Учреждение образования
«Белорусский государственный университет культуры и искусств»

Факультет культурологии и социально-культурной деятельности

Кафедра информационных технологий в культуре

СОГЛАСОВАНО
Заведующий кафедрой
_____ Т.С. Жилинская
«22» 05 2025 г.

СОГЛАСОВАНО
Декан факультета
_____ Н.Е. Шелупенко
«30» мая 2025 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

«Технологии создания баз данных в управлении и коммуникациях»

для специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, профилизации Мультимедийные технологии и цифровые коммуникации

Составитель:
П.В. Гляков, профессор кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств», кандидат физико-математических наук, доцент

Рассмотрен и утвержден на заседании Совета факультета культурологии и социально-культурной деятельности 27 06 2025 г.,
протокол № 10

Минск 2025

Учебно-методический комплекс составлен в соответствии с примерным учебным планом по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, утвержденным Первым заместителем Министра образования Республики Беларусь от 30.01.2023 рег. № 6-05-03-013/пр. и учебного плана учреждения высшего образования по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, профилизации «Мультимедийные технологии и цифровые коммуникации» рег. № 6-05-03-26/23уч. от 15.02.2023.

Рецензенты:

В. В. Казаченок, заведующий кафедрой компьютерных технологий и систем Белорусского государственного университета, доктор педагогических наук, профессор;
В. А. Касап, профессор кафедры информационных ресурсов и коммуникаций учреждения образования «Белорусский государственный университет культуры и искусств», кандидат педагогических наук, доцент.

Рассмотрен и рекомендован к утверждению:

кафедрой информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств» (протокол № 10 от .06.2025);
Советом факультета культурологии и социокультурной деятельности (протокол от . 27.06 .2025 г., №10).

ОГЛАВЛЕНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Обучение эффективному применению современных информационных технологий при разработке дизайна и верстки текстовой и графической информации с учетом современных тенденций развития настольных издательских систем является важной задачей подготовки высококвалифицированных специалистов сферы культуры.

Учебная дисциплина «Технологии создания баз данных в управлении и коммуникациях» имеет межпредметную связь с такими учебными дисциплинами, как «Основы информационных технологий» и «Информационные технологии в культуре».

Целью обучения дисциплине «Технологии создания баз данных в управлении и коммуникациях» является формирование знаний и умений для проектирования и разработки баз данных, позволяющих автоматизировать процессы управления и коммуникаций в сфере культуры.

Задачи учебной дисциплины:

– ознакомление с современным состоянием проектирования, разработки и ведения баз данных;

– изучение основных способов проектирования и разработки баз данных с помощью системы управления базами данных реляционного типа;

– приобретение умений разрабатывать базы банных в сфере культуры.

В результате изучения дисциплины студенты должны *знать:*

– понятия информационные системы и автоматизированные информационные системы (АИС);

– понятия базы данных и системы управления базами данных (СУБД);

– иерархическую, сетевую и реляционную модели баз данных;

– нормальные формы отношений;

– жизненный цикл АИС;

– концептуальную модель базы данных;

– инфологическую модель базы данных;

– технологию создания базы данных;

– виды и способы создания запросов;

– способы представления данных в виде форм;

– способы разработки отчетов;

– алгоритмы импорта, экспорта и связывания данных;

– способ подготовки серийных писем;

– основные операции поддержки баз данных;

– средства защиты базы данных.

После изучения дисциплины студенты должны *уметь:*

– строить концептуальную и инфологическую модели базы данных;

– создавать запросы для получения информации из базы данных;
– представлять данные в виде форм и отчетов;
– проектировать и разрабатывать учебные базы данных;
– разрабатывать пользовательский интерфейс базы данных;
– выполнять импорт, экспорт и связывание данных;
– подготавливать серийные письма;
– выполнять операции по поддержке базы данных;
– использовать средства защиты базы данных.

Учебно-методический комплекс содержит конспекты лекций учебной дисциплины, описание лабораторных и практических работ, методические рекомендации для их проведения, перечень контрольных вопросов, задания для контролируемой самостоятельной работы студентов, вопросы к зачету, учебную программу дисциплины и список литературы.

Методы и технологии обучения

Учебный материал излагается на основе современных методических требований с учетом педагогических целей на уровнях представления, понимания, знания, применения и творчества. При чтении лекций особое внимание уделяется рассмотрению примеров, иллюстрирующих то или иное понятие, приводятся различные способы интерпретации понятий.

Лабораторные и практические занятия направлены на формирование умений практического использования полученных знаний при решении конкретных задач. Методика их проведения содействует развитию творческих способностей каждого студента и приобретению навыков самостоятельной работы.

Используются такие новые формы активизации учебного процесса, как игры, викторины, работа в командах с распределением ролей и т. п. Хорошо зарекомендовал себя при разработке баз данных метод проектов. Он поддерживает педагогические цели на уровнях представления, понимания, знания, применения и творчества.

В процессе изучения учебной дисциплины «Технологии создания баз данных в управлении и коммуникациях», согласно требованиям образовательного стандарта, студенты должны приобрести следующие компетенции:

– владеть основами исследовательской деятельности, осуществлять анализ и синтез информации;

– решать стандартные задачи профессиональной деятельности на основе информационно-коммуникационной технологии.

– владеть технологиями проектирования и разработки баз данных социокультурной сферы, использовать программное обеспечение для сопровождения и управления социокультурными проектами.

В рамках образовательного процесса по учебной дисциплине «Технологии создания баз данных в управлении и коммуникациях» студент должен не только приобрести теоретические и практические знания, умения и навыки по специальности, но и развивать свой ценностно-личностный, духовный потенциал, сформировать качества патриота и гражданина, готового к активному участию в экономической, производственной, социально-культурной жизни страны.

Основными формами учебной работы являются лекционные, лабораторные работы, практические занятия и самостоятельное изучение отдельных вопросов. Текущая форма контроля проводится в виде опроса. Промежуточная форма контроля – зачет.

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Учебные издания

1. Гляков, П. В. Система управления базами данных Access 2.0 : учеб. пособие / П. В. Гляков, С. Н. Карачун. – Минск : РИПО, 1998. – 100 с.

2. Гляков, П.В. Базы данных: компьютерный практикум : учеб. пособие / П.В. Гляков. – Минск : БГУКИ, 2008. – 130 с.

3. Михеева, Е. В. Информационные технологии в профессиональной деятельности : учеб. / Е. В. Михеева. – М. : Академия, 2020. – 416 с.

4. Основы построения баз данных : учеб. пособие : / Д. В. Чмыхов [и др.]. – Москва ; Берлин : Директ-Медиа, 2021. – 124 с. : – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=602227 (дата обращения: 11.04.2022). – Библиогр. в кн. – ISBN 978-5-4499-2428-5. – Текст : электронный.

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

2.2 Конспект лекций

Лекция 1

Основные понятия баз данных

Основные вопросы:

1. Уровни управления ресурсами.
2. Информационная система и информационные технологии.
3. Документальные и фактографические системы.
4. База данных, банк данных и система управления базами данных.
5. Система с базой данных.

Цель: изучение основных понятий баз данных.

Уровни управления ресурсами

В деятельности любого предприятия, учреждения, организации существенную роль играют данные. *Данные* фиксируются в определенной форме, пригодной для последующей обработки, хранения и передачи (на бумаге, магнитных дисках, магнитных лентах, компакт-дисках и т. п.).

Из данных извлекается необходимая информация. Данные в этом смысле можно рассматривать как сырье (ресурс) для производства информации. В результате обработки данные приобретают смысл, т. е. становятся информацией.

Под *информацией* понимают любые сведения о каком-либо событии, процессе, объекте, которые можно воспринимать, передавать, хранить или использовать.

На предприятии или в учреждении необходима информация для управления финансовыми, трудовыми и материальными ресурсами. Для *управления финансовыми ресурсами* надо иметь следующую информацию: источники денежных поступлений и их объемы, сколько денег израсходовано и на что, сколько средств предстоит получить и сколько осталось.

Управлять трудовыми ресурсами можно, когда известна информация о числе сотрудников, их специальности или профессии, должностном окладе, местонахождения рабочего места, прошлых достижениях сотрудников, сегодняшнем положении, возможности продвижения по службе.

Для управления материальными ресурсами требуется знать какие материалы есть в наличии, откуда они поступают, какое количество их требуется, сколько уже израсходовано, сроки поставки материалов и другую информацию.

Информация, предназначенная для управления учреждением, условно может быть разбита на три уровня. К нижнему уровню управления относится *оперативная информация*. Эта информация используется сотрудниками подразделений учреждения в повседневной работе. Она представляет собой часто обновляемую, первичную, рутинную информацию. Поэтому в информационных системах в первую очередь подлежит автоматизации обработка оперативной информации.

На среднем уровне управления имеют дело с *тактической информацией*. Эта информация предназначена для руководителей среднего звена. Тактическая информация получается путем обобщения оперативной информации и может быть представлена в виде отчетов или различных вариантов решения.

К верхнему уровню управления относят *стратегическую информацию*. Она получается в результате обработки оперативной и тактической информации. Эта информация содержит краткие, но содержательные сводки, отчеты, прогнозы. На ее основе осуществляется долгосрочное планирование работы учреждения.

Информационная система и информационные технологии

Каждое предприятие, учреждение, организацию можно рассматривать как *информационную систему*, состоящую из элементов, связей между ними, по которым циркулирует некоторая информация. Информационная система функционирует на базе некоторой информационной технологии. В понятие *информационной технологии* входят устройства, носители информации, методы хранения, переработки и обмена информацией.

Задачей разработчиков *автоматизированных информационных систем* (АИС) является включение в существующие информационные системы со своей информационной технологией элементов автоматизации на всех уровнях и создание на базе персональных компьютеров и вычислительных сетей новой информационной технологии.

Автоматизированные информационные системы, основу которых составляют базы данных, появились в 60-х годах XX века в военной промышленности и бизнесе – там, где были накоплены значительные объемы полезных данных. Первоначально АИС были ориентированы лишь на работу с информацией фактического характера – числовыми или текстовыми характеристиками объектов. Затем по мере развития техники появилась возможность обработки текстовой информации на естественном языке.

В автоматических и автоматизированных информационных системах обеспечивается выполнение следующих основных функций:

– надежное хранение больших объемов информации в памяти компьютера;

– выполнение специфической для данной области обработки и передачи данных;

– предоставление пользователям удобного интерфейса. В литературе выделяют ряд этапов в развитии информационных систем.

Вначале данные обрабатывались с использованием оборудования с перфокартами и электромеханических машин для сортировки и табулирования миллионов записей. На следующем этапе данные хранились на магнитных лентах, и программы выполняли пакетную обработку последовательных файлов. Затем было введено понятие базы данных и обеспечен автоматический доступ к базам, были внедрены распределенная и клиент-серверная обработка данных.

В настоящее время используются информационные системы, которые хранят сложные структуры мультимедийных данных документы, графику, аудио- и видеоданные. Эти системы представляют собой базовые средства хранения для приложений Интернет и интранет.

Мультимедийные базы данных (БД) и средства доступа к ним будут краеугольным камнем в нашем движении к киберпространству. Значительные расширения в применении баз данных требуют решения сложных проблем, которые можно сгруппировать в следующие задачи:

– поддержка мультимедийных объектов;

– распределенное хранение информации;

– новые виды приложений баз данных;

– управление транзакциями и потоками работ;

– эффективность управления базами данных и их использования.

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

Базы данных используются в различных областях и сферах человеческой деятельности. В качестве примеров можно привести БД, содержащие информацию о клиентах, товарах, предоставляемых услугах, коммерческих операциях и многое другое. БД являются основой любой информационной системы и в широком смысле слова – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области.

БД представляет собой хранилище огромных массивов информации в связанном виде для совместного использования. Однако сама по себе БД не может обслужить запросы пользователей на поиск и обработку информации,

так как это просто "склад информации". Чтобы эффективно использовать подобный склад, необходима система управления базой данных.

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

В определениях БД наиболее часто присутствуют следующие характерные признаки:

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

2. Данные в БД логически связаны и структурированы с целью обеспечения возможности их эффективного поиска и обработки в системе. Структурированность подразумевает явное выделение составных элементов и связей между ними, типизацию элементов и связей, для которых имеет место определённая семантика и допустимые операции;

3. БД включает схему описания логической структуры БД в формальном виде. БД является основой для построения информационных систем, под которыми понимают совокупность базы данных и комплекса аппаратно-программных средств для хранения, модификации и поиска информации, взаимодействия пользователя с системой. Для управления базами данных необходима система управления, представляющая собой совокупность программных и языковых средств, предназначенных для создания, ведения и совместного использования базы данных коллективом пользователей. Эффективное управление системой памяти является основной функцией системы управления базами данных (СУБД).

Эти обычно специализированные средства настолько важны с точки зрения эффективности, что при их отсутствии система просто не сможет выполнять некоторые задачи уже потому, что их выполнение будет занимать слишком много времени. При этом ни одна из таких специализированных функций не является видимой для пользователя. Они обеспечивают независимость между логическим и физическим уровнями системы. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

История развития систем управления базами данных насчитывает десятки лет. Первая промышленная СУБД фирмы IBM была введена в эксплуатацию в 1968 году, а в 1975 году появился первый стандарт, который определил ряд основных понятий в теории систем баз данных. Развитие вычислительной техники, появление персональных компьютеров, мощных рабочих станций и компьютерных сетей обусловило развитие технологии баз данных.

С появлением локальных сетей возникла задача согласования данных, хранящихся и обрабатываемых в разных местах, но связанных логически. Решение этой задачи привело к появлению распределенных баз данных, позволяющих организовать параллельную обработку информации и сохранять целостность баз данных.

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

Документальные и фактографические системы

Принципы хранения разных видов информации в АИС аналогичны, но алгоритмы ее обработки определяются характером информационных ресурсов. Соответственно различают два класса АИС: документальные и фактографические.

Документальные АИС служат для работы с документами на естественном языке. Наиболее распространенный тип документальных АИС – информационно-поисковые системы, предназначенные для накопления и подбора документов, удовлетворяющих заданным критериям. Эти системы могут выполнять просмотр и подборку монографий, публикаций в периодике, сообщений пресс-агентств, текстов законодательных актов и т. д.

Фактографические АИС оперируют фактическими сведениями, представленными в формализованном виде, и используются для решения задач обработки данных.

Обработка данных – специальный класс решаемых на ЭВМ задач, связанных с вводом, хранением, сортировкой, отбором и группировкой записей данных однородной структуры. К задачам этого класса относятся: учет товаров в магазинах и на складах; начисление зарплаты; управление производством, финансами, телекоммуникациями и т. п.

Различают фактографические АИС оперативной обработки данных, подразумевающие быстрое обслуживание относительно простых запросов от

большого числа пользователей, и фактографические АИС аналитической обработки, ориентированные на выполнение сложных запросов, требующих проведения статистической обработки исторических (накопленных за некоторый промежуток времени) данных, моделирования процессов предметной области и прогнозирования развития этих процессов.

Таким образом, АИС применяются в следующих областях:

– организация хранилищ данных;

– системы анализа данных;

– системы принятия решений;

– мобильные и персональные базы данных;

– географические базы данных;

– мультимедиа базы данных;

– распределенные информационные системы;

– базы данных для всемирной сети World Wide Web.

База данных, банк данных и система управления базами данных

База данных – организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

Понятие базы данных можно применить к любой связанной между собой по определенному признаку информации, хранимой и организованной особым образом – как правило, в виде таблиц. По сути, база данных – это некоторое подобие электронной картотеки, электронного хранилища данных, которое хранится в компьютере в виде одного или нескольких файлов. При этом возникает необходимость в выполнении ряда операций с базой данных, в первую очередь это:

– добавление новой информации в существующие файлы базы данных;

– добавление новых пустых файлов в базы данных:

– изменение (модификация) информации в существующих файлах базы данных:

– поиск информации в базе данных;

– удаление информации из существующих файлов базы данных;

– удаление файлов из базы данных.

Процедуры хранения данных в базе должны подчиняться некоторым общим принципам:

– целостность и непротиворечивость данных, под которыми понимается

как физическая сохранность данных, так и предотвращение неверного использования данных,

– поддержка допустимых сочетаний их значений,

– защита от структурных искажений и несанкционированного доступа;

– минимальная избыточность данных обозначает, что любой элемент данных должен храниться в базе в единственном виде, что позволяет избежать необходимости дублирования операций, производимых с ним.

Современные компьютеры могут хранить самую разнообразную информацию: записи, документы, графику, звуко- и видеозаписи, научные и другие данные в разнообразных форматах. Совокупность сведений о каких-либо объектах, процессах, событиях или явлениях, организованная таким образом, чтобы можно было легко представить любую часть этой совокупности, называют *базой данных*, а совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования базы данных многими пользователями, называют *системой управления базами данных* (СУБД).

Система с базой данных

Система с базой данных состоит из следующих компонентов (рис. 1):

– пользователей, т. е. людей, которые используют данные;

– приложения, т. е. программ пользователей, которым требуются данные из системы;

– СУБД – программного обеспечения, которое управляет доступом к данным и обеспечивает указанные функциональные возможности системы с базой данных;

– данных, т. е. строк, хранящихся в файлах;

– системы-хоста – компьютерной системы, в которой хранятся файлы.

Доступ к строкам данных осуществляется системой-хостом. Роль СУБД состоит в том, чтобы генерировать запросы, позволяющие использовать функциональные возможности системы управления файлами системы-хоста для обслуживания различных приложений. СУБД – это дополнительный уровень программного обеспечения, надстроенный над программным обеспечением системы-хоста.

Таким образом, систему с базой данных можно представить в виде последовательности уровней, представленных на рисунке 1.

Рис. 1. Уровни системы с базой данных

На самом нижнем уровне находятся данные, хранящиеся в физических файлах (физическая память базы данных). На верхнем уровне – приложения с их собственными представлениями одних и тех же физических данных. Каждое представление базы данных – это определенная логическая структура, построенная из лежащих в основе физических данных.

Банк данных – это более широкое понятие, чем база данных. В состав банка данных входят база данных, система управления базами данных, а также вычислительная и коммуникационная техника с обслуживающим персоналом.

Заметим, что является грубой, хотя и весьма распространенной, ошибкой использовать термины "банк данных" и "база данных" как синонимичные.

Лекция 2

Классификация и функции СУБД

Основные вопросы:

1. Классификация баз данных.
2. Основные функции СУБД.
3. Языки баз данных.
4. Функциональные возможности СУБД.

Цель: изучение классификации и функций СУБД.

Классификация баз данных

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

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

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

Распределённая БД состоит из нескольких частей, хранимых в различных компьютерах сети, и работа такой БД происходит с помощью СУБД. По способу доступа к данным БД подразделяются на БД с локальным и удаленным доступом. БД называют с локальным доступом, если компьютер является компонентом сети и к такой БД возможен распределенный доступ. Для распределенного хранения данных и доступа к базе компьютеры объединяют в локальные, региональные и даже глобальные сети. Для

построения подобных сетей также используется технология клиент-сервер. Система клиент-сервер – это обычная локальная вычислительная сеть, которая содержит группу компьютеров-клиентов и один специальный компьютер – сервер. Компьютеры-клиенты обращаются к серверу за различными услугами. Компьютер-сервер может пересылать им различные программы, например обработку текстов, работы с таблицами, выполнение запросов к базе данных и возвращать результаты. Основная идея состоит в том, что каждый компьютер выполняет то, что он делает наиболее эффективно. Сервер извлекает и обновляет данные, клиент выполняет специальные расчеты и предоставляет результаты конечному пользователю.

Вначале серверы выполняли простейшие функции: серверы печати, файловые серверы, по запросу клиента на доступ к какому-нибудь файлу сервер пересылал данный файл компьютеру-клиенту.

Сервер базы данных – это программа, которая запускается на компьютере-сервере и обслуживает доступ клиентов к базе данных. Таким образом, в основе системы клиент-сервер лежит принцип разделения труда. Клиент – это компьютер, с которым работает пользователь, а компьютер-сервер выполняет обслуживание группы клиентов: доступ к базе данных, обновление базы данных и т. д. Прогрессивным путем коллективного доступа к БД в последние 20 лет является использование всемирной сети Интернет с группой ее служб.

Примерами серверов могут служить:

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

– вычислительный сервер, дающий возможность производить вычисления, которые невозможно выполнить на рабочих станциях;

– дисковый сервер, обладающий расширенными ресурсами внешней памяти и предоставляющий их в использование компьютерам-клиентам и, возможно, другим серверам;

– файловый сервер, поддерживающий общее хранение файлов для всех рабочих станций;

– сервер баз данных – фактически обычная СУБД, принимающая и обслуживающая запросы по локальной сети. Хотя обычно одна база данных целиком хранится в одном узле сети и поддерживается одним сервером, серверы баз данных представляют собой простое и дешевое приближение к распределенным базам данных, поскольку общая база данных доступна для всех пользователей локальной сети.

Одним из перспективных направлений СУБД является гибкое конфигурирование системы, при котором распределение функций между клиентской и пользовательской частями СУБД определяется при установке

системы. СУБД должны обеспечивать логическую целостность данных, подразумевающую поддержание непротиворечивой и полной информации, адекватно отражающей предметную область. С требованием логической целостности данных связано понятие транзакции.

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

Основные функции СУБД

СУБД представляют собой оболочку, с помощью которой после организации структуры таблиц и заполнения их данными формируется определенная БД. Программные средства включают систему управления вводом-выводом, хранение и обработку информации, создание и тестирование БД, трансляторы.

В качестве языков программирования могут использоваться объектно-ориентированные языки C++, Java или язык структурированных запросов SQL (Structured Query Language). Появление данного языка обусловлено ростом количества данных, необходимостью их хранения и обработки. С помощью SQL пользователи могут манипулировать данными независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции или универсальном компьютере. Данный язык в настоящее время получил очень широкое распространение и фактически превратился в стандартный язык реляционных баз данных. Стандарт на язык SQL был разработан Американским национальным институтом стандартов (ANSI)в 1986 г., а в 1987 г. Международная организация стандартов ISO приняла его в качестве международного.

Язык запросов выполняет поиск совокупностей параметров в файле в соответствии с заданным набором критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД.

К основным функциям СУБД можно отнести следующие:

– Управление данными во внешней памяти. Данная функция обеспечивает поддержку требуемых структур внешней памяти как для хранения данных в БД, так и для выполнения служебных функций.

– Управление буферной памятью. Размер БД обычно значительно превосходит емкость оперативной памяти, и при обращении к данным система будет работать со скоростью внешней памяти. Одним из способов повышения скорости является буферизация данных в оперативной памяти. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с индивидуальным протоколом управления буферной памятью.

– Управление транзакциями. Применение СУБД для работы с интегрированными БД выявило особую важность проблемы ее целостности, под которой понимают правильность и непротиворечивость содержимого БД. Нарушение целостности может быть вызвано, например, ошибками или сбоями в системе, так как в этом случае она не в состоянии обеспечить нормальную обработку или выдачу правильных данных.

Аспект целостности обеспечивается на уровне структур данных и отдельных операторов языковых средств СУБД и при нарушении такой целостности оператор отвергается. Для обеспечения целостности БД и служит аппарат транзакций. Таким образом, под транзакцией понимают неделимую с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации). При этом для поддержания ограничений целостности на уровне БД допускается их нарушение внутри транзакции так, чтобы к моменту завершения транзакции условия целостности были соблюдены.

– Журнализация. Одним из основных требований к развитым СУБД является надежность хранения баз данных. Это требование предполагает, в частности, возможность восстановления согласованного состояния БД после любого рода аппаратных и программных сбоев. Очевидно, что для выполнения восстановлений необходима некоторая дополнительная информация. В подавляющем большинстве современных реляционных СУБД такая избыточная дополнительная информация поддерживается в виде журнала изменений базы данных. Итак, общей целью журнализации изменений баз данных является обеспечение возможности восстановления согласованного состояния БД после любого сбоя. Поскольку основой поддержания целостного состояния базы данных является механизм транзакций, журнализация и восстановление тесно связаны. Общим принципом восстановления является сохранение результатов зафиксированных транзакций в восстановленном состоянии БД.

Примерами ситуаций, при которых требуется производить восстановление состояния БД, являются следующие. Восстановление после внезапной потери содержимого оперативной памяти – мягкий сбой. Такая ситуация может возникнуть при аварийном выключении электропитания, при возникновении неустранимого сбоя процессора и т. д. Ситуация характеризуется потерей той части БД, которая к моменту сбоя содержалась в буферах оперативной памяти.

Восстановление после отказа основного внешнего носителя базы данных – жесткий сбой. Эта ситуация при достаточно высокой надежности современных устройств внешней памяти может возникать сравнительно редко, но тем не менее, СУБД должна быть в состоянии восстановить БД даже и в этом случае. Основой восстановления является архивная копия и журнал изменений БД. Во всех случаях основой восстановления является избыточное хранение данных, которые хранятся в журнале, содержащем последовательность записей об изменении базы данных.

Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, иногда поддерживаются две копии журнала, располагаемые на разных физических дисках. В журнал поступают записи обо всех изменениях в БД. В разных СУБД изменения БД отображаются в журнале на разных уровнях. Иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда – минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

Языки баз данных

Для работы с БД используются специальные языки, называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка: язык определения схемы БД и язык манипулирования данными. В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания и завершая разработкой пользовательского интерфейса с БД.

Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык запросов SQL. Язык SQL содержит специальные средства определения ограничений целостности БД. Ограничения целостности хранятся в специальных таблицах-каталогах, и обеспечение контроля целостности БД производится на языковом уровне. При компиляции операторов модификации БД компилятор SQL на

основании имеющихся в БД ограничений целостности генерирует соответствующий программный код.

Операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми в БД запросами (результатом любого запроса к реляционной БД является таблица) с именованными столбцами. Авторизация доступа к объектам БД производится также на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL разного вида пользователь должен обладать различными полномочиями.

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

Функциональные возможности СУБД

По степени универсальности различают СУБД двух видов:

– СУБД общего назначения, реализованные как программный продукт, способный функционировать на компьютере с определённой операционной системой и поставляемый пользователям в качестве законченного коммерческого изделия;

– специализированные СУБД, создаваемые в случаях невозможности или нецелесообразности разработки СУБД общего назначения. СУБД общего назначения представляют собой сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией БД информационной системы. Рынок программного обеспечения персональных компьютеров располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем СУБД общего назначения.

Производительность СУБД оценивается следующими параметрами:

– временем выполнения запросов;

– временем выполнения операций импортирования данных, имеющих отличные форматы;

– скоростью выполнения таких операций как обновление, вставка, удаление данных и других;

– степенью параллелизма при обращении к данным в многопользовательском режиме;

– временем генерации отчёта. СУБД, поддерживающие целостность данных, предполагают наличие средств, позволяющих определять, что информация в БД всегда остаётся корректной и полной.

Примерами операций, обеспечивающих безопасность СУБД, являются шифрование прикладных программ и данных, защита паролем, ограничение доступа к БД и другие. Например, хороший уровень безопасности имеют СУБД Oracle, Access.

Целостность должна обеспечиваться независимо от того, каким образом данные заносят в память, от действий пользователей, сбоев сети и т. д. Может предусматриваться назначение отдельных паролей для пользователей, присвоение различных прав доступа для таблиц, запросов, отчётов и т. д.

Лекция 3
Проектирование и создание базы данных Microsoft Access

Основные вопросы:

1. Логические модели баз данных.
2. Иерархическая модель данных.
3. Сетевая модель данных.
4. Объектно-ориентированная модель данных.
5. Реляционная модель данных.
6. Нормализация отношений в реляционной модели.
7. Этапы проектирования баз данных.

Цель: изучение вопросов проектирования и создания базы данных Microsoft Access.

Логические модели баз данных

Прежде чем создавать базу данных необходимо выбрать наиболее эффективную модель данных для решения поставленной задачи. Модель данных – это совокупность структур данных и операций их обработки. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

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

– выбор методов описания типов и логических структур данных в БД;

– определение способов обработки данных в БД;

– обеспечение метода поддержки целостности БД.

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

Модель данных – это абстрактное логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную структуру доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы – поведение данных.

Каждая БД и СУБД строится на основе некоторой явной модели данных и построенные на одной и той же модели данных, относятся к одному типу. Длительное время термин «модель данных» не имел

формального определения. Одним из первых специалистов, который ввел достаточно формальное понятие в статье «Модели данных в управлении базами данных», был Э. Кодд. Он определил модель данных как комбинацию трех компонентов:

– множество типов объектов данных, образующих базовые строительные блоки для любой базы данных соответствующей модели;

– набор общих правил целостности, ограничивающих набор экземпляров тех типов объектов, которые законным образом могут появиться в любой такой базе данных;

– множество операций, применимых к таким типам объектов для выборки и других функций.

Модель данных представляет сочетание следующих компонентов: структурную часть, т. е. набор правил, по которым может быть построена база данных, и управляющую часть, определяющую типы дополнительных операций над данными (обновление, изменение связей, удаление).

Выбор модели данных возлагается на разработчика и зависит от технического и программного обеспечения, определяется сложностью задач и объемом информации. Организация способов хранения данных в компьютерных системах задается на логическом и на физическом уровнях.

Физический уровень определяет способ размещения данных непосредственно на машинном носителе. Этот уровень обеспечивается автоматически прикладными программами без вмешательства пользователя. Пользователь в прикладных программах оперирует представлениями логической организации данных.

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

Различают следующие основные четыре модели данных:

– иерархическая;

– сетевая;

– объектно-ориентированная;

– реляционная.

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

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

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

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

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

Рассмотрим более подробно указанные модели данных.

Иерархическая модель данных

Первые иерархические модели данных появились в конце 50-х годов. Они представляли собой древовидную структуру, где данные были распределены по уровням от главного – к подчиненному и представляли собой неориентированный граф. Пример иерархической модели данных приведен на рисунке 2.

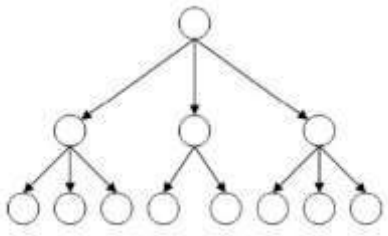


Рис. 2. Иерархическая модель данных

Данная модель характеризуется количеством уровней и узлов. Каждый уровень представляет собой один или несколько объектов (данных) и может иметь несколько узлов подчиненных уровней, причем связи между всеми

объектами жестко закреплены, и один потомок может иметь не более одного предка. Основные типы структур данных рассматриваемой модели – поле, запись, файл. Запись является основной структурной единицей обработки данных и единицей обмена между оперативной и внешней памятью. В модели на основе записей база данных состоит из записей фиксированного формата, которые могут быть разного типа. Каждый тип записи определяет фиксированное количество полей, каждое из которых имеет фиксированную длину. Организация данных в СУБД иерархического типа определяется в следующих терминах:

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

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

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

Групповое отношение – иерархическое отношение между записями двух типов.

Родительская запись называется исходной записью, а дочерние записи – подчиненными.

Иерархическая база данных может хранить только такие древовидные структуры. Корневая запись каждого дерева обязательно должна содержать ключ с уникальным значением. Ключи некорневых записей должны иметь уникальное значение только в рамках группового отношения. Каждая запись идентифицируется полным ключом, под которым понимается совокупность ключей всех записей от корневой по иерархическому пути. При графическом изображении групповые отношения изображают дугами ориентированного графа, а типы записей – вершинами (диаграмма Бахмана).

Для групповых отношений в иерархической модели обеспечивается автоматический режим включения записи. Это означает, что для запоминания любой некорневой записи в БД должна существовать ее родительская запись. Примером иерархической базы данных является Каталог папок Windows, с которым можно работать, запустив Проводник. Верхний уровень занимает папка Рабочий стол. На втором уровне находятся папки Мой компьютер, Мои документы, Сетевое окружение и Корзина, которые являются потомками папки Рабочий стол, а между собой является близнецами. В свою очередь, папка Мой компьютер является предком по отношению к папкам третьего уровня – папкам дисков.

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

Сетевая модель данных

На разработку данной модели большое влияние оказал американский ученый Ч. Бахман. Основные принципы сетевой модели данных были разработаны в середине 60-х годов. Под сетевой моделью понимается модель данных, подобная иерархической, но допускающая свободную систему связей между узлами различных уровней. Таким образом, она является расширением иерархической модели данных (рисунок 3).

В отличие от иерархической модели у потомка сетевой модели может быть более одного предка и один объект может быть одновременно главным и подчиненным. Таким образом, в данной модели отношения между данными такие, что каждая запись может быть подчинена записям более, чем из одного файла. В сетевых моделях можно по ключу иметь непосредственный доступ к любому объекту независимо от уровня, на котором он находится в модели.

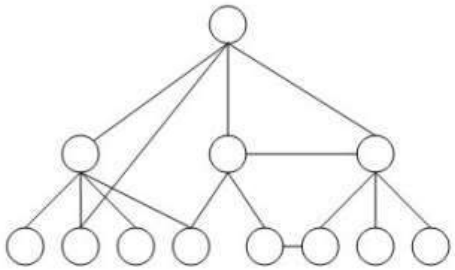


Рис. 3. Сетевая модель данных

К достоинству сетевой модели можно отнести эффективность реализации по степени затрат памяти и быстроты доступа. Недостатком является повышенная сложность схемы данных, построенной на её основе. Типичным представителем систем, основанных на сетевой модели данных, является СУБД IDMS (Integrated Database Management System), разработанная компанией Cullinet SoftwareInc. и изначально ориентированная на использование мэйнфреймов (компьютеров общего назначения) компании IBM. Архитектура системы основана на предложениях Data Base Task Group (DBTG) организации CODASYL (Conferenceon Data Systems Languages), которая отвечала за определение языка программирования COBOL. Отчет DBTG был опубликован в 1971 г., и вскоре после этого появилось несколько систем, поддерживающих архитектуру CODASYL, среди которых 14

присутствовала и СУБД IDMS. В настоящее время IDMS принадлежит компании Computer Associates.

Объектно-ориентированная модель данных

В настоящее время общепринятого определения объектно-ориентированной модели данных не существует и поэтому можно только говорить лишь о некотором "объектном" подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации. Объектно-ориентированная модель позволяет создавать, хранить и использовать информацию в форме объектов (рисунок 4).

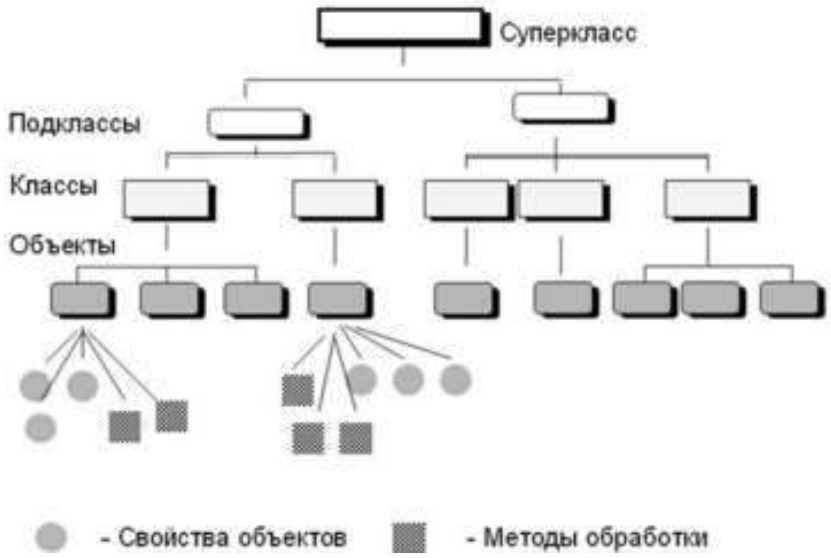


Рис. 4. Объектно-ориентированная модель данных

Объект при своем создании получает генерируемый системой уникальный идентификатор, который связан с объектом все время его существования и не меняется при изменении состояния объекта. Каждый объект имеет состояние и поведение. Состояние объекта – набор значений его атрибутов, а поведение объекта – это набор методов (программный код), оперирующих над состоянием объекта. Значение атрибута объекта – это тоже некоторый объект или множество объектов. Состояние и поведение объекта инкапсулированы в объекте. Взаимодействие объектов производится на основе передачи сообщений и выполнении соответствующих методов. Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов.

Объект должен принадлежать только одному классу, если не учитывать возможности наследования. Допускается наличие примитивных предопределенных классов, объекты которых не имеют атрибутов: целые, строки и т. д. Класс, объекты которого могут служить значениями атрибута

объектов другого класса, называется доменом этого атрибута. Допускается порождение нового класса на основе существующего класса – наследование. В этом случае новый класс, называемый подклассом существующего класса, наследует все атрибуты и методы суперкласса. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы.

Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько. Если в языке или системе поддерживается единичное наследование классов, набор классов образует древовидную иерархию. При поддержании множественного наследования классы связаны в ориентированный граф с корнем, называемый решеткой классов. Объект подкласса считается принадлежащим любому суперклассу этого класса.

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

Реляционная модель данных

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

Предложения Кодда были настолько эффективны для баз данных, что за эту модель он был удостоен престижной премии Тьюринга в области теоретических основ вычислительной техники. В реляционных базах все данные хранятся в простых таблицах, разбитых на строки (их называют записями) и столбцы (их называют полями), на пересечении которых расположена информация о данных. В общем виде это может быть представлено на рис. 4.

В основе *реляционной модели данных* лежит понятие отношения (англ. relation). Отношение удобно представляется в виде двумерной таблицы при соблюдении определенных ограничивающих условий. База данных при этом являет собой совокупность таблиц. Таблица привычна для пользователя, понятна и обозрима, ее легко запомнить.

Основоположник теории реляционных баз данных Е. Ф. Кодд показал, что набор отношений (таблиц) может быть использован для хранения данных об объектах реального мира и моделирования связей между ними.

Например, для хранения сущности ‘студент’ используют отношение СТУДЕНТ, в котором свойства сущности располагаются в столбцах таблицы, как показано на рисунке 5.

ФИО	Факультет	Курс	Группа
Сидорова А. И.	ФКиСКД	3	308
Василевский В. И.	ФКиСКД	2	212
Дубова В. А.	ТБКиСИ	3	314
Алексеенко С. А.	ФХМИ	1	116
Королева В .С.	ТБКиСИ	2	213

Рис. 5. Отношение СТУДЕНТ

Столбцы отношения называют атрибутами, атрибутам присваивают имена. Список имен атрибутов отношения называют схемой отношения. Список значений атрибутов отношения называют картежом. Схема отношения СТУДЕНТ записывается так:

СТУДЕНТ (ФИО, Факультет, Курс, Группа).

Реляционная БД – это набор взаимосвязанных таблиц. Набор взаимосвязанных таблиц на физическом уровне (на внешних носителях информации) хранится в виде файла БД. Соответствие между элементами файла БД, таблицы, отношения и сущности, когда файл БД содержит одну таблицу, представлено на рисунке 6.

Файл БД	Таблица	Отношение	Сущность
запись	строка	кортеж	экземпляр сущности
поле	столбец	атрибут	атрибут

Рис. 6. Соответствие между основными понятиями реляционной модели

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

различные поля отличаются именами. Каждая запись может идентифицироваться уникальным ключом записи, которые бывают двух типов: первичный и вторичный. Первичный ключ – это одно или несколько полей, однозначно идентифицирующих запись. Если первичный ключ состоит из одного поля, он называется простым, если из нескольких полей – составным ключом. Вторичный ключ – это поле, значение которого может повторяться в нескольких записях файла, т. е. он не является уникальным. Внешний ключ подчиненной таблицы – это вторичный ключ данного отношения, который, в то же время, выполняет функции первичного ключа в главной таблице. Если по значению первичного ключа может быть найден один единственный экземпляр записи, то по значению внешнего ключа несколько.

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

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

– операторы определения данных;

– операторы манипуляции данными;

– операторы определения доступа к данным.

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

Достоинствами реляционной модели являются:

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

– при проектировании реляционной БД применяются строгие правила, базирующиеся на математическом аппарате;

– полная независимость данных, при изменении структуры таблицы в прикладных программах модификации минимальны;

– для построения запросов и написания прикладных программ нет необходимости знания конкретной организации БД во внешней памяти.

Недостатками реляционной модели являются:

– относительно низкая скорость доступа и большой объем внешней памяти;

– трудность понимания структуры данных из-за появления большого количества таблиц в результате логического проектирования;

– не всегда предметную область можно представить в виде совокупности таблиц.

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

Нормализация отношений в реляционной модели

В зависимости от содержания отношений реляционных БД мы будем различать два типа отношений: объектные и связные. Объектное отношение хранит данные об объектах. Рассмотренное выше отношение СТУДЕНТ является примером объектного отношения.

В объектном отношении один из атрибутов однозначно идентифицирует объект. Такой атрибут называют ключом отношения. В отношении СТУДЕНТ ключом может быть атрибут Фамилия. Для удобства ключ размещают в первом столбце таблицы. Ключ может состоять из нескольких атрибутов или быть частью одного атрибута.

Основное ограничение реляционной модели БД состоит в следующем. В объектном отношении не должно быть строк с одинаковыми ключами. Это ограничение позволяет обеспечить целостность БД.

Связное отношение хранит ключи двух или более объектных отношений, по этим ключам устанавливаются связи между объектами. Пусть в БД имеются два отношения: ФИРМА (Название, Адрес) и ТОВАР (Наименование, Цена). Из них может быть получено связное отношение ПОСТАВЛЯЕТ (Фирма, Товар).

Связное отношение кроме связываемых ключей может иметь и другие атрибуты, которые будут функционально зависеть от этой связи. Например,

отношение ПОСТАВЛЯЕТ может иметь следующий вид: ПОСТАВЛЯЕТ(Фирма, Товар, Цена). Ключи в связных отношениях называются внешними ключами, поскольку они являются первичными ключами других отношений.

Каждому внешнему ключу должна соответствовать строка какого-либо объектного отношения. Невыполнение этого ограничения может привести к нарушению ссылочной целостности данных. Ключ должен ссылаться на объект, который существует.

Отношения в БД должны быть нормализованы. Это означает, что каждый атрибут должен быть простым – содержать неделимые значения. В приведенном на рисунке 7 отношении ПРЕПОДАВАТЕЛЬ условие нормализации не выполнено.

Фамилия	Кафедра	Предмет	Телефон	
			рабочий	домашний
Голубев	Информатика	Гипертекстовые системы	20-08-35	70-01-32
Погорелова	Психология	Практическая психология	20-08-35	50-85-30
Михайлов	Методика воспитания	Методика воспитания	20-15-91	53-33-45
Новожилов	Информатика	Информационные технологии	20-08-35	50-85-30
Антипов	Педагогика	Педагогика	20-13-04	55-93-14

Рис. 7. Отношение ПРЕПОДАВАТЕЛЬ

Это отношение можно нормализовать разбиением сложного атрибута Телефон на два простых: Телефон рабочий и Телефон домашний. Схема нормализованного отношения выглядит так:

ПРЕПОДАВАТЕЛЬ (Фамилия, Кафедра, Предмет, Телефон рабочий, Телефон домашний).

Отношение, у которого все атрибуты простые, называется приведенным к первой нормальной форме (1НФ).

Перечислим условия и ограничения, накладываемые на отношения реляционной моделью данных, которые позволяют таблицы считать отношениями:

1. Не может быть одинаковых первичных ключей, т. е. все строки таблицы должны быть уникальны.

2. Все строки таблицы должны иметь одну и ту же структуру, т. е. одно и то же количество атрибутов с соответственно совпадающими именами.

3. Имена столбцов таблицы должны быть различны, а значения столбцов должны быть однотипными.

4. Значения атрибутов должны быть простыми, следовательно, отношения не могут иметь в качестве компонент другие отношения.

5. Должна соблюдаться ссылочная целостность для внешних ключей.

6. Порядок следования строк в таблице несущественен - он лишь влияет на скорость доступа к строке.

Задавая отношения над элементами данных, проектировщик БД определяет, какие из атрибутов объекта являются зависимыми. Термин функциональная зависимость означает следующее. Пусть имеется два атрибута: A и B. Если в любой момент времени каждому значению A соответствует не более чем одно значение атрибута B, говорят, что B функционально зависит от A. Функциональная зависимость обозначается так:

$$A \rightarrow B.$$

Рассмотрим следующее отношение: СЛУЖАЩИЙ (Номер служащего, Имя служащего, Зарплата, Номер проекта, Дата окончания). Функциональные зависимости между атрибутами отношения СЛУЖАЩИЙ для наглядности представим на рисунке 8 в виде диаграммы.

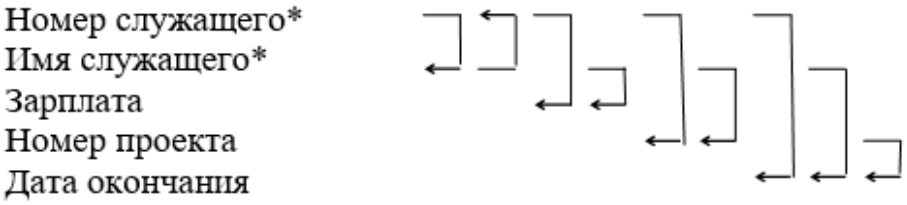


Рис. 8. Функциональные зависимости атрибутами

В нашем примере звездами отмечены основные атрибуты. Основные атрибуты являются элементами возможных ключей отношения. Атрибут Имя служащего функционально зависит от атрибута Номер служащего и, в свою очередь, атрибут Номер служащего функционально зависит от атрибута Имя служащего. Атрибут Зарплата функционально зависит от атрибута Номер служащего или Имя служащего. Аналогично атрибут Номер проекта функционально зависит от атрибута Номер служащего или Имя служащего. Атрибут Дата окончания обладает функциональной зависимостью от атрибутов: Номер служащего, Имя служащего и Номер проекта.

Атрибут может функционально зависеть не от одного какого-то атрибута, а от целой группы атрибутов. Рассмотрим, например, отношение, содержащее информацию о расходе рабочего времени программиста:

РАБОТА ПРОГРАММИСТА (Номер программиста, Номер программы, Имя программиста, Имя программы, Количество часов).

Атрибут Количество часов функционально зависит от составного ключа (Номер программиста, Номер программы) или от одного из следующих возможных ключей: (Номер программиста, Имя программы), (Имя программиста, Номер программы) или (Имя программиста, Имя программы). Заранее предполагается, что среди программистов нет однофамильцев, и что две программы не могут иметь одинаковых имен.

Атрибут (или набор атрибутов) B из отношения R называется полностью зависимым от другого набора атрибутов A отношения R, если B функционально зависит от всего множества A, но не зависит ни от какого подмножества A.

Например, в отношении РАБОТА ПРОГРАММИСТА атрибут Количество часов является полностью зависимым от составного ключа (Номер программиста, Номер программы), так как он задает количество рабочего времени, затраченного данным программистом на конкретную программу. При этом ни один из атрибутов Номер программиста или Номер программы в отдельности не определяет значение атрибута Количество часов.

Говорят, что отношение задано во второй нормальной форме (2НФ), если оно является отношением в первой нормальной форме, и каждый атрибут, не являющийся основным атрибутом в этом отношении, полностью зависим от любого возможного ключа этого отношения.

Если все возможные ключи отношения содержат по одному атрибуту, как это было в ранее рассмотренном отношении СЛУЖАЩИЙ, то это отношение задано во второй нормальной форме. Если ключи состоят более чем из одного атрибута, отношение, заданное в 1НФ, может не быть отношением в 2НФ. Отношение РАБОТА ПРОГРАММИСТА задано в 2НФ, потому что единственный атрибут Количество часов, не являющийся основным, полностью зависит от каждого возможного ключа.

Рассмотрим пример отношения, которое не является отношением в 2НФ:

ИСТОЧНИК СНАБЖЕНИЯ (Номер поставщика, Номер партии товара, Имя поставщика, Сведения о поставщике, Цена).

У этого отношения один возможный ключ, который является составным: (Номер поставщика, Номер парии товара). Атрибут Имя поставщика не входит в ключ, так как одной и той же фирме в разных районах могут быть присвоены различные номера поставщика. Таким образом, атрибут Номер поставщика не определяется значением атрибута Имя поставщика. В этом отношении атрибуты Имя поставщика и Сведения о

поставщике, не будучи основными, функционально зависят от атрибута Номер поставщика, который является подмножеством составного ключа.

Нарушение условий 2НФ приводит к ряду неудобств:

1. Графа Сведения о поставщике не может быть заполнена до фактической поставки конкретной партии товара поставщиком, либо необходимо задать какой-нибудь фиктивный номер партии.

2. Если поставщик временно задержал поставку некоторой партии, то удаление кортежа, соответствующего данному значению атрибута Номер поставщика, вызовет удаление сведений о нем.

3. Если требуется изменить значение атрибута Сведения о поставщике, то придется внести одни и те же изменения сразу в несколько кортежей.

Устранение подобных трудностей достигается разложением отношения на два отношения, представленных в 2НФ, говоря другими словами, приведением отношения к 2НФ. Наше отношение ИСТОЧНИК СНАБЖЕНИЯ можно разложить на два отношения:

ПОСТАВЩИК (Номер поставщика, Имя поставщика, Сведения о поставщике) и ТОВАР (Номер поставщика, Номер партии товара, Цена). Легко убедиться, что отношения ПОСТАВЩИК и ТОВАР находятся в 2НФ.

В ряде случаев и вторая нормальная форма порождает неудобства. Для их устранения используется еще один шаг нормализации, преобразующий 2НФ в третью нормальную форму (3НФ). На этом шаге ликвидируется так называемая транзитивная зависимость атрибутов.

Если для атрибутов A, B, C выполняются условия A -> B и B -> C, но обратная зависимость отсутствует, то говорят, что C зависит от A транзитивно. Отношение находится в 3НФ, если оно находится в 2НФ и в нем отсутствуют транзитивные зависимости неосновных атрибутов от каждого возможного ключа.

В отношении СЛУЖАЩИЙ, ранее представленном диаграммой, атрибут Дата окончания зависит от атрибута Номер проекта, который в свою очередь зависит от атрибута Номер служащего. Таким образом, Дата окончания транзитивно зависит от атрибута Номер служащего. Это отношение можно привести к 3НФ, расщепив его на два отношения:

СЛУЖАЩИЙ (Номер служащего, Имя служащего, Зарплата, Номер проекта), ПРОЕКТ (Номер проекта, Дата окончания).

Следует отметить, что уже известны и другие нормальные формы отношений: усиленная 3НФ – нормальная форма Бойса-Кодда, четвертая нормальная форма и пятая нормальная форма. Они позволяют устранить зависимости ключей от неосновных атрибутов, независимые многозначные зависимости и т. д. Рассмотрение этих форм выходит за пределы данной публикации.

При проектировании БД процессу нормализации отношений отводится следующее место:

– вначале составляются исходные отношения проекта БД с использованием объектно-связной модели для отображения объектов предметной области и связей между ними;

– затем производится нормализация, т. е. разложение исходных отношений и назначение ключей новых отношений в соответствии с правилами нормализации;

– далее схемы нормализованных отношений описываются средствами СУБД и вводятся в компьютер.

В Microsoft Access, прежде чем создавать таблицы, формы и другие объекты, необходимо задать структуру базы данных. Хорошая структура базы данных является основой для создания адекватной требованиям, эффективной базы данных.

Этапы проектирования баз данных

К этапам проектирования баз данных можно отнести следующие:

– Определение цели создания базы данных.

– Определение таблиц базы данных.

– Определение необходимых в таблице полей.

– Задание индивидуального значения каждому полю.

– Определение связей между таблицами.

– Обновление структуры базы данных.

– Добавление данных и создание других объектов базы данных.

– Использование средств анализа в Microsoft Access.

Определение цели создания базы данных

На первом этапе проектирования базы данных необходимо определить цель создания базы данных, основные ее функции и информацию, которую она должна содержать. То есть нужно определить основные темы таблиц базы данных и информацию, которую будут содержать поля таблиц.

База данных должна отвечать требованиям тех, кто будет непосредственно с ней работать. Для этого нужно определить темы, которые должна покрывать база данных, отчеты, которые она должна выдавать, проанализировать формы, которые в настоящий момент используются для записи данных, сравнить создаваемую базу данных с хорошо спроектированной, подобной ей базой.

Определение таблиц база данных

Одним из наиболее сложных этапов в процессе проектирования базы данных является разработка таблиц, так как результаты, которые должна

выдавать база данных (отчеты, выходные формы и др.) не всегда дают полное представление о структуре таблицы.

При проектировании таблиц вовсе не обязательно использовать Microsoft Access. Сначала лучше разработать структуру на бумаге. При проектировании таблиц рекомендуется руководствоваться следующими основными принципами:

– Информация в таблице не должна дублироваться. Не должно быть повторений и между таблицами.

Когда определенная информация хранится только в одной таблице, то и изменять ее придется только в одном месте. Это делает работу более эффективной, а также исключает возможность несовпадения информации в разных таблицах. Например, в одной таблице должны содержаться адреса и телефоны клиентов.

– Каждая таблица должна содержать информацию только на одну тему.

Сведения на каждую тему обрабатываются намного легче, если содержатся они в независимых друг от друга таблицах. Например, адреса и заказы клиентов хранятся в разных таблицах, с тем чтобы при удалении заказа информация о клиенте осталась в базе данных.

Определение необходимых в таблице полей

Каждая таблица содержит информацию на отдельную тему, а каждое поле в таблице содержит отдельные сведения по теме таблицы. Например, в таблице с данными о клиенте могут содержаться поля с названием компании, адресом, городом, страной и номером телефона. При разработке полей для каждой таблицы необходимо помнить:

– Каждое поле должно быть связано с темой таблицы.

– Не рекомендуется включать в таблицу данные, которые являются результатом выражения.

– В таблице должна присутствовать вся необходимая информация.

– Информацию следует разбивать на наименьшие логические единицы (Например, поля «Имя» и «Фамилия», а не общее поле «Имя»).

Создание ключей и связей между таблицами

С тем чтобы Microsoft Access мог связать данные из разных таблиц, например, данные о клиенте и его заказы, каждая таблица должна содержать поле или набор полей, которые будут задавать индивидуальное значение каждой записи в таблице. Такое поле или набор полей называют основным ключом.

После распределения данных по таблицам и определения ключевых полей необходимо выбрать схему для связи данных в разных таблицах. Для этого нужно определить связи между таблицами.

Желательно изучить связи между таблицами в существующей базе данных.

После проектирования таблиц, полей и связей необходимо еще раз просмотреть структуру базы данных и выявить возможные недочеты. Желательно это сделать на данном этапе, пока таблицы не заполнены данными.

Для проверки необходимо создать несколько таблиц, определить связи между ними и ввести несколько записей в каждую таблицу, затем посмотреть, отвечает ли база данных поставленным требованиям. Рекомендуется также создать черновые выходные формы и отчеты и проверить, выдают ли они требуемую информацию. Кроме того, необходимо исключить из таблиц все возможные повторения данных.

Добавление данных и создание других объектов БД

Если структуры таблиц отвечают поставленным требованиям, то можно вводить все данные. Затем можно создавать любые запросы, формы, отчеты, макросы и модули.

В Microsoft Access существует два инструмента для усовершенствования структуры баз данных. Мастер анализа таблиц исследует таблицу, в случае необходимости предлагает новую ее структуру и связи, а также переделывает ее.

Анализатор быстродействия исследует всю базу данных, дает рекомендации по ее улучшению, а также осуществляет их.

Защита информации в базах данных

Существуют различные приемы управления доступом к базе данных Microsoft Access и ее объектам. Эти приемы кратко описаны ниже в порядке повышения уровня безопасности.

Кодирование базы данных – это простейший способ защиты. При кодировании базы данных ее файл сжимается и становится недоступным для чтения с помощью служебных программ или текстовых редакторов. Кодирование незащищенной базы данных неэффективно, поскольку каждый сможет открыть такую базу данных и получить полный доступ ко всем ее объектам. Кодирование обычно применяется при электронной передаче базы данных или сохранении ее на компакт-диск или флешку.

Чтобы приступить к кодированию базы данных Microsoft Access, необходимо быть либо ее владельцем, либо, если база данных использует средства защиты, членом группы «Admins» в файле рабочей группы, который содержит учетные записи, используемые для защиты базы данных. Кроме

того, базу данных надо открыть в монопольном режиме, для чего необходимо иметь разрешения «открытие/запуск» и «монопольный доступ».

Декодирование базы данных является операцией, обратной кодированию. Другим способом защиты объектов в базе данных от посторонних пользователей является скрытие объектов в окне базы данных. Этот способ защиты является наименее надежным, поскольку относительно просто можно отобразить любые скрытые объекты.

Параметры запуска позволяют задать такие настройки, как стартовая форма, которая автоматически открывается при открытии базы данных, а также заголовок и значок приложения базы данных. Кроме того, можно скрыть окно базы данных и установить собственную кнопочную форму. В новой базе данных параметры запуска отсутствуют до тех пор, пока не внесены изменения в диалоговом окне Параметры запуска.

Другим простейшим способом защиты является установка пароля для открытия базы данных (.mdb). После установки пароля при каждом открытии базы данных будет появляться диалоговое окно, в которое требуется ввести пароль. Только те пользователи, которые введут правильный пароль, смогут открыть базу данных. После открытия базы данных все объекты становятся доступными для пользователя (пока не определены другие типы защиты, описанные ниже в этом разделе). Для базы данных, которая совместно используется небольшой группой пользователей или на автономном компьютере, обычно оказывается достаточно установки пароля.

Microsoft Access хранит пароль базы данных в незашифрованном виде. Если это нарушает безопасность защищаемой паролем базы данных, то для защиты базы данных не следует использовать пароль. Вместо этого определите защиту на уровне пользователей, которая помогает управлять доступом к важной информации в базе данных.

Использование защиты на уровне пользователя

Наиболее гибкий и распространенный способ реализации средств защиты базы данных называют защитой на уровне пользователя. Защита на уровне пользователя позволяет установить различные уровни доступа к важным данным и объектам в базе данных. Чтобы воспользоваться базой данных, защищенной на уровне пользователя, необходимо ввести пароль при запуске Microsoft Access. После этого анализируется файл рабочей группы, в котором каждый пользователь идентифицируется уникальным кодом. Уровень доступа и объекты, доступ к которым получает пользователь, зависят от кода и пароля.

Хотя установка защиты на уровне пользователей для большинства баз данных является сложной задачей, мастер защиты позволит быстро и легко

защитить базу данных. Более того, благодаря использованию общих схем защиты мастер позволяет уменьшить или даже вообще исключить необходимость использования команды Защита в меню Сервис.

После запуска мастера защиты можно создать собственные группы пользователей и определить разрешения на работу с базой данных и ее таблицами, запросами, формами, отчетами и макросами для различных пользователей или групп пользователей. Также могут быть установлены разрешения на доступ, по умолчанию присваиваемые вновь создаваемым объектам базы данных. Группам и пользователям предоставляются разрешения, определяющие возможность их доступа к каждому объекту базы данных.

В многопользовательской среде часто возникают ситуации, требующие использования средств защиты базы данных. Возможно, потребуется запретить репликацию базы данных. Репликация позволяет пользователям создавать копию общей базы данных, а также добавлять поля и вносить другие изменения в текущую базу данных. Кроме того, может потребоваться запрещение установки пароля базы данных пользователями, поскольку, если это произойдет, никто, не зная пароля, не сможет открыть базу данных. Также следует рассмотреть возможность установки запрета на изменение параметров запуска, которые определяют такие свойства, как настраиваемые меню, настраиваемые панели инструментов и стартовую форму.

Если общая база данных не имеет защиты на уровне пользователей, невозможно запретить пользователям вносить подобные изменения. При установке защиты на уровне пользователей пользователь или группа для репликации базы данных, установления пароля базы данных и изменения параметров запуска должны иметь такое же разрешение на доступ, как и администратор. Только члены группы «Admins» текущей рабочей группы имеют права администратора.

Если пользователь или группа имеют в настоящий момент разрешение на доступ к базе данных, соответствующее полномочиям администратора, удаление разрешения запретит пользователю или группе внесение изменений. Можно присвоить соответствующее разрешение на доступ пользователю или группе, что позволит им выполнять эти задачи. Невозможно независимо управлять доступом для каждой из таких задач.

3 ПРАКТИЧЕСКИЙ РАЗДЕЛ

3.1 Описание лабораторных работ

Лабораторная работа № 1

Создание базы данных

Цель работы: сформировать умения создавать структуры таблиц и устанавливать связи между ними.

Создание файла пустой базы данных выполняют следующим образом. Запускают MS Access и нажимают на кнопку **Создать** на панели **База данных** (можно выполнить команду **Создать** в меню **Файл**). На панели **Область задач** выбирают задачу **Новая база данных**. В появившемся диалоговом окне **Файл новой базы данных** указывают имя создаваемого файла базы данных и открывают папку, в которой будет храниться база данных. После этого нажимают на кнопку **Создать**. На экране появится окно созданной пустой базы данных.

Для создания структуры (макета) таблицы MS Access во вкладке **Таблицы** окна базы данных предлагает три способа: в режиме конструктора, с помощью мастера и путем ввода данных. Если надо осуществить импорт таблицы из другого приложения или установить связь с таблицей, созданной другим приложением, можно поступить следующим образом: во вкладке **Таблицы** окна базы данных нажать кнопку **Создать** и из появившегося окна **Новая таблица** выбрать команду **Импорт таблиц** или **Связь с таблицами** соответственно. После этого в диалоговом окне **Новая таблица** надо указать режим создания таблицы.

Наиболее удобным способом создания структуры таблицы является способ, использующий режим конструктора. При его использовании выполняют двойной щелчок мышью на команде **Создание таблицы в режиме конструктора** во вкладке **Таблицы** окна базы данных. На экране появится окно таблицы в режиме конструктора (см. рисунок 1).

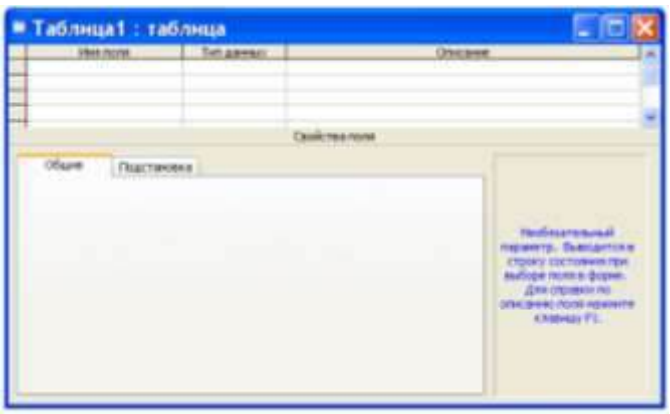


Рис. 1. Окно таблицы в режиме конструктора

В области проекта таблицы указывают имена полей, типы данных и описание. Колонка **Описание** является не обязательной – в ней можно записывать комментарии. Нижняя часть окна таблицы в режиме таблицы может быть использована для установки свойств данных, отличных от принципа умолчания.

После того, как в таблицу, представленную в режиме таблицы, внесена необходимая информация, ее следует сохранить. Для этого в меню **Файл** можно выбрать команду **Сохранить** или на панели **База данных** нажать на кнопку **Сохранить**. В появившемся окне **Сохранение** указывают имя сохраняемой таблицы и нажимают кнопку **ОК**.

Когда все макеты таблиц для проектируемой базы данных созданы, приступают к установке связей между таблицами. Это делается именно в этот период для того, чтобы при вводе данных в таблицы MS Access проверял целостность данных.

Установка связей между таблицами выполняется в окне **Схема данных**, которое выводится на экран нажатием кнопки **Схема данных** на панели **База данных**. При этом появляется диалоговое окно **Добавление таблицы**, в котором надо выделить имена тех таблиц, между которыми будут устанавливаться связи. После этого нажимают кнопки **Добавить** и **Закрыть**. Затем в окне **Схема данных** с помощью мыши перетаскивают ключевое поле одной таблицы на соответствующее поле в другой таблице. В появившемся окне **Связи** задают режим **Обеспечение целостности данных** и его подрежимы: **каскадное обновление связанных полей** и **каскадное удаление связанных записей** и нажимают кнопку **Создать**.

Задание

1. Создайте файл пустой базы данных **Библиотека**.
2. Создайте структуру таблицы **Издательства**, которая содержит следующие поля: **Код издательства**, **Наименование**, **Город** (см. таблицу 1). Имена полей таблиц из базы данных **Библиотека**: **Издательства**, **Книги** и **Темы**, типы данных, свойства полей, отличные от принципа умолчания, а также поля, являющиеся ключами, приведены в таблице 1.

Таблица 1– Поля таблиц базы данных **Библиотека**

Название таблицы	Имя поля	Тип данных	Свойства поля	Ключ
Издательства	Код издательства	Числовой	Размер поля – целое, обязательное поле	Да
Издательства	Наименование	Текстовый	Размер поля – 20	
Издательства	Город	Текстовый	Размер поля – 15	
Книги	Код книги	Числовой	Размер поля – целое, обязательное поле	Да

Книги	Название	Текстовый	Размер поля – 25	
Книги	Автор	Текстовый	Размер поля – 15	
Книги	Код издательства	Числовой	Размер поля – целое, обязательное поле	
Книги	Объем	Числовой	Размер поля – целое	
Книги	Год издания	Числовой	Размер поля – целое	
Книги	Стоимость	Денежный	Формат поля – денежный	
Темы	Код книги	Числовой	Размер поля – целое, индексированное поле (совпадения допускаются)	
Темы	Тема	Текстовый	Размер поля – 30	

После того, как вы набрали имена полей таблицы **Издательства**, указали для них типы данных, установили требуемые свойства и определили ключ, нажмите на кнопку **Закрыть** окна таблицы в режиме конструктора. При этом появится сообщение: **Сохранить изменения макета или структуры таблицы «Таблица1» ?** Ответьте на него утвердительно. После этого появится диалоговое окно **Сохранение**, в котором наберите имя таблицы **Издательства** и нажмите копку **ОК**.

3. Описанные выше действия выполните для создания таблиц **Книги** и **Темы** базы данных **Библиотека**.

4. Установите связи между таблицами **Издательства**, **Книги** и **Темы** базы данных **Библиотека** так, как это показано на рис. 2. Обратите внимание на то, что между таблицами базы данных **Библиотека** будут связи одного типа – один ко многим.

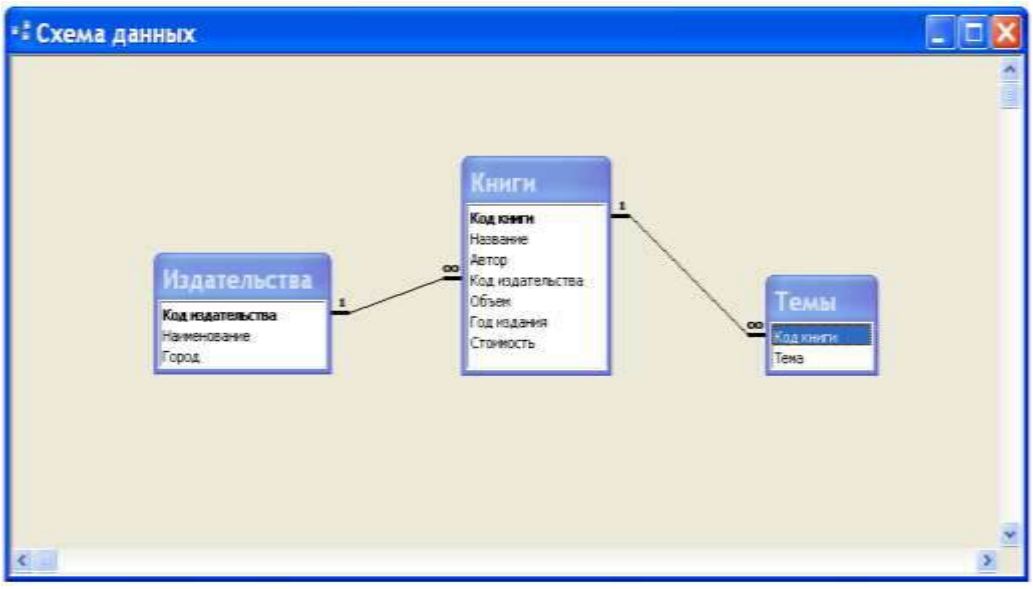


Рис. 2. Схема базы данных Библиотека

Поскольку при установке связей между таблицами мы указали режим обеспечения целостности данных с его подрежимами, то тем самым мы задали порядок, в котором надо вводить данные в таблицы. Вначале надо

вводить данные в таблицу **Издательства**, затем – в таблицу **Книги** и лишь после этого – в таблицу **Темы**.

5. Введите в таблицы **Издательства**, **Книги** и **Темы** базы данных **Библиотека** данные, приведенные ниже в таблицах 2-4. Ввод данных в таблицу осуществляется в режиме таблицы. Чтобы открыть таблицу в режиме таблицы достаточно в окне базы данных во вкладке **Таблицы** выполнить двойной щелчок мышью на имени таблицы. Быстро перевести таблицу из режима конструктора в режим таблицы и наоборот можно нажатием кнопки **Вид** на панели **База данных**. Для перехода от одного поля к другому, когда осуществляется ввод данных в таблицу, удобно использовать клавишу **Tab**.

Таблица 2 – Издательства

Код издательства	Наименование	Город
1	Наука	Москва
2	Мир	Москва
3	Радио и связь	Минск
4	Машиностроение	Киев
5	Финансы и статистика	Москва

Таблица 3 – Книги

Код книги	Название	Автор	Код издательства	Объем	Год издания	Стоимость
1	Педагогика	Беспалько	2	340	2024	24,00р.
2	Сборник задач	Сканави	2	634	2022	60,00р.
3	Программирование	Арсак	1	273	2019	18,00р.
4	Язык Ада	Перминов	3	278	2017	16,00р.
5	Операционные системы	Грибанов	3	446	2021	23,00р.
6	Базы данных	Ульман	4	563	2022	32,00р.
7	Программное обеспечение	Фигурнов	5	368	2024	22,00р.

Таблица 4 – Темы

Код книги	Тема	Код книги	Тема
1	Личность человека	5	Управление заданиями
1	Проектирование ППС	5	Управление задачами
1	Технология обучения	5	Управление данными
1	Анализ учебного процесса	6	Операции с поставщиками
2	Уравнения	6	Бухгалтерская книга
2	Прогрессии	6	Платежная ведомость
2	Геометрические задачи	6	Реляционная алгебра
3	Игры с числами	6	Правила нормализации
3	Игры без стратегии	7	Устройства компьютера
3	Комбинаторные задачи	7	Файлы и каталоги
3	Стратегия без игры	7	Диалог пользователя
4	Программные модули	7	Работа с дисками
4	Лексика	7	Программы архивации
4	Предопределенные типы	7	Конфигурирование системы
4	Операторы	7	Обслуживание дисков
5	Структура ОС ЕС	7	Редактирование текстов

Лабораторная работа № 2

Расширение базы данных Библиотека

Цель работы: сформировать умения добавлять таблицы в базу данных с целью расширения ее функциональных возможностей.

Иногда в процессе разработки базы данных или в процессе опытной эксплуатации ее возникает необходимость добавления в нее новых таблиц. Очевидно, что спроектированная нами в предыдущей работе база данных **Библиотека** обладает очень ограниченными возможностями. Эта база данных, состоящая из трех таблиц: **Издательства**, **Книги** и **Темы**, не позволяет автоматизировать работу с читателями. В ней отсутствует информация о читателях.

В данной работе мы научимся добавлять таблицы в базу данных с целью расширения ее функциональных возможностей. Создание новых таблиц осуществляется точно так же, как это мы делали в предыдущей работе. Для добавления таблиц в ранее созданную схему данных и установления связи между таблицами используется кнопка **Отобразить таблицу**, размещенная на панели инструментов **Связь**.

Задание

1. Откройте базу данных **Библиотека**. Создайте в ней структуру таблицы **Читатели**, которая будет содержать следующие поля: **Код читателя**, **Фамилию**, **Имя**, **Отчество**, **Домашний телефон**, **Домашний адрес.** Типы данных для полей таблицы, их свойства определите самостоятельно по смыслу. В качестве ключа укажите поле **Код читателя**.

2. Аналогичным способом создайте структуру таблицы **Выдача книг**. В эту структуру включите три поля: **Код читателя**, **Код книги**, **Дата заказа**. В этой таблице ключевое поле не задавайте. Для поля **Дата заказа** укажите тип данных – **Дата/время**. Обратите внимание на то, что в последствии ключ **Код читатели** в таблице **Читатели** будет связываться с полем **Код читателя** в таблице **Выдача книг**. Поэтому эти поля должны иметь соответствующие типы данных и свойства.

3. Установите между добавленными таблицами: **Читатели** и **Выдача книг**, а также ранее созданными таблицами: **Издательства**, **Книги** и **Темы**, связи так, как это показано в окне **Схема данных** на рисунке 3.

Напомним, что для установления связи между таблицами надо открыть окно **Схема данных.** При его открытии появляется диалоговое окно **Добавление таблицы**, в котором надо выделить имена тех таблиц, между которыми будут устанавливаться связи. После этого нажимают кнопки **Добавить** и **Закрыть**. Затем в окне **Схема данных** с помощью мыши

перетаскивают ключевое поле одной таблицы на соответствующее поле в другой таблице. В появившемся окне **Связи** задают режим **Обеспечение целостности данных** и его подрежимы: **каскадное обновление связанных полей** и **каскадное удаление связанных записей** и нажимают кнопку **Создать**.

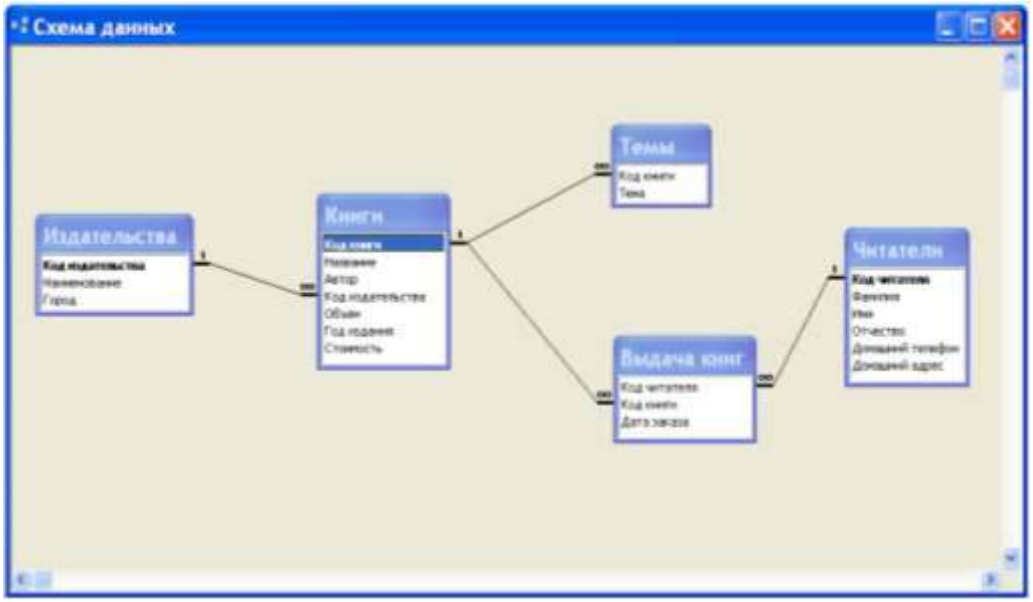


Рис. 3. Схема расширенной базы данных **Библиотека**

4. Откройте таблицу **Читатели** и введите в нее данные, приведенные в таблице 1.

Таблица 1 – Данные для ввода в таблицу **Читатели**

Код чита-теля	Фамилия	Имя	Отчество	Домашний телефон	Домашний адрес
1	Аксенов	Виктор	Сергеевич	252-88-13	ул. Есенина, 15-19
2	Голубева	Елена	Андреевна	220-99-29	ул. Чкалова, 7-38
3	Васильев	Игорь	Петрович	232-64-78	ул. Богдановича, 102-34
4	Кучеров	Валентин	Степанович	266-24-95	ул. Кнорина, 27-5
5	Мастяница	Вячеслав	Иванович	246-42-25	ул. Плеханова, 34-98
6	Победимская	Лариса	Анатольевна		ул. Чкалова, 9-10
7	Литвин	Борис	Николаевич	239-55-76	пр. Независимости, 46-54
8	Германович	Рита	Мироновна	278-31-51	ул. Казинца, 26-9
9	Бинцаровский	Теодор	Петрович		ул. Корженевская, 1-288

5. Введите в таблицу **Выдача книг** данные, приведенные в таблице 2.

Таблица 2 – Данные для ввода в таблицу **Выдача книг**

Код читателя	Код книги	Дата заказа	Код читателя	Код книги	Дата заказа
1	1	1.09.23	4	3	7.01.24
1	3	5.07.24	4	4	25.10.23
1	4	21.10.23	5	2	23.04.24
2	1	4.11.23	6	1	18.06.24
3	2	3.08.24	7	3	20.01.24
8	7	25.12.23	9	6	2.02.24

Обратите внимание на то, что, если бы вы попробовали вначале ввести данные в таблицу **Выдача книг**, а затем в таблицу **Читатели**, то MS Access это не позволил бы сделать. Поэтому мы специально раньше установили связи между таблицами, а затем уже вводили данные в таблицы. В этом случае MS Access будет проверять целостность данных.

Лабораторная работа № 3

Создание простых запросов

Цель работы: сформировать умения создавать простые запросы для выбора данных.

Запрос в MS Access – это требование предоставить информацию, накопленную в таблицах базы данных. Запрос можно получить с помощью с помощью инструментов запроса. Запрос может относиться к одной или к нескольким связанным таблицам. На основании запроса MS Access формирует динамический набор записей. Физически он выглядит как таблица, хотя фактически не является ею. Динамический набор записей является временным (или виртуальным) набором записей и не хранится в базе данных. После закрытия запроса динамический набор записей этого запроса прекращает свое существование.

MS Access поддерживает различные типы запросов, которые можно разбить на шесть основных категорий.

Запрос на выборку. Извлекает данные из одной или нескольких таблиц (основываясь на заданных критериях) и результаты представляет в виде динамического набора записей.

Групповой запрос. Представляет специальную версию запроса на выборку. Позволяет вычислять суммы, подсчитывать количество записей и

выполнять расчет итоговых значений. Для этого запроса MS Access добавляет в бланк запроса строку **Групповая операция.**

Запрос на изменение. Позволяет создавать новые таблицы (команда **Создание таблицы**) или изменять данные в существующих таблицах (команды **Удаление**, **Обновление** и **Добавление**). Если в наборе результатов запроса на выборку можно вносить изменения только в одну запись за раз, то запрос на изменение разрешает вносить изменения в несколько записей сразу при выполнении этой операции.

Перекрестный запрос. Отображает результаты статистических расчетов (такие как суммы, количество записей и средние значения). Эти результаты группируются по двум наборам данных в формате перекрестной таблицы. Первый набор выводится в столбце слева и образует заголовки строк, а второй выводится в верхней строке и образует заголовки столбцов.

Запрос SQL. Существуют три типа запросов SQL: запрос на объединение, запрос к серверу и управляющий запрос, которые используются для манипуляций с базами данных SQL. Создаются эти запросы с помощью написания специальных инструкций SQL.

Запрос с ограничением, или Top(n). Этот ограничитель запроса можно использовать только в паре с одним из предыдущих пяти типов запросов. Он позволяет задавать число первых записей или часть общего количества записей в процентах, которую вы хотели бы получить в любом виде запроса.

С помощью запросов можно выполнять следующее: выбирать таблицы, выбирать поля, выбирать записи, сортировать записи, выполнять вычисления, создавать таблицы, создавать формы и отчеты на основе запроса, создавать диаграммы на основе запроса, использовать запрос в качестве источника данных для других запросов (подчиненных запросов) и вносить изменения в таблицы.

Создание запроса и работа с ним выполняется во вкладке **Запросы** окна базы данных. Для работы с запросом можно воспользоваться панелью инструментов **Конструктор запросов** (см. рисунок 1).

Рис. 1. Панель инструментов **Конструктор запросов**

MS Access допускает два способа создания запроса: с помощью мастера и в режиме конструктора. Для того чтобы приступить к созданию запроса с помощью мастера можно выполнить двойной щелчок мышью на строке **Создание запроса с помощью мастера** во вкладке **Запросы** окна

базы данных или щелчок мышью на кнопке **Создать**, а затем выбрать вариант **Простой запрос** в окне диалога **Новый запрос**.

Создание запроса на выборку для сортировки информации

В работе далее для создания запросов будем использовать режим конструктора. Самый быстрый способ запустить этот режим – выполнить двойной щелчок мышью на строке **Создание запроса в режиме конструктора**. При этом появится окно диалога **Добавление таблицы** (см. рисунок 2).

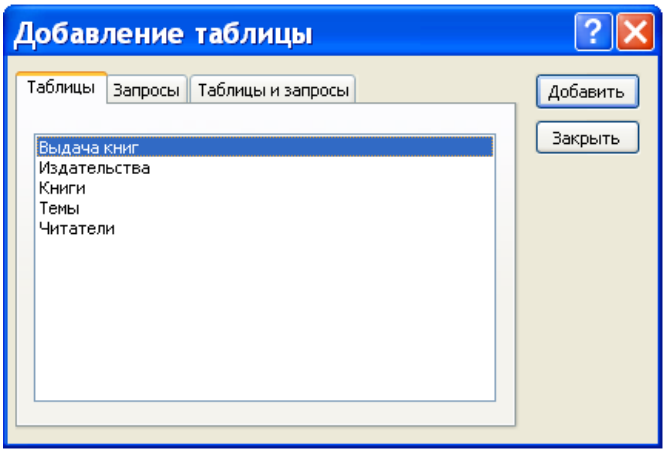


Рис. 2. Окно диалога **Добавление таблицы**

Создание запроса для сортировки информации рассмотрим на следующем примере. Требуется составить список книг московских издательств, рассортированных по фамилиям авторов. В динамический набор надо включить следующие поля: **Автор**, **Название**, **Наименование** и **Год издания**.

Обратим внимание на то, что в нашем запросе будут использоваться поля из двух таблиц: **Издательства** и **Книги**. Поэтому в окне диалога надо выделить имена этих двух таблиц. Для этих целей щелкните вначале, например, по имени **Издательства**, а затем, удерживая клавишу **CTRL**, щелкните по имени **Книги**. После того как требуемые имена таблиц выделены, надо в окне диалога **Добавление таблицы** щелкнуть мышью по кнопке **Добавить**, а затем – **Закрыть**. В результате выполнения таких действий в верхней части окна запроса в режиме конструктора появятся списки полей для каждой из выбранных таблиц (см. рисунок 3).

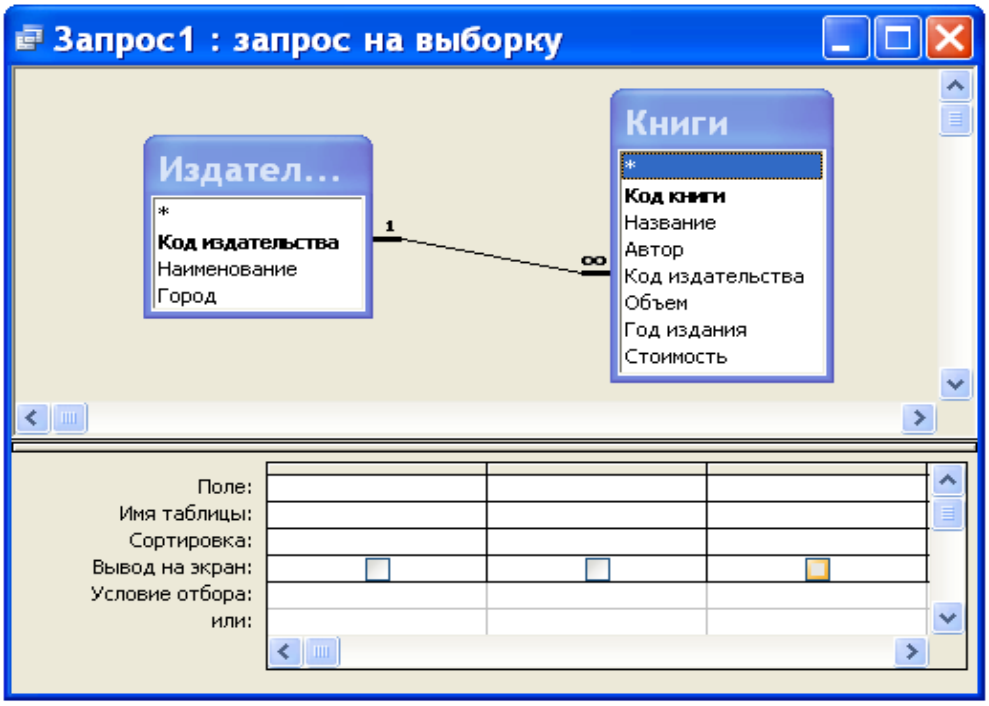


Рис. 3. Окно запроса в режиме конструктора

Окно запроса в режиме конструктора предназначено для создания новых и изменения существующих запросов. При создании запросов в этом режиме используется механизм запросов по образцу **QBE** (Query by Example). Окно в этом случае состоит из двух частей. В верхней части окна размещаются списки полей, из которых будет формироваться запрос. В нижней части окна располагается бланк **QBE**, в который нужные для запроса поля перемещаются при помощи мыши из списков полей, размещенных в верхней части окна.

Для изменения относительной высоты верхней и нижней частей окна используется специальная разделительная линия. При установке курсора на эту линию курсор приобретает вид двунаправленной стрелки. В это момент разделительную линию можно перемещать вверх или вниз.

Имена полей, которые будут образовывать динамический набор, должны быть в соответствующем порядке размещены в строке бланка **QBE**. Сделать это можно несколькими способами. Самый простой способ состоит в двойном щелчке мышью на имени в списке полей. Указанным способом в строке **Поле** бланка **QBE** поместите поля: **Автор**, **Название**, **Наименование, Год издания** и **Город**. Последнее поле нам понадобилось, чтобы задать условие отбора для выбора для выбора книг московских издательств.

В строку **Условие отбора** для поля **Город** наберите текст "Москва" (задание условий отбора подробнее будет рассмотрено ниже). Даже, если вы текст в кавычки не возьмете, MS Access сам это сделает. Условие отбора нам понадобилось для того, чтобы в запросе выбирались не все книги, а только книги, изданные в Москве.

Поскольку по условию задачи поле **Город** не надо выводить на экран, то в строке **Вывод на экран** для этого поля уберите щелчком мыши пометку ("птичку").

Для того чтобы в динамическом наборе записи выводились в алфавитном порядке по фамилиям авторов, надо в строке **Сортировка** для поля **Автор** задать направление сортировки. Выполните щелчок мышью на ячейке в строке **Сортировка** для поля **Автор**. При этом справа в этой ячейке появится кнопка раскрытия списка направления сортировки. Выберите в этом списке направление сортировки – по возрастанию.

Порядок обработки полей при сортировке по нескольким полям определяется их положением в бланке **QBE**: сначала сортируются значения в крайнем левом поле и далее слева направо. После указанных действий бланк **QBE** будет иметь вид, представленный на рисунке 4.

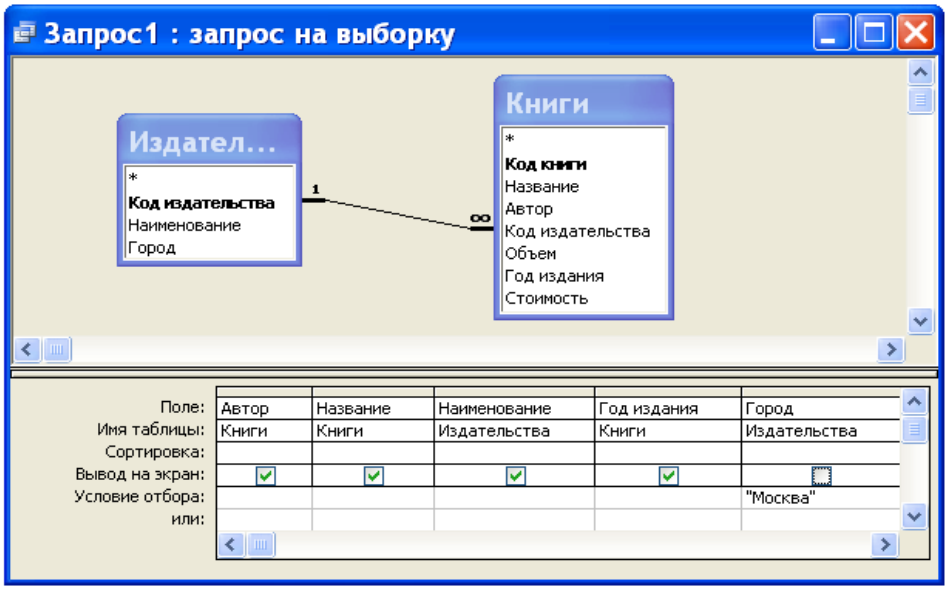


Рис. 4. Вид бланка **QBE** для решения задачи

Сейчас выполним созданный нами запрос. Для этого нажмите кнопку **Режим таблицы** на панели инструментов **Конструктор запросов** (первая кнопка – см. рисунок 1). После нажатия этой кнопки вы увидите список книг московских издательств, рассортированный в алфавитном порядке по фамилиям авторов (см. рисунок 5).

Для того чтобы установить оптимальную ширину столбца списка, надо выполнить двойной щелчок мышью на правой границе столбца в строке заголовков полей. Установите оптимальную ширину для всех столбцов списка книг, как это сделано на рисунке 5.

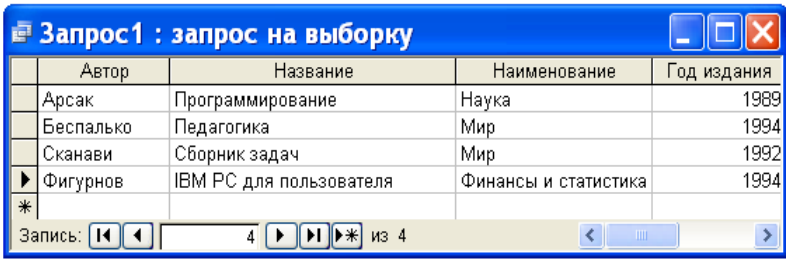


Рис. 5. Результат выполнения запроса.

После того как запрос создан, его можно сохранить. Для этой цели надо выполнить команду **Сохранить запрос** или **Сохранить запрос как** в меню **Файл**. Если мы выполняем сохранение первый раз, то выполнение этих команд приводит к одному и тому же результату – на экране появляется окно диалога, приведенное на рисунке 6.

Рис. 6. Окно диалога для сохранения запроса

Сохраните созданный нами запрос под именем Список книг московских издательств. Для этого введите новое имя (старое имя Запрос1, которое предложил Access, после нажатия первой клавиши исчезнет, так что нет необходимости специально его убирать) и нажмите кнопку OK.

Отбор данных

Основное назначение запроса состоит в формировании динамического набора, записи которого удовлетворяют некоторым условиям. Условия отбора записей вводятся как выражения. Выражение указывает, какие записи следует включить в динамический набор при выполнении запроса. Выражения могут быть простыми (например, <30) или сложными (например, Between 100 And 500).

Определить условия отбора можно самостоятельно, введя нужное выражение в ячейку **Условия отбора**, соответствующую данному полю, или воспользоваться построителем выражения. Для определения условия с помощью построителя выражений вначале устанавливают указатель в ячейку **Условия отбора** в бланке **QBE**, в которой следует определить выражение, и нажимают кнопку мыши. После этого нажимают кнопку **Построить** на панели инструментов. На экране появляется диалоговое окно **Построитель выражений**, приведенное на рисунке 7.

Если в ячейке бланка **QBE**, из которой вызывался построитель, содержится значение, то это значение автоматически копируется в поле построения выражения. Используя построитель выражений, можно вводить

символы в область ввода или нажимать кнопки для ввода операторов, а также вставлять ссылки на объекты и другие элементы выражения, выбирая их из папок.

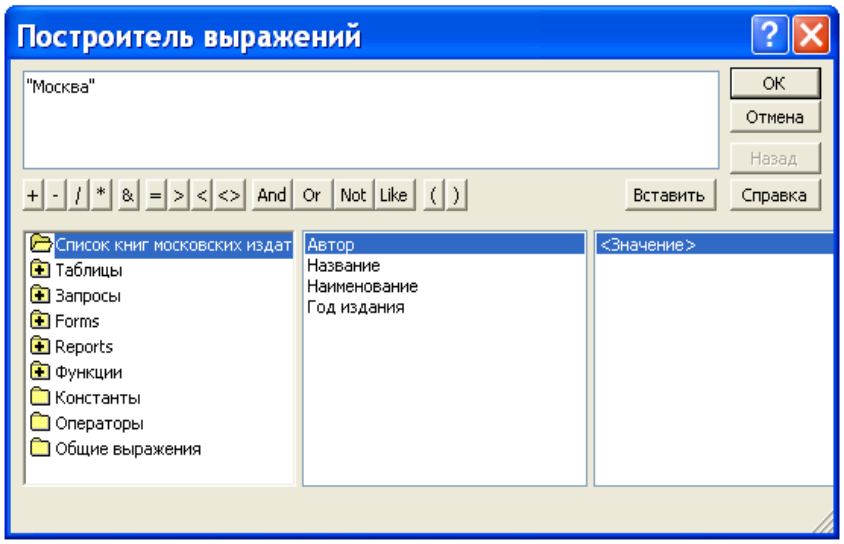


Рис. 7. Окно диалога **Построитель выражений**

Вставка операторов в выражение из строки операторов, расположенной ниже поля построителя, выполняется щелчком мыши на операторе.

Для вставки элемента поступают следующим образом. В левом нижнем поле построителя выбирают папку, содержащую нужный элемент. В нижнем среднем поле дважды щелкают элемент, чтобы вставить его в поле выражения, или выбирают тип элемента. Если выбран тип в нижнем среднем поле, то значения будут отображаться в нижнем правом поле. Дважды щелкните значение, чтобы вставить его в поле выражения.

Вставьте необходимые операторы в выражение. Для этого поместите указатель мыши в определенную позицию поля выражения и выберите одну из кнопок со знаками операций, расположенных в середине окна построителя. Закончив создание выражения, нажмите кнопку **OK**.

MS Access скопирует созданное выражение в ту позицию, из которой был вызван построитель выражений. Если в данной позиции уже содержится значение, то исходное значение будет заменено новым выражением.

Следует иметь в виду, что любая часть выражения или все выражение может быть введено в поле выражения непосредственно с клавиатуры. Может также случиться, что выражение можно быстрее ввести в строку **Условие отбора** без использования построителя выражений.

Выражение – комбинация операторов, констант, литералов, значений, функций, названий свойств, имен полей и элементов управления, при оценке которых получается одно значение. Оператор – это символ или слово (например, > или Or), указывающее на операцию, которую следует выполнить над одним или несколькими элементами. Операторы

сгруппированы в классы операторов, например, арифметические, сравнения, логические.

В выражениях для условий отбора допускается использование символов шаблона. Символами шаблона являются звездочка (*), знак вопроса (?), знак номера (#), восклицательный знак (!), дефис (-) и квадратные скобки ([]). Эти символы можно использовать в запросах, командах и выражениях для включения всех записей, имен файлов или других элементов, которые начинаются с определенной последовательности букв или удовлетворяют указанному шаблону. Назначение и примеры использования символов шаблонов приведены в таблице 1. При вводе шаблонов можно использовать как прописные, так и строчные буквы. Например, шаблон "ст*" эквивалентен шаблону "Ст*".

Таблица 1 – Символы шаблона

Символ	Назначение	Пример	Результат отбора
*	Заменяет любую группу символов; может быть первым или последним символом в шаблоне.	ст* *иск	"стол", "станок" и т. п. "иск", "диск", "риск" и т. п.
?	Заменяет любой один символ.	ко?а	"кора", "коса", "коза" и т. п.
#	Заменяет любую одну цифру.	5#4	504, 554, 514 и т. п.
[]	Заменяет любой один символ, указанный в скобках.	ко[рс]а	"кора" и "коса", но не коза
!	Заменяет любой один символ, кроме символов, указанных в скобках.	ко[!рс]а	"коза" и "кожа", но не "кора" и "коса"
-	Заменяет любой один символ из указанного диапазона.	ко[к-м]а	"кока", "кола" и "кома"

После завершения ввода выражения в ячейку строки **Условие отбора** (например, нажатием клавиши Enter, клавиш управления курсором или щелчком мыши в другой ячейке) выполняется синтаксический анализ этого выражения и выражение приводится в соответствие с правилами синтаксиса MS Access. Например, если введено слово Москва, то добавляются прямые кавычки и это слово выводится как "Москва".

Если выражение не содержит оператор, то подразумевается оператор равняется (=). Например, если в ячейку **Условие отбора** для поля **Город** введено слово Москва, то выражение интерпретируется как **Город** = "Москва".

Задание

1. Выведите список книг, цена которых находится в диапазоне от 20 до 30 рублей. Динамический набор этого запроса должен содержать поля: **Автор**, **Название**, **Год издания**, **Стоимость**. Для задания условия отбора вначале используйте оператор **Between** … **And**, а затем операторы >=, <=, **And**. Записи в динамическом наборе расположите по возрастанию цены книг. Сохраните первый запрос под именем **Цена книг из диапазона**, а второй – под именем **Операторы сравнения для поиска цены**.

2. Выведите список читателей, у которых нет домашнего телефона. В список включите следующие поля: **Фамилия**, **Имя**, **Отчество**, **Домашний адрес**. Список рассортируйте в алфавитном порядке по фамилии, имени и отчеству. Для поиска требуемых записей в строке **Условие отбора** для поля **Домашний телефон** используйте выражение **Is Null**. Это выражение предназначено для поиска записей, у которых поле не содержит значение (является пустым). Если требуется отобрать записи, у которых поле имеет значение, то можно использовать выражение **Is Not Null**. Запрос сохраните под именем **Читатели без домашних телефонов**.

3. Выведите список читателей, у которых в домашнем телефоне вторая цифра есть 5 или 6. В динамический набор включите поля: **Фамилия**, **Имя**, **Отчество**, **Домашний телефон**. Условие отбора для поля может иметь следующий вид: **Like "?[56]*"**.

Запрос сохраните под именем **Использование символов шаблона**. Измените условие отбора предыдущего запроса так, чтобы он выводил список всех читателей, в номерах телефонов которых вторая цифра не 5 и не 6. Полученный запрос сохраните под именем **Символ отрицания в квадратных скобках**.

4. Создайте запрос, который будет выводить список книг, заказанных читателями в 2007 году. В динамический набор включите следующие поля: **Автор**, **Название**, **Наименование**, **Город**. Для решения задачи вначале используйте функцию DatePart(interval; date; firstweekday; firstweek), а затем Format(expr; fmt; firstweekday; firstweek). Запросы сохраните под именами **Использование функции DatePart** и **Использование функции Format** соответственно.

5. В таблицу **Выдача книг** базы данных **Библиотека** добавьте поле **Дата возврата**. В это поле для записей, приведенных в таблице 2, введите даты возврата. В остальных записях поле **Дата возврата** должно остаться пустым.

Таблица 2 – Учет возврата книг

Код читателя	Код книги	Дата заказа	Дата возврата
1	1	01.09.2023	15.10.2023
1	3	05.07.2024	23.09.2024
4	3	07.01.2024	02.03.2024
5	2	23.04.2024	03.05.2024
7	3	20.01.2024	11.04.2024
9	6	02.02.2024	03.03.2024

Составьте запрос, который будет выводить список читателей, которые не сдали своевременно книги (предполагается, что читатель может держать книгу на руках не более 100 дней). В динамический набор включите следующие поля: **Фамилия**, **Имя**, **Отчество**, **Домашний телефон**, **Автор**, **Название**, **Стоимость**. Для решения задачи воспользуйтесь функцией **Date**(). Запрос сохраните под именем **Читатели, не сдавшие своевременно книги**.

Лабораторная работа № 4

Просмотр и изменение динамического набора

Цель работы: сформировать знания о редактировании динамического набора. Сформировать умения изменять свойства запросов и его элементов.

Запрос и его элементы (поля и списки полей) имеют свойства. Свойства запроса определяют поведение запроса в целом. Например, можно определить свойство, запрещающее включение повторяющихся значений в динамический набор. Свойства поля определяют поведение данных в поле. Например, можно определить формат изображения чисел в поле. Свойства можно определять для любого поля, кроме звездочки и полей, для которых не установлен флажок (имеет вид крестика) **Вывод на экран**. Свойства списка полей влияют на один из списков полей таблицы и запроса, включенных в запрос. Например, можно определить свойство, задающее нестандартное название списка полей.

Опишем последовательность действий, которые надо выполнять при просмотре, определении или изменении свойств запроса или его элементов. Чтобы это сделать, надо вначале выделить запрос или его элементы:

– для выделения всего запроса следует выполнить щелчок мышью в любом месте окна запроса вне бланка QBE и списков полей;

– для выделения поля следует выполнить щелчок мышью в соответствующей ячейке в строке Поле;

– для выделения списка полей следует выполнить щелчок мышью в любом месте этого списка.

После этого надо выбрать команду Свойства в меню Вид или нажать кнопку Свойства на панели инструментов. На экране появится окно свойств выделенного объекта. Окно свойств объекта для случая, когда объектом является поле, приведено на рисунке 1.

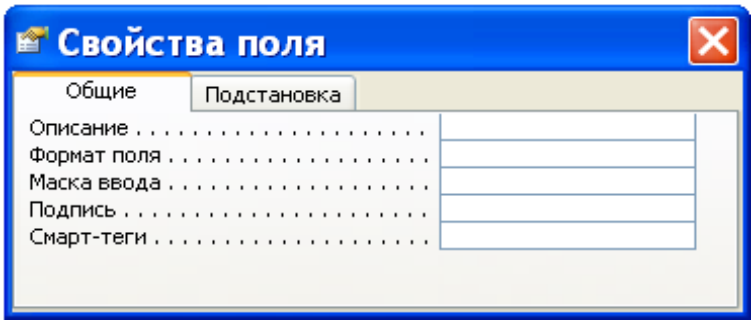


Рис. 1. Окно Свойства поля

Затем, если требуется определить или изменить какое-либо свойство в этом бланке, раскрывают список возможных значений данного свойства, нажав кнопку раскрытия списка (если такая кнопка есть), и выбирают требуемое значение из списка или вводят допустимое значение. На рисунке 2 раскрыт список значений свойства **Формат поля** с типом данных дата/время.

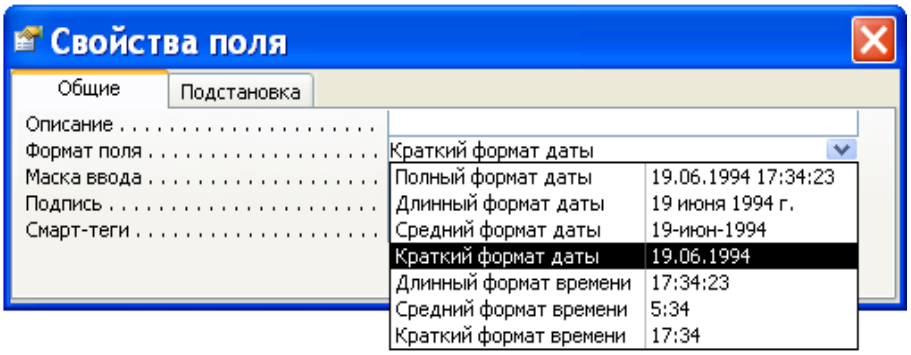


Рис. 2. Определение свойства Формат поля

Для определения значений некоторых свойств, например, Маска ввода, можно использовать построители. Справа от ячейки бланка свойств, соответствующей такому свойству, расположена кнопка Построить, которую можно нажать для вызова построителя.

Из списка значений свойства Формат поля, приведенного на рис. 2, видно, что дата может иметь четыре формата: полный, длинный, средний и краткий. В макете базовой таблицы Выдача книг поле Дата заказа имеет краткий формат (например, "19.06.1994"). Полный, длинный и средний форматы для этого значения даты имеют соответственно вид: "19.06.1994 17:34:23", "19 июня 1994 г." и "19-июн-1994". На этом же рисунке показано, что время может изображаться в трех форматах: длинном ("17:34:23"), среднем ("5:34") и кратком ("17:34").

Аналогичным образом можно определять свойства полей для изображения других типов данных. Например, числа можно изображать с десятичным разделителем или знаком процента.

В запросе для изображения данных можно устанавливать форматы, отличные от форматов полей базовой таблицы. При этом важно знать следующее. По умолчанию поля в запросе наследуют все свойства соответствующих полей базовой таблицы или запроса. При изменении свойства поля в макете базовой таблицы это изменение будет автоматически отражено в макете запроса, однако после изменения свойства поля в режиме конструктора запросов новое значение заменит установленное для базового поля, и все последующие изменения этого свойства в макете таблицы не будут отражены в запросе.

Рассмотрим, как можно выполнять операции над полями (изменение порядка, вставка и удаление) после включения их в запрос. Перемещение поля в бланке QBE запроса выполняется просто: вначале выделяют поле (выполняют щелчок мышью в области маркировки столбца), а затем, не меняя положения указателя, нажимают кнопку мыши и перемещают столбец на новое место.

Для вставки поля в бланк QBE запроса надо в списке полей выделить поле, которое следует вставить, и перенести его из списка полей в нужный столбец бланка QBE.

Удаление всех полей из бланка QBE выполняется командой Очистить бланк в меню Правка. Чтобы удалить одно поле, надо вначале его выделить, а затем выполнить команду Удалить в меню Правка или нажать клавишу Del.

Улучшить внешний вид запроса можно, изменив ширину столбцов. Для изменения ширины столбца устанавливают указатель на правую границу в области маркировки столбца и перемещают ее влево (для уменьшения ширины) или вправо (для увеличения ширины). Очень быстро можно установить оптимальную ширину столбца. Оптимальной называют такую ширину, при которой в поле помещается самое длинное значение в этом столбце (в расчет принимается и ширина заголовка столбца). Установка оптимальной ширины столбца осуществляется двойным щелчком мышью на правой границе столбца в области маркировки. Если до выполнения этой операции было выделено несколько столбцов, то после ее выполнения для каждого из выделенных столбцов будет установлена оптимальная для него ширина.

Следует иметь в виду, что, если после установки оптимальной ширины в столбец будет добавлено значение, длина которого превышает текущую ширину столбца, то для отображения значения полностью придется снова повторить описанную выше процедуру. Обратите внимание на то, что рассмотренные процедуры изменяют ширину столбцов в бланке QBE. При просмотре результата выполнения запроса в режиме таблицы вам еще раз придется изменять ширину столбцов.

Иногда для улучшения наглядности и читабельности запроса в режиме таблицы требуется изменить название поля. Сделать это можно следующим образом. Откройте запрос в режиме конструктора или перейдите в режим конструктора, если текущим является режим таблицы. Установите указатель слева от первой буквы имени поля в бланке QBE и введите перед старым именем новое имя с двоеточием (см. рисунок 3).

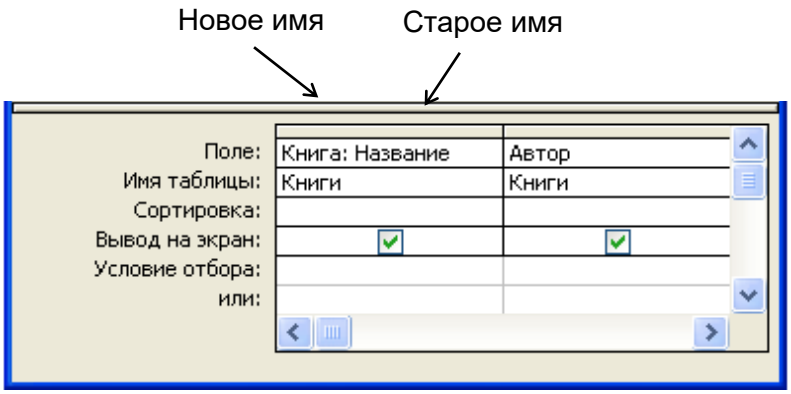


Рис. 3. Переименование поля

Лабораторная работа № 5

Запросы с параметрами

Цель работы: сформировать умения для создания запросов с параметрами.

Часто встречаются ситуации, когда перед выполнением запроса надо изменять условия отбора. В таком случае целесообразно создать запрос с параметрами. При выполнении запроса с параметрами не требуется открывать окно запроса и вносить изменения в бланк **QBE**. Вместо этого пользователю надо ввести нужное условие отбора в диалоговое окно **Введите значение параметра**. Запрос с параметрами может содержать несколько параметров. Тогда перед каждым выполнением запроса на экране будет появляться определенная пользователем последовательность диалоговых окон, предназначенных для ввода условий отбора.

Параметр имеет имя. Это имя определяет разработчик запроса и записывает его в квадратных скобках. Имя параметра может записываться в строке **Условия отбора** или в строке **Поле** бланка **QBE**. Если в бланке **QBE** Access встречает в квадратных скобках текст, не совпадающий с именем поля, то он автоматически считает его именем параметра.

Для создания запроса с параметрами выполните следующие действия:

– Создайте запрос в режиме конструктора и включите в него нужные таблицы. Перенесите нужные поля в бланк **QBE**.

– В ячейку **Условие отбора** поля, которое планируется использовать для определения параметра, введите текст (см. рисунок 1), заключенный в квадратные скобки. Этот и будет именем параметра, он появится на экране при выполнении запроса. Имя параметра должно отличаться от имен полей.

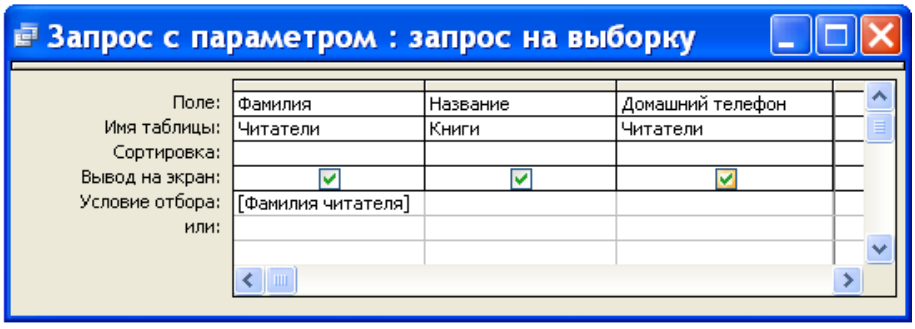


Рис. 1. Пример записи имени параметра

– Выберите команду **Параметры** в меню **Запрос**. На экране появится диалоговое окно **Параметры запроса** (см. рисунок 2). В первую ячейку **Параметры** введите имя параметра, которое было введено в первую ячейку

бланка **QBE** при составлении запроса с параметром. В ячейке, расположенной справа от имени параметра, установите нужный тип данных. Если запрос содержит несколько параметров, продолжите ввод имен параметров и установку типов данных. При выполнении запроса пользователю будет предложено ввести значения параметров в том порядке, в котором они расположены в диалоговом окне **Параметры запроса**. Тип данных для параметров можно и не задавать, тогда он наследуется из базовой таблицы.

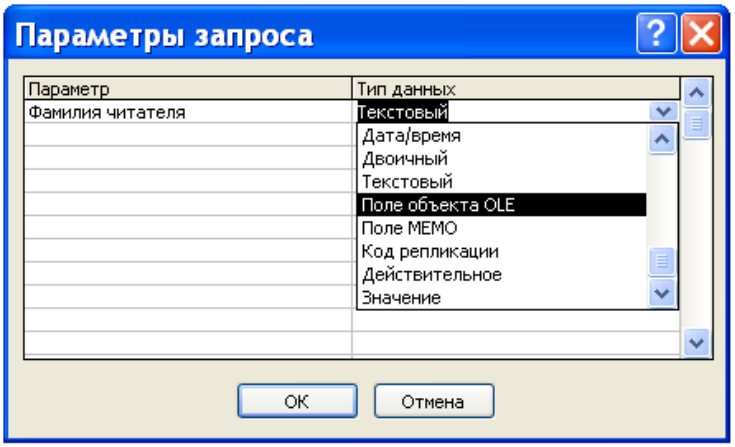


Рис. 2. Установка типа данных для параметров.

– Выберите команду **Таблица** в меню **Вид** или нажмите кнопку **Режим таблицы** на панели инструментов. На экране появится диалоговое окно **Введите значение параметра** (см. рисунок 3).

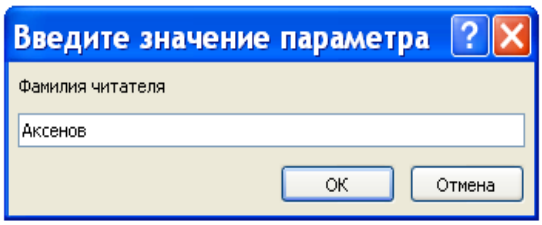


Рис. 3. Диалоговое окно для ввода значения параметра

– Введите значение параметра и нажмите кнопку **OK**. MS Access выполнит отбор данных и выведет на экран динамический набор или, если запрос содержит несколько параметров, на экране появится диалоговое окно для ввода значения следующего параметра. После ввода значений всех параметров на экране появится динамический набор.

Задание

1. В качестве упражнения создайте запрос, который будет осуществлять поиск книг по ключевому слову в теме. Назовите данный запрос **Книги с ключевым словом в теме**. Бланк **QBE** для данного запроса приведен на рисунке 4. Обратите внимание на то, что при вводе значения

параметра в диалоговое окно надо указывать корень слова, а не само слово, иначе некоторые книги могут быть не найдены.

Сократить количество релевантных сведений при поиске книг можно, указав несколько ключевых слов. Создайте запрос, который будет осуществлять поиск книг по двум ключевым словам в теме. Запрос назовите **Поиск по двум ключевым словам**.

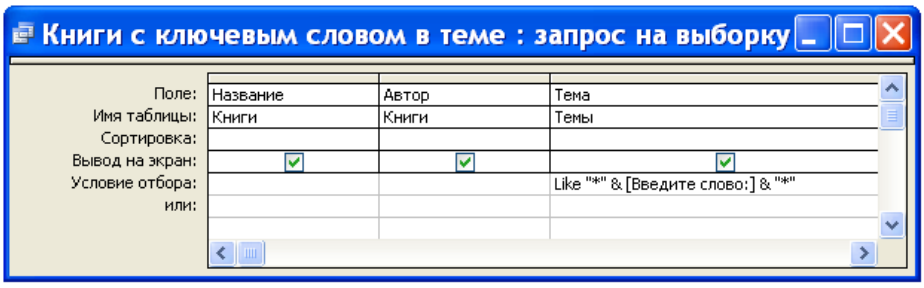


Рис. 4. Поиск книг по ключевому слову в теме

2. Запросы с параметрами удобно использовать для указания нескольких первых букв искомого значения. На рис. 5 показан бланк QBE для запроса, который будет осуществлять поиск книг по нескольким первым буквам фамилии автора.

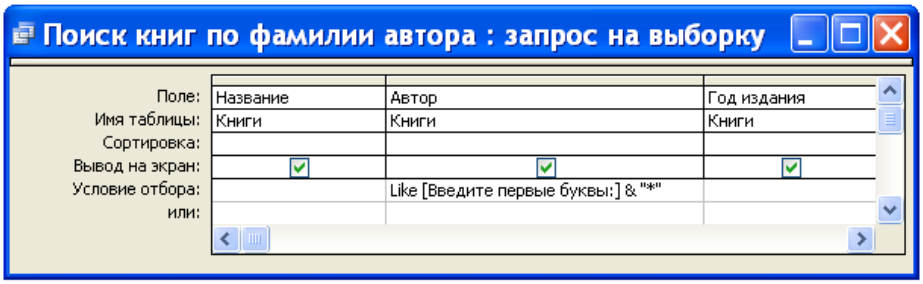


Рис. 5. Поиск книг по первым буквам фамилии автора

Сохраните данный запрос под именем Поиск книг по фамилии автора и выполните его несколько раз для наборов первых букв фамилии автора, содержащих различное количество символов.

3. Создайте запрос, который будет выводить список читателей, которые вовремя не сдали книги. В качестве параметра возьмите период (количество дней), в течение которого читатель может держать книги на руках. Для этих целей измените ранее созданный запрос с именем Читатели, своевременно не сдавшие книги, добавив в него параметр. Новый запрос назовите Список читателей для вызова.

4. Создайте запрос, который будет подсчитывать количество книг, заказанных в конкретном месяце года. Год и номер месяца возьмите в качестве параметров. Запрос назовите Заказы книг по месяцам. В динамический набор включите три поля, которым дайте следующие имена:

Год, Номер месяца, Количество книг. Напомним, что для создания этого запроса в бланке QBE понадобится строка Групповая операция.

Лабораторная работа № 6

Вычисляемые поля

Цель работы: сформировать умения создавать вычисляемые поля для организации вычислений.

Вычисляемое поле – это такое поле, которое не содержится ни в одной из таблиц базы данных, а создается с помощью выражений. Для расчетов с использованием формул, определяемых пользователем, требуется создать новое вычисляемое поле прямо в бланке запроса. Вычисляемое поле создается с помощью выражения, которое вводится в пустую ячейку **Поле** бланка запроса.

Вычисляемое поле имеет следующий формат:

Имя вычисляемого поля: выражение для построения вычисляемого поля

Если при создании вычисляемого поля пользователь не указывает имя, то Access по умолчанию присвоит ему имя **Выражение1**. Имя для вычисляемого поля рекомендуется задавать по двум причинам: во-первых, для заголовка столбца таблицы, содержащей динамический набор запроса, и, во-вторых, для обращения к этому полю в форме, отчете или другом запросе.

Рассмотрим пример. Требуется вывести список читателей в алфавитном порядке, содержащий фамилии, инициалы и домашний адрес. Поскольку поля с инициалами читателей ни в одной таблице базы данных **Библиотека** нет, нам потребуется создать вычисляемое поле для выделения инициалов из имени и отчества читателей. Вычисляемое поле назовем **Фамилия и инициалы**. Запрос сохраним под именем **Список читателей с инициалами**.

Вид бланка запроса для вывода списка читателей с фамилиями и инициалами представлен на рисунке 1.

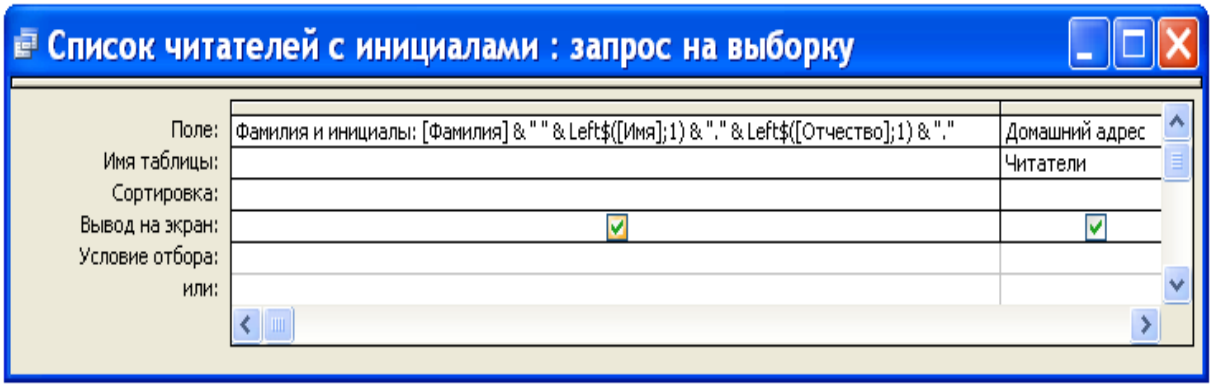


Рис. 1. Бланк запроса для вывода списка читателей

Выражение в вычисляемом поле, приведенном на рис. 1, является текстовым. Оператор конъюнкция ("&") в нем используется для сцепления строк, так, например, выражение [Фамилия]&" " сцепляет значение поля **Фамилия** с пробелом. Функция Left$([Имя];1) в данном контексте используется для выделения одной левой буквы из значения поля **Имя**. Динамический набор записей в результате выполнения данного запроса будет иметь вид, представленный на рисунке 2.

Рис. 2. Динамический набор запроса

Убедитесь в том, что в рассмотренном примере вместо функции Left$([Имя];1) мы бы могли воспользоваться более общей функцией Mid$([Имя];1;1). Функция Mid$([Имя поля];n;m) позволяет из указанного поля выделить подряд расположенных m символов, начиная с номера n. Более того, тот же результат получился бы, если мы в названиях используемых функций убрали бы символ доллара ("$").

Задание

1. Создайте вычисляемое поле для вычисления новой цены книг. Новая цена должна определяться умножением значения поля **Стоимость** на величину 1,1. Вычисляемое поле назовите **Новая цена**. Для этого поля установите денежный формат. В динамический набор запроса включите следующие поля: **Автор**, **Название**, **Год издания**, **Новая цена**. Запрос сохраните под именем **Стоимость книг с учетом инфляции**.

2. Создайте вычисляемое поле для вычисления цены книг в условных единицах. Курс доллара примите равным 3,5 белорусских рублей. В динамический набор включите те же поля, что и в предыдущем примере, но вычисляемое поле назовите **Цена в у_е**. Для вычисляемого поля в его свойствах задайте пользовательский формат, позволяющий отображать информацию так, как это показано на рисунке 3.

Рис. 3. Форматирование вычисляемого поля

Получение указанного результата могут обеспечить свойства вычисляемого поля, приведенные на рисунке 4. Запрос сохраните под именем **Стоимость книг в условных единицах**.

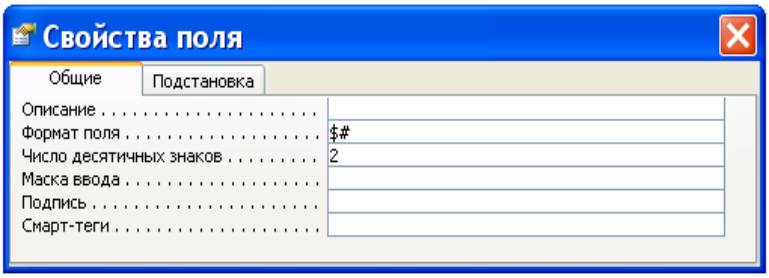


Рис. 4. Свойства вычисляемого поля

3. Подсчитайте, сколько книг заказывал каждый читатель за весь период использования абонемента библиотеки. В динамический набор включите вычисляемое поле **Фамилия и инициалы** и новое поле **Количество**, значения которого вычисляются в результате применения групповой операции Count над полем **Код книги** из таблицы **Выдача книг** (см. рисунок 5).

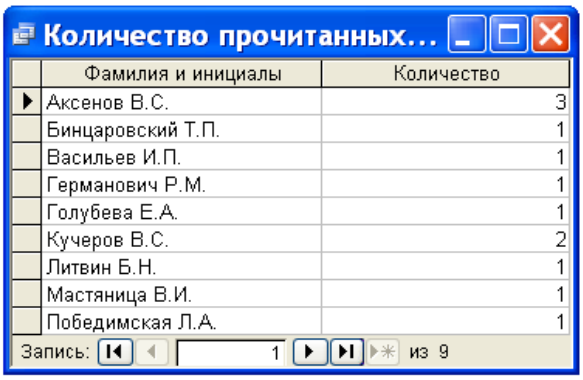


Рис. 5. Использование групповых операций для вычислений

Для поля **Фамилия и инициалы** в строке **Групповая операция** бланка запроса выберите операцию Группировка, а в строке **Сортировка** выберите направление сортировки: по возрастанию. Запрос сохраните под именем **Количество прочитанных книг**.

4. По аналогии с предыдущим заданием, используя групповые операции, создайте запрос, который позволит получить информацию о том, сколько раз заказывали каждую книгу. В динамический набор запроса включите поля **Автор**, **Название** из таблицы **Книги** и новое поле **Рейтинг**, значение которого указывает, сколько раз заказывалась данная книга (см. рисунок 6).

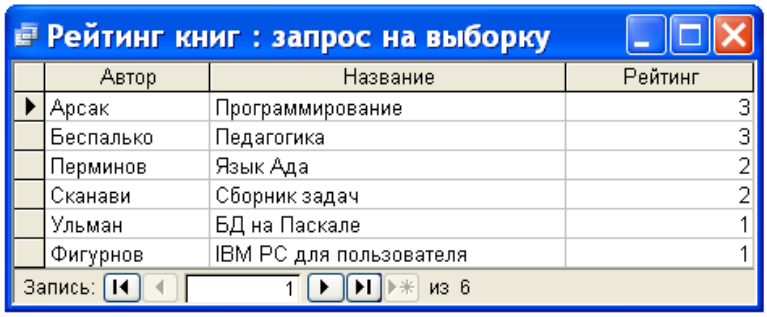


Рис. 6. Результат выполнения запроса **Рейтинг книг**

Книги, которые не заказывались, в динамический набор запроса включать не надо. Запрос сохраните под именем **Рейтинг книг**.

Лабораторная работа № 7

Перекрестный запрос

Цель работы: сформировать умения для создания перекрестных запросов.

Перекрестные запросы предназначены для группирования данных и представления их в компактном виде, напоминающем электронную таблицу. Перекрестный запрос позволяет представить большой объем данных в виде, удобном для восприятия, анализа и сравнения. Более того, перекрестный запрос удобно использовать в качестве базового при создании отчета. Следует иметь в виду, что в результате выполнения перекрестного запроса получается не динамический, а статический набор записей, то есть такой набор записей, который нельзя обновлять.

Проще всего создать перекрестный запрос с помощью мастера по разработке перекрестных запросов. При необходимости перекрестный запрос можно создать без помощи мастера. Рассмотрим два этих способа создания перекрестного запроса.

Создание перекрестного запроса с помощью мастера выполняют следующим образом:

– Находясь в окне базы данных, выбирают корешок **Запрос** и нажимают кнопку **Создать**.

– В диалоговом окне **Создание запроса** нажимают кнопку **Мастера по разработке запросов**.

– В первом окне **Мастера по разработке запросов** выбирают пункт **Перекрестный запрос**.

– Далее выполняют инструкции, появляющиеся в диалоговых окнах. В последнем диалоговом окне нажимают кнопку **Готово**.

Для создания перекрестного запроса без помощи мастера надо выполнить следующую последовательность действий:

1. В окне базы данных выберите корешок **Запрос** и нажмите кнопку **Создать**. В появившемся диалоговом окне **Создание запроса** нажмите кнопку **Новый запрос**.

2. Выберите таблицы или запросы, содержащие поля, которые следует включить в запрос. Перенесите нужные поля в строку **Поле** бланка **QBE** и задайте условия отбора.

3. Выберите команду **Перекрестный** в меню **Запрос** или нажмите кнопку **Перекрестный** на панели инструментов. В бланке **QBE** появятся строки **Групповые операции** и перекрестная таблица. По умолчанию ячейки

Групповые операции, соответствующие каждому полю, включенному в запрос, будут содержать надпись **Группировка**.

4. Установите указатель в ячейку **Перекрестная таблица**, соответствующую полю, которое содержит заголовки строк, и нажмите кнопку мыши, а затем нажмите кнопку раскрытия списка и выберите строку **Заголовки строк**. Можно указать несколько полей с заголовком строк. В ячейке **Групповые операции**, соответствующей по крайней мере одному из этих полей, должна содержаться надпись **Группировка**.

5. Установите указатель в ячейку **Перекрестная таблица**, соответствующую полю, которое содержит заголовки столбцов, и нажмите кнопку мыши, а затем нажмите кнопку раскрытия списка и выберите строку **Заголовки столбцов**. Только одно поле в перекрестном запросе может содержать заголовки столбцов. Ячейка **Групповые операции**, соответствующая этому полю, должна содержать надпись **Группировка**.

6. Установите указатель в ячейку **Перекрестная таблица**, соответствующую полю, которое содержит значения для вычислений, и нажмите кнопку мыши, а затем нажмите кнопку раскрытия списка и выберите строку **Значения**. Только одно поле в перекрестном запросе может содержать значения для вычислений. Если поле следует использовать для группирования, сортировки или размещения условий отбора, но не следует включать в результирующий набор записей, то нажмите кнопку раскрытия списка в ячейке **Перекрестная таблица**, соответствующей этому полю, и выберите строку **не выводить**.

7. Установите указатель в ячейку **Групповые операции**, соответствующую полю, которое содержит значения для вычислений, и нажмите кнопку мыши, а затем нажмите кнопку раскрытия списка и выберите тип групповой операции (например, **Sum**, **Max** или **Count**). Поле, которое содержит значения для вычислений, нельзя использовать для группирования; оно должно содержать результат выполнения групповой операции.

8. Для вывода на экран полученного набора записей выберите команду **Таблица** в меню **Вид** или нажмите кнопку **Режим таблицы** на панели инструментов.

Следует иметь в виду, что перекрестный запрос может содержать несколько полей с заголовками строк, но только одно поле с заголовками столбцов. Если в перекрестный запрос следует включить несколько полей с заголовками столбцов, но только одно поле с заголовками строк, поменяйте местами заголовки строк и столбцов.

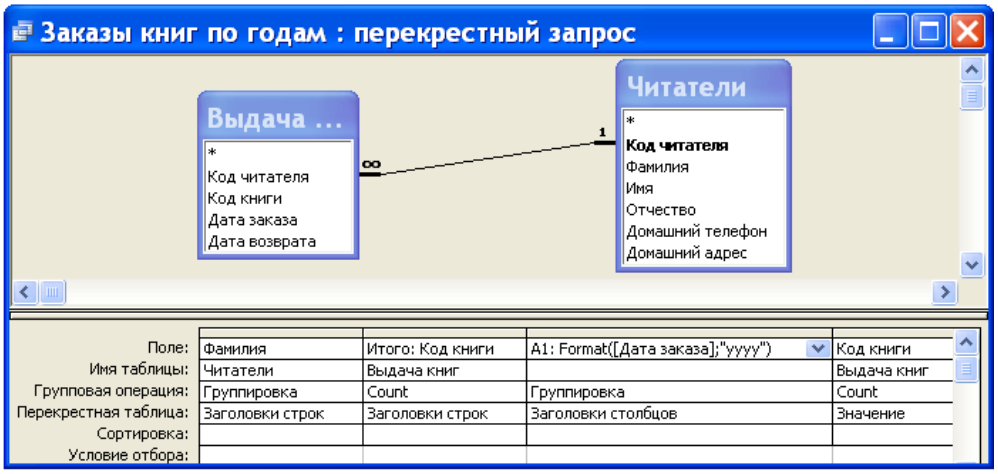


Рис. 1. Окно в режиме конструктора для запроса **Заказы книг по годам**

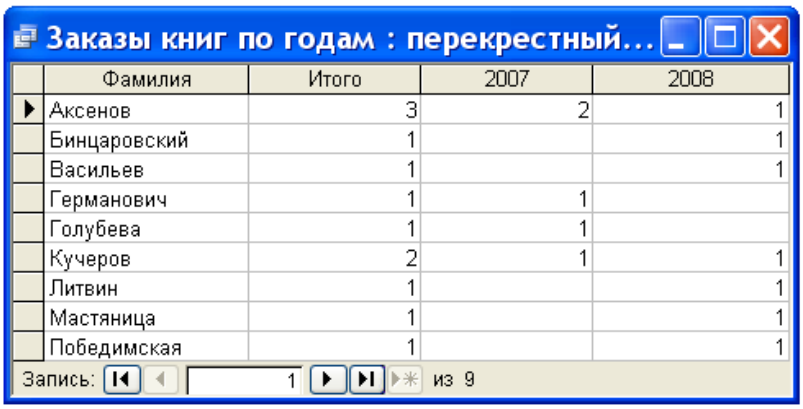
Рассмотрим следующий пример. Требуется создать список фамилий читателей, в котором для каждого читателя будет указано общее количество заказанных книг за весь период пользования услугами библиотеки и количество книг, заказанных в каждом году данного периода.

Для решения этого примера нам понадобятся две таблицы из базы данных **Библиотека**: **Читатели** и **Выдача книг**. Разместите списки этих таблиц в верхней части окна запроса, как это сделано на рисунке 1.

В бланке **QBE** окна запроса надо заполнить четыре столбца. Переместите нужные поля из верхней части окна запроса в строку **Поле** бланка **QBE**. Первые два столбца должны определять заголовки строк перекрестной таблицы: **Фамилия** и **Итого**. Для того чтобы появились в бланке **QBE** строки **Групповая операция** и **Перекрестная таблица**, надо выполнить команду **Перекрестная** в меню **Запрос** или нажать кнопку **Перекрестный** на панели инструментов. После этого заполните строки **Групповая операция** и **Перекрестная таблица**. Для первого столбца укажите групповую операцию **Группировка**, а для второго – **Count**.

Третий столбец должен определять заголовки столбцов перекрестной таблицы. В нашем примере в качестве заголовков столбцов должны быть выбраны годы из поля **Дата заказа** таблицы **Выдача книг**. Для того чтобы выбрать элемент год из значений **Дата заказа** можно использовать функцию **Format**. В четвертом столбце бланка **QBE** надо указать, какие данные будут выведены в перекрестной таблице под заголовками столбцов. С этой целью указано вычисляемое значение, которое определяется функцией **Count**.

Набор данных, созданный в результате выполнения перекрестного запроса, приведен на рисунке 2. Сохраните его под именем **Заказы книг по годам**.



Заказы книг по годам : перекрестный...

Фамилия	Итого	2007	2008
Аксенов	3	2	1
Бинцаровский	1		1
Васильев	1		1
Германович	1	1	
Голубева	1	1	
Кучеров	2	1	1
Литвин	1		1
Мастяница	1		1
Победимская	1		1

Запись: 1 из 9

Рис. 2. Результат выполнения запроса **Заказы книг по годам**

Рассмотрим еще один пример перекрестного запроса, иллюстрирующий использование постоянных заголовков столбцов. Требуется создать запрос, который будет выводить информацию о количестве выдач каждой книги на протяжении всего периода работы библиотеки, а также суммарное количество выдач каждой книги по месяцам данного периода.

Вид бланка **QBE** для решения примера приведен на рисунке 3. Первые два столбца этого бланка определяют заголовки строк для результирующей таблицы запроса. Третий столбец используется для задания заголовков столбцов. Как и в предыдущем примере для выбора месяца из поля **Дата заказа** используется функция **Format**. Поскольку данный столбец используется для группирования записей, в строке **Групповая операция** бланка **QBE** указана операция **Группировка**. Последний столбец указывает, что для определения количества выдач книг с конкретным кодом книги используется групповая операция **Count** и в требуемой таблице в качестве значений ячеек, расположенных на пересечении строки с кодом книги и столбца с названием месяца, будет располагаться значение этой операции.

Выдача книг по месяцам : перекрестный запрос

Поле:	Код книги	Итого: Код книги	B1: Format([Дата заказа];"mmm")	Код книги
Имя таблицы:	Выдача книг	Выдача книг		Выдача книг
Групповая операция:	Группировка	Count	Группировка	Count
Перекрестная таблица:	Заголовки строк	Заголовки строк	Заголовки столбцов	Значение
Сортировка:				
Условие отбора:				
или:				

Рис. 3. Вид бланка **QBE** для запроса **Выдача книг по месяцам**

Поскольку мы желаем получить результирующую таблицу перекрестного запроса, содержащую все названия месяцев, даже если некоторые из них отсутствуют в записях базы данных, надо определить эти

названия при задании свойств запроса. Заголовки столбцов перекрестного запроса, определенные как свойства запроса, называются постоянными.

Постоянные заголовки столбцов имеют одно важное преимущество по сравнению с обычными заголовками столбцов, рассмотренными в предыдущем примере, использование их повышает скорость выполнения перекрестного запроса. Другое преимущество состоит в следующем: пользователь сам может задать очередность вывода заголовков столбцов, кроме того, он может уменьшить или увеличить количество заголовков столбцов в наборе записей независимо от наличия значений для поля, которое используется при выборе значений для заголовков столбцов.

Определим постоянные заголовки столбцов для нашего примера. Будем считать, что к настоящему времени бланк **QBE** имеет такой вид, как на рисунке 3. Выделите поле **Выражение1** (напомним, что оно используется для задания заголовков столбцов) и нажмите кнопку **Свойства**. После этого на экране появится бланк свойства запроса. В ячейку **Заголовки столбцов** введите названия месяцев, которые надо использовать в качестве заголовков столбцов (см. рисунок 4).

Введенные заголовки должны полностью совпадать с соответствующими значениями в полях базы данных. Разделяются заголовки столбцов точкой с запятой или другим разделителем, который установлен на **Панели управления** Windows. При этом кавычки можно не вводить – они появятся после нажатия клавиши **Enter** или при перемещении курсора в другую ячейку.

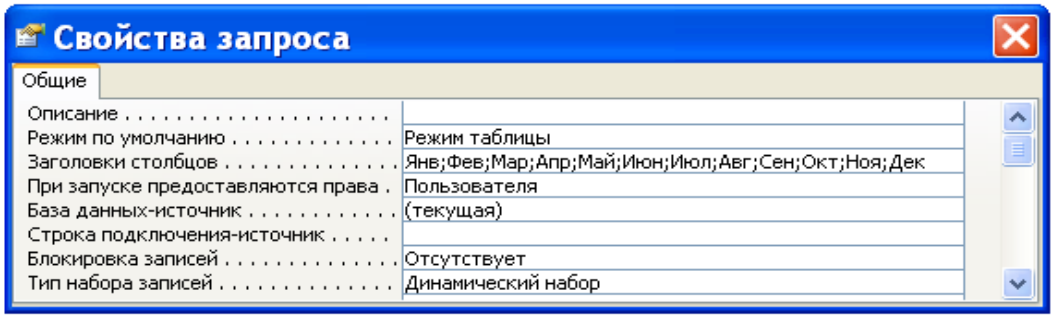


Рис. 4. Вид свойства запроса для постоянных заголовков столбцов

Для вывода на экран полученного набора записей надо выполнить команду **Таблица** в меню **Вид** или нажать кнопку **Режим таблицы** на панели инструментов. Выполните одно из указанных действий, и вы увидите на экране набор записей примерно в таком виде, в каком он приведен на рисунке 5. Сохраните этот запрос под именем **Выдача книг по месяцам**.

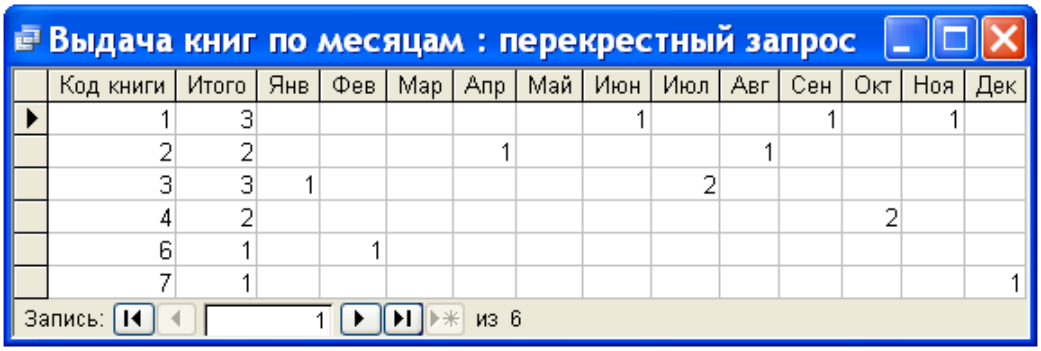


Рис. 5. Результат выполнения запроса с постоянными заголовками столбцов

Задание

1. Измените решение последнего примера так, чтобы перекрестная таблица содержала информацию о выдаче книг не за весь период работы библиотеки, а только за 2017 год. Полученный запрос назовите **Заказы книг в 2017 году**.

2. Создайте в электронной таблице MS Excel таблицу телефонных звонков, приведенную ниже.

Фамилия абонента	**Город**	**Телефон**	**Количество минут**
Голубев В.И.	Витебск	20-34-85	5
Василевская И.Л.	Гродно	36-67-91	3
Петрович А.А.	Гомель	44-34-29	8
Мухин П.И.	Брест	34-76-89	5
Петрович А.А.	Могилев	87-45-90	2
Мухин П.И.	Гомель	23-87-46	5
Голубев В.И.	Гродно	68-75-84	4
Мухин П.И.	Брест	65-78-34	6
Василевская И.Л.	Гродно	54-90-28	2

Сделайте экспорт данной таблицы в систему управления базами данных MS Access. В MS Access создайте запрос, который для всех абонентов телефонной сети определяет количество звонков в каждый город и суммарную продолжительность звонков.

Лабораторная работа № 8

Язык конструирования запросов SQL

Цель работы: сформировать умения создавать запросы с помощью языка структурированных запросов SQL**.**

В предыдущих лабораторных работах мы научились создавать запросы с помощью таких средств, как мастер и конструктор. В данной работе мы научимся использовать для этих целей язык структурированных запросов SQL (Structured Query Language).

Основным оператором языка SQL, позволяющим осуществлять отбор информации из базы данных, является оператор SELECT, который в простейшем виде может быть задан следующим образом:

```
SELECT <список колонок, включаемых в ответ> FROM <список таблиц>
WHERE <условие>;
```

Предложения SELECT (отобрать) и FROM (из) должны присутствовать обязательно. Условие WHERE (где) может быть опущено. Тогда в ответ войдут все строки, имеющиеся в таблице (SQL позволяет управлять выводом в ответ повторяющихся строк, и можно добиться как вывода только уникальных строк, так и включения в ответ повторяющихся строк).

Оператор SELECT может включать в себя и другие предложения, позволяющие, в частности, осуществлять упорядоченность ответа, выполнять обобщающие функции. Если в ответ должны войти все колонки, имеющиеся в исходной таблице, то вместо их перечисления в SELECT можно поставить знак «*».

Так, например, запрос «Выдать всю информацию о читателях из таблицы **Читатели**, которые проживают на улице Чкалова» может быть представлен на SQL следующим образом:

```
SELECT Читатели.*
FROM Читатели
WHERE ((Читатели.[Домашний адрес]) Like "ул. Чкалова" & "*");
```

Условие, задаваемое в предложении WHERE, может быть простым и сложным. Для формулирования сложного условия могут быть использованы логические операторы And и Or. Так, например, ранее составленный запрос

Операторы сравнения для поиска цены может быть представлен на SQL следующим образом:

```
SELECT Книги.Автор, Книги.Название, Книги.[Год издания], Книги.Стоимость
FROM Книги
WHERE ((Книги.Стоимость)>=20 And (Книги.Стоимость) <=30)
ORDER BY Книги.Стоимость;
```

Оператор SELECT оперирует над множествами и результатом обработки в общем случае является множество строк. К этим множествам могут быть применены теоретико-множественные операции объединение (UNION), пересечение (INTERSECTION), разность (DIFFERENCE, MINUS, EXCEPT) и др. В разных реализациях языка SQL наборы теоретико-множественных операций различаются.

Язык SQL позволяет запрашивать вычисляемые значения. В этом случае в предложении SELECT указывается выражение для вычисления значения колонки. Например, в рассмотренном ранее запросе **Стоимость книг в условных единицах** запрашивается вывод стоимости книг в условных единицах путем ее вычисления на основе хранящейся в таблице **Книги** стоимости по соответствующей формуле:

```
SELECT Книги.Автор, Книги.Название, Книги.[Год издания], [Стоимость] /3,5 AS [Цена в y_e]
FROM Книги;
```

С помощью конструкции AS в этом запросе задано имя столбца-результата.

Запрос может быть простым, состоящим из одного оператора SELECT, и вложенным, когда один оператор SELECT включается в состав другого оператора. Этот включенный оператор называется подзапросом (subselect) или подчиненным запросом. Существуют два типа вложенных подзапросов: обычный и коррелированный. В обычном подзапросе внутренний запрос выполняется первым, и его результат используется для выполнения основного запроса. В коррелированном подзапросе внешний запрос выполняется первым, и его результат используется для выполнения внутреннего запроса. Внутренний запрос выполняется для каждой строки, возвращенной внешним запросом.

В запросе можно указать упорядоченность ответа по определенному признаку (полю, совокупности полей, выражению).

Возможна подгруппировка данных в целях получения подытогов или других обобщающих величин (среднее, минимум, максимум и др.). Набор агрегатных функций отличается в разных системах. В запросе допускается только один уровень группировки. Группировка может осуществляться как по одному полю, так и по совокупности полей.

В некоторых реализациях языка SQL отобранные оператором SELECT данные могут быть сохранены в виде таблицы базы данных

При выполнении запроса может возникнуть необходимость соединения двух или более таблиц. Возможны разные способы задания условия соединения (вложенные запросы, задание условия соединения в предложении WHERE, операция JOIN в предложении FROM).

Общая характеристика оператора SELECT

Для отбора информации из базы данных служит оператор SELECT. Синтаксис оператора выглядит следующим образом:

```
SELECT [DISTINCT]
        {{функция агрегирования | выражение для вычисления значения [AS
имя столбца]}.,}
| {спецификатор.*}
|*
FROM {{ имя таблицы [AS][имя корреляции].[(имя столбца.,..)]}
| {подзапрос [AS][имя корреляции.[имя столбца.,..]}
| соединенная таблица }.,..
[WHERE предикат]
        [GROUP BY {{[ имя таблицы | имя корреляции]}.| имя столбца}.,..}]
[HAVING предикат]
[UNION IINTERSECT | EXCEPT}[ALL]
[CORRESPONDING [BY (имя столбца.,..)]]
        оператор SELECT | TABLE имя таблицы | конструктор значений
таблицы]
[ORDER BY{{столбец-результат [ASC | DESC]}.,..}
| {{положительное число[ASC | DESC]}.,..}]};
```

Оператор состоит из предложений SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, которые должны быть записаны в команде именно в той последовательности, в которой они перечислены в синтаксической формуле.

Предложение SELECT определяет столбцы таблицы, получаемой в результате выполнения запроса. Столбец результатной таблицы может быть задан именем столбца исходной таблицы. Если в запросе используется

несколько таблиц и в них имеются поля, имеющие одинаковые имена, то для указания такого поля используется конструкция <имя таблицы>.<имя поля>. Кроме того, в предложении SELECT могут использоваться любые допустимые выражения, которые зададут формулу для определения вычисляемого поля. С помощью конструкции [AS <имя столбца>] можно задать имя столбца-результата. Конструкцию AS можно использовать не только тогда, когда определяются вычисляемые поля, но и во всех других случаях, когда нужно задать имя столбца-результата, отличающееся от имени столбца исходной таблицы.

Результат выборки может в принципе содержать повторяющиеся строки. Чтобы избежать вывода повторяющихся строк в ответе, используется параметр DISTINCT.

Запросы могут использовать функции агрегирования. Стандарт языка SQL предусматривает использование следующих функций агрегирования: Count – подсчет, Sum – сумма, Max – максимум, Min – минимум, Avg - среднее.

Чаще всего функции агрегирования используются совместно с предложением GROUP BY, но могут применяться и самостоятельно. В последнем случае результат относится не к какой-то группе, а ко всей выборке.

Существуют два типа функции COUNT. Первый тип использует символ «*». В этом случае функция подсчитывает количество строк в группе. Отдельные значения столбцов при этом не учитываются, и результат не будет зависеть от того, имеются ли в полях значения Null и указан ли параметр DISTINCT. Второй тип функции COUNT игнорирует значения Null.

Если в ответ требуется включить все поля таблицы, то для этого можно использовать символ «*». Если запрос многотабличный, то следует применять конструкцию {спецификатор. *}.

В предложении FROM указываются таблицы, которые используются при формулировании запроса. Кроме этого, в качестве источника данных в запросе могут быть заданы представления.

Начиная со стандарта SQL-92, в предложение FROM можно включать встроенный оператор JOIN, который служит для задания разнообразных условий соединения таблиц, участвующих в запросе.

В предложении WHERE задается условие отбора записей. Предложение может включать одно выражение или несколько. Части сложного условия соединяются логическими операторами AND (И) или OR (ИЛИ). В выражениях могут использоваться следующие операторы сравнения: = (равно), <> (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), которые могут предваряться оператором

NOT. Выражение может принимать одно из трех значений: TRUE, FALSE, UNKNOWN. В результатную таблицу переносятся те строки, для которых значение предиката равно TRUE.

Кроме стандартных операторов сравнения в SQL можно использовать специальные операторы предикатов: <интервальный предикат >, <предикат IN>, <предикат проверки на неопределенное значение>, <предикат подобия>.

При использовании интервального предиката диапазон значений можно задавать в виде

WHERE [NOT] <выражение> BETWEEN <нижнее выражение> AND <верхнее выражение>

При использовании предиката IN предложение WHERE будет иметь следующий вид:

WHERE [NOT] <выражение> [NOT] IN <список значений>|<подзапрос>

Предикат подобия применяется для поиска подстроки в указанной строке. Предложение WHERE при использовании предиката этого типа будет иметь следующий вид:

WHERE [NOT] <выражение для вычисления значения строки 1> [NOT] LIKE <выражение для вычисления значения строки 2>

Предикат проверки на неопределенное значение имеет вид

предикат NULL ::= конструктор значения строки IS [NOT] NULL

При использовании подзапросов в условии WHERE может быть использован квантор существования EXISTS. Формат условия WHERE в этом случае имеет вид

WHERE [NOT] EXISTS <подзапрос>

Предложение GROUP BY используется для определения групп выходных строк, к которым могут применяться те или иные агрегатные функции. Предложение GROUP BY всегда используется со встроенными агрегатными функциями. Обратное утверждение неверно. Агрегатные функции могут использоваться в предложениях SELECT, HAVING. Если агрегатные функции используются без предложения GROUP BY, то они будут применяться ко всему набору строк, удовлетворяющему условию запроса. Конструкция GROUP BY работает только на одном уровне. Нельзя разбить каждую из этих групп на группы более низкого уровня, а затем применять стандартную функцию на каждом уровне подчиненности.

Фраза GROUP BY означает логическую перекомпоновку (группировку) таблицы по указанной колонке (колонкам). Физически таблицы в базе данных не перекомпоновываются. Логика выполнения запроса при использовании GROUP BY несколько отличается от реализации обычного запроса. Фраза SELECT при использовании GROUP BY применяется к

каждой группе, а не к каждой строке, как обычно.

Каждое выражение во фразе SELECT должно принимать единственное значение для группы, т. е. оно может быть либо самой колонкой, либо арифметическим выражением, включающим эту колонку, либо агрегатной функцией, которая получает в результате единственное значение для группы. Кроме того, в SELECT может быть включена константа.

Вместе с предложением GROUP BY может использоваться предложение HAVING, которое для групп имеет то же значение, что и фраза WHERE – для строк.

Корректирующие операторы

Оператор INSERT позволяет включить в таблицу новые строки. Он имеет следующий вид:

INSERT INTO имя таблицы [(имя столбца ,...)]

выражение запроса [конструктор значений таблицы] |{DEFAULT VALUES}

Если список столбцов не задан, то значения должны вводиться в каждый столбец таблицы; если список столбцов задан, то значения соответственно должны вводиться в те столбцы, которые перечислены в списке, и в том порядке, в котором они расположены в нем.

Элементы в списке значений могут быть константами, функциями, переменными памяти. Если эти элементы являются константами, то при их задании используются определенные разделители в зависимости от типа вводимых данных: символьные данные заключаются в кавычки, даты – в фигурные скобки, логические – в точки, числовые данные вводятся без разделителей.

Пример использования оператора INSERT:

INSERT INTO Издательства VALUES (6, "Новое знание", "Минск");

В данном примере значения вводятся во все столбцы таблицы, поэтому <список столбцов> не указан.

Если значения, которые необходимо ввести, являются результатом выполнения запроса, то эти значения также помещаются в специфицированные колонки и должны соответствовать им по типу. При использовании <подзапроса> в указанную таблицу вводятся данные, отобранные из другой таблицы (или даже нескольких таблиц).

Командой, позволяющей корректировать содержание таблицы, является оператор UPDATE, имеющий следующий формат:

UPDATE <имя таблицы> SET <имя столбца> = <новое значение> [,

<имя столбца> = <новое значение> ...] [<предложение WHERE>];

Используя оператор UPDATE, можно изменить значения указанного столбца для всех записей таблицы, если предложение WHERE не задано, или для записей, удовлетворяющих условию запроса, если используется предложение WHERE.

Оператор UPDATE Книги SET Стоимость=Стоимость*1.1; увеличивает стоимость книг для всех записей в таблице Книги на 10 %.

Оператор UPDATE Книги SET Стоимость=Стоимость*0.9 WHERE [Год издания] < 2010; уменьшает стоимость книг, изданных до 2010 года, на 10 %.

Оператор DELETE можно использовать для удаления строк таблицы: DELETE FROM <имя таблицы> [<предложение WHERE>];

Следует быть осторожным при использовании оператора DELETE, поскольку, если фраза WHERE в операторе DELETE отсутствует, будут удалены все строки таблицы. То же самое произойдет, если неправильно указать условие отбора и в результате не будет отобрано ни одной строки в таблице. Оператор DELETE физически удаляет строки таблицы.

Задание

1. Просмотрите на SQL ранее созданные запросы: **Поиск книг по фамилии автора**, **Рейтинг книг**, **Список читателей с инициалами**. Чтобы увидеть, как выглядит запрос в MS Access на SQL, надо в окне базы данных во вкладке **Запросы** выделить имя запроса, нажать на кнопку **Открыть** или **Конструктор**, а затем в меню **Вид** выбрать команду **Режим SQL**. Например, запрос **Книги с ключевым словом в теме** на SQL будет выглядеть так, как показано на рисунке 1.

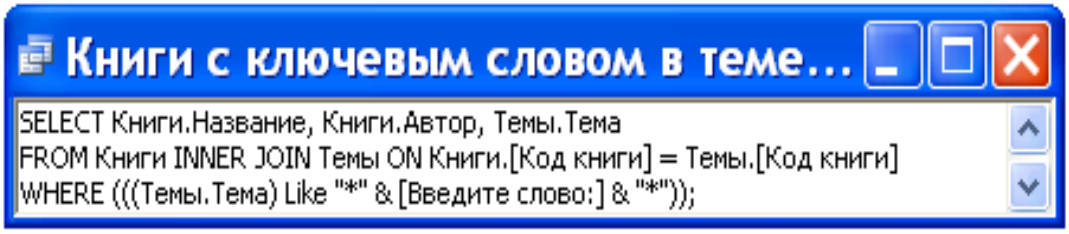


Рис. 1. Окно с текстом запроса на SQL

2. Чтобы приступить к созданию запроса на SQL, надо открыть вкладку **Запросы** окна базы данных, выполнить двойной щелчок мышью на команде **Создание запроса в режиме конструктора**, закрыть диалоговое окно **Добавление таблицы** и в меню **Вид** выбрать команду **Режим SQL**. Создайте на SQL запрос, который будет выводить о читателях, заказавших

книгу **Язык Ада**, следующую информацию: Фамилию, Имя, Отчество, Домашний адрес. Запрос назовите **Поиск читателей по заказанной книге**.

3. Сформулируйте задачу, которую решает следующий запрос на SQL:

```
SELECT Книги.*
FROM Книги
WHERE Название Like "Я"&"*";
```

4. Выясните, что делает приведенный ниже запрос на SQL.

```
SELECT Автор, Название, Наименование, Город, [Год издания]
FROM Издательства,  Книги
WHERE Издательства.[Код издательства]=Книги.[Код издательства];
```

Назовите этот запрос так, чтобы было ясно, что он делает.

5. Выполните следующий запрос на SQL:

```
SELECT Count(*) AS Количество
FROM Книги;
```

и укажите, что он делает. Дайте ему соответствующее имя.

6. Выясните назначение приведенных ниже двух запросов на SQL.

```
SELECT Читатели.[Код читателя], Читатели.Фамилия,
(SELECT COUNT ([Выдача книг].[Код книги])
FROM [Выдача книг]
WHERE [Выдача книг].[Код читателя]=Читатели.[Код читателя]) AS Количество
FROM Читатели;
```

```
SELECT Читатели.Фамилия, Count([Выдача книг].[Код книги]) AS Количество
FROM [Выдача книг], Читатели
WHERE ([Выдача книг].[Код читателя])=Читатели.[Код читателя]
GROUP BY Читатели.Фамилия;
```

7. Составьте на SQL запрос, который будет вычислять количество прочитанных каждым читателем страниц. Попытайтесь это сделать двумя способами, показанными в предыдущем задании.

Лабораторная работа № 9

Представление данных в виде форм

Цель работы: сформировать умения создавать формы для представления данных.

Форма, так же как таблица и запрос, может использоваться в MS Access для ввода информации в базу данных и для ее обработки. Более того, если база данных постоянно пополняется новыми записями или информация базы данных часто изменяется, удобней использовать форму. При создании формы можно указать, какие поля и в какой последовательности должны быть в ней представлены, разбить поля на логически связанные группы, задать удобное расположение на экране. Любая форма создается на основе таблиц и запросов. Данные из одной таблицы могут быть представлены в нескольких формах, и, наоборот, в одной форме могут содержаться данные из различных таблиц и запросов. Кроме того, формы могут содержать иллюстрации, графически представлять хранящуюся в базе данных информацию. Таким образом, формы позволяют создать удобный пользовательский интерфейс для работы с данными.

Конечно, создание форм требует дополнительных усилий. Однако потраченное время будет возмещено за счет уменьшения ошибок при вводе, удобства доступа к хранящейся в базе данных информации, наглядности ее представления и облегчения восприятия. Кроме того, форма может служить защитой базы данных от действий неквалифицированных пользователей.

Так же, как и в случае таблиц или запросов, самым простым способом создания форм является использование мастера. Мастер задает пользователю вопросы о структуре и оформлении формы, предлагая на выбор несколько вариантов. В процессе создания формы можно вернуться на несколько шагов назад, чтобы изменить принятые решения или выбрать другой вариант. В результате такого диалога появляется готовая к применению форма.

Для создания формы с помощью мастера необходимо выполнить следующие действия:

– В окне базы данных раскройте вкладку **Форма** или выберите команду **Объекты базы данных | Формы** в меню **Вид.**

– Нажмите кнопку **Создать** в окне базы данных или в меню **Вставка** выберите команду **Форма**. На экране появится диалоговое окно **Новая форма** (см. рисунок 1).

– В поле ввода с раскрывающимся списком можно ввести имя таблицы или запроса, данные из которых необходимо представить в форме. Это

целесообразно сделать лишь в том случае, если в форме будут использоваться данные из одной таблицы или одного запроса. В противном случае источник данных в этом окне можно не задавать. Нажмите кнопку **Мастер форм**, а затем – кнопку **ОК.**

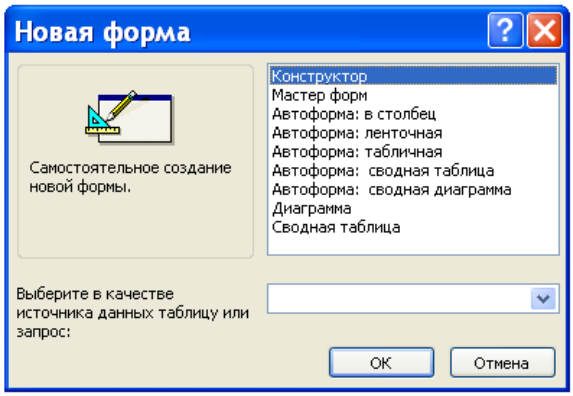


Рис. 1. Диалоговое окно **Новая форма**

После этих действий на экране появится диалоговое окно **Создание форм**, показанное на рисунке 2. Это же окно можно получить на экране более быстрым способом – двойным щелчком мыши на команде **Создание формы с помощью мастера** во вкладке **Формы** окна базы данных.

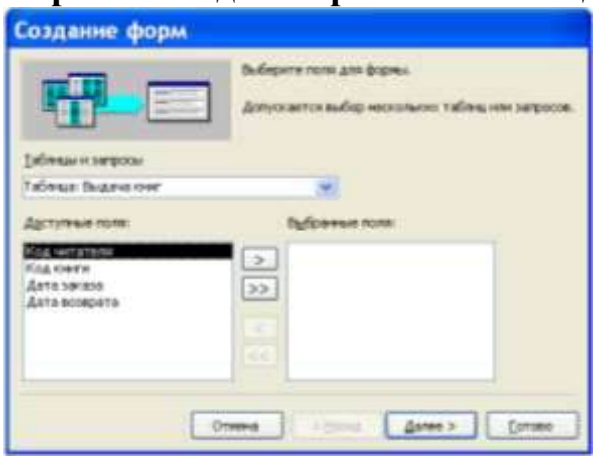


Рис. 2. Диалоговое окно **Мастер форм**

– В этом окне из списка **Таблицы и запросы** выберите требуемые имена, а затем в нижней части окна из левого столбца перенесите требуемые поля в правый столбец, используя для переноса кнопки с символами >, >>. Копки <, << используются для отмены переноса полей.

– После нажатия кнопки **Далее** появится диалоговое окно, в котором надо выбрать вид представления формы. Допустимы три вида представления: одиночная форма, подчиненная форма или связанная форма. Выберите требуемый вид представления формы.

– Для перехода к следующему окну нажмите кнопку **Далее**. В этом окне выберите внешний вид формы: в один столбец, ленточную, табличную или выровненную форму.

– После этого действуйте в соответствии с инструкциями, приведенными в ряде диалоговых окон.

Процесс построения формы рассмотрим на следующем примере. Требуется создать форму, которая позволит просматривать содержание книг.

В окне базы данных **Библиотека** во вкладке **Формы** выполните двойной щелчок мышью на команде **Создание формы с помощью мастера**.

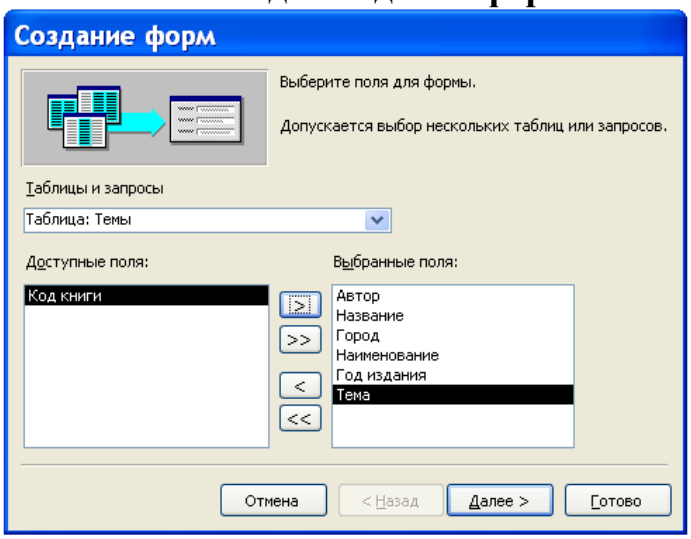


Рис. 3. Диалоговое окно **Создание форм** с выбранными полями

В появившемся диалоговом окне (см. рисунок 3) вначале выберите таблицу **Книги** и из нее в правый столбец перенесите поля **Автор** и **Название**. После этого выберите таблицу **Издательства** для переноса полей **Город** и **Наименование**. Поступая таким образом, в правый столбец перенесите поля **Год издания** и **Тема**. Результат выполнения этих действий мы видим на рисунке 3. Нажмите кнопку **Далее**.

Следующее диалоговое окно предназначено для выбора вида представления данных. Оно показано на рис. 4. В нем надо определить таблицу для подчиненной формы. Подчиненная форма в нашем случае должна базироваться на таблице **Темы**. Для этого выполните щелчок мышью на таблице **Книги** в левой части окна и, если кнопка **Подчиненные формы** не активна, выполните на ней щелчок мышью. Затем нажмите кнопку **Далее**.

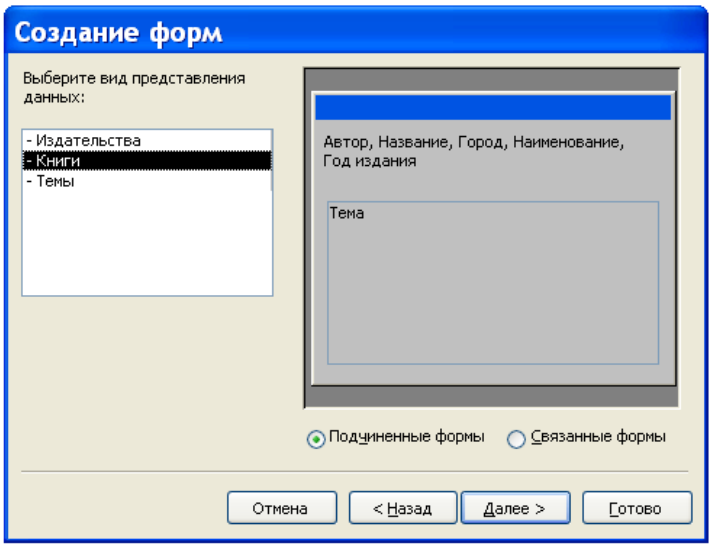


Рис. 4. Диалоговое окно для выбора вида представления данных

В следующем диалоговом окне (см. рисунок 5) для определения внешнего вида подчиненной формы нажмите кнопку **табличный**, а затем кнопку – **Далее**.

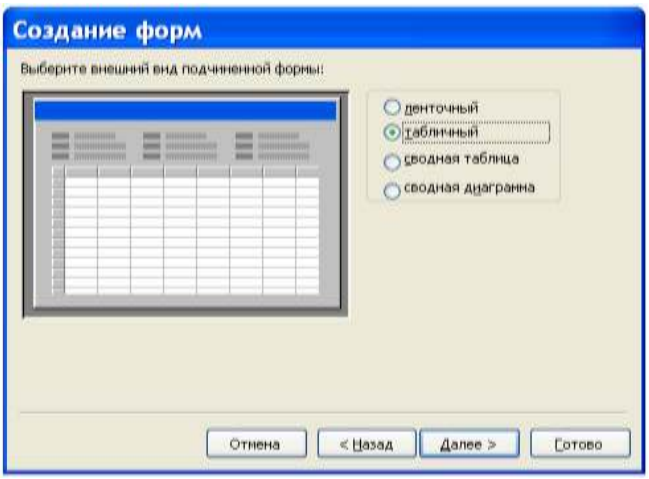


Рис. 5. Диалоговое окно для выбора внешнего вида подчиненной формы

Диалоговое окно, показанное на рисунке 6, предназначено для задания стиля формы. Выберите стиль формы – **Камень** и нажмите на кнопку **Далее**.

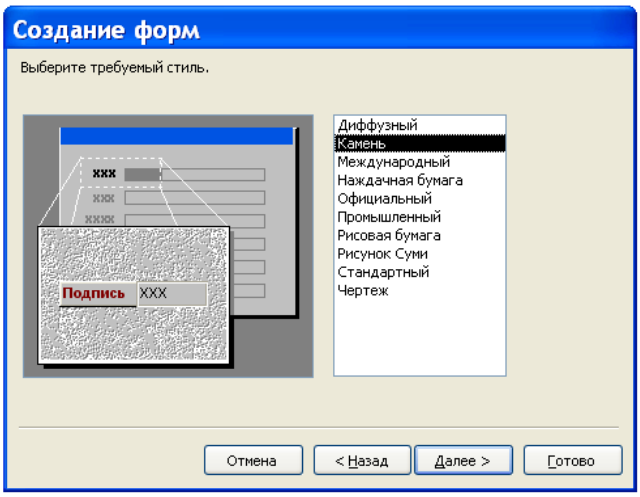


Рис. 6. Диалоговое окно для выбора стиля

Последнее диалоговое окно при создании формы с помощью мастера представлено на рисунке 7. В нем укажите имя формы **Содержание книг** и имя подчиненной формы **Подчиненная форма Темы**. Поскольку мы еще не знаем, как изменить макет формы, то сделайте активной кнопку (если она не активна) **Открыть форму для просмотра и ввода данных**, а затем нажмите на кнопку **Готово**.

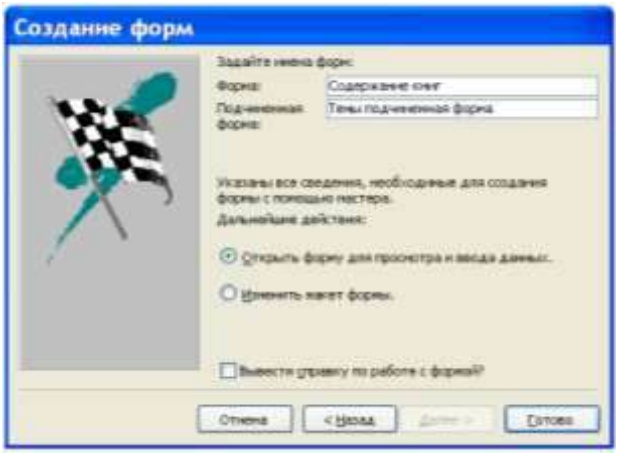


Рис. 7. Последнее диалоговое окно при создании формы

В результате выполнения описанных действий откроется приведенная на рисунке 8 форма **Содержание книг**. Безусловно, эта форма в том виде, как мы ее создали, обладает рядом недостатков, которые можно устранить при выполнении ее редактирования. Важно отметить, что поставленную задачу она все же решает – достаточно удобно позволяет просматривать содержание книг.

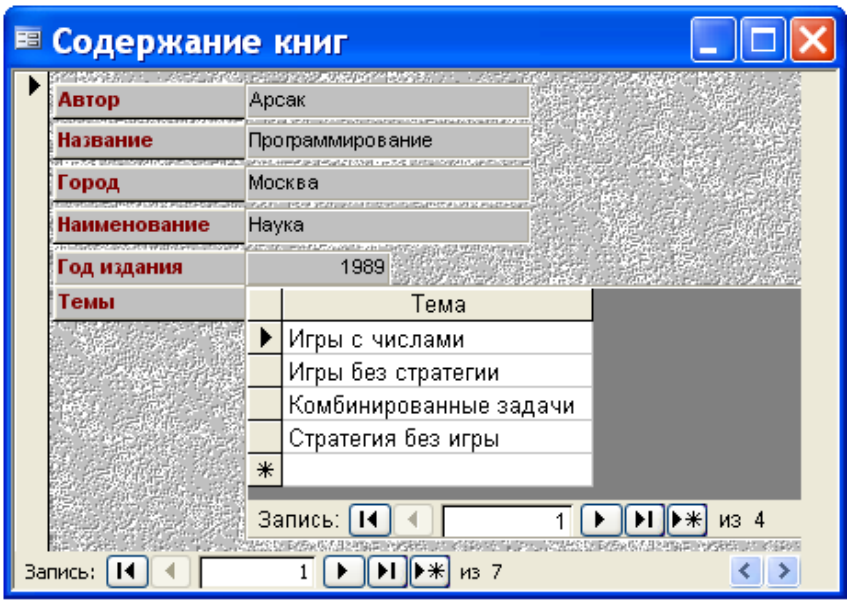


Рис. 8. Составная форма **Содержание книг**

Каждая форма в MS Access может быть представлена на экране в

одном из четырех видов: в основном режиме работы с формой, в табличном режиме, в режиме конструирования и в режиме предварительного просмотра.

Задание

1. Для запроса **Стоимость книг с учетом инфляции** создайте автоформу в столбец. Напомним, что для создания автоформ надо выполнить следующие действия. В окне базы данных во вкладке **Формы** надо выполнить щелчок мышью на команде **Создать**, а затем в появившемся диалоговом окне **Новая форма** выбрать в качестве источника данных требуемую таблицу или запрос, далее выбрать соответствующую автоформу и нажать кнопку **ОК**. Форму для нашего задания назовите **Автоформа в столбец**.

2. Выясните, что вам напоминает форма, приведенная на рисунке 9. Какую функцию она автоматизирует?

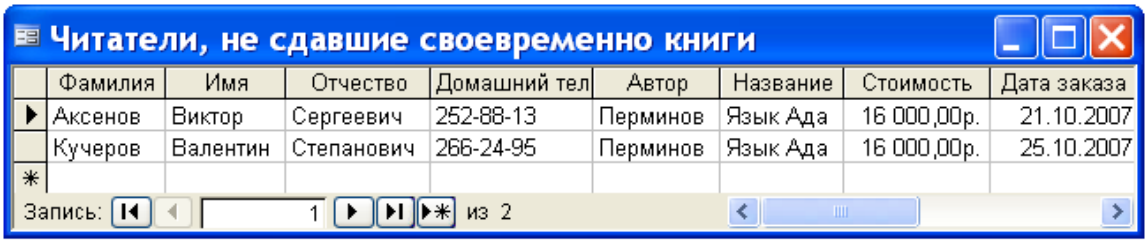


Рис. 9. Табличная автоформа

Создайте эту автоформу и самостоятельно дайте ей название.

3. Создайте форму, приведенную на рисунке 10. Самостоятельно выясните, какую таблицу или какой запрос для этой формы надо взять в качестве источника данных. Определите, какой стиль выбран для создания этой формы.

Созданную вами форму назовите **Ленточная автоформа**. Перечислите недостатки, которые вы в форме обнаружили. Каким способом можно изменить макет **Ленточной автоформы**?

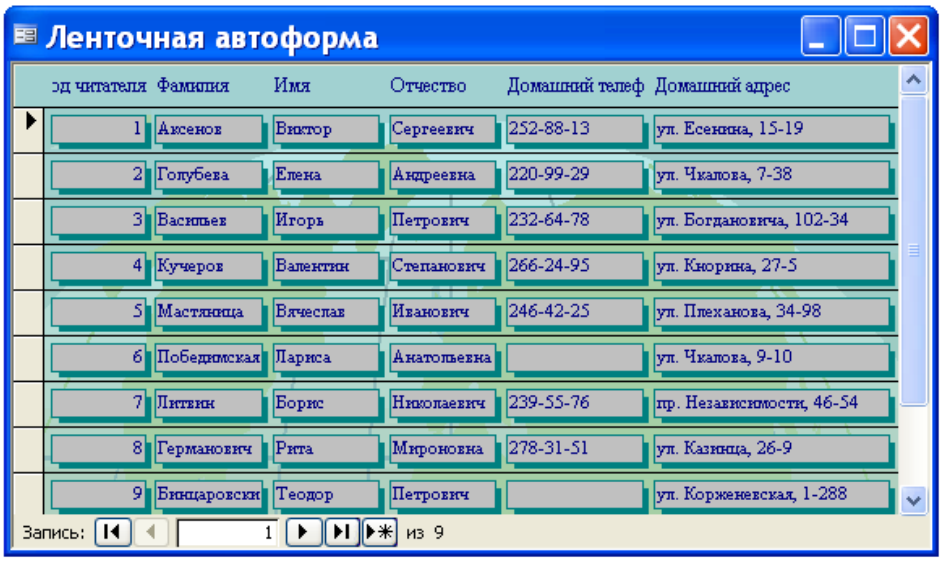


Рис. 10. Пример ленточной автоформы

4. MS Access имеет возможность представлять числовую информацию из базы данных в графической форме. Эту форму представления числовой информации называют диаграммой. Создайте диаграмму, приведенную на рисунке 11.

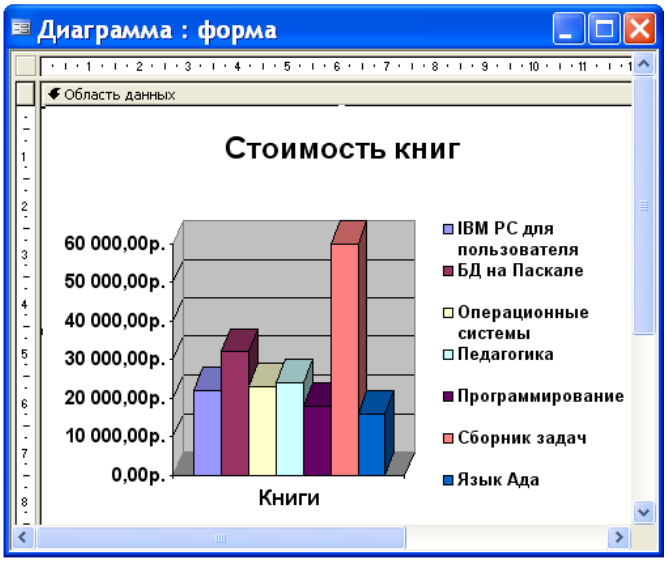


Рис. 11. Графическое представление стоимости книг

Возможности MS Access для построения и форматирования диаграмм являются очень ограниченными. Поэтому дополнительную обработку диаграммы можно выполнить с помощью приложения MS Graph. Вызвать его можно в режиме конструирования формы двойным щелчком мыши в области диаграммы.

Лабораторная работа № 10

Обработка данных с помощью отчетов

Цель работы: сформировать умения для представления данных в виде отчетов.

Наилучшим средством для представления данных в виде печатного документа являются отчеты. Отчет предоставляет возможность наглядно представить извлеченную из базы данных информацию, дополнив ее результатами анализа и вычислений. В них можно отобразить данные в виде диаграммы или графика, использовать другие средства оформления. Отчеты очень похожи на формы, однако между ними имеется существенное различие – отчеты предназначены исключительно для вывода данных на печать. Поэтому в них можно отказаться от элементов управления, предназначенных для ввода данных: списков, полей ввода со списком, флажков, переключателей и т. п.

Сконструируем отчет, который будет выводить имеющуюся в базе данных информацию о книгах, их общем количестве и средней стоимости по каждому издательству. Для этого нам понадобятся данные из двух таблиц: **Издательства** и **Книги**. Поэтому перед созданием отчета необходимо сначала создать запрос.

В окне базы данных перейдите на вкладку **Запрос**, нажмите кнопку **Создать** и выберите вариант **Новый запрос**. В диалоговом окне **Добавление таблицы** выберите **Издательства** и **Книги**. В область конструктора запроса перенесите поля **Название**, **Автор**, **Объем**, **Год издания**, **Стоимость** таблицы **Книги** и поле **Наименование** таблицы **Издательства**. Сохраните запрос с именем **Каталог**.

Теперь можно начать создание отчета. Раскройте вкладку **Отчеты** в окне базы данных **Библиотека** и нажмите кнопку **Создать**. В появившемся диалоговом окне **Новый отчет** в поле ввода с раскрывающимся списком для выбора источника данных введите имя только что созданного запроса. Для создания отчета выберите команду **Конструктор** и нажмите кнопку **ОК** (см. рисунок 1).

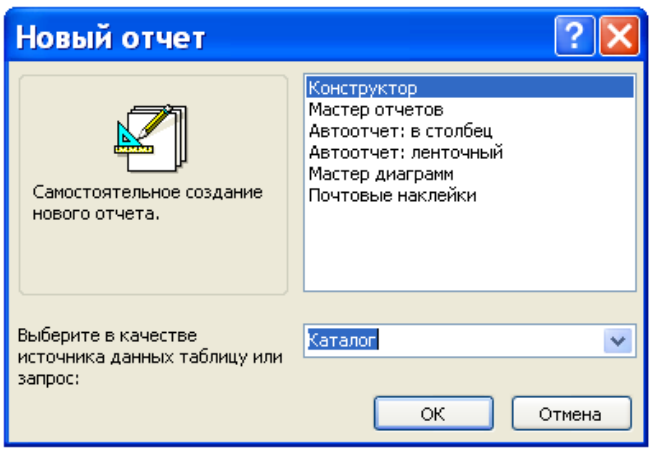


Рис. 1. Диалоговое окно **Новый отчет**.

MS Access выведет на экран пустой бланк отчета с разделами **Верхний колонтитул**, **Нижний колонтитул**, в центре между которыми находится **Область данных**. Этот бланк показан на рисунке 2.

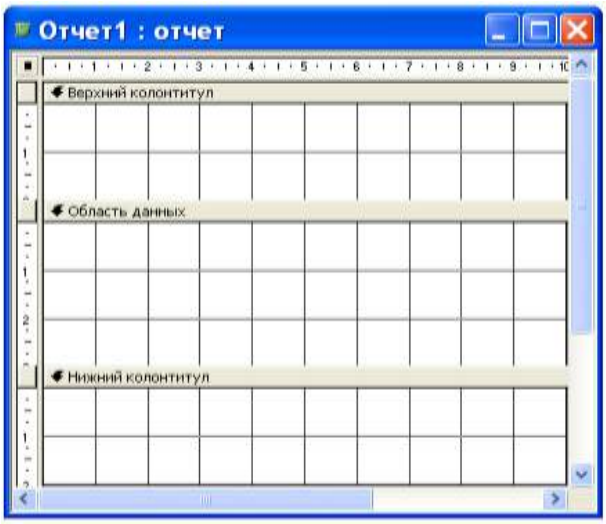


Рис. 2. Пустой бланк для создания отчета.

Линейки с делениями по верхнему и левому краям отчета помогут Вам спланировать расположение данных на странице. Если Вы хотите сделать по бокам печатной страницы поля по 2 см, то вы должны располагать предназначенные для печати элементы отчета в области шириной до 17 см для страницы стандартного размера 210x297 мм. Расположение данных на странице по вертикали определяется тем, как вы зададите верхний и нижний колонтитулы (верхнее и нижнее поля страницы). Как и при работе с формами, вы можете перетащить край любого раздела, чтобы сделать его больше или меньше. Заметьте, что ширина всех разделов должна быть одной и той же, поэтому если вы измените ширину одного из разделов, то MS Access автоматически изменит ширину и всех других разделов.

Отчет, в отличие от формы, предоставляет пользователю возможность задать группировку данных прямо в отчете с помощью команды **Сортировка и группировка** меню **Вид**. В диалоговом окне можно определить до 10 полей или выражений, которые будут использоваться в отчете для группировки данных. Первый элемент в списке определяет основную группу, а последующие элементы – подгруппы внутри группы предыдущего уровня.

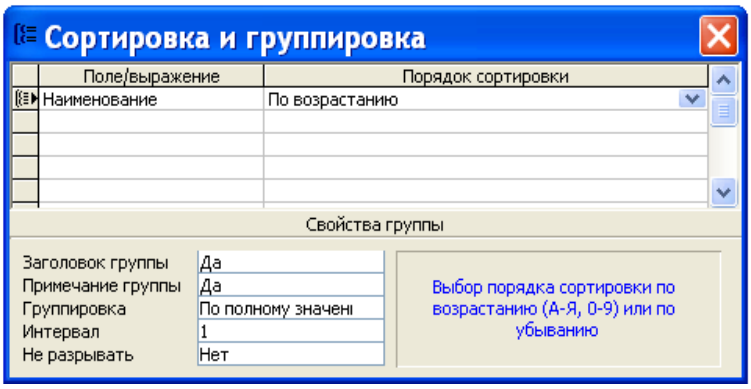


Рис. 3. Диалоговое окно **Сортировка и группировка**

Чтобы подсчитать общее количество имеющихся в настоящее время книг и среднюю стоимость книги по каждому издательству, необходимо сгруппировать данные по полю **Наименование**. Для этого в меню **Вид** выберите команду **Сортировка и группировка.** В появившемся диалоговом окне **Сортировка и группировка** (см. рисунок 3) щелкните по первой строке в столбце **Поле/выражение** и раскройте кнопку раскрывающегося списка. Выберите поле **Наименование** (вы можете также использовать выражение, содержащее ссылку на любое поле таблицы или запроса). По умолчанию MS Access сортирует значения по возрастанию. В отчете должно быть зарезервировано место, куда вы поместите заголовок для каждой группы и примечание для вычисляемых полей (общего количества и среднего значения). Чтобы добавить эти разделы, установите для свойств **Заголовок группы** и **Примечание группы** значения **Да**. MS Access добавит в отчет требуемые разделы. Закройте окно сортировки и группировки.

Теперь вы можете закончить построение отчета, выполнив приведенные ниже действия (результат ваших действий должен выглядеть так, как показано на рисунке 4).

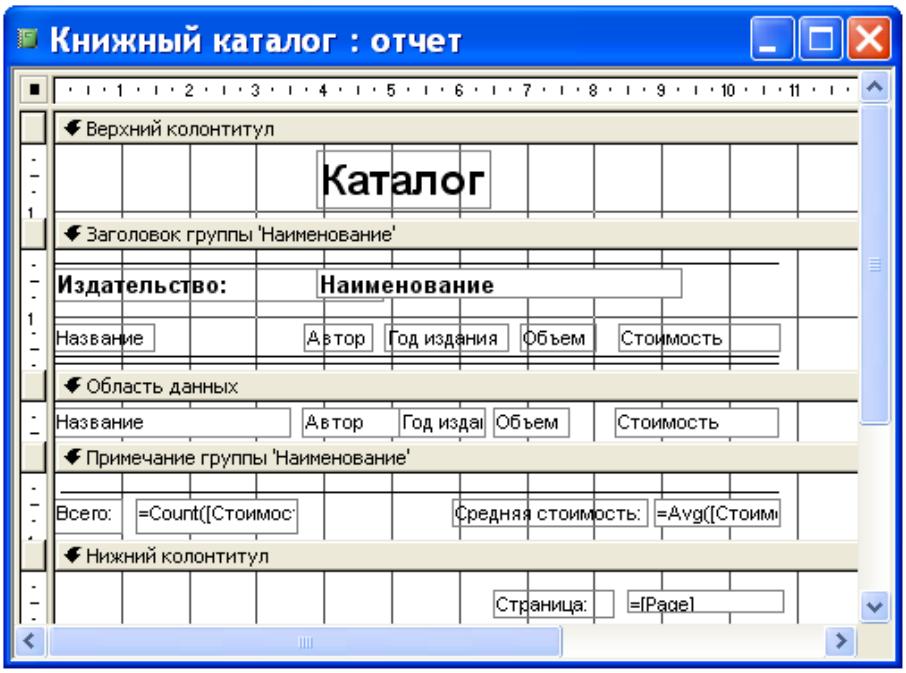


Рис. 4. Отчет **Каталог** в режиме конструктора.

– Разместите несвязанную подпись в области верхнего колонтитула в введите в нее **Каталог**. Выделите эту подпись и установите для нее полужирный шрифт Arial Cyr размером 18 пунктов. Чтобы привести размер элемента управления в соответствие с назначенным шрифтом, выполните команду **Размер | по размеру данных** в меню **Формат**.

– Раскройте список полей и перетащите поле **Наименование** в область заголовка группы. Для подписи и элемента управления установите полужирный шрифт Arial Cyr размером 10 пунктов. Измените текст подписи **Наименование** на **Издательство**.

– В заголовок группы необходимо поместить в качестве заголовков столбцов несколько несвязанных подписей. Для этого увеличьте область данных и перетащите поля **Название**, **Автор**, **Год издания**, **Объем** и **Стоимость** из списка полей в область данных. Выделите подпись поля **Название** и выполните команду **Вырезать** меню **Правка**, чтобы поместить ее в буфер обмена. Выделите заголовок группы и выполните команду **Вставить** меню **Правка**. Подпись будет вставлена в левый верхний угол области заголовка группы. Теперь она стала несвязанной и ее можно разместить независимо от поля. Аналогичным образом отделите остальные подписи и переместите их в заголовок группы.

– Разместите подписи столбцов в заголовке группы таким образом, чтобы они располагались точно над соответствующими элементами управления. После этого выделите все подписи и выполните команду **Выровнять | по верхнему краю** в меню **Формат**, чтобы выровнять эти подписи по горизонтали.

– Чтобы улучшить внешний вид отчета, с помощью инструмента **Линия** проведите линию вдоль верхней границы заголовка группы и две линии вдоль нижней границы заголовка группы.

– Размер области данных определяет расстояние между строками в отчете. Нам не требуется, чтобы между строками отчета было какое-то расстояние. Поэтому уменьшите размер области данных до высоты размещенных в ней полей.

– Проведите линию вдоль верхней границы примечания группы **Наименование** и разместите под ней два несвязанных поля.

– Измените подпись первого поля на **Всего:**. Откройте бланк свойств элемента управления и в качестве значения свойства **Данные** введите выражение **=Count([Название])**. С помощью этой функции вычисляется количество записей внутри группы, т.е. общее количество книг данного издательства. Поместите это поле под полем **Название**.

– Измените значение подписи второго поля на текст **Средняя стоимость:**. В качестве значения свойства **Данные** элемента управления

введите выражение **=Avg([Стоимость])**. По данной формуле вычисляется среднее значение стоимости внутри группы. Чтобы вместе с числовым значением выводился денежный знак, для свойства **Формат поля** установите значение **Денежный**. Поместите это поле под полем **Стоимость**.

– В правый нижний угол нижнего колонтитула поместите несвязанное поле. Измените подпись этого поля на текст **Страница**, а для свойства **Данные** элемента управления введите выражение =[Page].

Иногда отчет должен содержать информацию, относящуюся к заголовку отчета или его примечанию. Увидеть заголовок отчета и его примечание в режиме конструктора можно, выполнив команду **Заголовок/примечание** отчета в меню **Вид**.

Чтобы просмотреть результаты работы, выберите команду **Предварительный просмотр** в меню **Файл** (см. рисунок 5). Сохраните созданный отчет под именем **Книжный каталог**.

Рис. 5. Созданный отчет в режиме предварительного просмотра

Познакомимся более детально со структурой макета отчета. Макет отчета разделен на несколько областей, каждая из которых имеет свое назначение и особенности (см. рисунок 6). У верхнего края окна отчета расположена область заголовка. Заголовок появляется в сформированном отчете только один раз – в начале отчета – и будет расположен на первой странице. В нем обычно размещают подпись, а также выражение =Now(). В режиме предварительного просмотра на месте этого выражения будет помещена текущая дата. Благодаря этой функции в заголовке всегда присутствует информация о том, насколько свежи данные отчета. Ниже расположен верхний колонтитул, который в режиме просмотра вы увидите вверху каждой страницы отчета. Если отчет имеет табличную структуру, то в верхнем колонтитуле обычно располагаются имена полей.

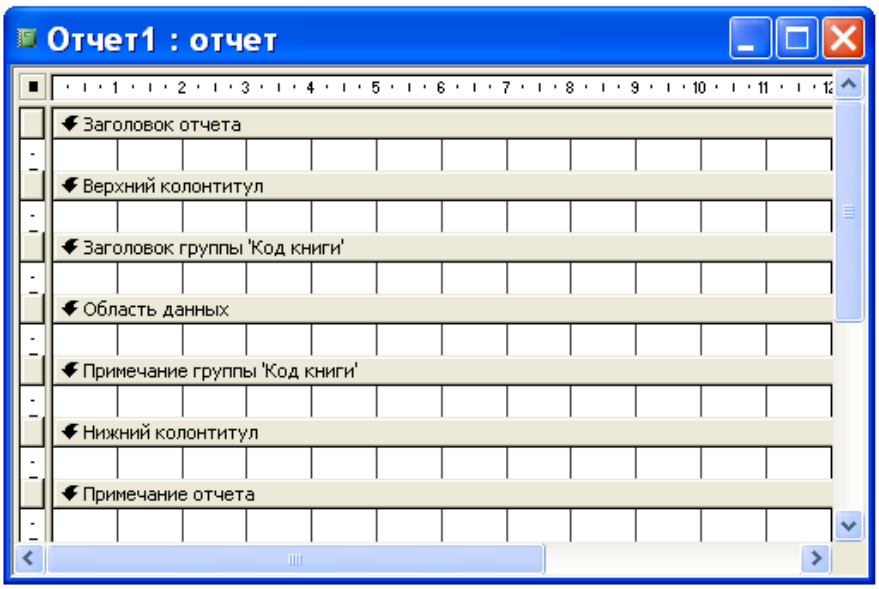


Рис. 6. Макет отчета.

Далее расположен заголовок группы, который MS Access печатает для каждой группы данных. В случае группировки по нескольким полям в макете отчета будут присутствовать несколько заголовков групп. Ниже заголовка группы расположена область данных, содержащая основные, детальные данные для отчета. В сформированном отчете каждая группа будет представлена детальными данными для каждой записи данных из этой группы.

Далее расположено примечание группы, которое появляется в конце каждой группы записей детальных данных. Оно используется для указания промежуточных итогов по группе. Ниже расположен нижний колонтитул, который появляется в конце каждой страницы отчета. Мастер по разработке отчетов вносит в нижний колонтитул функцию =[Page]. Если отчет займет несколько страниц, то благодаря этой функции они будут пронумерованы автоматически.

И последняя область – примечание отчета. Оно появляется только один раз в конце отчета и содержит общие итоги по всем суммируемым полям. И хотя примечание отчета является последней областью макета отчета, оно появляется в сформированном отчете перед нижним колонтитулом последней страницы.

Лабораторная работа № 11

Использование мастера отчетов

Цель работы: сформировать умения для создания отчетов с помощью мастера.

Простейший путь создания отчета состоит в использовании мастера отчетов. Для создания отчета с помощью мастера необходимо выполнить следующие действия:

– В окне базы данных раскройте вкладку **Отчет** или выберите команду **Отчет** в меню **Вид**.

– Выполните двойной щелчок мышью на команде **Создание отчета с помощью мастера**. На экране появится диалоговое окно **Создание отчетов** (см. рисунок 1).

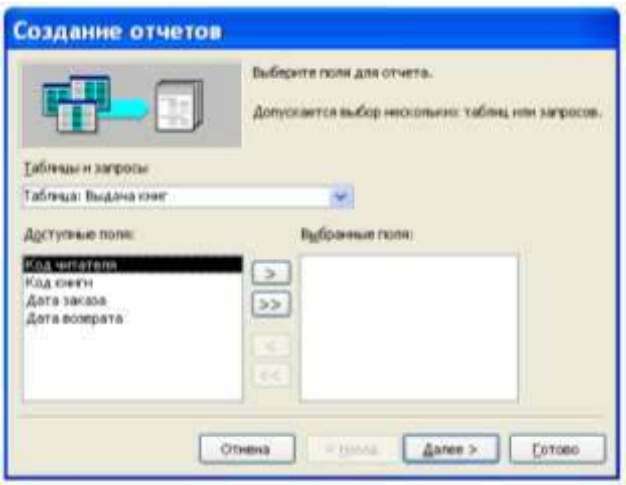


Рис. 1. Диалоговое окно **Создание отчетов**

– В этом окне из списка **Таблицы и запросы** выберите требуемые имена, а затем в нижней части окна из левого столбца перенесите требуемые поля в правый столбец, используя для переноса кнопки с символами >, >>. Кнопки <, << используются для отмены переноса полей. Выбирать можно несколько таблиц и запросов, при этом в той очередности, которая бы обеспечила требуемую очередность расположения полей в отчете.

– Нажимая кнопку **Далее,** вы будете получать очередное диалоговое окно, в котором надо действовать в соответствии с приведенными инструкциями. Нажатие кнопки **Готово** приведет к завершению построения отчета. Так, например, если вы в диалоговом окне, приведенном на рисунке 1, в качестве источника данных выберите таблицу **Выдача книг**, а затем с помощью кнопки >> перенесете все поля из этой таблицы в отчет и

нажмете кнопку **Готово**, то сразу получите отчет, приведенный на рисунке 2. Автоматически этому отчету будет присвоено имя **Выдача книг**.

Мы уже познакомились с двумя способами создания отчетов: в режиме конструктора и с помощью конструктора. Рассмотрим другие возможности создания отчетов.

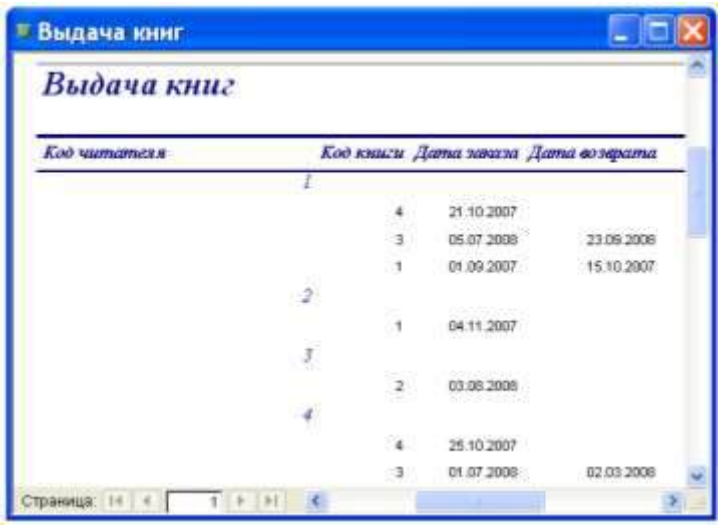


Рис. 2. Отчет **Выдача книг**

MS Access позволяет создавать отчет в один столбец автоматически. Созданный таким способом отчет называют автоотчетом в столбец. Для создания такого отчета поступают следующим образом. Во вкладке **Отчеты** базы данных нажимают кнопку **Создать**. В появившемся диалоговом окне **Новый отчет** выбирают команду **Автоотчет: в столбец**, задают источник данных (в нашем примере в качестве источника данных выбрана таблица **Издательства**) и нажимают кнопку **ОК**.

Вид полученного при этом отчета очень похож на форму, выводящую данные в один столбец: каждое поле расположено в отдельной строке, а записи данных строго друг под другом (см. рисунок 3). Обратите внимание на то, что в строке заголовка окна отчета автоматически появилось имя отчета, совпадающее с именем источника данных – таблицы **Издательства**. При закрытии отчета мы можем дать ему требуемое имя. Назовите этот отчет в один столбец **Списком издательств**.

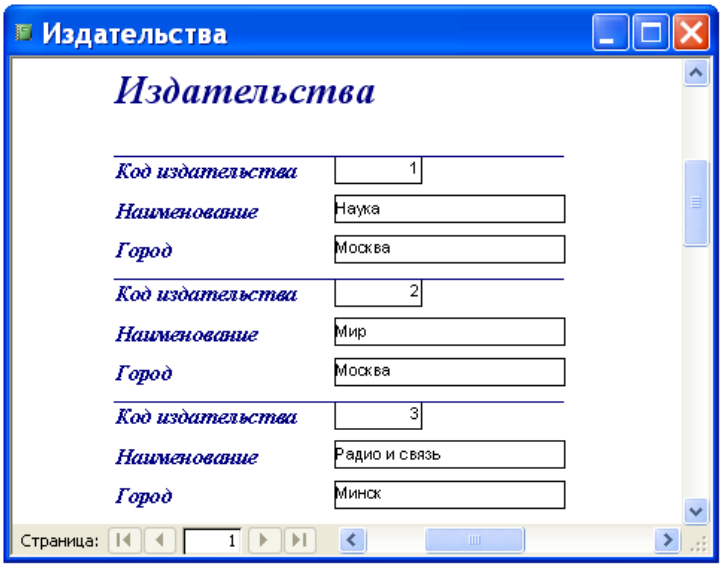


Рис. 3. Автоотчет в столбец

Автоотчет ленточный создается точно таким же образом, как и автоотчет в столбец, но вместо команды **Автоотчет: в столбец** в диалоговом окне **Новый отчет** выбирается команда **Автоотчет: ленточный**. Вид ленточного отчета соответствует виду таблицы или запроса базы данных. Поэтому не случайно в предыдущих версиях MS Access этот отчет назывался табличным. Вид ленточного отчета для таблицы **Издательства** приведен на рисунке 4. Последовательность расположения полей в ленточном отчете соответствует последовательности расположения этих полей в источнике данных.

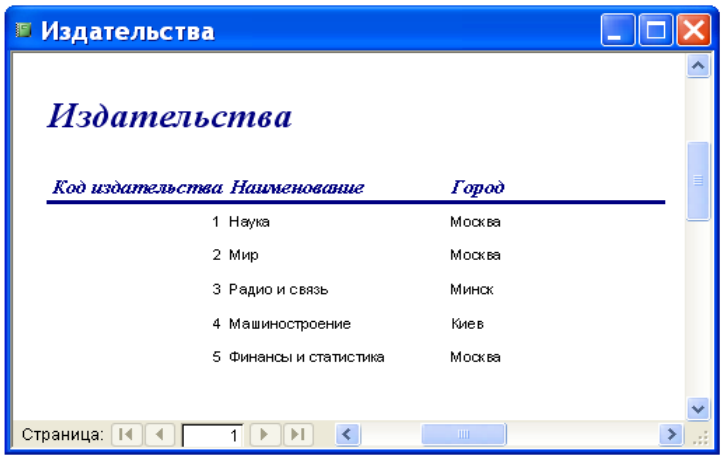


Рис. 4. Автоотчет ленточный

MS Access предоставляет удобный способ для создания почтовых наклеек на конверты при рассылке писем. Рассмотрим на примере, как можно создать почтовые наклейки для писем читателям. Для этого выполните следующие действия:

– Во вкладке **Отчеты** окна базы данных **Библиотека** нажмите на

кнопку **Создать**.

– В появившемся диалоговом окне **Новый отчет** выберите команду **Почтовые наклейки**, установите в качестве источника данных таблицу **Читатели** и нажмите кнопку **ОК**.

– В появившемся диалоговом окне **Создание наклеек** (см. рисунок 5) выберите размер наклейки со следующими параметрами: **Avery J8163; 99,0x38,1; 2**. В качестве системы единиц выберите метрическую, задайте тип наклеек – **на листах**, установите фильтр по изготовителю – **Avery** и нажмите кнопку **Далее**.

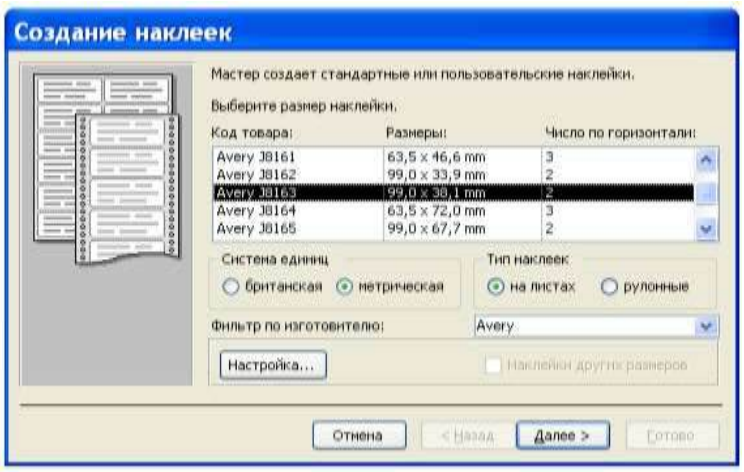


Рис. 5. Диалоговое окно **Создание наклеек**

– В следующем диалоговом окне все оставьте без изменения за исключением размера шрифта. Сделайте его равным 10 пунктам и нажмите кнопку **Далее**.

– В появившемся диалоговом окне для создания прототипа в первой строке поля прототипа наклейки наберите текст **Куда:**, сделайте пробел, выполните щелчок мышью в левой части окна на поле **Домашний адрес**, нажмите на кнопку > для переноса домашнего адреса в область прототипа наклейки. Затем нажмите на клавиатуре клавишу **Enter** и приступите к формированию второй, а после и третьей строки прототипа наклейки (см. рисунок 6). Когда прототип наклейки будет создан, нажмите кнопку **Далее**.

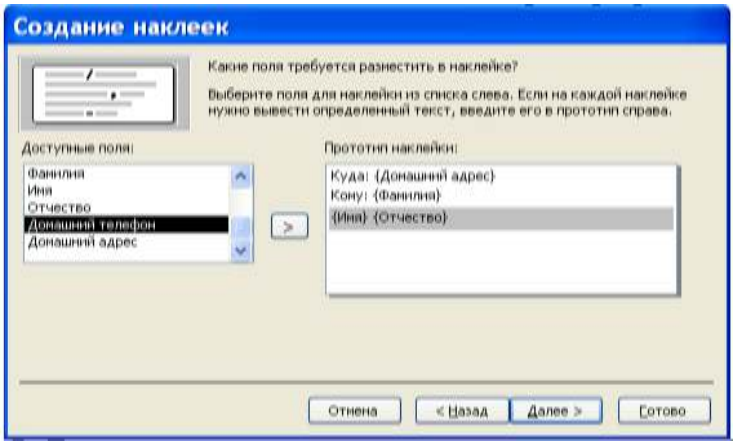


Рис. 6. Диалоговое окно для создания прототипа наклейки

– Следующее диалоговое окно предлагает возможность отсортировать наклейки по полям таблицы **Читатели**. Эта операция используется только для того, чтобы расположить наклейки на листе в определенном порядке и облегчить работу по отысканию соответствующей наклейки. Поэтому поля, используемые для сортировки, не обязательно должны присутствовать в тексте наклейки. Отсортируйте наклейки по полям: **Фамилия**, **Имя**, **Отчество** и нажмите на кнопку **Далее**.

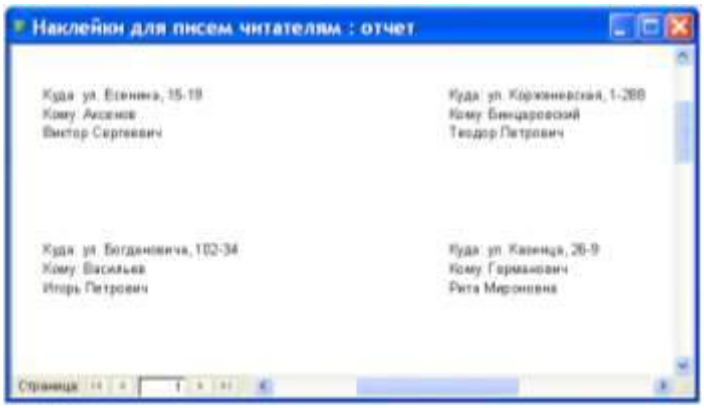


Рис. 7. Отчет с наклейками для писем читателям

– Заключительное диалоговое окно предлагает ввести имя отчета и определить дальнейшие действия. Дайте имя отчету **Наклейки для писем читателям** и нажмите на кнопку **Готово**. Мы увидим почтовые наклейки, представленные на рисунке 7. Для вывода почтовой наклейки на печать войдите в меню **Файл** и выберите команду **Печать**.

В рассмотренных выше случаях отчеты использовались просто для вывода на печать данных, оформленных специальным образом. Мастер отчетов предоставляет возможность для группировки записей и вычисления промежуточных итогов по числовым полям и общего итога для всех групп. В

таком отчете записи данных располагаются в строку, в табличной форме, а подписи полей находятся у верхнего края страницы.

Сформируем этот вид отчета на основании таблицы **Книги** и сгруппируем все записи по полю **Год издания** по десятилетиям. Для создания отчета во вкладке **Отчеты** окна базы данных выполните двойной щелчок мышью на команде **Создание отчета с помощью мастера**. В появившемся диалоговом окне **Создание отчетов** задайте в качестве источника данных таблицу **Книги** и выберите из нее поля: **Название**, **Автор**, **Объем**, **Год издания**, **Стоимость** для включения их в отчет, а затем нажмите кнопку **Далее** для перехода к выполнению следующего шага.

В следующем диалоговом окне необходимо определить поля, по которым надо провести группировку (выбрать можно до 4 полей). Порядок выбора полей для группировки определяет иерархию группировки: первой выполняется группировка по первому выбранному полю, затем внутри этих групп проводится группировка по второму полю и т. д. Для данного отчета в качестве поля, по которому будет производиться группировка, выберите поле **Год издания** и нажмите кнопку **Группировка**. В следующем диалоговом окне **Интервалы группировки** необходимо определить, как именно должна производиться группировка данных. Содержимое раскрывающегося списка **Группировка** зависит от типа данных в поле, по которому производится группировка. Текстовые поля могут быть сгруппированы по полному значению или по нескольким совпадающим первым символам – от одного до пяти. Поля даты и времени могут быть сгруппированы по полному значению, по годам, по кварталам, по месяцам, по неделям, дням, часам или минутам.

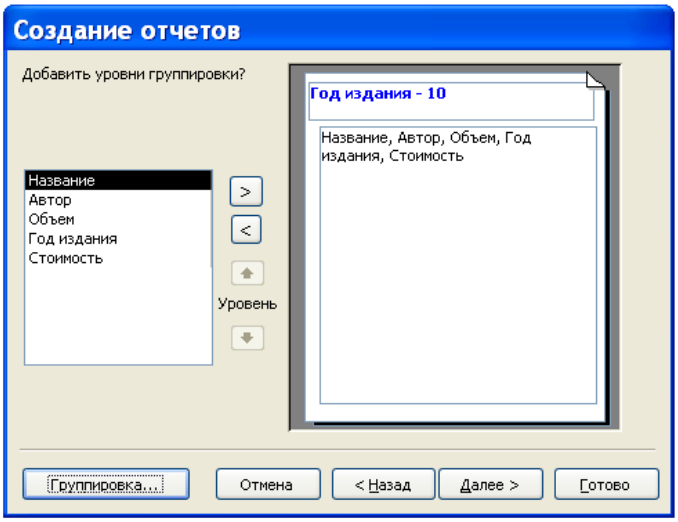


Рис. 8. Диалоговое окно **Создание отчетов** после задания группировки

Для нашего числового поля **Год издания** предоставляются следующие интервалы группирования**: обычный** (означает по полному значению)**, 10, 50, 100, 500, 1000, 5000, 10000.** Выберите интервал группирования, равный **10**, и нажмите кнопку **ОК**, чтобы вернуться в диалоговое окно **Создание отчетов**. Оно в этот момент времени будет иметь вид, приведенный на рис. 8. Нажмите в нем кнопку **Далее**.

Далее укажите поля, по которым следует произвести дополнительную сортировку в пределах соответствующих групп. Сортировка внутри групп будет также осуществлена в той последовательности полей, которую вы определите. Произведите дополнительную сортировку по полям **Название** и **Автор** и нажмите кнопку **Итоги**. В появившемся диалоговом окне **Итоги** укажите, какие итоговые значения надо вычислить. Установленный в этом окне флажок **Вычислить проценты** означает, что под каждой из итоговых сумм будет выводиться процентная доля каждой группы в общей сумме. Настройте это окно так, как показано на рисунке 9, и нажмите кнопку **ОК**. Затем в диалоговом окне **Создание отчетов** нажмите кнопку **Далее**.

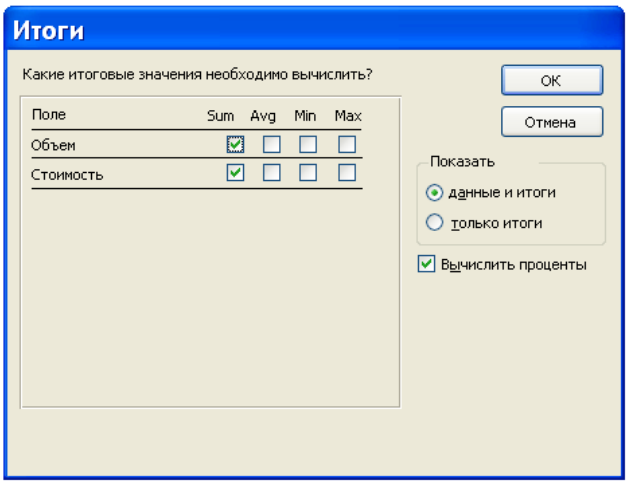


Рис. 9. Диалоговое окно для задания итоговых значений

В следующем диалоговом окне, предназначенном для определения вида макета отчета, выберите макет с названием **по левому краю 2,** снимите флажок **Настроить ширину полей для размещения на одной странице**, ориентацию сохраните книжную и нажмите кнопку **Далее**.

Следующее диалоговое окно позволяет выбрать стиль отчета. Выберите для нашего отчета стиль **Деловой** и нажмите кнопку **Далее**. В последнем диалоговом окне **Создание отчетов** задайте имя отчета **Выпуск книг по десятилетиям**, щелкните мышью по переключателю **Просмотреть отчет** и нажмите кнопку **Готово**. Вы получите отчет, представленный на рисунке 10.

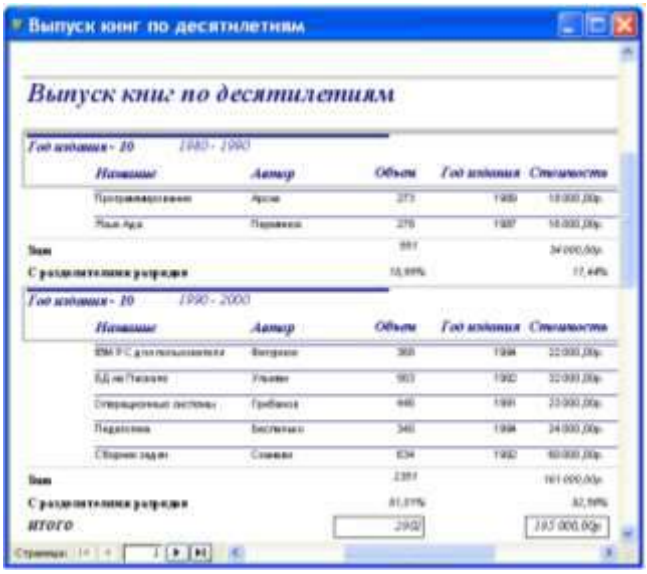


Рис. 10. Отчет с итогами

Следует заметить, что наиболее предпочтительным способом создания отчетов является комбинированный способ – вначале создается “заготовка” отчета с помощью рассмотренных выше средств, а затем в режиме конструктора выполняется доработка отчета.

Режим конструктора для отчетов почти идентичен аналогичному режиму для форм. Это также касается панели инструментов, панели элементов и средств, предназначенных для размещения и работы с элементами управления. Однако в отчетах отсутствует необходимость наличия управляющих элементов для ввода данных. Поэтому в них можно отказаться от использования списков, полей со списком, флажков и т. п.

Задание

1. В последнем созданном отчете **Выпуск книг по десятилетиям** в режиме конструктора из области данных удалите строку со значениями полей: **Название**, **Автор**, **Объем**, **Год издания** и **Стоимость**. Отчет назовите **Групповые вычисления**.

2. Отчет **Групповые вычисления**, полученный в предыдущем пункте, создайте с помощью мастера отчетов, не привлекая для этих целей режим конструктора. Дайте название отчету **Возможности мастера отчетов**.

3. Создайте отчет с наклейками для писем читателям, которые своевременно не сдали книги. Для этих целей вначале создайте запрос, который будет содержать необходимую информацию, а затем его возьмите в качестве источника данных для создания отчета. Отчет назовите **Наклейки для уведомления читателей.**

4. Создайте отчет, который будет содержать только итоговые строки, показывающие заказы книг по годам. Отчет сохраните под именем **Итоги заказов по годам**.

Лабораторная работа № 10
Использование кнопок в формах

Цель работы: сформировать умения разрабатывать кнопочную форму для автоматизации работы с приложением.

На заключительной стадии разработки базы данных для автоматизации работы с приложением разрабатывают так называемую кнопочную форму, которая позволяет запускать все процессы, выполняемые в базе данных. Как правило, такая форма является заставкой приложения и появляется на экране при открытии базы данных.

Кнопочную форму можно создать следующим образом.

1. В окне базы данных выполняют щелчок мышью на кнопке **Формы** на панели **Объекты** окна базы данных.
2. Создают форму без источника записей. Для создания формы выполняют двойной щелчок мышью на команде **Создание формы в режиме конструктора.**
3. Настраивают в режиме конструктора макет формы. Для задания высоты раздела **Область данных** щелчком правой клавиши мыши в области данных формы вызывают контекстное меню, из него выбирают команду **Свойства** и для свойства Высота устанавливают требуемое значение.

Настройка непосредственно свойств макета формы осуществляется в окне **Форма**. Напомним, что для вызова этого окна надо выполнить двойной щелчок мышью на маркере выделения формы, расположенный на пересечении горизонтальной и вертикальной линеек в окне формы в режиме конструктора. В окне **Форма** можно настраивать такие свойства макета, как **Подпись**, **Ширина**, **Кнопка контекстной справки**, **Полосы прокрутки**, **Область выделения** и другие (всего 31 свойства).

4. Размещают на макете формы кнопки. Для создания кнопок в кнопочной форме используют **Панель элементов** (см. рисунок 1). Эта панель появляется автоматически при открытии формы в режиме конструктора.

Рис. 1. **Панель элементов** с кнопками

Самый простой способ создать кнопку в кнопочной форме –

воспользоваться мастером создания кнопок. При использовании мастера создания кнопок выполняют следующее. Вначале включают щелчком мыши кнопку **Мастера** на **Панели элементов** (если она отключена, то есть не выделена цветом). После этого выполняют щелчок мышью на элементе **Кнопка** на **Панели элементов**, перемещают указатель мыши в требуемое место раздела **Область данных** и выполняют щелчок.

В появившемся диалоговом окне мастера **Создание кнопок** определяют, например категорию **Работа с формой** и действие **Открыть форму** и нажимают кнопку **Далее**. Новое диалоговое окно требует выбора формы, которую надо будет открывать нажатием данной кнопки. После нажатия кнопки **Далее** надо указать, требуется ли отбор сведений для отображения в форме, и нажать кнопку **Далее**. В следующем диалоговом окне надо указать, что необходимо разместить на кнопке: текст или рисунок и нажать кнопку **Далее**. В последнем диалоговом окне задают имя кнопке, которое будет упрощать дальнейшие ссылки на нее, и нажимают кнопку **Готово**.

5. Размещают на кнопочной форме другие элементы, позволяющие улучшить ее дизайн.

Для размещения текста в форме можно воспользоваться кнопкой **Надпись** на **Панели элементов**. Выполняют это следующим образом. Выбирают элемент **Надпись** на **Панели элементов** и в разделе **Область данных** формы этим элементом прорисовывают прямоугольник, в котором следует поместить текст. Далее требуемый текст вводят с клавиатуры**.** Когда элемент **Надпись** является активным (по его контуру размещены маленькие прямоугольники) его можно форматировать, используя кнопки панели **Формат**.

Элемент **Рисунок** на **Панели элементов** позволяет размещать изображения в разделе **Область данных** формы. После того как вы выбрали этот элемент и начертили прямоугольную область, в которую надо поместить рисунок, появляется диалоговое окно **Выбор рисунка**. В этом окне надо найти графический файл с требуемым рисунком и нажать кнопку **ОК**.

Задание

Создайте главную кнопочную форму, похожую на ту, которая показана рисунке 2. На этой форме имеются восемь кнопок: **Поступление книг**, **Месячная загрузка**, **Рейтинг книг**, **Поиск книг**, **Просмотр содержания**, **Выдача книг**, **Возврат книг**, **Письменное уведомление**, **Выход из Access**, а также текст и рисунок.

1. В верхней части окна формы в режиме конструктора наберите текст, показанный на рисунке 2. Для набора текста в форме используется кнопка **Надпись** на **Панели элементов**. Нажмите кнопку **Надпись** на **Панели элементов**. Установите указатель мыши в области данных окна формы в

режиме конструктора в том месте, где должен размещаться текст, и при нажатой левой клавише создайте рамку надписи, а затем в ней наберите текст. Отформатируйте набранный текст подходящим образом.

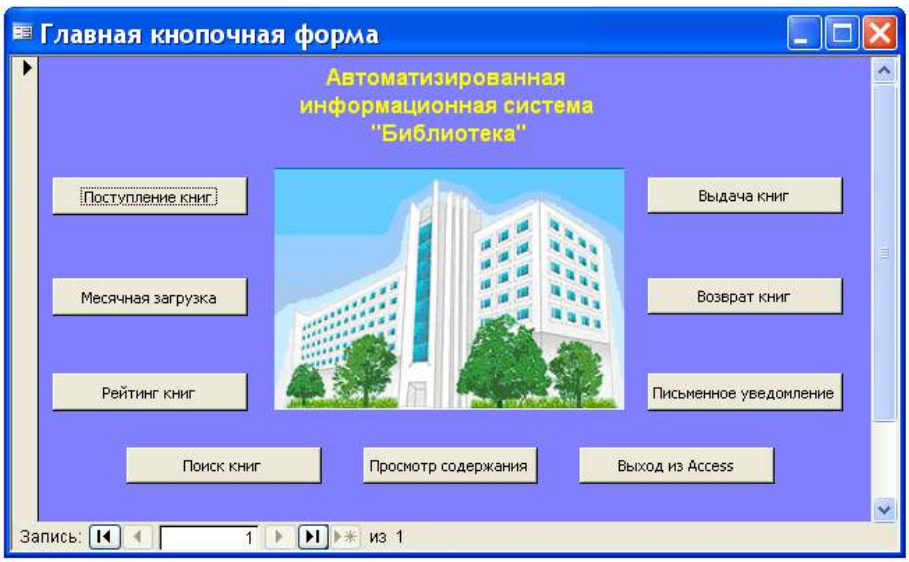


Рис. 2. Пример главной кнопочной формы

2. Вместо фотографии с изображением БГУ культуры и искусств, показанной в примере кнопочной формы, вставьте рисунок, который, на ваш взгляд, будет подходить для разрабатываемой информационной системы. Выберите элемент **Рисунок** на **Панели элементов** и начертите прямоугольную область, в которую надо поместить рисунок. В диалоговом окне **Выбор рисунка** найдите графический файл с требуемым рисунком и нажмите кнопку **ОК**.

3. Создайте запрос, который будет содержать следующие поля: **Код книги**, **Автор**, **Название**, **Код издательства**, **Наименование**, **Город**, **Объем**, **Год издания**, **Стоимость**. Запрос назовите **Поступление книг**. По данному запросу создайте автоформу в столбец, которую также назовите **Поступление книг** (см. рисунок 3).

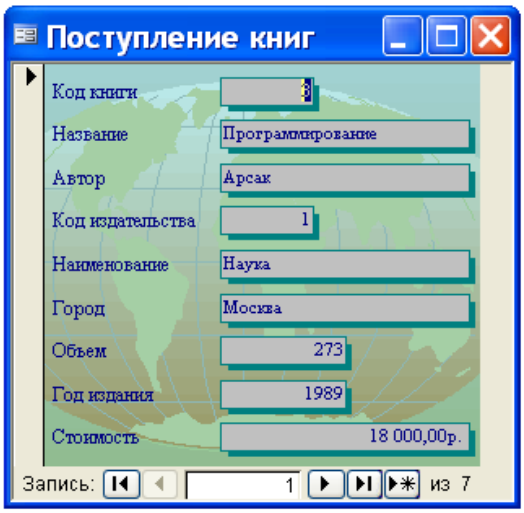


Рис. 3. Форма **Поступление книг**

Выполните щелчок мышью на элементе **Кнопка** на **Панели элементов** и изобразите прямоугольник в разделе Область данных главной кнопочной формы. В появившемся диалоговом окне мастера **Создание кнопок** выберите категорию **Работа с формой** и действие **Открыть форму**. В следующем диалоговом окне мастера кнопок укажите в качестве источника данных форму **Поступление книг**. В соответствующем диалоговом окне мастера кнопок укажите, что на кнопке надо поместить текст **Поступление книг**, и нажмите на кнопку **Готово**.

4. Для расчета количества выданных книг по месяцам ранее мы составили перекрестный запрос **Выдача книг по месяцам**. Сейчас мы свяжем этот запрос с кнопкой **Месячная загрузка**. Для этого в диалоговом окне мастера кнопок укажите категорию **Разное**, а действие – **Выполнить запрос**. При указании имени источника данных в следующем диалоговом окне укажите имя запроса **Выдача книг по месяцам**.

6. Для создания кнопки **Поиск книг** воспользуйтесь ранее созданным запросом с параметром **Поиск книг по фамилии автора**. Параметр позволяет набирать не все буквы фамилии автора, а только несколько первых букв. Напомним, что диалоговое окно для ввода значения параметра в этом запросе имело вид, показанный на рисунке 4.

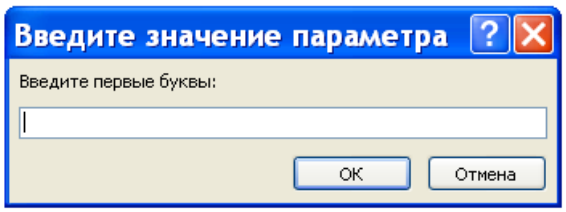


Рис. 4. Диалоговое окно для ввода значения параметра

7. Чтобы можно было просматривать содержание книг, воспользуемся ранее созданной формой **Содержание книг**. Для связывания кнопки **Просмотр содержания** с источником данных, являющимся формой **Содержание книг**, в диалоговом окне мастера кнопок выбирают категорию **Работа с формой** и действие **Открыть форму**, а в следующем диалоговом окне выбирают форму **Содержание книг**, которая будет открываться нажатием данной кнопки.

5. Кнопка **Рейтинг книг** создается подобно кнопке **Просмотр содержания**, но источником данных для нее будет не форма, а страница, которую мы ранее назвали **Books**. Поэтому в диалоговом окне мастера кнопок укажите категорию **Работа с формой**, а действие – **Открыть страницу**. При указании имени страницы выберите имя **Books**.

8. Создайте кнопку **Выдача книг,** которая в качестве источника данных будет использовать форму **Выдача книг,** содержащую поля**: Код читателя**, **Код книги**, **Дату возврата и Дату заказа**.

9. При создании кнопки **Возврат книг** вначале создайте запрос. Этот запрос должен для конкретного читателя выводить список книг, которые он

не вернул в библиотеку. Для ввода информации о читателе создайте параметр **Код читателя**. В список книг включите следующие поля: **Фамилия**, **Имя**, **Отчество**, **Автор**, **Название**, **Год издания**, **Стоимость**, **Дата заказа, Дата возврата**. Запрос назовите **Возврат книг**.

На языке SQL этот запрос будет выглядеть следующим образом:

```
SELECT Читатели.Фамилия, Читатели.Имя, Читатели.Отчество,
Книги.Автор, Книги.Название, Книги.[Год издания], Книги.Стоимость,
[Выдача книг].[Дата заказа], [Выдача книг].[Дата возврата]
FROM Читатели INNER JOIN (Книги INNER JOIN [Выдача книг]
ON Книги.[Код книги] = [Выдача книг].[Код книги]) ON Читатели.[Код
читателя] = [Выдача книг].[Код читателя]
WHERE ((([Выдача книг].[Код читателя])=[Введите Код читателя:])
AND (([Выдача книг].[Дата возврата]) Is Null));
```

После этого при создании кнопки **Возврат книги** в диалоговом окне мастера кнопок выберите категорию **Разное**, а в ней выберите действие **Выполнить запрос**. В следующем диалоговом окне в качестве источника данных укажите созданный запрос **Возврат книг**.

10. Кнопку **Письменное уведомление** создайте точно таким же образом, как и предыдущую, но в качестве источника данных для нее используйте запрос с параметром **Список читателей для вызова**. В качестве параметра в этом запросе задается количество дней, которые читатель может на руках держать книгу.

11. Кнопка **Выход из Access** предназначена для завершения работы с приложением MS Access. Для ее создания выполните следующее. В режиме конструктора форм с помощью мастера кнопок выберите категорию **Приложение**, а в ней – действие **Выйти из приложения**.

12. Создайте форму, которая приведена на рисунке 5.

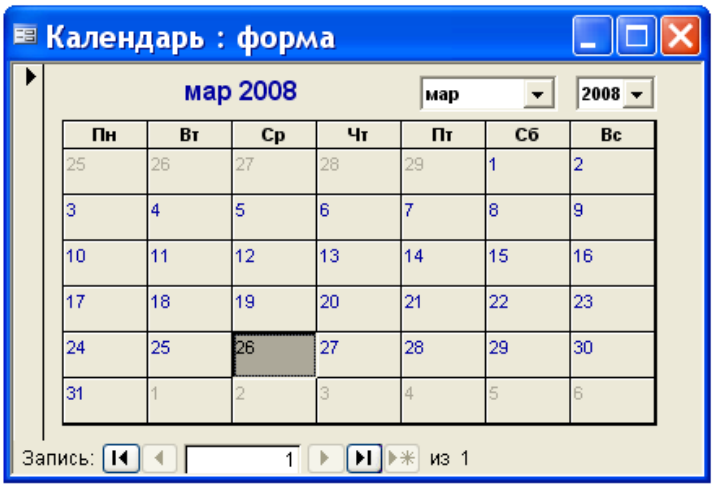


Рис. 5. Форма с элементом управления Календарь

Для создания этой формы используйте на **Панели элементов** элемент **Другие элементы** (см. рисунок 1). В списке элементов, открываемых этим элементом, выберите элемент управления **Календарь 11.0**. Форму назовите **Календарь**. Выясните с помощью этой формы, на какой день недели попадает 1 января 2025 года.

3.2 Описание практических работ

Практическая работа № 1

База данных музыкальных групп

Цель: закрепление умения определять типы данных полей и их свойства в таблицах баз данных.

Задание

1. Для схемы базы данных музыкальных групп, показанной на рисунке 1, найдите в интернете информацию, которую надо ввести в таблицы: Альбомы, Туры, Группы и Концерты. В каждой из таблиц должно быть не менее шести записей.

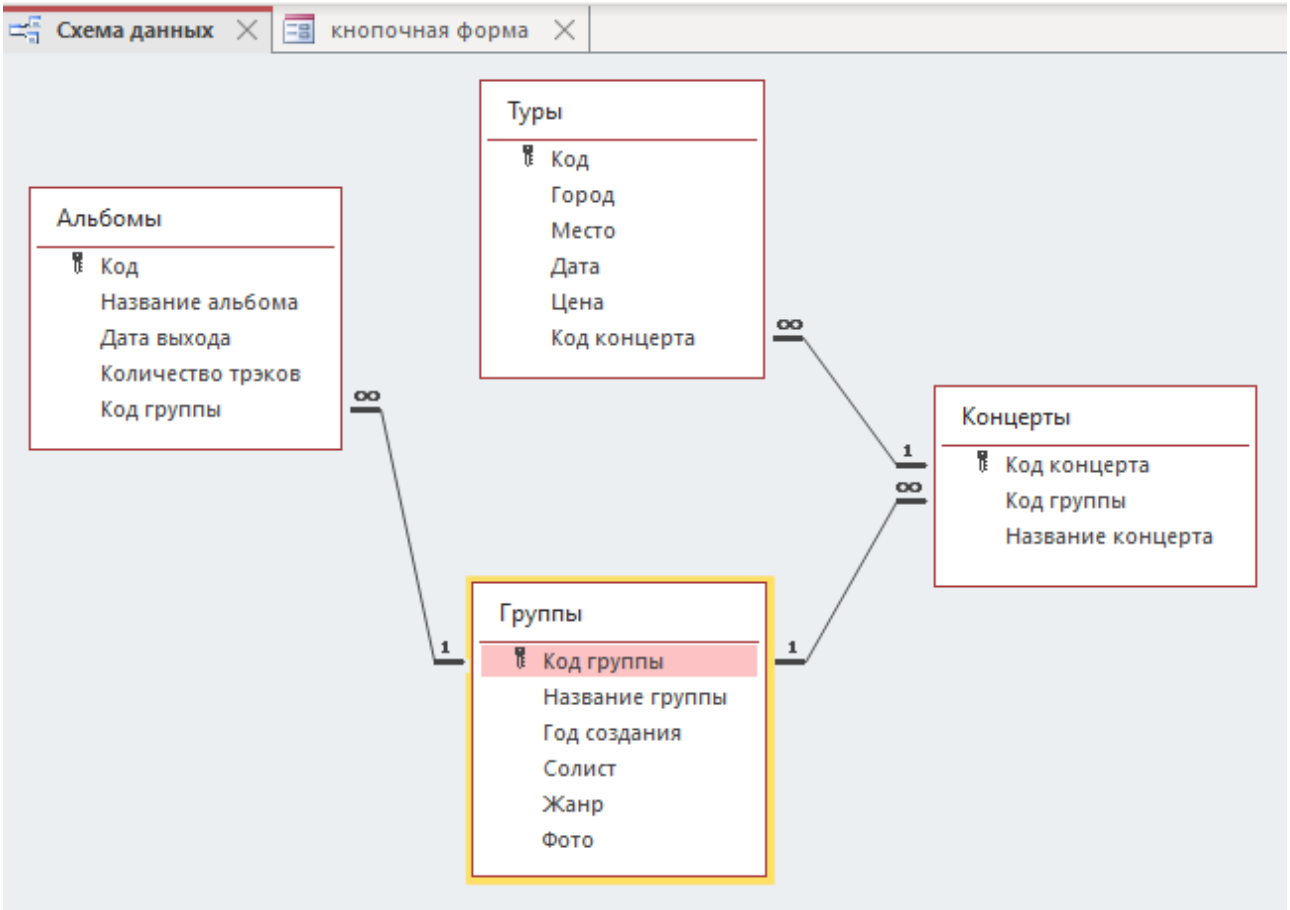


Рис. 1. Схема базы данных музыкальных групп

Найденную информацию сохраните в текстовом редакторе Microsoft Word.

2. Выполнив анализ найденных данных, заполните столбцы таблицы 1: Название таблицы, Имя поля, Тип данных, Свойства поля и Ключ.

Ключом в таблице может быть поле или группа полей, которые однозначно идентифицируют записи. При указании свойств полей надо обратить внимание на экономию памяти для хранения данных. Например, когда поле имеет тип данных Целое, то по умолчанию оно имеет свойство:

Размер – длинное целое. Часто для хранения с этим типом данного можно установить свойство: Размер – целое, тем самым сократив память для этого данного в 2 раза.

Таблица 1. Имена полей, типы данных и свойства полей

Название таблицы	Имя поля	Тип данных	Свойства поля	Ключ
Альбомы				
⋮				
Туры				
⋮				
Концерты				
⋮				
Группы				
⋮				

3. Создайте в Microsoft Access пустую базу данных с именем Альбомы. В этой базе данных создайте макеты таблиц: Альбомы, Туры, Группы и Концерты, используя при этом информацию из таблицы 1, полученную после ее заполнения.

4. Установите между таблицами связи один-ко-многим, показанные на рисунке 1.

5. Введите в таблицы базы данных Альбомы: Альбомы, Туры, Группы и Концерты информацию, которую Вы нашли в Интернете после выполнении пункта 1 задания лабораторной работы.

Практическая работа № 2

База данных замков Беларуси

Цель: формирование умения создавать объекты базы данных замков Беларуси.

Задание

1. Создайте в системе управления базами данных MS Access пустую базу данных Замки Беларуси.

2. В базе данных Замки Беларуси создайте макеты пяти таблиц: Замки, Архитекторы, Состояние, Экскурсии и Экскурсоводы, используя для этих целй режим Конструктора. Поля, которые должны содержаться в указанных таблицах, а также типы полей представлены в таблице 1.

Таблица 1. Поля таблиц базы данных Замки Беларуси

Таблица	Поле	Тип поля	Ключ
Экскурсии	Код экскурсии		
Экскурсии	Название экскурсии		
Экскурсии	Код экскурсовода		
Экскурсии	Код замка		Да
Экскурсии	Численность группы		
Экскурсии	Язык		
Состояние	Код состояния		
Состояние	Описание		
Архитекторы	Код архитектора		Да
Архитекторы	Фамилия		
Архитекторы	Имя		
Архитекторы	Отчество		
Замки	Код замка		Да
Замки	Название		
Замки	Фото		
Замки	Код состояния		
Замки	Код архитектора		
Экскурсоводы	Код экскурсовода		Да
Экскурсоводы	Фамилия экскурсовода		
Экскурсоводы	Имя экскурсовода		
Экскурсоводы	Отчество экскурсовода		
Экскурсоводы	Дата		
Экскурсоводы	Время		

3. Для каждого поля в таблице1 укажите тип поля. После этого приступите к созданию схемы базы данных, показанной на рисунке 1. При установке связи меду таблицами не забудьте активизировать режим обеспечения целостности данных и подрежимы каскадное обновление связанных полей и каскадное удаление связанных записей. В этом случае Microsoft Access будет при вводе проверять данные и не позволит ввести ланные, нарушающие целостность базы данных.

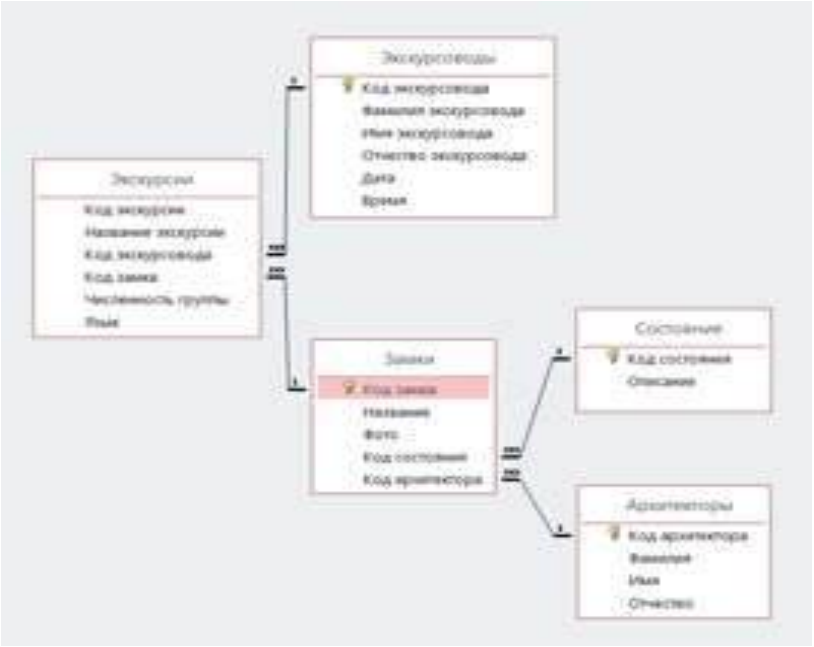


Рис. 1. Схема данных базы данных Замки Беларуси

4. Сейчас приступите к поиску информации, необходимой для ввода в макеты таблиц: Замки, Архитекторы, Состояние, Экскурсии и Экскурсоводы. Поскольку мы установили при создании связей между таблицами режим целостности данных, то тем самым мы задали очередность заполнения таблиц при вводе данных. Об этом надо помнить. Введите в таблицы по 6-8 записей, чтобы можно было в дальнейшем проверить работоспособность базы данных.

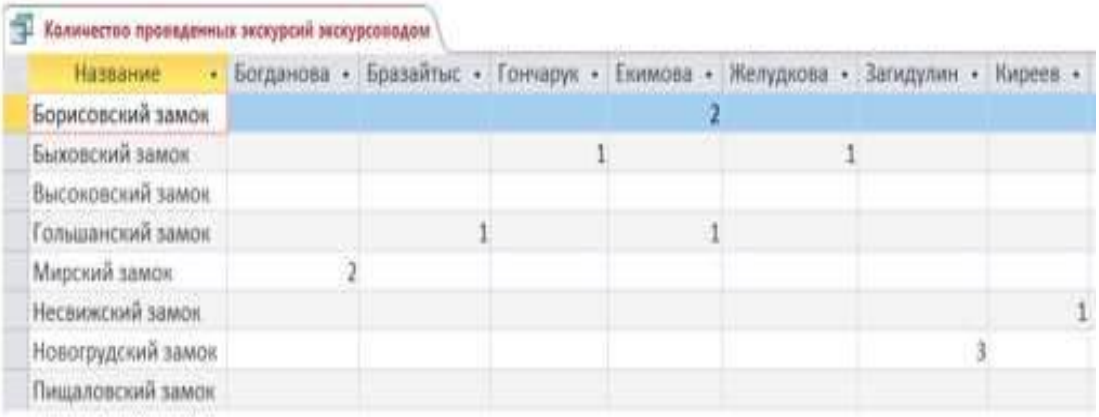
5. Используя конструктор запросов, создайте запрос Количество проведенных экскурсий экскурсоводом. В режиме Конструктора он будет выглядеть так, как это показано на рисунке 2. Созданный запрос называется перекрестным. Для его создания мы использовали 3 таблицы: Замки, Экскурсии и Экскурсоводы. Эти таблицы размещены в верхней части бланка QBE (запроса по образцу).

В нижней части бланка в строке Групповая операция для полей: Название и Фамилия экскурсовода указали операцию Группировка, а для поля Название экскурсии операцию – Count (Считать). В строке Перекрестная таблица для поля Название мы выбрали параметр Заголовки

строк, для поля Фамилия экскурсовода – параметр Заголовки столбцов, для поля Название экскурсии – параметр Значение.

Рис. 2. Режим конструктора

Проверить работу запроса можно с помощью перехода в режим Таблицы. Запрос выводит общее количество экскурсий каждого экскурсовода по замкам Беларуси, при этом можно заметить, что экскурсоводы, которые не провели ни одной экскурсии, не отображаются (см. рисунок 3).



Название	Богданова	Бразайтыс	Гончарук	Екимова	Желудкова	Загидулин	Киреев
Борисовский замок				2			
Быковский замок			1		1		
Высоковский замок							
Голышанский замок		1		1			
Мирский замок	2						
Несвижский замок							1
Новогрудский замок						3	
Пищаловский замок							

Рис. 3. Режим таблицы

6. Создайте запрос Поиск замка по фамилии архитектора. Для того чтобы создать новый запрос нужно нажать на кнопку Конструктор запроса.

Данный запрос будет основываться на таких таблицах, как Архитекторы и Замки. В динамический набор включите следующие поля: Фамилия и Название. В условие отбора поля Фамилия введите текст Like [Введите первые буквы:] & "*". Это показано на рисунке 4.

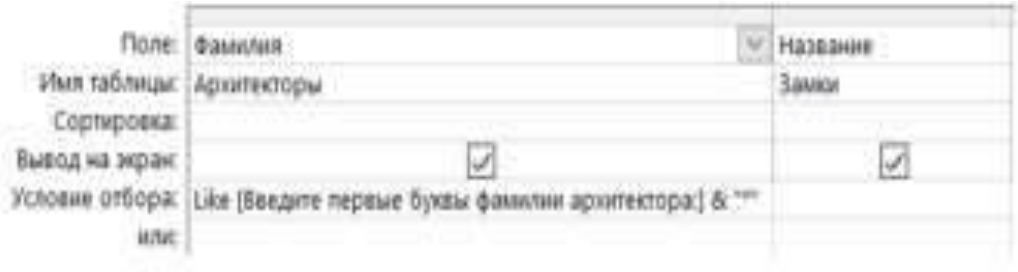


Рис. 4. Условие отбора

Запрос сохраните под именем Поиск замка по фамилии архитектора и проверьте его работу. Перейдите в режим таблицы, при этом появится всплывающее окно для ввода значения параметра. Введите первые буквы фамилии архитектора, например Бортк (см. рисунок 5).

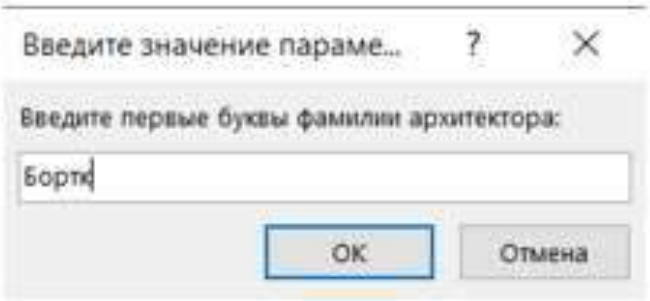


Рис. 5. Ввод значения параметра

Нажмите кнопку OK. Результат выполнения запроса показан на рисунке 6.

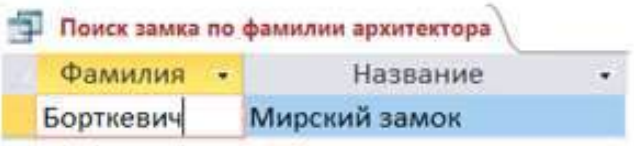


Рис. 6. Результат выполнения запроса

Таким образом мы можем найти все замки по первым буквам архитекторов, которых вы внесли в базу данных Замки Беларуси.

7. Создайте запрос, который по численности группы смог бы определить название экскурсии. Данный запрос точно такой же, как и предыдущий, только он отличается от запроса Поиск замка по фамилии архитектора строкой Условие отбора. В Условии отбора поля Численность группы надо создать параметр [Введите численность группы].

8. Создайте запрос, который бы показывал, в каком состоянии находится замок. Данный запрос надо создавать на основе таблиц Замки и Состояние, а динамический набор должен содержать поля Название и

Состояние. В условии отбора поля Название следует написать параметр [Введите название замка].

9. Создайте еще один запрос, который бы определял название экскурсии по фамилии экскурсовода. Запрос следует создавать на основе таблиц Экскурсии и Экскурсоводы, а динамический набор должен содержать поля Название экскурсии и Фамилия экскурсовода. В условие отбора поля Фамилия экскурсовода создайте параметр [Введите фамилию экскурсовода].

10. В форме СУБД Microsoft Access, как на витрине магазина, удобно просматривать и открывать нужные элементы. Так как форма – это объект, с помощью которого пользователи могут добавлять, редактировать и отображать данные, хранящиеся в базе данных Access, ее внешний вид играет важную роль.

Создайте по своему усмотрению 3 формы, которые, на ваш взгляд, улучшат внешний вид базы данных.

11. Основными элементами кнопочной формы базы данных «Замки Беларуси» являются таблицы, запросы и формы. Главная кнопочная форма является своеобразным меню, которое облегчает работу с базой данных.

Создайте главную кнопочную форму базы данных «Замки Беларуси». Можете ее вид сделать похожим на форму, приведенную на рисунке 7.

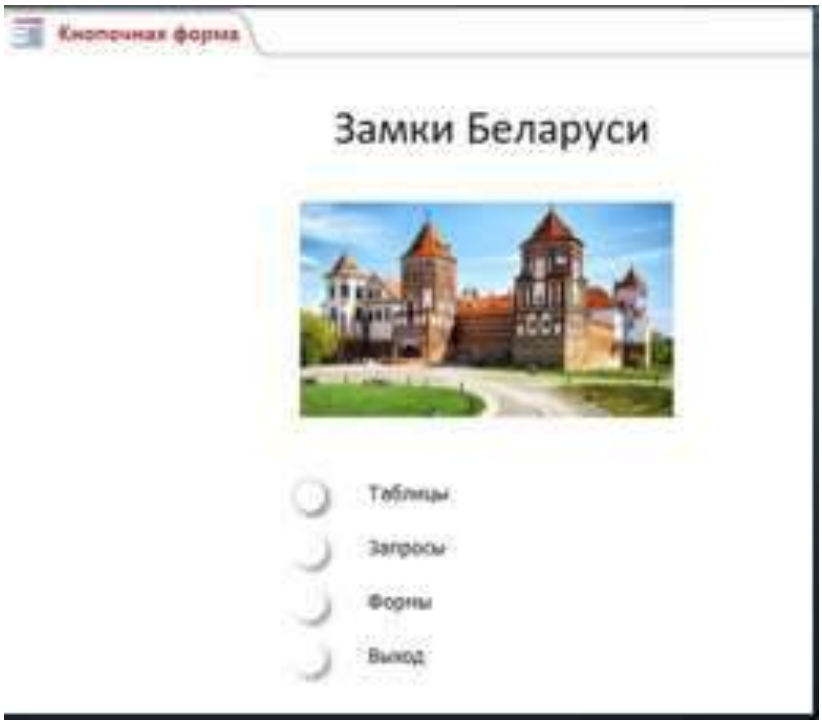


Рис. 7. Главная кнопочная форма

4 РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

4.1 Задания для контролируемой самостоятельной работы студентов

Самостоятельная работа студентов направлена на совершенствование их умений и навыков по дисциплине «Аналитическая обработка источников информации». Цель самостоятельной работы студентов – способствование усвоению в полном объеме учебного материала дисциплины через систематизацию, планирование и контроль собственной деятельности. Преподаватель дает задания по самостоятельной работе и регулярно проверяет их исполнение.

Содержание и формы контролируемой самостоятельной работы студентов рекомендуется непосредственно связывать с использованием метода проектов, что позволяет реализовывать индивидуальный подход к обучению.

4.2 Перечень вопросов к зачету

1. Понятие информационной системы. Информационная технология. Автоматизированные информационные системы.
2. Требования к организации баз данных.
3. Экспорт в текстовый файл. Типы форматов текстовых файлов, в которые можно экспортировать данные из базы данных.
4. Понятие базы данных. Понятие системы управления базами данных. Понятие банка данных.
5. Создание кнопочной формы с помощью мастера создания кнопок.
6. Использование Панели элементов. Размещение изображения.
7. Создание перекрестного запроса с помощью мастера.
8. Использование групповых операций для вычислений.
9. Операторы для работы со строками. Функции Left, Right, Mid.
10. Импорт данных из электронных таблиц. Учет заголовков столбцов при импорте. Изменение определения полей таблицы.
11. Создание запроса с вычисляемым полем. Использование выражений в вычисляемом поле.
12. Использование ключевого слова для поиска информации. Поиск по двум ключевым словам.
13. Поиск по первым буквам искомого значения.
14. Импорт текстовых файлов. Требования к подготовке импортируемого текстового файла. Стандартные разделители полей.
15. Создание запроса с параметрами.
16. Назначение макроса Autoexec.
17. Информационная модель базы данных. Иерархическая модель данных. Сетевая модель данных. Реляционная модель данных.
18. Создание перекрестного запроса в режиме конструктора.
19. Понятия: отношение, атрибут, схема отношения, кортеж.
20. Требования к определению заголовков строк, заголовков столбцов и значений перекрестной таблицы.
21. Связь между элементами файла базы данных, таблицы, отношения и сущности.
22. Объектные и связные отношения. Понятие ключа. Ссылочная целостность данных.
23. Понятие формы. Использование мастера для создания форм.
24. Просмотр и изменение свойств запроса и его элементов.
25. Примеры использования функций DatePart, Format, Date.
26. Символы шаблона. Использование оператора Like с символами шаблона.

27. Задание условий отбора. Элементы выражения в условии отбора.

28. Создание запроса в режиме конструктора.

29. Кнопочная форма. Создание формы в режиме конструктора.

30. Создание запроса с помощью мастера.

31. Понятие запроса. Динамический набор записей. Типы запросов.

32. Отличие связывания от импорта. Объекты связывания. Способы обработки связанных данных.

33. Первая нормальная форма отношения (1НФ). Требования реляционной модели к отношениям.

34. Назначение отчетов. Использование конструктора отчетов.

35. Добавление в БД новых таблиц. Добавление в схему данных новых таблиц.

36. Создание структуры таблицы в режиме конструктора.

37. Способы создания структуры таблицы.

38. Установка связей между таблицами. Типы связей. Требования к типам данных и свойствам при связывании таблиц.

39. Режим обеспечения целостности данных. Режим каскадное обновление связанных полей. Режим каскадное удаление связанных записей.

40. Режимы работы с таблицей. Переход из одного режима в другой.

4.3 Критерии оценки результатов учебной деятельности студентов

Для выявления и исключения пробелов в знаниях студентов рекомендуется использовать следующие средства:

1) фронтальный опрос на лекциях, лабораторных и семинарских занятиях;

2) критериально-ориентированные тесты для контроля знаний информационных технологий;

3) выполнение тестовых заданий с произвольной формой ответа для контроля умения анализировать, грамотно излагать и формулировать свои соображения и выводы в данной предметной области;

4) выполнение творческих заданий, которые предполагают эвристическую деятельность и поиск неформальных решений.

5 ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

5.1 Учебная программа

Технологии создания баз данных в управлении и коммуникациях: учебная программа для специальности «Социально-культурный менеджмент и коммуникации», профилизации «Мультимедийные технологии и цифровые коммуникации»; сост. П. В. Гляков. – УД-284/эуч. – . .2025.

5.2 Учебно-методическая карта учебной дисциплины для дневной формы получения высшего образования

Номер раздела, темы	Наименование разделов и тем	Количество аудиторных часов			Количество часов УСР	Форма контроля знаний
		Лекции	Практические занятия	Лабораторные занятия		
1	Введение в базы данных					
1.1	Основные понятия баз данных	2				
1.2	Классификация и функции СУБД	2				
1.3	Проектирование и создание базы данных MS Access	2				
2	Работа с базой данных					
2.1	Создание и связывание таблиц в базе данных			4		
2.2	Выбор данных с помощью запросов			8		
2.3	Перекрестный запрос			2		
2.4	Язык конструирования запросов SQL			2	2	Проект
3	Представление данных					
3.1	Представление данных в виде форм			4		
3.2	Обработка данных с помощью отчетов			2	2	Отчет
3.3	Использование мастера отчетов			2	2	Отчет
3.4	Использование кнопок в формах			4		
4	Базы данных сферы культуры					
4.1	База данных музыкальных групп		4		2	Проект
4.2	База данных замков Беларуси		4			
Всего		6	8	28	8	Зачет

5.3 Список основной литературы

1. *Аврунев, О. Е*. Модели баз данных : учеб. пособие / О. Е. Аврунев, В. М. Стасышин. – Новосибирск : Новосибирский государственный технический университет, 2018. – 124 с. : – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=575324 (дата обращения: 11.04.2022). – Библиогр. в кн. – ISBN 978-5-7782-3749-0. – Текст : электронный.

2. *Шилин, А. С.* Перспективные методы проектирования реляционных баз данных : учеб. пособие / А. С. Шилин. – Москва ; Берлин : Директ-Медиа, 2021. – 136 с. : – Режим доступа: по подписке. – URL: https://biblioclub.ru/-index.php?page=book&id=602240 (дата обращения: 11.04.2022). – Библиогр. в кн. – ISBN 978-5-4499-1890-1. – Текст : электронный.

3. *Информационные процессы и системы*: *базы данных* [Электронный ресурс] : учеб.-метод. комплекс / сост. П. В. Гляков. – Минск : БГУКИ, 2017. – 205 с. – Режим доступа: http://repository.buk.by/123456789/14568. – Дата доступа: 18.02.2021.

4. *Информационные технологии в искусствоведении*. С электронным приложением в QR-кодах : учеб. пособие / Т. С. Жилинская, Н. Г. Гончарик, Т. И. Песецкая, В. С. Якимович. – Минск : БГУКИ, 2023. – 210 с. – ISBN 978-985-522-335-2. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/438572

5.4 Список дополнительной литературы

1. Гвоздева, В. А. Базовые и прикладные информационные технологии : учеб. / В. А. Гвоздева. – М. : ИД ФОРУМ, 2020. – 384 с.

2. Гляков, П. В. Система управления базами данных Access 2.0 : учеб. пособие / П. В. Гляков, С. Н. Карачун. – Минск : РИПО, 1998. – 100 с.

3. Гляков, П. В. Импорт, экспорт и связывание данных в Microsoft Access : метод. рекомендации / П. В. Гляков. – Минск : РИПО, 2005. – 34 с.

4. Гринчук, С. Н. Система управления базами данных Microsoft Access / С. Н. Гринчук, И. А. Дюба. – Минск : АПО, 2006. – 187 с.

5. Информатика для гуманитариев : учебник и практикум для вузов / под ред. Г. Е. Кедровой . – 2-е изд. – М. : Юрайт, 2021. – 653 с.

6. Михеева, Е. В. Информационные технологии в профессиональной деятельности : учеб. / Е. В. Михеева. – М. : Академия, 2020. – 416 с.

7. Михеева, Е. В. Практикум по информационным технологиям в профессиональной деятельности : учеб. пособие / Е. В. Михеева. – М. : Академия, 2019. – 288 с.

8. Основы построения баз данных : учеб. пособие / Д. В. Чмыхов, А. С. [и др.]. – Москва ; Берлин : Директ-Медиа, 2021. – 124 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=602227 (дата обращения: 11.04.2022). – Библиогр. в кн. – ISBN 978-5-4499-2428-5. – Текст : электронный.