т.к. позволяет связать таблицы, обладающие одинаковой информацией, чтобы обеспечить согласованность данных. Однако эквисоединение имеет «дефект» избыточности: атрибуты *NП* и *NП1* различаются только именем. Если эти два столбца «склеить» в один, то получим отношение, являющееся результатом другой операции – естественного соединения.

Естественное соединение (**NATURAL JOIN**, ⋈) – эквисоединение двух отношений A и B, выполненное по всем общим атрибутам, из результата которого исключается по одному экземпляру каждого общего атрибута.

Пример этого соединения для приведенных выше отношений дан на рис. 22.

A NATURAL JOIN B (или A ⋈ B)

NП	*Организация*	*Адрес*	*Наименование*	*NД*	*Количество*
1	Металлург	Томск	Подшипник	1	100
1	Металлург	Томск	Втулка	2	250
2	Моторстрой	Уфа	Обод	3	32
2	Моторстрой	Уфа	Двигатель	5	23
3	Северсталь	Тюмень	Втулка	4	10

Рис. 22. Естественное соединение отношений A и B по атрибуту NП.

Естественное соединение отношений A (с атрибутами a_1, a_2, …d, …, a_n) и B (с атрибутами b_1, b_2, …d, …b_m) по общему атрибуту d эквивалентно, таким образом, следующей последовательности реляционных операций:

1. Декартово произведение A×B с предварительным переименованием одного из атрибутов A.d или B.d. Например, заголовок A×B может иметь вид: (a_1, a_2, …d, …, a_n, b_1, b_2, …d1, …b_m).

2. Выборка по условию равенства значений атрибутов d и d1:

(A×B) WHERE d=d1, т. е. остаются только те строки, в которых значения d и d1равны.

3. Проекция результата на все атрибуты, кроме d1:

((A×B) WHERE d=d1) [a_1, a_2, …d, …, a_n, b_1, b_2, …, b_m], т.е. удаляем столбец d1, являющийся дублем (по значениям) столбца d.

Естественное соединение можно строить над несколькими отношениями, выполняя его последовательно для каждой пары, при этом не важно, с какой пары (имеющей общие атрибуты) начать, т. к. эта операция обладает свойствами **коммутативности** и **ассоциативности**:

(A NATURAL JOIN **B)** NATURAL JOIN **C** = **A** NATURAL JOIN**(B** NATURAL JOIN **C)**

Бывает так, что при естественном соединении отношений, кортеж одного отношения не находит соответствующего кортежа в другом отношении, т. е. в столбцах соединения оказываются несовпадающие значения. Может потребоваться, чтобы строка из одного отношения была представлена в результате соединения, даже если в другом отношении нет совпадающего значения. Эта цель достигается с помощью операции *внешнего соединения*. Эта операция некоммутативна, ее результат зависит от того, строки какого отношения-операнда должны быть обязательно включены. Поэтому используют разные виды внешних соединений.

Левое внешнее соединение (⟕) – соединение отношений A и B, при котором кортежи отношения A, не имеющие совпадающих значений в общих столбцах отношения B, также включаются в результирующее

отношение. Для обозначения отсутствующих значений в B используется определитель **NULL** (рис. 23).

A

c	d
a	1
b	2

B

d	e
1	x
1	y
3	z

A ⟕ B

c	d	e
a	1	x
a	1	y
b	2	NULL

A ⋈B

c	d	e
a	1	x
a	1	y

Рис. 23. Левое внешнее соединение и естественное соединение (справа) отношений A и B

Аналогично определяется правое внешнее соединение, в которое обязательно включаются строки правого операнда B.

Правое внешнее соединение (⟖) – соединение отношений A и B, при котором кортежи отношения B, не имеющие совпадающих значений в общих столбцах отношения A, также включаются в результирующее отношение (рис. 24).

A

c	d
a	1
b	2

B

d	e
1	x
1	y
3	z

A ⟖ B

c	d	e
a	1	x
a	1	y
NULL	3	z

A ⋈B

c	d	e
a	1	x
a	1	y

Рис. 24. Правое внешнее соединение и естественное соединение (справа) отношений A и B

Полное внешнее соединение – соединение отношений A и B, при котором в результат попадают все кортежи из обеих отношений, имеющие несовпадающие значения в общих столбцах; для обозначения несовпадающих значений используется **NULL** (рис. 25).

A

c	d
a	1
b	2

B

d	e
1	x
1	y
3	z

A ⟕ B UNION A ⟖ B

c	d	e
a	1	x
a	1	y
b	2	NULL
NULL	3	z

A ⋈B

c	d	e
a	1	x
a	1	y

Рис. 25. Полное внешнее соединение отношений A и B

Внешние соединения строятся для того, чтобы не утрачивать при естественных соединениях исходную информацию. Бывает потребность, наоборот, сократить количество кортежей, которые нужно обработать для получения соединения. Это полезно, например, при вычислении соединений в распределенных БД. Для этой цели служат операции *полусоединения*.

Полусоединение ($\triangleright_q$) – операция над отношениями A и B, которая определяет новое отношение, содержащее только те кортежи отношения A, которые входят в соединение отношений A и B (по условию q).

В зависимости от типа условия q существуют *полутетасоединения*, *полуэквисоединения* и *полуестественные* ($\triangleright$) соединения. Пример такого соединения приведен на рис. 26.

A

c	d
a	1
b	2

B

d	e
1	x
1	y
3	z

A ▷B

c	d
a	1

Рис. 26. Полуестественное соединение отношений A и B

8) Деление (DIVIDEBY, или ÷). Пусть отношение A определено на множестве атрибутов $\mathcal{A}$, отношение B – на множестве атрибутов $\mathcal{B}$, причем $\mathcal{B} \subseteq \mathcal{A}$. Пусть $C = \mathcal{A} - \mathcal{B}$, т.е. C является множеством атрибутов отношения A, которые не являются атрибутами отношения B. Тогда результатом операции деления A÷B (или A **DIVIDEBY** B) является набор кортежей отношения A,

определенных на множестве атрибутов *C*, которые соответствуют комбинации всех кортежей отношения B.

Деление можно представить композицией других реляционных операций:

$T_1 = П_C(A)$ – проекция A на множество атрибутов *C*;

$T_2 = П_C((B \times T_1) - A)$;

$T = T_1 - T_2$.

Типичные запросы, требующие деления, имеют в формулировке слово «все».

Пусть, например, отношение A, определенное на атрибутах *НП*, *НД*, содержит сведения о поставках деталей (*НД*) поставщиками (*НП*); список имеющихся деталей представлен в отношении B, определенном на атрибуте *НД*. Запрос: определить номера поставщиков, поставляющих *все* детали, реализуется операцией деления A÷B (рис. 27).

A

НП	*НД*
1	1
1	2
1	3
2	1
2	2
3	1

B

НД
1
2
3

A ÷ B

НП
1

Рис. 27. Операция деления отношения A на отношение B

Невыразимые в реляционной алгебре запросы. К таким запросам относятся, например:

1. Выдать список (имен) атрибутов, удовлетворяющих некоторому условию.

2. Построить перекрестную (кросс-) таблицу (аналог сводной таблицы в Excel) (рис.28).

Исходная таблица

Товар	*Месяц*	*Количество*
Чайник	март	100
Миксер	март	200
Пылесос	март	300
Чайник	апрель	150
Миксер	апрель	250
Пылесос	апрель	350

Кросс-таблица

Товар	*Март*	*Апрель*
Чайник	100	150
Миксер	200	250
Пылесос	300	350

Рис. 28. Перекрестная таблица

Многие СУБД допускают более широкий, чем в реляционной алгебре, спектр операций над данными; в частности, в Access есть средства построения кросс-таблиц.

6. ЦЕЛОСТНОСТЬ ДАННЫХ

Понятие *целостность данных* используется для описания точности и корректности хранящейся в базе информации. В базе данных может храниться любое число ограничений произвольной сложности, обеспечивающих целостность. Языки проектирования баз данных обычно содержат средства описания различных правил и процедур, поддерживающих так называемую **семантическую целостность** данных – непротиворечивость информации, которую нельзя формально вывести из определения модели данных. Например, «отдел, содержащий менее четырех сотрудников, должен быть распущен по истечении месячного срока со дня ревизии» или «нельзя зачислять во второй отдел сотрудников, имеющих домашних животных» и т.п. Такие ограничения называют «прикладными», «бизнес-правилами».

Часть таких ограничений относится к допустимым значениям атрибутов и может быть выражена *ограничениями на домен*. Другие могут относиться к отношениям, например, «вес детали из олова не должен превышать 4 кг». Наконец, ограничения на взаимосвязи между таблицами относятся к целостности базы данных. Например: «поставщики из Тюмени не могут поставлять двигатели».

Кроме прикладных ограничений, которые СУБД не может проверить без вмешательства разработчика конкретной базы, существуют правила, поддерживающие согласованное состояние БД в соответствии с используемой моделью данных. В реляционной модели таких правил два: **целостность сущностей** и **ссылочная целостность**. Эти правила реляционная СУБД проверяет автоматически в соответствии с выбранной *стратегией поддержки целостности*. Определение этих правил и стратегий использует понятие NULL.

Определитель NULL – неизвестное, неопределенное в данный момент значение атрибута. Это не пустая строка или число 0. Ключевое слово NULL *обозначает отсутствие* какого-либо значения, несмотря на то, что его часто и называют NULL-значением.

Использование этого определителя фактически приводит к использованию в реляционной модели не двузначной (истина, ложь) логики, а трехзначной (истина, ложь, неопределенность). Вывод в трехзначной логике может привести к истинным следствиям, полученным из неопределенных утверждений. Поэтому использование понятия NULL в реляционной модели часто дискутируется и не принимается некоторыми специалистами как элемент теории реляционных БД.

Целостность сущностей означает, что ни один атрибут *первичного ключа* отношения не может быть неопределенным, т.е. значение любого атрибута первичного ключа не может содержать NULL.

В самом деле, первичный ключ – это минимальное значение, которое позволяет выбрать любую запись однозначно; если допустить в этом значении NULL, то либо мы вообще не сможем выбрать запись, либо ключ не является минимальным – среди его атрибутов есть лишние.

СУБД проверяет целостность сущности при всяком обновлении первичного ключа или добавлении новых записей: значение ключа не должно быть неопределенным или повторяться.

Ссылочная целостность означает, что база данных не содержит значений внешних ключей, не имеющих соответствия со значениями каких-либо потенциальных ключей базовых отношений.

Другими словами, в БД не должно быть «висячих» ссылок: если b ссылается на a, то a должно существовать. Как уже рассматривалось в разделе 4, ссылка идет от внешнего ключа дочерней таблицы к первичному ключу родительской таблицы.

Обеспечивая ссылочную целостность, СУБД действует в соответствии с определенной стратегией, тип которой определяет проектировщик БД на логическом уровне проектирования. Чтобы понять, как работают наиболее известные стратегии, рассмотрим вначале, какие операции с БД могут нарушить ссылочную целостность.

Очевидно, что выборка не нарушит целостности, т.к. она не изменяет состояние БД; «подозрительными» могут стать операции обновления, добавления и удаления записей, если они «задевают» значения ключей.

Рассмотрим связанные (по номеру поставщика) таблицы «Поставщики» (родительская таблица) и «Деталь» (дочерняя таблица). Стрелки указывают ссылки от внешнего ключа в дочерней таблице к первичному ключу родительской (рис. 29).

Поставщики

Организация	*Адрес*	*Телефон*	***NП***
Металлург	Томск	2233445	1
Моторстрой	Уфа	6677558	2
Северсталь	Тюмень	9874531	3

Деталь

NП	*Наименование*	*Вес*	*NД*
1	Подшипник	1,2	1
1	Втулка	0,8	2
2	Обод	0,55	3
3	Винт	0,35	4
2	Двигатель	50	5

Рис. 29. Иллюстрация к понятию ссылочной целостности

Родительская таблица «Поставщики»:

1. Обновление записи. Если в таблице обновить запись, изменив, например, адрес поставщика, то ничего не нарушится. Если же изменить значение первичного ключа на новое, например, установить для первой записи *НП*=5, то «повиснут» ссылки для двух записей таблицы «Деталь», т.к. *НП* с номером 1 уже не существует. Таким образом, *обновление в родительской таблице может нарушить целостность, если обновляется первичный или потенциальный ключ, на который имеются ссылки*.

2. Удаление записи. *Удаление любой записи в родительской таблице приведет к нарушению целостности, если на нее были ссылки из дочерней таблицы*.

3. Добавление новой записи. Эта операция не нарушит целостности, т.к. на новое значение первичного ключа *еще не было* ссылок.

Дочерняя таблица «Деталь»:

1. *Обновление записи может нарушить целостность, если обновленное значение внешнего ключа не совпадает ни с одним значением первичного ключа таблицы- родителя*. Например, изменив запись <2, Обод, 32, 3> на <4, Обод, 32, 3>, мы получим висячую ссылку, т.к. не существует поставщика с *НП*=4.

2. Удаление любой записи в дочерней таблице не приведет к нарушению целостности, т.к. запись удаляется вместе со всеми ее ссылками.

3. Добавление новой записи. *Добавление записи в дочернюю таблицу может нарушить целостность, если вводится значение внешнего ключа, не совпадающее ни с одним значением первичного ключа родительской таблицы.*

Итак, **операции, которые могут нарушить целостность**:

– для родительской таблицы – обновление, удаление записи;

– для дочерней таблицы – обновление, добавление (вставка) записи.

Стратегия поддержки целостности – это правила, выполняемые СУБД для обеспечения целостности данных. Чаще всего в реляционных БД используются следующие стратегии:

• **RESTRICT** или **NO ACTION–** запретить выполнение операции, если она приводит к нарушению целостности.

• **CASCADE –** последовательно проводить изменения значений атрибутов, меняя соответствующие значения ключей, ***начиная от родительской таблицы (!!)*** к дочерней; если эта дочерняя таблица является родительской для другой связанной с ней таблицы, то изменения проводятся и в ней, и так далее, «каскадом». Например, при удалении из таблицы «Поставщики» записи с *НП* = 2, автоматически удалятся из таблицы «Деталь» записи <2,Обод, 32, 3> и <2, Двигатель, 23, 5>, ссылающиеся на нее. Аналогично, при обновлении – замене значения *НП* с 2 на 7, автоматически обновятся и эти записи на <7,Обод, 32, 3> и <7, Двигатель, 23, 5>. Если на любом уровне каскада какая-либо операция не может быть выполнена, то происходит «откат» - возвращение всей БД в исходное состояние.

• **SET NULL –** разрешить выполнение операций, но все возникающие некорректные значения (потенциально «висячие» ссылки) заменить на NULL, если это возможно. Например, при удалении в *родительском* отношении записи <Северсталь, Тюмень, 9874531, 3> ссылающаяся на нее запись в таблице «Деталь» будет иметь значение <NULL, Двигатель, 23, 5>. Вставка в *дочернюю* таблицу кортежа <4, Спица, 100, 6> будет соответственно заменена вставкой кортежа <NULL, Спица, 100, 6>, поскольку *НП* с номером 4 в родительской таблице нет. Аналогично при некорректном обновлении родительской или дочерней таблицы «зависшее» значение внешнего ключа заменяется на NULL.

☝**!** Обратите внимание, что NULL появляется только во **внешнем ключе дочерней** таблицы и только *если этот атрибут не входит, как часть, в составной первичный ключ* этой таблицы.

Например, пусть отношение «Хранение» содержит информацию о том, на каких складах (*NC* – номер склада) находятся детали. В нем пара *НД, NC* – первичный ключ, а *часть* этой пары – атрибут *НД* является также и внешним ключом, по которому таблица «Хранение» связывается с

родительской таблицей «Деталь». Поэтому при удалении кортежа из таблицы «Деталь» стратегия SET NULL не может проставить NULL во внешнем ключе, т.к. части первичного ключа не могут быть неопределенными (рис. 30).

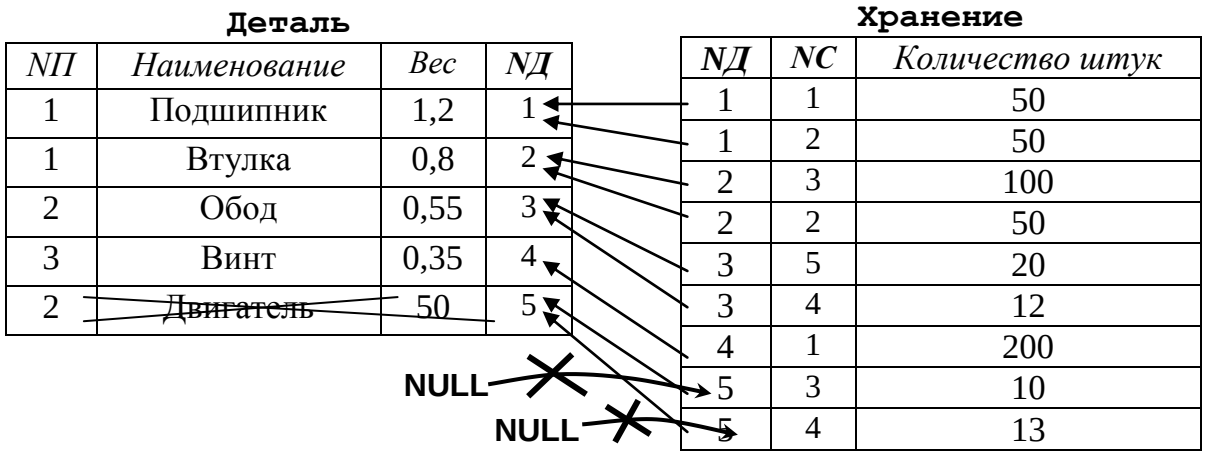


Рис. 30. Пример невозможности применения стратегии SET NULL

• **SET DEFAULT –** разрешить выполнение операций, но все возникающие некорректные значения (потенциально «висячие» ссылки) *внешних ключей* заменить на значение, указанное по умолчанию для этих атрибутов. Например, если для атрибута *НП* в таблице «Деталь» задано значение по умолчанию 1, то, например, при удалении записи <Северсталь, Тюмень, 9874531, 3> ссылающаяся на нее запись в таблице «Деталь» будет иметь значение <1, Двигатель, 23, 5>. В качестве значения по умолчанию может быть выбрано некое «спецзначение», имеющее особый статус. Тогда в родительскую таблицу помещается соответствующий «спецкортеж», имеющий в качестве неизменяемого значения первичного ключа это «спецзначение», на которое и образуется ссылка.

• **IGNORE** или **NO CHECK –** разрешаются любые операции, никаких действий по сохранению ссылочной целостности данных не производится.

Стратегии поддержки целостности могут быть смешанных типов, например, при удалении данных из таблицы – одна стратегия, а при обновлении – другая. Наиболее «корректно» ведет себя стратегия CASCADE, хотя она ведет при удалениях к потере информации, которая позже могла бы быть

востребована. Более «щадящая» в этом смысле стратегия SET NULL приводит к полной потере связи с прежними родительскими сущностями: по неопределенности NULL невозможно определить, какому кортежу ранее соответствовал данный кортеж. Стратегия SET DEFAULT также «чревата беспамятством», а при использовании специальных значений ведет к хранению «особых» кортежей, и эту «неравноправность» СУБД должна иметь в виду.

7. НОРМАЛЬНЫЕ ФОРМЫ ОТНОШЕНИЙ

Реляционные таблицы изначально могут быть плохо спроектированы, например, описание нескольких разных сущностей «втиснуто» в одну таблицу. Несмотря на то, что внешне такая таблица выглядит вполне «респектабельно», имеет правильно определенные ключи, удовлетворяет всем свойствам отношений, она имеет внутреннюю «червоточину», «дурную» избыточность, которые приводят к *аномалиям* при удалении, вставке и обновлении данных. Чтобы исправить ситуацию, «плохие» таблицы подвергаются процедуре *нормализации*, которая обычно приводит к декомпозиции этих таблиц – разбиению их на несколько новых таблиц, уже без аномалий.

Исходные таблицы могут обладать нежелательными свойствами в разной степени, ошибки проектирования могут быть грубыми и дорого обходиться СУБД, а могут быть тонкими и, в принципе, терпимыми. Соответственно этому классифицируются и «хорошие» свойства таблиц, так называемые *нормальные формы*, обладание которыми избавляет от различных аномалий. Говорят, что отношение *находится* в определенной нормальной форме (НФ), если оно удовлетворяет заданному набору условий.

Нормальные формы отношений образуют последовательность, в которой каждая форма включает все условия предыдущих форм и добавляет свои, новые. Таким образом, нормализация отношений – процесс последовательного их «улучшения», перехода к формам более высоких порядков.

Наиболее известные нормальные формы можно представить иерархией, в которой каждая новая форма, добавляя новые условия, сужает пространство отношений, этим формам соответствующих (рис. 31).

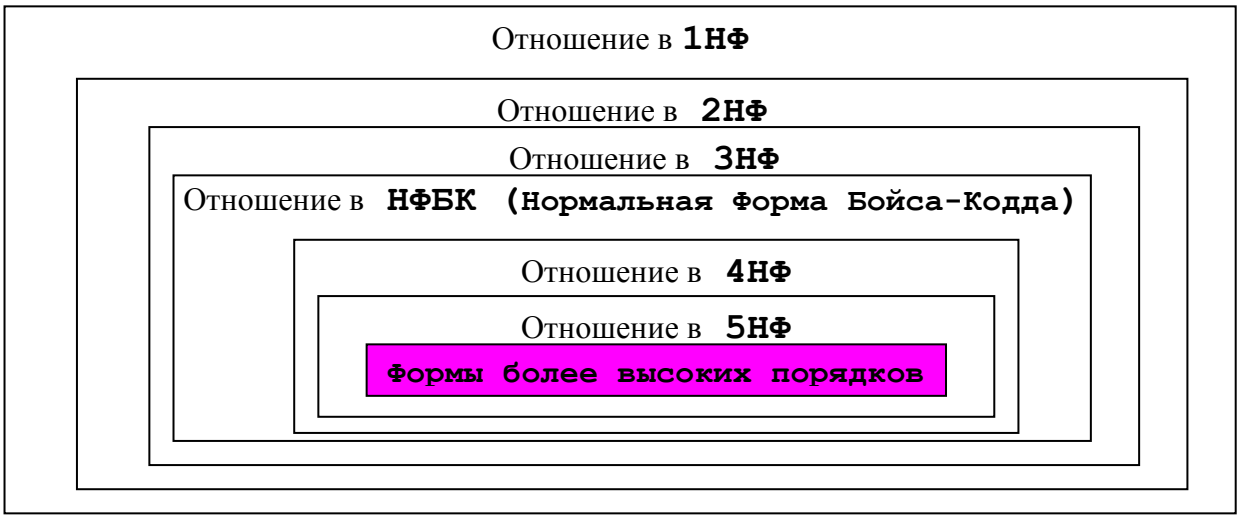


Рис. 31. Уровни нормализации

На практике часто бывает достаточно выполнения условий 3НФ, т.к. приведение к остальным формам может быть сложным и не всегда эффективным.

Определение нормальных форм опирается на понятия, формализующие различные виды зависимостей между атрибутами. Рассмотрим основные.

Функциональная зависимость. Пусть **R** является отношением, а **X** и **Y** – его атрибуты (или подмножества его атрибутов). Тогда **Y** *функционально зависит* от **X** (записывается как **X** → **Y**) тогда и только тогда, когда каждое значение атрибута (или множества атрибутов) **X** в отношении **R** связано в точности с одним значением атрибута (или множества атрибутов) **Y** в этом отношении.

Другими словами, если два кортежа в **R** совпадают по значению **X**, то они совпадают и по значению **Y**. Но одному значению **Y** может соответствовать несколько разных значений атрибута **X**; здесь связь от **X** к **Y** типа «многие к одному».

Проще представить, о чем идет речь, вспомнив определение функции y=f(x). Разным x может соответствовать один и тот же y, но любому заданному аргументу x соответствует лишь одно значение y (не может быть разных значений y функции y=sin(x), если задан x, например, x=π). Понятие функциональной зависимости в БД в точности совпадает с определением функции в каждый *фиксированный* момент времени.

Детерминант функциональной зависимости – левая часть (**X**) в записи **X → Y** этой зависимости.

Полная функциональная зависимость. Зависимость **X → Y** называется полной, если **Y** не зависит ни от какого точного подмножества **X,** т.е. **Y** не зависит от части детерминанта **X.** Зависимость **частичная**, если в **X** есть атрибут, при удалении которого зависимость **Y** от **X** сохраняется.

Транзитивная зависимость. Если для атрибутов **X, Y, Z** некоторого отношения существуют функциональные зависимости **X → Y** и **Y → Z**, то говорят, что атрибут **Z** *транзитивно зависит от атрибута* **X** через атрибут **Y** (при условии, что **X** функционально не зависит ни от **Y**, ни от **Z**).

Первая нормальная форма (1НФ). Отношение находится в 1НФ, если значения всех его атрибутов атомарны.

Атомарность означает, что на пересечении любого столбца и любой строки в этом отношении содержится только одно значение.

Отношения в *реляционной* БД по определению уже находятся в 1НФ. Однако при проектировании отношений или преобразовании данных из формата источника (например, стандартной формы ввода данных) в формат реляционной таблицы часто требуется нормализация – приведение к 1НФ. В исходной таблице устраняются группы атрибутов, в которых возможно наличие нескольких значений для единственного значения ключевого атрибута. Полученное в результате отношение, находящееся в 1НФ, называют *нормализованным*.

Пример ненормализованной и нормализованной таблиц (рис. 32).

Сотрудники

Отдел	*Фамилия*
1	Иванов
	Петров
	Сидоров
2	Ляпина
	Журов

Сотрудники

Отдел	Фамилия
1	Иванов
1	Петров
1	Сидоров
2	Ляпина
2	Журов

Рис. 32. Приведение таблицы к 1НФ

Для понимания нормальных форм более высоких порядков рассмотрим пример «аномальной» таблицы. Пусть таблица «Рекламное Агентство» содержит информацию о заказах на рекламу, атрибуты: *N3* - номер заказчика, *Адрес* заказчика, *Статус* заказчика, назначаемый в зависимости от города пребывания заказчика, *Индекс* – почтовый индекс города, *NP* – номер рекламы, *Тип* размещения рекламы и *Количество* заказанных объявлений. Пара атрибутов *N3* и *NP* образуют единственный (отметим это) потенциальный ключ, являющийся поэтому и первичным (рис. 33).

Рекламное Агентство

N3	*Адрес*	*Статус*	*Индекс*	***NP***	*Тип*	*Количество*
1	Ростов	1	344	1	Газета	2
1	Ростов	1	344	2	Растяжка	3
7	Ростов	1	344	4	Радио	20
2	Шахты	3	341	5	Магазин	4
3	Каменск	4	346	1	Газета	1
4	Таганрог	2	340	2	Растяжка	5
4	Таганрог	2	340	4	Радио	10
6	Таганрог	2	340	3	ТВ	100
3	Каменск	4	346	3	ТВ	20
5	Ростов	1	344	1	Газета	3
5	Ростов	1	344	4	Радио	8
1	Ростов	1	344	3	ТВ	10

Рис. 33. Пример «плохой» таблицы

Эта таблица, очевидно, страдает избыточностью: например, для каждого заказчика повторяется его адрес, для каждого типа рекламы повторяется

значение цены за единицу, для каждого города – его индекс. Эта избыточность порождает аномалии при изменении данных.

Аномалии вставки (операция **INSERT**). Нельзя поместить информацию о том, что некоторый заказчик находится в определенном городе, не указав сведения хотя бы об одной рекламе, поскольку для потенциального заказчика *нельзя сформировать первичный ключ*. По той же причине нельзя вставить сведения о новом типе рекламы, (например, Интернет), если у нее нет пока заказчика.

Аномалии удаления (операция **DELETE**). Если из таблицы удалить кортеж, который является единственным для некоторого заказчика, например с *N3*=2, то мы утратим и всю информацию о рекламе с размещением в магазинах. То же самое, если вдруг реклама в магазинах отменена, из БД уйдет и вся информация о заказчике из Шахт. Т.е. при удалении кортежей теряется *слишком* много информации.

Аномалии обновления (операция **UPDATE**). Повтор адреса для каждого заказчика приводит к тому, что при перемещении, например, заказчика с *N3* =1 из Ростова в Краснодар, необходимо будет отыскать в таблице «Рекламное агентство» *все* кортежи, в которых значения *N3* =1 и Ростов связаны между собой. Иначе база данных окажется в противоречивом состоянии – в одних кортежах адрес заказчика 1 будет Ростов, а в других – Краснодар.

Эти аномалии возникли из-за того, что в таблице собрана информация о разных сущностях – рекламе и заказчиках. Каждая сущность имеет свой первичный ключ – *NP* (реклама) и *N3* (заказчики). Другие атрибуты этих сущностей зависят от этих (своих) первичных ключей, т.е. в таблице имеется зависимость неключевых атрибутов от *части* исходного (парного) первичного ключа. Определим эту ситуацию более точно.

Неключевой атрибут – атрибут, не входящий в состав первичного ключа рассматриваемого отношения.

Рассмотрим функциональные зависимости атрибутов в нашей таблице:

N3, NP → Адрес, Статус, Индекс, Тип, Количество – зависимость от первичного ключа

N3 → Адрес, Статус, Индекс – частичная зависимость

NP → Тип – частичная зависимость

Адрес → Статус, Индекс – транзитивная зависимость

Вторая нормальная форма (2НФ). Отношение находится во второй нормальной форме тогда и только тогда, когда оно находится в первой нормальной форме и каждый его неключевой атрибут характеризуется *полной* функциональной зависимостью от первичного ключа этого отношения.

Другими словами, в 2НФ нет неключевых атрибутов, зависящих от *части* первичного ключа.

В отношении «Рекламное агентство» есть частичные зависимости, значит оно не находится в 2НФ.

Процесс приведения к 2НФ состоит в декомпозиции исходной таблицы – разбиению ее на несколько новых таблиц. Те атрибуты, которые зависят от части ключа, выносятся в отдельное отношение вместе с копией детерминанта. Разобьем нашу таблицу (рис. 34), получим три отношения.

Заказчики

N3	*Адрес*	*Статус*	*Индекс*
1	Ростов	1	344
2	Шахты	3	341
3	Каменск	4	346
4	Таганрог	2	340
5	Ростов	1	344
6	Таганрог	2	340
7	Ростов	1	344

Реклама

NP	*Тип*
1	Газета
2	Растяжка
3	ТВ
4	Радио
5	Магазин

Заказы

N3	***NP***	*Количество*
1	1	2
1	2	3
7	4	20
2	5	4
3	1	1
4	2	5
4	4	10
6	3	100
3	3	20
5	1	3
5	4	8
1	3	10

Рис. 34. Отношения, находящиеся в 2НФ

Каждая из этих таблиц находится в 2НФ: неключевые атрибуты в отношениях «Заказчики» и «Реклама» зависят от своих первичных ключей полностью (ключ состоит из одного атрибута), в отношении «Заказы» количество заказов зависит и от заказчика, и от типа рекламы, т.е. частичная зависимость отсутствует.

☑ Если первичный ключ отношения состоит из *единственного* атрибута, то это отношение уже находится в 2НФ.

☑ Важно, что исходное отношение **может быть восстановлено** естественным соединением полученных трех таблиц. Это означает, что при декомпозиции информация не была утрачена. Такая декомпозиция называется **декомпозицией без потерь.**

Отметим, что теперь избыточность и аномалии, связанные с совмещением информации о заказчиках и рекламе отсутствуют. Однако некоторые аномалии еще остались. Они касаются таблицы «Заказчики», т.к. в ней совмещена информация о заказчиках и городе, в котором заказчик находится. Отметим, что статус назначался заказчику в зависимости от города, а не от его личности.

Сохранившаяся транзитивная зависимость *N3 → Адрес*, *Адрес → Статус*, *Индекс* не позволяет поместить в БД сведения о некотором городе (его статусе, почтовом индексе), пока не появится какой-либо заказчик из этого города (значение первичного ключа не может быть не определено). Это аномалия вставки. При удалении адреса, представленного единственным кортежем, здесь с *N3*=2, удалится не только вся информация о заказчике, но и о том, что городу Шахты был присвоен статус 3. Это аномалия удаления. В отношении «Заказчики» значения статуса и индекса для каждого города многократно повторяются, т.е. избыточность сохраняется. Значит, при изменении статуса, например, Ростова с 1 на 5, нужно будет отыскать в таблице все кортежи, в которых значения Ростова и 1 связаны между собой. Это аномалия обновления. Для избавления от этих неприятностей потребуется

еще одна декомпозиция, избавляющая от зависимости неключевых атрибутов.

Третья нормальная форма (3НФ). Отношение находится в третьей нормальной форме тогда и только тогда, когда оно находится в 2НФ, и в нем *нет неключевых атрибутов, транзитивно зависящих от первичного ключа*.

Другими словами, в 3НФ нет функциональной зависимости неключевых атрибутов друг от друга, т.е. они *взаимно независимы*.

В таблице «Заказчики» есть такая зависимость между неключевыми атрибутами *Адрес* и *Статус*, *Адрес* и *Индекс*. Значит это отношение *не* находится в 3НФ.

Для приведения к 3НФ также потребуется декомпозиция таблицы «Заказчики». Те неключевые атрибуты, которые являются взаимно зависимыми, выносятся в отдельное отношение вместе с копией детерминанта. В результате получим два новых отношения (рис. 35).

Клиент

N3	*Адрес*
1	Ростов
2	Шахты
3	Каменск
4	Таганрог
5	Ростов
6	Таганрог
7	Ростов

Города

Адрес	*Статус*	*Индекс*
Ростов	1	344
Таганрог	2	340
Шахты	3	341
Каменск	4	346

Рис. 35. Отношения, находящиеся в 3НФ

Теперь информацию о заказчике и городах можно менять независимо одну от другой, замена статуса, города или индекса происходит только в одной записи в таблице «Города». Таким образом, выявленные аномалии отсутствуют.

Отметим, что исходное отношение восстанавливается операцией естественного соединения, т.е. произведенная декомпозиция – без потерь.

Отношения баз данных проектируются таким образом, чтобы исключить

частичные и транзитивные зависимости от первичного ключа. Однако в определениях соответствующих нормальных форм предполагалось, что отношение имеет только один потенциальный ключ. Если это не так, наличие 3НФ не всегда удовлетворительно. Например, 3НФ может не дать желаемого результата в случаях, когда:

1. Отношение имеет два или более потенциальных ключа;

2. Эти потенциальные ключи являются составными;

3. Два или более потенциальных ключа перекрываются (то есть имеют по крайней мере один общий атрибут).

Поэтому впоследствии определение 3НФ было заменено более строгим определением нормальной формы Бойса-Кодда. Заметим, однако, что если для отношения эти три условия не выполняются (что на практике чаще всего и бывает), то 3НФ полностью эквивалентна нормальной форме Бойса-Кодда.

Нормальная форма Бойса-Кодда (НФБК). Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

Для проверки НФБК для отношения, нужно найти все его детерминанты (детерминант, напомним, это атрибут или группа атрибутов, от которой полностью зависит другой атрибут) и убедиться, что они являются потенциальными ключами.

Различие 3НФ и НФБК в том, что 3НФ допускает зависимости $A \rightarrow B$, если B является первичным ключом, а атрибут A не обязательно является потенциальным ключом (действительно, такая зависимость не является транзитивной, т.к. B – ключевой атрибут). В НФБК подобная зависимость допускается *только* тогда, когда атрибут A является потенциальным ключом. Следовательно, НФБК является более жесткой версией 3НФ: отношение в НФБК находится в 3НФ, но не всякая 3НФ является НФБК.

Например, отношение «Рекламное агентство» (см. рис. 33) не находится в

НФБК, несмотря на то, что у него *один* потенциальный ключ, являющийся и первичным. Но зависимости *N3* → *Адрес*, *Статус*, *Индекс*; *NP* → *Тип*; *Адрес* → *Статус*, *Индекс* выявляют детерминанты *N3*, *Адрес*, *NP*, не являющиеся потенциальными ключами.

☑ Обратите внимание, что в определении НФБК *нет* требования принадлежности отношения к 2НФ, т.к. условия НФБК автоматически включают 2НФ.

Отношение «Заказчики» (см. рис. 34), находящееся в 2НФ, но не в 3НФ, также *не* находится и в НФБК, т. к. оставшаяся зависимость *Адрес* → *Статус*, *Индекс* имеет детерминант *Адрес*, не являющийся потенциальным ключом.

Полученные в результате декомпозиции отношения «Клиент», «Города», «Реклама» и «Заказы» находятся как в 3НФ, так и в НФБК.

Приведем пример отношения, находящегося в 3НФ, но *не* в НФБК. Рассмотрим таблицу, в которой учтены связи между учениками, занимающимися некоторыми видами спорта у определенных тренеров. При этом должны выполняться ограничения:

1. Каждый ученик занимается определенным видом спорта только у *одного* тренера. 2. Любой вид спорта может иметь несколько тренеров, но *каждый тренер* может работать *только в одном виде спорта*.

Исходное отношение имеет вид как на рис. 36.

Тренировка

Ученик	*Спорт*	*Тренер*
Легков	Теннис	Белый
Легков	Плавание	Синеев
Тяжелов	Теннис	Белый
Тяжелов	Плавание	Желтов

Рис. 36. Отношение, не находящееся в НФБК

Рассмотрим имеющиеся зависимости. Очевидно, что:

{*Ученик, Спорт*} → *Тренер* (следует из ограничения1);

{*Ученик*, *Тренер*} → *Спорт* (следует из ограничения1) ;

Тренер → Спорт (следует из ограничения 2).

Из ограничений 1 и 2 следует, что в этом отношении имеются два перекрывающихся потенциальных ключа: {*Ученик, Спорт*} и {*Ученик, Тренер*}. Взяв первый из них в качестве первичного ключа, убедимся, что отношение *находится в 3НФ*, т.к. неключевой атрибут всего один, и он не зависит от части этого ключа.

Однако это отношение *не находится в НФБК*, так как существует детерминант (*Тренер*), не являющийся потенциальным ключом. Обратите внимание, что если бы мы взяли в качестве первичного *другой* потенциальный ключ, то это отношение не находилось бы даже в 2НФ (из-за частичной зависимости *Тренер → Спорт*).

Отсутствие НФБК приводит к аномалиям: если, например, удалить сведения о занятиях ученика Тяжелова плаванием, то утратится информация о том, что Желтов является тренером по этому виду спорта.

С помощью декомпозиции отношения «Тренировка» можно получить два новых отношения, теперь уже находящихся в НФБК (рис. 37).

УТ

Ученик	***Тренер***
Легков	Белый
Легков	Синеев
Тяжелов	Белый
Тяжелов	Желтов

ТС

Тренер	*Спорт*
Белый	Теннис
Синеев	Плавание
Желтов	Плавание

Рис. 37. Отношения, находящиеся в НФБК

Легко можно проверить, что отмеченные аномалии отсутствуют. Из всех зависимостей осталась только одна, в «ТС»: *Тренер → Спорт*, которая является «правильной» - полной зависимостью единственного неключевого атрибута от первичного ключа.

Однако появилась новая проблема – невозможность восстановления исходных зависимостей при обратной операции – соединении таблиц «УТ» и «ТС». Прежняя функциональная зависимость {*Ученик, Спорт*} → *Тренер* не

может быть выведена из оставшейся зависимости *Тренер → Спорт*. В результате два отношения «УТ» и «ТС» *не могут обновляться независимо*. Например, попытка вставки в «УТ» кортежа <Легков, Желтов> должна быть отвергнута системой из-за нарушения ограничения 1: Легков уже занимается плаванием у другого тренера. Однако обнаружить этот факт система не может, не «заглянув» в отношение «ТС». Отметим, что потеряна *зависимость*, однако само отношение – таблица «Тренировка» восстанавливается естественным соединением без потерь.

☑ Декомпозиция без потерь не всегда приводит к независимым отношениям. Некоторые отношения *не могут быть* декомпозированы на *независимые* компоненты.

Рассмотрим пример отношений с перекрывающимися ключами, но находящихся, тем не менее, в НФБК.

Пусть имеется отношение «Магазин» с атрибутами *Клиент*, *Товар* и *Позиция*, отражающее информацию о том, что некоторый клиент состоит в очереди, в записанной позиции, за получением своего товара. Ограничение состоит в том, что никакие два клиента не могут иметь одну и ту же позицию в списке получающих один и тот же товар. Имеющиеся тогда функциональные зависимости:

{Клиент, Товар} → Позиция;

{Позиция, Товар} → Клиент.

Однако, несмотря на то, что потенциальные ключи {Клиент, Товар} и {Позиция, Товар} перекрываются, отношение находится в НФБК, поскольку эти ключи являются его единственными детерминантами.

НФБК позволяет устранить любые аномалии, связанные с функциональными зависимостями. Однако существует еще один тип зависимости - *многозначная зависимость*, которая при проектировании отношений также может вызвать проблемы, связанные с избыточностью данных.

Многозначные зависимости часто возникают из-за необходимости приведения проектируемых таблиц к 1НФ, когда не допускается наличие *набора* значений на пересечении любой строки и любого столбца. Если в отношении, например, имеются два многозначных атрибута, то в нем фактически присутствуют две независимые связи типа 1:М. Тогда для обеспечения непротиворечивости информации, кортежи должны содержать каждое значение одного атрибута в сочетании с каждым значением другого атрибута. Это порождает избыточность данных.

Пусть, например, отношение «Фирма» содержит сведения о ремонте автомобилей: атрибуты *NФ* – номер отделения фирмы, *Авто* – тип автомобиля и *Сотрудник* – фамилия сотрудника. В отделении компании с номером Ф2 работают три сотрудника и зарегистрированы два типа автомобиля. Однако если в этом отделении между сотрудниками и типами автомобилей нет никакой прямой связи, то необходимо создать строку для каждой комбинации сотрудника и типа автомобиля, только так можно гарантировать, что отношение находится в непротиворечивом состоянии (рис. 38).

Фирма

NФ	*Авто*	*Сотрудник*
Ф2	Volvo	Иванов
Ф2	Volvo	Петров
Ф2	Volvo	Сидоров
Ф2	Opel	Иванов
Ф2	Opel	Петров
Ф2	Opel	Сидоров

Рис. 38. Многозначная зависимость в отношении

Эта необходимость отражает наличие в отношении многозначной зависимости. Здесь для каждого значения атрибута *NФ* имеется набор значений атрибута *Авто* и набор значений атрибута *Сотрудник*. Таким образом, в отношении верно ограничение:

если кортежи <ф, а1, с1> *и* <ф2, а2,с2> *присутствуют одновременно, то и кортежи* <ф, а1, с2> *и* <ф, а2, с1> *также присутствуют одновременно*.

Многозначная зависимость – это такая зависимость между атрибутами A, B, C некоторого отношения, в которой для каждого значения A имеется набор значений атрибута B и набор значений атрибута C. Однако входящие в эти наборы значения атрибутов B и C не зависят друг от друга.

Обозначение зависимости: $A \twoheadrightarrow B$, $A \twoheadrightarrow C$ или $A \twoheadrightarrow B|C$.

Имеющаяся функциональная зависимость в отношении «Фирма»:

НФ →Авто, *НФ → Сотрудник*.

Отметим, что это отношение находится в НФБК, т.к. есть только один потенциальный ключ. Однако явная избыточность порождает аномалии. Например, для добавления информации о поступлении автомобиля типа Ford, необходимо создать три новых кортежа, по одному для каждого сотрудника. Аналогичная ситуация возникнет и при обновлении, и удалении.

Многозначные зависимости бывают тривиальными и нетривиальными. Многозначная зависимость $A \twoheadrightarrow B$ отношения R считается **тривиальной**, если либо атрибут B является подмножеством атрибута A, либо $A \cup B = R$.

В рассмотренном отношении «Фирма» зависимость нетривиальная, т.к. ни одно из этих условий не выполняется.

Четвертая нормальная форма (4НФ). Отношение находится в 4НФ тогда и только тогда, когда оно находится в НФБК и не содержит нетривиальных многозначных зависимостей.

По-другому можно так определить 4НФ (Дейт):

Отношение находится в 4НФ, если оно находится в НФБК и все многозначные зависимости в нем фактически представляют собой функциональные зависимости от ее ключей.

Приведение к 4НФ состоит в декомпозиции таблицы путем выделения в новое отношение одного или нескольких участвующих в многозначной зависимости атрибутов вместе с копией одного или нескольких детерминантов.

Так, отношение «Фирма» разбивается на два, уже находящихся в 4НФ (рис. 39).

Автомобили

НФ	*Авто*
Ф2	Volvo
Ф2	Opel

Сотрудники

НФ	*Сотрудник*
Ф2	Иванов
Ф2	Петров
Ф2	Сидоров

Рис. 39. Отношения в 4НФ

В полученных отношениях остались многозначные зависимости *НФ* → *Авто* (в «Автомобили») и *НФ* → *Сотрудник* (в «Сотрудники»), но эти зависимости являются тривиальными.

Такая декомпозиция обосновывается следующей теоремой (Фейгин):

Пусть A, B, C являются множествами атрибутов отношения R{A, B, C}. Отношение R будет равно соединению его проекций {A, B} и {A, C} тогда и только тогда, когда для отношения R выполняется многозначная зависимость $A \twoheadrightarrow B \mid C$.

Отсюда следует достижимость четвертой нормальной формы: любое отношение может быть подвергнуто декомпозиции без потерь в эквивалентный набор отношений в 4НФ.

Среди реляционных отношений можно выделить такие, для которых невозможно выполнить декомпозицию без потерь на две проекции, но можно выполнить декомпозицию без потерь на три и более проекции.

Пример отношения (**R**), которое нельзя декомпозировать без потерь ни на какие две проекции (рис. 40).

R

X	Y	Z
1	1	2
1	2	1
2	1	1
1	1	1

⟹

R_1

X	Y
1	1
1	2
2	1

R_2

X	Z
1	2
1	1
2	1

R_3

Y	Z
1	2
2	1
1	1

Рис. 40. Декомпозиция отношения R на все возможные проекции

(по два атрибута)

Какую *пару* из полученных трех проекций **R_1, R_2, R_3** ни взять, исходного отношения **R** не получить с помощью естественного соединения. Например, соединения **R_1, R_2** или **R_1, R_3** или **R_2, R_3** дают лишнюю строку (рис. 41):

R_1 NATURAL JOIN R_2

X	Y	Z
1	1	2
1	1	1
1	**2**	**2**
1	2	1
2	1	1

R_1 NATURAL JOIN R_3

X	Y	Z
1	1	2
1	1	1
1	2	1
2	**1**	**2**
2	1	1

R_2 NATURAL JOIN R_3

X	Y	Z
1	1	2
1	1	1
1	2	1
2	**2**	**1**
2	1	1

Рис. 41. Соединение любых двух проекций отношения R дают лишнюю строку

Но исходное отношение **R** может быть полностью восстановлено соединением всех *трех* проекций (убедитесь самостоятельно).

Отношения, которые можно восстановить соединением n проекций (для $n > 2$), но нельзя восстановить соединением m проекций (для $m < n$), называют **n-декомпозируемыми**. Так, рассмотренное отношение **R** является 3-декомпозируемым, но не является 2-декомпозируемым.

В таких случаях имеет место так называемая *зависимость соединения*, которая устраняется с помощью пятой нормальной формы.

Зависимость соединения – свойство декомпозиции, которое вызывает генерацию ложных строк при обратном соединении декомпозированных отношений с помощью операции естественного соединения.

Обозначают зависимость соединения как ***{A_1, A_2, …, A_n}**, где A_1, A_2, …, A_n – подмножества атрибутов, образующих проекции, соединение которых дает исходное отношение. Так, зависимость соединения для **R** можно обозначить как *{XY, XZ, YZ}.

Возникает вопрос, а нужно ли декомпозировать отношение, если оно уже находится в 4НФ и, казалось бы, лишено аномалий? На самом деле, аномалии

имеются, если есть зависимость соединения.

Заметим, что факт: **R** равно соединению *трех* своих проекций эквивалентен утверждению:

Если пара <1, 1> есть и в $\mathbf{R_1}$, и в $\mathbf{R_3}$, и в $\mathbf{R_2}$, то и тройка <1, 1, 1> есть в **R**.

Это утверждение можно сформулировать в виде ограничения для **R**:

Если кортежи <1, 1, 2>, <2, 1, 1>, <1, 2, 1> присутствуют в **R**, то и кортеж <1, 1, 1> также присутствует в **R**.

Эти ограничения, вытекающие из зависимости соединения, провоцируют аномалии.

В рассмотренном отношении **R**, очевидно, первичным и единственным ключом является вся тройка атрибутов X, Y, Z, и оно находится в 4НФ. Но в отношении присутствуют аномалии (рис. 41).

R

X	Y	Z
1	1	2
1	2	1

- если вставляется кортеж <2, 1, 1>, то также должен быть вставлен кортеж <1, 1, 1>;
- обратное утверждение не верно;

R

X	Y	Z
1	1	2
1	2	1
2	1	1
1	1	1

- кортеж <2, 1, 1> может быть удален без побочных эффектов;
- если удаляется кортеж <1, 1, 1>, то должен быть удален и кортеж <2, 1, 1> или кортеж <1, 1, 2> или кортеж <1, 2, 1>

Рис.42. Аномалии вставки и удаления при зависимости соединения

Новые отношения $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, полученные в результате 3-декомпозиции отношения **R** такими аномалиями «не страдают», в них отсутствует нетривиальная зависимость соединения.

Зависимость соединения $*\{A_1, A_2, \ldots, A_n\}$ называется **тривиальной**, тогда и только тогда, когда одна из проекций $A_1, A_2, \ldots, A_n$ является проекцией, идентичной **R** (т.е. проекцией по всем атрибутам отношения **R**).

Пятая нормальная форма (5НФ) – отношение без нетривиальных зависимостей соединения.

Полной декомпозицией отношения называют такую совокупность произвольного числа его проекций, соединение которых полностью совпадает с этим отношением.

Определение 5НФ можно сформулировать по-другому:

Отношение находится в 5НФ тогда и только тогда, когда в каждой его полной декомпозиции все проекции содержат потенциальный ключ. Отношение, не обладающее ни одной полной декомпозицией, также находится в 5НФ.

Лабораторная работа №1.

Создание базы данных (БД) Access

1.1. Создание нового файла базы данных

Откройте приложение Access:

Пуск → Все программы → Microsoft Office 2013 → Access 2013.

В открывшемся окне предлагается выбрать шаблон для создания базы данных. Начиная с версии 2013 [4-6], можно создавать базы данных двух типов: «Пользовательское веб-приложение» или «Пустая база данных рабочего стола». Выберите второй вариант. В новом окне выберите папку, в которой будет сохранена база данных, в поле «Имя файла» введите название создаваемой базы: *Учебная* и нажмите кнопку «Создать» (рис. 43).

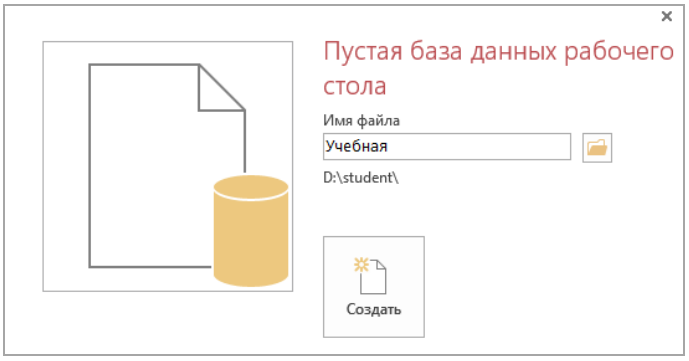


Рис. 43. Сохранение БД

Если потребуется открыть базу Access 2013 на компьютере, где отсутствует установленная полная версия Access 2013, можно воспользоваться бесплатным программным обеспечением «Среда выполнения Microsoft Access 2013» http://www.microsoft.com/ru-ru/download/details.aspx?id=39358.

После сохранения файла базы данных откроется главное окно программы. По умолчанию при открытии новой базы автоматически создастся и откроется новая таблица с именем «Таблица1». Начиная с Access 2007, графический интерфейс является ленточным (взамен текстовых меню и панелей инструментов в предыдущих версиях). Данный интерфейс получил название Microsoft Fluent: в верхней части окна имеется лента (рис. 44), разбитая на категории (вкладки) – рис. 44 п.2; на каждой вкладке присутствуют группы (рис. 44 п. 3) с элементами управления (кнопки, выпадающие списки и т.д. – рис. 44 п. 4). Лента содержит основные вкладки с группами наиболее часто используемых команд, контекстные вкладки, которые появляются только тогда, когда их использование допустимо. Некоторые кнопки на вкладках ленты предоставляют выбор действий, а другие позволяют выполнить определенную команду. Если какие-то команды не поместились на ленту и размещены в отдельном окне, то в группе отображается специальная кнопка (рис. 44 п. 5.)

Рис. 44. Обзор графического интерфейса Microsoft Fluent

В пользовательском интерфейсе Office Fluent реализована единая точка доступа ко всем возможностям системы Microsoft Office – «Представление Backstage». Режим Backstage появился в Access 2010. Он содержит команды и

сведения, применимые ко всей базе данных, например, «Сжать и восстановить», а также команды, которые в более ранних версиях содержались в меню Файл. Представление Backstage открывается при открытии вкладки ФАЙЛ (рис. 44 п. 1) или при запуске приложения Access (например, из меню "Пуск"), если при этом не открывается база данных.

Основное окно базы данных состоит из *Ленты* в верхней части окна, *Области навигации* в левой части, *Строки состояния* внизу и *Основной рабочей области* с вкладками.

Область навигации. При открытии имеющейся или создании новой базы данных имена объектов базы данных появляются в области навигации. К объектам базы данных относятся таблицы, формы, отчеты, страницы, макросы и модули. Эти объекты открываются из области навигации. На рис. 45 область навигации содержит только раздел «Таблицы», с одним объектом «Таблица1»; другие разделы не отображаются, так как пока ещё пусты. Если область навигации не отображается, выполните команду: ФАЙЛ → **Параметры → Текущая база данных** и в разделе Навигация установите флаг **Область навигации**.

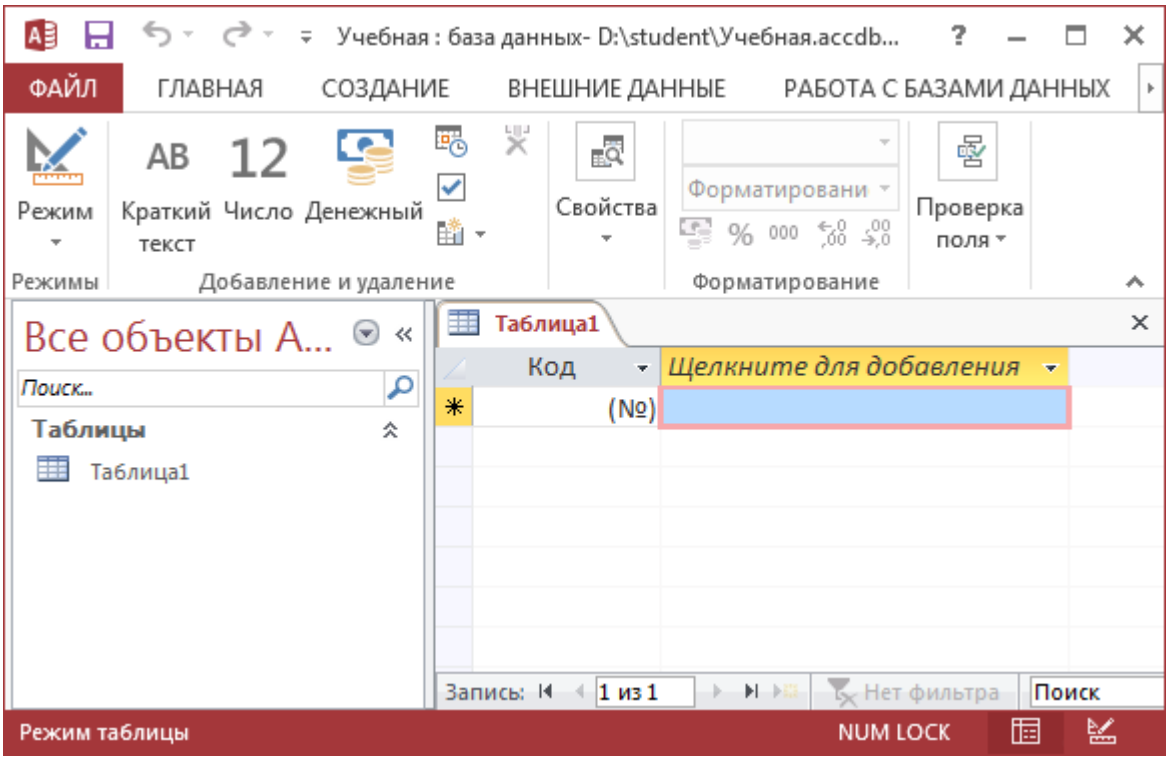


Рис. 45. Главное окно Access 2013

Строка состояния. Она может отображаться вдоль нижней границы окна. Этот стандартный элемент пользовательского интерфейса используется для отображения сообщений о состоянии, свойств, индикаторов хода выполнения и т. д. В Access строка состояния также предоставляет доступ к двум стандартным функциям, которые видны в строке состояния и в других программах Office: управление окнами и изменение масштаба.

С помощью элементов управления в строке состояния можно быстро переключать различные режимы просмотра активного окна (например, режим Таблицы или Конструктора для текущей таблицы). При просмотре объекта, который поддерживает изменение масштаба, можно регулировать степень увеличения или уменьшения с помощью ползунка в строке состояния.

Включить или отключить отображение строки состояния можно в диалоговом окне Параметры Access (ФАЙЛ → **Параметры → Текущая база данных → Строка состояния**).

Вкладки документов. Начиная с Office Access 2007, можно использовать для отображения объектов базы данных вкладки документов (рис. 46) вместо перекрывающихся окон. Отключение и включение вкладок документов осуществляется путем настройки параметров Access: ФАЙЛ → **Параметры → Текущая база данных**, в группе **Параметры** окна документа установите переключатель **Вкладки** и флаг **Вкладки документов**. Сохраните и повторно откройте файл базы данных.

На рис. 46 открыто 3 вкладки: таблицы «МоиСотрудники» и «Отделы» и запрос «Запрос1». Для закрытия вкладки можно использовать кнопку-крестик в правом верхнем углу или пункт «Закрыть» контекстного меню вкладки (щелчок правой кнопкой мыши по заголовку вкладки), или щелчок колёсиком мыши по заголовку вкладки. Если содержимое вкладки изменялось, то перед закрытием Access автоматически предложит сохранить изменения и, при необходимости, ввести имя объекта.

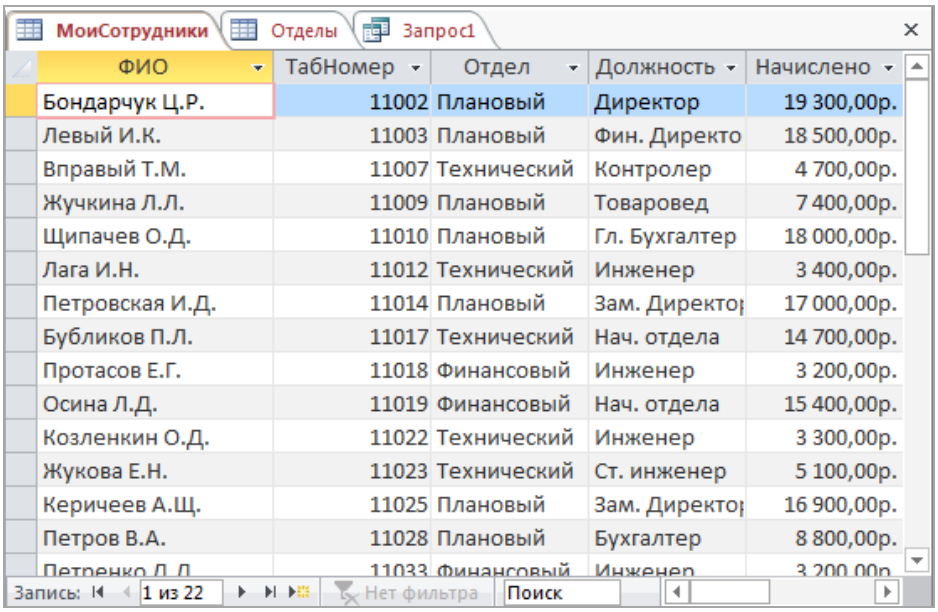


Рис. 46. Основная рабочая область Access в виде вкладок

1.2. Создание таблиц

Откройте на ленте Access: вкладка **Создание** → группа **Таблицы** → кнопка **Конструктор таблиц**. При создании новой БД автоматически открывается конструктор для таблицы с именем «Таблица1».

В режиме конструктора таблицы создается ее схема – задаются имена полей (атрибутов), их типы и свойства (условия на домен). Столбцы схемы заполняются построчно: каждая строка содержит сведения об одном атрибуте.

В первом столбце указывается название атрибута, во втором – его тип (тип домена), в третьем – необязательный комментарий. После выбора типа атрибута внизу открывается окно свойств, предоставляющее возможность задать ограничения на домен.

Имена полей не должны содержать знаков препинания, скобок, кавычек; желательно, чтобы они не содержали также и пробелов.

Зададим ***схему таблицы***, содержащей сведения о сотрудниках фирмы:

1. Введите в столбец **Имя поля** название первого атрибута – поля таблицы: *ФИО*.

2. В соседнем столбце **Тип данных** выберите в списке (он появляется при щелчке мышью в этом столбце) тип атрибута: *Короткий текст*.

3. В открывшемся окне **Свойства поля**, на вкладке **Общие**, в качестве значения свойства **Обязательное поле** выберите *Да*.

4. В следующем столбце схемы **Описание** можно (но не обязательно) ввести комментарий к этому атрибуту, например, *Фамилия и инициалы сотрудника*.

Далее переходим на новую строку – заполняем сведения о втором атрибуте таблице – табельном номере сотрудника.

В столбец **Имя поля** вводим название *Табномер*; в столбце **Тип данных** выбираем *Числовой*; в окне свойств отмечаем в списках: **Размер поля** – *Длинное целое*, **Обязательное поле** – *Да*, **Индексированное поле** – *Да (Совпадения не допускаются)*. Последние два свойства обязательны, если данный атрибут является потенциальным (возможным) ключом. В столбец **Описание** введите *Табельный номер сотрудника*.

Заполняем сведения о третьем атрибуте таблицы – отделе, в котором работает сотрудник.

В столбец **Имя поля** вводим название *Отдел*; в столбце **Тип данных** выбираем *Числовой*; в окне свойств отмечаем: **Размер поля**: *Целое*, **Правило проверки**: >0 And <=10, **Сообщение об ошибке**: *Такого отдела не существует*, **Обязательное поле**: *Нет*, **Индексированное поле**: *Нет*. В столбец **Описание** вводим *Номер отдела*.

Обратите внимание, что сообщение об ошибке возникнет на экране при попытке ввести данное, не удовлетворяющее заданному условию на значение. Для нашей таблицы сообщение «Такого отдела не существует» будет выдано в случае, если пользователь базы введет отрицательный номер отдела или номер, превышающий число 10.

Зададим следующий атрибут – должность сотрудника.

В столбец **Имя поля** вводим название атрибута: *Должность*, в столбце

Тип данных выбираем *Короткий текст*, в свойствах отмечаем **Обязательное поле**: *Нет*, **Индексированное поле**: *Нет*. В столбец **Описание** вводим текст: *Занимаемая должность*.

Вводим сведения о последнем атрибуте – зарплате сотрудника.

В столбец **Имя поля** вводим название атрибута: *Начислено*, в столбце **Тип данных** выбираем *Денежный*, в свойствах отмечаем: **Формат поля**: *Денежный*, **Правило проверки**: >0, **Сообщение об ошибке**: *Таких зарплат не существует*, **Обязательное поле**: *Нет*, **Индексированное поле**: *Нет*. В столбец **Описание** вводим: *Зарплата без налоговых и страховых вычетов*.

После того, как атрибуты определены, необходимо выбрать первичный ключ. Если этого не сделать, СУБД Access предложит вставить в таблицу свой ключ – дополнительный атрибут *Счетчик*, значениями которого являются номера строк. Кроме того, таблицу с неопределенным первичным ключом нельзя будет связать с другими таблицами.

Учитывая свойства уникальности и минимальности потенциальных ключей, в качестве первичного ключа выберем атрибут *Табномер*. Чтобы зафиксировать это свойство в схеме, выделите строку с описанием этого атрибута: подведите указатель мыши к крайнему левому столбцу окна таблицы и щелкните в этой строке. Затем нажмите кнопку 

с изображением ключа на ленте Access (вкладка **Конструктор**). Щелкните в другом поле, чтобы убрать выделение. В результате получим схему таблицы (рис. 47).

МоиСотрудники

Имя поля	Тип данных	Описание (необязательно)
ФИО	Короткий текст	Фамилия и инициалы сотрудника
ТабНомер	Числовой	Табельный номер сотрудника
Отдел	Числовой	Номер отдела
Должность	Короткий текст	Занимаемая должность
Начислено	Денежный	Зарплата без налоговых и страховых вычетов

Рис. 47. Таблица в режиме конструктора

Первичный ключ является по определению индексированным полем. Чтобы убедиться в том, что Access правильно зафиксировал ключ, нажмите кнопку 

(**Лента** → вкладка **Конструктор** → группа **Показать или скрыть** → кнопка **Индексы**). Откроется окно (рис. 48).

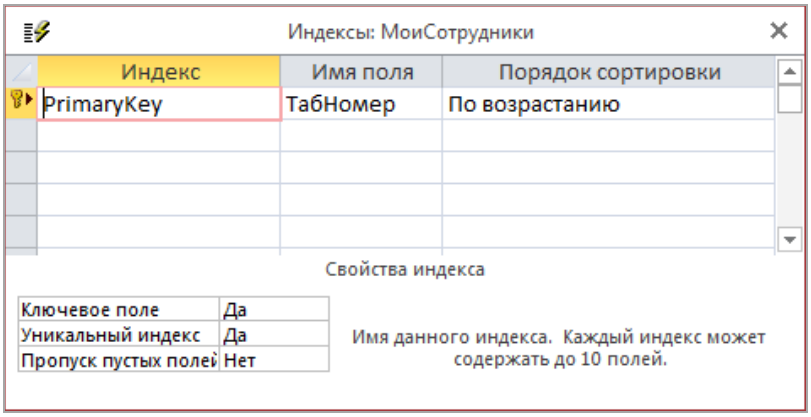


Рис. 48. Индексы таблицы

Строки таблицы индексов можно при необходимости заполнять вручную для других атрибутов, выбранных в качестве индексов.

Индексы обычно создаются также и для внешних ключей для ускорения поиска в связанных таблицах.

Созданную схему таблицы необходимо сохранить (обычным для Windows способом). В окне сохранения задайте имя таблицы «МоиСотрудники».

!! В будущем при открытии вашей базы данных или базы от известного источника при появлении предупреждения системы безопасности (жёлтая панель сразу под лентой Access – рис. 49) для корректной работы нажимайте кнопку **Включить содержимое**.

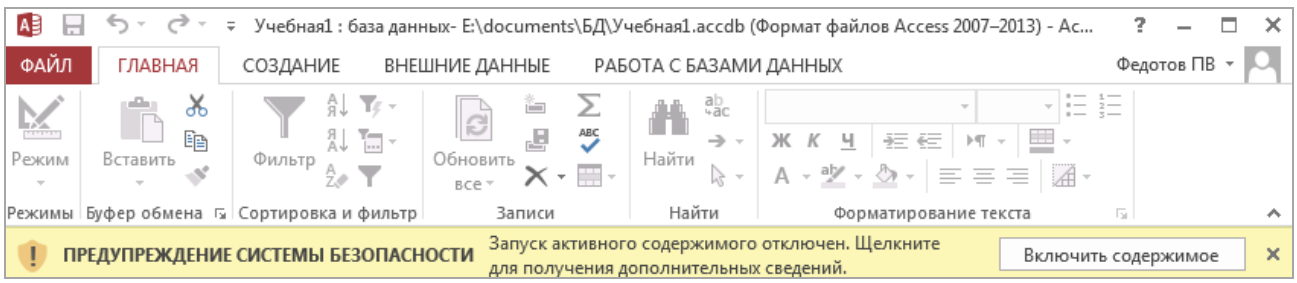


Рис. 49. Предупреждение системы безопасности при открытии файла БД

Теперь таблицу нужно заполнить данными – задать значения атрибутов.

Для ***заполнения таблицы*** перейдем из режима конструктора в режим таблицы. Для этого нажмите кнопку **Режим таблицы** 

.

Откроется окно, содержащее имена полей и свободную строку для ввода первой записи. Записи должны вводиться одна за другой (нельзя заполнять один столбец, потом другой по своему усмотрению). В каждой записи необходимо ввести значения первичного ключа и обязательных полей, иначе СУБД выдаст сообщение об ошибке и запретит дальнейший ввод. Записи добавляются в конец таблицы; в середину или в начало таблицы вставка невозможна.

Если первичным ключом является счетчик, то его значения заполняются автоматически системой.

В режиме таблицы данные можно не только вводить, но и изменять, и удалять. Для ***удаления записи*** ее необходимо выделить, щелкнув мышью на ее позиции в левом столбце таблицы, и затем нажать кнопку <Delete> на клавиатуре.

Перемещаться от одного поля к другому и от одного значения к другому можно с помощью мыши, клавиш <Enter>, <Tab>, и стрелок перехода на клавиатуре. Кроме того следующие сочетания клавиш осуществляют навигацию по таблице: <Ctrl> + <Home> – переход на первое поле первой записи, <Ctrl> + <End> – переход на последнее поле последней записи, <Ctrl> + <PgUp> – переход на начало текущей записи, <Ctrl> + <PgDn> – переход на конец текущей записи, <Ctrl> + <↑> - переход на начало текущего столбца, <Ctrl> + <↓> - переход на конец текущего столбца. Перемещаться по записям таблицы можно также с помощью специальной панели кнопок в нижней части окна таблицы (рис. 50).

Рис. 50. Панель навигации по записям таблицы

Регулировать ширину столбцов и строк можно с помощью мыши, подводя указатель к разделительной линии строк или заголовков полей и перетаскивая этот разделитель. Настроить ширину столбца по максимальной длине содержащегося в нем значения можно, как и в Excel, двойным щелчком мыши по правому разделителю этого столбца.

Заполните таблицу «МоиСотрудники» (рис. 51) следующими данными (в поле *Начислено* достаточно ввести только число, обозначение рубля «р.» добавляется автоматически).

Создайте самостоятельно еще одну таблицу – «Отделы» с полями: *Заведующий* (Короткий текст), *Отдел* (Числовой), *Название* (Короткий текст), *Телефон* (Короткий текст), *Характеристика* (Длинный текст), *Фото* (Поле объекта OLE).

ФИО	Табномер	Отдел	Должность	Начислено
Бондарчук Ц.Р.	11002	1	Директор	19 300,00р.
Левый И.К.	11003	1	Фин. директор	18 500,00р.
Вправый Т.М.	11007	3	Контролер	4 700,00р.
Жучкина Л.Л.	11009	1	Товаровед	7 400,00р.
Щипачев О.Д.	11010	1	Гл. бухгалтер	18 000,00р.
Лага И.Н.	11012	3	Инженер	3 400,00р.
Петровская И.Д.	11014	1	Зам. директора	17 000,00р.
Бубликов П.Л.	11017	3	Нач. отдела	14 700,00р.
Протасов Е.Г.	11018	2	Инженер	3 200,00р.
Осина Л.Д.	11019	2	Нач. отдела	15 400,00р.
Козленкин О.Д.	11022	3	Инженер	3 300,00р.
Жукова Е.Н.	11023	3	Ст. инженер	5 100,00р.
Керичеев А.Щ.	11025	1	Зам. директора	16 900,00р.
Петров В.А.	11028	1	Бухгалтер	8 800,00р.
Петренко Д.Д.	11033	2	Инженер	3 200,00р.
Штагер А.З.	11036	1	Бухгалтер	8 300,00р.
Ли А.А.	11043	3	Инженер	3 200,00р.
Алексеев В.В.	11044	2	Ст. инженер	5 100,00р.
Опенкин Р.П.	11045	1	Референт	1 800,00р.
Отмывалова Ю.Ю.	11049	2	Техник	1 950,00р.
Буйный А.Ю.	11050	1	Нач. отдела	16 300,00р.

Рис. 51. Значения атрибутов таблицы «МоиСотрудники»

В качестве первичного ключа выберите поле *Отдел*. Поскольку каждая строка этой таблицы характеризует один конкретный отдел, значения номера отдела не должны повторяться. Установите для поля *Отдел* такие же ограничения на домен, как и для поля *Отдел* из таблицы «МоиСотрудники»: в окне свойств отмечаем: Размер поля: *Целое*, Правило проверки: >0 And <=10, Сообщение об ошибке: *Такого отдела не существует*, Обязательное поле: *Да*, Индексированное поле: *Да (Совпадения не допускаются)* (поскольку в этой таблице атрибут *Отдел* является первичным ключом).

Заполните эту таблицу данными, взятыми из таблицы «МоиСотрудники»: в столбец *Заведующий* внесите фамилию и инициалы начальников отделов, в столбец *Отдел* - номера соответствующих отделов, значения поля *Телефон* заполните по своему усмотрению.

В качестве *Названий отделов* введите Плановый, Финансовый, Технический для номеров отделов 1, 2, 3 соответственно.

В поле *Характеристика* внесите краткие сведения о заведующем, например, «добрый, имеет троих детей» или «свиреп, но справедлив» и т.п. Тип Длинный текст означает, что поле предназначено для ввода текстовой информации большого размера (до 1 гигабайта, но контроль отображения полного текста ограничен первыми 64000 знаков). Отличается от типа Короткий текст тем, что в таблице находятся не сами данные, а ссылки на блоки данных, хранимые отдельно (для ускорения обработки таблиц).

Тип данных Поле объекта OLE содержит ссылки на ОЛЕ-объекты – изображения, графики или другие объекты ActiveX из другого приложения Windows – листы MS Excel, документы Word, звуковые данные, изображения. Для вставки в таблицу рисунка (поле *Фото*) необходимо для каждой записи, щелкнув по соответствующей ячейке таблицы правой кнопкой мыши, выбрать пункт контекстного меню **Вставить объект**… ; в открывшемся окне выбрать тип объекта и установить переключатель **Создать новый** или **Создать из файла**. Объект можно также просто перетащить мышью в поле таблицы. В

качестве объекта можно выбрать **BitmapImage**, тогда при установке переключателя **Создать новый** можно нарисовать портрет в приложении Paint. Либо выберите объект из списка Package. Выполните вставку объектов в поле *Фото* с помощью этих средств.

Возможный вид таблицы «Отделы» (рис. 52).

Отделы

Заведующи	Отдел	Название	Телефон	Фото	Характеристика
Буйный А.Ю.	1	Плановый	1111111	Package	Добрый, имеет троих детей
Осина Л.Д.	2	Финансовый	2222222	Bitmap Image	Та ещё мигера
Бубликов П.Л.	3	Технический	3333333	Package	Свиреп, но справедлив

Рис. 52. Таблица «Отделы»

1.3. Создание связей между таблицами

Таблицы «Отделы» и «МоиСотрудники» информационно связаны: в таблице «Отделы» речь идет об отделах, в которых работают сотрудники, указанные в таблице «МоиСотрудники». Каждой записи в таблице «Отделы» соответствуют несколько записей в таблице «МоиСотрудники» с тем же самым номером отдела. Таким образом, таблицы «Отделы» и «МоиСотрудники» связаны по номеру отдела, т.е. по полю *Отдел*. Такая связь называется «*один ко многим*». В данном случае в качестве таблицы со стороны «один» выступает «Отделы» (так называемая *родительская* таблица), со стороны «многие» – «МоиСотрудники» (*дочерняя*). Поле таблицы, участвующей в связи со стороны «один» всегда является *первичным ключом* этой таблицы. Поле таблицы, идущей со стороны «многие», является *внешним ключом*. Эти ключи должны иметь одинаковый домен.

Зафиксируем эту связь в Access. Выберите меню: **Лента Access** → вкладка РАБОТА С БАЗАМИ ДАННЫХ → группа **Отношения** → кнопка

Схема данных 

. С помощью появившегося окна **Добавление таблицы** добавьте в схему данных таблицы «Отделы» и «МоиСотрудники» (щелкая на их именах мышью и нажимая кнопку **Добавить**), после чего нажмите кнопку **Закрыть**. В окне **Схема данных** будут отображаться схемы («шапки») таблиц, содержащие имена их полей с выделенным полем – первичным ключом.

Для ***создания связи*** щелкните левой кнопкой мыши поле *Отдел* (по которому будет идти связь) в таблице «Отделы» и не отпуская перетащите его на поле *Отдел* таблицы «МоиСотрудники». Таким образом, *перетаскивание осуществляется от родительской таблицы к дочерней, от первичного ключа к внешнему ключу*.

В появившемся окне «Изменение связей» установите флажки **Обеспечение целостности данных**, **Каскадное обновление связанных полей**, **Каскадное удаление связанных записей**. Нажмите кнопку **Создать.**

Созданная связь будет изображаться в окне схемы данных в виде линии с метками на концах: рядом с полем Отдел таблицы «Отделы» указывается цифра «1», а рядом с полем Отдел таблицы «МоиСотрудники» – знак бесконечности «∞» (рис.53).

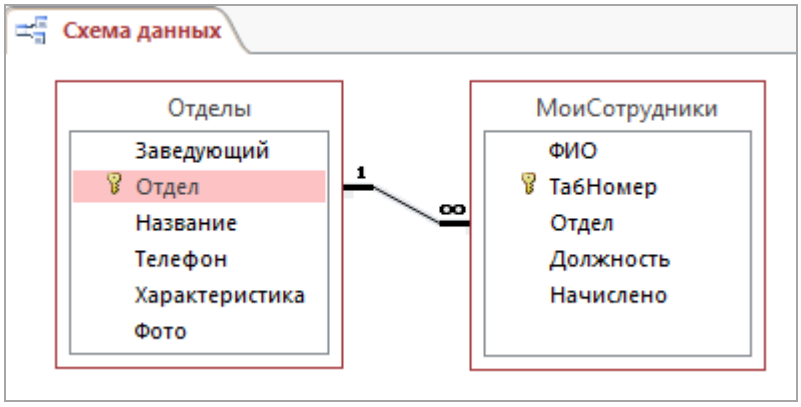


Рис. 53. Отображение связи между таблицами в окне схемы данных

Эти метки означают, что данная связь имеет тип «ОДИН КО МНОГИМ»: с одной («1») записью таблицы «Отделы» можно связать много («∞») записей из

таблицы «МоиСотрудники» (иными словами, в одном отделе может работать несколько сотрудников).

Установка флажков каскадного удаления и обновления связанных данных гарантирует, что за целостностью данных будет следить СУБД Access, при этом будет применяться стратегия CASCADE при обновлении и удалении записей.

Эта стратегия означает, что удаление записей из родительской таблицы влечет удаление всех связанных с ней записей в дочерней таблице. Обновление первичного ключа в родительской таблице приводит к обновлению соответствующих значений внешнего ключа в дочерней таблице. Значения внешнего ключа дочерней таблицы должны совпадать с какими-либо значениями первичного ключа родительской таблицы. Например, при попытке ввести в поле *Отдел* таблицы «МоиСотрудники» номер отдела, отсутствующий в таблице «Отделы», будет выдано сообщение об ошибке. Заметим, что установить связь с обеспечением целостности данных можно только между полями *одного и того же типа и размера* (в нашем случае поле связи имеет тип «Числовой» и размер «Целое»).

Эффект «каскада» применения этой стратегии возникает, если дочерняя таблица по отношению к какой-либо другой таблице является родительской. Например, если существует таблица «Имущество», фиксирующая информацию об объектах собственности каждого сотрудника, то, поскольку у одного сотрудника может быть несколько таких объектов, эта таблица выступала бы как дочерняя по отношению к таблице «МоиСотрудники». Тогда удаление отдела из таблицы «Отделы» повлекло бы автоматическое удаление всех его сотрудников из таблицы «МоиСотрудники» и далее – удаление всей информации о собственности этих сотрудников из таблицы «Имущество». В реальных базах данных такая каскадная цепочка удалений и обновлений может быть сколь угодно длинной.

Для ***удаления связи*** между таблицами следует щелкнуть мышью на линии, изображающей данную связь (при этом толщина линии увеличится),

нажать клавишу Del; либо щелкнуть правой кнопкой мыши по линии и выбрать **Удалить**. Далее следует подтвердить удаление в диалоговом окне, выбрав в нем вариант **Да.**

Для ***изменения свойств связи*** – повторного отображения окна «Изменение связей» надо выполнить двойной щелчок мышью на линии, изображающей данную связь, либо щелкнуть правой кнопкой мыши по линии и выбрать **Изменить связь….**

Сохраните схему данных и закройте окно схемы данных.

1.4. Создание поля подстановок

Для ускорения поиска в связанных таблицах значения внешних ключей определяют обычно как числовые, как коды истинных значений. Это создает неудобства при заполнении таких таблиц, т.к. пользователь должен «держать в голове» соответствие между этими кодами и понятными для него значениями. Поля подстановок используются для того, чтобы облегчить процесс заполнения таблиц. Если поле определено с подстановкой, пользователь видит на экране список понятных для него значений, делает нужный выбор, а в таблицу идет не само выбранное значение, а его код. Часто поля подстановок создаются для внешних ключей связанных таблиц-справочников.

Создадим поле подстановок для атрибута *Отдел* – внешнего ключа таблицы «МоиСотрудники». В результате пользователь при заполнении этой таблицы вместо введения номера отдела будет выбирать одно из названий отделов.

Поля подстановок делаются *до* связывания таблиц, поэтому придется вначале разрушить связь между нашими таблицами, а затем, *после создания поля подстановок* ее восстановить.

Шаги выполнения:

1. Откройте схему данных и удалите связь между таблицами «Отделы» и «МоиСотрудники» (см. раздел 1.3 на предыдущей странице).

2. Откройте таблицу «МоиСотрудники» в режиме конструктора, установите курсор на поле *Отдел* и в списке **Тип данных** выберите **Мастер подстановок.**

3. В первом окне Мастера подтвердим, что «**объект … будет использовать значения из таблицы или запроса**», т.к. в качестве подстановки мы будем использовать названия отделов, взятые из таблицы «Отделы». Нажмите **Далее.**

4. Во втором окне выберем таблицу – источник подстановок – «Отделы».

5. В следующем окне выберем поле – источник значений для подстановок, это – *Название*. Выделите его на левой панели и нажмите кнопку со стрелкой для переноса поля на правую панель. Нажмите **Далее.**

6. В новом окне укажите тип сортировки: выберите в первом списке поле *Название* и отметьте тип, например, по возрастанию. Нажмите **Далее.**

7. Задайте с помощью мыши ширину столбцов так, чтобы поместилось самое длинное название отдела. **Далее.**

8. В следующем окне оставим предлагаемое название поля – *Отдел*. Нажмите **Готово.**

Обратите внимание теперь, что свойства поля *Отдел* в таблице «МоиСотрудники» остались прежними – тип поля - *Числовой* со всеми заданными ранее ограничениями. Но вкладка **Подстановка** окна свойств этого поля изменила свое содержание – в качестве источника строк появился SQL-запрос на выборку поля *Название* из таблицы «Отделы».

Откройте таблицу «МоиСотрудники» в режиме таблицы, поставьте курсор на любую (возможно, и пустую) строку поля *Отдел* и убедитесь, что теперь при заполнении значений этого поля можно выбрать из предлагаемого списка название отдела вместо его номера.

☑ Несмотря на то, что в качестве значений поля *Отдел* мы видим названия, реально в таблице хранятся номера отделов, поэтому мы можем связать эту таблицу с таблицей «Отделы», как и прежде.

Откройте схему данных и восстановите связь между этими таблицами.

Лабораторная работа № 2.
Создание простых запросов на выборку

Запрос – это объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц. Для создания запроса можно использовать бланк конструктор запросов или написать инструкцию SQL. Можно создавать запросы на выборку, обновление, удаление или добавление данных. С помощью запросов можно также создавать новые таблицы, используя данные из одной или нескольких существующих таблиц [7].

С формальной точки зрения запрос можно представить одной или несколькими операциями реляционной алгебры, результатом запроса является реляционная таблица. В Access таблица – результат запроса – является временным объектом, существующим, пока не закрыт объект, его создавший. Однако можно при необходимости сохранить ее и как таблицу БД.

2.1. Средства создания запросов на выборку

Этот тип запросов используют наиболее часто. При выполнении запроса данные, удовлетворяющие условиям отбора, выбирают из одной или нескольких таблиц и выводят в стандартном «табличном» виде.

Рассмотрим несколько запросов, используя базу данных «Учебная».

Пример 1. *Выдать сведения о сотрудниках, фамилии которых начинаются на букву «П».*

Для реализации запроса выполните следующие шаги:

1. На ленте Access перейдите на вкладку СОЗДАНИЕ. Щелкните по кнопке **Конструктор запросов** в группе **Запросы**. В появившемся окне **Добавление таблицы** выделите на вкладке **Таблицы** таблицу «МоиСотрудники» и щелкните на кнопке **Добавить**, а затем **Закрыть**. На экране появится так называемый *бланк запроса* (окно конструктора запросов) с загруженной схемой таблицы «МоиСотрудники».

☑ Добавить в запрос новую таблицу можно и после перехода в режим конструктора запросов: для этого достаточно нажать кнопку **Отобразить таблицу** в группе **Настройка запроса** вкладки **Конструктор.** Можно выбрать команду **Добавить таблицу…** из контекстного меню (вызов правой кнопкой мыши по пустой области бланка).

2. Определите *поля запроса*: в строку Поле бланка нужно поместить те поля таблицы, которые содержат требуемые сведения, а также те, которые участвуют в условиях отбора; в нашем запросе – это *ФИО* и остальные поля. Дважды щелкните мышью по полю *ФИО* в схеме таблицы «МоиСотрудники», и оно появится в первом столбце бланка (поле *ФИО* можно перетащить мышью на любой столбец или выбрать в выпадающем списке столбца). Аналогичными действиями поместите в следующие столбцы бланка запроса поля *Табномер*, *Отдел*, *Должность*, *Начислено*.

3. Задайте *условие отбора*: в строку Условие отбора для поля *ФИО* введите строку: П* (введенный текст затем автоматически преобразуется в следующий: Like "П*"). Запрос в режиме конструктора примет вид как на рис. 54.

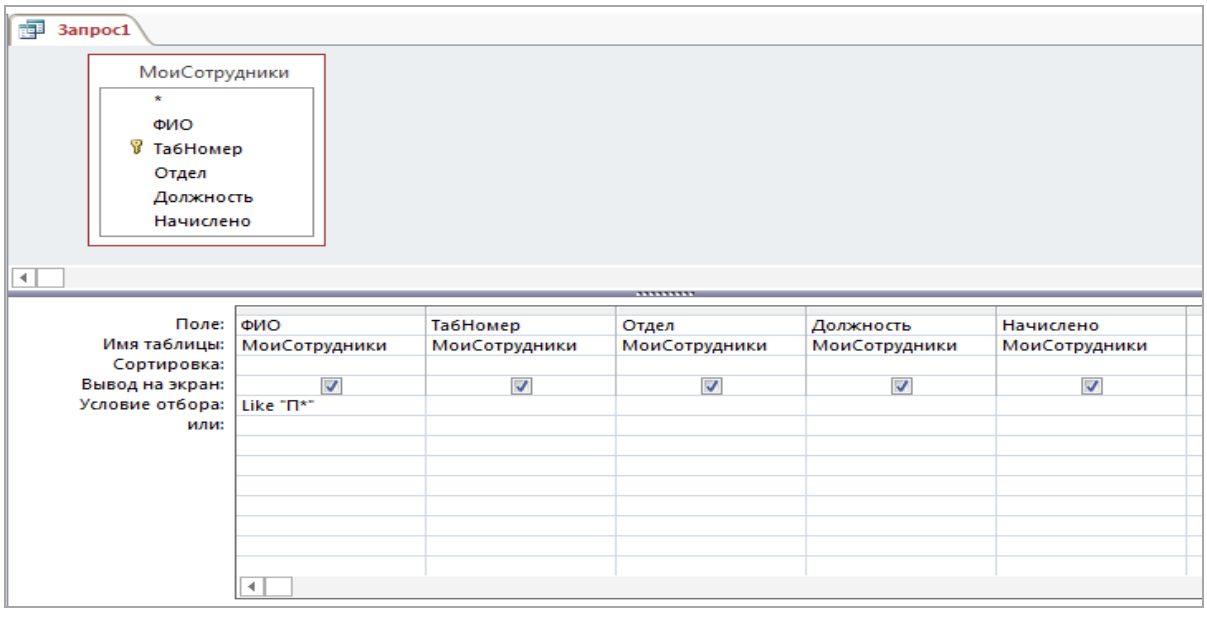


Рис. 54. Бланк запроса для примера 1

☑ Символ «*» используется в шаблонах (образцах) строк для обозначения любого количества любых символов; операция Like обеспечивает сравнение строки с шаблоном. В нашем случае это сравнение будет истинным в том случае, если поле *ФИО* начинается с буквы «П» (оставшаяся часть фамилии может быть произвольной). Например, шаблон *ск* позволяет отобрать только те строки, в которых содержится текст «ск», а с помощью комбинации операции отрицания Not и операции сравнения (Not Like "*ск*") можно отобрать строки, *не содержащие* данного текста.

В шаблонах строк можно также использовать специальный символ «?», обозначающий *ровно один произвольный символ*, и символ «#», обозначающий ровно одну *цифру* (табл. 1).

4. Запустите запрос *на выполнение*: нажмите кнопку 

(**Лента Access** → вкладка КОНСТРУКТОР → группа **Результаты** → кнопка **Выполнить**). В результате на экране появится таблица, столбцы которой соответствуют полям запроса (и располагаются в том же порядке), а строки являются записями, удовлетворяющими заданному условию.

Таблица 1

Подстановочные знаки Access 2013

Символы	Соответствия
? или _(в некоторых версиях Access)	Любой символ
* или % (в некоторых версиях Access)	Ноль или больше символов
#	Любая одна цифра (0–9)
[*список_символов*]	Любой один символ, входящий в *список_символов*
[!*список_символов*]	Любой один символ, не входящий в *список_символов*

Примечание. Диапазоны символов указываются при помощи «-». Например, [а-яА-Я0-9] соответствует любому буквенно-цифровому символу. Диапазон символов необходимо указывать в алфавитном порядке или порядке

возрастания. Например, [А-Я] – правильный, [Я-А] – неправильный шаблон.

Для возврата в режим конструктора запросов можно нажать кнопку  (**Лента Access** → вкладка ГЛАВНАЯ → группа **Режимы** → кнопка **Режим таблицы**).

5. Сохраните созданный запрос под предлагаемым именем «Запрос1».

☑ Шаг 2 можно было бы выполнить проще: дважды щелкнуть по полю ***** в схеме таблицы (***** *в строке* Поле *бланка обозначает вывод всех полей*) и отдельно – по полю *ФИО*, поскольку оно участвует в условии отбора. Для него тогда следует сбросить флажок **Вывод на экран**, чтобы *ФИО* не появилось дважды.

Пример 2. *Перечислить инженеров второго и третьего отделов*.

В данном запросе необходимо задать *составное* условие отбора: в строку Условие отбора для поля *Должность* введите слово: Инженер (кавычки будут поставлены автоматически), для поля *Отдел* введите условие: 2 Or 3 (что означает «второй или третий» отдел) (рис. 55).

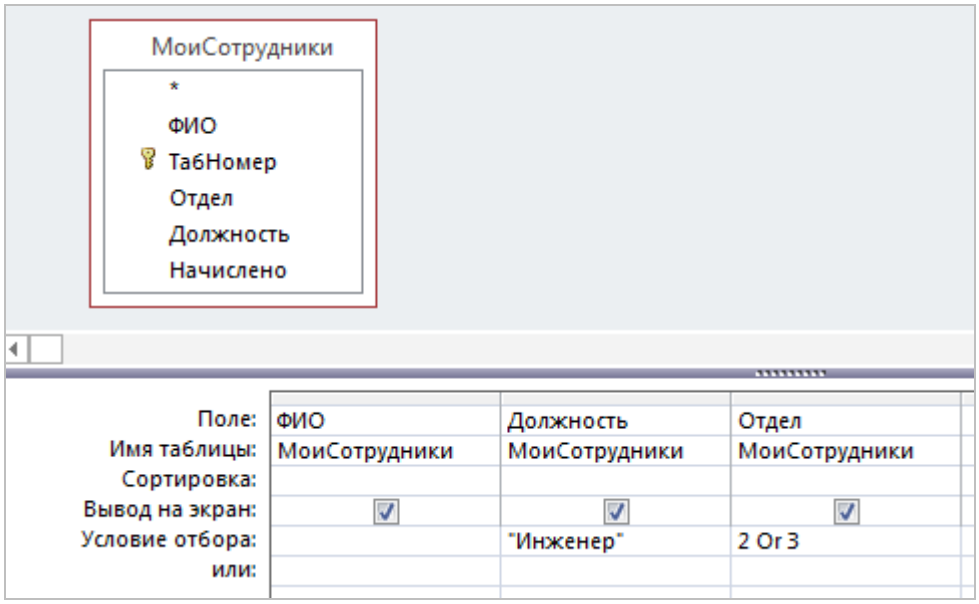


Рис. 55. Запрос в режиме конструктора для примера 2

‼☝ Обратите внимание, что, несмотря на то, что мы говорим «второго **и** третьего отдела», в условии отбора нужно ставить *в данном случае* операцию «или» (OR), т.к. логическая операция «и» (AND) выполняется, когда истинны оба операнда, т.е. здесь требовалось бы, чтобы сотрудник работал во втором и третьем отделе одновременно.

☑ Условия отбора, стоящие *в одной строке бланка запроса* связаны логической операцией «и» (AND), т.е. условие данного запроса можно прочитать так: Должность = «Инженер» И (Отдел = 2 ИЛИ Отдел = 3). Условия, стоящие *в разных строках*, связаны логической операцией «или» (следующий пример).

Пример 3. *Выдать сведения об инженерах третьего отдела или тех сотрудниках, у кого зарплата больше 5000 р.*

Условие отбора с операцией «или» в данном запросе касается разных полей, поэтому для поля *Начислено* частное условие отбора (>5000) поместите в строке «или», расположенной ниже строки Условие отбора (рис. 56). Это условие можно читать так: (Должность = «Инженер» И Отдел = 3) ИЛИ Начислено > 5000.

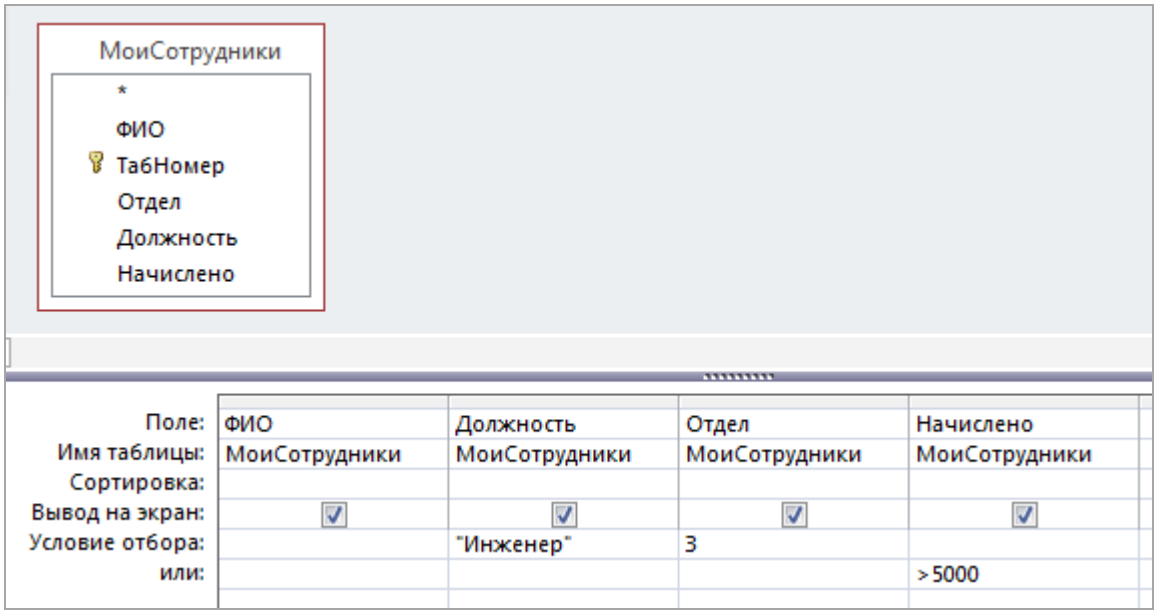


Рис. 56. Запрос в режиме конструктора для примера 3

Пример 4. *Представить фамилии тех сотрудников, которым начислено менее 2000 р. и тех, которым начислено более 7000 р.*

Есть два способа задать несколько частных условий для одного поля, связанных оператором Or. Можно ввести все условия в одну ячейку строки Условие отбора, соединив их оператором Or (рис. 57).

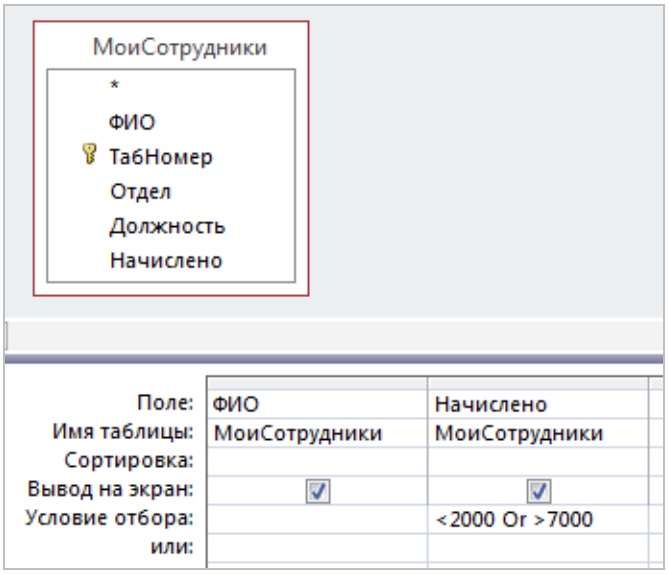


Рис. 57. Бланк запроса для примера 4 с вводом условия ИЛИ в одну строку

Другой вариант – ввести каждое частное условие в отдельную ячейку строки или (рис. 58).

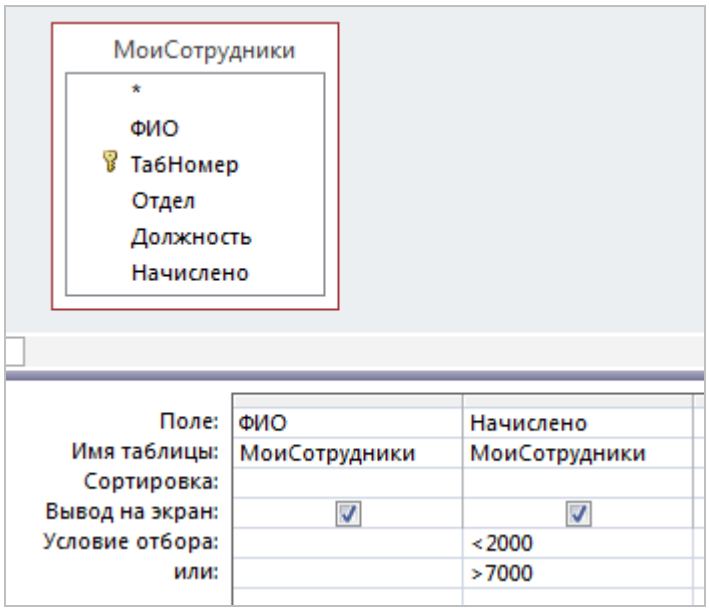


Рис. 58. Бланк запроса для примера 4 с вводом условия ИЛИ в две строки

Задания

1. Перечислите всех сотрудников первого отдела.

2. Определите должности сотрудников, фамилии которых начинаются с буквы «О» или «Л» и зарплата (поле *Начислено*) которых не превышает 4500 р.

3. Выдайте фамилии сотрудников первого или второго отделов, должность которых включает строку «бухг», а также сотрудников третьего отдела, зарплата которых превышает 3000 р.

4. Определите, в каких отделах работают сотрудники, зарплата которых лежит в диапазоне от 6000 до 9000 р.

Подсказка: В строке Условие отбора для соответствующего поля введите: >=6000 And <=9000 или используйте предикат Between, определяющий диапазон значений: Between 6000 And 9000.

5. Выдайте фамилии и табельный номер сотрудников отдела, начальником которого является Буйный.

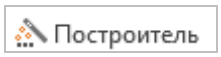
Подсказка: В данном запросе будут участвовать обе таблицы – «МоиСотрудники» и «Отделы», добавьте вторую таблицу. Проверьте: таблицы в бланке должны быть связанными; если это не так – создайте связь.

6. Выдайте должности сотрудников отдела, телефон начальника которого 277-88-99 (возьмите любой номер из своей таблицы).

7. Выдайте фамилии инженеров отдела, начальником которого является Бубликов.

2.2. Вычисляемые поля в запросах

Вычисляемыми полями называют поля, отсутствующие в таблице-источнике; их значения вычисляются непосредственно в запросе по заданному выражению. Выражение (формула) может включать в качестве операндов названия полей таблицы (или таблиц, вынесенных в бланк запроса), отдельные значения полей, статистические, математические и другие функции,

допустимые данной версией СУБД Access (перечень всех возможных функций и соответствующих операций можно увидеть, воспользовавшись *Построителем выражений*, открывающемся кнопкой **Построитель** 

на вкладке КОНСТРУКТОР в группе **Настройка запроса**).

Пример 5. *Создать вычисляемое поле «К выдаче», в котором будет вычисляться сумма, выдаваемая сотруднику с учетом 13% подоходного налога к «Начислено».*

Откройте новый бланк запроса, выбрав таблицу «МоиСотрудники» и выполните следующие действия:

1. *Определите вычисляемое поле* в запросе: перейдите на второй пустой столбец бланка запроса (в первый столбец внесите поле *ФИО*), введите в строку **Поле** текст: К выдаче:Начислено*(1-0,13), нажмите Enter. После нажатия Enter текст будет преобразован так: К выдаче: [Начислено]*(1-0,13), т.е. имена всех полей будут заключены в квадратные скобки; эти скобки можно было сразу указать при написании текста (рис. 59).

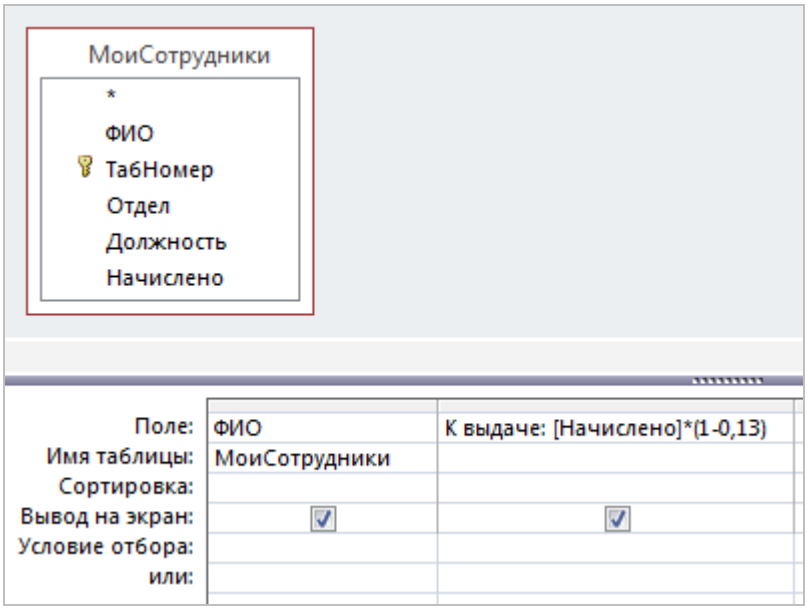
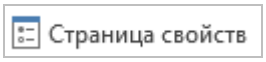


Рис. 59. Бланк запроса с вычисляемым полем для примера 5

☑ Таким образом, для определения *вычисляемого поля* (то есть поля, определяемого на основе других полей таблицы) в строку Поле бланка запроса

надо ввести имя вычисляемого поля, *двоеточие* и формулу, по которой данное поле вычисляется.

2. *Настройте свойства* вычисляемого поля*:* оставив курсор в столбце бланка К выдаче, нажмите кнопку 

в группе **Показать или скрыть** на вкладке КОНСТРУКТОР, либо нажмите на клавиатуре кнопку<F4>. В появившемся **Окне свойств** на вкладке **Общие** определите свойство **Формат поля** как Денежный; закройте окно свойств.

3. Выполните созданный запрос и сохраните его.

Задания

8. Создайте вычисляемое поле «Премия» с учетом 30%-ной премии.

9. Создайте вычисляемое поле «Без надбавки» с учетом 7%-ного штрафа.

2.3. Запросы с участием нескольких таблиц

Пример 6. *Выдать фамилии сотрудников, работающих в отделе, телефон которого начинается с цифр 67.*

В этом запросе должны участвовать две таблицы – «МоиСотрудники» (с информацией о сотрудниках) и «Отделы» (с информацией о телефонах).

Шаги выполнения:

1) Создайте запрос в режиме конструктора и выберите обе таблицы.

2) Выберите в бланк поле *ФИО* из таблицы «МоиСотрудники» и поле *Телефон* из таблицы «Отделы».

3) В столбце *Телефон* задайте условие отбора 67* и снимите флажок Вывод на экран.

4) Выполните запрос (рис.60).

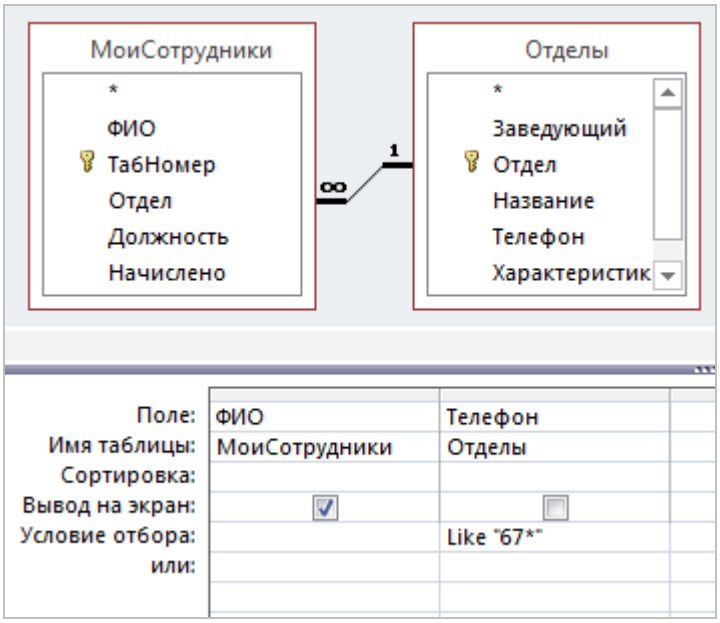


Рис. 60. Запрос с двумя источниками данных

Задания

10. Выдайте сотрудников, имеющих зарплату не выше 10 000 и работающих в отделах, заведующим которых не является Осина.

11. Выдайте инженеров отделов, телефон которых не начинается с цифр 12.

Лабораторная работа № 3. Создание итоговых запросов

3.1. Однотабличные итоговые запросы

Итоговые запросы позволяют вычислять значения для *групп* данных, «подводить итоги» по каждой группе записей. Например, вычислить по *каждому отделу* суммарную зарплату, найти максимальную премию для *каждой должности* и т.п.

Группы имеет смысл образовывать, если какое-либо поле в каждой группе имеет одно и то же значение, по этому полю и идет группировка. Например, группировка по полю *Отдел*: создаются группы записей, в каждой из которых значение этого поля (номер отдела) одинаково. В таблице «МоиСотрудники» три отдела, поэтому групп по этому полю будет также три: в

первую группу попадут все записи (кортежи целиком!) с номером отдела 1, во вторую – все записи с номером отдела 2, в третью – с номером отдела 3. Если группировка идет, например, по полю *Должность*, то группы образуют записи с одинаковым значением должности, групп будет столько, сколько разных должностей в таблице.

В итоговых запросах в общем случае нужно указывать: поле, по которому идет группировка (например, *Отдел*); поле, по которому вычисляется значение итоговой функции (например, *Начислено*), итоговую функцию (например, Sum). Каждая группа дает в результат запроса только одну запись с итогами по этой группе.

Если функция указана, а поле группировки – нет, то группой считается вся таблица, и итог вычисляется для всего столбца аргументов функции (например, Sum([Начислено])).

Если же в качестве поля группировки указан столбец, не имеющий повторяющихся значений, например, *ФИО*, то каждая запись таблицы тогда образует группу и итоговые вычисления теряют смысл (например, функция Sum([Начислено]) даст значение, равное значению поля *Начислено* для каждой записи).

Пример 7. *Укажите для каждого отдела среднюю зарплату и количество сотрудников в нем*.

Шаги выполнения:

1. Откройте новый запрос в режиме конструктора. Поместите в бланк запроса поля, участвующие в данном запросе: *Отдел* (поле группировки), *Начислено* (поле для вычисления зарплаты), *ФИО* (поле для вычисления количества сотрудников) из таблицы «Сотрудники».

2. На панели инструментов нажмите кнопку **Групповые операции** Σ или войдите в меню **Вид → Групповые операции**. В результате в бланке запроса появится строка Групповая операция, и для всех полей в данной строке будет

установлен вариант Группировка.

3. В поле *Начислено* в строке Групповая операция замените, используя выпадающий список, вариант Группировка на функцию вычисления среднего - Avg, а в поле *ФИО* – на функцию вычисления количества - Count.

Обязательные действия закончены, запрос можно выполнить. Однако результирующая таблица будет иметь «некрасивые» заголовки столбцов, соответствующих вычисленным итоговым значениям, и слишком «длинные» числа (выполните запрос и убедитесь). Поэтому отформатируем результат:

4. Измените названия полей *Начислено* и *ФИО* в бланке запроса на *Средняя зарплата* и *Количество работников*. Для этого во втором столбце бланка запроса перед словом *Начислено* наберите Средняя зарплата и поставьте двоеточие, нажмите Enter. Аналогичные действия выполните для третьего столбца (рис. 61).

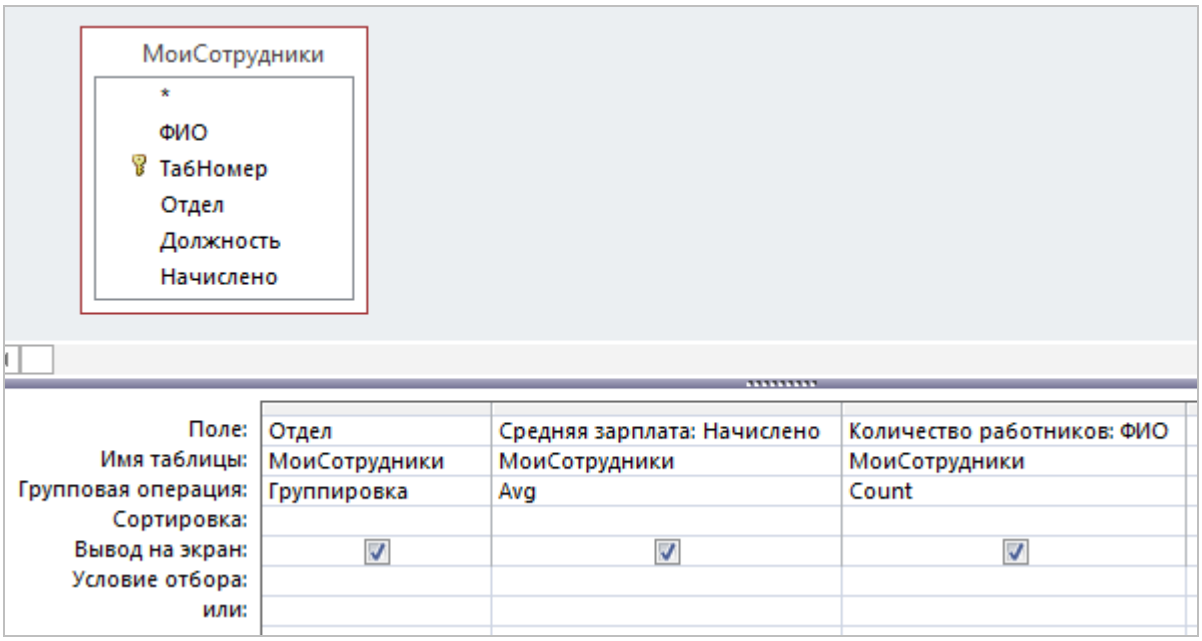


Рис. 61. Запрос с группировкой в режиме конструктора для примера 7

5. Поставьте курсор во второй столбец бланка запроса и измените свойства поля, задав **Формат поля** – Денежный.

6. Запустите на выполнение созданный запрос.

‼☝ Обратите внимание, что для определения количества используется функция Count , а не Sum. Функция Sum годится для суммирования числовых значений, в текстовом поле *ФИО* она бессмысленна.

☑ Функция Count на самом деле считает количество записей в таблице. Поэтому, вопреки ожиданиям, тот же результат был бы получен, если бы вместо *ФИО* вывести в запрос, например, *Должность*.

Задания

12. Посчитайте среднюю зарплату для каждой должности.

13. Укажите, сколько человек работает в отделе, начальником которого является Буйный.

14. Посчитайте среднюю зарплату в отделе, начальник которого – Осина.

15. Укажите максимальную зарплату в каждом отделе.

Пример 8. *Посчитать среднюю зарплату среди сотрудников 1-го и 2-го отделов, зарплата которых не превышает 7000 р.*

В этом запросе – два условия отбора на значения поля *Отдел* и на значения поля *Начислено*.

Если для поля, по которому идет вычисление, требуется выполнить еще и условие отбора, это поле (здесь – *Начислено*) нужно вывести в бланке запроса дважды: один раз с условием отбора и второй – с нужной функцией.

Шаги выполнения:

1. Создайте запрос конструктором, выведите поле *Начислено* дважды, поле *Отдел* – один раз.

2. Щелкните в поле *Отдел* и нажмите кнопку **Групповые операции** Σ.

3. В строке Условие отбора поместите соответствующие условия для поля *Отдел* (1 OR 2) и поля *Начислено* в одной из двух копий (<=7000).

4. В строке Групповая операция слово Группировка в этих двух столбцах замените на Условие (выбрав эту команду из выпадающего списка).

5. В строке Групповая операция слово Группировка во втором варианте поля *Начислено* замените на групповую операцию Avg.

Задания

16. Посчитать максимальную, минимальную и суммарную зарплату по каждой должности, исключая техника, директора и главного бухгалтера. Замените названия полей в результате запроса.

17. Посчитать минимальную зарплату в третьем и втором отделах, не включая должность референта.

18. Посчитать, сколько должностей в каждом отделе.

19. Посчитать количество сотрудников в первом и втором отделах, исключая Бондарчука, Петровскую и Керичеева.

20. Определить минимальную и суммарную зарплату по отделам для сотрудников, получающих не меньше 3000 р. и работающих не в первом отделе.

21. Определить среднюю и максимальную зарплату в отделах, не учитывая зарплату свыше 16500 р.

3.2. Групповые операции в многотабличных запросах

Групповые операции можно выполнять в запросах с несколькими таблицами. В этом случае система вначале создает таблицу-соединение (например, если таблицы связаны, то естественное соединение, если не связаны – декартово произведение), а затем производит группировку в указанных полях.

Пример 9. *Выдать количество сотрудников отдела, заведующий которого Бубликов.*

В этом запросе должны участвовать две таблицы – «МоиСотрудники» (с информацией о сотрудниках) и «Отделы» (с информацией о заведующих).

Шаги выполнения:

1. Создайте запрос в режиме конструктора, выберите в бланк указанные таблицы.

2. Из таблицы «Отделы» выберите поле *Заведующий*, а из таблицы

«МоиСотрудники» - поле *ФИО*.

3. Щелкните по полю *ФИО* и задайте группировку (нажмите Σ).

4. Замените в этом столбце Группировка на функцию Count.

5. В поле заведующий задайте условие отбора Бубликов* (т.к. мы не указываем инициалы).

6. Замените в этом столбце Группировка на Условие. При этом автоматически погасится Вывод на экран (рис. 62). Выполните запрос.

!! Обратите внимание, что в такой конструкции запроса после соединения таблиц выполняется условие отбора, в результате которого остается только отдел Бубликова, т.е. только *одна* группа сотрудников. Поскольку поле группировки (*Отдел*) не указано, функция Count считает количество по всему столбцу *ФИО*, из которого «чужие» сотрудники уже выброшены.

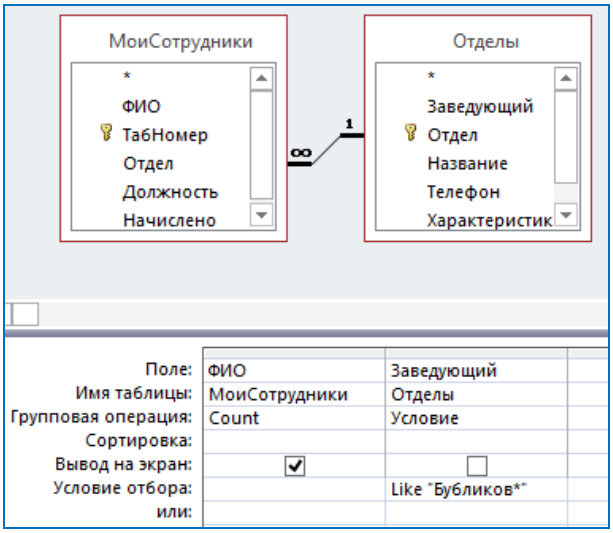


Рис. 62. Запрос с группировкой и двумя источниками данных

Задания

22. Посчитайте максимальную зарплату в отделе, телефон которого 678964 (выберите свой номер, имеющийся в таблице).

23. Выдайте среднюю зарплату в отделах, заведующим которых не является Буйный.

24. Выдайте суммарную зарплату в отделах, телефон которых не начинается с цифр 22.

Лабораторная работа № 4.
Создание запросов на основе других запросов

В качестве источника данных запроса может служить не только таблица БД, но и другой запрос, точнее, его результат. Необходимость в таких запросах возникает, когда нужные сведения извлекаются в два этапа, и второй – использует результаты первого. Первый, вспомогательный, запрос называют иногда *подзапросом*.

Построение одного запроса на основе другого – это также еще один способ работы с данными из нескольких таблиц: сначала создается один запрос, решающий одну задачу на основе данных из нескольких (или одной) таблицы, а затем строится другой запрос, использующий результаты первого и, возможно, другие таблицы.

☑ Таблицы БД и таблицы-результаты подзапросов, выбранные в бланк запроса, можно связывать непосредственно в бланке запроса, соединяя (перетаскивая из одной таблицы в другую) мышью названия одинаковых полей.

Пример 10. *Определить минимальную среди максимальных зарплат по каждому отделу.*

Поиск ответа предполагает два этапа: 1) найти максимальную зарплату в каждом отделе, и 2) найти минимум из полученных значений. Выполним эту задачу следующим образом:

1. Строим запрос на получение максимальных зарплат по каждому отделу. Поскольку максимум выбирается из группы чисел, соответствующих зарплатам сотрудников отдела, задаем запрос с группировкой, выбирая в качестве источника данных таблицу «МоиСотрудники», а в качестве итоговой функции – Max (рис. 63). Выполните запрос и сохраните его как «МаксЗарплаты».

2. Строим запрос на получение минимальной из этих зарплат. Это также запрос с группировкой, в качестве группы выступают все записи результата предыдущего запроса. Берем в качестве источника данных запрос «МаксЗарплаты», а в качестве итоговой функции – Min (рис. 64). Выполните запрос и сохраните его под именем «МинМаксЗарплата».

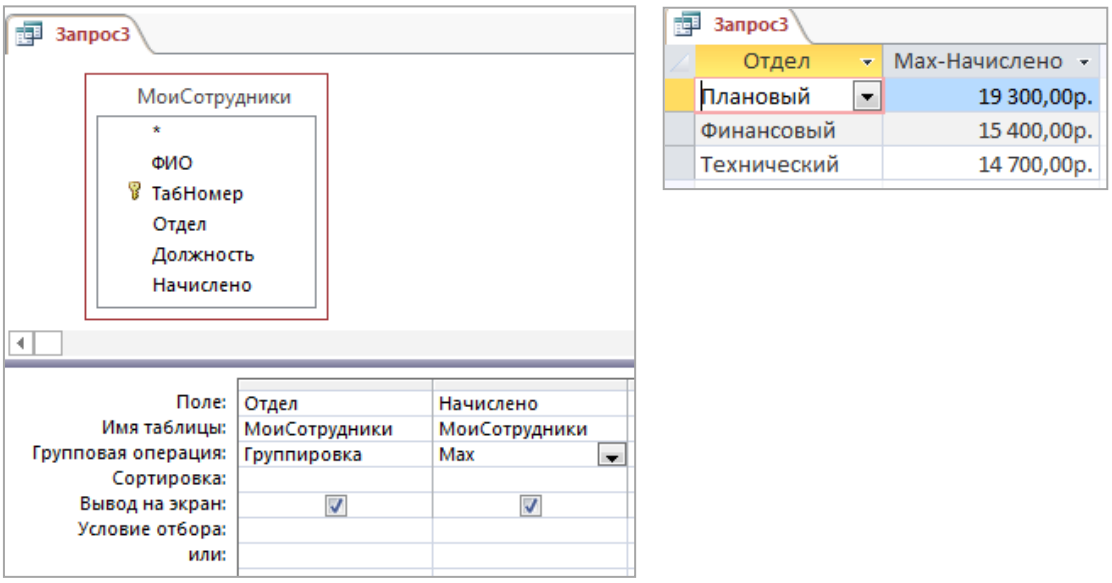


Рис. 63. Выборка максимальных зарплат по каждому отделу

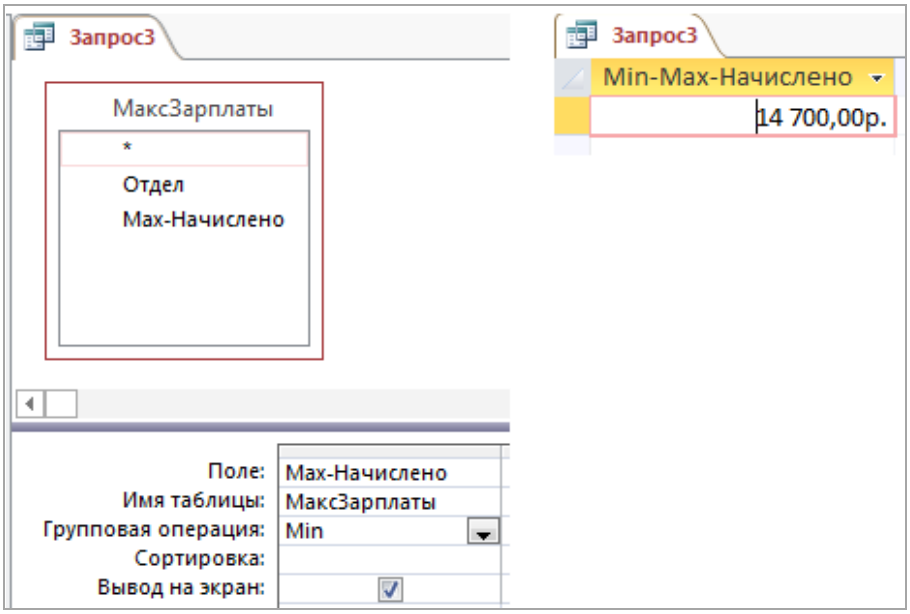


Рис. 64. Выборка минимальной среди максимальных зарплат в отделах

Пусть теперь требуется определить обладателя этой зарплаты, т.е. ответить на запрос:

Пример 11. *Выдать фамилию сотрудника, получающего минимальную из максимальных зарплат в отделах*.

Поскольку полученный минимум не вычисленное, а реальное значение, можно найти сотрудника, используя таблицу «МоиСотрудники» и запрос «МинМаксЗарплата», задав условие отбора:

[МоиСотрудники]![Начислено]=[МинМаксЗарплата]![Min-Max-Начислено],

или связать эти два источника данных в бланке запроса (рис. 65).

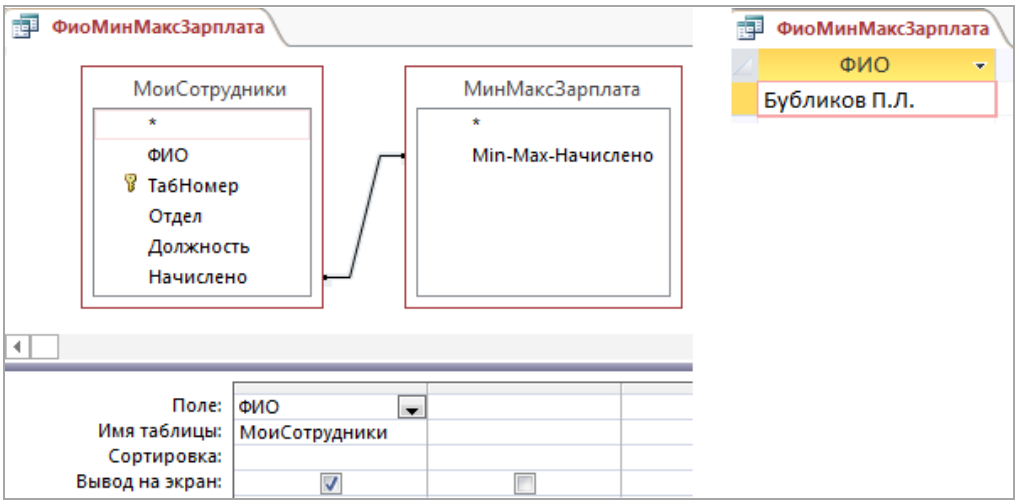


Рис. 65. Выборка сотрудника с минимальной среди максимальных зарплатой

Выполните запрос и сохраните результат под именем «ФиоМинМаксЗарплата».

Поставим теперь вопрос так:

Пример 12. *Определить номер телефона отдела, в котором работает этот сотрудник.*

Решение может быть таким (рис. 66).

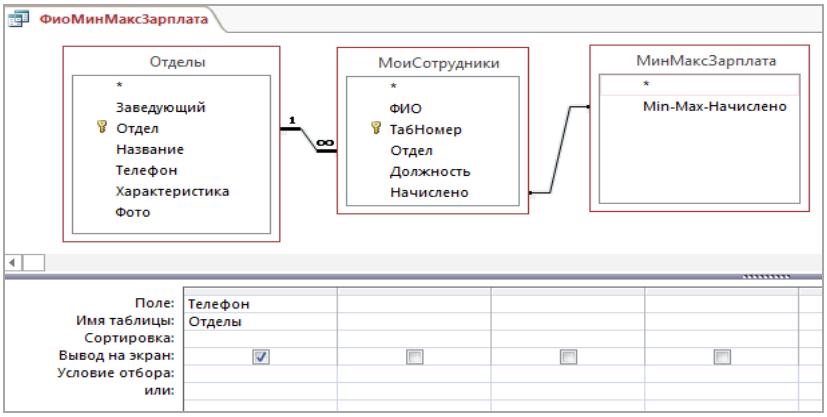


Рис. 66. Запрос со связанными разнотипными источниками данных

!!☝ Если бы таблицы «МоиСотрудники» и «Отделы» не были связаны, то их нужно было бы связать непосредственно в бланке запроса (иначе полученный телефон мог и не относиться к нужному отделу). Связывать по одинаковым полям в бланке можно любые источники данных, например, несколько запросов, таблиц и запросов, таблиц.

Задания

Создайте запросы:

25. Определить отделы, в которых работает более пяти человек.

26. Найти ФИО сотрудника, не являющегося директором и начальником отдела, обладающего максимальной зарплатой среди всех отделов, кроме первого.

27. Определить фамилии людей, у которых зарплата выше средней на 20%. При подсчете средней зарплаты исключить директора и финансового директора.

28. Выдать фамилию начальника отдела, у которого работает наибольшее количество сотрудников.

29. Выдать фамилию начальника отдела, в котором самая низкая средняя зарплата.

30. Выдать ФИО сотрудника, обладающего максимальной среди минимальных зарплат по отделам среди инженеров.

31. Выдать ФИО и телефон самого высокооплачиваемого начальника отдела.

32. Выдать фамилию и характеристику начальника отдела, в котором работает сотрудник с самой низкой зарплатой.

33. Выдать телефоны заведующих отделов, в которых работает меньше шести человек.

34. Выдать телефон и зарплату заведующего отделом, в котором минимальный оклад меньше 1800.

Лабораторная работа № 5.
Сложные запросы над несколькими таблицами

Использование учебной БД «Борей»

Для дальнейшей работы нам понадобятся несколько связанных таблиц, которые можно извлечь из учебной базы «Борей», шаблон которой можно

скачать по ссылке http://office.microsoft.com/ru-ru/templates/TC001228997.aspx. Откройте шаблон и сохраните БД с именем Борей.accdb.

Выполните следующие действия:

В окне вашей базы данных **лента Access** → вкладка ВНЕШНИЕ ДАННЫЕ → группа **Импорт и связи** → кнопка **Access**. В открывшемся окне «Внешние данные» нажмите кнопку **Обзор…** и укажите путь к файлу Борей.accdb, нажмите кнопку **ОК**. В окне «Импорт объектов» на вкладке **Таблицы** выделите таблицы:

- Customers (Клиенты);
- Заказы;
- Операции с запасами;
- Сведения о заказе;
- Сотрудники;
- Товары.

Аналогичным образом импортируйте запрос

- «Дополнительные сведения о клиентах».

Нажмите **ОК**. Проверьте, связаны ли полученные таблицы в вашей базе (**лента Access** → вкладка РАБОТА С БАЗАМИ ДАННЫХ → группа **Отношения** → кнопка **Схема данных**. Если в схеме данных этих таблиц нет, добавьте их, используя контекстное меню – щелчок правой кнопкой мыши по пустой области окна – Добавить таблицу… или контекстное меню – Отобразить все). Если таблицы оказались не связаны, удалите их и повторите процедуру импорта. Переименуйте таблицу Customers в Клиенты. Проверьте наличие связи между таблицами Клиенты и Заказы, если она отсутствует – добавьте связь между таблицами Клиенты (поле ИД) и Заказы (ИД Клиента). Откройте запрос «Дополнительные сведения о клиентах» и замените устаревшую таблицу Customers на Клиенты.

Внимательно рассмотрите схему данных (рис. 67). Обратите внимание, что каждый клиент в разное время делал несколько заказов (связь типа 1:M

между таблицами «Клиенты» и «Заказы»). Каждый заказ, в свою очередь, состоял из нескольких товаров, имеющих свою цену, заказанное количество и скидку (связь типа 1:M между «Заказы» и «Сведения о заказах»).

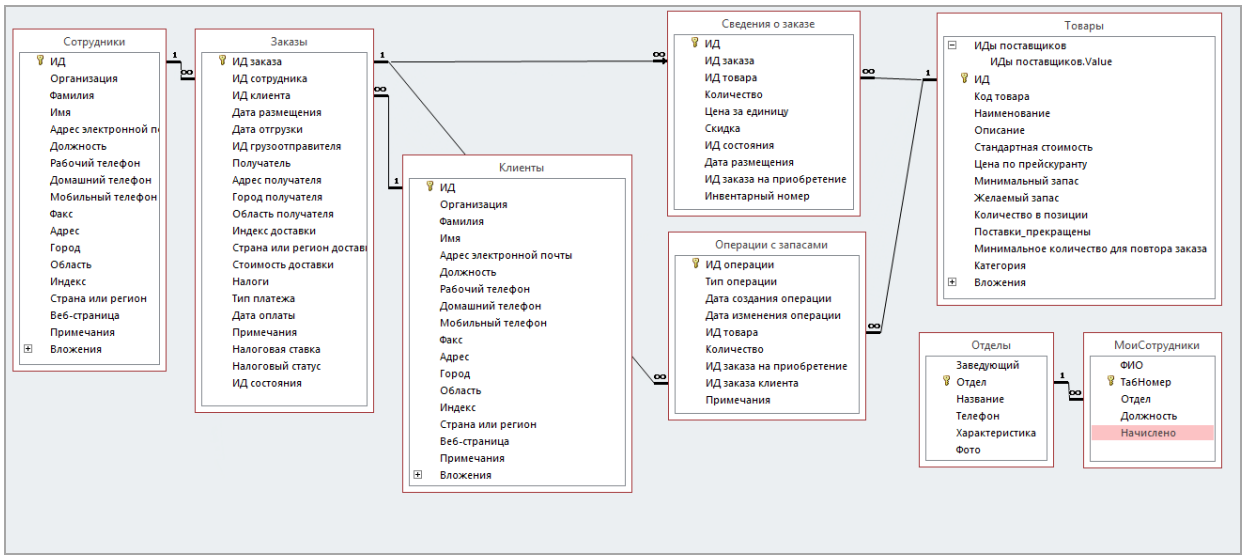


Рис. 67. Схема БД после импорта таблиц из учебной БД «Борей»

Пример 13. *Определить дату размещения каждого заказа, его стоимость и стоимость его доставки*.

Для выполнения этого запроса необходимо посчитать стоимость каждого заказанного товара: [Цена за единицу]*[Количество]*(1-[Скидка]) – эта информация содержится в таблице «Сведения о заказе».

Затем нужно просуммировать полученные данные для каждого заказа (группировка по *[ИД заказа]* с итоговой функцией Sum).

Далее нужно связать полученные данные с заказами из таблицы «Заказы» – именно там содержится информация о дате размещения каждого заказа и стоимости его доставки. Связь осуществляется по полю *[ИД заказа]*. Поскольку эта связь между таблицами уже установлена, нет необходимости связывать таблицы в бланке запроса. Шаги выполнения:

1. Создайте запрос в режиме Конструктора, выберите таблицы «Заказы» и «Сведения о заказе».

2. Выберите поля *[ИД заказа]*, [*Дата Размещения]* и [*Стоимость Доставки]* из таблицы «Заказы». Поле [*ИД заказа*] необходимо для поддержки связи.

3. В четвертом поле бланка наберите имя нового поля Сумма, поставьте двоеточие и наберите выражение Ccur([Цена за единицу]*[Количество]*(1-[Скидка])).

☑ Функция Ccur позволяет выводить значение в денежном формате.

4. Задайте группировку (нажмите Σ Итоги).

5. В этом же столбце вместо слова Группировка выберите итоговую функцию Sum (рис. 68).

6. Выполните запрос.

При выполнении появится ошибка «Недопустимое использование Null». Ошибка возникает из-за недопустимости использования в функции CCur null-значений (имеются заказы, для которых в таблице Сведения о заказе отсутствуют записи), для устранения ошибки добавьте в запрос поле Количество и установите Условие отбора: >0, у данного поля снимите флаг Вывод на экран.

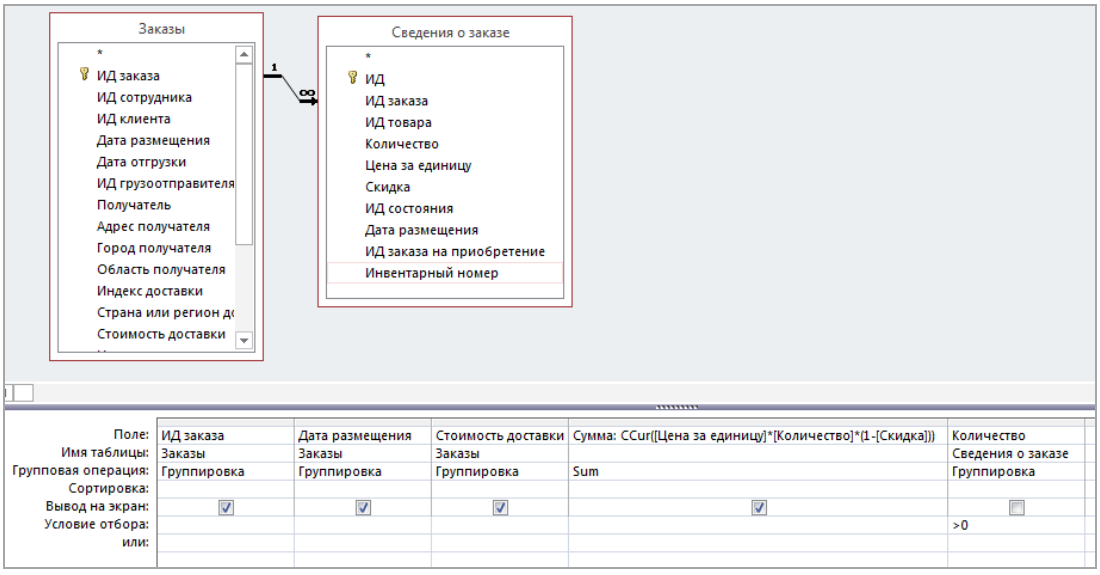


Рис. 68. Определение даты и стоимости каждого заказа

Пример 14. *Определить, сколько заплатил каждый клиент за все свои заказы, с учетом скидок и стоимости доставки.*

Этот запрос можно выполнить в два этапа (два запроса):

1) **Определяем стоимость каждого товара в одном заказе** и **суммарную стоимость товаров в заказе** (стоимость всего заказа). Эту часть можно выполнить, используя только таблицу «Сведения о заказе». Здесь необходимо сохранить *[ИД заказа]* для последующей связи (рис. 69). Сохраним запрос под именем «СтоимостьЗаказа».

2) Второй этап включает в себя две задачи: добавление к стоимости заказа стоимости доставки – это полная стоимость каждого заказа и суммирование этих стоимостей – это полные затраты клиента по всем своим заказам. Обе эти задачи можно выполнить в рамках одного запроса. Итак:

- **Определяем стоимость заказа вместе со стоимостью доставки.** Здесь в качестве источника данных потребуются предыдущий запрос и таблица «Заказы», содержащая поле *Стоимость Доставки* для каждого заказа. Эти источники связаны по коду заказа (если связь автоматически не проставлена, протяните ее). Вычисляем новое поле *Стоимость*, прибавляя к имеющейся сумме стоимость доставки. Пишем в пустой столбец заголовок поля, двоеточие и формулу:

Стоимость: [СтоимостьДоставки] + [Сумма].

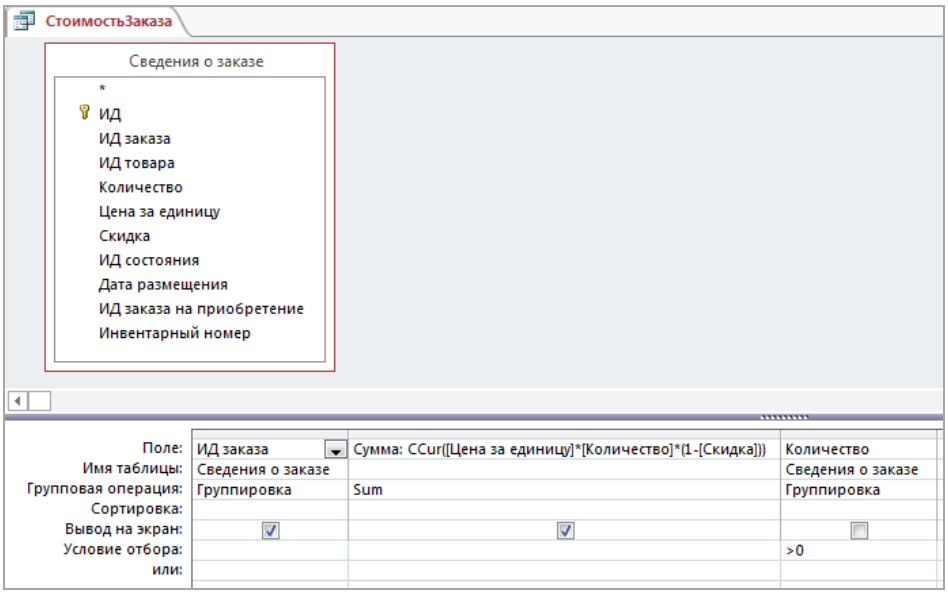


Рис. 69. Первый этап: определение стоимости каждого заказа

• **Определяем стоимость всех заказов для каждого клиента.** Выбираем поле *КодКлиента* из таблицы «Заказы» и выставляем группировку в этом столбце, выбирая Sum в качестве итоговой функции в новом столбце *Стоимость* (рис. 70).

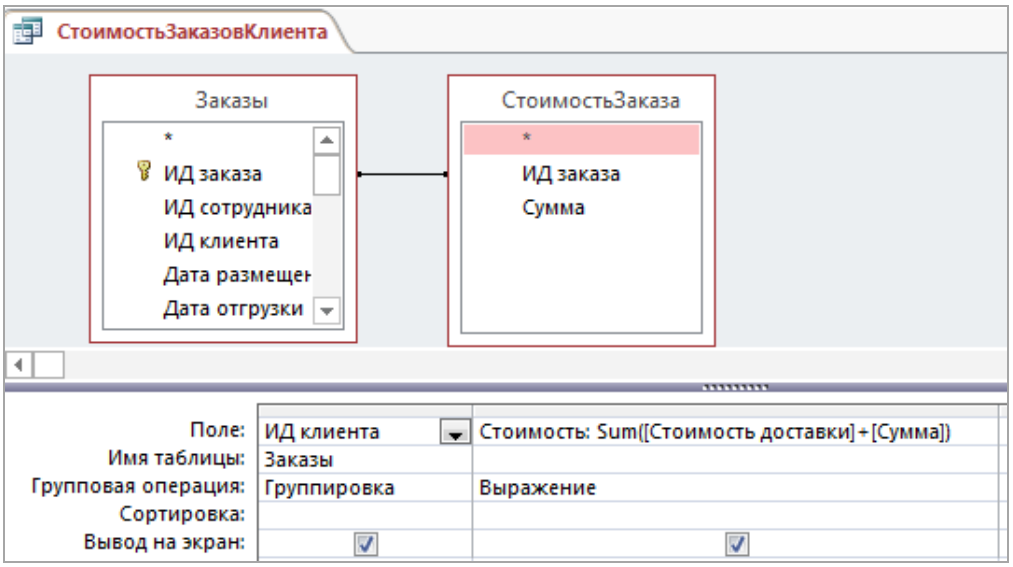


Рис. 70. Определение стоимости всех заказов для каждого клиента

Сохраните этот запрос под именем «СтоимостьЗаказовКлиента».

Пример 15. *Определить общую стоимость всех заказов для каждого города*.

В результате предыдущего запроса мы получили стоимость всех заказов каждого клиента. Просмотрев таблицу «Клиенты», убедимся, что в одной стране может находиться несколько клиентов, т.е. значения поля *Город* повторяются. Значит, для выполнения нового запроса нужно выполнить группировку по полю *Город*, суммируя стоимости всех заказов. Шаги:

1. Создайте запрос, выбрав в качестве источника данных таблицу «Клиенты» и запрос «СтоимостьЗаказовКлиента». Эти источники связываются по полю *ИД Клиента* (если этой связи нет, добавьте ее).

2. Выберите в бланк поля *Стоимость* из запроса и *Город* из таблицы.

3. В столбце *Город* поставьте группировку.

4. В столбце *Стоимость* выберите итоговую функцию Sum и наберите перед старым названием поля новое название: *СтоимостьПоГороду* (рис. 71).

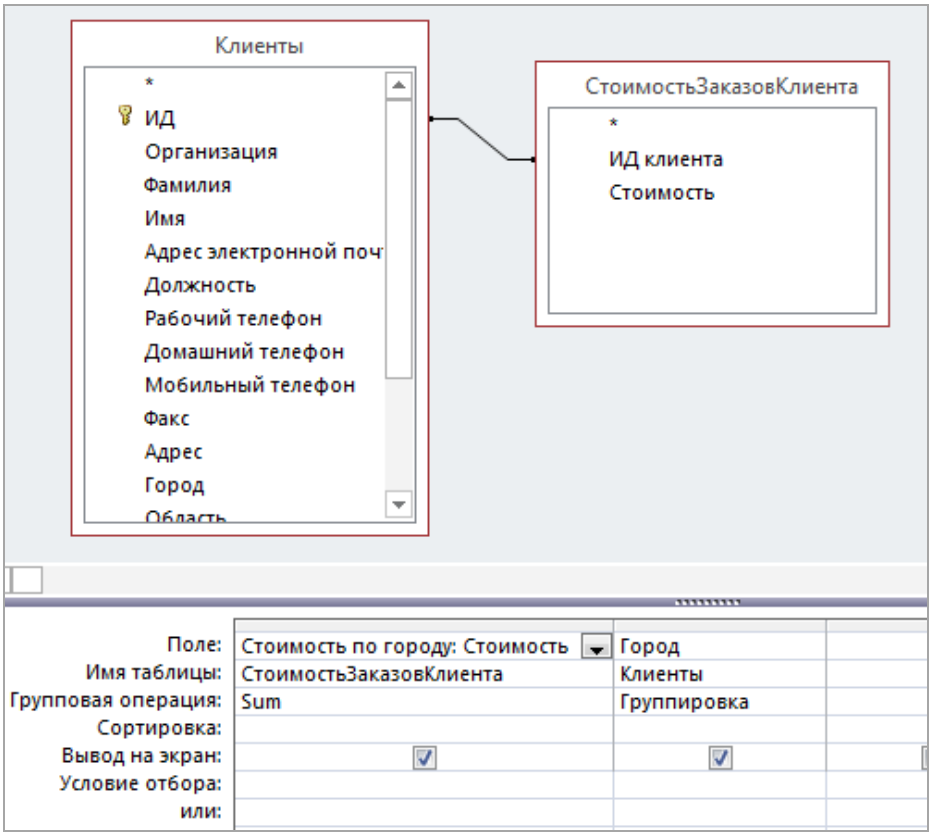


Рис. 71. Определение стоимости всех заказов по каждому городу

5. Выполните запрос (рис. 72).

СтоимостьПоГороду

Стоимость по городу	Город
3 312,50р.	Вологда
13 800,00р.	Воронеж
8 619,50р.	Казань
15 522,50р.	Курск
4 963,00р.	Москва
4 888,00р.	Омск
4 824,75р.	Орел
4 086,50р.	Пермь
2 300,50р.	Саратов
2 410,75р.	Сочи
3 105,00р.	Тюмень
2 571,00р.	Уфа

Рис. 72. Результат запроса Примера 15

Задания

35. Используя результат предыдущего запроса, определите город с максимальной стоимостью заказов.

36. Определите среднюю стоимость заказов для каждого клиента.

37. Определите полную стоимость заказов по каждому году их исполнения.

38. Определите для каждого года месяц с наибольшим количеством заказов.

39. Определите клиента с наименьшей общей стоимостью своих заказов.

40. Определите количество клиентов, общая стоимость заказов которых превышает 10 000.

Лабораторная работа № 6. Создание перекрестного запроса

Перекрестные запросы предназначены для получения итоговых расчетов в удобном для анализа виде. Они позволяют на основе реляционных таблиц базы данных формировать таблицы особого вида, в которых присутствуют заголовки не только столбцов, но и строк, составленные из значений полей исходной таблицы, а в ячейках располагаются соответствующие итоговые значения (аналог сводных таблиц в Excel).

Перекрестный запрос можно построить как с помощью Мастера, так и вручную, в режиме конструктора.

☑ Мастер строит перекрестные запросы только на основе одной таблицы БД. Если данные берутся из нескольких связанных таблиц, то можно: либо сформировать перекрестный запрос вручную, либо предварительно построить простой запрос на выборку данных из нескольких таблиц, а затем, взяв его в качестве источника данных, использовать Мастер.

6.1. Перекрестный запрос с одной таблицей

Пример 16. *Определить среднюю зарплату по каждой должности в каждом отделе.*

В рамках обычного запроса на выборку с группировкой можно вычислить среднюю зарплату либо по каждому отделу, либо по каждой должности, но не то и другое вместе. Перекрестный запрос позволяет это сделать.

Поскольку все необходимые данные содержатся в одной таблице «МоиСотрудники», проще использовать Мастер запросов.

Выполните следующие шаги:

1. Панель **лента Access** → вкладка **СОЗДАНИЕ** → группа **Запросы** → кнопка **Мастер запросов.** В окне **Новый запрос** → пункт **Перекрестный запрос.**

2. В первом окне Мастера «Создание перекрестных таблиц» необходимо выбрать источник данных; выберите таблицу «МоиСотрудники», нажмите кнопку **Далее**.

3. Во втором окне нужно выбрать поле этой таблицы, значения которого будут использованы в качестве заголовков строк. В нашем случае выбрать нужно либо поле *Отдел*, либо поле *Должность*. Для более наглядного представления итогов на экране в качестве заголовков строк лучше выбрать поле, у которого больше разных значений, т.е. *Должность*. Выделите это поле слева и нажмите кнопку со стрелкой для его переноса в правую область окна. Нажмите **Далее**.

4. В третьем окне выбираем поле, значения которого будут служить заголовками столбцов перекрестной таблицы. Выберите *Отдел*, **Далее**.

5. В четвертом окне выбираем поле, по которому подводится итог (проводятся вычисления), у нас – это *Начислено*. Выберите это поле мышью. Справа в этом же окне появится список итоговых функций, которые можно использовать для этого поля. Выберите **Среднее**, нажмите **Далее**.

6. В последнем окне задайте имя полученного запроса или оставьте предлагаемое. Переключатель **Просмотреть результаты запроса** должен быть включен, если запрос требуется выполнить. Нажмите **Готово**.

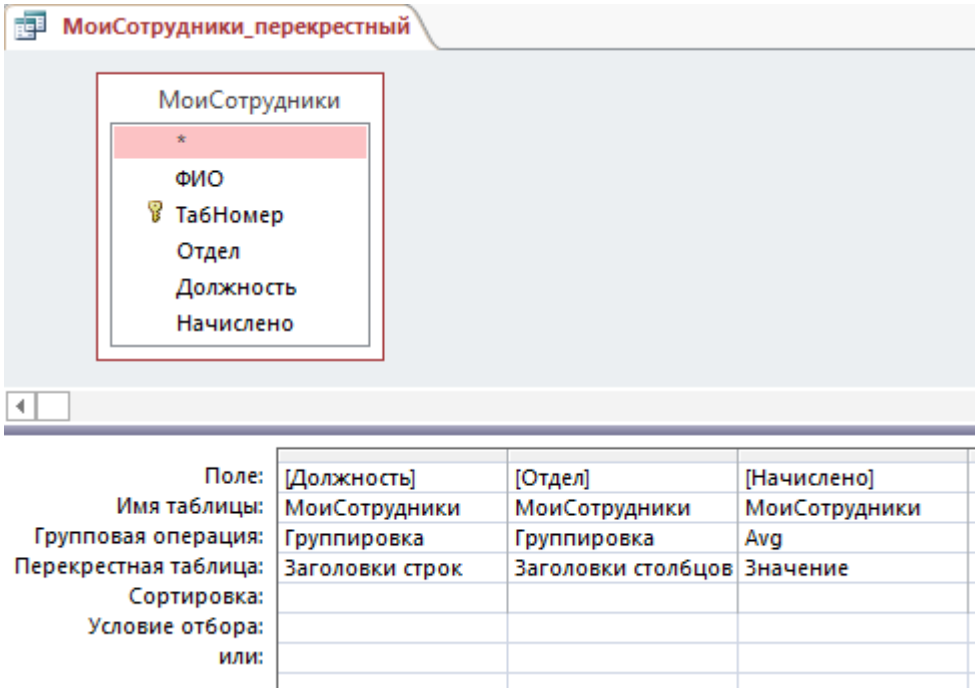
Полученная перекрестная таблица показана на рис. 73.

МоиСотрудники_перекрестный

Должность	Итоговое значение Начислено	1	2	3
Бухгалтер	8 550,00р.	8 550,00р.		
Гл. Бухгалтер	18 000,00р.	18 000,00р.		
Директор	19 300,00р.	19 300,00р.		
Зам. Директора	16 950,00р.	16 950,00р.		
Инженер	3 260,00р.		3 200,00р.	3 300,00р.
Контролер	4 700,00р.			4 700,00р.
Копия	100,00р.	100,00р.		
Нач. отдела	15 466,67р.	16 300,00р.	15 400,00р.	14 700,00р.
Референт	1 800,00р.	1 800,00р.		
Ст. инженер	5 100,00р.		5 100,00р.	5 100,00р.
Техник	1 950,00р.		1 950,00р.	
Товаровед	7 400,00р.	7 400,00р.		
Фин. Директор	18 500,00р.	18 500,00р.		

Рис. 73. Перекрестная таблица – результат перекрестного запроса

Здесь строки соответствуют должностям, столбцы – отделам, за исключением столбца *Итоговое значение Начислено*, в котором выведена средняя зарплата по всем отделам вместе. Если этот столбец (который мы не заказывали) не нужен, его можно удалить, открыв запрос в режиме конструктора (**Лента Access** → вкладка ГЛАВНАЯ → группа **Режимы** → кнопка **Конструктор** или пункт меню **Конструктор** контекстного меню вкладки). После удаления этого столбца запрос (в режиме конструктора) примет вид, идентичный тому, как если бы мы сформировали запрос вручную (рис. 74).



Поле:	[Должность]	[Отдел]	[Начислено]
Имя таблицы:	МоиСотрудники	МоиСотрудники	МоиСотрудники
Групповая операция:	Группировка	Группировка	Avg
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение
Сортировка:			
Условие отбора:			
или:			

Рис. 74. Перекрестный запрос в режиме конструктора

Для тренировки попробуйте создать аналогичный запрос Конструктором, определив максимальную зарплату в каждом отделе по каждой должности. Выполните для этого действия:

1. **Лента Access** → вкладка СОЗДАНИЕ → группа **Запросы** → кнопка **Конструктор запросов**, выберите в бланк таблицу «МоиСотрудники».

2. Выберите поля - участники запроса: *Отдел*, *Должность* и *Начислено*.

3. Лента **Access** → вкладка **КОНСТРУКТОР** → группа **Тип запроса** – кнопка **Перекрёстный** 

, в бланке появятся строки «Групповая операция» и «Перекрестная таблица».

4. В строке Перекрестная таблица для поля *Отдел* выберите из списка Заголовки столбцов, для поля *Должность* – Заголовки строк, для поля *Начислено* – Значение.

5. В строке Групповая операция для поля *Начислено* выберите Max.

6. Выполните запрос.

6.2. Перекрестный запрос с несколькими таблицами

Пример 17. *Определить месячные объемы продаж в 2006 г. по каждому виду товаров, используя таблицы БД «Борей».*

Для выполнения задачи потребуются таблицы «Товары», содержащая коды и наименования товаров и «Заказы», содержащая даты размещения заказов; но поскольку эти две таблицы связаны через посредника – таблицу «Сведения о заказе», то в бланк запроса необходимо включить и ее. Кроме того, эта таблица потребуется для вычисления объема заказов, т.к. содержит цены и заказанное количество товаров.

Шаги выполнения:

1. Создайте запрос Конструктором, выбрав в бланк указанные три таблицы. Из таблицы «Товары» включите в бланк поле *КодТовара* и поле *Наименование*. Из таблицы «Заказы» - поле *ДатаРазмещения*.

2. Лента **Access** → вкладка КОНСТРУКТОР → группа **Тип запроса** → кнопка **Перекрёстный**. В бланке появятся две новые строки.

3. В столбце поля *КодТовара* выберите в строке Перекрестная таблица элемент списка Заголовки строк. Для поля *Марка* сделайте то же самое.

4. В столбце поля *ДатаРазмещения* в строке Групповая операция выберите элемент списка – Условие. В строке Условие отбора введите условие: Like ”*.2006”.

5. Объем продаж по каждой дате можно определить, умножив цену на количество товара в заказе. Чтобы узнать объем по каждому месяцу полученные значения нужно просуммировать. Поэтому введите в последний (пустой) столбец бланка в строку Поле новое название и выражение:

Объем продаж: Sum([Количество]*[Цена за единицу])

В строке Групповая операция для этого поля выберите из списка Выражение, а в строке Перекрестная таблица выберите Значение.

6. В качестве столбцов формируемой перекрестной таблицы должны фигурировать месяцы 2006 года, т.е. дата размещения заказов должна быть выдана в специальном формате, например, «mmm», задающем названия месяцев в трехбуквенном сокращении. Для этого в последний (пустой) столбец введите в строку Поле новое название и выражение:

Месяцы: Format([Заказы]![Дата размещения];"mmm")

В строке Перекрестная таблица для этого поля выберите Заголовки столбцов. Окончательный вид бланка показан на рис. 75.

7. Выполните запрос. Обратите внимание, что результирующая таблица содержит названия месяцев, расположенные по алфавиту, а не в естественном порядке. Чтобы исправить эту ситуацию, можно в режиме Конструктора войти в окно **Свойства запроса** (открыв контекстное меню запроса и выбрав пункт **Свойства...** или Лента **Access** → вкладка КОНСТРУКТОР → группа **Показать или скрыть** → кнопка **Страница свойств** или нажмите на клавиатуре клавишу <F4>) и в строке Заголовки столбцов этого окна набрать вручную названия месяцев через точку с запятой (рис. 76).

8. Выполните скорректированный запрос.

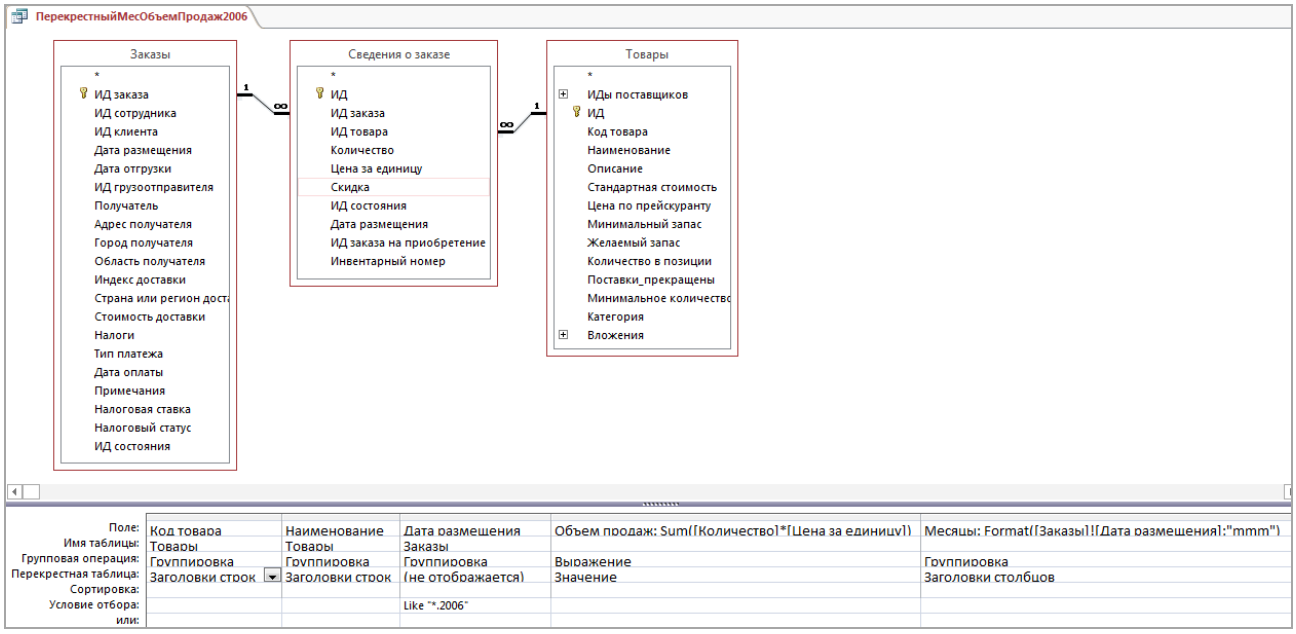


Рис. 75. Перекрестный многотабличный запрос

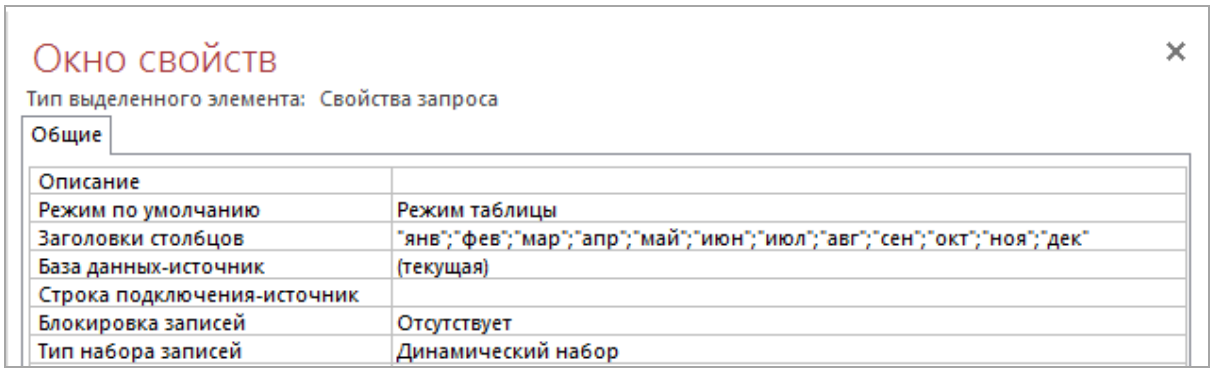


Рис. 76. Переопределение заголовков столбцов запроса

Можно изменить исходный запрос, потребовав не ежемесячные, а поквартальные объемы продаж. В этом случае нужно заменить формат даты размещения: в строке Поле в столбце *Месяцы* сотрите прежнее содержимое и внесите новые название и формулу:

Кварталы: Format([Заказы]![Дата размещения];"""Квартал ""q")

(перед символом **q** поставьте пробел). Войдите снова в окно **Свойства запроса** и сотрите названия месяцев в строке Заголовки столбцов.

Выполните новый запрос, получится результат (рис. 77).

☑ Перекрестные запросы нельзя задать операциями реляционной алгебры (и, соответственно, формулой реляционного исчисления). Они также не поддерживаются стандартом языка SQL для реляционных БД, однако входят в диалекты SQL, используемые отдельными СУБД, в частности, СУБД Access.

ПерекрестныйКварталОбъемПродаж2006

Код товара	Наименование	Квартал 1	Квартал 2
NWTB-1	Цейлонский чай	720	
NWTB-34	Пиво	1400	5418
NWTB-43	Кофе	29670	230
NWTB-81	Зеленый чай	822,25	
NWTBGM-19	Шоколадные бисквиты	552	230
NWTBGM-21	Лаваш		200
NWTCA-48	Шоколад	1402,5	1147,5
NWTCFV-17	Фруктовый салат		1560
NWTCM-40	Тихоокеанские крабы		2208
NWTCO-3	Сироп		500
NWTCO-4	Французская приправа	220	660
NWTD-72	Моцарелла		3132
NWTDFN-51	Сушеные яблоки	530	1590
NWTDFN-7	Сушеные груши	300	900
NWTDFN-74	Миндаль		200
NWTDFN-80	Сушеные сливы	210	52,5
NWTG-52	Длиннозерный рис		280
NWTJP-6	Ежевичный джем	250	2250
NWTJP-6	Мармелад		3240

Рис. 77. Перекрестная таблица запроса на выдачу поквартального объема продаж

Задания

41. С помощью перекрестного запроса определите, сколько человек работает по каждой должности у каждого заведующего отделом.

42. Постройте перекрестный запрос, используя таблицы БД «Борей», определив средний объем продаж по каждой категории товаров (из таблицы «Товары») по всем кварталам 2006 г.

43. Постройте перекрестный запрос, используя таблицы БД «Борей»: для каждого клиента определить максимальное количество заказанных товаров по каждой категории товаров. Сконструируйте запрос двумя способами: вручную - с помощью Конструктора, и Мастером, создав предварительно многотабличный запрос на выборку нужных полей и взяв этот запрос для Мастера в качестве единственного источника данных перекрестного запроса.

Лабораторная работа № 7. Запросы, использующие самообъединения таблиц и параметры

7.1. Запросы с самообъединением таблиц

Существуют запросы, касающиеся данных одной таблицы, на которые невозможно ответить, используя один экземпляр этой таблицы. Например:

Пример 18. *Определить фамилии сотрудников, имеющих одинаковую зарплату*.

Выбрав в бланк таблицу «МоиСотрудники», мы не сможем построить выражение над полем *Начислено*, задающее равенство значений. Однако есть способ, позволяющий перечислить попарно сотрудников с одинаковой зарплатой. Для этого нужно выбрать таблицу «МоиСотрудники» в бланк дважды и связать оба экземпляра по полю, обладающему равными значениями, в данном случае – по *Начислено*. Такого рода связи называются *самообъединением*.

Соответствующий бланк приведен на рис. 78.

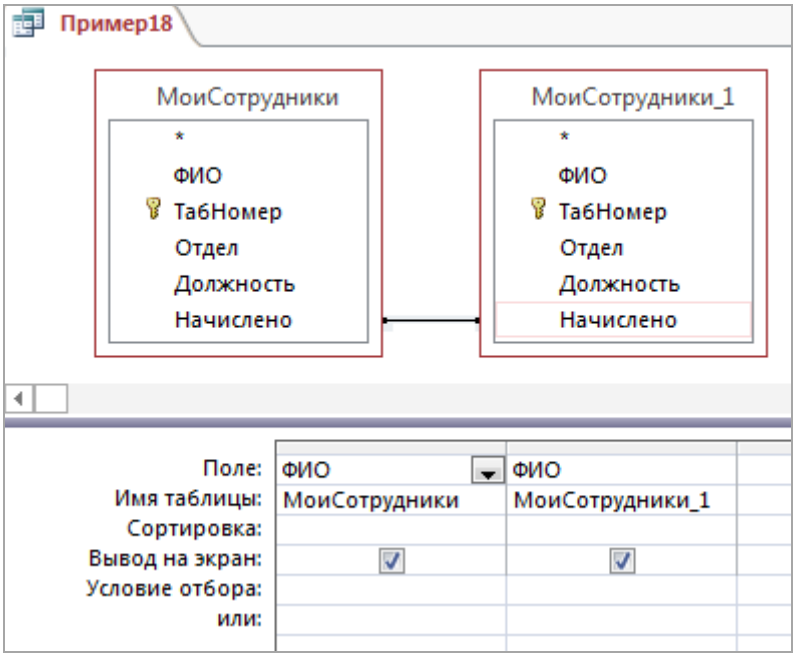


Рис. 78. Самообъединение таблиц в запросе

Обратите внимание, что новый экземпляр таблицы имеет имя «МоиСотрудники_1».

Можно было бы не связывать таблицы в бланке, но тогда следовало бы выбрать в бланк поле *Начислено* из одной из таблиц, например, из «МоиСотрудники» с условием отбора: = МоиСотрудники_1.Начислено, и сбросить с этого поля флажок вывода на экран (рис. 79).

Связывание таблиц в запросе – это способ избежать громоздких условий отбора на равенство значений полей.

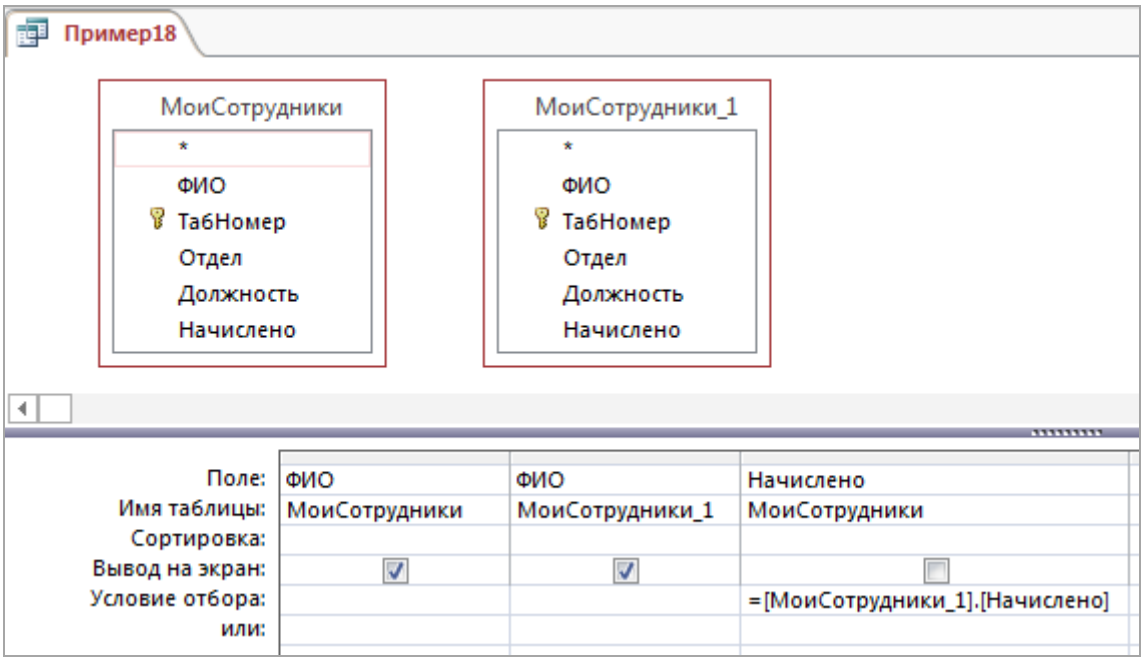


Рис. 79. Запрос на равенство значений

Выполнив любой из этих запросов, убедимся, что результат запроса страдает «дурным» избытком - содержит пары с одинаковыми частями (например, «Иванов | Иванов»). Запрос можно улучшить, убрав эти очевидные равенства с помощью условия отбора в любом из столбцов с *ФИО*. Например, в столбец поля *ФИО* из таблицы «МоиСотрудники» можно включить условие отбора: <>[МоиСотрудники]![ФИО].

Однако результат запроса будет еще содержать одинаковые по существу пары вида «Иванов | Петров» и «Петров | Иванов». Чтобы убрать и этот недостаток, можно отсортировать исходную таблицу в запросе по полю *ФИО* и задать в нем Условие отбора: >[МоиСотрудники]![ФИО] (рис. 80).

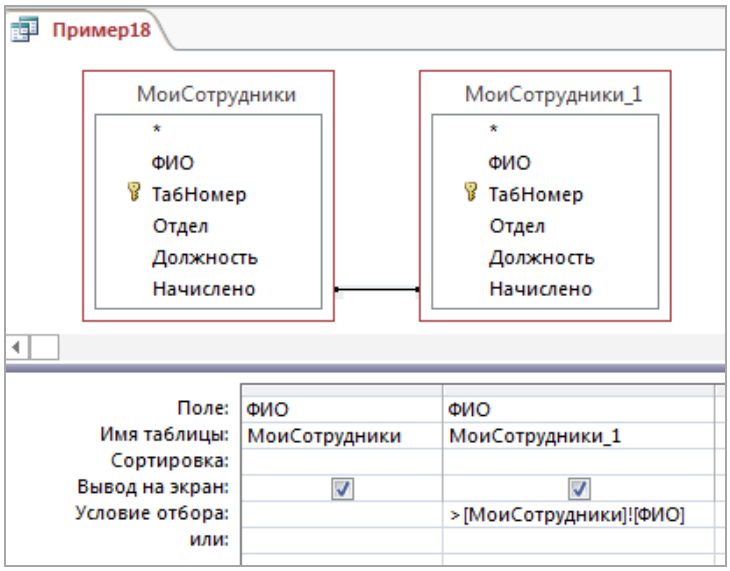


Рис. 80. Исключение тавтологий

Задание

44. Постройте запрос, используя самообъединение: определить различные пары сотрудников, имеющих одинаковые должности.

7.2. Запросы с параметрами

Введение параметра позволяет в рамках одного запроса (одного бланка запроса) определить множество однотипных запросов, отличающихся друг от друга значением параметра. Параметром могут служить несложные условия отбора или части условия, имена таблиц и полей таблиц в вычисляемых выражениях, имена полей таблиц, выносимых в бланк при формировании запроса, части (например, числа) вычисляемых выражений и т.п.

Место параметра в запросе выделяется квадратными скобками, внутри которых можно поместить любой текст, например, просьбу ввести какое-либо значение. При выполнении такого запроса на экране возникает окно с этим текстом и полем ввода, в которое пользователь записывает нужное значение. Это значение подставляется системой вместо параметра и затем запрос исполняется обычным образом.

В одном запросе может быть несколько параметров. Тогда окно ввода появляется для каждого из них.

☑ Если появляется окно ввода значения параметра, который вы не определяли, проверьте правильность написания имен полей в выражении или присутствие лишних квадратных скобок – система любое неожиданное содержимое квадратных скобок воспринимает как текст приглашения к вводу значения параметра.

Пример 19. *Выбрать из таблицы «МоиСотрудники» информацию о людях, начальная буква фамилии которых, отдел, зарплата и еще одно выводимое поле определяются параметрами.*

Заполните в режиме Конструктора бланк, как показано на рис. 81.

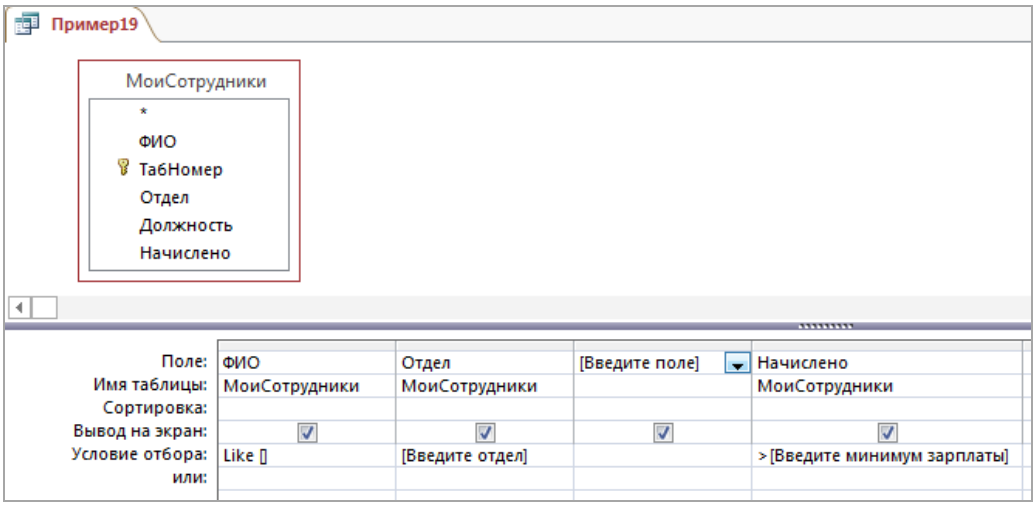


Рис. 81. Запрос с параметрами

Показанный на рис. 81 запрос содержит четыре параметра: три в условиях отбора и один – в названии столбца:

в поле *ФИО* условие – Like [];

в поле *Отдел* условие – [Введите отдел];

в поле *Начислено* условие – >[Введите минимум зарплаты];

в пустом поле – [Введите поле].

Обратите внимание, что один из них (шаблон фамилии) только обозначен скобками, без текста. Это, в принципе, возможно, тогда в окне ввода значения

параметра никакого текста, кроме стандартного «Введите значение параметра» не приводится. Очевидно, это не лучший комментарий к параметру, т.к. пользователь может не знать или не помнить, о каком параметре идет речь, ведь бланка он не видит. Поэтому, чем точнее текст, приведенный в скобках, отражает, что именно можно ввести, тем быстрее и точнее будет ответ. В данном случае, например, условие отбора могло выглядеть так:

Like [Введи первую букву фамилии и символ * за ней]

Выполните этот запрос, внося значения параметров в поля ввода открывающихся окон, как, например на рис. 82.

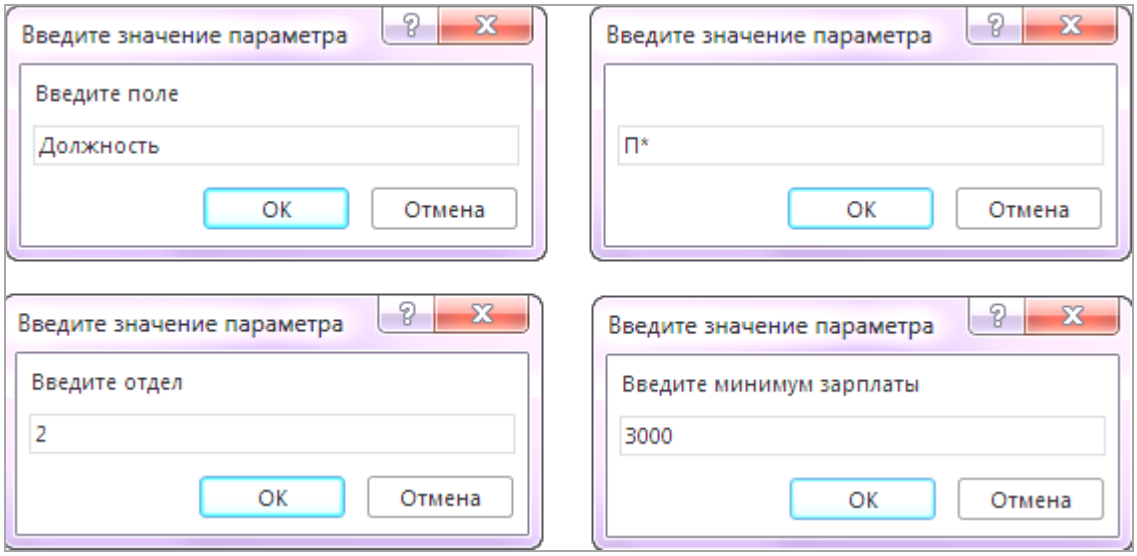


Рис. 82. Окна ввода значений параметров

☑ На основе запросов с параметрами можно создавать и формы с параметрами. В этом случае перед появлением формы на экране потребуется ввести значение параметра, которое и определит конкретное содержание формы.

Задание

45. Используя таблицу «Сотрудники» из БД «Борей», сформируйте запрос с параметрами, определив в качестве параметров окончание фамилии, город, страну, должность и год рождения сотрудников и какое-либо из оставшихся полей.

Лабораторная работа № 8. Запросы на изменение записей

8.1. Запрос на добавление данных в таблицу

Использование запроса для добавления записей в таблицу имеет смысл, если добавляется сразу несколько записей, взятых, например, из другой таблицы или из нескольких таблиц.

Таблица, в которую добавляются строки, является в этом случае *приемником*, а таблица (или несколько связанных таблиц), из которой берутся записи, - *источником*. В запросе должны быть сформированы записи с полями, которые соответствуют полям обновляемой таблицы. Структура записей в запросе необязательно должна совпадать со структурой записей в таблице-приемнике, полей может быть меньше и их типы могут иногда отличаться (но только в случае, когда эти типы могут быть преобразованы один к другому).

Добавление проводится по следующей схеме:

1) **Лента Access** → вкладка СОЗДАНИЕ → группа **Запросы** → кнопка **Конструктор запросов**, в бланк выбрать таблицы-*источники*.
2) **Лента Access** → вкладка КОНСТРУКТОР → группа **Тип запроса** → кнопка → **Добавление**.
3) В появившемся диалоговом окне заполнить имя таблицы-*приемника*.
4) Заполнить бланк по образцу:

Поле:	Поле1 источника	Поле2 источника	…	
Имя таблицы:	Имя источника	Имя источника	…	
Сортировка:				
Добавление:	Поле1 приемника	Поле2 приемника	…	
Условие отбора:	Условие на поле1 источника	Условие на поле2 источника	…	

Условие отбора служит для выбора нужных записей из таблицы-источника для добавления их в таблицу-приемник.

☑ Если таблица-приемник имеет ключевые поля, они должны быть включены в бланк обязательно!

Для упражнения создадим новую таблицу «Рекруты», которая будет служить приемником данных из источника «МоиСотрудники». Используем панель **Лента Access** → вкладка СОЗДАНИЕ → группа **Таблицы** → кнопка **Конструктор таблиц**, введем следующие имена и типы полей:

Фамилия – Короткий текст;

НомОтдела – Числовой;

КемРаботает – Короткий текст;

и сохраним таблицу под именем Рекруты. Ключевые поля задавать необязательно, т.к. это временная таблица для иллюстрации.

Пример 20. *Добавить в таблицу «Рекруты» из таблицы «МоиСотрудники» сведения об инженерах, фамилии которых начинаются на букву «П» или на букву «Л».*

Выполните шаги 1-3 рассмотренной схемы, взяв в бланк в качестве источника таблицу «МоиСотрудники», и внеся в окошке добавления имя таблицы-приемника «Рекруты».

В запросе будут присутствовать два условия отбора - на поле *ФИО* и поле *Должность*.

Выполните шаг 4, в результате получится бланк (рис. 83).

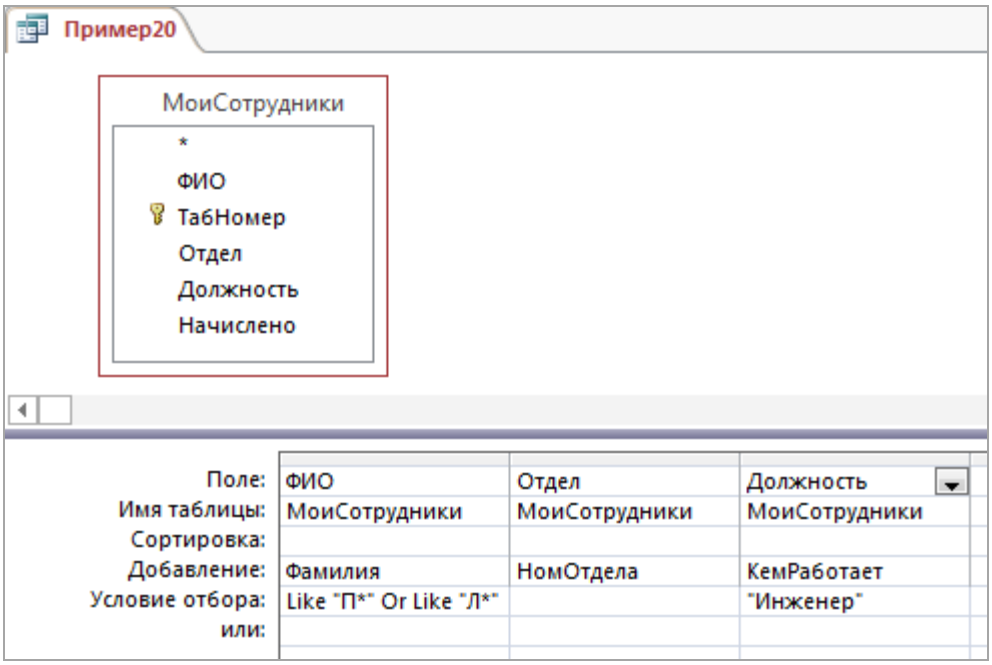


Рис. 83. Бланк запроса на добавление

Выполните запрос. В результате таблица «Рекруты» будет выглядеть как на рис. 84.

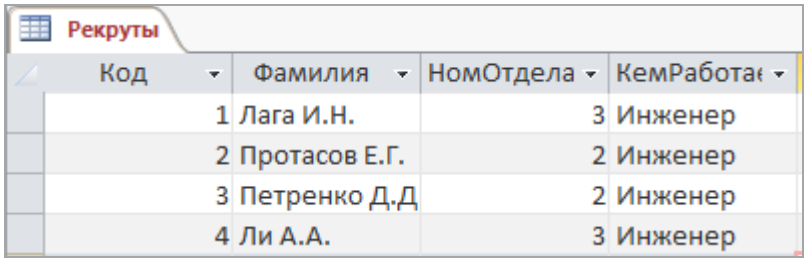


Рис. 84. Таблица, заполненная с помощью запроса на добавление

Задание

46. Добавьте в таблицу «Рекруты» товароведов и бухгалтеров планового отдела. *Подсказка*: в бланк запроса выберите также и таблицу «Отделы», из которой возьмите поле «Название» только для формулировки условия отбора.

8.2. Запрос на изменение (обновление) данных в таблице

С помощью такого запроса можно изменить одновременно группу записей. Изменения вносятся в те записи, которые удовлетворяют заданным условиям отбора. Новые значения полей определяются прямо в бланке запроса.

Пример 21. *Заменить должность «Инженер» на «Менеджер» для сотрудников второго отдела.*

Шаги выполнения:

1. Откройте бланк нового запроса, выбрав таблицу «МоиСотрудники».

2. **Лента Access** → вкладка КОНСТРУКТОР → группа **Тип запроса** → кнопка **Обновление**. В бланк добавится новая строка Обновление: .

3. Выберите в бланк поля, требующие обновления и поля, по которым задаются условия отбора. В данном случае это *Должность* (обновляемое поле) и *Отдел* (поле с условием отбора).

4. В строку Условие отбора для поля *Отдел* внесите значение 2 (второй отдел), а для поля *Должность* внесите значение Инженер.

5. В строку Обновление для поля *Должность* внесите новое значение Менеджер. Бланк примет вид, как на рис. 85.

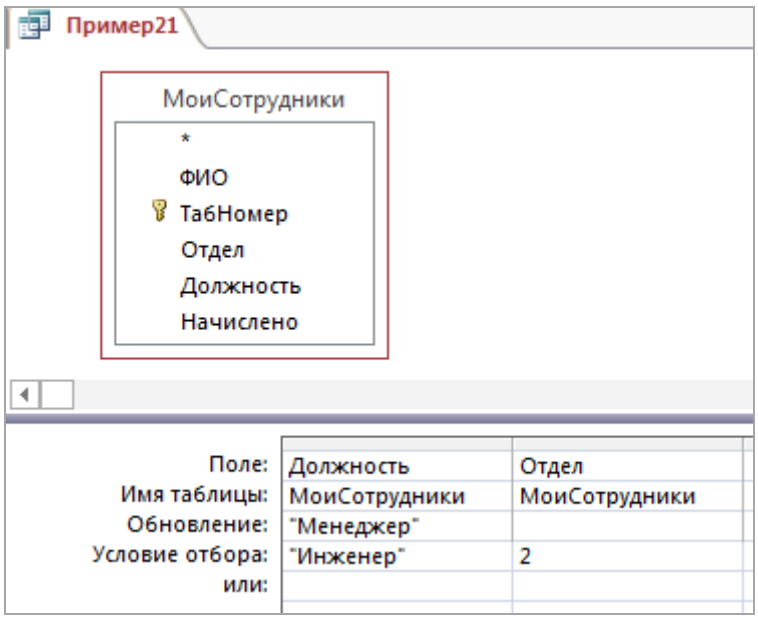


Рис. 85. Запрос на обновление

Если просмотреть полученный запрос в режиме таблицы до выполнения, то увидим записи (выбранные поля в них), которые *будут* обновлены.

6. Выполните запрос.

☑ Если в схеме БД при связывании таблиц была установлена стратегия целостности «CASCADE», то обновление может быть заблокировано, если в дочерней таблице обновляются ключевые поля.

☑ В качестве нового значения, указываемого в строке Обновление, можно задавать выражения, составленные из названий полей других таблиц (таблицы тогда нужно также включить в бланк).

Задание

47. Увеличьте зарплаты всех менеджеров на 1000 р.

8.3. Запрос на удаление записей из таблицы

Запросы этого типа позволяют удалить записи, выбираемые по некоторому критерию – условию отбора, из одной или нескольких связанных таблиц. В запросе указываются таблицы, из которых производится удаление, и условие отбора удаляемых записей.

☑ Если СУБД поддерживает стратегию поддержки целостности CASCADE, то в бланк можно включать только родительские таблицы – удаление из дочерних произойдет автоматически.

Для формирования запроса на удаление нужно:

1. Создать запрос в режиме Конструктора, в бланк выбрать таблицы, из которых удаляются записи.

2. **Лента Access** → вкладка КОНСТРУКТОР → группа **Тип запроса** → кнопка **Удаление.**

3. Заполнить бланк по схеме:

а) перетащить (можно двойным щелчком мыши) в бланк в строку Поле «*» для всех связанных таблиц, из которых идет удаление; в полученных столбцах в строке Удаление появится значение Из;

б) перетащить в бланк поля, участвующие в условиях отбора; в строке Удаление в этих столбцах появится значение Условие; в строку Условие отбора ввести требуемое условие.

Пример 22. *Удалить из таблицы «Рекруты» сотрудников третьего отдела*.

Бланк запроса будет выглядеть как на рис. 86.

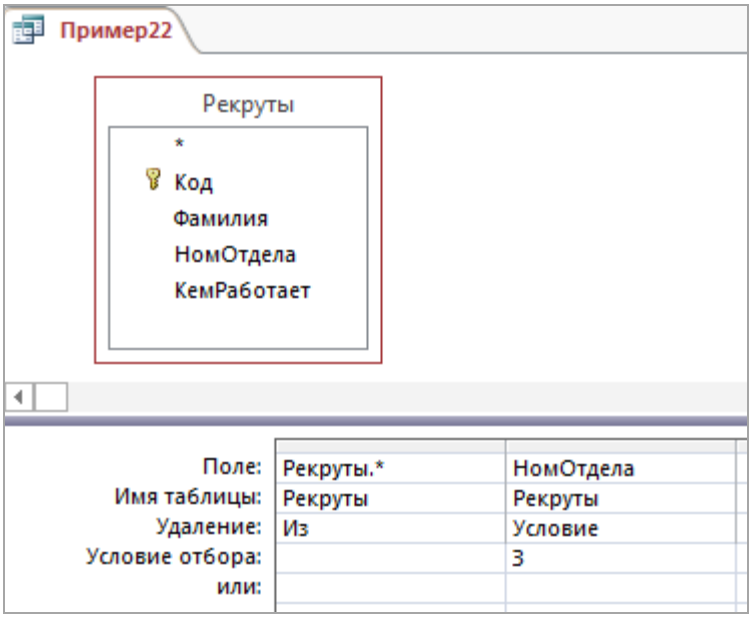


Рис. 86. Бланк запроса на удаление записей

Выполните этот запрос.

Задание

48. Создайте копии таблиц «Отделы» и «МоиСотрудники», изменив их названия. Свяжите новые таблицы. Удалите из полученных таблиц все сведения о сотрудниках и начальнике отдела, заведующий которого Л.Д. Осина

8.4. Запрос на создание таблицы

Результатом выполнения любого запроса является таблица, однако она не хранится в базе данных. Запрос на создание таблицы позволяет «увековечить» полученные временные объекты.

Запрос создается по следующей схеме:

1. Сформировать нужный запрос обычным образом. С полей, участвующих в условиях отбора, сбросить флажок вывода на экран.

2. **Лента Access** → вкладка КОНСТРУКТОР → группа **Тип запроса** → кнопка **Создание таблицы.**

3. Ввести новое имя для создаваемой таблицы в диалоговом окне.

4. Выполнить запрос, подтвердив создание (кнопкой **ОК**).

На панели таблиц появится новая таблица, соответствующая результату выполненного запроса.

Пример 23. *Создать таблицу «Численность» по запросу, определив для каждого заведующего отделом количество работающих в отделе сотрудников.*

Этапы выполнения запроса приведены на рис. 87, 88.

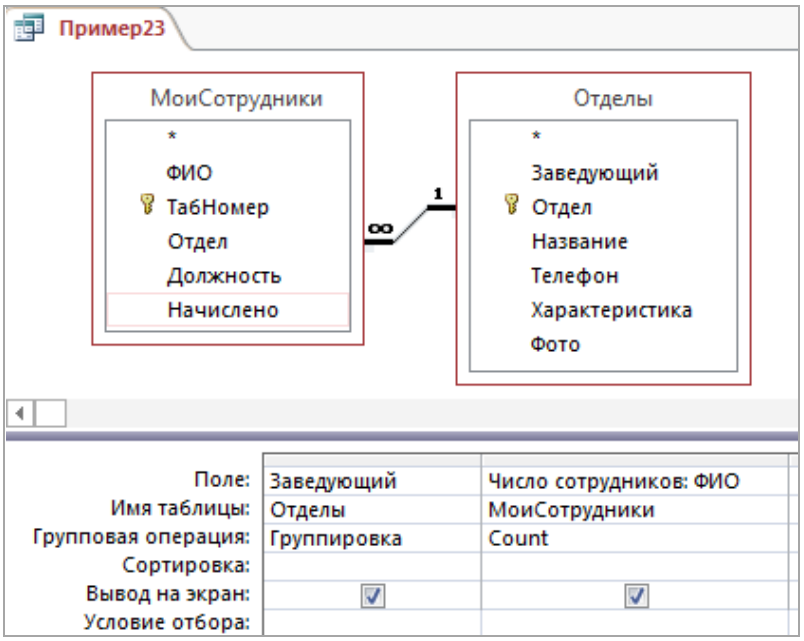


Рис. 87. Бланк запроса на создание таблицы

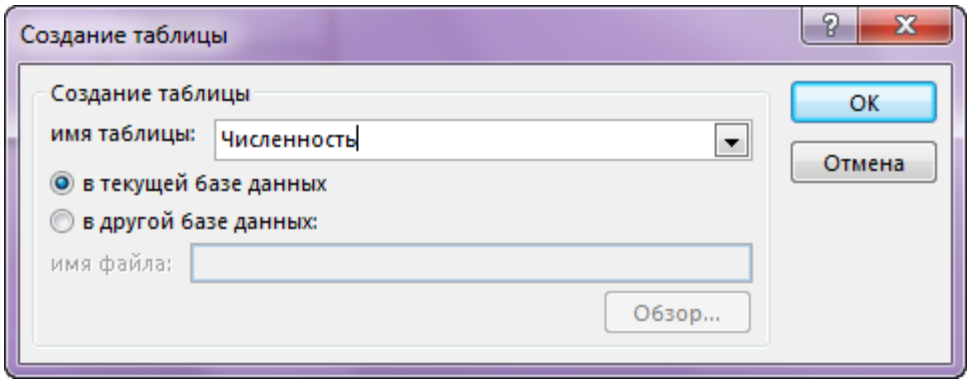


Рис. 88. Окно добавления названия создаваемой таблицы

Задание

49. Создайте таблицу почетных работников, включив в нее сотрудников, не являющихся начальниками отделов (и директором), зарплата которых превышает, например, 4000.

Лабораторная работа № 9. Создание форм

Форма – это объект, предназначенный для ввода, корректировки и отображения данных в удобном для пользователя виде; формы также могут служить для управления работой приложения, навигации по приложению. Формы можно использовать для более наглядного представления данных таблиц или наборов записей – результатов запросов; при необходимости форму можно вывести на печать. С помощью формы также можно в ответ на некоторое событие (например, изменение значения поля или нажатие кнопки) запустить *макрос* или *процедуру* Microsoft Visual Basic for Application. Пользователь приложения, созданного в Access, обычно имеет дело только с формами, стандартное окно базы данных должно быть доступно только разработчику.

9.1. Создание формы

Формы чаще всего создаются по таблицам или запросам базы данных. Форма может быть создана с помощью Конструктора или Мастера форм. Создадим форму по таблице «МоиСотрудники», используя Мастер форм.

1. В окне базы данных «Учебная» перейдите на вкладку СОЗДАНИЕ ленты ACCESS, в группе **Формы** нажмите кнопку **Мастер форм.** В появившемся окне «Создание форм» в выпадающем списке «Таблицы и запросы» выберите таблицу «МоиСотрудники» – *источник данных* для создаваемой формы, перенесите (нажимая кнопки со стрелкой) в форму нужные поля (рис. 89). Нажмите **Далее**.

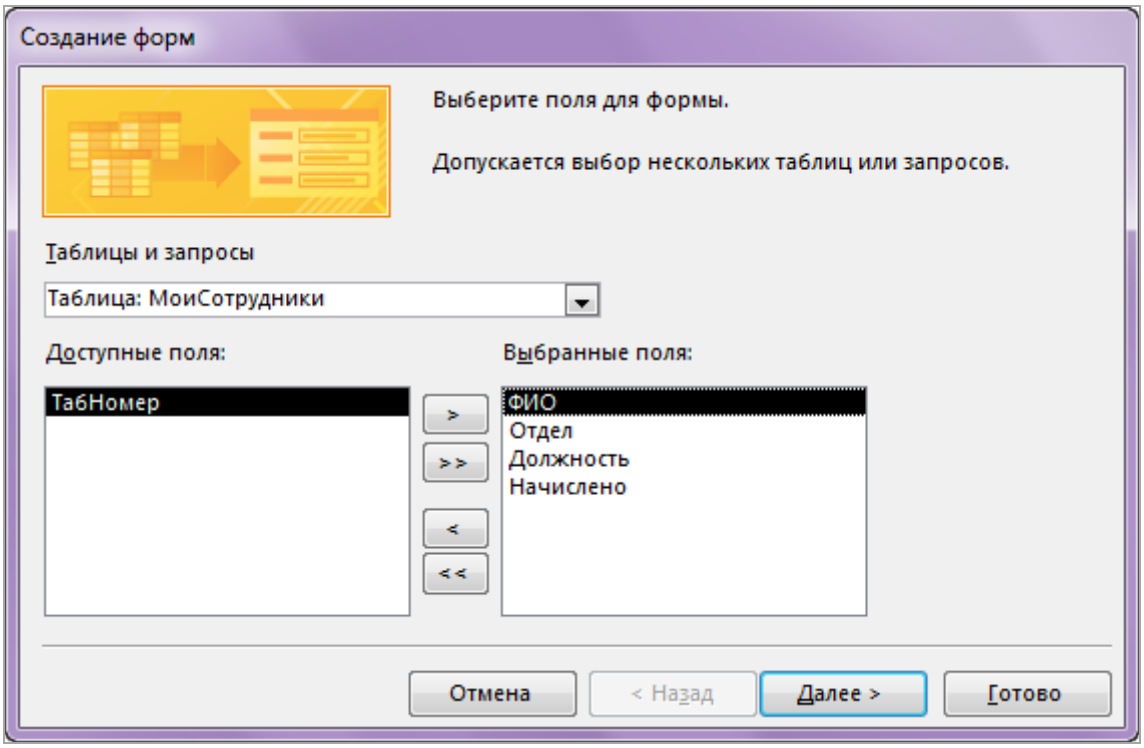


Рис. 89. Выбор полей для формы.

2. В качестве внешнего вида формы выберите **В один столбец**. Название формы оставьте МоиСотрудники. Нажмите **Готово**.

Для создания формы с помощью ***Конструктора***: **Лента Access** → вкладка СОЗДАНИЕ → группа **Формы** → кнопка **Конструктор форм**. Если на вкладке КОНСТРУКТОР в группе **Сервис** нажать кнопку **Добавить поля**, то справа появится панель **Список полей** (рис. 90).

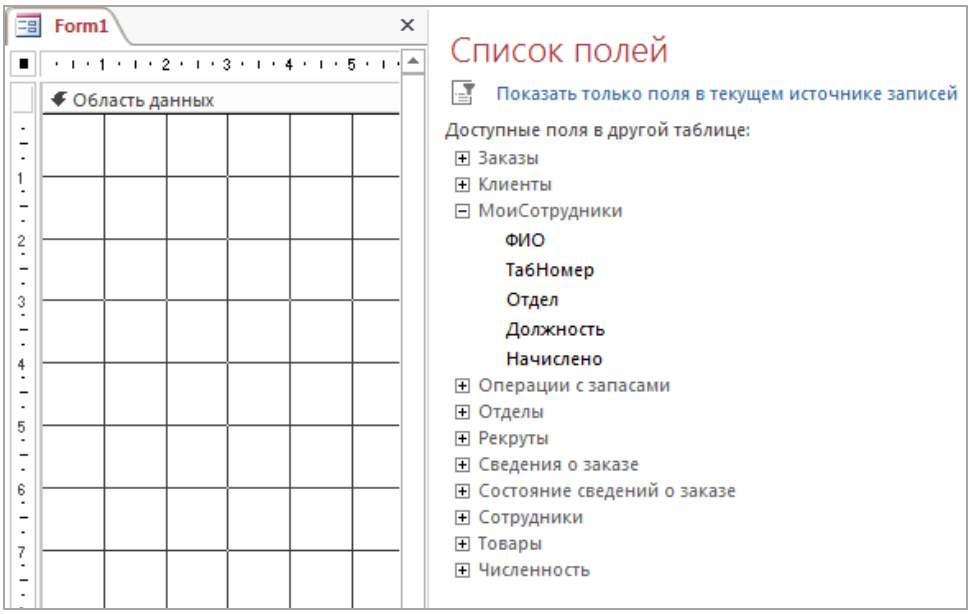
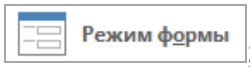


Рис. 90. Создание формы с помощью Конструктора форм.

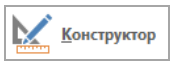
Теперь нужно просто перетащить мышью нужные поля из этого списка на форму. Чтобы выйти из конструктора в режим формы, нужно:

- на **ленте Access** → вкладка ГЛАВНАЯ (или вкладка КОНСТРУКТОР) → группа **Режимы** → кнопка **Режим формы** 

;
- контекстное меню вкладки формы (щелчок правой кнопкой мыши по заголовку вкладки) – пункт **Режим формы**;
- строка состояния – кнопка **Режим формы**.

9.2. Форматирование элементов формы

Перейдите в режим конструктора форм:

- на **ленте Access** → вкладка ГЛАВНАЯ → группа **Режимы** → кнопка **Конструктор** 

;

- контекстное меню вкладки формы (щелчок правой кнопкой мыши по заголовку вкладки) – пункт **Конструктор**;
- строка состояния – кнопка **Конструктор**.

Прежде, чем изменять параметры, положение элемента формы или свойства самой формы, эти элементы (или форму) нужно ***выделить***.

Любой ***элемент*** макета формы можно выделить, щелкнув на нем мышью; при этом около него появятся квадратные маркеры, позволяющие изменять размеры элемента. При перемещении курсора мыши на рамку вокруг выделенного элемента вид курсора меняется на изображение руки. Если в этот момент нажать левую кнопку мыши, то элемент можно перетащить на новое место. Если этот элемент – поле, то он перемещается вместе с надписью.

Для перетаскивания *только* поля или *только* надписи необходимо перетаскивать мышью за большой маркер, расположенный в левом верхнем углу поля или надписи соответственно (при этом курсор изменит вид на «руку-указатель»). Эти маркеры обычно используются для изменения взаимного положения и расстояния между полем и его надписью.

Для того чтобы ***выделить всю форму***, нужно щелкнуть мышью на левом верхнем квадрате под строкой заголовка формы, появится черный маркер: ⊡ .

Выделенные текстовые поля и надписи к ним можно ***форматировать,*** как обычно в Word, задавая тип, цвет и размер шрифта, цвет и фактуру заливки, выравнивание, используя стандартную панель форматирования. Можно применять также настройки окна свойств. Это окно открывается из контекстного меню *выделенного* элемента (правой кнопкой мыши щелкнуть на этом элементе, выбрать **Свойства** и вкладку **Макет**) или кнопкой «Страница свойств» 

(**Лента Access** → вкладка КОНСТРУКТОР → группа **Сервис** – кнопка **Страница свойств**) или нажатием клавиши <F4> на клавиатуре. Окно свойств в этом окне позволяет, например, задать или убрать полосы прокруток, заблокировать свободный доступ к полю, задать рисунки для фона, правильно вписать готовые элементы-рисунки в отведенную рамку и т.п.

☑ Перечень возможностей, перечисляемых в ***окне свойств*** для каждого элемента – свой, контекстный. Поэтому всегда обращайте внимание на заголовок этого окна, в котором указан объект, для которого и отображены свойства.

Пример 24. *Заменить фоновый рисунок формы*.

Шаги выполнения:

1. Выделите форму, щелкнув мышью маркер выделения формы (рис. 91).

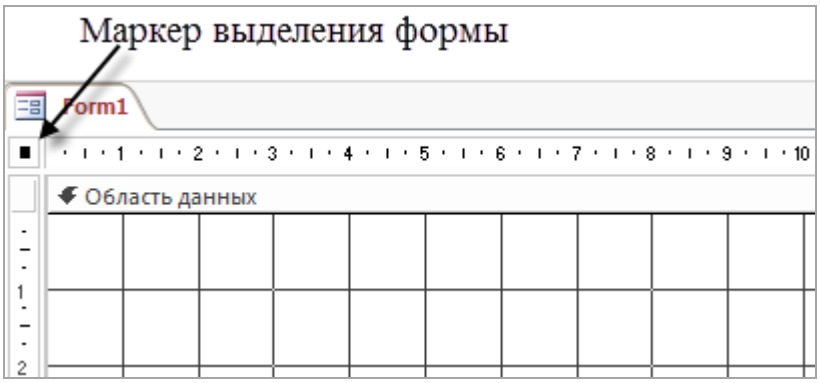


Рис. 91. Выделение формы

2. Откройте окно свойств формы, перейдите в нем на вкладку **Макет**, в строке **Рисунок**, укажите изображение, нажав кнопку обзора ... в этой строке (рис. 92).

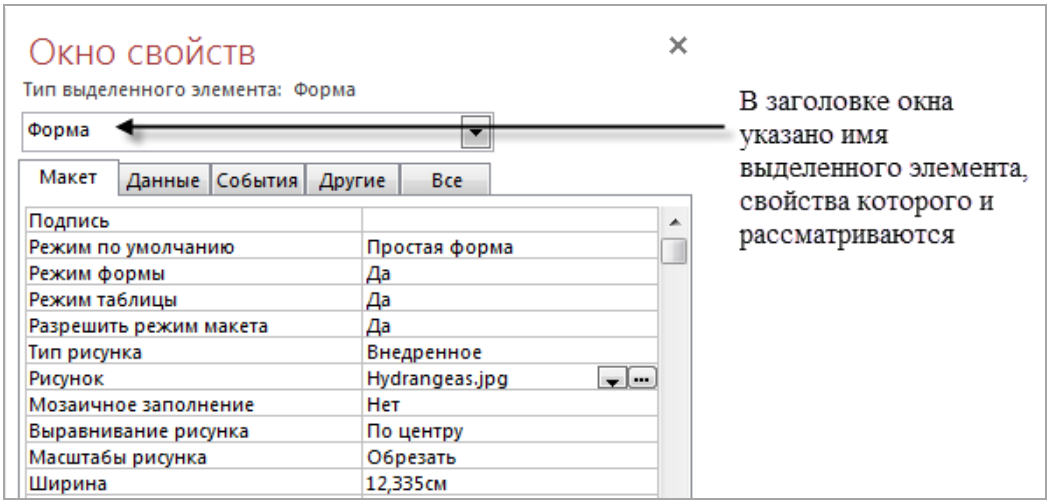


Рис. 92. Настройка вида формы в окне ее свойств

3. Настройте **Масштабы рисунка**, вписав его в рамку (рис. 93).

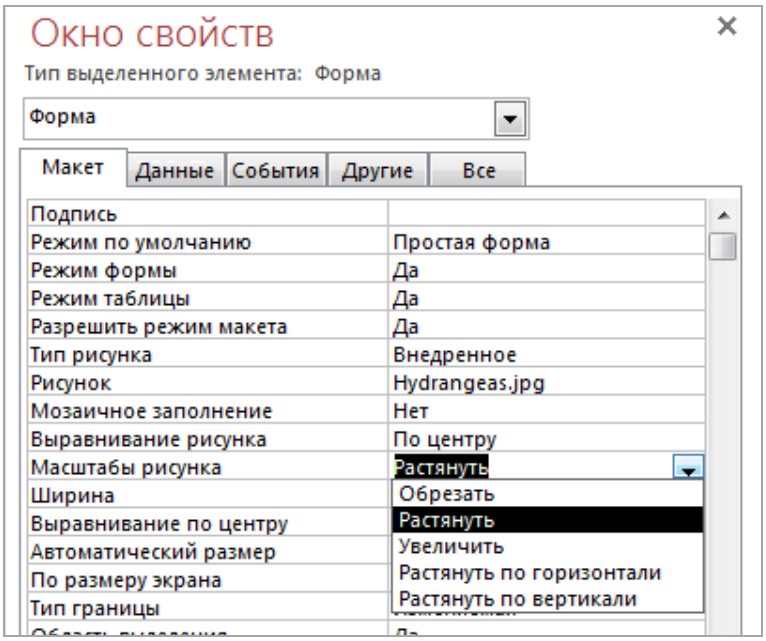


Рис. 93. Настройка фонового рисунка

Аналогичным образом настраиваются все изображения, являющиеся элементами формы (фото, собственные рисунки, клипы): элемент выделяется, открывается окно свойств и настраивается масштаб изображения.

Отключение доступа к нередактируемому полю: Поле, доступ к которому запрещается, выделяется. В окне *его* свойств на вкладке **Данные**, в строке **Включена** выбирается вариант **Нет**. Тогда в стандартном режиме формы на данное поле нельзя перейти, и это поле изображается как недоступное (серым цветом).

‼☝ Обратите внимание, что если сделать недоступным, например, поле *Табномер* (первичный ключ), то ни одну запись нельзя будет добавить с помощью формы: ввод этого номера запрещен, его значение тогда для новой записи рассматривается как неопределенное, что не допускается для первичного ключа из-за контроля целостности БД. Поэтому в случае блокировки первичного ключа на форме можно только просматривать и редактировать записи без добавления.

Укажите источник данных для Формы: выделите форму, перейдите в **Окно свойств** и на вкладке **Данные** в строке **Источник записей** выберите таблицу «МоиСотрудники»

Сохраните созданную форму, перейдите в стандартный режим формы.

С помощью панели навигации по записям в нижней части формы (рис. 94), нажав кнопку , введите в базу данных еще 3 записи (с двадцать второй по двадцать четвертую), после чего закройте форму ,(рис. 94).

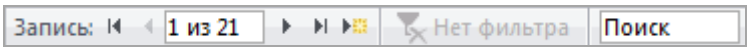


Рис. 94. Панель навигации по записям формы

Задания

50. Создайте 2 формы по таблице «Отделы» двумя способами: с помощью Мастера и Конструктора. Отредактируйте первую из них, увеличив шрифты для надписей и полей, изменив их цвет, залить цветом поля, отформатируйте поле *Фото* по размеру рамки. Компактно разместите поля на форме.

51. Создайте форму по запросу: выдать названия отделов и среднюю зарплату в каждом отделе.

Лабораторная работа № 10. Вычисляемые поля на форме и подчиненные формы

10.1. Добавление в форму вычисляемых полей

В форму можно добавлять поля, не принадлежащие никакой таблице или запросу; они отображают значения, получаемые в результате вычисления некоторых выражений. Такие выражения могут содержать в качестве операндов встроенные функции, имена полей таблиц (доступных из данной формы), значения различных типов, а в качестве операций – арифметические, логические операции, сравнения. Подробный перечень средств построения выражений (в том числе встроенных функций) можно получить, открыв окно построителя выражений (кнопка **Построить** панели инструментов Access в режиме Конструктора формы).

Некоторые полезные функции:

▪ IIf(expr;truepart;falsepart) – условная операция «если» (аналогичная операции **ЕСЛИ** в Excel) с тремя аргументами:

1) условие - логическое выражение (expr). Формируется чаще всего на основе операций сравнения и может содержать связки OR (или), And (и), Imp (следует), Not (не), Eqv (эквивалентно).
2) операция, выполняемая, если условие истинно (truepart);
3) операция, выполняемая, если условие ложно (falsepart).

▪ Now() – текущие дата и время; обновить эти данные можно, выбрав в меню **Записи** (в режиме Формы, не в конструкторе) команду **Обновить.**

▪ Date() – текущая дата; формат выводимой на форму даты можно настроить, выделив поле с этим выражением (в режиме Конструктора), затем выбрав **Свойства** в контекстном меню, вкладку **Макет**, свойство **Формат поля** и нужный формат из предлагаемого списка.

▪ Avg(expr), Sum(expr), Max(expr), Min(expr) – функции, вычисляющие среднее, суммарное, максимальное и минимальное значения выражения expr, составляемого по правилам для выражений в Access. В частности, это может быть название поля открытой формы, например, Sum([Начислено]).

Для использования вычисляемых полей на форму в режиме Конструктора добавляется новое поле, в которое и заносится выражение (формула), **начинающееся всегда со знака равенства =**, например, =Now() или =Min([Начислено]). В режиме Формы в этом поле отображается значение выражения, вычисленное на основе текущих данных БД. Надпись к полю формируется как отдельный независимый элемент.

☑ Выражение можно задать Построителем выражений. Для этого нужно выделить созданное (свободное) поле, открыть **Свойства** → вкладка **Данные** → строка **Данные**, щелкнуть в этой строке по полю ..., в открывшемся окне

Построитель выражений выбрать нужные объекты и операции. Можно также в строке **Данные** набрать выражение вручную, начав с символа =.

Пример 25. *Добавить на форму «МоиСотрудники» вычисляемое поле «Премия», рассчитываемое следующим образом: для инженеров она составляет 30% от зарплаты, для остальных – 20%.*

Шаги выполнения:

1. Откройте форму «МоиСотрудники» в режиме Конструктора. Добавьте на форму элемент **Поле**: щелкните мышью по этому элементу - кнопка ab| (**лента Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** → кнопка **Поле**), затем выделите мышью местоположение этого поля на форме. Поле появляется вместе со стандартной надписью.

2. Отформатируйте надпись: замените стандартную надпись «поле *N*» на «Премия». Задайте для нее нужные размер и вид шрифта.

3. В само поле вместо слова «Свободный» введите выражение (без пробелов!):

=IIf([Должность]=’Инженер’; [Начислено]*0,3;[Начислено]*0,2)

Здесь [Должность]=’Инженер’ – условие (логическое выражение); [Начислено]*0,3 – операция, выполняемая, если условие окажется истинным; [Начислено]*0,2 – операция, выполняемая, если условие окажется ложным.

4. Перейдите в режим Формы и просмотрите записи.

‼☝ Обратите внимание, что значения текстовых полей в выражении (здесь – в условии) берутся в *одинарные* кавычки. Не копируйте их из Word, в Access кавычки другие. ☹

Если условие усложнить, например, премия инженеров – 30%, бухгалтеров – 20, а остальных – 10, то можно использовать «вложенный» IIf, в котором falsepart (операция, если условие ложно) – вновь будет IIf:

= IIf([Должность]=’Инженер’; [Начислено]*0,3;IIf([Должность]=’Бухгалтер’; [Начислено]*0,2; [Начислено]*0,1))

Читается эта операция так: *Вычисляемое значение равно* *Если* [Должность]=’Инженер’, *то* [Начислено]*0,3, *иначе: если* [Должность]=’Бухгалтер’, *то* [Начислено]*0,2, *иначе*: [Начислено]*0,1

Задания

52. Выведите на форму поля с текущими Датой и Датой/Временем; для даты задайте «Длинный формат даты» (используйте Свойства этого поля, Макет → Формат поля).

53. Выведите на форму поле «Штраф» в размере 5% от зарплаты для сотрудников 1-го и 2-го отделов за несоблюдение правил пожарной безопасности.

54. Выведите на форму поле «Налог»: для зарплат, не больших 3000, он составляет 10% от **средней** зарплаты; не больших 10000 – 20, больших – 40.

10.2. Создание подчиненной формы

Часто бывает удобно на одной форме просматривать и исправлять записи, выбирая их из разных таблиц или запросов. Особенно часто такая необходимость возникает при работе со связанными таблицами, т.к. данные в них согласованы.

В Access для этой цели используются так называемые *подчиненные формы*. Подчиненная форма строится непосредственно на *главной форме*; в этой паре главная форма соответствует родительской таблице (идет со стороны «один» в связи «один-ко-многим»), подчиненная – дочерней (идет со стороны «многие» в связи «один-ко-многим»). В этом случае одной записи главной формы соответствует несколько записей подчиненной формы.

Для подчиненной формы в качестве источника данных может быть выбрана также таблица (или запрос), не являющаяся дочерней (например, при связи «один-к-одному»), вообще любая таблица, даже та же самая (для одновременного просмотра разных записей одной и той же таблицы). Тогда

одной записи основной формы будет соответствовать только одна запись подчиненной.

Создать форму с подчиненной можно несколькими способами. Рассмотрим некоторые из них.

1-й способ. Главная и подчиненная формы строятся одновременно, на основе имеющихся таблиц и/или запросов. Возьмем для этой цели связанные таблицы «Отделы» (родительская) и «МоиСотрудники» (дочерняя). Шаги:

1. В окне БД **лента Access** → вкладка СОЗДАНИЕ → группа **Формы** → кнопка **Мастер форм**. Перенесите поля из таблиц МоиСотрудники (ФИО, Должность, Начислено) и Отделы (Заведующий, Отдел, Телефон, Название). Нажмите Далее (рис. 95).

2. В следующем окне выберите главной формой «Отделы» и установите переключатель «Подчиненные формы» (рис. 96).

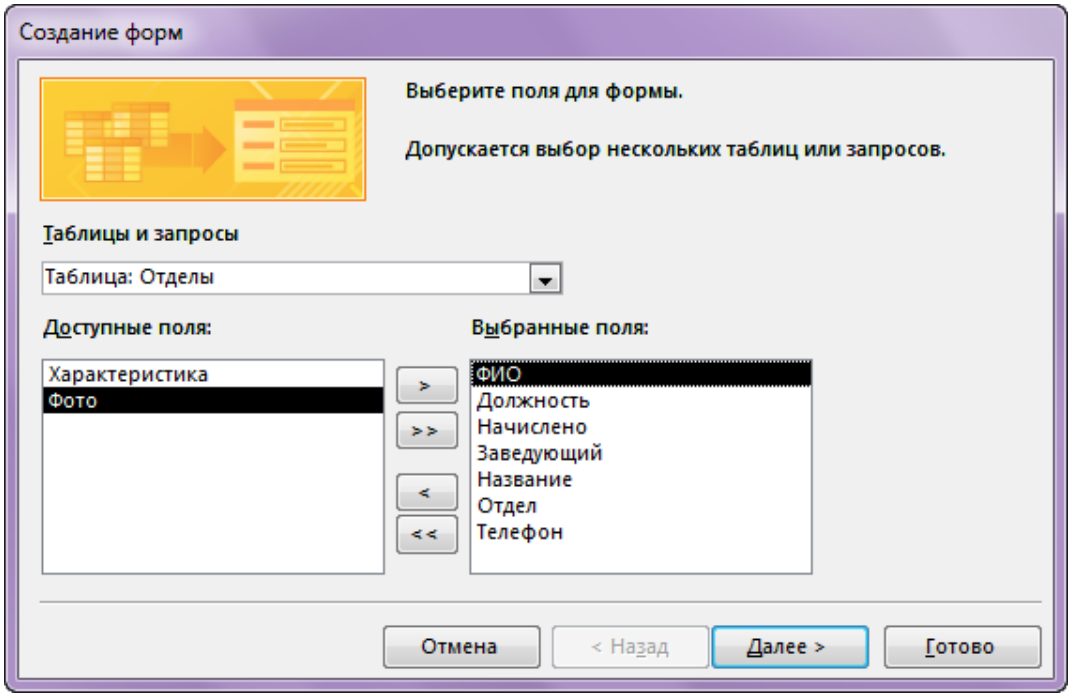


Рис. 95. Выбор источника данных для главной и починенной форм

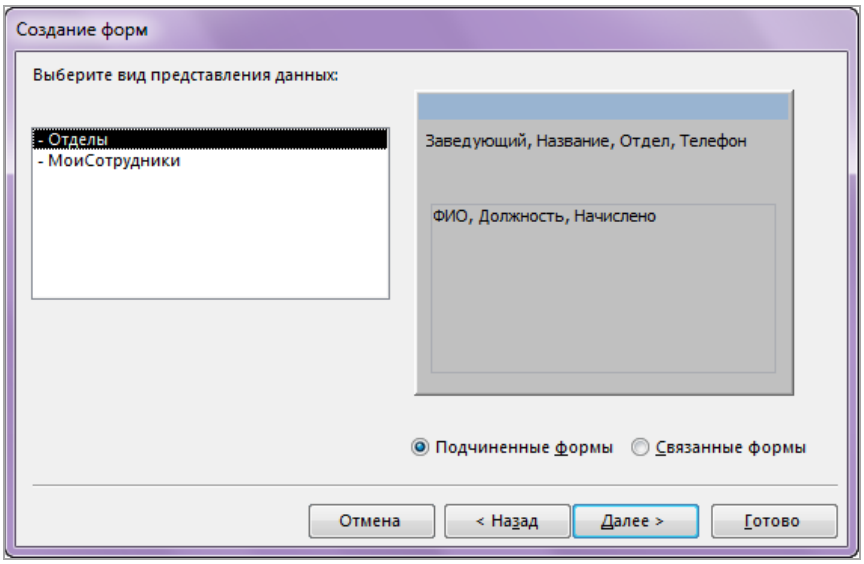


Рис. 96. Выбор главной формы

Результат (рис. 97) – форма с подчиненной формой; для каждой записи главной формы подчиненная показывает все соответствующие записи из связанной (дочерней) таблицы.

3. Перейдите в режим конструктора, отформатируйте вид подчиненной формы так, чтобы выбранные поля были видны без использования полосы прокрутки.

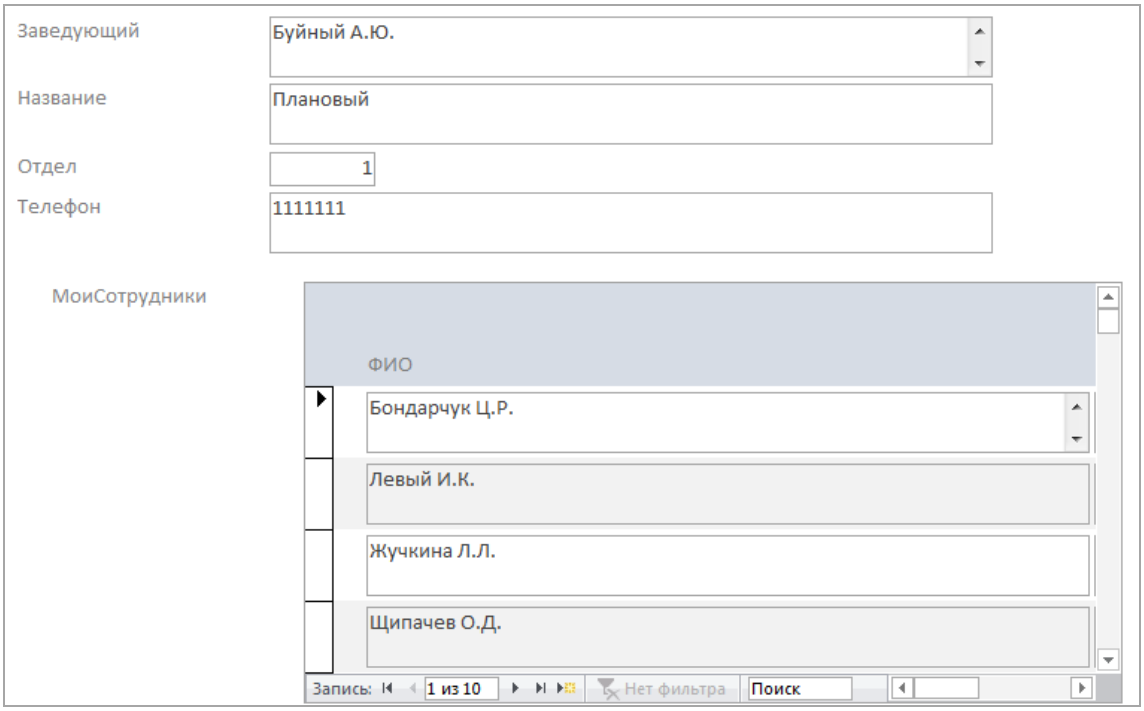


Рис. 97. Форма с подчиненной формой

2-й способ. Предполагается, что формы для главной и подчиненной таблиц созданы заранее. Будем использовать ранее созданные формы для таблиц «МоиСотрудники» и «Отделы» (или их копии). Шаги:

1. Откройте форму для родительской (главной) таблицы «Отделы» в режиме конструктора. Расширьте область данных так, чтобы можно было разместить подчиненную форму.

2. **Лента Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** → кнопка **Подчиненная форма/Отчет**  . Выделите, удерживая левую кнопку мыши, область, куда будет помещена подчиненная форма.

3. В открывшемся окне Мастер подчиненных форм из списка «Имеющиеся формы» выберите форму «МоиСотрудники». Нажмите «Готово».

4. Отформатируйте полученные объекты, замените надпись «Внедренный» на «Сотрудники» (рис. 98).

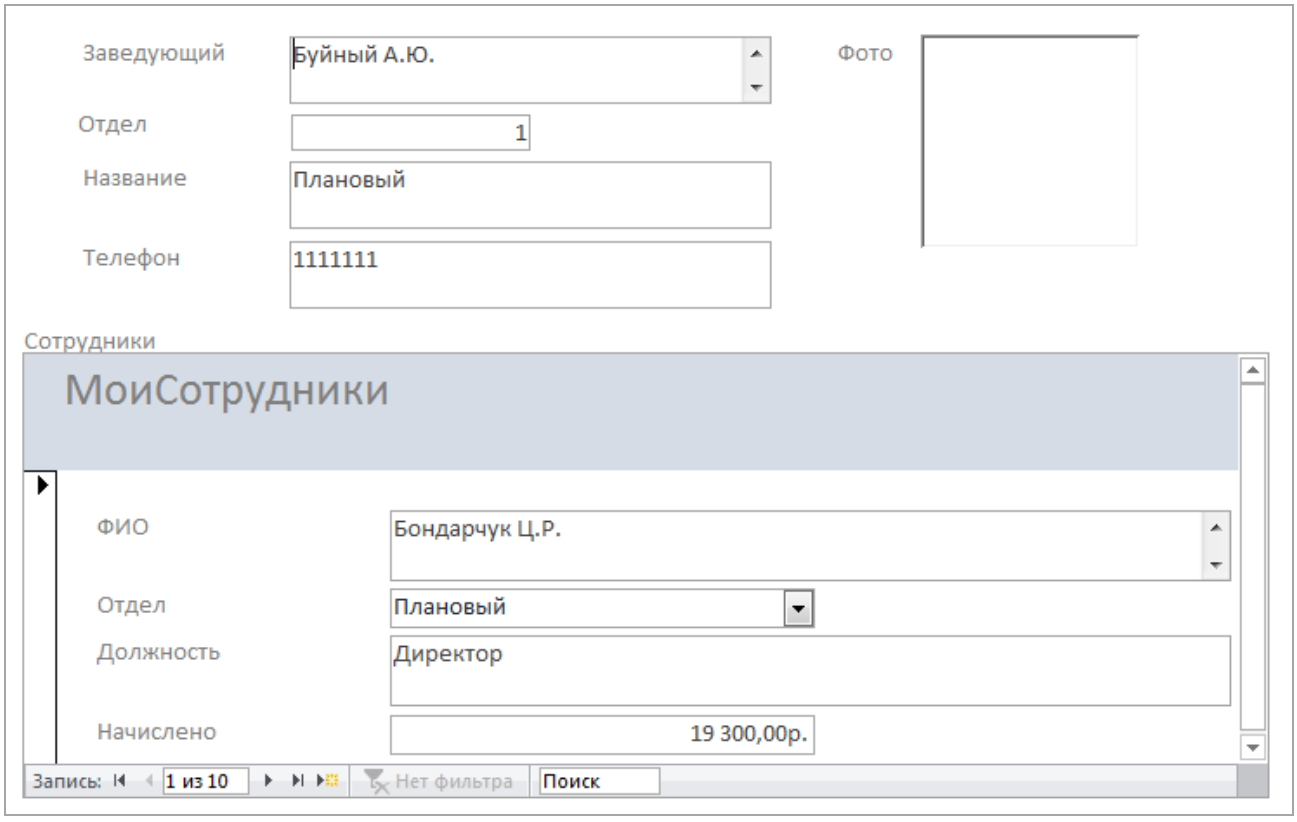


Рис. 98. Форма с подчиненной. 2-й способ

3-й способ. Самый простой. Предполагается наличие в БД двух готовых форм – будущих главной и подчиненной. Шаги:

1) Открыть главную форму в режиме конструктора. Расширить (мышью) ее область данных.

2) Перетащить мышью подчиненную форму из Области навигации БД на главную. Попробуйте этот способ с копией формы «Отделы».

Лабораторная работа № 11. Добавление элементов на форму

На форму можно добавлять, кроме полей таблицы или запроса, по которым форма создавалась, другие элементы для улучшения вида формы, удобства пользователей или задания реакции на действия пользователя. Отметим некоторые из них:

 – **Рисунок** – позволяет поместить в форму статический рисунок. Его нельзя будет редактировать, но можно форматировать, например, вписать в рамку.

 – **Свободная рамка объекта** – используется для включения любого OLE-объекта. Объект становится частью формы, но не хранится в таблице БД. Можно таким образом включить звук, видео, слайды. Можно объект форматировать (Например, **Свойства → Макет → Установка размеров → По размеру рамки**). Независимо от того, какая запись отображается на форме, объект будет отображен один и тот же. Он может быть изменен прямо из формы с помощью того приложения, в котором он был создан.

– **Присоединенная рамка объекта** – служит для отображения ОЛЕ-объектов, которые либо сами хранятся в поле таблицы БД, либо в таблице хранятся ссылки на них, т.е. источник данных этого элемента – поле базовой таблицы. Поэтому при переходе от записи к записи объект в присоединенной рамке объекта будет изменяться (например, фото).

 – **Разрыв страницы** – вставка разрыва в многостраничной форме.

 – **Список** – используется для создания элемента «Список», который включает несколько возможных значений. Он всегда раскрыт. Значения можно задать явно (свободный список) или используя поле таблицы. Если этот список присоединен к полю базовой таблицы или запроса, то значение этого поля можно изменить, щелкнув по нужной строке списка. Способ построения списка аналогичен построению поля со списком.

 ,  – **Прямоугольник, Линия** – служат для оформления внешнего вида, например, для группировки логически связанных элементов в один блок. Типы и цвет линий и границ, прозрачность, оформление (приподнятое, утопленное, с тенью) устанавливаются в свойствах этих элементов.

11.1. Создание поля со списком

Поле со списком обеспечивает возможность выбора готового значения из предоставляемого пользователю *списка* или ввода другого значения в *поле* этого элемента. Источником выбираемых значений может служить либо поле таблицы или запроса, либо набор заданных при построении списка значений.

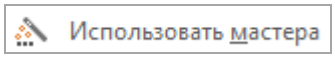
☑ Для полей с подстановкой элемент **Поле со списком** создается автоматически при создании формы. Такие поля часто организуются для внешних ключей, например, в БД «Учебная» – это поле *Отдел* в таблице «МоиСотрудники» (проверьте для формы «МоиСотрудники»).

В зависимости от того, что является источником значений поля со списком и каковы его функции, могут существенно измениться некоторые свойства этого объекта.

Рассмотрим несколько назначений этого элемента.

Откройте таблицу «Отделы» в режиме конструктора и добавьте поле *Статус* типа **Короткий текст**. Закройте таблицу, сохранив изменения.

Пример 26. *Создать на форме «Отделы» поле со списком фиксированных значений для заполнения поля Статус таблицы Отделы*.

Откройте форму, созданную по таблице «Отделы» в режиме Конструктора. На **ленте Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** → раскройте выпадающий список и нажмите кнопку **Использовать мастера** 

(рис. 99**)**. Нарисуйте мышью на форме область для поля со списком  . Далее следуйте инструкциям мастера:

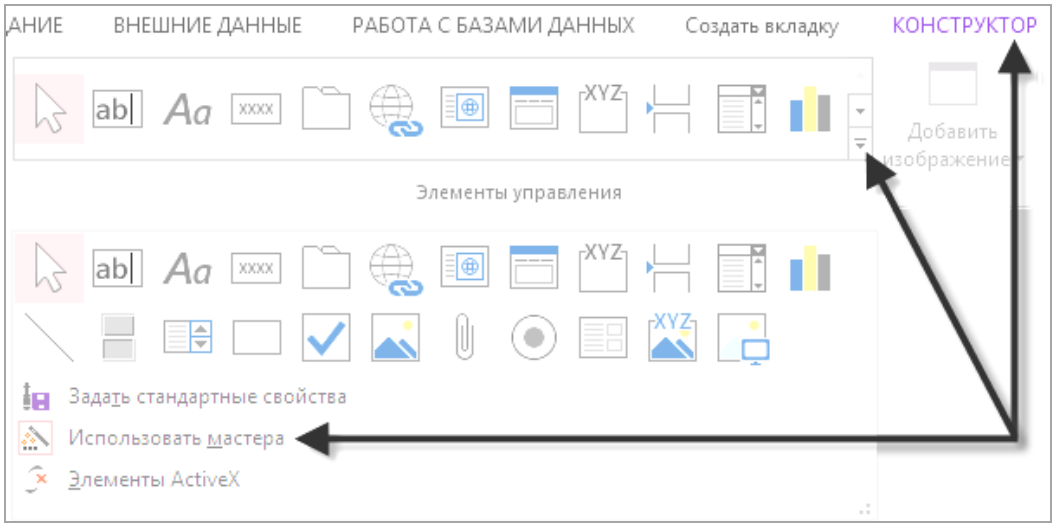


Рис. 99. Использование Мастера для элементов управления формы

1. В первом окне **Создание полей со списком** отметим переключатель **`Будет введен фиксированный набор значений`. Далее.**

2. Во втором окне оставим **`число столбцов 1`**, а сам **`столбец 1`** заполним значениями (построчно): высокий, средний, низкий, высший, без права подписи (рис. 100). **Далее.**

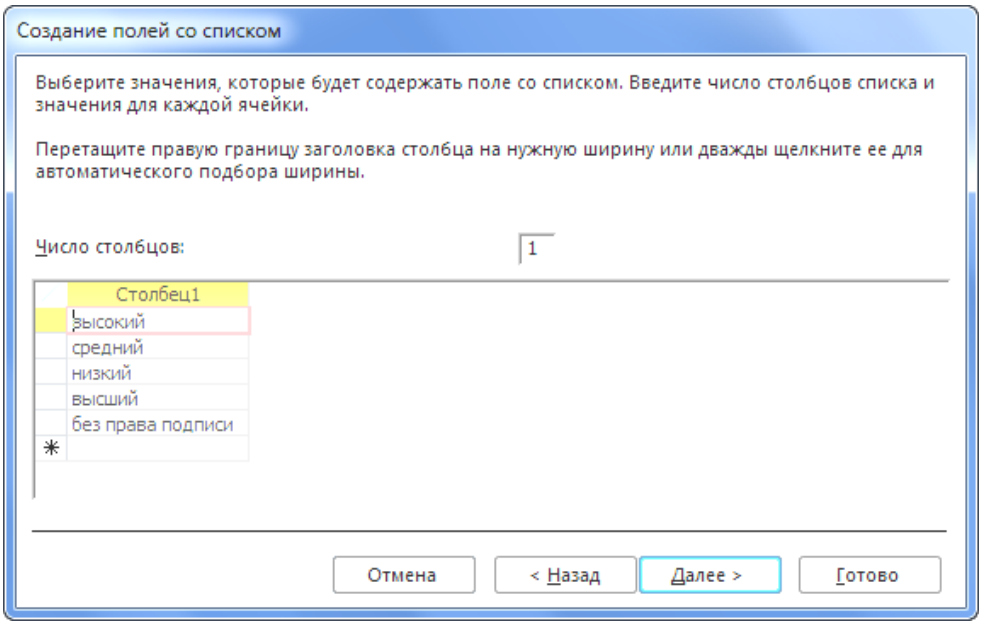


Рис. 100. Задание значений поля со списком

3. В следующем окне отмечаем переключатель **Сохранить в поле**, а в выпадающем списке рядом выбираем поле *Статус*. **Далее.**

4. Назначаем название (подпись) для этого элемента: *Статус*. **Готово.**

5. Перейдите в режим Формы и выберите из поля со списком значения статуса для каждой записи.

Просмотрите (в форме в режиме Конструктора) **Свойства** полученного поля со списком (рис. 101).

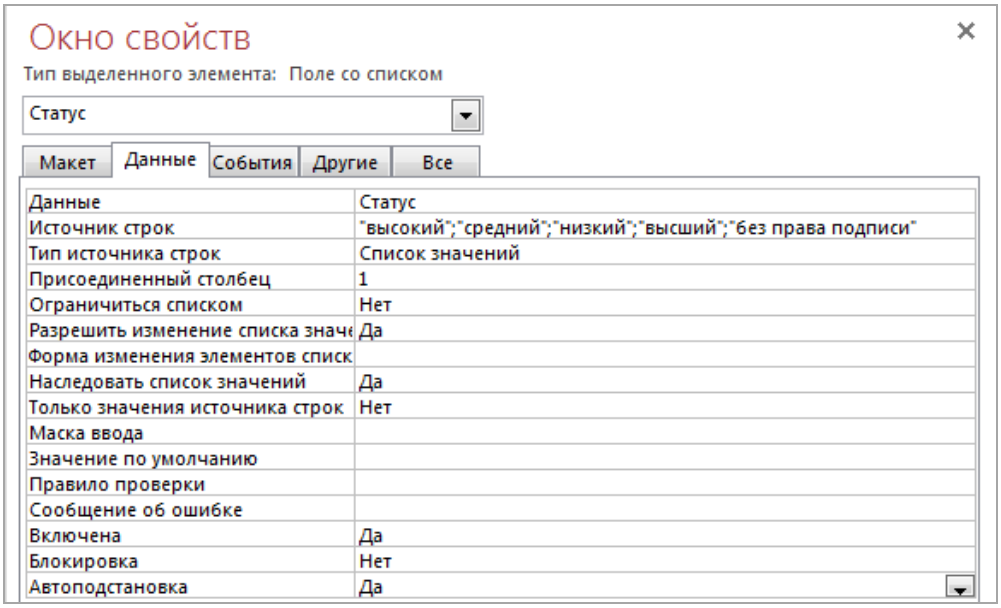


Рис. 101. Свойства поля со списком фиксированных значений для ввода в поле таблицы

☑ Обратите внимание на следующие свойства:

Данные – название поля таблицы, которое будет содержать выбранное в списке значение. Если список является свободным (на шаге 3 выбран переключатель **Запомнить значение**), то это свойство остается пустым.

Тип источника строк – объект-источник данных для списка; обычно это Список значений (как в данном случае) или **Таблица или запрос** (как в примере 27).

Источник строк – сам список значений. В данном случае он задан безотносительно к объектам БД; если же значения берутся *из заполненного поля* таблицы или запроса, то в качестве источника вставляется инструкция SQL (пример 27).

Присоединенный столбец – столбец, значения которого будут *на самом деле* проставлены в таблицу. Для поля с подстановкой в качестве значений вводятся соответствующие значения скрытого (ключевого) столбца. Если значения списка берутся из таблицы, то столбцов может быть два, первый из которых - ключевой (он скрыт). В нашем примере значения берутся не из таблицы, поэтому столбец один.

Ограничиться списком – указывает, можно ли вводить в *поле* объекта значение, *не* совпадающее ни с одним из элементов списка. Выбор **Да** означает, что нельзя. **Нет** – означает, что список может дополняться другими значениями, вводимыми в текстовую часть поля.

Автоподстановка – разрешает (если значение этого свойства – **Да**) вводить в поле элемента *часть* значения из списка (например, первые буквы или цифры), тогда совпадающее в этой части значение выберется автоматически.

Пример 27. *Создать на форме «Отделы» поле со списком, значения которого могут быть заданы свободно или выбираться из поля Телефон и сохраняться в этом же поле*.

1. В первом окне Мастера выберите переключатель **Объект «поле со списком» будет использовать значения из таблицы или запроса** таблицу «Отделы».

2. В следующих окнах Мастера выберите таблицу «Отделы» и поле *Телефон*.

3. В следующем окне обратите внимание, что **число столбцов** будет **2**: один (первый) – ключевой (он, как правило, скрывается) и другой (второй) – тот, значения которого вы планируете увидеть в списке.

4. Далее отметьте переключатель **Сохранить в поле** и выберите поле *Телефон*.

5. Задайте имя поля со списком и нажмите **Готово**.

6. Перейдите в режим Формы и попробуйте воспользоваться созданным инструментом; в случае «непредвиденной аварии» исправьте свойства:

!!☝ При использовании полученного поля со списком в таблицу будет проставлено не выбранное вами значение из списка, а соответствующее значение ключевого (скрытого) столбца. Это может спровоцировать выдачу сообщения об ошибке несоответствия типов данных. Для того чтобы в поле попадало выбранное вами значение, войдите в **Свойства** поля со списком, вкладка **Данные**, строка **Присоединенный столбец**, поставьте значение 2. Если при этом запретить ввод новых значений (свойство **Ограничиться списком** задать **Да**), то список значений, корректируемый вашим выбором, может существенно сузиться (поэтому выберите значение – **Нет**) (рис. 102).

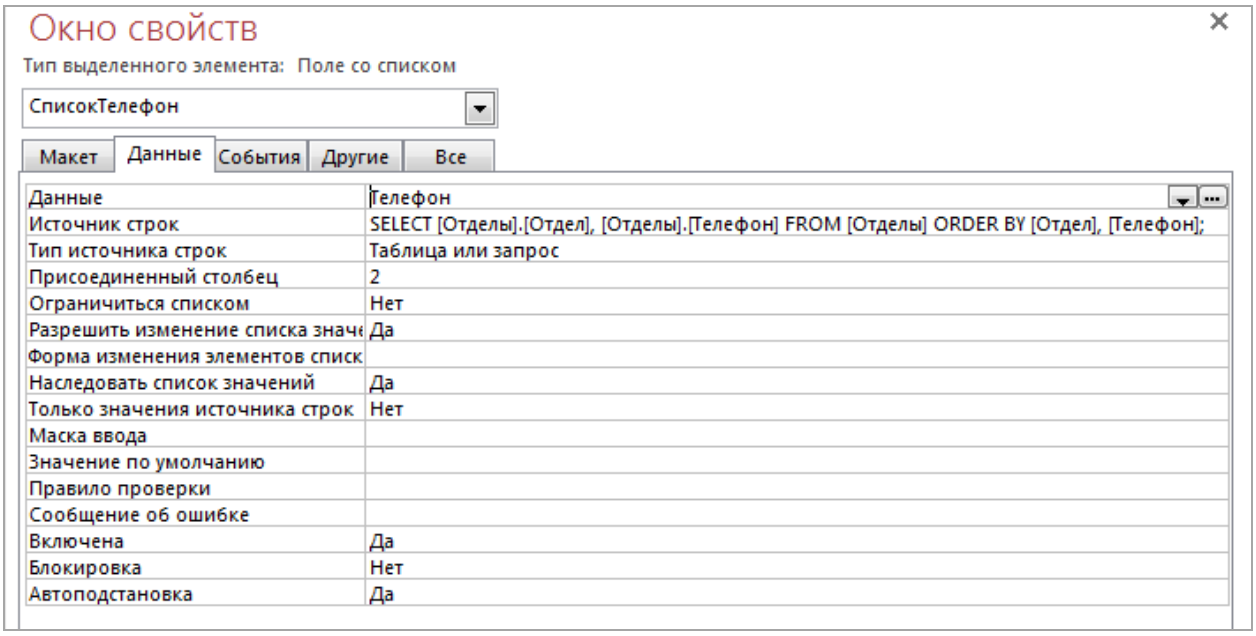


Рис. 102. Свойства поля со списком значений, выбираемых из таблицы и вносимых в ее столбец, не являющийся полем с подстановкой

Пример 28. *Создать на форме «Рекруты» поле со списком заданных значений Назначаемый разряд, не связанное ни с одним объектом БД*.

Выведите на форму в режиме Конструктора элемент **Поле со списком**.

1. В первом окне **Создание полей со списком** отметим переключатель **Будет введен фиксированный набор значений**. **Далее**.

2. Во втором окне заполним **столбец 1** значениями 1, 2, 3, 4, 5, 6, 7.

3. В следующем окне отметим переключатель **Запомнить значение .**

4. Далее назовем элемент «Назначаемый разряд», **Готово**.

5. Войдите в **Свойства** этого поля со списком и выберите значение **Да** для свойства **Ограничиться списком** (рис. 103). Обратите внимание, что свойство **Данные** осталось пустым.

6. Перейдите в режим Формы и убедитесь, что любое выбранное для любой записи значение остается тем же самым для всех остальных записей.

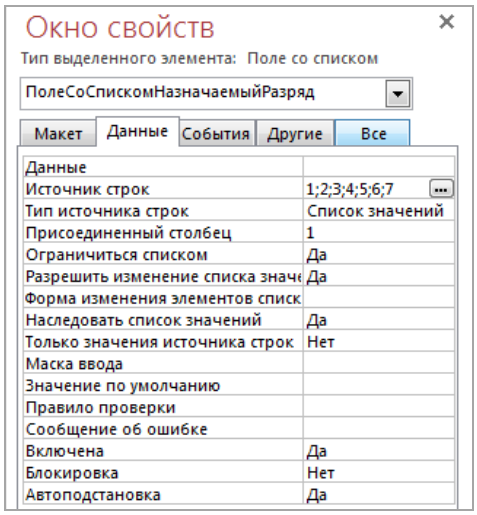


Рис. 103. Свойства поля со списком, не связанного ни с одним объектом БД

☑ Если создается поле со списком, не связанное ни с каким объектом БД (таблицей, запросом), то вместо переключателя **Сохранить в поле** нужно выбрать **Запомнить значение**. Но тогда при последующей работе с формой любое выбранное значение из списка будет *одним и тем же* для всех записей формы. Смысл такого выбора – немедленная реакция (реализуемая, как правило, программно) СУБД на этот выбор. Например, при выборе статуса «низкий» следует удаление данного сотрудника из ведомости с премией; для следующего сотрудника этот статус можно изменить и получить другую (немедленную) реакцию.

Сравните свойства полученных полей со списком (рис. 94, 95, 96), обратите внимание на зависимость значений свойств **Данные**, **Тип источника строк**, **Источник строк**, **Присоединенный столбец** и **Ограничиться списком** от постановки задачи.

Задание

55. Создайте поле со списком *Хобби* для каждого сотрудника организации. Попробуйте для разных копий формы три способа задания списка.

11.2. Добавление флажка, переключателя, выключателя

Назначение этих элементов управления, как и в случае поля со списком – немедленная реакция на включение-выключение или заполнение поля таблицы.

Но поле таблицы тогда должно иметь логический тип. Рассмотрим, например, флажок.

Откройте таблицу «Отделы» в режиме конструктора и добавьте поле *Наличие наград* типа **Логический**. Закройте таблицу, сохранив изменения.

Пример 29. *Создать флажок для заполнения поля Наличие наград.*

1. Откройте форму «Отделы» в режиме конструктора. На ленте Access – вкладка КОНСТРУКТОР – группа Элементы управления – кнопка **Флажок** ☑. Выделите мышью место для флажка на форме. Он появится вместе с элементом **Надпись** (с подписью «Флажок *N*»).

2. Откройте окно **Свойства** для этого флажка (не надписи!) и выберите вкладку **Данные**, строку **Данные** и в выпадающем списке поле *Наличие наград* (рис. 104). Закройте это окно.

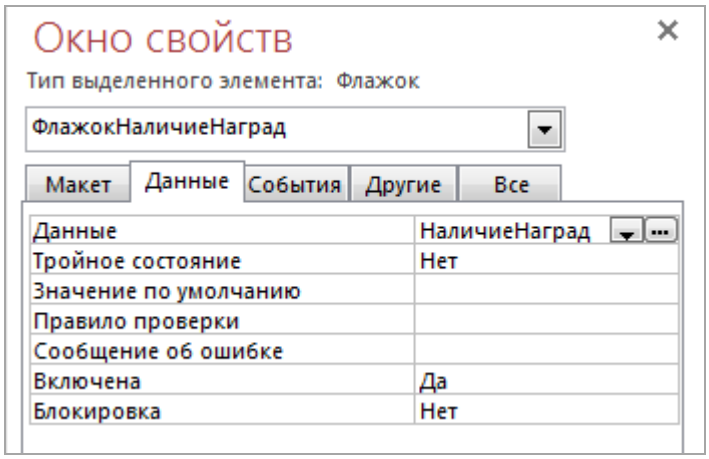


Рис. 104. Выбор источника данных для флажка

3. Измените подпись: вместо слова Флажок введите *Наличие наград* либо непосредственно на надписи, либо войдя в **Свойства** элемента **Надпись**, вкладка **Макет → Подпись**.

Перейдите в режим Формы и отметьте с помощью флажка наличие или отсутствие наград для каждого заведующего. Проверьте затем таблицу «Отделы», столбец *Наличие наград*.

☑ Флажок может быть не привязан к полю таблицы, а служить лишь для задания (программно) реакции на его включение.

☑ Флажок имеет значения: **Да (True)** – включен и **Нет (False)** – выключен.

Задание

56. Создайте в таблице «МоиСотрудники» логическое поле *В отпуске* и заполните его из формы с помощью соответствующего флажка.

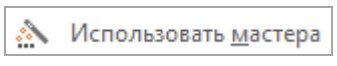
11.3. Создание группы переключателей

Группа переключателей – это отдельный элемент, имеющий свое имя, свойства, состоящий из нескольких элементов-переключателей, также обладающих собственными именами и свойствами. Изменение значения отдельного переключателя означает изменение значения всей группы. Особый интерес этот элемент имеет при программировании процедур-реакций СУБД на подобные изменения значений. Можно использовать группу переключателей и для добавления или обновления какого-либо поля таблицы.

Рассмотрим работу группы переключателей для занесения значений в поле таблицы.

Добавьте в таблицу «МоиСотрудники» в режиме Конструктора числовое поле (размер поля – Целое) *Рейтинг*. Закройте таблицу, сохранив изменения.

Пример 30. *Создать группу переключателей для заполнения поля Рейтинг таблицы «МоиСотрудники»*.

Откройте форму, созданную по таблице «МоиСотрудники» в режиме Конструктора. Нажмите кнопку 

на панели элементов, на **ленте Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** – затем кнопку **Группа переключателей** 

. Нарисуйте мышью на форме область для этой группы. Далее следуем инструкциям мастера:

1. В первом окне зададим подпись для каждого переключателя (построчно): Прекрасно, Хорошо, Удовлетворительно, Посредственно, Неважно, Безответственно, Хуже некуда. **Далее**.

2. Во втором окне на вопрос, задать ли переключатель, используемый по умолчанию ответим **Нет**. **Далее**.

3. В следующем окне подтвердим значения, соответствующие заданным характеристикам рейтинга (рис. 105). **Далее**.

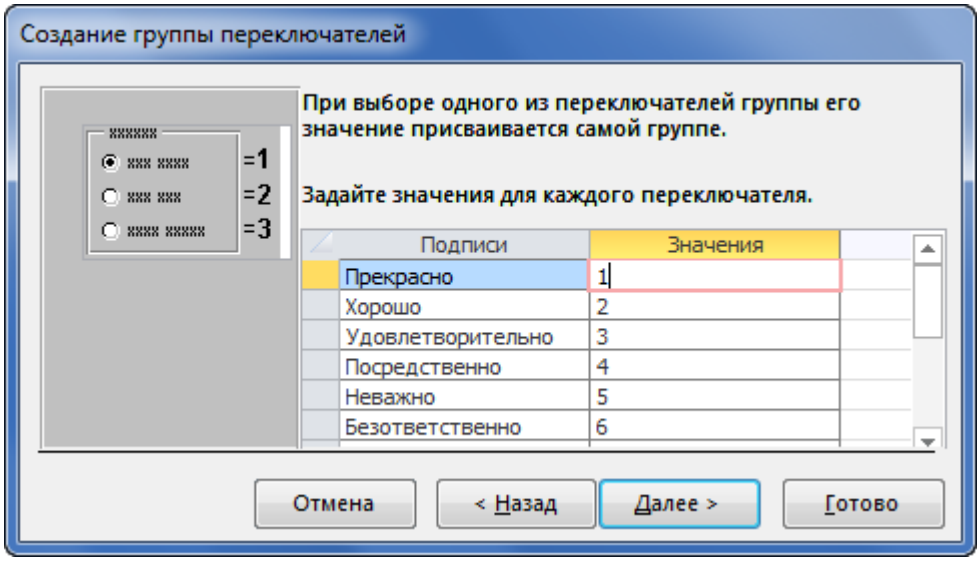


Рис. 105. Задание значений группе переключателей

4. В новом окне отмечаем **Сохранить значение в поле** и в выпадающем списке выбираем поле *Рейтинг*. **Далее**.

5. Теперь Мастер предоставляет возможность выбора элементов управления в группе. Выберем **Переключатели**. **Далее**.

6. Зададим подпись для всей группы: *Как работает*. **Готово**.

7. Отформатируйте эту подпись на форме – поменяйте шрифт, цвет и т.п.

8. Перейдите в режим формы и оцените работу каждого сотрудника (рис. 106).

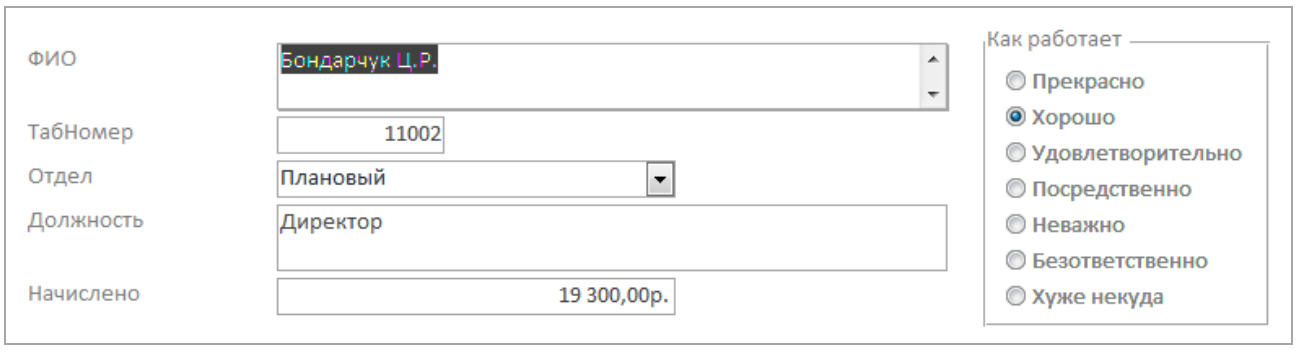


Рис. 106. Форма с группой переключателей

9. Проверьте в таблице «МоиСотрудники» значения поля Рейтинг. Обратите внимание, что в поле внесены не *подписи*, а *значения* переключателей из группы.

☑ Значение всей группы переключателей для каждой записи совпадает со значением выбранного переключателя. Для новой записи можно заранее выбрать значение по умолчанию: в окне **Свойства** для группы на вкладке **Данные** в строке **Значение по умолчанию** задать значение желаемого переключателя. Чтобы «погасить» *все* значения, нужно в эту строку ввести ноль (рис. 107):

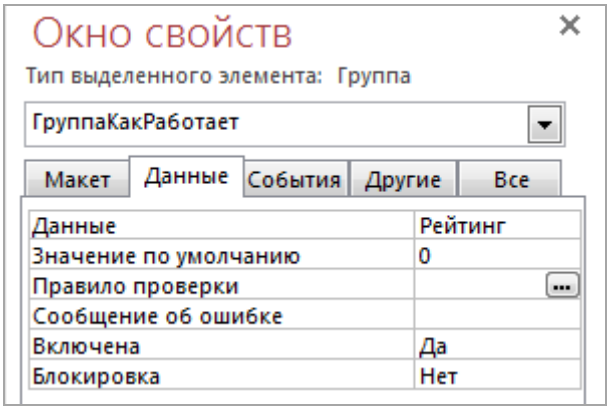


Рис. 107. Внесение значения по умолчанию для группы переключателей

Задание

57. Выведите на форме «Отделы» группу переключателей *Время для отпуска* со значениями Весна, Лето, Осень, Зима и заполните значения для каждого заведующего (поле в таблице назовите *Отпуск*).

11.4. Создание вкладок

Вкладки являются удобным средством выбора нужных сведений, минуя не требуемые в данный момент. Создадим новую форму для таблицы «Отделы», содержащую вкладки.

Создайте форму, используя конструктор: **Лента Access** → вкладка СОЗДАНИЕ → группа **Формы** → кнопка **Мастер форм** выбрать в списке

таблицу «Отделы». Перенесите на форму из списка полей поля *Отдел* и *Название*. Назовите форму «МоиОтделы».

Пример 31. *Создать несколько вкладок на форме «МоиОтделы», открывающие служебную, личную информацию и сведения о месте расположения отделов*.

Шаги выполнения:

1. На **ленте Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** → кнопка **Вкладки**  . Выделите мышью место для вкладок на форме.

2. На форме появятся две вкладки. Чтобы добавить еще, нажмите правую кнопку мыши на любом из названий вкладок или на пустом месте этой строки, выберите из контекстного меню **Вставить вкладку**. Повторите процедуру до получения четвертой вкладки.

3. Измените названия вкладок: по очереди вызывайте для каждой вкладки **Свойства** из контекстного меню (правой кнопкой мыши по выделенной вкладке), вкладка **Макет** и в строке **Подпись** задайте следующие названия: *Служебные сведения*, *Личные сведения*, *Место пребывания*, *Скрыть все*.

4. Выделите вкладку Служебные сведения. Перенесите на нее из списка полей формы поля *Заведующий* и *Телефон*. Отформатируйте их.

5. Выделите вкладку Личные сведения. Перенесите из списка полей *Фото*, *Статус* и *Наличие наград*. Компактно расположите эти данные.

6. Выделите вкладку Место пребывания. Используя элементы управления **Свободная рамка объекта** или **Рисунок**, добавьте изображение места пребывания отдела.

7. Вкладку Скрыть все оставьте пустой.

Перейдите в режим формы, проверьте работу вкладок и закройте сведения, выбрав вкладку Скрыть все.

Задание

58. Создайте форму по запросу «Выдать среднюю, минимальную и максимальную зарплату в каждом отделе», назовите ее «Статистика». Добавьте на полученную в этом разделе форму «МоиОтделы» вкладку *Наша статистика*. Поместите на эту вкладку форму «Статистика». Добавьте вкладку *Сотрудники отдела*, на которой разместите подчиненную форму «МоиСотрудники» (любым из известных способов).

Лабораторная работа № 12. Добавление кнопок

12.1. Добавление кнопки для открытия другой формы

На форму можно добавить кнопку для открытия другой (уже готовой) формы. Эта другая форма может быть связанной (соответствовать связанной, например, дочерней таблице) или не связанной. Рассмотрим, как создать кнопку для связанной формы.

Пример 32. *Создать на форме «Отделы» кнопку, открывающую сведения о сотрудниках отдела*.

На форме «Отделы» (или ее копии) нажмите кнопку 

на панели элементов, затем кнопку 

. Далее следуем инструкциям Мастера.

1. В первом окне выберите категорию **Работа с формой** и действия **Открыть форму**.

2. Во втором окне выберите форму «МоиСотрудники». **Далее**.

3. В третьем окне отметьте переключатель **Открыть форму для отобранных записей**. Если выбрать другой переключатель, то открываемая форма будет **несвязанной**. **Далее**.

4. В четвертом окне нужно отметить связанные поля, по которым идет соединение таблиц. Это поля «Отдел» в родительской и дочерней таблицах (рис. 108). Нажмите кнопку 

, затем **Далее**.

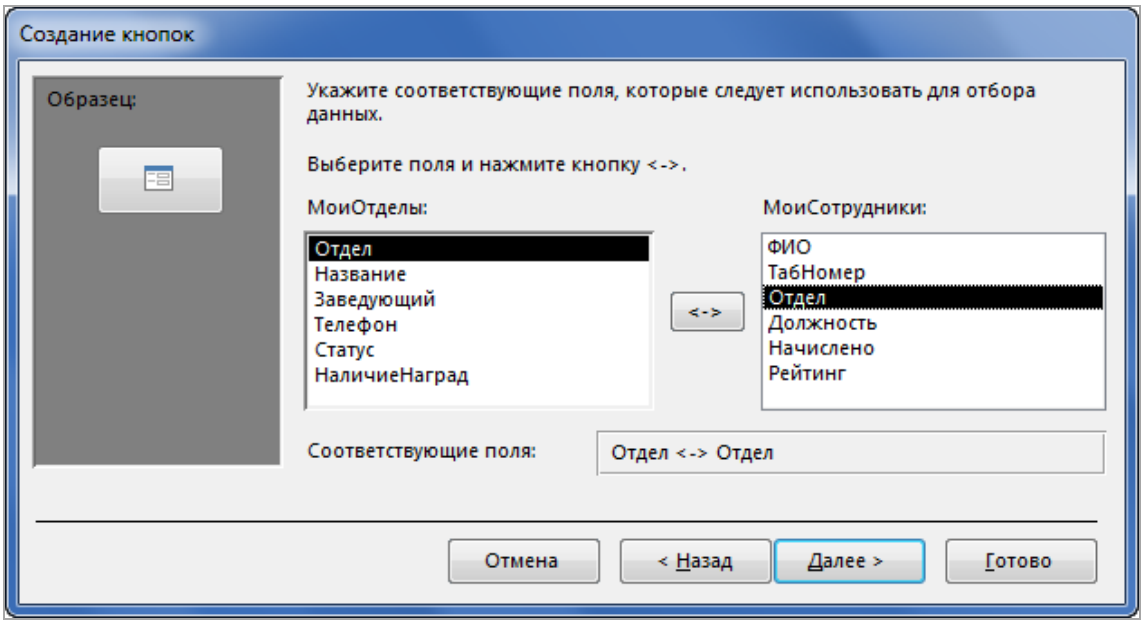


Рис. 108. Выбор полей, реализующих связь между таблицами - источниками данных

5. В пятом окне введите текст на кнопке: *Сотрудники отдела*. **Далее.**

6. Шестое окно служит для задания внутреннего имени кнопки. Для нашей задачи можно оставить предлагаемое имя. **Готово.**

Перейдите в режим формы и воспользуйтесь полученной кнопкой.

Задание:

59. Создайте кнопку «Друзья» на форме «МоиСотрудники», которая открывает несвязанную форму «Сотрудники» из БД «Борей».

12.2. Кнопки фильтрации данных

Часто бывает удобно отображать на форме не все данные таблицы или запроса, а лишь некоторые. Построим форму по таблице «МоиСотрудники» с кнопками, позволяющими фильтровать записи в форме. Пусть это будут кнопки с буквами алфавита. При нажатии любой такой кнопки в форме должны отображаться данные о тех сотрудниках, чьи фамилии начинаются на эту букву.

1. Создайте новую форму по таблице «МоиСотрудники» ленточного типа, выберите поля *ФИО, Отдел, Должность, Начислено (При помощи Мастера*

форм отберите необходимые записи и укажите внешний вид формы Ленточный). Откройте ее в режиме Конструктора. Увеличьте высоту области заголовка формы, чтобы поместились все кнопки.

2. Отключите кнопку **Мастер Элементов** на панели элементов.

3. Выберите на панели элементов элемент управления **Кнопка** и затем щелкните левой кнопкой мыши в левом верхнем углу области заголовка. Откройте (правой кнопкой мыши по выделенной кнопке) **Свойства → Макет → Подпись** и введите подпись А (в русском алфавите).

4. Сделайте эту надпись полужирной, затем подведите указатель мыши к любой границе кнопки. Когда он превратится в двунаправленную стрелку, дважды щелкните левой кнопкой мыши. Кнопка станет маленькой и примет квадратную форму.

5. Создайте макрос, управляющий фильтрацией. **Лента Access →** вкладка СОЗДАНИЕ **→** группа **Макросы и код →** кнопка **Макрос** 

. Откроется **Конструктор макросов.**

6. Создайте несколько вложенных макросов в одном объекте, каждый макрос будет иметь свое имя. Для этого нажмите на панели **Каталог макрокоманд** в разделе **Управление** нажмите кнопку **Вложенный макрос.** Введите имя первого макроса: А. В выпадающем списке **Добавить новую макрокоманду** выберите пункт **ПрименитьФильтр**. В Конструкторе появятся поля, в которых можно задать условия фильтрации записей. Введите в поле Условие отбора= строку Ucase([ФИО]) Like 'А*' (А – русская!).

7. Аналогичные макросы задайте для всех букв алфавита. Каждый макрос будет состоять из макрокоманды **ПрименитьФильтр**, но в условиях отбора будут присутствовать разные буквы (рис. 109).

8. Сохраните полученные макросы в одном объекте (обычным сохранением файла) с именем «ФильтрСотрудников».

9. Подключим этот макрос к кнопке формы. Выделите кнопку с буквой А, откройте ее **Свойства** и вкладку **События**. Выберите событие **Нажатие кнопки**, и в раскрывающемся списке этой строки выберите элемент **ФильтрСотрудников.А**. Закройте **Свойства**. Теперь при нажатии этой кнопки в форме будет выполняться фильтрация записей.

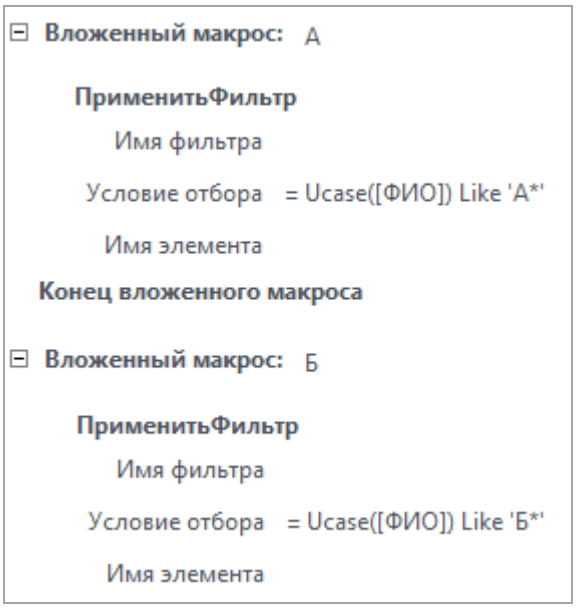


Рис. 109. Задание группы макросов «ФильтрСотрудников»

10. Создадим кнопки для остальных букв с помощью копирования кнопки «А». Для этого выделите эту кнопку, нажмите <Ctrl>+<C> или <Ctrl>+<Insert> (копирование в буфер), затем <Ctrl>+<V> или <Ctrl>+<Shift> (вставка). Переместите мышью новую кнопку в нужное место, измените ее подпись на Б. Снова выполним выбор **Свойства → События → Нажатие кнопки → ФильтрСотрудников.Б**.

11. Продолжите эту процедуру для получения всех кнопок (рис. 110).

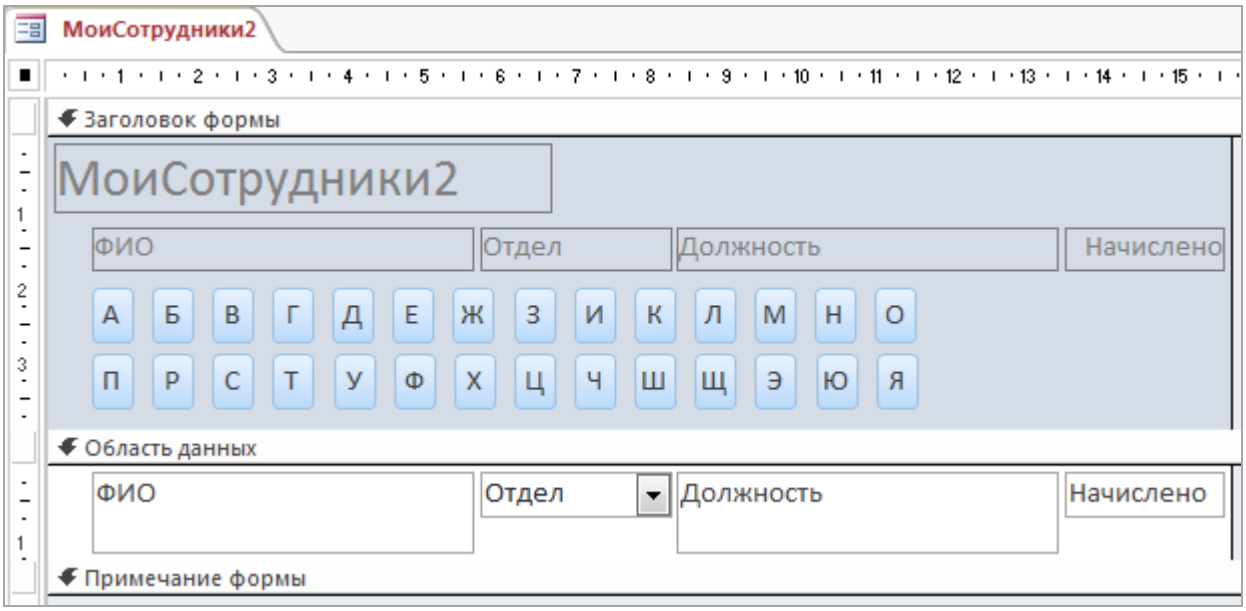


Рис. 110. Добавление кнопок фильтрации на форму

12. Переключитесь в режим Формы и проверьте, как действуют созданные кнопки.

12.3. Кнопка выбора информации, связанной с одной записью формы

Создадим кнопку, при нажатии которой будет открываться связанная форма, например, «Отделы», соответствующая тому сотруднику, который выделен в списке.

1. **Лента Access** → вкладка КОНСТРУКТОР → группа **Элементы управления** → включите **Использовать мастера** 

и добавьте новую кнопку 

на форму из предыдущего упражнения в раздел **Примечание формы**.

2. В первом окне Мастера в списке **Категории** выберите **Работа с формой**, в списке **Действия → Открыть форму**, **Далее**.

3. Во втором окне выберите форму «Отделы». **Далее.**

4. В следующем окне выберите переключатель **Открыть форму для отобранных записей .**

5. Далее, в списках полей двух форм выберите поля, по которым идет связь (первичный и внешний ключи) и нажмите кнопку с двунаправленной стрелкой между списками (рис. 111).

6. В остальных окнах мастера задайте рисунок на кнопке или текст «Просмотр». **Готово.**

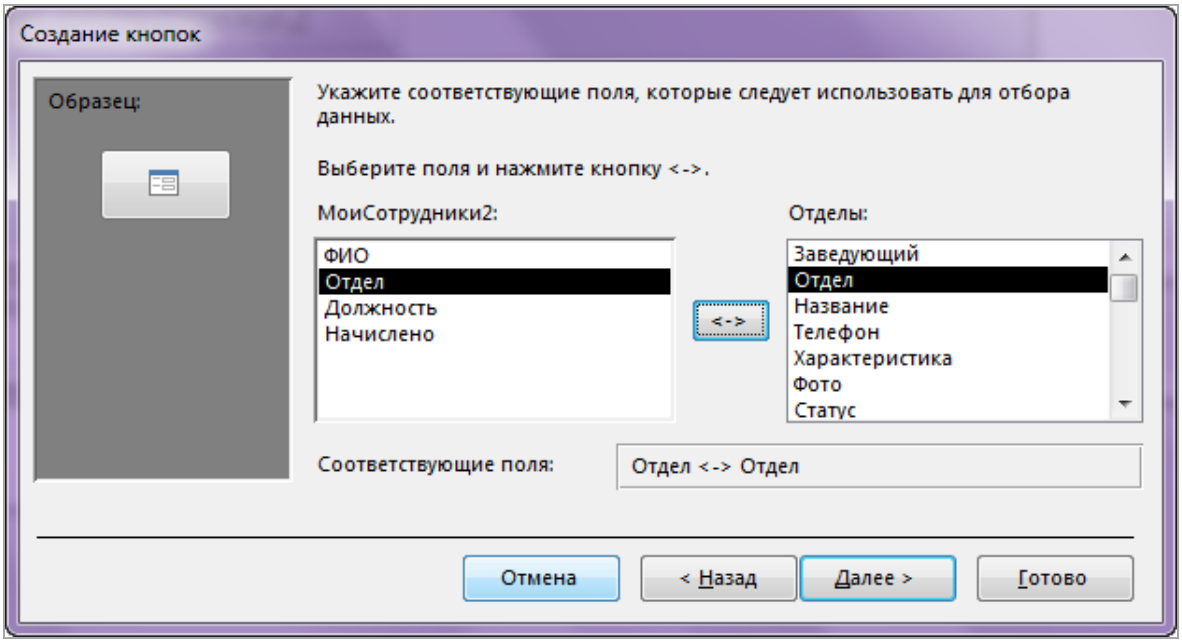


Рис. 111. Выбор связанных полей

7. Перейдите в режим формы, выберите букву фильтра и в полученном списке записей выделите любую. Щелкните мышью созданную кнопку. Получим сведения об отделе, в котором работает данный сотрудник.

Задание

60. Создайте на старой форме «МоиСотрудники» поисковую кнопку, используя добавление кнопки с помощью Мастера элементов; в списке **Категории** окна Мастера выберите **Переходы по записям**, а в списке **Действия → Найти запись.**

12.4. Настройка области навигации

В Office Access реализована новая функция «Область навигации». **Эта область заменяет окно базы данных, и ее можно использовать вместо**

кнопочных форм. Область навигации работает с новой моделью пользовательского интерфейса, используемого Office Access. Эта модель (называемая моделью однодокументного интерфейса (SDI)) размещает все открытые объекты : формы, отчеты и т.д. в одном окне и добавляет для каждого объекта свою вкладку. Если открыто несколько объектов, для переключения между этими объектами используются вкладки.

В отличие от кнопочных форм, область переходов продолжает отображаться до тех пор, пока пользователь ее не скроет. Для отображения и скрытия этой области используется кнопка **Открыть/закрыть границу области переходов** или клавиша **F11**.

Любой объект в области навигации можно использовать, дважды щелкнув его. Если объект щелкнуть правой кнопкой мыши, появится контекстное меню, которое позволяет выполнить различные действия, например, открыть объект в режиме конструктора.

Объекты в базе данных можно организовать в группы и категории. Группы позволяют упорядочить и связать объекты в области навигации, а категории — упорядочить и связать группы (рис. 112).

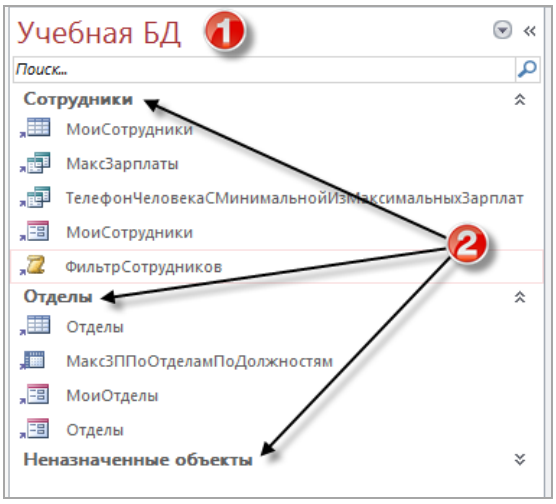


Рис. 112. Пользовательская панель навигации: 1 – категория; 2 – группы

Щелчок левой кнопкой мыши по заголовку категории открывает меню перехода к категориям и фильтрам по группам для этих категорий. По-

умолчанию, Access уже имеет следующие группы: **Таблицы и связанные представления, Тип объекта, Дата создания, Дата изменения, Другие**. Обычно используется категория **Тип объекта** и фильтр **Все объекты Access**. Фильтры зависят от текущей категории (рис. 113).

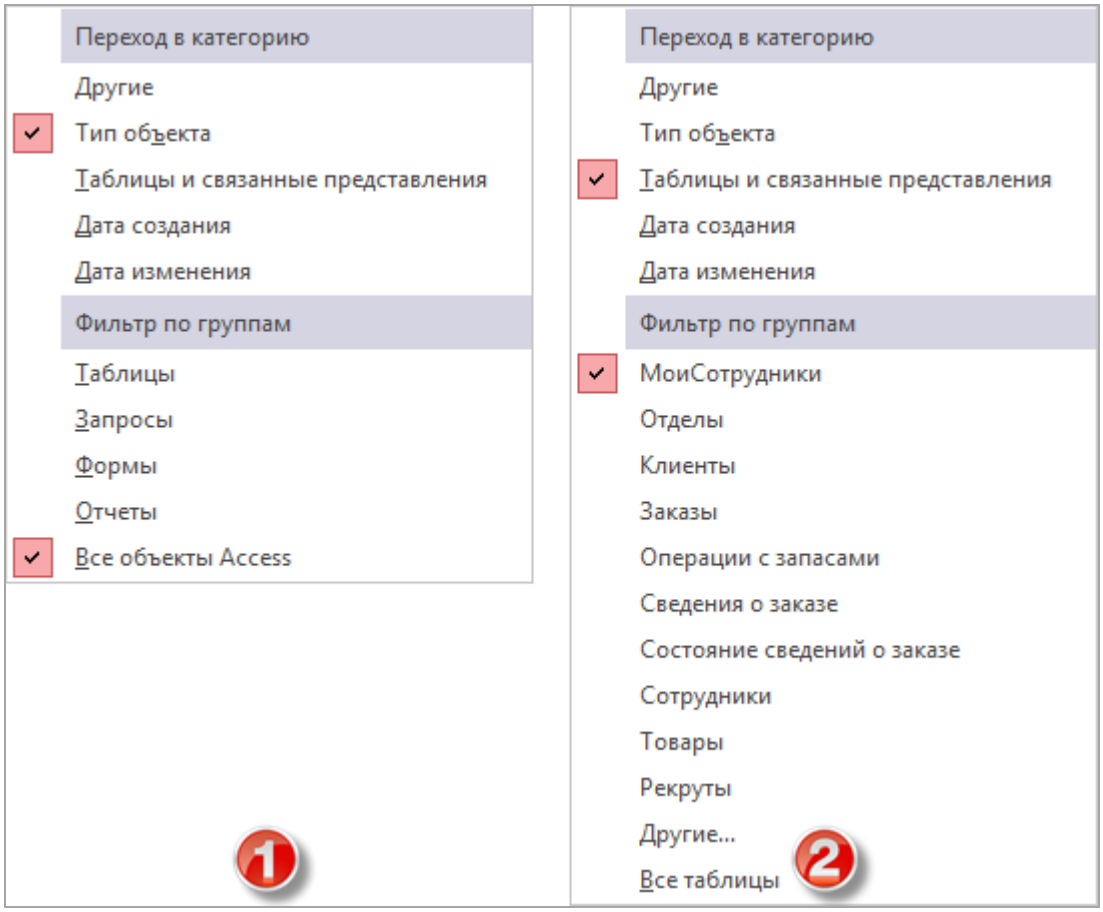


Рис. 113. Контекстнозависимость доступных фильтров от активной категории:
1 – фильтры для категории Тип объекта;
2 – фильтры для категории Таблицы и связанные представления.

12.4.1. Создание пользовательских категорий и групп

Категория **Другие** предназначена для пользовательской настройки. Перейдите в данную категорию. Данная категория содержит две группы: **Настраиваемая группа1** (изначально пуста) и **Неназначенные объекты** (все объекты БД, которые могут быть в произвольном списке перенесены в группу **Настраиваемая группа1**) (рис. 114). Объекты, находящиеся в системных группах и в группе **Неназначенные объекты** являются непосредственными,

физическими объектами. Их удаление приведет к удалению данных объектов из БД. Элементы, которые отображаются в пользовательских категориях — это ярлыки, которые указывают на объекты базы данных. Удаление ярлыков не влияет на сам объект. Одна группа может содержать только один ярлык на один и тот же объект. Многие группы могут содержать ярлык для одного и того же объекта.

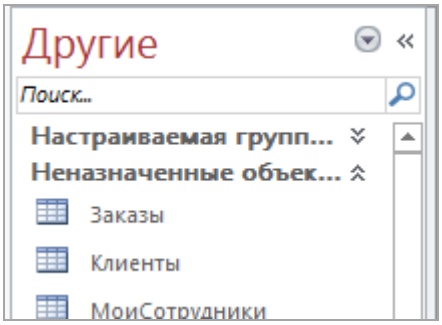


Рис. 114. Категория «Другие» и её группы

Переименуйте группу в «Сотрудники» (пункт **Переименовать** в контекстном меню группы, либо нажатие клавиши <F2> на клавиатуре); перетащите в эту группу все объекты БД из группы **Неназначенные объекты**, которые связаны с сотрудниками (например, таблицу и форму «МоиСотрудники»). Создадим ещё одну группу, для этого из контекстного меню заголовка категории (щелчок правой кнопкой мыши по **Другие**) выберем пункт  Параметры навигации... , откроется окно, представленное на рис. 115.

12.4.2. Скрытие и отображение объектов и групп

Объекты и группы можно скрыть. Для скрытия объекта, в его контекстном меню (щелкните объект правой кнопкой мыши) выберите пункт **Скрыть в этой группе**. Для скрытия всей группы целиком щелкните группу правой кнопкой мыши и открывшемся контекстном меню выберите **Скрыть.**

Для отображения скрытых групп в окне **Параметры навигации** установите флажок напротив нужных групп. Для отображения скрытых объектов сначала установите флаг **Показывать скрытые объекты** в окне

Параметры навигации, а затем в контекстном меню объекта выберите **Показать в этой группе**.

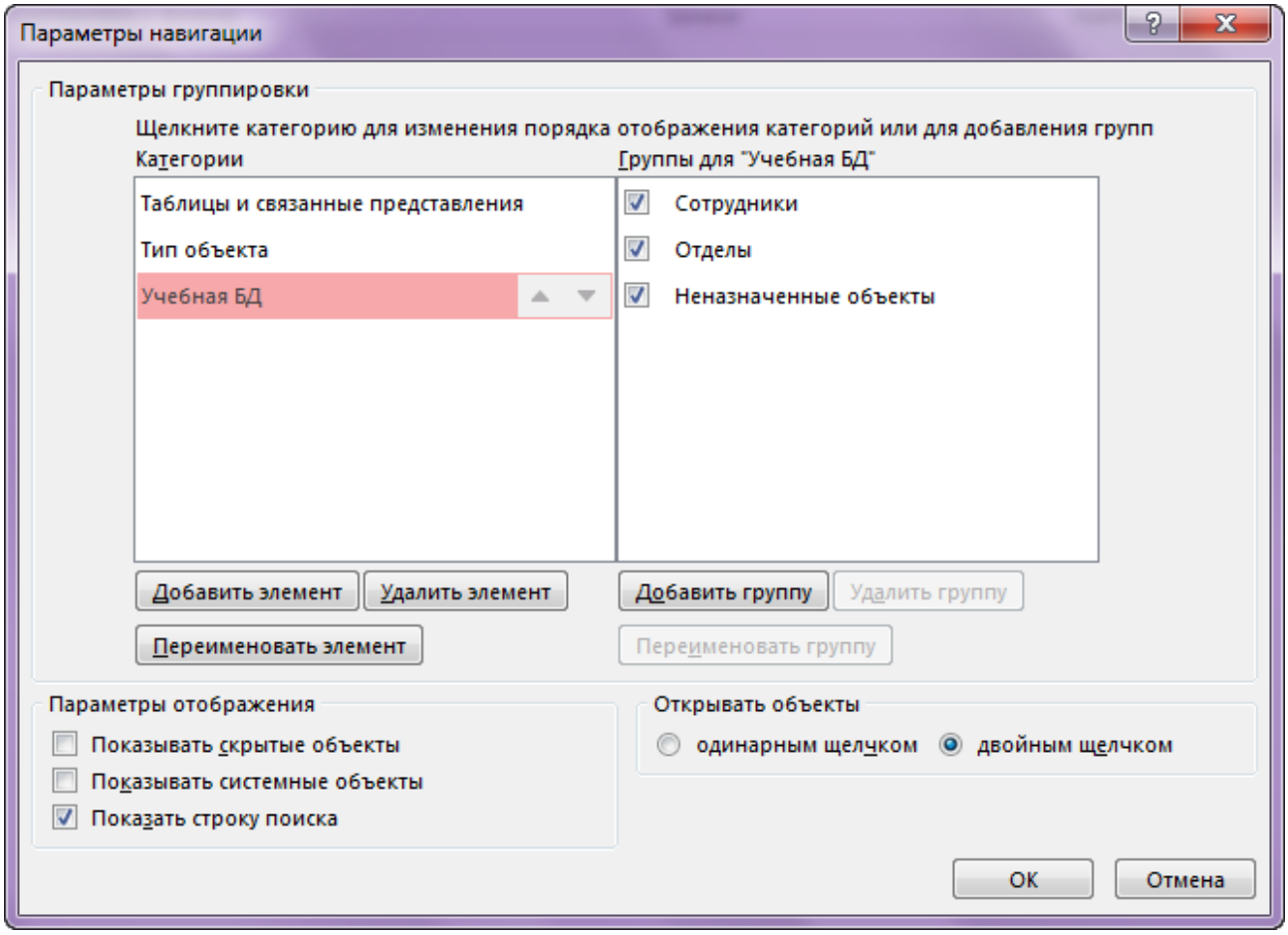


Рис. 115. Окно параметры навигации

В левой панели **Категории** выберите группу **Учебная БД**, нажмите кнопку **Добавить группу**, введите имя «Отделы». Перетащите мышью объекты, связанные с таблицей «Отделы» в группу **Отделы** из группы **Неназначенные объекты**.

12.4.3. Автоматическое открытие определенной категории и группы при открытии БД

Можно автоматически открывать определенную группу или категорию Области навигации, используя макрокоманду **ПерейтиК**. (**Лента ACCESS** → вкладка СОЗДАНИЕ → кнопка **Макрос**, выбрать из списка макрокоманду

ПерейтиК, установить необходимые категорию и группу Области навигации. Обязательно сохранить макрос с именем **AutoExec** – такой макрос выполняется при открытии БД)

Для ярлыков в Области навигации можно устанавливать запрет на открытие объектов в режиме конструктора. Для этого щелкните правой кнопкой по ярлыку, выберите Свойства и установите флаг **Отключить ярлыки режима конструктора**.

Важно:

- Используя эти функции, можно эффективно управлять организацией области навигации и сделать ее более удобной в использовании. Однако эти функции не предотвратят внесение нежелательных изменений в базу данных злоумышленниками. Более того, они не предотвращают несанкционированный доступ к данным. Целью внедрение этих функций для управления областью навигации является повышение удобства пользования базой данных, а не повышение ее защиты.
- При использовании макроса для управления областью навигации и ее фиксации помните, что пользователи могут отключать макрос при запуске, нажав и удерживая клавишу <SHIFT>.
- Хотя ярлыки области навигации имеют свойство **Отключить режим конструктора** для ярлыков, сами объекты базы данных такого свойства не имеют. Можно скрывать категории и группы, которые содержат объекты базы данных, или скрывать сами объекты, однако пользователи могут их отобразить.

12.4.4. Управление областью навигации с помощью макроса

Приложение Access содержит несколько макрокоманд для управления областью навигации. С помощью макрокоманды можно выполнять следующие действия:

- выборочно отображать или скрывать категории;
- фиксировать область навигации, чтобы предотвратить внесение

случайных изменений;

- переходить в категорию или группу;
- скрывать область навигации.

Использование макрокоманд для управления отображением в области навигации

1. Выполните одно из указанных ниже действий.
- Для создания нового макроса на вкладке **Создать** в группе **Макрос и код** нажмите кнопку **Макрос**.
- Чтобы добавить команду в существующий макрос, в области навигации щелкните макрос правой кнопкой мыши, а затем выберите **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** выберите пункт **Показать все действия**.
3. В конструкторе макроса из выпадающего списка выберите **ЗадатьОтображаемыеКатегории**. Эта макрокоманда появится в конструкторе макроса.
4. В поле **Показать** выберите **Да**, если необходимо отобразить категорию в области навигации, или **Нет**, чтобы ее не отображать.
5. В поле **Категория** выберите имя категории, которую необходимо отобразить или скрыть.
6. Повторите шаги с третьего по пятый для каждой категории, которой необходимо управлять с помощью макроса.
7. В конструкторе макроса из выпадающего списка выберите параметр **ЗафиксироватьОбластьПереходов**. Эта макрокоманда появится в конструкторе макроса.
8. Укажите значение для аргумента **Блокировка**. Чтобы заблокировать область навигации, выберите значение **Да.**

Использование макрокоманды для перехода в категорию или группу

1. Выполните одно из указанных ниже действий.
- Для создания нового макроса на вкладке **Создать** в группе **Макрос и код** нажмите кнопку **Макрос**.
- Чтобы добавить команду в существующий макрос, в области навигации щелкните макрос правой кнопкой мыши, а затем выберите команду **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** выберите пункт **Показать все действия**.
3. В конструкторе макроса из выпадающего списка выберите параметр **ПерейтиК**. Эта макрокоманда появится в конструкторе макроса.
4. В поле **Категория** щелкните имя категории, в которую необходимо перейти.
5. Если необходимо перейти в определенную группу в категории, щелкните стрелку в поле **Группа**, а затем выберите нужную группу.

12.4.4.3. Использование макрокоманды для скрытия Области навигации

Если необходимо создать собственную систему навигации, можно скрыть область навигации, используя макрокоманду **ВыполнитьКоманду** и аргумент **ОкноСкрыть**.

1. Выполните одно из указанных ниже действий.
- Для создания нового макроса на вкладке **Создать** в группе **Макрос и код** нажмите кнопку **Макрос**.
- Чтобы добавить команду в существующий макрос, в области навигации щелкните макрос правой кнопкой мыши, а затем выберите команду **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** выберите пункт **Показать все действия**.

3. В конструкторе макроса из выпадающего списка выберите параметр **ЗапускКомандыМеню**. Эта макрокоманда появится в конструкторе макроса.

4. В поле **Команда** выберите **ОкноСкрыть.**

Примечание. Выключить область навигации можно также с помощью установки флажка **Область навигации** в диалоговом окне **Параметры Access**. При использовании макрокоманды **ЗапускКомандыМеню** с аргументом **ОкноСкрыть** область навигации будет скрыта независимо от того, установлен флажок **Область навигации** или нет.

Задание 61. Настройте Область навигации для вашей БД. Создайте категории, группы, распределите объекты. Отключите конструктор для ярлыков. Скройте все категории кроме вашей. Создайте макрос, открывающий вашу категорию и группу при открытии БД. Создайте макрос, скрывающий все категории и группы кроме ваших.

Задание 62. Создайте форму с кнопками, которые открывают другие формы. Установите эту форму формой просмотра (**ФАЙЛ → Параметры → Текущая база данных → Форма просмотра**).

Задание 63. Создайте макрос, открывающий при открытии БД вашу форму (используйте макрокоманду ОткрытьФорму макроса AutoExec).

Лабораторная работа № 13. Связь с другими приложениями

13.1. Создание гиперссылки на форме

Гиперссылка позволяет задавать адрес ссылки на документ или файл в интернете, интрасети, локальной сети или на локальном компьютере. Гиперссылки удобны для быстрого перехода из одной формы (или отчета) в другую форму (или отчет) или во внешнюю среду, например, на Интернет-страницу.

Гиперссылку обычно связывают с элементами управления **Надпись, Кнопка**, **Рисунок**. Гиперссылки могут указывать на объект текущей БД или на внешний объект, задаваемый URL-адресом.

Для создания гиперссылки нужно:

1. Поместить на форму подходящий элемент управления, представляющий эту ссылку. Выберите, например, элемент «Рисунок».

2. Войти в окно **Свойства → Макет** этого элемента и найти строки **Адрес гиперссылки** или **Дополнительный адрес**.

Адрес гиперссылки используется, если требуется перейти во внешнюю среду, а *Дополнительный адрес*, если переход осуществляется на объект текущей базы данных. Выберите для примера **Дополнительный адрес**.

3. Нажать на кнопку обзора … в выбранной строке свойств. Появится окно **Добавление гиперссылки**; на его левой панели «Связать с» выбрать нужный тип объекта. В нашем случае выбираем «объектом в базе данных» и на правой панели – объект «Формы». Раскройте список форм (щелкнув по значку +), выберите форму «Сотрудники», взятую из БД «Борей», нажмите **ОК**.

Если бы требовалось связать гиперссылку с Интернет-сайтом, то нужно было на левой панели выбрать «новой страницей» и в появившемся окне набрать адрес страницы, например, http://www.yandex.ru.

В окне **Свойства** рисунка-гиперссылки в строке **Дополнительный адрес** теперь появился целевой объект – Form Сотрудники (рис. 116).

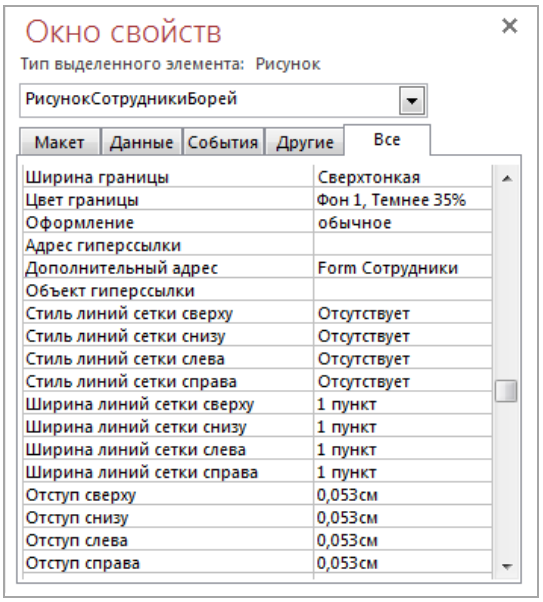


Рис. 116. Окно свойств элемента-гиперссылки

Перейдите в режим Формы и убедитесь, что гиперссылка работает (щелкнув левой кнопкой мыши на рисунке).

Адрес гиперссылки можно связать и с файлом другого типа, например, с документом Word, который откроется при активизации ссылки.

13.2. Связи с приложениями MS Office, экспорт и импорт таблиц

13.2.1. Вставка таблиц в Excel и Word

Для вывода таблиц в приложения Excel или Word можно просто воспользоваться стандартной процедурой копирования и вставки или выполнить следующие шаги [8]:

1. Выделить таблицу в окне БД.

2. Для экспорта в Word: **Лента ACCESS** → вкладка ВНЕШНИЕ ДАННЫЕ → группа **Экспорт → Дополнительно → Word**.

Для экспорта в Excel: **Лента ACCESS** → вкладка ВНЕШНИЕ ДАННЫЕ → группа **Экспорт → Excel**.

13.2.2. Импорт таблиц из Excel

Можно импортировать электронную таблицу целиком или только ее часть, как в новую, так и в существующую таблицу Access.

При импорте в новую таблицу, первая строка электронной таблицы должна содержать заголовки столбцов, они будут использованы в качестве имен полей новой таблицы Access.

Типы данных для полей новой таблицы Access определяет, анализируя значения в первых импортируемых строках:

алфавитно-цифровая информация электронной таблицы сохраняется в Access в текстовых полях типов Короткий/Длинный текст;

числовые данные – в числовых полях со свойством **Размер поля**, установленным в **Двойное с плавающей точкой**;

числа в денежном формате – в денежных полях;

значения дат и времени – в полях типа **Дата/время**.

Если в первых строках столбца присутствуют смешанные данные, Access импортирует столбец в текстовое поле.

При импорте диапазона или всей электронной таблицы в *существующую* таблицу Access лучше вначале импортировать таблицу Excel целиком в *новую* таблицу Access, затем отредактировать ее в режиме Конструктора, подогнав типы (домены) полей и только после этого вставить эти данные в существующую таблицу, используя запрос на добавление.

Процедура импорта из Excel такова:

a. Откройте в Access окно БД.

b. **Лента ACCESS →** вкладка ВНЕШНИЕ ДАННЫЕ **→** группа **Импорт и связи → Excel**.

c. В открывшемся укажите путь к файлу-источнику, нажав кнопку «Обзор…».

d. Укажите куда должен быть произведен импорт (в новую таблицу в текущей БД, в имеющуюся таблицу, либо осуществить связывание с листом Excel).

e. В первом диалоговом окне Мастера выберите переключатель **Листы,** если импортируется весь файл, или **Именованные диапазоны**, если импортируется диапазон, имевший в Excel имя. Нажмите **Далее.**

f. В следующем окне установите флажок **Первая строка содержит заголовки столбцов**, если вы хотите использовать значения первой строки в качестве имен полей таблицы Access. **Далее.**

g. Просмотрите и, при необходимости, настройте импорт отдельных полей (установите необходимые имена полей, их типы данных, индексы, а также необходимость пропуска некоторых полей). **Далее.**

h. Определите первичный ключ (автоматически, по импортируемому полю, либо без создания ключа). **Далее.**

i. Укажите имя таблицы для импорта (если импорт будет производиться в новую таблицу). **Готово**.

Если анализ выдает ошибки, то можно завершить работу и с ошибками, исправив затем полученную таблицу в Access, или вернуться в Мастер, исправив Excel-таблицу.

13.2.3. Связь с таблицами других БД Access

Для создания связи с таблицами других БД Access нужно:

1. Открыть в Access окно исходной БД (базы-приемника).

2. **Лента ACCESS →** вкладка ВНЕШНИЕ ДАННЫЕ **→** группа **Импорт и связи → Access**.

3. В открывшемся окне указать путь к файлу БД, используя кнопку **Обзор…**.

4. Установите переключатель **Создать связанную таблицу для связи с источником данных. ОК**.

5. Выделить нужные таблицы и нажать **ОК**.

В результате выбранные таблицы добавятся в исходную базу. Процедура импорта (при установлении переключателя **Импорт таблиц, запросов, форм, отчетов, макросов и модулей в текущую БД**) также закончится добавлением таблиц. Однако *импортированные* таблицы будут отсоединены от приложения, их создавшего. Что же лучше выбрать – связанные или импортированные таблицы?

Связь выбирают, если:

- внешний файл-таблица используется совместно с другими пользователями, которые могут ее изменять;
- файл модифицируется с помощью другой СУБД;
- файл расположен на другом компьютере (например, сервере) и его объем слишком велик;
- решено разместить данные и остальные объекты приложения в двух отдельных БД.

Импорт выбирают, если:

• в разрабатываемом приложении поля таблицы должны иметь другой тип или размер, чем тот, который выбирает Access при установке связи;

• для работы приложения нужны ключевые поля, отличные от определенных для внешних ключей импортируемых таблиц;

• для работы приложения нужно, чтобы в поле таблицы, являющемся для внешнего файла первичным ключом, допускалось дублирование данных;

• у пользователей приложения нет интерактивного доступа к файлу БД в реальном времени, связь с ним установить просто нельзя.

При связывании некоторые форматы доступны только для чтения. Подробнее см. табл. 2.

Таблица 2

Внешние источники, данные которых можно импортировать или связывать

	Импорт	Связывание
Microsoft Excel	Да	Да (только для чтения)
Microsoft Access	Да	Да
Базы данных ODBC (например, SQL Server)	Да	Да
Текстовые файлы или файлы данных с разделителями (CSV)	Да	Да (только для добавления новых записей)
Список SharePoint	Да	Да
XML	Да	
Службы данных		Да (только для чтения)
Документ HTML	Да	Да
Папка Outlook	Да	Да

Контрольные работы

По базе данных «Учебная»:

1. Выбрать среднюю, минимальную, максимальную зарплату сотрудников с учетом 14% подоходного налога и 1,5 – страхового вычета.
2. Выбрать суммарную зарплату инженеров и бухгалтеров 2-го и 1-го отделов, фамилии которых начинаются на «П».
3. Построить перекрестный запрос, содержащий сведения о минимальной зарплате среди бухгалтеров и инженеров по каждому отделу.
4. Определить сотрудника (фамилия), получающего максимальную зарплату (исключая директора).
5. Найти пары сотрудников, у которых совпадает должность.
6. Построить таблицу с полями *Должность*, *Максимальная зарплата*, *Дата утверждения* для каждой должности из таблицы «Мои Сотрудники». Связать ее с таблицей «Мои Сотрудники».
7. Определить, сколько человек имеют должность, утвержденную до 20.01.04.
8. С помощью запроса на изменение заменить зарплату инженеров 2-го и 3-го отделов на 2500 р.
9. Найти минимальную среди максимальных зарплат по каждой должности.

По базе данных «Борей»:

Вариант 1

1. По таблице «Товары» («Борей») определить:
   А) Сколько всего на складе грецких орехов?
   Б) Какие поставщики прекратили поставки?
   В) Какова общая стоимость имеющихся товаров каждого типа?
2. Какие фирмы (название) заказали товар «Кофе»:
   А) просто;
   Б) ещё не оплачены.
   В) сколько клиентов делали заказы в 2006-м году?

3. Сколько должен заплатить каждый клиент за свои заказы с учетом стоимости доставки?
4. Удалить клиентов, все заказы которых находятся в состоянии Закрыт (создать копии соответствующих таблиц).
5. Добавить в полученную в результате задания 4 таблицу клиентов, чьи фамилии начинаются на букву «Е».
6. Выдать названия клиентов для определенного города получателя (город задавать параметром).
7. Построить перекрестный запрос: для каждого поставщика и категории товаров определить среднюю цену товара.

Вариант 2

1. По таблице «Товары» (Борей) определить:

a) Сколько товаров каждой категории на складе?
b) По каким категориям товаров не прекращены поставки?
c) Сколько поставщиков у товаров каждой категории?

2. Какие клиенты (название) заказали товар «Кондитерские изделия» и «Кофе»:

a) просто;
b) позже марта 2006 года.
c) сколько клиентов делали заказы по каждой зафиксированной дате?

3. Какова средняя стоимость заказов каждого клиента с учетом стоимости доставки и скидки?
4. Изменить заказы, содержащие «Хлебобулочные изделия», заменив этот товар на «Кондитерские изделия».
5. Удалить товары, поставки которых прекращены (используя копию таблицы «Товары»).
6. Определить стоимость доставки в определенный город получателя (город задавать параметром).
7. Используя перекрестный запрос, определить для каждой категории товаров, по которой не прекращены поставки, и каждого поставщика максимальную цену товара.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дейт К. Дж. Введение в системы баз данных. М.: Вильямс, 2006.
2. Конноли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. М.: Вильямс, 2000.
3. Кузнецов С. Базы данных. М.: Academia, 2012.
4. Michael Alexander, Access 2013 Bible. WILEY, 2013.
5. Microsoft. Центр разработки Office. Технические статьи по Access 2013 // http://msdn.microsoft.com/ru-ru/library/office/ , 2014.
6. Microsoft. Центр поддержки Access // http://office.microsoft.com/ru-ru/access-help , 2014.
7. Вейскас Джон. Эффективная работа: Microsoft Office Access 2003. – СПб.: Питер, 2005.
8. Харитонова И., Рудикова Л. Microsoft Office Access 2007. Санкт-Петербург: БХВ, 2008.

ПРИЛОЖЕНИЕ 1

Типы данных для приложений Access и их соответствие MS SQL Server

Тип данных	Параметр свойства подтипа	Описание	Соответствующий тип данных SQL Server 2012
Счетчик	Н/Д	Уникальное значение, создаваемое Access для каждой новой записи	int
Короткий текст	Н/Д	Буквенно-цифровые данные, состоящие из 1–4000 символов (ограничение по умолчанию —255 символов)	nvarchar (от 1 до 4000 символов)
Длинный текст	Н/Д	Буквенно-цифровые данные объемом не более 2^{31}-1 байт	nvarchar(max)
Числовой	Целое число (без десятичных знаков)	Числовые данные	int
Числовой	Число с плавающей запятой (переменное количество десятичных знаков)	Числовые данные	double
Числовой	Число с фиксированной запятой (6 десятичных знаков)	Числовые данные	decimal(28,6)
Дата и время	Дата	Значения дат	date
Дата и время	Время	Значения времени	time(3)
Дата и время	Дата и время	Значения даты и времени	datetime2(3)
Денежный	Н/Д	Денежные данные	decimal(28,6)
Логический	Н/Д	Логические данные (да/нет)	bit (значение по умолчанию — Ложь)
Гиперссылка	Н/Д	Адрес ссылки на документ или файл в Интернете или интрасети	nvarchar(max)
Изображение	Н/Д	Цифровые изображения	Двоичные цифровые изображения varbinary(max), 2^31-1 байт
Вычисляемый	Н/Д	Результат выражения, созданного с использованием данных из одного или нескольких полей таблицы	Зависит от результатов выражения
Мастер подстановок	Список значений	Использует содержимое списка значений для проверки содержимого поля	nvarchar(220)
Мастер подстановок	Другие таблица или запрос	Использует поле идентификатора другой таблицы или запроса для проверки содержимого поля	int

ПРИЛОЖЕНИЕ 2

Пользовательские форматы для полей даты / времени

Пользовательские форматы для полей даты/времени могут включать в себя две части — для даты и для времени. В качестве разделителя этих частей используется точка с запятой. Например, можно создать копию полного формата даты и длинного формата времени в следующем виде: «дд/м/гггг;ч:мм:сс».

Для получения форматированной строки используется функция Format (<выражение>, <шаблон форматирования>). Где <выражение> – дата или время, а шаблон форматирования составлен из знаков

В таблице приведены, перечислены подстановочные знаки и разделители, которые используются для пользовательских форматов.

Таблица 3

Форматирование дат и времени

Знак	Описание
Разделитель даты	Определяет расположение разделителя для дней, месяцев и лет в Access. Следует использовать разделитель, указанный в региональных стандартах Windows.
c	Отображение полного формата даты.
d или dd	Отображение дня в виде одной или двух цифр. Для одной цифры следует использовать один подстановочный знак, для двух цифр — два знака.
ddd	Сокращение дня недели. Например, понедельник будет отображаться как «пн».
dddd	Отображение полного названия дня недели.
ddddd	Отображение краткого формата даты.
dddddd	Отображение длинного формата даты.
w	Отображение номера дня недели. Например, понедельник будет отображаться как «2».
m или mm	Отображение месяца как однозначного или как двузначного числа.
mmm	Сокращение названия месяца до трех букв. Например, октябрь будет отображаться как «окт».
mmmm	Отображение полного названия месяца.
q	Отображение номера текущего квартала (1-4). Например, если сотрудник начал работать в мае, в Access отображается «2» в качестве значения квартала..
y	Отображение номера дня в году (1-366).
yy	Отображение двух последних цифр года.
yyyy	Отображение всех цифр года в диапазоне 0100-9999.

Окончание табл. 3

h или hh	Отображение часов в виде одной или двух цифр.
n или nn	Отображение минут в виде одной или двух цифр.
s или ss	Отображение секунд в виде одной или двух цифр.
tttt	Отображение длинного формата времени.
AM/PM	Отображение значений времени в 12-часовом формате с добавлением в конце сокращения AM или PM. При определении данного значения в Access используются системные часы компьютера.
A/P или a/p	Отображение значений времени в 12-часовом формате с добавлением в конце букв A, P, a или p. При определении данного значения в Access используются системные часы компьютера.
AMPM	Отображение времени в 12-часовом формате; однако при этом для обозначения времени суток используются региональные стандарты Windows.
Пробел, + - $ ()	При необходимости в строки формата можно вставлять пробелы, математические знаки (+, -) и финансовые символы ($, ¥, £). Если необходимо использовать другие математические знаки, например, косую черту (\ или /) и звездочку (*), следует заключить их в прямые кавычки, но при этом они могут располагаться в любом месте строки.
Текст	Текст, который должен отображаться для пользователей, следует заключить в прямые кавычки.
\	В Access отображается следующий знак. Такое же назначение имеют прямые кавычки.
*	При использовании звездочки знак, следующий за ней, считается заполняющим, т. е. знаком для заполнения пробелов. Как правило, в Access используется выравнивание текста по левому краю, а область справа от значения заполняется пробелами. Заполняющие знаки можно добавить в любое место строки; при этом Access заменяет пробелы указанным знаком.
[цвет]	Позволяет применить цвет ко всем значениям в одном из разделов формата. Имя цвета необходимо заключить в квадратные скобки; следует использовать следующие имена цветов: черный, синий, голубой, зеленый, розовый, красный, желтый и белый.

Учебное издание

Ольга Андреевна Ильичева
Павел Викторович Федотов

БАЗЫ ДАННЫХ.
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ И ЛАБОРАТОРНЫЙ ПРАКТИКУМ
В СУБД MS ACCESS

Учебное пособие

Редактор Н.Е. Гладких
Компьютерная верстка и макет О.И. Пушкиной
Темплан 2015 г., поз. 36.

Подписано в печать 7.09.2015. Формат 60х84/16. Бумага писчая.
Ризограф. Уч.- изд. 7.0. Тираж 100 экз. Заказ 183/15.

Редакционно-издательский отдел
Ростовского государственного строительного университета 344022,
Ростов-на-Дону, ул. Социалистическая,162