Федеральное государственное автономное образовательное учреждение
высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики им. А.Н Тихонова

Департамент компьютерной инженерии

РАБОТА С БАЗАМИ ДАННЫХ. СУБД ACCESS

Методические указания к лабораторной работе № 5 по курсу "Базы данных"

Москва

2017

Составители: доцент, к.т.н. И.П. Карпова
магистрант А.А. Малышев

УДК 681.3

Методические указания к лабораторной работе №5 по курсу "Базы данных": Работа с базами данных. СУБД Access. / Московский институт электроники и математики НИУ ВШЭ; Сост.: Карпова И.П., Малышев А.А. – М., 2016. – 39 с.

Лабораторная работа посвящена созданию средствами СУБД Access приложения к базе данных, работающей под управлением СУБД MySQL. Приложение включает различные элементы интерфейса: экранные формы, отчёты и запросы.

Для студентов III-IV курсов технических факультетов, изучающих системы управления базами данных и автоматизированные информационные системы.

Табл. 4. Ил. 50. Библиогр.: 3 назв.

Содержание

Цель выполнения работы

Цель выполнения лабораторной работы – получение практических навыков создания реляционных баз данных (БД) и приложений для работы с ними на примере одной из самых распространённых настольных СУБД – MS Access 2010. Выполнение работы состоит в подключении к базе данных, созданной под управлением СУБД MySQL во время выполнения лабораторной работы №1, а также построении схемы этой БД, создании экранных форм, отчётов и запросов.

Работы являются продолжением лабораторного практикума предыдущих модулей и выполняются по тому варианту задания, который был получен студентом в начале курса.

Описание работы в данном пособии приводится на примере БД «Проектная организация». Вы же должны выполнить эти работы с учетом особенностей той базы данных, которая была создана вами при выполнении лабораторной работы №1 по данному курсу.

В ходе выполнения лабораторной работы №5 необходимо подключить к системе Access в качестве внешних источников данных таблицы базы данных, созданные при выполнении лабораторной работы №1, и создать схему БД. Схема должна отражать все внешние ключи в ваших таблицах. Затем для каждой таблицы создаются экранные формы. Каждое поле внешнего ключа необходимо определить как "Поле со списком". Минимум одна экранная форма должна содержать подчиненную экранную форму. Далее необходимо написать запросы и отчёты. Каждая из таблиц БД должна входить хотя бы в один запрос и один отчет; минимум два запроса и два отчета должны включать более одной таблицы (соединение таблиц или подзапрос). В завершение требуется создать поисковую форму. Она может быть основана на любом запросе, содержащем не менее двух таблиц.

Общий показатель качества выполненной работы – полноценное функциональное приложение, соответствующее потребностям пользователя вашей предметной области.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Архитектура клиент-сервер для баз данных

В настоящее время большинство систем управления базами данных (СУБД) поддерживают режим работы клиент-сервер. Технология клиент-сервер обеспечивает прикладным программам – клиентам – доступ к данным, которыми управляет сервер, и позволяет нескольким клиентам работать с одним сервером.

Сервер определим как процесс, который выполняет запросы от других процессов. (Иногда под сервером подразумевают также узел в сети, на котором выполняется процесс сервера). **Сервер базы данных** определим как логический процесс, отвечающий за обработку запросов к базе данных. **Клиента** определим как процесс, отправляющий серверу запрос на обслуживание. К функциям клиента относятся: установление связи с сервером базы данных (БД); запрос конкретного вида обслуживания; получение результатов запроса; подтверждение окончания обслуживания.

При использовании технологии клиент-сервер клиент посылает запрос серверу, который в соответствии с запросом выбирает данные из базы данных, возможно, подвергает их предварительной обработке и отправляет результаты клиенту. Таким образом, основную работу с базой данных выполняет сервер, что позволяет уменьшить сетевой трафик.

В качестве языка, на котором формулируются запросы к базе данных, обычно выступает язык SQL.

Основной принцип технологии клиент-сервер – разделение функций стандартного интерактивного приложения на четыре группы [1]:

- Функции ввода и отображения данных (интерфейс).
- Прикладные функции.
- Функции хранения данных и управления информационными ресурсами.
- Служебные функции.

Прикладные функции зависят от предметной области, например, для системы продажи авиабилетов такими функциями являются поиск мест на рейс, продажа и бронирование билетов и т.д.

Служебные функции играют роль связок между функциями групп 1–3. В соответствие с этими группами выделяют три логических компонента:

1. Компонент представления (для ввода и отображения данных).
2. Прикладной компонент (для реализации прикладных функций).
3. Компонент доступа к информационным ресурсам (для управления данными).

Рассмотрим модель организации архитектуры клиент-сервер для баз данных, предложенную специалистами Garther Groups (

Рис. 1 Модель организации архитектуры клиент-сервер для баз данных

) [1].

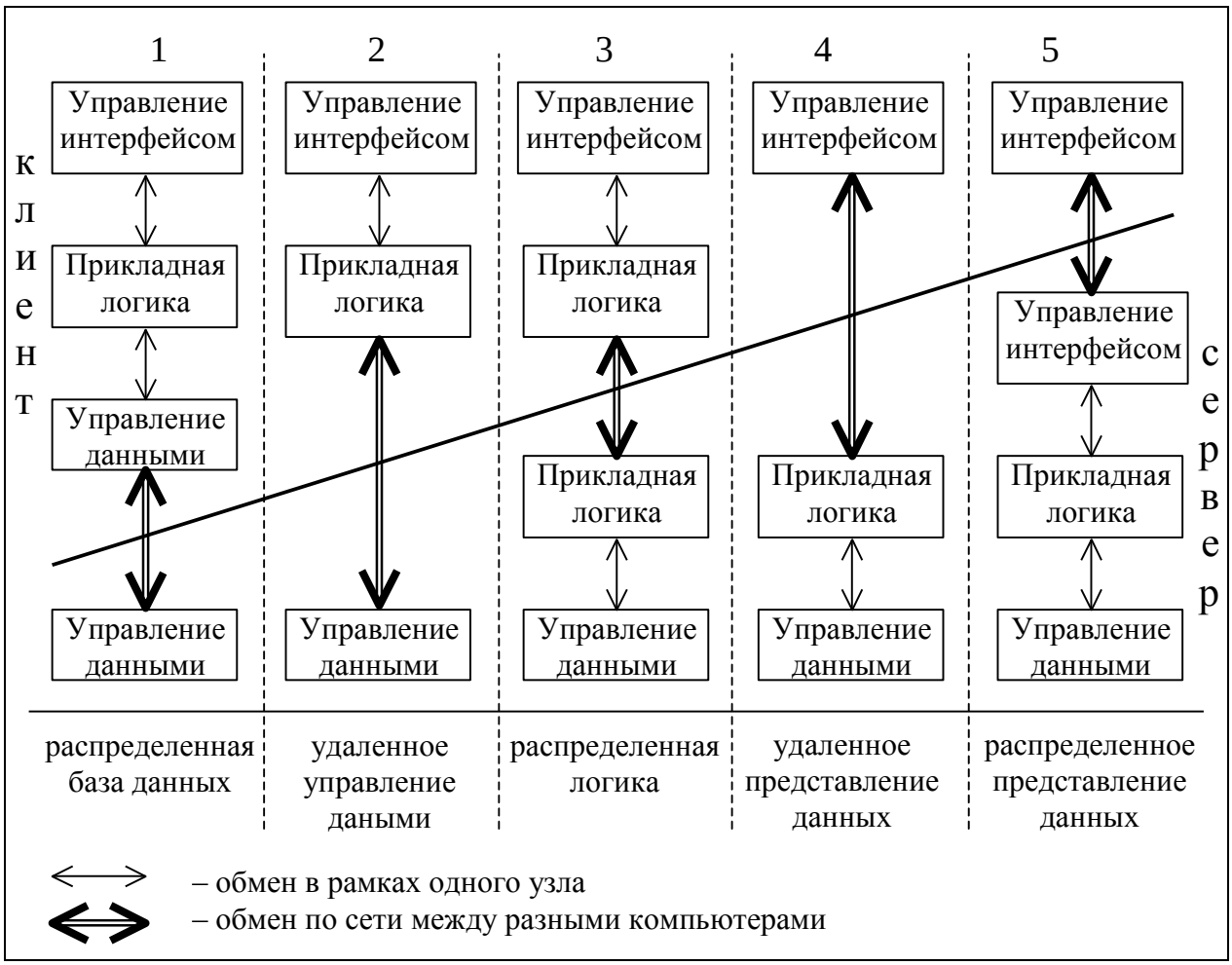


Рис. 1 Модель организации архитектуры клиент-сервер для баз данных

Как видно из Рис. 1, модель архитектуры клиент-сервер зависит от распределения функций между клиентом и сервером. Для создания распределённой базы данных (1) необходима СУБД, поддерживающая функции распределённости. Это сложный в реализации и очень дорогостоящий вариант. Удалённое управление данными (2) используется в файл-серверах, основной недостаток которых – большой трафик и низкая параллельность работы с данными.

В реальных информационных системах чаще всего используются модели распределённой логики (3) и удалённого представления данных (4). Первая из них применяется тогда, когда нужно разгрузить сервер и передать часть функций по обработке данных клиенту. А удалённое представление данных – это классическое применение сервера баз данных.

Модель распределённого представления данных чаще всего говорит о неудачном проектировании АИС. Сервер не должен заниматься вопросами организации представления данных: эти задачи должно выполнять клиентское приложение. К вопросам представления, в частности, относится преобразование типов данных, формирование отчётов, диаграмм и т.п.

1.2. Технологии доступа к базе данных

Доступ к базе данных обычно осуществляется с помощью интерфейса, который реализован как приложение к базе данных. Приложение может быть

написано на различных языках программирования высокого уровня и с использованием различных программных средств. Для того чтобы упростить процесс разработки приложений баз данных, были созданы различные технологии, скрывающие от разработчика специфику работы с конкретной базой данных.

К таким технологиям можно отнести:

- ADO (Active Data Objects) – модель программирования, которая предназначена для создания на Web-серверах динамических интерактивных Web-страниц для организации подключения к базам данных.
- BDE (Borland Database Engine) – интерфейс между приложением и базой данных, реализованный в рамках продуктов фирмы Borland.
- ODBC (Open Database Connectivity) – стандарт, предусматривающий использование единого интерфейса для доступа к базам данных, поддерживающим язык SQL.

Структура организации доступа к данным с использованием этих компонентов приведена на Рис. 2 [2].

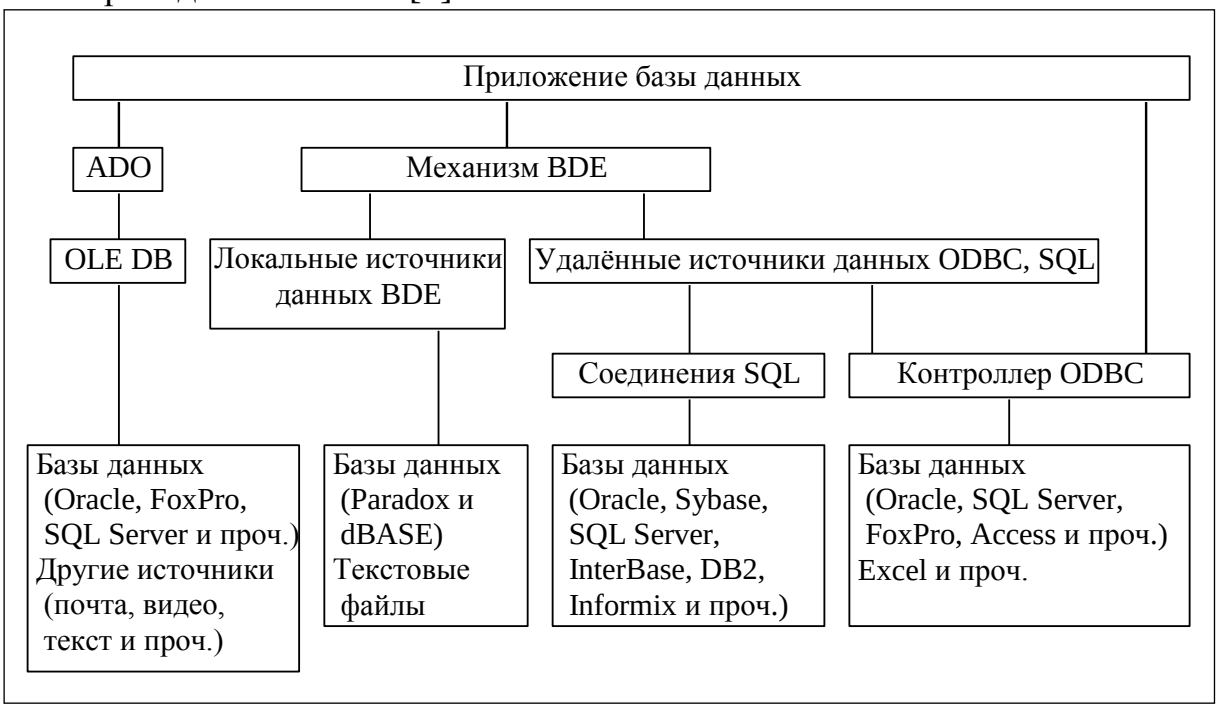


Рис. 2. Компоненты доступа к данным

Как видно из Рис. 2, модель ADO базируется на технологии OLE DB (Object Linking and Embedding Databases – связывание и внедрение объектов баз данных). OLE является объектно-ориентированной технологией разработки повторно используемых программных компонентов. Она позволяет приложениям совместно использовать объекты, которые обладают специальными функциями. В качестве источника данных OLE-приложений могут выступать таблицы баз данных, представления, а также текстовые файлы, электронные таблицы, диаграммы и другие графические изображения.

Механизм BDE действует как интерфейс между приложением и базой данных, обеспечивая работу компонентов доступа. Он реализован как набор системных файлов DLL (Dynamic Link Library). Через BDE из приложений можно

непосредственно обращаться к базам данных фирмы Borland (Paradox, dBASE), а обращение к другим базам данных BDE перенаправляет менеджеру драйверов ODBC или SQL-серверам. Технология BDE применяется в таких широко распространённых продуктах, как C++ Builder, Borland C++, Delphi, IntraBuilder и JBuilder. Для доступа к базе данных через BDE достаточно знать алиас базы данных (т.е. имя, по которому к ней обращаются).

И третья технология – ODBC. В стандарте ODBC язык SQL рассматривается как базовое средство доступа к данным. Этот интерфейс встраивается непосредственно в язык Си и обеспечивает высокий уровень универсальности. В результате одно и то же приложение может получить доступ к базам данных разных СУБД без внесения изменений в текст программы. Для связи приложения с любой выбранной пользователем СУБД достаточно иметь соответствующий ODBC-драйвер.

В интерфейс ODBC включены следующие элементы:

- Библиотека функций, вызов которых позволяет приложению подключаться к базе данных, выполнять SQL-операторы и извлекать информацию из результирующих наборов данных.
- Стандартный метод подключения и регистрации СУБД.
- Стандартное представление для различных типов данных.
- Стандартный набор кодов ошибок.
- Типовой синтаксис SQL-операторов, построенный на использовании спецификаций стандартов X/Open и ISO CLI.

Архитектура ODBC включает четыре элемента:

1) **Приложение**. Оно выполняет вызов функций библиотеки ODBC для оправки SQL-операторов в СУБД и обработку возвращаемых СУБД данных.
2) **Менеджер драйверов**. Этот компонент выполняет загрузку драйверов по требованию приложений. Менеджер драйверов представляет собой библиотеку DLL.
3) **Драйверы баз данных**. Они обрабатывают вызовы функций ODBC и направляют SQL-запросы в конкретные источники данных, а также возвращают полученные от источников данные приложению. При необходимости драйверы выполняют модификацию запросов с целью приведения их в соответствие с синтаксическими требованиями конкретной СУБД. Драйверы предоставляют только те возможности, которые реализованы в целевой СУБД.
4) **Источники данных**. Источником данных является база данных, электронная таблица и т.п. Данные в базе данных контролируются СУБД и операционной системой.

2. Подключение к Access внешнего источника данных

Итак, технология ODBC позволяет получать доступ к различным источникам данных. В связи с этим интересным представляется следующий подход. С помощью системы Access и драйверов ODBC можно создавать приложения

по работе с различными СУБД [3]. Это может показаться странным: Access сама по себе позволяет создавать базы данных, зачем использовать в дополнение ещё одну СУБД? Но в том случае, если база данных достаточно большая, и Access не может эффективно обрабатывать такой объём данных, то базу данных целесообразно хранить под управлением другой, более мощной СУБД, а с помощью Access создавать пользовательские приложения. Тем более что Access является удобным средством визуального программирования приложений.

2.1. Создание связи с внешней базой данных

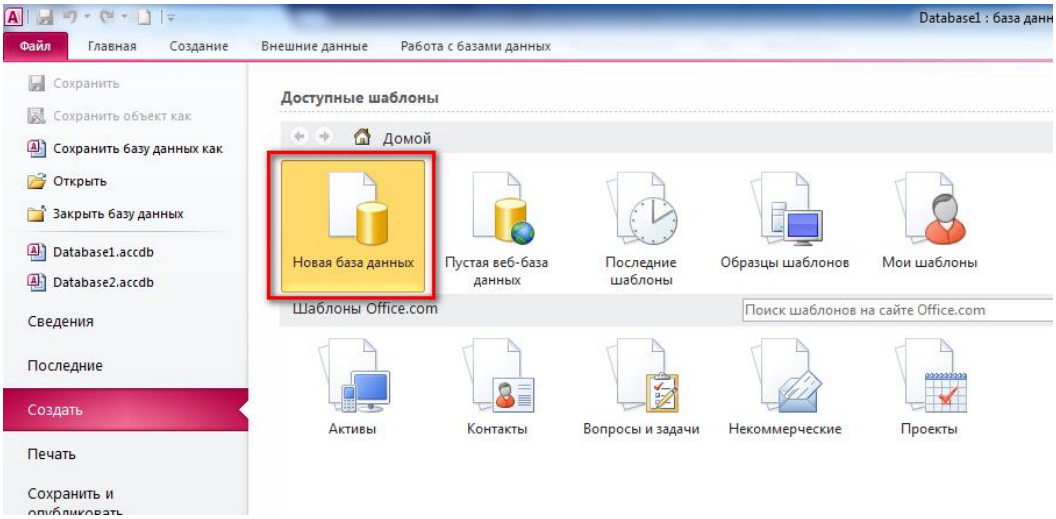


Рис. 3. Создание новой БД

Рассмотрим эту операцию на примере базы данных, работающей под управлением СУБД MySQL. Если запустить Access, создать новую БД (Рис. 3), выбрать *на ленте во вкладке Внешние данные → Базы данных ODBC*, то появится окно менеджера внешних данных. Первый шаг – указание системе источника и места назначения данных (Рис. 4). В данной лабораторной работе нужно выбрать пункт «Создать связанную таблицу для связи с источниками данных» и подтвердить выбор кнопкой ОК.

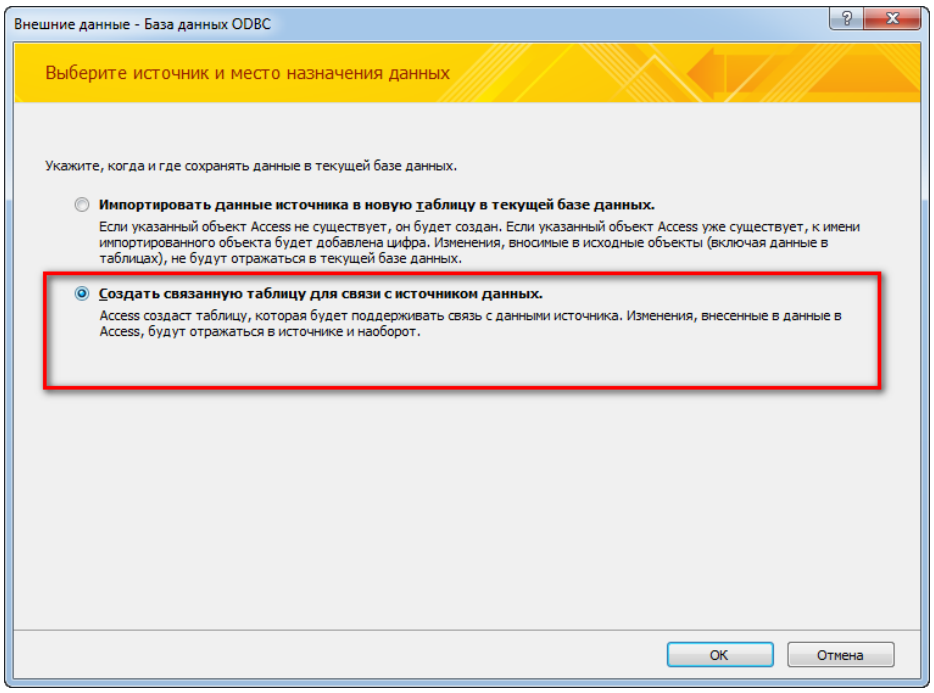


Рис. 4. Окно выбора источника и места назначения данных

В появившемся окне нужно из списка источников выбрать необходимый источник (Рис. 5).

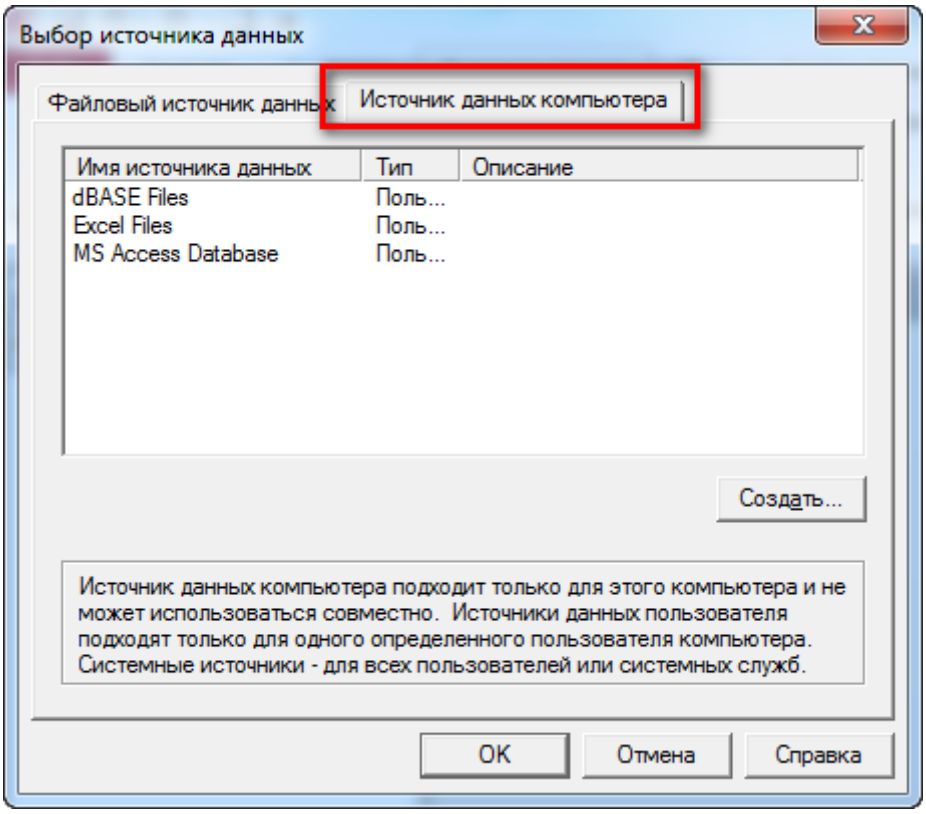


Рис. 5. Выбор источника данных

Если в списке источников данных нет требуемого источника (для нашего примера – MySQL), то его нужно создать (Рис. 6) (подробнее см. в [3]).

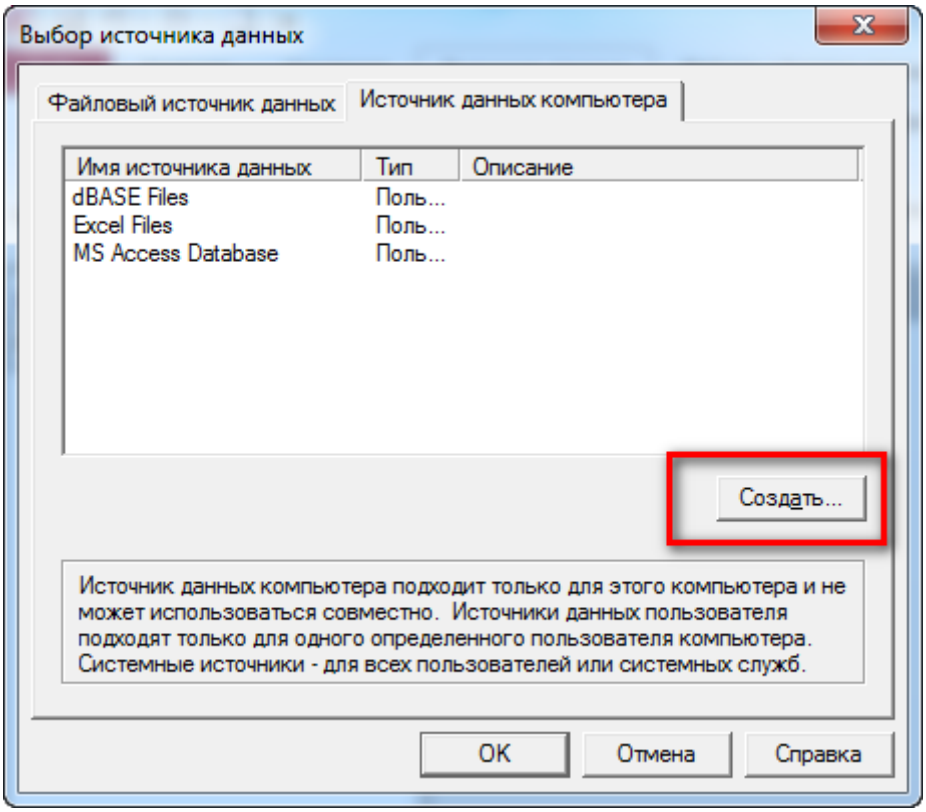


Рис. 6. Создание нового источника данных

Примечание: необходимо скачать и установить MySQL Connector/ODBC (для систем x64 рекомендуется поставить одновременно версии x32 и x64).

При создании система просит выбрать драйвер, для которого создаётся источник (для нашего примера – это драйвер для MySQL, Рис. 7).

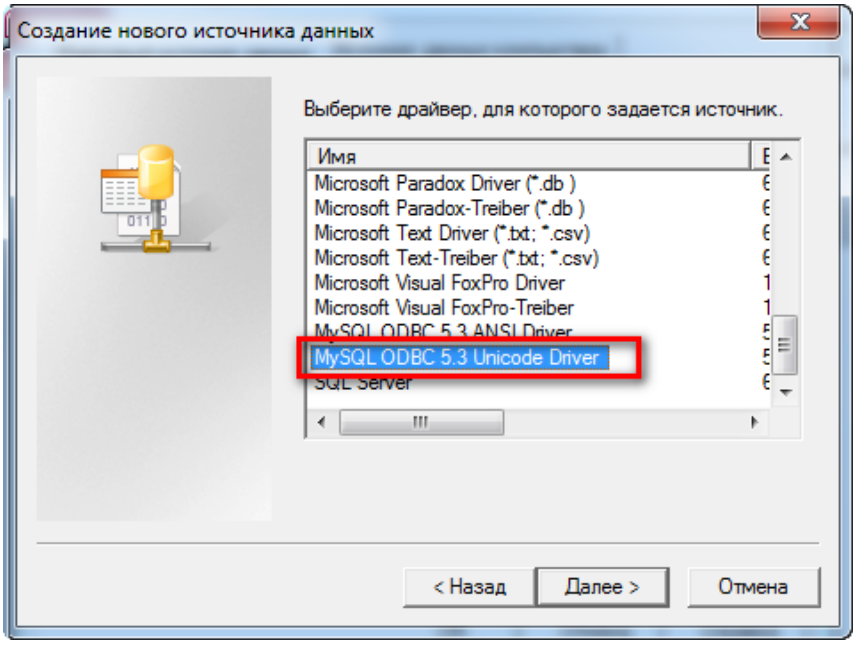


Рис. 7. Добавление нового источника данных

Для получения доступа выводится окно с полями, состав которых зависит от СУБД. Для MySQL обычно необходимо указывать хост (localhost для локальной БД), имя базы данных (в нашем случае – mysql) и имя пользователя (root) (**Ошибка! Источник ссылки не найден.**). Не забудьте протестировать соединение (Test Data Source, Рис. 8).

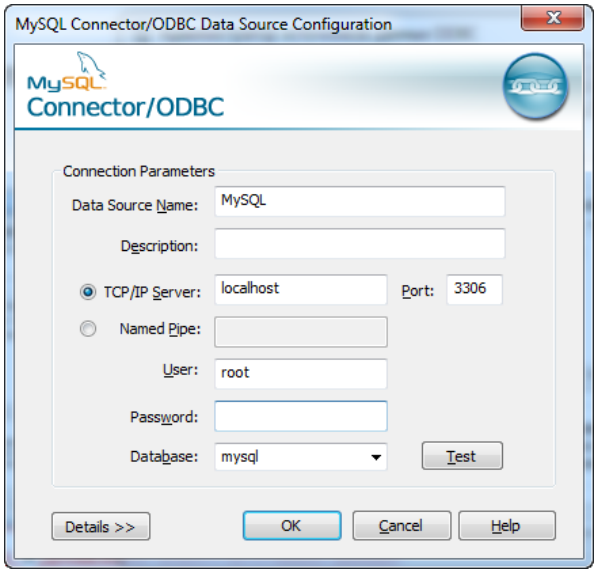


Рис. 8. Задание параметров доступа к базе данных

При успешном соединении с БД, появится окно связи с таблицами (Рис. 9). Из него выбираются таблицы, которые будут использоваться в Access.

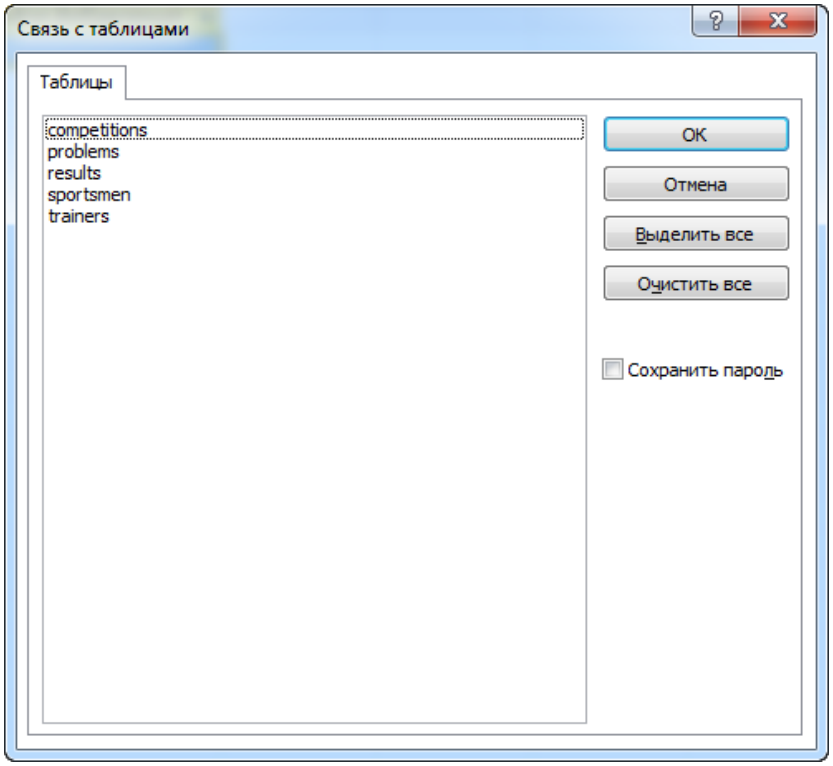


Рис. 9 Связь Access с таблицами из базы данных в MySQL

2.2. Описание таблиц тестовой базы данных

В качестве примера будет использоваться база данных проектной организации – предприятия, которое занимается выполнением различных проектов по договорам с заказчиками. Схема этой базы данных приведена на Рис. 10. Она отражает связи в предметной области:

– Каждый сотрудник работает в определённом отделе, в каждом отделе могут работать несколько сотрудников.
– Каждый проект относится к определённому отделу, каждый отдел может отвечать за выполнение нескольких проектов.
– Каждый сотрудник может принимать участие в выполнении нескольких проектов, над каждым проектом могут трудиться несколько сотрудников.

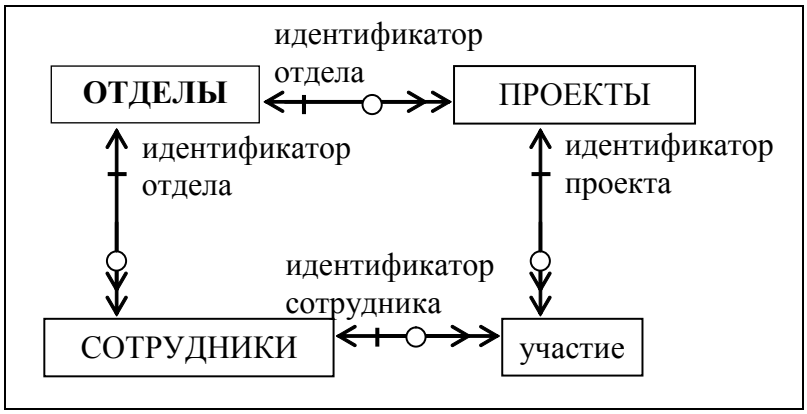


Рис. 10 Схема базы данных проектной организации

Схемы таблиц этой базы данных приведены в табл.1-4.

Таблица 1 Схема таблицы "Отделы" (Depart)

Поле	*Название*	*Тип данных*	*Размер*	*Ограничения целостности*
Номер отдела	Did	счетчик	длинное целое	**первичный ключ**
Название отдела	Name	текстовый	50	обязательное
Телефон	Phone	текстовый	20	

Таблица 2 Схема таблицы "Сотрудники" (Emp)

Поле	Название	Тип данных	Размер	Ограничения целостности
Номер сотрудника	Id	счетчик	длинное целое	уникальное
ФИО сотрудника	EName	текстовый	50	обязательное
Дата рождения	Born	Дата	авто	обязательное
Пол	Sex	текстовый	1	значения 'м' или 'ж'
Номер отдела	Depno	числовой	длинное целое	обязательное
Должность	Post	текстовый	30	обязательное
Зарплата	Salary	числовой	длинное целое	обязательное
Паспортные данные	Passport	текстовый	100	обязательное
Телефон	Tel	текстовый	20	
Адрес	Addr	текстовый	100	

Таблица 3 Схема таблицы "Проекты" (Project)

Поле	Название	Тип данных	Размер	Ограничения целостности
Номер проекта	Pid	числовой	длинное целое	уникальное
Название проекта	Title	текстовый	255	обязательное
Заказчик	Client	текстовый	100	обязательное
Шифр проекта	Agreement	текстовый	20	обязательное
Начало проекта	Dbegin	дата	авто	обязательное
Окончание проекта	Dend	дата	авто	обязательное
Стоимость проекта	Cost	числовой	длинное целое	обязательное
Номер отдела	Depno	числовой	длинное целое	обязательное

Таблица 4 Схема таблицы "Участие" (Job)

Поле	Название	Тип данных	Размер	Ограничения целостности
Номер проекта	Pid	числовой	длинное целое	обязательное
Номер сотрудника	Id	числовой	длинное целое	обязательное
Роль сотрудника в проекте	Role	текстовый	20	Обязательное; принимает значения: руководитель, консультант, исполнитель

2.3. Создание схемы базы данных

После подключения всех таблиц требуется связать их внешними ключами в соответствии со схемой базы данных (окно ***Работа с базами данных*** – ***> Схема данных***). Сначала система предлагает перенести на рабочее поле существующие таблицы с помощью кнопки "Добавить" (Рис. 11).

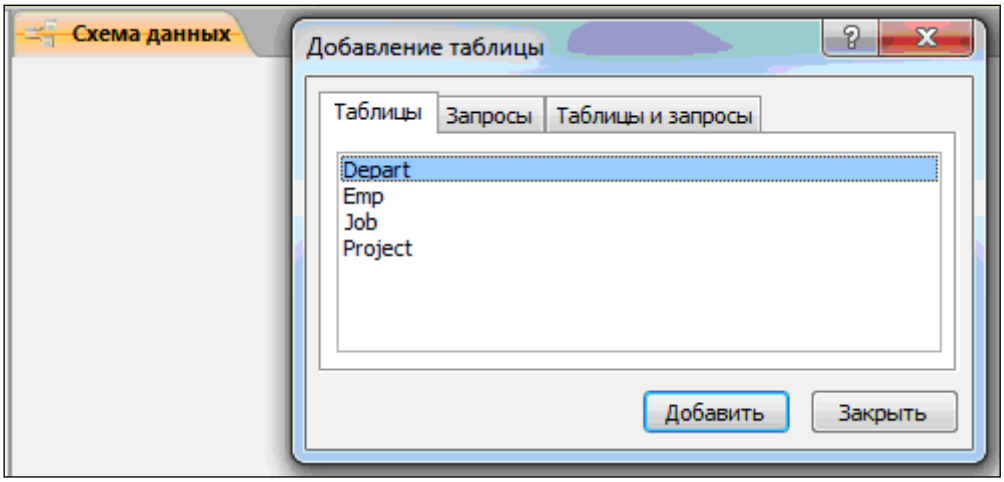


Рис. 11 Перенос таблиц на схему базы данных

После добавления всех таблиц можно устанавливать связи (Рис. 12). Для этого нужно навести курсор "мыши" на ключевое поле родительской таблицы (например, поле `did` таблицы `Depart` на Рис. 12), нажать на левую клавишу и, не отпуская её, перевести курсор на то поле дочерней таблицы, которое являет-

ся внешним ключом (поле `depno` таблицы `Project` на Рис. 12). Затем отпустить клавишу. При этом появится окно "Изменение связей", в котором нужно установить флаг "Обеспечение целостности данных", а затем нажать кнопку "Создать".

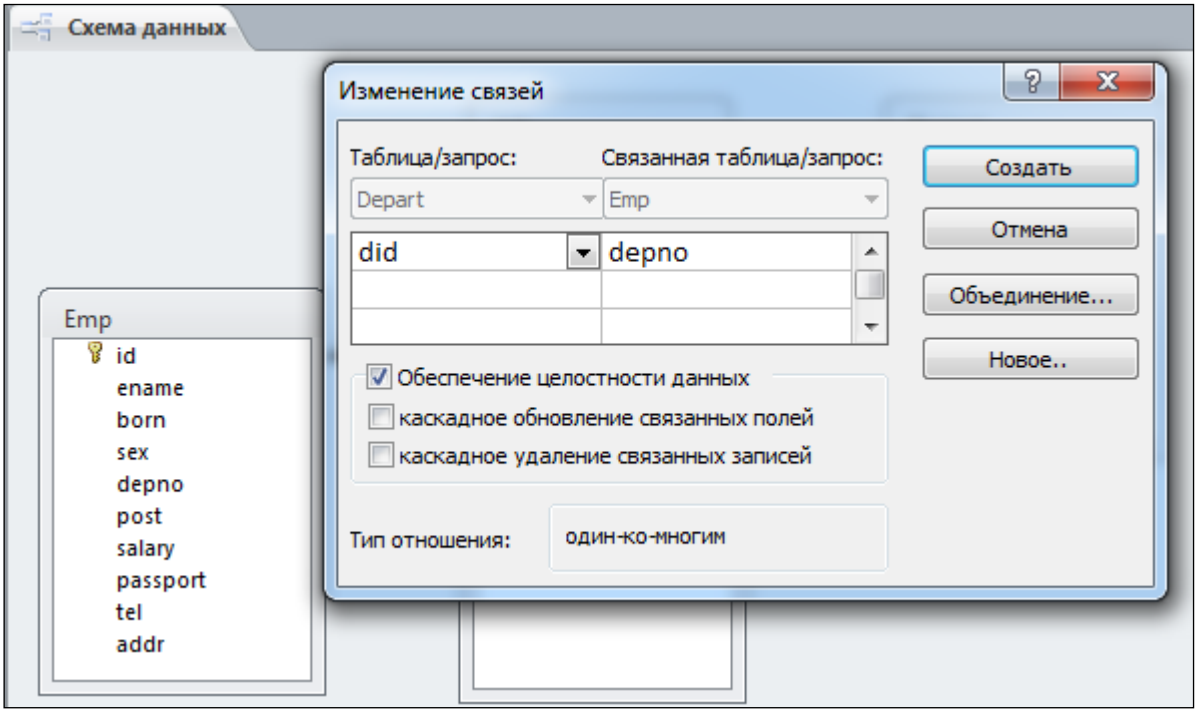


Рис. 12 Создание связей таблиц на схеме базы данных

Аналогично создаются связи между таблицами `Depart – Emp`, `Emp – Job` и `Project – Job`. Окончательная схема базы данных приведена на Рис. 13.

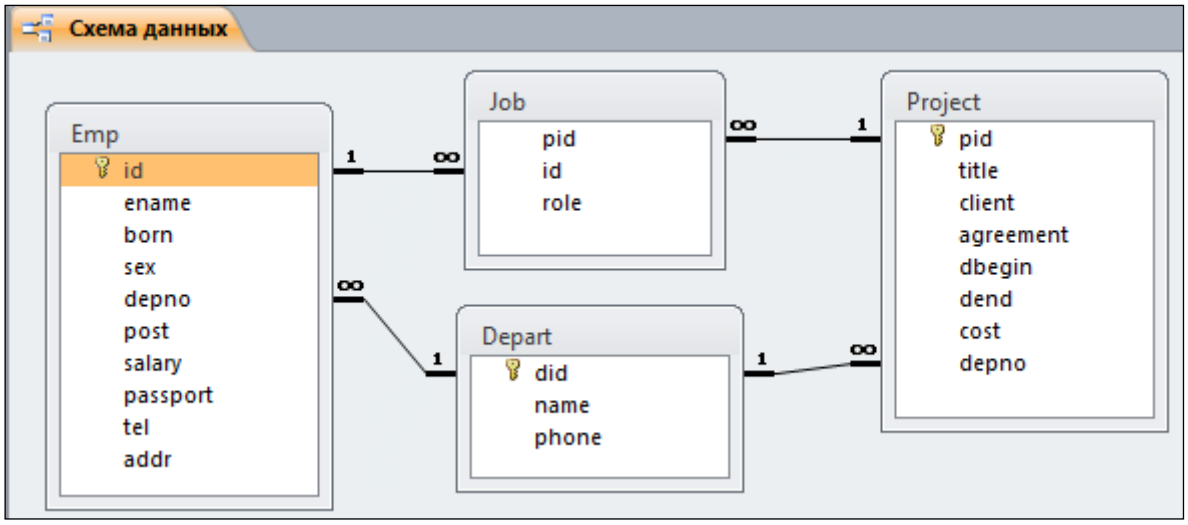


Рис. 13 Окончательная схема базы данных

Внимание! Связываемые поля разных таблиц должны иметь **одинаковый тип данных**, иначе система будет выдавать ошибку при попытке установить связь на схеме БД.

После создания схемы базы данных можно раскрыть закладку "Таблицы" и двойным щелчком "мыши" на имени таблицы вызвать табличный редактор для ввода данных (если они не были введены ранее). Правильность значений внешних ключей всех таблиц будет проверяться системой автоматически.

Итак, вы подключили к системе Access в качестве внешних источников данных таблицы базы данных, созданные при выполнении лабораторной работы №1, и создали схему БД. Эта схема отражает все внешние ключи в ваших таблицах. Теперь можно приступать к написанию интерфейса.

3. Создание экранных форм

Экранные формы позволяют получить доступ к данным в более удобном виде, чем обычная таблица. Формы создаются на закладке ***Создание –> Формы*** с помощью конструктора или мастера форм. Конструктор позволяет в ручном режиме перенести на форму нужные поля из конкретной таблицы, а мастер делает эту работу в полуавтоматическом режиме. Мы воспользуемся мастером.

После запуска мастера создания форм появится окно, представленное на Рис. 14. Из списка таблиц (поле "Таблицы и запросы") выберем таблицу `Depart` (Отделы). После этого в левом нижнем окошке появится список полей выбранной таблицы. Эти поля можно перенести в правое окошко ("Выбранные поля") с помощью кнопок '>' (по одному) или '>>' (все сразу). Затем нужно нажать кнопку "Далее >".

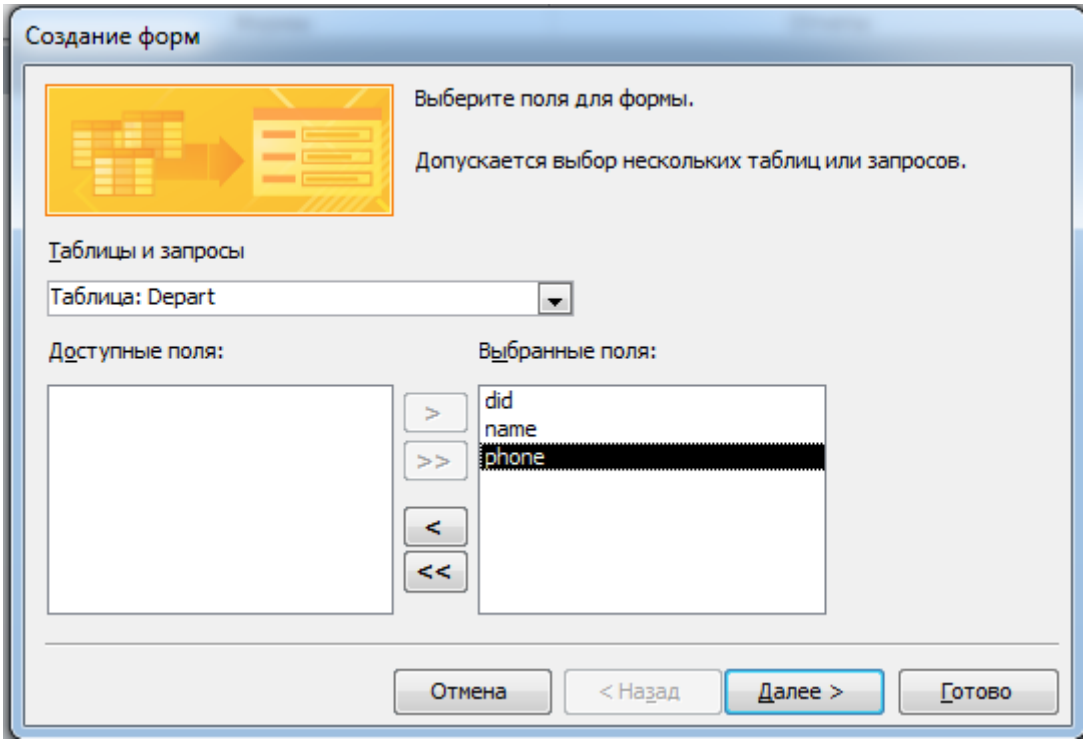


Рис. 14 Создание формы с помощью мастера

На следующем окне можно выбрать внешний вид формы (например, в один столбец), и снова нажать кнопку "Далее >". На третьем окне выберем стиль (стандартный), а на четвертом укажем имя формы (Отделы). После этого форма будет создана (Рис. 15). Переходить от одной записи к другой можно с помощью кнопок перехода, расположенных в нижней части формы.

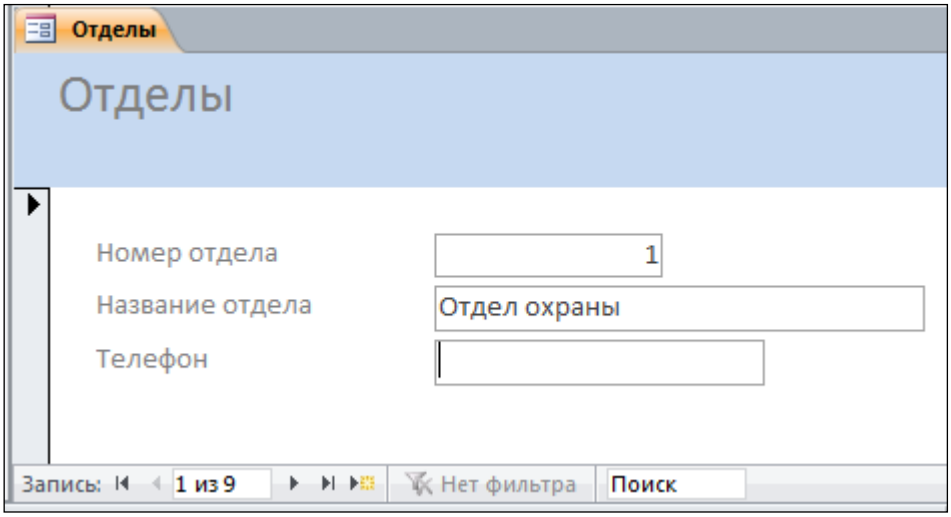


Рис. 15 Форма "Отделы"

Аналогично с помощью мастера форм создадим форму "Сотрудники", только выберем для неё табличную форму представления (укажем внешний вид формы – табличный). Фрагмент полученной формы – на Рис. 16.

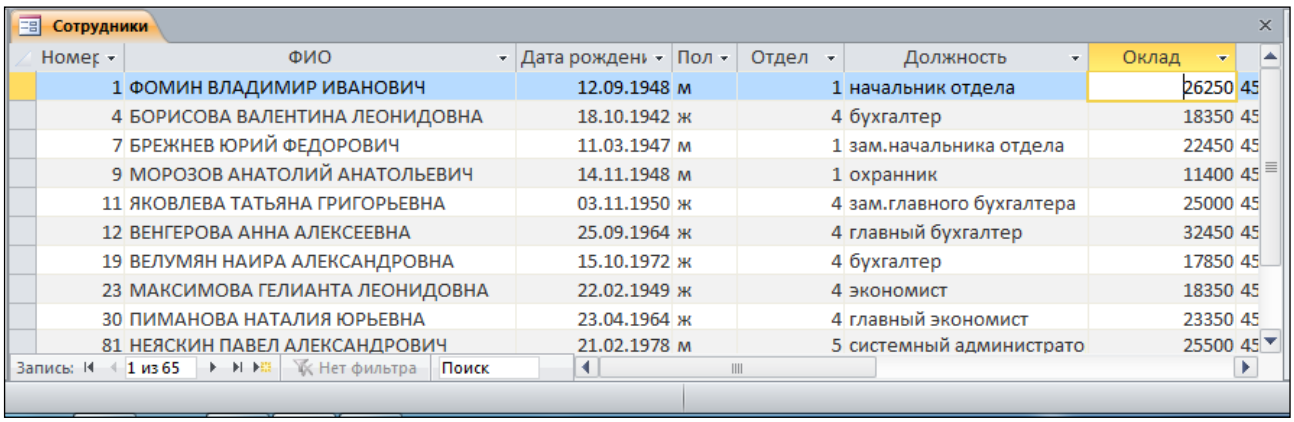


Рис. 16 Форма "Сотрудники" (фрагмент)

Теперь расположим форму "Сотрудники" на форме "Отделы". Войдём в форму "Отделы" в режиме конструктора (пиктограмма  вызывается правой кнопкой "мыши" при наведении курсора на имя формы). Увеличим размер области данных формы (просто "растаскивая" курсором "мыши" границы формы при нажатой левой клавише). На панели инструментов выберем пиктограмму "Подчинённая форма", раскрывая список инструментов (Рис. 17).

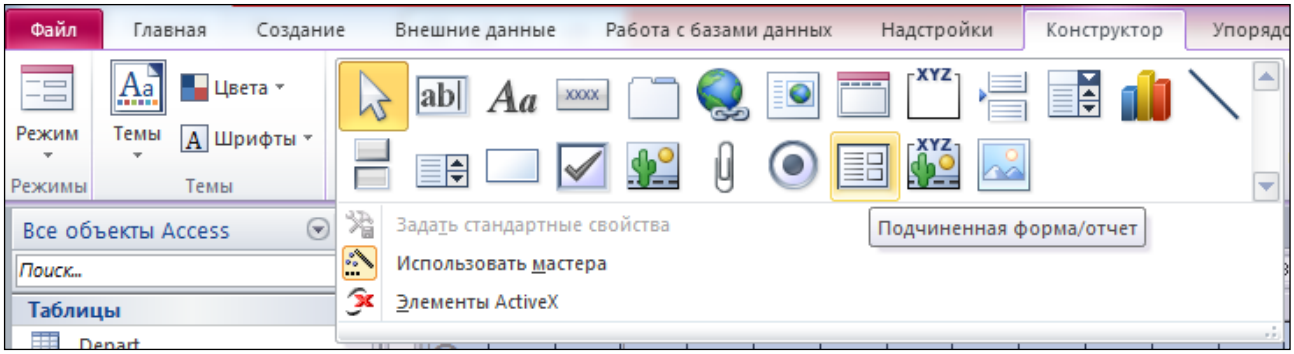


Рис. 17 Выбор пиктограммы "Подчиненная форма"

Далее подведём курсор "мыши" в то место на форме, где будет расположен левый верхний угол подчинённой формы. Нажмём левую клавишу "мыши" и, не отпуская клавишу, переместим курсор в то место, где будет расположен правый нижний угол подчинённой формы. После этого отпустим клавишу. Должен запуститься мастер подчинённых форм, в котором надо выбрать в качестве источника данных для подчинённой формы ранее созданную форму "Сотрудники".

Примечание. Если мастер подчинённой формы не запускается автоматически, то нужно вызвать для подчинённой формы окно ***Свойства*** правой клавишей «мыши», перейти в этом окне на закладку ***Данные***, и выбрать из ниспадающего списка в поле ***Объект-источник*** название подчинённой формы. Связанные поля таблиц при этом должны прописаться автоматически (если они имеют одинаковое название в разных таблицах или ранее была создана схема базы данных). Их также можно указать вручную, но в любом случае связанные поля разных таблиц должны иметь одинаковый тип данных.

Таблица "Сотрудники" (Emp) на схеме базы данных была связана с таблицей "Отделы" (Depart) через поле `depno` (*Номер отдела*). Эта связь сохранится и для созданных форм. Таким образом, в подчинённой форме будут выводиться данные о сотрудниках того отдела, который выведен в основной форме. Например, на Рис. 18 в основной форме – данные об отделе "Администрация", а в подчинённой форме – данные о сотрудниках администрации.

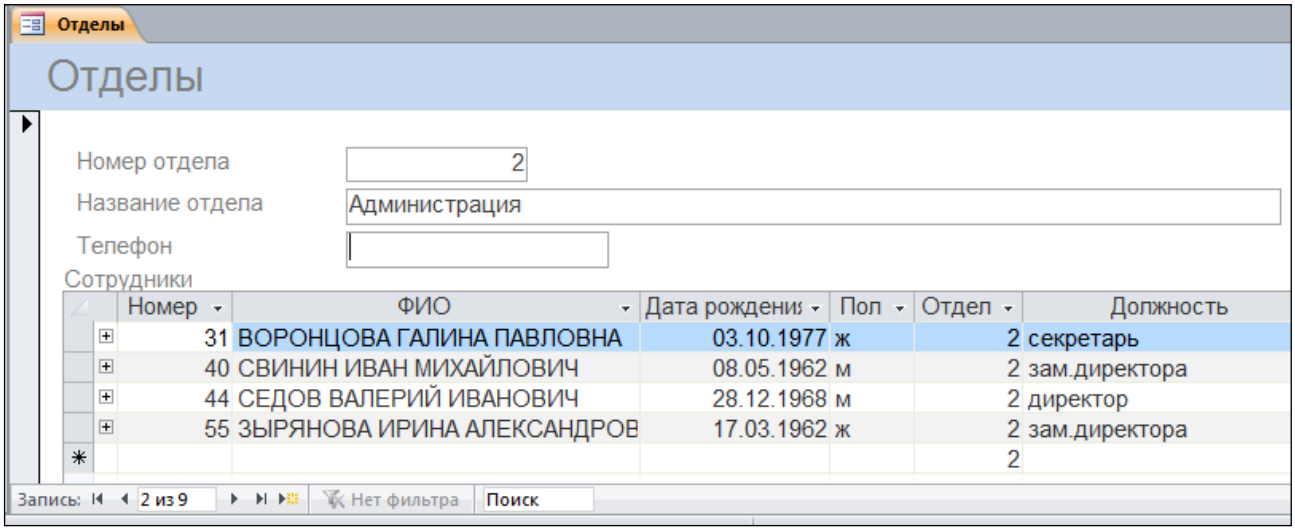


Рис. 18 Форма "Отделы" и подчинённая форма "Сотрудники"

Далее с помощью мастера создадим форму "Участие" на основе таблицы `Job`, выбрав для неё табличную форму представления (т.е. указав внешний вид формы "табличный"). После создания этой формы войдём в неё в режиме конструктора, чтобы изменить представление полей `pid` (*Проект*) и `id` (*Сотрудник*). Дело в том, что в этих полях хранятся идентификаторы проекта и сотрудника – номера, которые являются малоинформативными. Надо сделать так, чтобы вместо номера проекта выводился шифр проекта, а вместо номера сотрудника – его ФИО. Для этого следует навести курсор "мыши" на поле `pid` (*Проект*) и нажать правую клавишу. Появится ниспадающий список, из кото-

рого надо выбрать пункт "Преобразовать элемент в…" –> "Поле со списком" (Рис. 19).

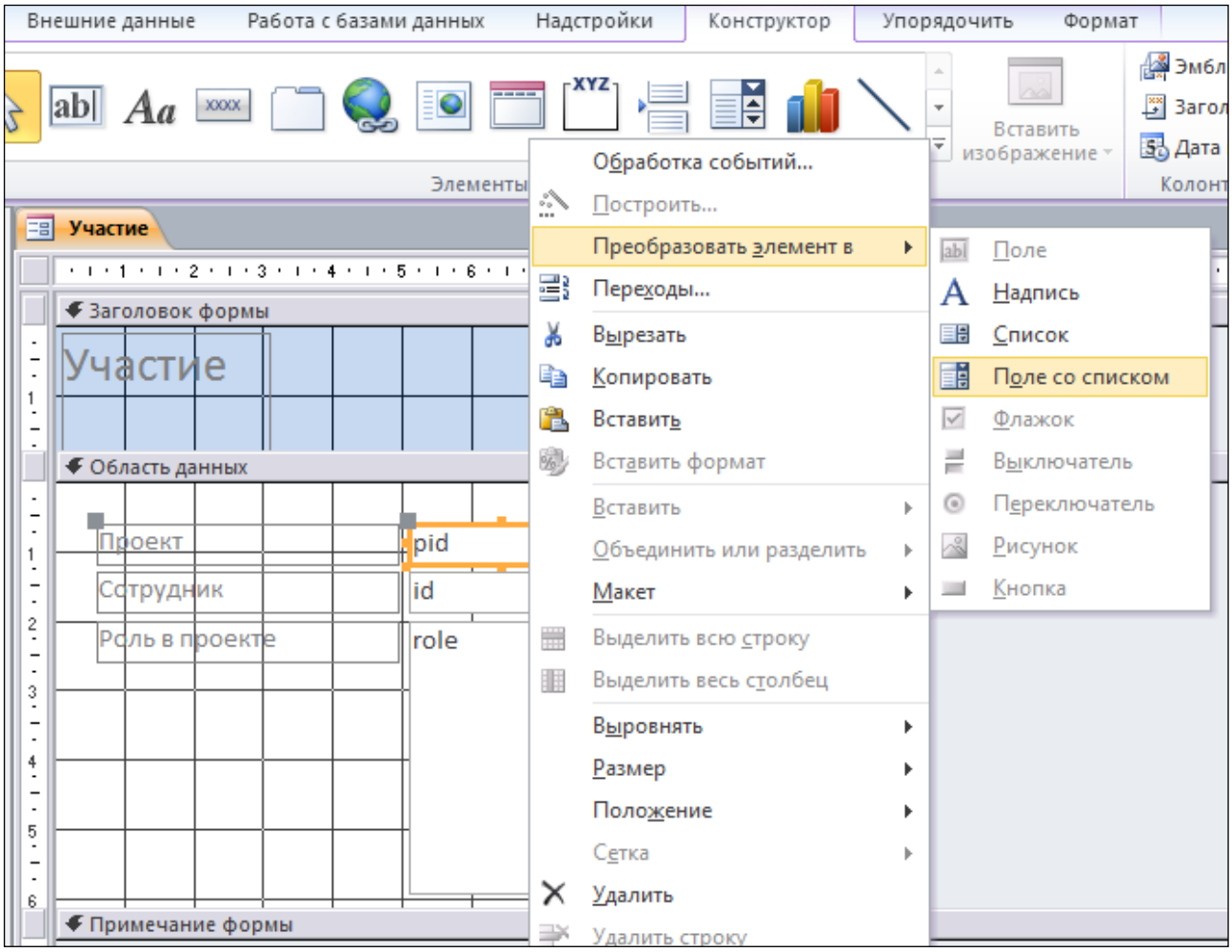


Рис. 19 Создание поля со списком

Затем также по правой клавише "мыши" из ниспадающего списка выбрать пункт "Свойства", а в открывшемся окне – закладку "Данные" (Рис. 20) На этой закладке в поле "Источник строк" нужно записать строку:

```
SELECT Project.agreement, Project.Pid FROM Project;
```

Эту строку можно сформировать в конструкторе запросов. Для этого надо выбрать справа от поля "Источник строк" пиктограмму с тремя точками (…), добавить на окно конструктора запросов таблицу `Project`, а из неё выбрать – 2 поля: `agreement` и `pid`. Первое поле будет отражать суть проекта (его шифр), а второе – поле связи. При этом значение "Присоединённый столбец" должно быть равно 2. (Более подробно работа с конструктором запросов описана далее).

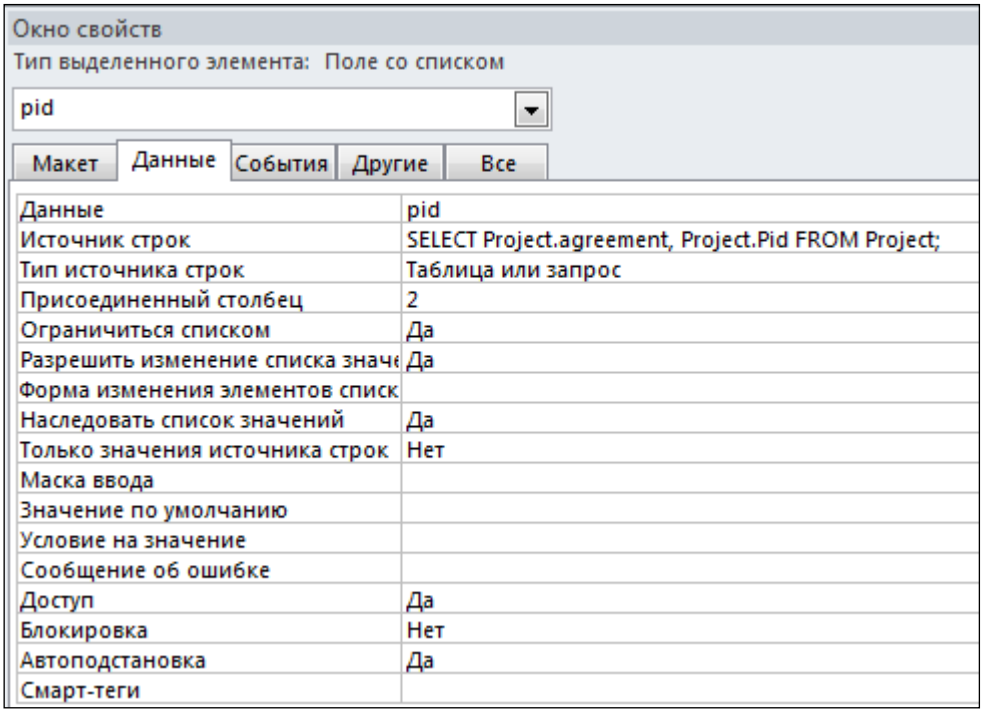


Рис. 20 Указание источника строк для поля со списком на закладке "Свойства"

Затем аналогично нужно изменить поле `Id` (*Сотрудник*) на поле со списком и присоединить к нему таблицу `Emp`. Значение "Присоединённый столбец" на закладке "Свойства" –> "Данные" должно быть также равно 2, а значение поля "Источник строк" при этом будет таким:

```
SELECT Emp.ename, Emp.id FROM Emp;
```

После этого необходимо сохранить сделанные изменения (пиктограмма ) и вернуться в обычный режим (пиктограмма ).

Полученная в итоге форма "Участие" приведена на Рис. 21.

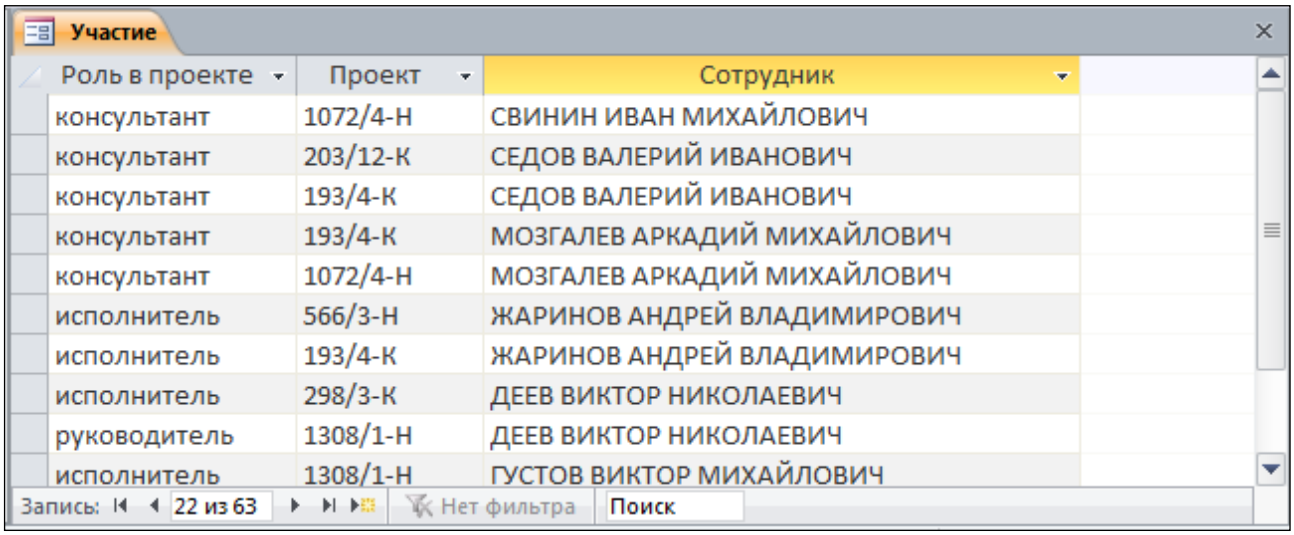


Рис. 21 Форма "Участие" (фрагмент)

Далее нужно войти в форму "Сотрудники" в режиме конструктора и добавить на неё в качестве подчинённой форму "Участие", связав их по полю `id` (*Сотрудник*). Теперь можно из формы "Сотрудники" (а также из формы "Отде-

лы") получать информацию о том, в каких проектах участвует сотрудник. Для этого надо подвести курсор "мыши" к значку '+', который появится слева от записи на форме "Сотрудники", и нажать левую клавишу "мыши" (Рис. 22).

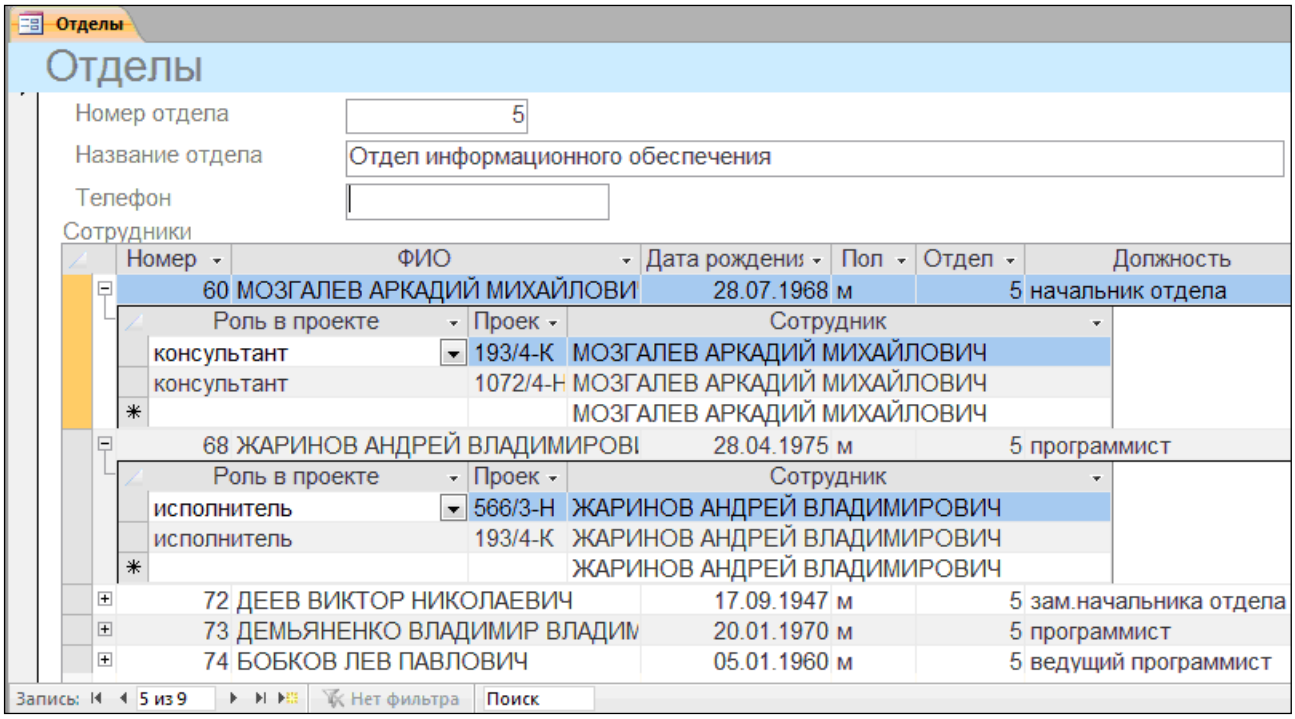


Рис. 22 Форма "Отделы" с двумя подчинёнными формами

Далее надо создать форму "Проекты" для таблицы `Project` и разместить на ней подчинённую форму "Участие", связав их по полю "Проект" (`pid`). С помощью этой формы (Рис. 23) пользователь получит возможность просматривать списки участников каждого проекта.

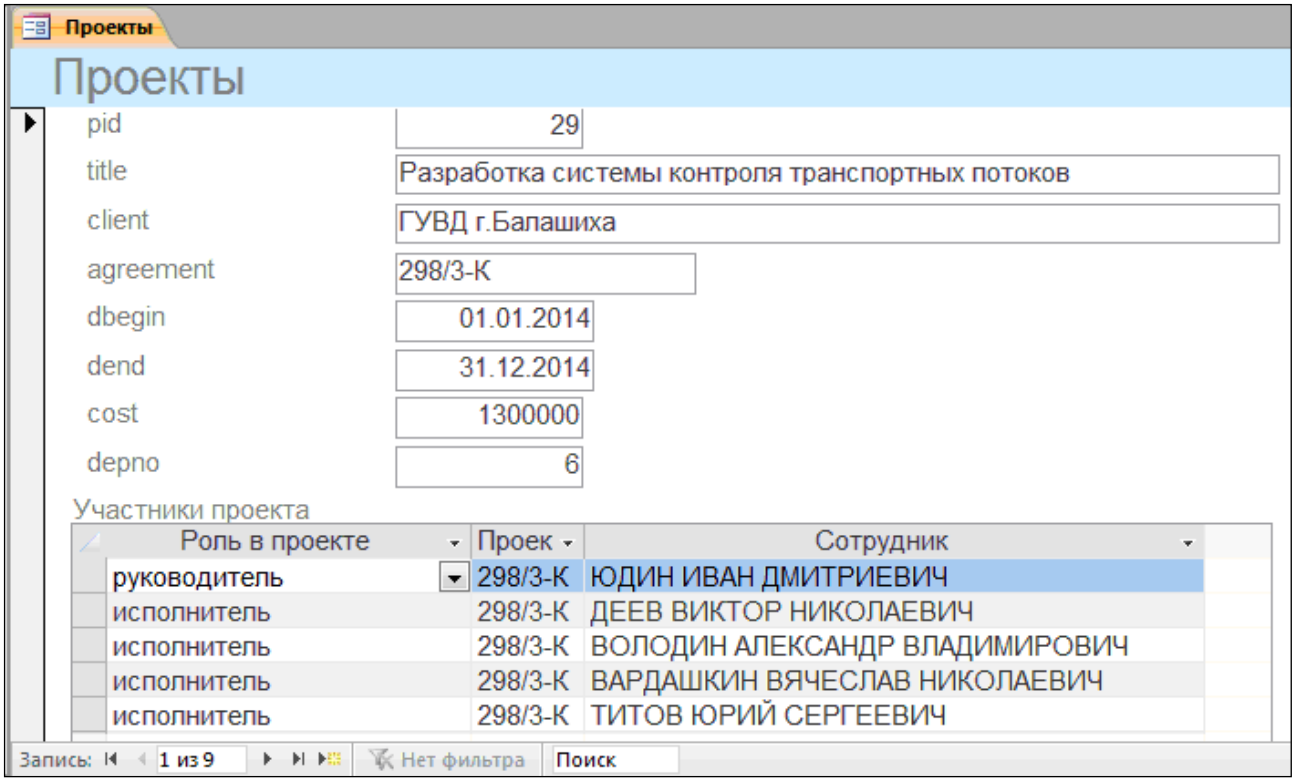


Рис. 23 Форма "Проекты" с подчинённой формой "Участники"

Форму можно редактировать вручную. Для этого надо войти в неё в режиме конструктора. Изменять местоположение и значение параметров элементов формы можно с помощью "мыши" или путём изменения свойств элемента на закладке "Свойства". Для вызова этой закладки надо навести курсор "мыши" на элемент формы и нажать правую клавишу "мыши".

Таким образом, созданные вами экранные формы позволяют просматривать данные всех таблиц вашей БД, каждое поле внешнего ключа определяется как "Поле со списком". Минимум одна экранная форма содержит подчиненную экранную форму.

4. Создание запросов и отчётов

Запросы создаются на соответствующей закладке с помощью конструктора или мастера запросов. Конструктор имеет более широкие возможности по формированию запросов, поэтому мы воспользуемся им.

После запуска конструктора запросов появится окно, представленное на Рис. 24. В нём можно выбрать одну или несколько таблиц, входящих в запрос (с помощью кнопки Добавить).

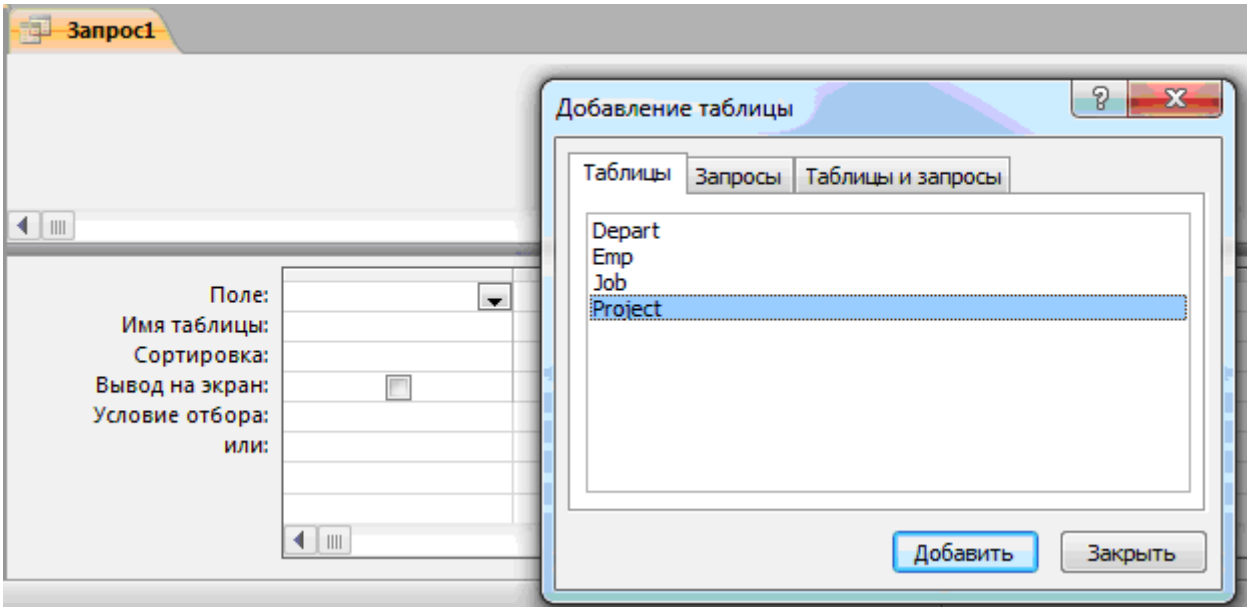


Рис. 24 Выбор таблицы для запроса

Мы выберем таблицу `Project` и создадим запрос "Текущие проекты". Мы хотим получить в этом запросе все поля исходной таблицы, поэтому в первом столбце из ниспадающего списка мы выберем вариант `Project.*` (Рис. 25). Но нам нужно ещё указать условие выбора: текущая дата должна находиться между датами начала и окончания проекта. Для получения текущей даты мы воспользуемся функцией `now()`. Во втором столбце мы выберем поле `dbegin` (*Начало проекта*) и поставим условие `<=now()`, а в третьем столбце – поле `dend` (*Окончание проекта*) и поставим условие `>=now()`. Для двух последних полей надо снять флаг "вывод на экран" (Рис. 25), чтобы эти поля не выводились по два раза.

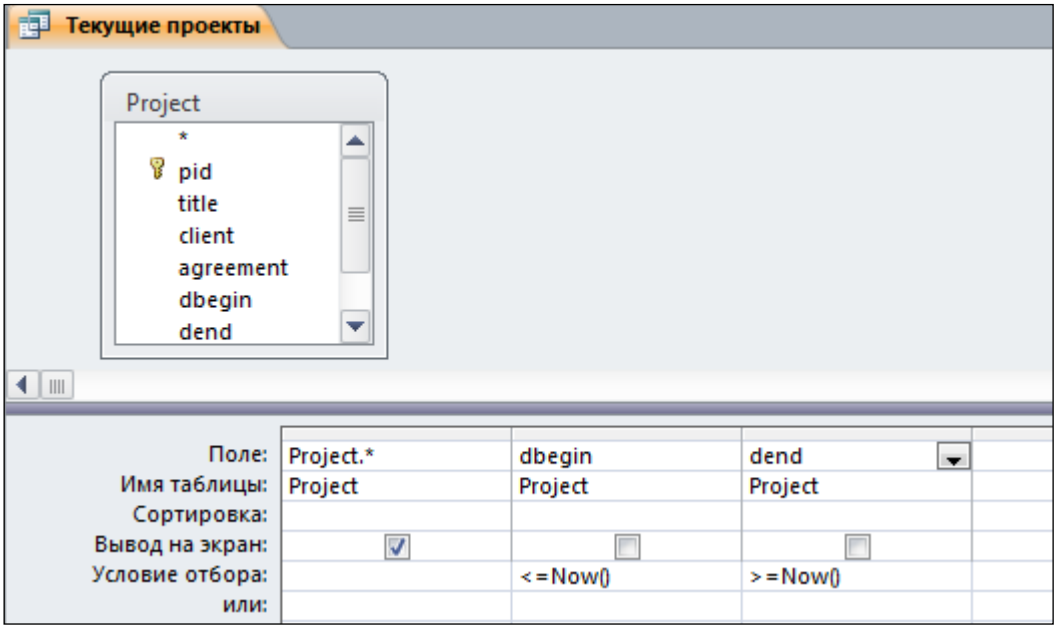


Рис. 25 Выбор полей запроса и ввод условий отбора

Затем надо сохранить текст запроса под именем "Текущие проекты" (Рис. 26) и закрыть окно конструктора.

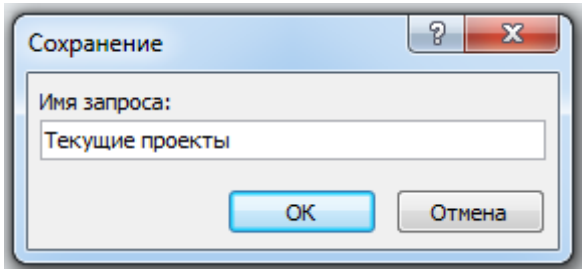
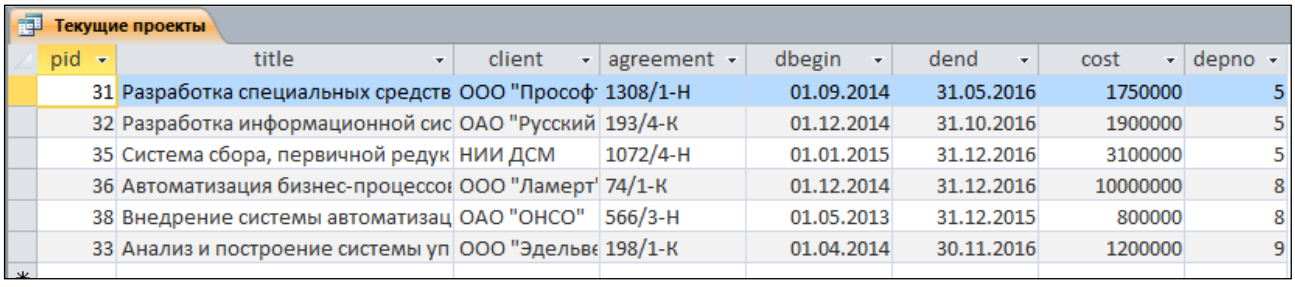


Рис. 26 Сохранение запроса "Текущие проекты"

После этого на закладке "Запросы" появится созданный нами запрос, который можно выполнить двойным нажатием левой клавиши "мыши" на имени этого запроса. Результат выполнения запроса "Текущие проекты" приведён на Рис. 27.



pid	title	client	agreement	dbegin	dend	cost	depno
31	Разработка специальных средств	ООО "Прософ	1308/1-Н	01.09.2014	31.05.2016	1750000	5
32	Разработка информационной сис	ОАО "Русский	193/4-К	01.12.2014	31.10.2016	1900000	5
35	Система сбора, первичной редук	НИИ ДСМ	1072/4-Н	01.01.2015	31.12.2016	3100000	5
36	Автоматизация бизнес-процессов	ООО "Ламерт"	74/1-К	01.12.2014	31.12.2016	10000000	8
38	Внедрение системы автоматизац	ОАО "ОНСО"	566/3-Н	01.05.2013	31.12.2015	800000	8
33	Анализ и построение системы уп	ООО "Эдельве	198/1-К	01.04.2014	30.11.2016	1200000	9

Рис. 27 Результат выполнения запроса "Текущие проекты"

Затем создадим запрос "Руководители проектов". Для этого запустим конструктор запросов и выберем 3 таблицы: `Emp`, `Project` и `Job`. Из первой таблицы нам понадобится поле `ename` (*ФИО сотрудника*), из второй – поля `pid` (*Номер проекта*) и `title` (*Название проекта*), а третья будет служить для

связи двух первых таблиц и проверки условия `role='руководитель'` (Рис. 28).

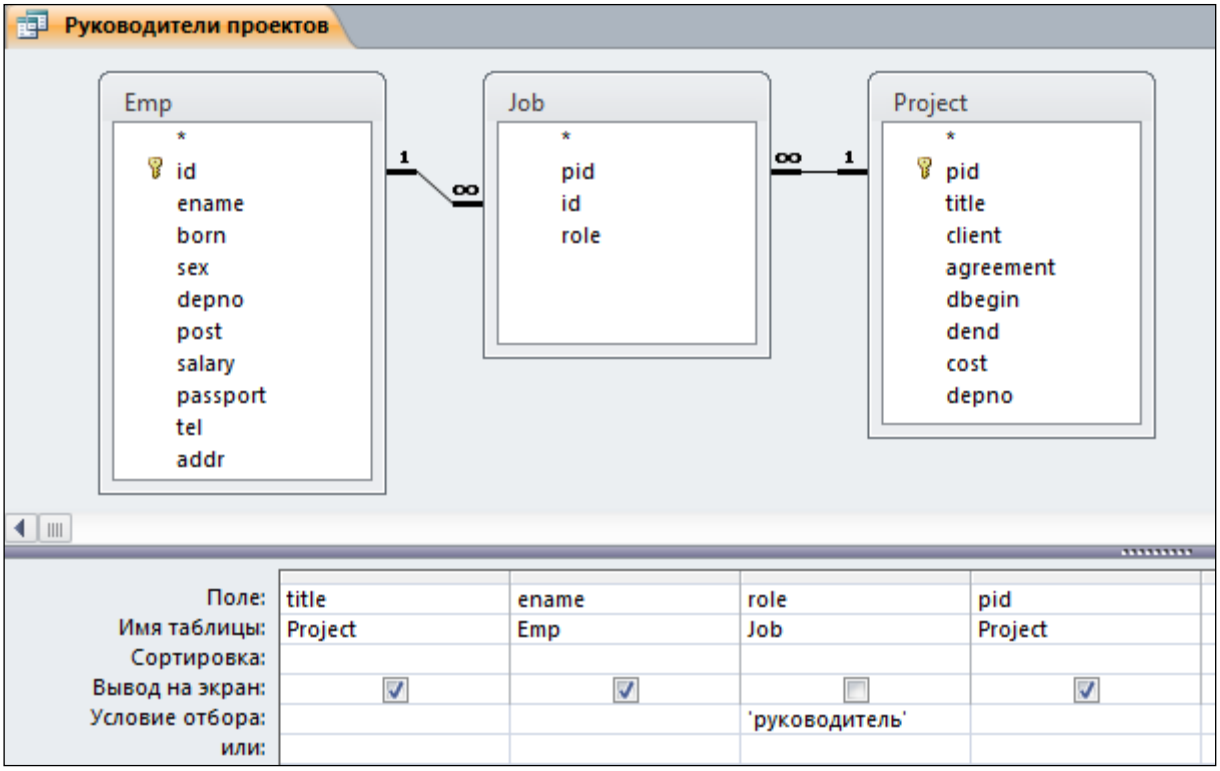


Рис. 28 Создание запроса "Руководители проектов"

Обратите внимание: поле `role` мы не выводим, т.к. его значение для всех строк результата будет одинаковым.

Если таблиц две и более и эти таблицы связаны друг с другом на схеме БД, то эти связи автоматически будут учитываться в запросе как условие соединения таблиц. При отсутствии таких связей результатом запроса будет декартово произведение исходных таблиц, т.е. все возможные комбинации всех строк исходных таблиц. Результат выполнения запроса "Руководители проектов" приведён на Рис. 29.

Руководители проектов

title	ФИО
Внедрение системы автоматизации административной деятельности	АНОХИН НИКОЛАЙ ИВАНОВИЧ
Анализ и построение системы управления вентиляционными установками	ГРИГОРЬЕВА МАРИЯ ФЕДОСЕЕВНА
Разработка системы контроля транспортных потоков	ЮДИН ИВАН ДМИТРИЕВИЧ
Разработка специальных средств и систем защиты информации	ДЕЕВ ВИКТОР НИКОЛАЕВИЧ
Система автоматизированного управления предприятием	РОЗЕНБЛИТ НАУМ ШМУЛЬЕВИЧ
Разработка информационной системы интеллектуальной поддержки принятия	БОБКОВ ЛЕВ ПАВЛОВИЧ
Система сбора, первичной редукции и архивизации данных о предприятиях сыр	АШУРОВ МИХАИЛ ДАВЫДОВИЧ
Автоматизация бизнес-процессов и документооборота на основе программной	ОЛИВЕТСКИЙ НИКОЛАЙ БОРИСОВИЧ
Создание автоматизированной системы "Сервисная Диспетчерская (Service Des	ПОТАПОВ АЛЕКСАНДР НИКОЛАЕВИЧ

Рис. 29 Результат запроса "Руководители проектов" (фрагмент)

Если для этого запроса в меню "Конструктор" выбрать пункт "Режим SQL" (Рис. 30), то можно увидеть текст запроса:

```
SELECT Emp.ENAME, Project.title, Project.pid
     FROM Project INNER JOIN (Emp INNER JOIN Job ON
          Emp.ID = Job.id) ON Project.Pid = Job.pid
WHERE (((Job.role)='руководитель'));
```

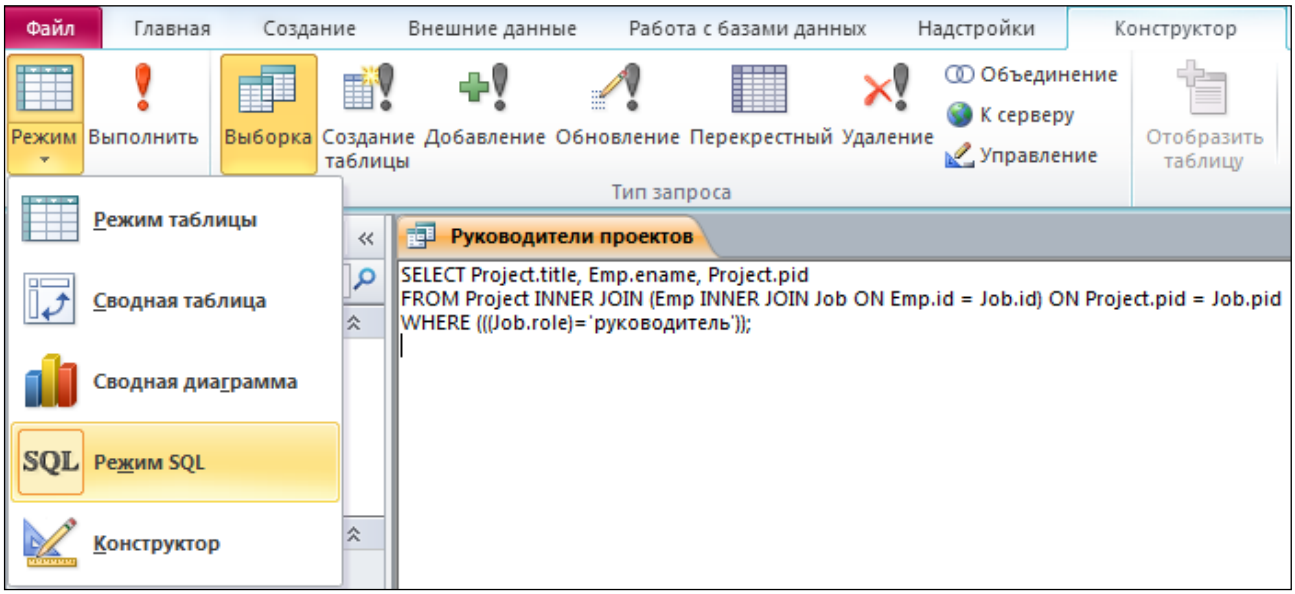


Рис. 30 Выбор пункта "Режим SQL" для запроса

В тексте запроса ключевые слова `INNER JOIN` как раз означают операцию внутреннего соединения таблиц, которая была добавлена системой автоматически в соответствии со схемой БД.

Примечание. В режиме SQL текст запроса можно исправлять и дополнять вручную. Можно изменить список выбора, список таблиц, дополнить запрос необходимыми условиями, в частности, условиями соединения таблиц, для которых на схеме БД отсутствуют связи.

Созданные запросы можно наравне с таблицами использовать для создания отчётов и экранных форм.

Отчёты позволяют определить состав и внешний вид данных, выводимых на печать. Отчёты создаются на закладке "Отчеты" с помощью конструктора или мастера отчётов. Конструктор позволяет в ручном режиме перенести на отчёт нужные поля из конкретной таблицы (таблиц), а мастер делает эту работу в полуавтоматическом режиме. Мы воспользуемся мастером.

После запуска мастера создания отчётов появится окно, представленное на Рис. 31. Из списка таблиц (поле "Таблицы и запросы") выберем таблицу `Depart` (Отделы). После этого в левом нижнем окошке появится список полей выбранной таблицы. Из этих полей перенесём в правое окошко ("Выбранные поля") с помощью кнопки '>' (по одному) поле `name` (*Название отдела*).

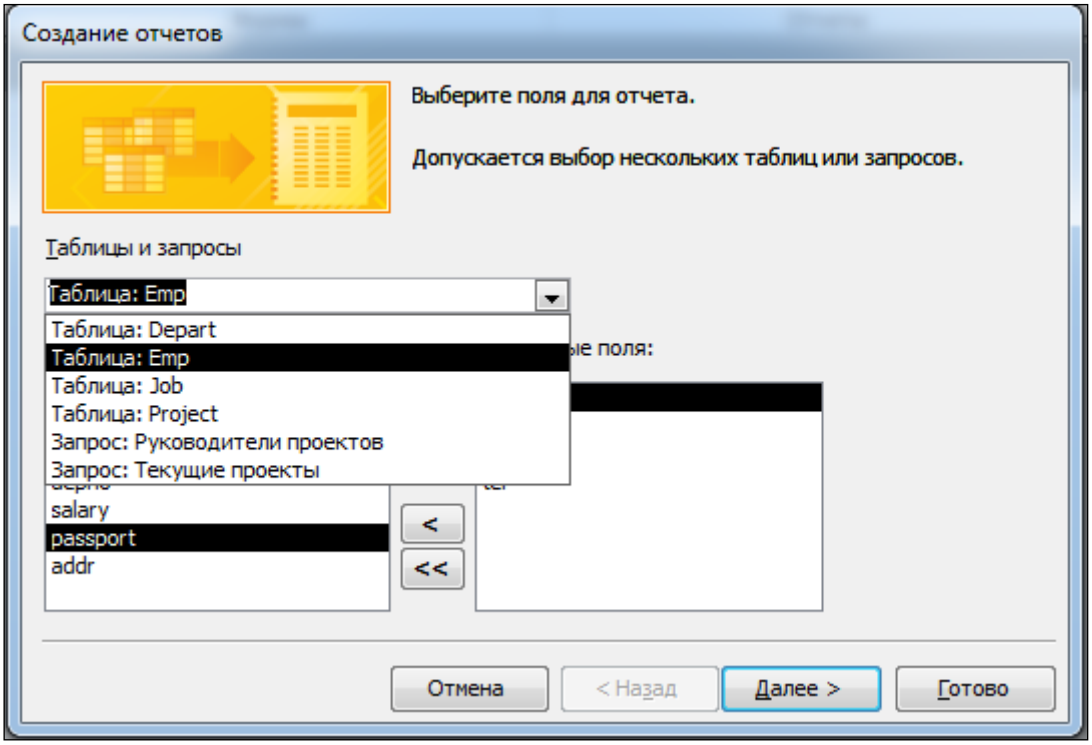


Рис. 31 Мастер создания отчётов, выбор таблицы

Затем нужно выбрать другую таблицу – `Emp` (Сотрудники), выбрать из неё поля `ename, post, tel` (*ФИО сотрудника*, *Должность*, *Телефон*) и нажать кнопку "Далее >" (Рис. 32). Эти таблицы связаны на схеме базы данных, поэтому эти связи будут учтены в отчёте автоматически. Это означает, что для каждого отдела будут выводиться данные о сотрудниках именно этого отдела.

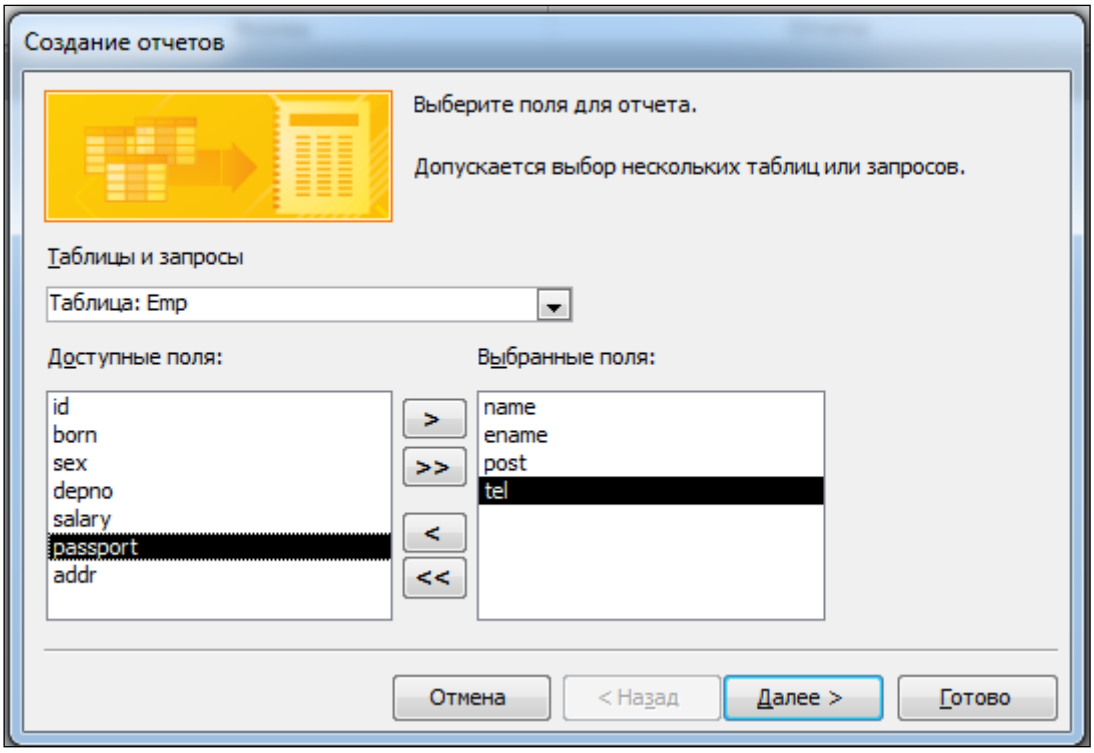


Рис. 32 Выбор полей для отчёта кнопкой [>]

В следующем окне выберем вид представления данных (Рис. 33).

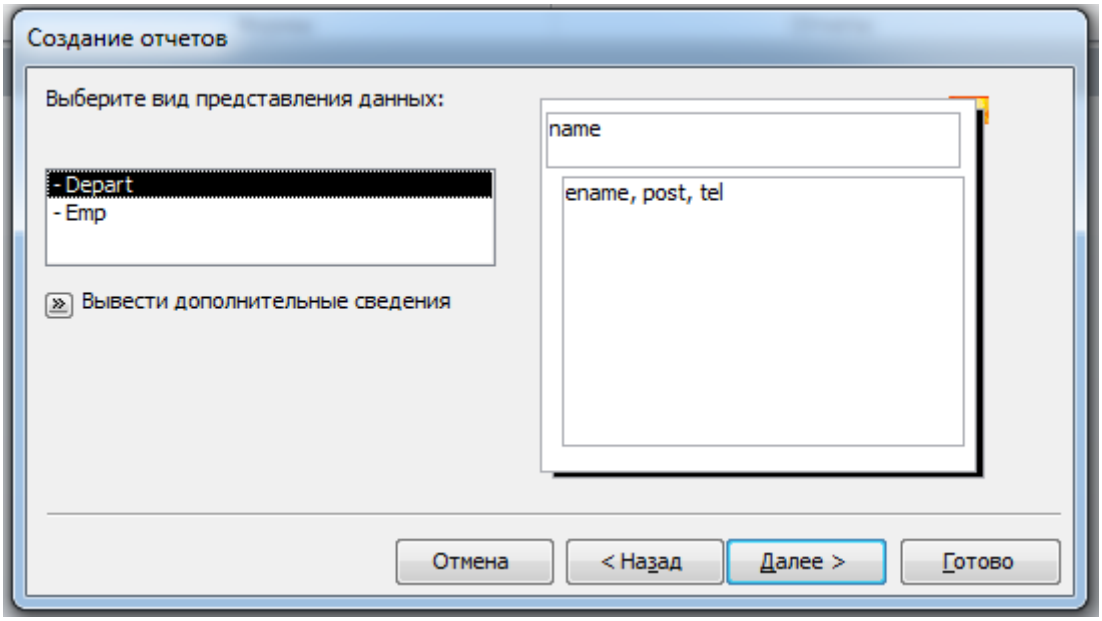


Рис. 33 Выбор вида представления данных

Окно "Уровни группировки" пропустим (кнопка "Далее >"). В качестве поля для сортировки выберем `ename`, чтобы сотрудники выводились в алфавитном порядке (Рис. 34).

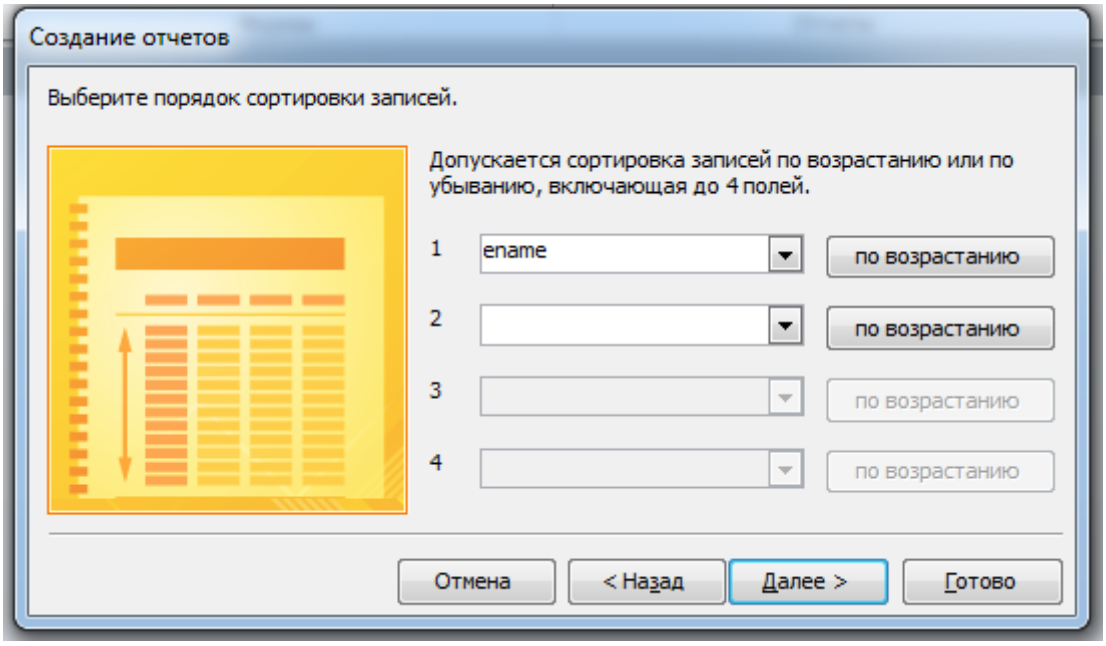


Рис. 34 Выбор порядка сортировки данных

В следующем окне выберем ступенчатый макет, затем – деловой стиль. Зададим имя отчета – "Сотрудники отделов". Полученный отчёт приведён на Рис. 35.

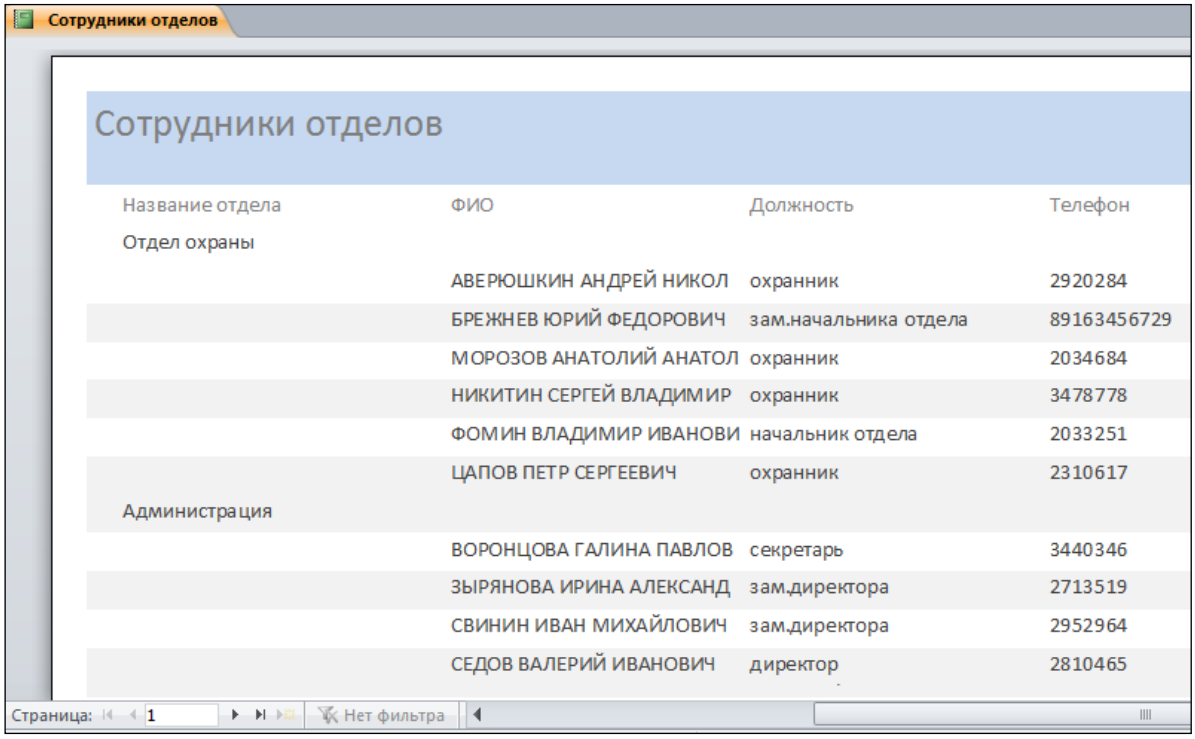


Рис. 35 Общий вид полученного отчёта "Сотрудники по отделам" (фрагмент)

Создадим другой отчет. Запустим мастера создания отчётов, выберем из таблицы `Project` поля `agreement` и `title` (*Шифр проекта* и *Название*), из таблицы `Depart` поле `name` (*Название отдела*), из таблицы `Emp` поле `ename` (*ФИО сотрудника*), из таблицы `Job` поле `role` (*Роль*). Все эти таблицы связаны на схеме базы данных, эти связи будут учтены в отчёте автоматически.

Сгруппируем данные по уровням так, как показано на Рис. 36 (с помощью кнопки '>' постепенно перенося поля из левой части в правую).

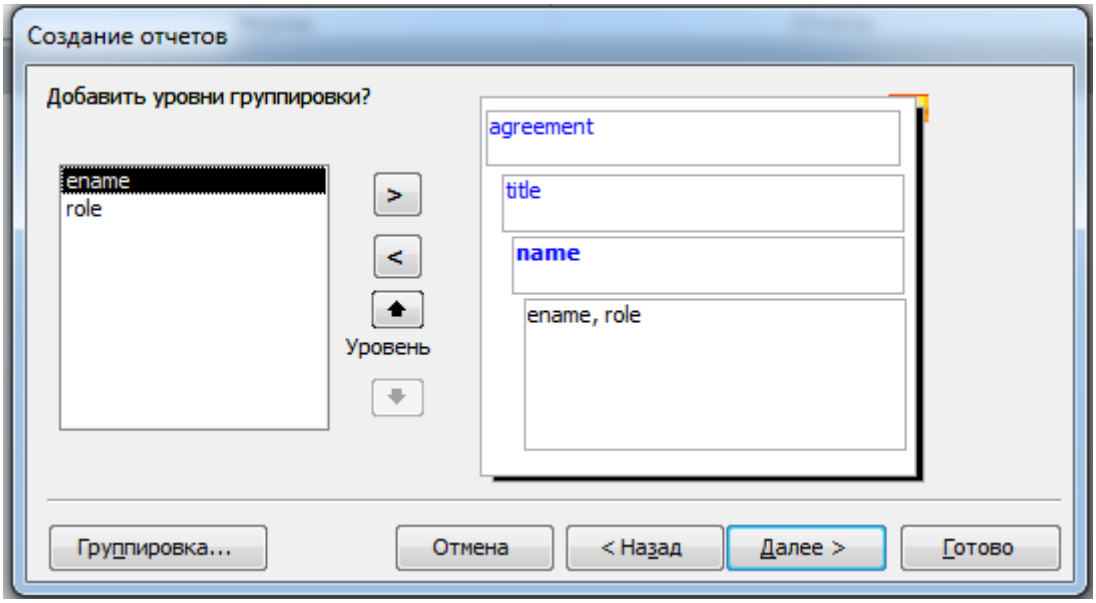


Рис. 36 Добавление уровней группировки

Установим сортировку по двум полям: по полю `role` по убыванию, по полю `ename` – по возрастанию (Рис. 37).

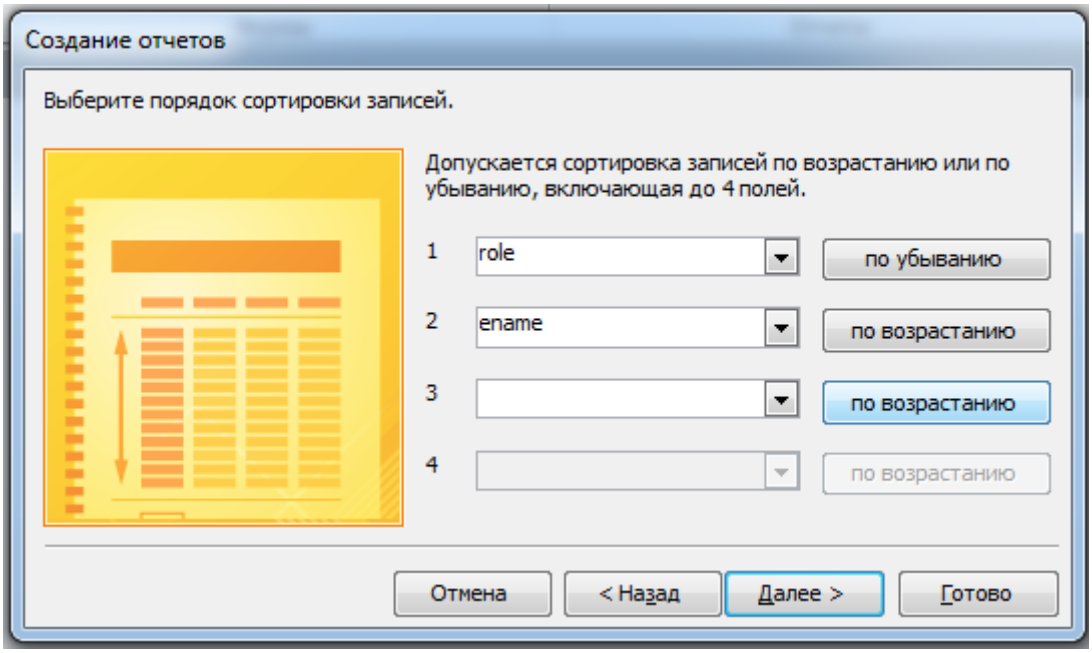


Рис. 37 Добавление сортировки Добавление сортировки

В следующем окне выберем тип макета "Структура" и зададим имя отчета – "Участие в проектах". Полученный отчёт приведён на Рис. 38.

Рис. 38 Отчёт "Участие в проектах" (фрагмент)

В том случае, если значение отдельного поля (например, поля *Название проекта*) выходит за отведённые границы, оно показывается не полностью. Тогда следует войти в отчёт в режиме конструктора и вручную увеличить размер области, отведённой для этого поля (при нажатой левой клавише "мыши" правый нижний угол поля перенести вправо или ниже).

Созданный отчёт можно открыть и вывести на экран или на печать (пиктограмма "Печать" в основном меню).

Итак, теперь для каждой из таблиц БД создан хотя бы один запрос и один отчет; минимум два запроса и два отчета включают более одной таблицы (соединение таблиц или подзапрос).

5. Создание поисковой и кнопочной форм

Мы будем создавать поисковую форму для организации вывода данных в соответствии с поисковым критерием. Завершит нашу работу над приложением создание кнопочной формы, которая будет основным интерфейсом нашего приложения.

Для начала создаем форму «Поиск», выполняющую выбор данных по полям, в которых мы будем указывать какие-либо значения.

Прежде чем приступить к созданию формы, нужно определиться, какую таблицу или какой запрос нам следует использовать. Создадим запрос, который будет выполняться для поиска данных – «ЗапросПроекты» (Рис. 39):

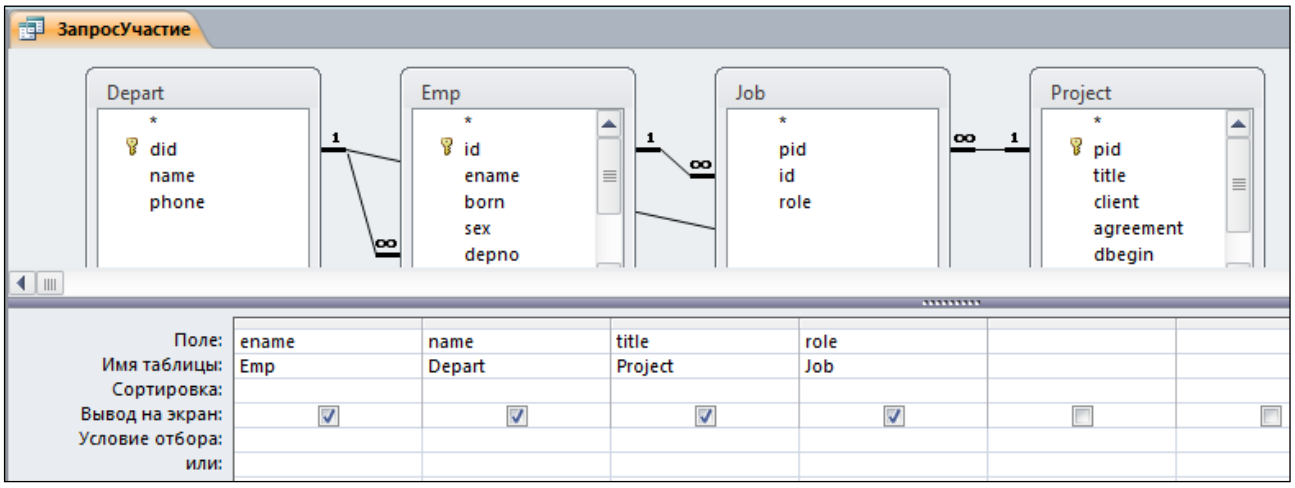


Рис. 39 Создание запроса «ЗапросПроекты»

В SQL-форме этот запрос выглядит так:

```
SELECT Emp.ename, Depart.name, Project.title, Job.role
  FROM ((Depart INNER JOIN Emp ON Depart.did = Emp.depno)
  INNER JOIN Job ON Emp.id = Job.id) INNER JOIN Project
  ON (Project.pid = Job.pid)
  AND (Depart.did = Project.depno);
```

Он выдаёт список участников проектов с указанием названия отдела, проекта и роли сотрудника в проекте.

Теперь нам нужно вставить этот запрос в форму «Поиск» в качестве подчиненной формы. На панели инструментов выберем пиктограмму "Подчинённая форма". Далее подведём курсор "мыши" в то место на форме, где будет расположен левый верхний угол подчинённой формы. Нажмём левую клавишу "мыши" и, не отпуская клавишу, переместим курсор в то место, где будет расположен правый нижний угол подчинённой формы. После этого отпустим клавишу. Должен запуститься мастер подчинённых форм, в котором надо выбрать в качестве источника данных для подчинённой формы ранее созданный запрос "ЗапросПроекты". Задаём имя этой формы – ПФ_ЗапросПроекты.

Сейчас в нашей подчинённой форме выводится вся информация по запросу. Пришло время создать поля для ввода критериев поиска и кнопку «Найти» для выполнения поиска по таблицам.

Переходим в режим «Конструктор». На панели инструментов выберем пиктограмму "Поле". Создадим новое поле с подписью «ФИО», а в свойствах поля (вкладка «Другие») указываем имя этого поля – `sfio`.

Теперь создаём кнопку «Найти». Отменяем работу мастера, дальнейшую настройку мы будем проводить вручную. В свойствах кнопки указываем имя (вкладка «Другие») «КнЗапрос». Так же во вкладке «Макет» в поле «Подпись» пишем название кнопки (Найти), как она будет подписана на форме (Рис. 40).

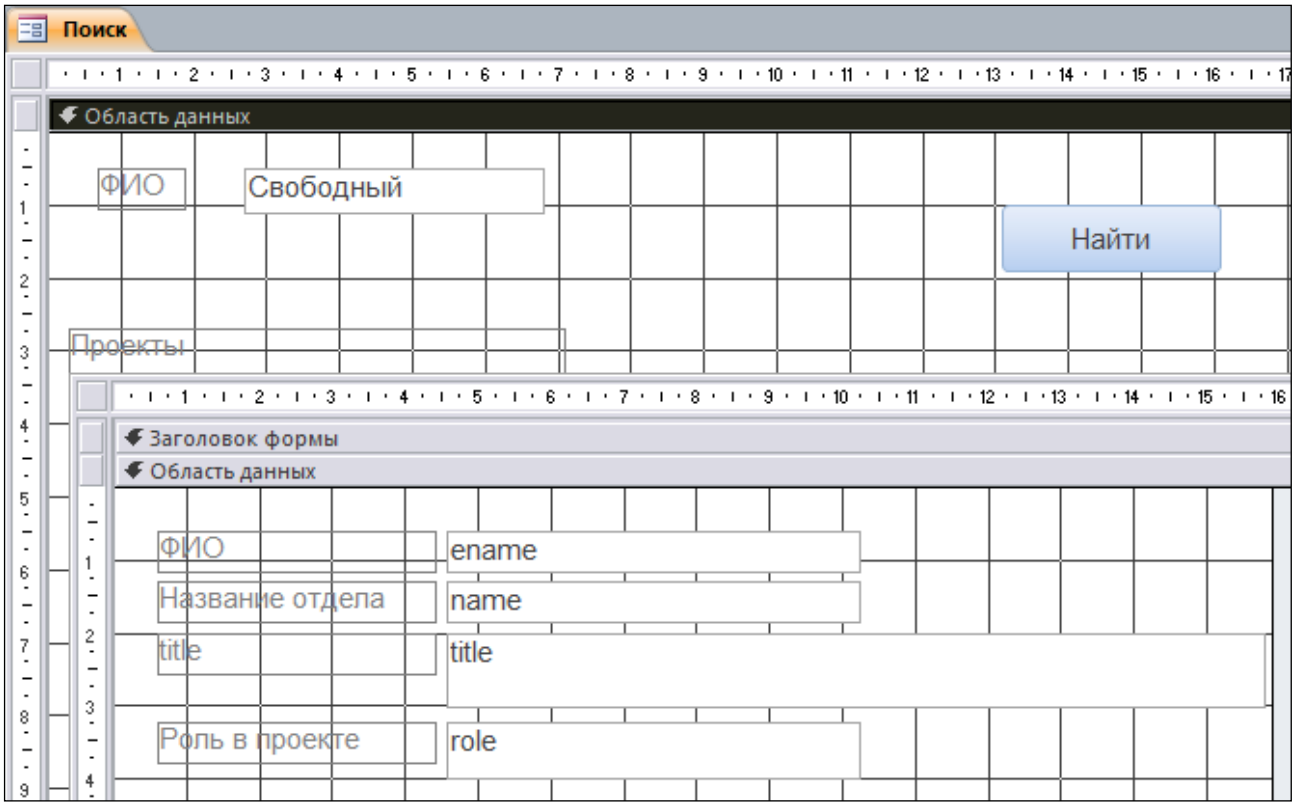


Рис. 40 Создание кнопки «Найти» и поля для ввода данных

Осталось связать наше поле `sfio` с кнопкой «Найти» так, чтобы производился поиск по значению, введенному в поле `sfio`. В свойствах кнопки находим вкладку «События» и ищем строку «Нажатие кнопки». Выбираем вариант «Программа» и попадаем в редактор Visual Basic. При этом там автоматически создаётся заготовка для подпрограммы, которая будет вызываться при нажатии кнопки «КнЗапрос»:

```
Private Sub КнЗапрос_Click()
End Sub
```

Для того чтобы наша кнопка заработала, достаточно вписать следующее:

```
ПФ_ЗапросПроекты.Form.RecordSource = "SELECT * FROM
  ЗапросПроекты WHERE ENAME like '*" & sfio.Value & "*'"
```

где `ПФ_ЗапросПроекты` – имя объекта (подчиненной формы), `Form` – тип объекта (форма), `RecordSource` – свойство объекта, `sfio.Value` – значение, которое мы ввели в поле `sfio`. Обратите внимание, что вместо привычного символа расширения '%' здесь используется символ '*'.

Так как у нас пока одно поле, по которому мы производим поиск, то этот вариант нас вполне устраивает. Но правильнее будет сделать проверку поля перед началом поиска, является ли поле пустым:

```
If (Not IsNull(sfio.Value) And(Trim(sfio.Value) <> "") Then
  S = "(ENAME LIKE '*" & sfio.Value & "*') "
  If condstr <> "" Then
  condstr = condstr & " AND " & S
  Else
    condstr = S
  End If
End If

If condstr <> "" Then condstr = "WHERE " & condstr
```

Обратите внимание: Visual Basic – интерпретируемый язык, поэтому каждая строка в нем должна быть законченной командой. И если команда `IF` занимает одну строку, то операторные скобки `END IF` не требуются.

Надо в начало подпрограммы добавить описание переменных `S` и `condstr`. И поменять наш запрос:

```
Dim condstr, S
condstr = ""
ПФ_ЗапросПроекты.Form.RecordSource = "SELECT * FROM
  ЗапросПроекты " & condstr
```

Попробуем ввести какие-либо данные в поле ФИО и нажать на кнопку «Найти», результат – на Рис. 41.

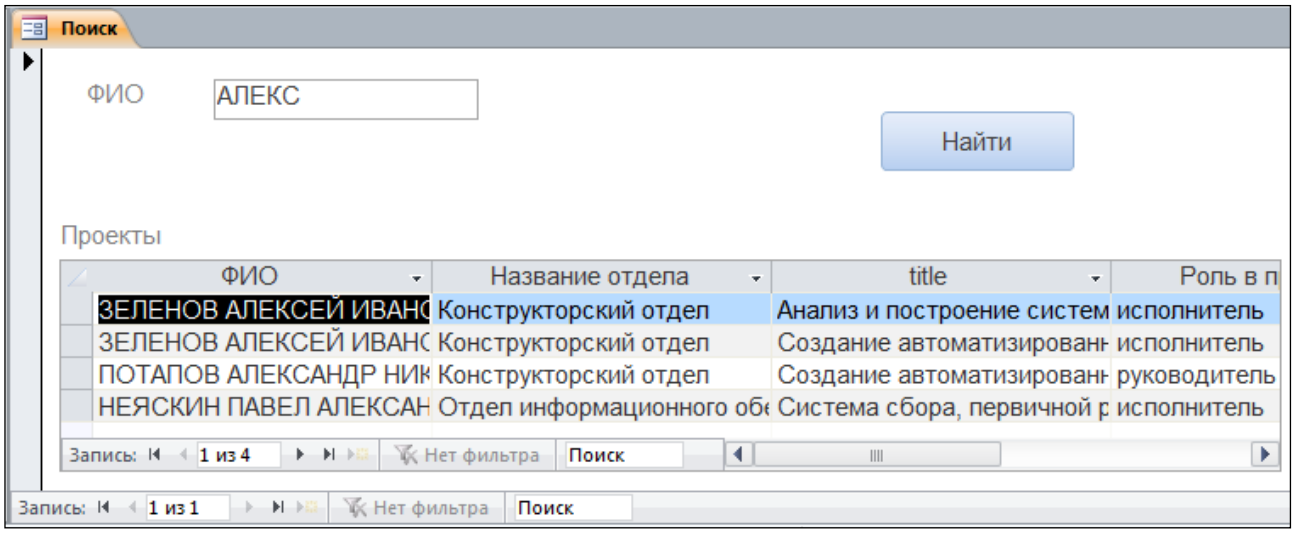


Рис. 41 Пример работы поисковой формы

Создадим второе поле «Отдел», как мы делали это ранее. В свойствах поля зададим имя `sotdel`, по которому мы будем к нему обращаться через Visual Basic. Поле `sfio` мы оставили без изменений, а поле «Отдел» нам следует преобразовать в «Поле со списком» путём нажатия правой кнопки "мыши" на поле

и выбора пункта выпадающего меню «Преобразовать в -> Поле со списком». В свойствах поля во вкладке «Данные» указываем «Источник строк» (Рис. 42):

```
SELECT Name FROM Depart;
```

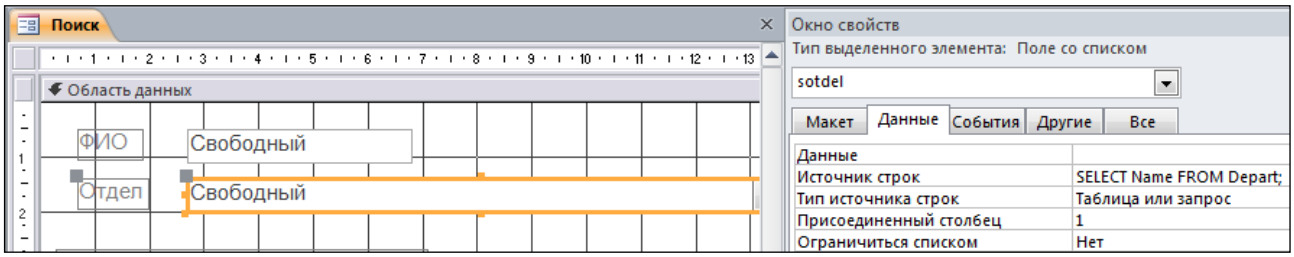


Рис. 42 Создание свободного поля со списком и указание источника данных

Переходим в режим формы и видим, что наше новое поле позволяет выбрать один из уже существующих отделов. Осталось связать новое поле с кнопкой «Найти» и проверить работоспособность.

Для этого нам нужно вернуться в редактирование программы кнопки «Найти». Как и ранее, мы описываем проверку, является ли поле пустым, и добавляем это условие в строку условий `condstr`:

```
If (Not IsNull(sotdel.Value) And(Trim(sotdel.Value)<>"") Then
  S = "(did = " & sotdel.Value & "') "
  If condstr <> "" Then
     condstr = condstr & " AND " & S
  Else
     condstr = S
  End If
End If
```

Пример работы поисковой формы приведен на Рис. 43.

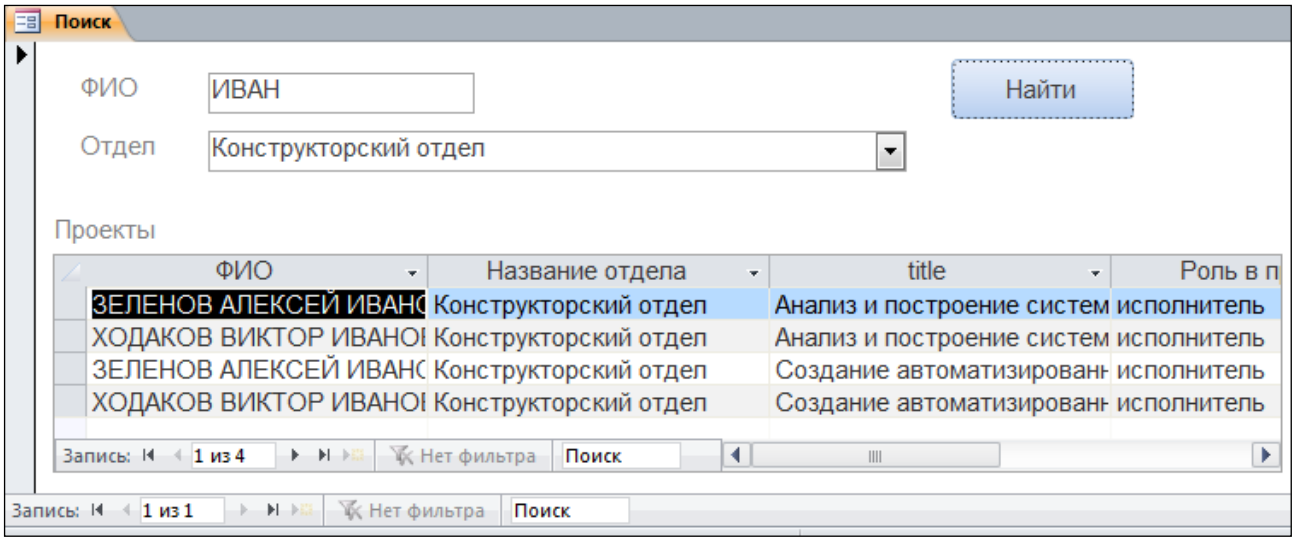


Рис. 43 Пример работы поисковой формы с двумя полями

Полный текст подпрограммы приведен на Рис. 43.

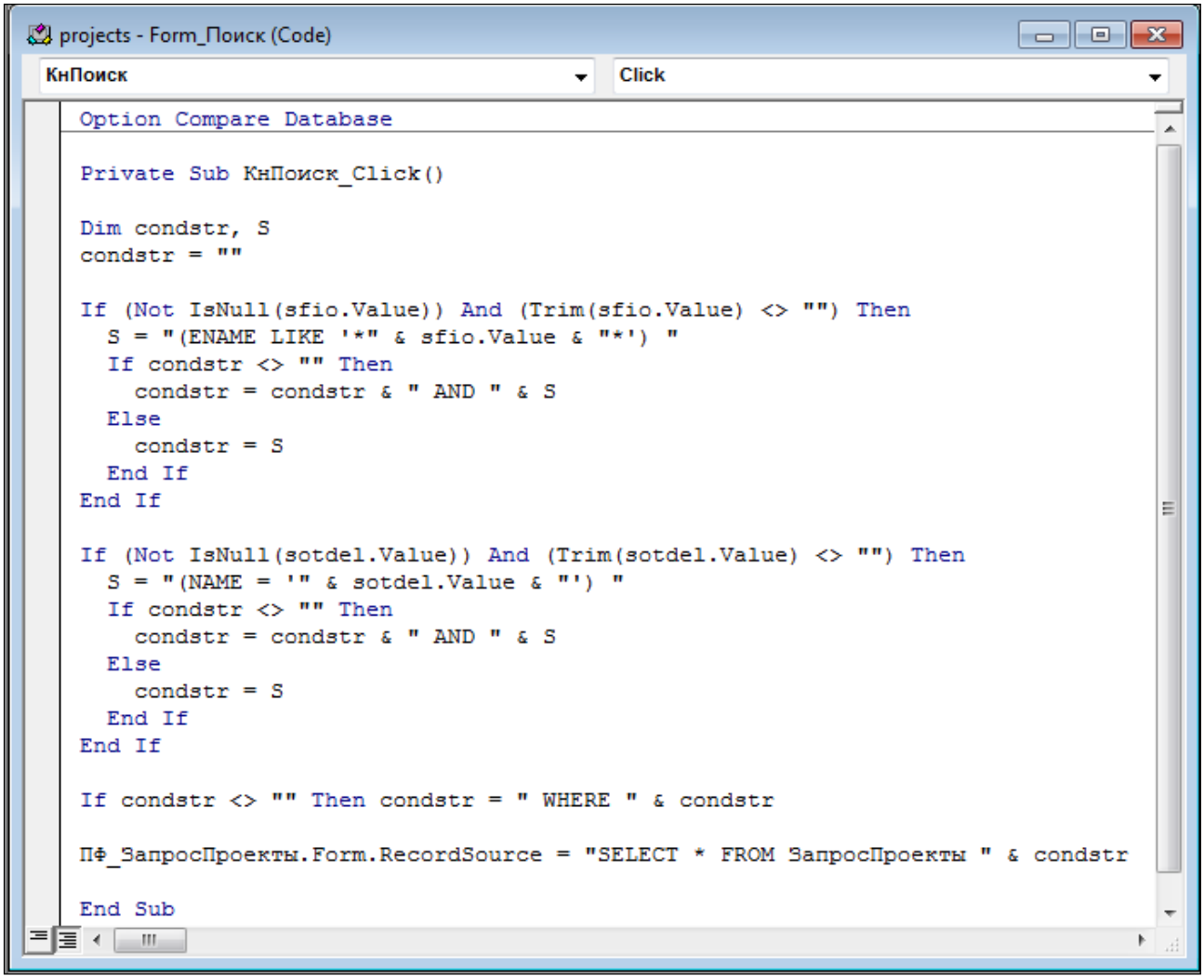


Рис. 44 Текст подпрограммы

Теперь мы создадим кнопочную форму и сделаем её основной для нашего приложения. Для этого перейдём на закладку "Создание" и в разделе "Формы" запустим конструктор форм. Далее из панели инструментов выберем пиктограмму "Кнопка" (Рис. 45) и расположим её в левом верхнем углу окна формы.

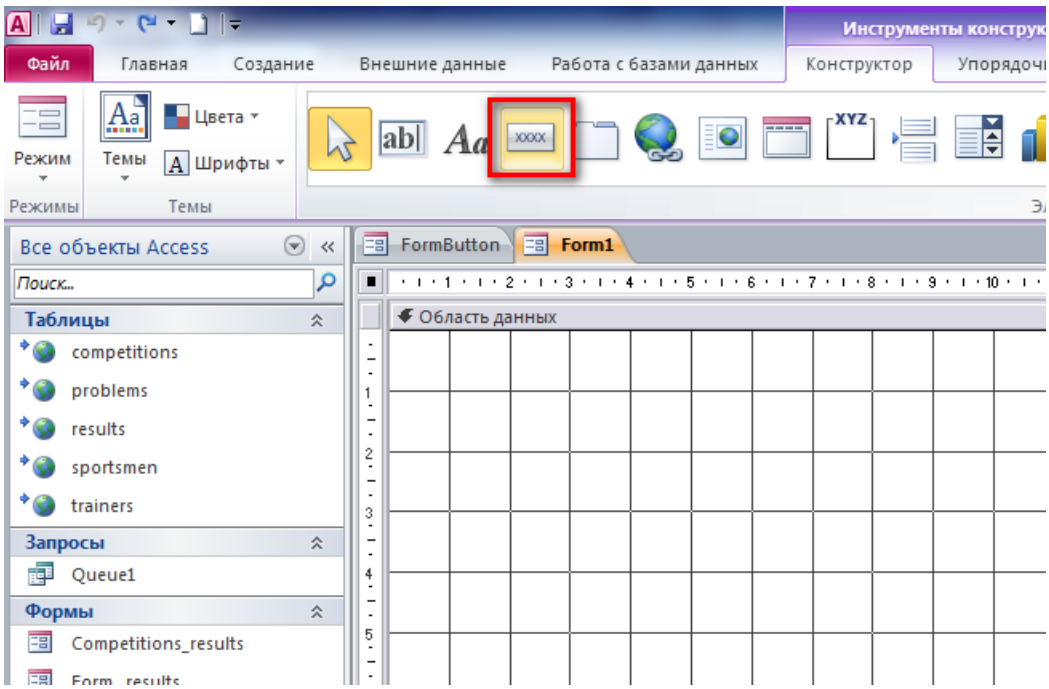


Рис. 45 Создание кнопки на форме

После этого запустится мастер создания кнопок, и мы выберем для этой кнопки категорию "Работа с формой" и действие "Открыть форму" (Рис. 46).

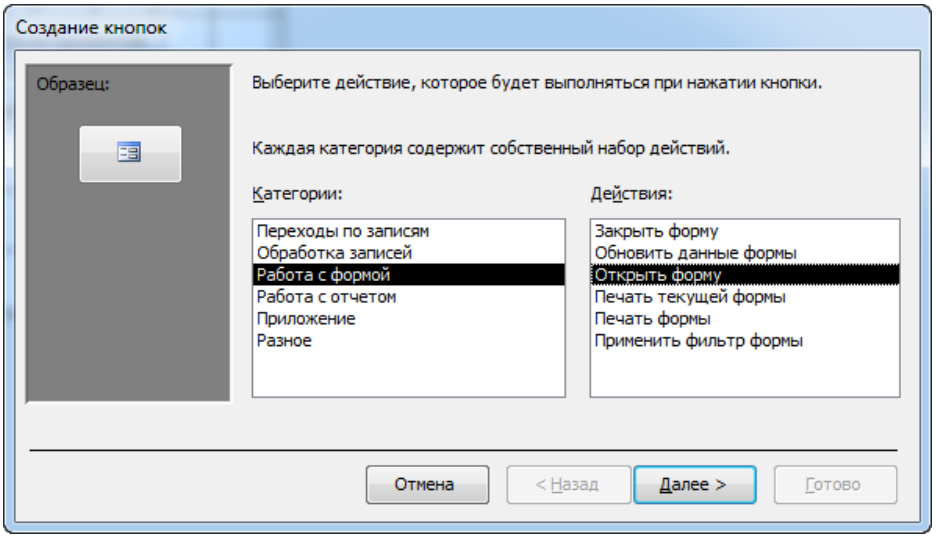


Рис. 46 Выбор действия при нажатии кнопки

В следующих окнах мы выберем форму "Отделы", которая будет открываться при нажатии данной кнопки, установим флаг "Открыть форму и показать все записи" и введём текст "Отделы", который будет располагаться на этой кнопке (Рис. 47).

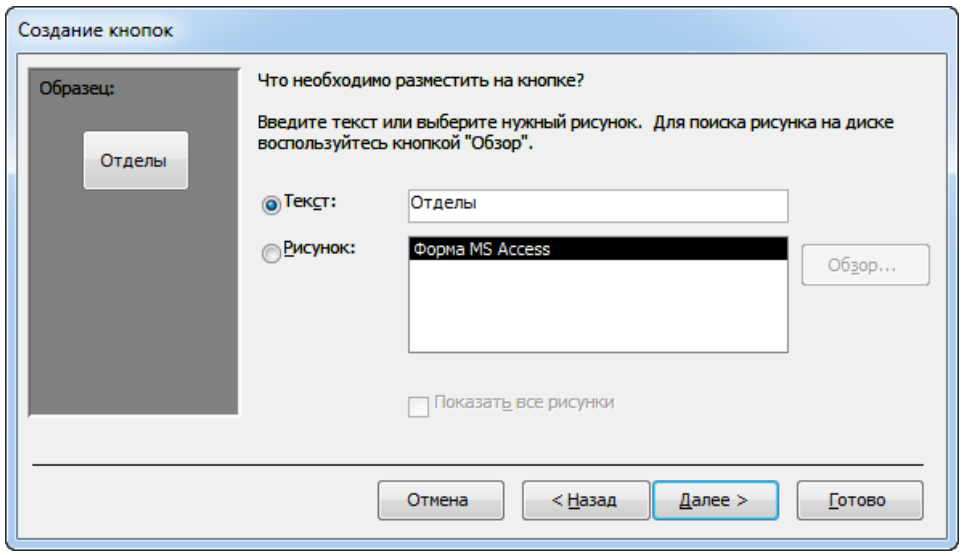


Рис. 47 Выбор надписи на кнопке

В последнем окне этого мастера переименуем кнопку (дадим ей имя КнОтделы, чтобы можно было по названию понять, к чему относится эта кнопка), и кнопка создана. При нажатии на эту кнопку откроется форма "Отделы", которую мы создали ранее.

Аналогичным образом создаются кнопки для просмотра отчётов. В качестве категории и действия при этом надо выбрать "Работа с отчётом" – "Просмотр отчёта" (Рис. 48).

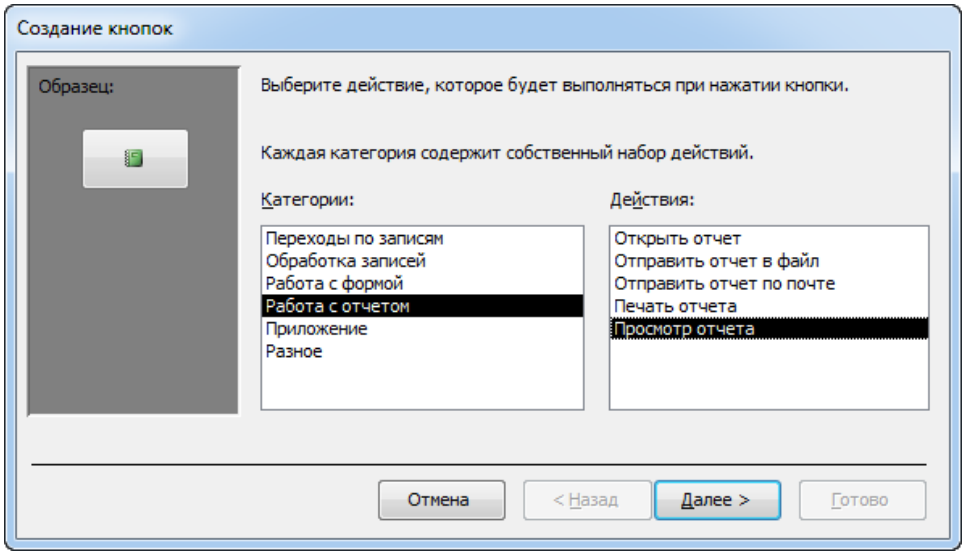


Рис. 48 Создание кнопки для просмотра отчётов

В следующих окнах мастера из списка выбирается один из существующих отчётов, вводится надпись на кнопки и имя кнопки.

Для того чтобы поместить на форму кнопку, связанную с запросом, нужно в качестве категории выбрать "Разное", а в качестве действия – "Выполнить запрос". Остальные действия выполняются аналогично. Создадим на этой форме кнопки для запросов "Руководители проектов" и "Текущие проекты" (Рис. 49). Сохраним полученную форму под именем "Главная форма".

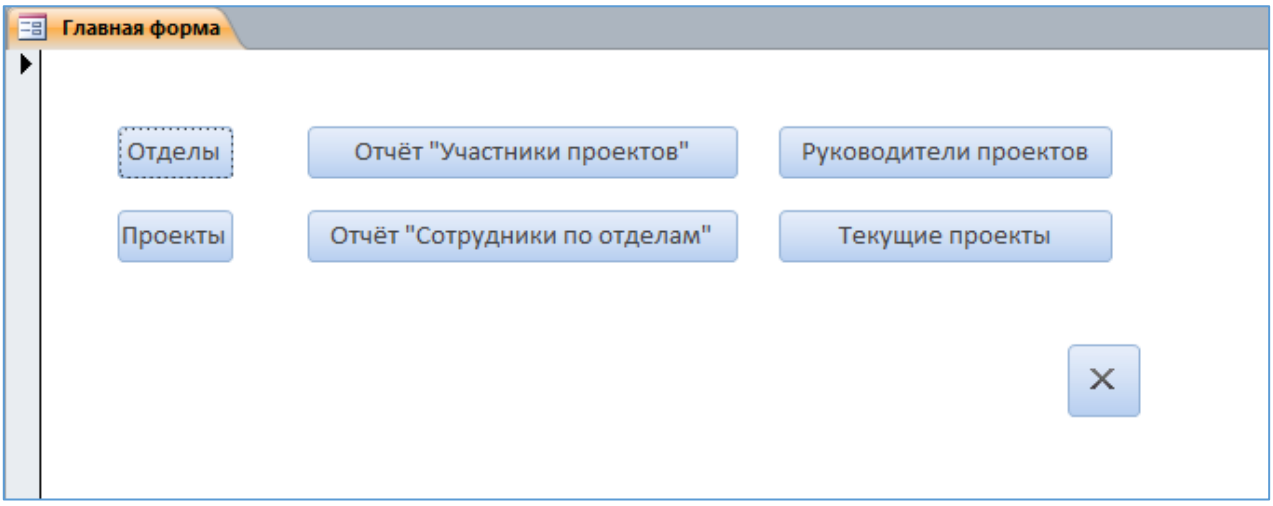


Рис. 49 Общий вид кнопочной формы

На данной форме мы ещё добавили кнопку "Выход" (в нижней правой части формы, Рис. 49). Для неё мы выбрали категорию "Приложение" и действие "Выйти из приложения", и на кнопке мы расположили пиктограмму "Выход" из предлагаемого системой списка пиктограмм. При нажатии этой кнопки работа приложения Access завершается.

Далее сделаем так, чтобы эта форма открывалась автоматически при запуске базы данных. В основном меню выберем пункт "Файл" –> "Параметры" –> "Текущая база данных". В окне этого меню (Рис. 50) введём заголовок приложения ("Проектная организация") и выберем главную форму в качестве основной.

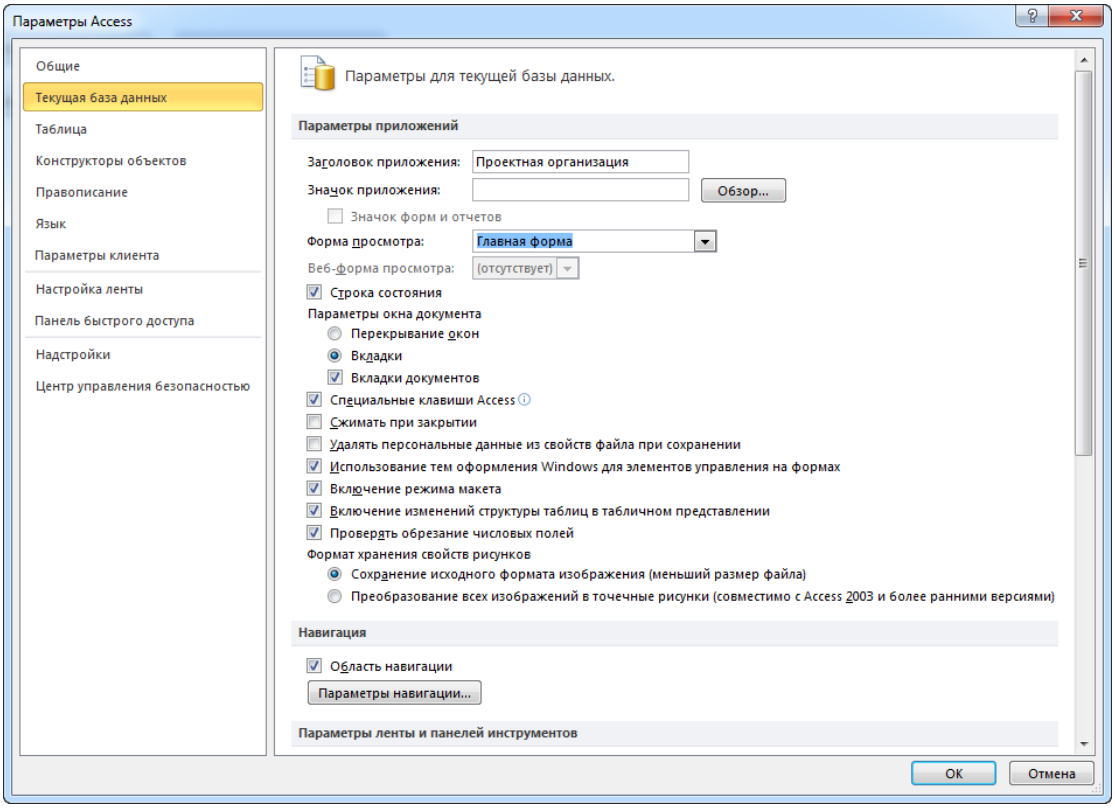


Рис. 50 Выбор главной формы в качестве основной при запуске

При необходимости закрыть созданное приложение от изменений обычными пользователями можно ещё убрать флаги меню и панелей инструментов: тогда они не будут выводиться на экран при запуске БД. Если же вам потребуется что-нибудь изменить в приложении или базе данных, надо запустить базу данных комбинацией клавиш Shift+Enter.

Итак, созданная вами поисковая форма позволяет задавать условия поиска и выводит на экран данные, соответствующие этим условиям. А кнопочная форма открывается автоматически при запуске приложения и позволяет работать с ранее созданными формами и отчетами.

После создания кнопочной формы ваше приложение готово к работе.

Библиографический список

1. Энсор Д., Стивенсон Й. Oracle. Проектирование баз данных: Пер. с англ. – К.: Издательская группа BHV, 1999. – 560 с.
2. Технологии баз данных: SQL, T-SQL, PL/SQL, реляционные БД: Внедрение SQL-операторов в прикладные программы. – http://datasql.ru/basesql/18.htm
3. Кузин А. В., Демин В. М. Разработка баз данных в системе Microsoft Access. – Издательство: "Форум", 2007. – 223 с.